This file contains the following:

Supplementary Figures

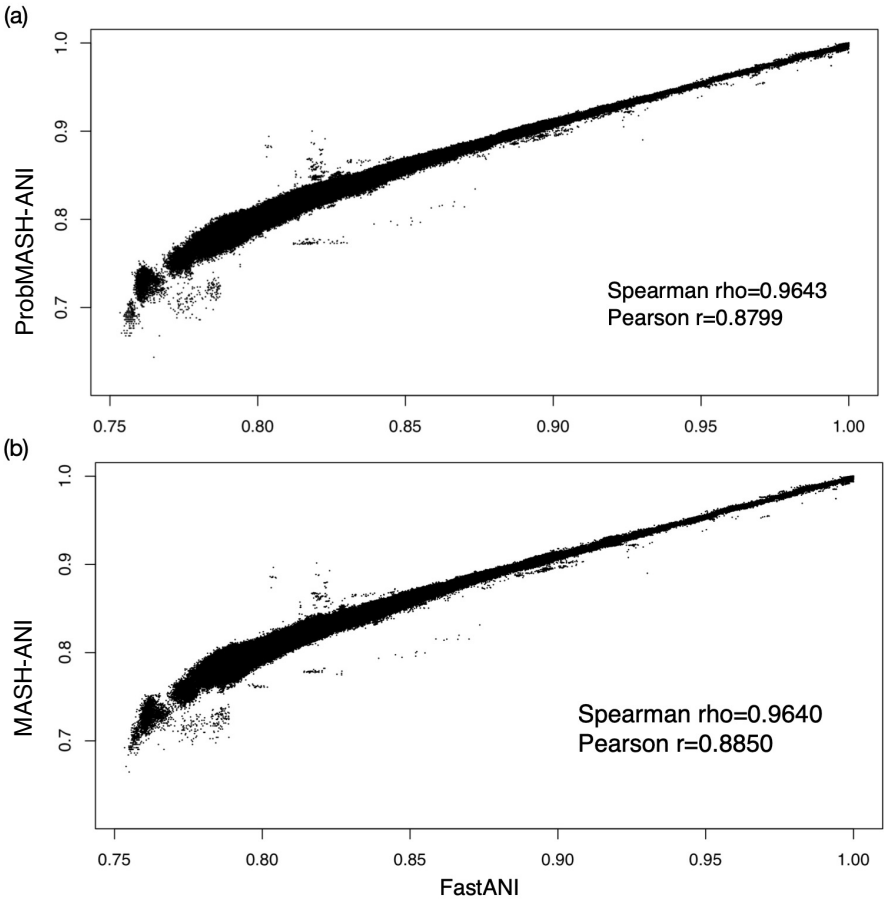

Figure S1. Relationship between FastANI and ProbMASH-ANI (a) and FastANI and MASH-ANI (b) for closely related genomes. Spearman and Pearson correlations were significant at $p<0.001$ in both (a) and (b). ProbMASH-ANI is transformed from ProbMinHash distance according to the MASH distance equation (See Materials and Methods, ProbMinHash section). Mean absolute error (MAE) were 0.023 and 0.021 respectively for (a) and (b) for data > 80% ANI, slightly larger than FastANI (MAE=0.014) for the same comparisons; see Figure S13.

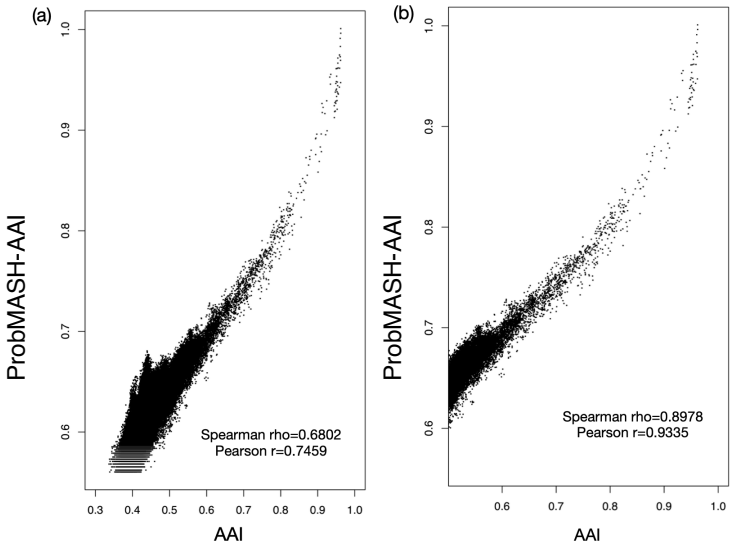

Figure S2. Relationship between AAI (entire proteome) and ProbMASH distance for (a) all AAI values and (b) AAI values between 0.52 and 0.95 among 2000 genomes randomly selected from the GTDB v207 database. Both Spearman and Pearson correlation significance test showed $p<0.01$.

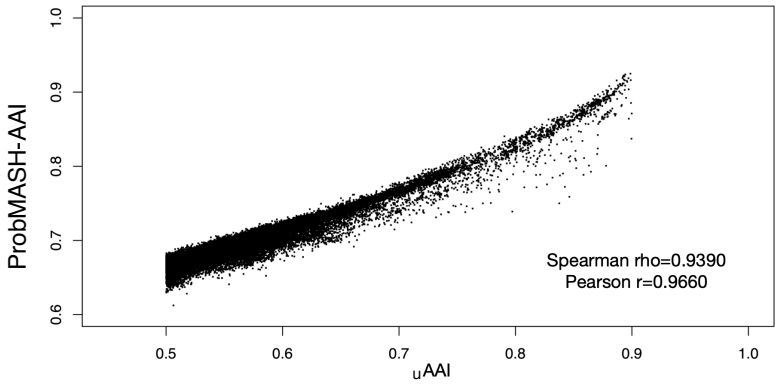

Figure S3. Relationship between AAI (universal gene set, or uAAI) and probMASH distances for AAI values between 0.5 and 0.9 among 2000 genomes randomly selected from the GTDB v207 database. Both Spearman and Pearson correlations were significant at $p<0.01$.

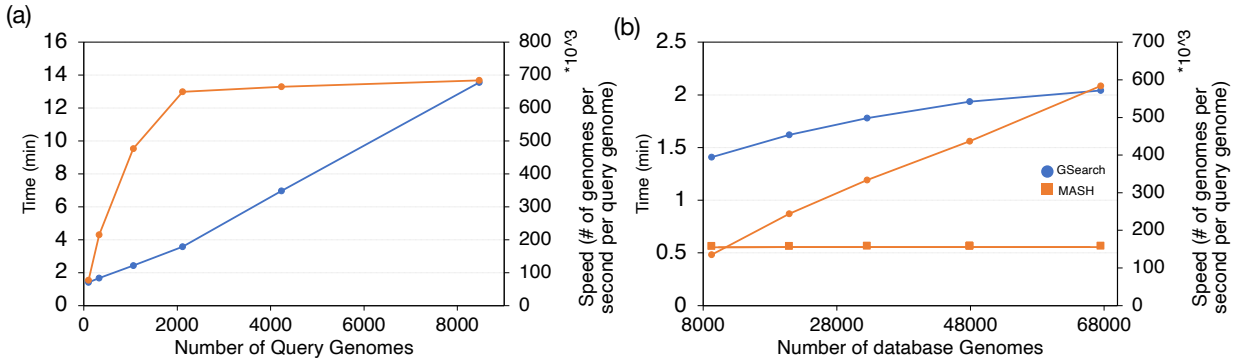

Figure S4. Performance of GSearch search step. (a) Search time (blue) and speed (orange) for an increasing number of query genomes against a fixed database of 67503 genomes and, (b) search time (blue) and speed (orange) for an increasing number of database genomes and a fixed number of 1059 query genomes. Search speed is defined as the number of database genomes searched, on average, per query genome per second. The orange line with squared data points represents Mash, whose speed was constant for any database size for 1059 query genomes, for comparison.

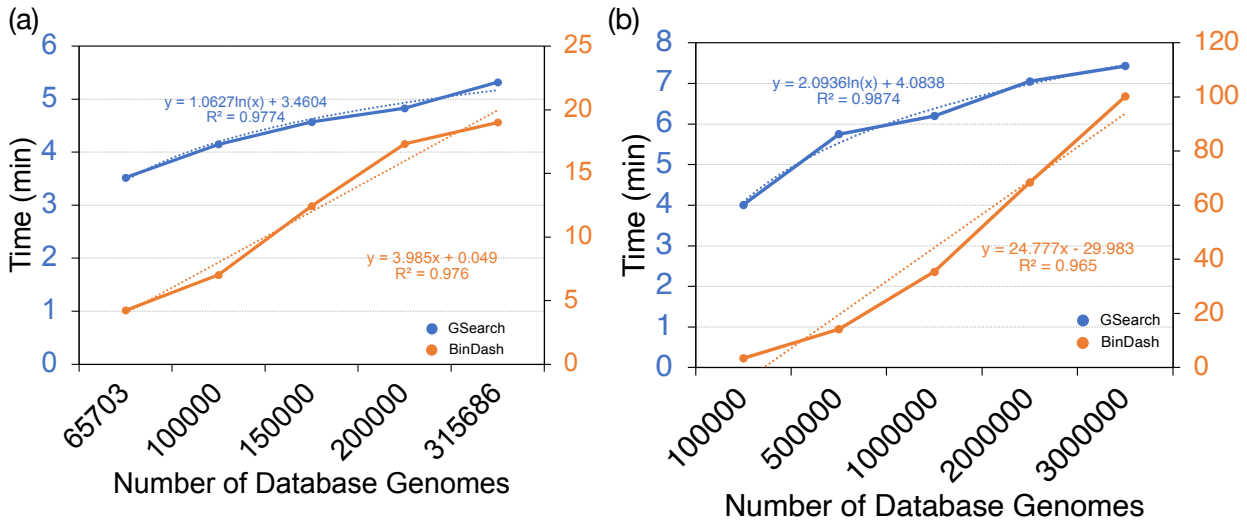

Figure S5. GSearch request speed comparison with BinDash software for (a) all bacterial genomes (~318K, 2TB) and (b) all viral genomes (~3 million) as database. A total of 8,466 and 10,000 query genomes were used, respectively, for this analysis. Note that GSearch and BinDash are plotted on different axes, primary y and secondary y axis, respectively. A total of 24 threads were used. Sketch size m=10^5 was used to achieve similar accuracy with ProbMinHash.

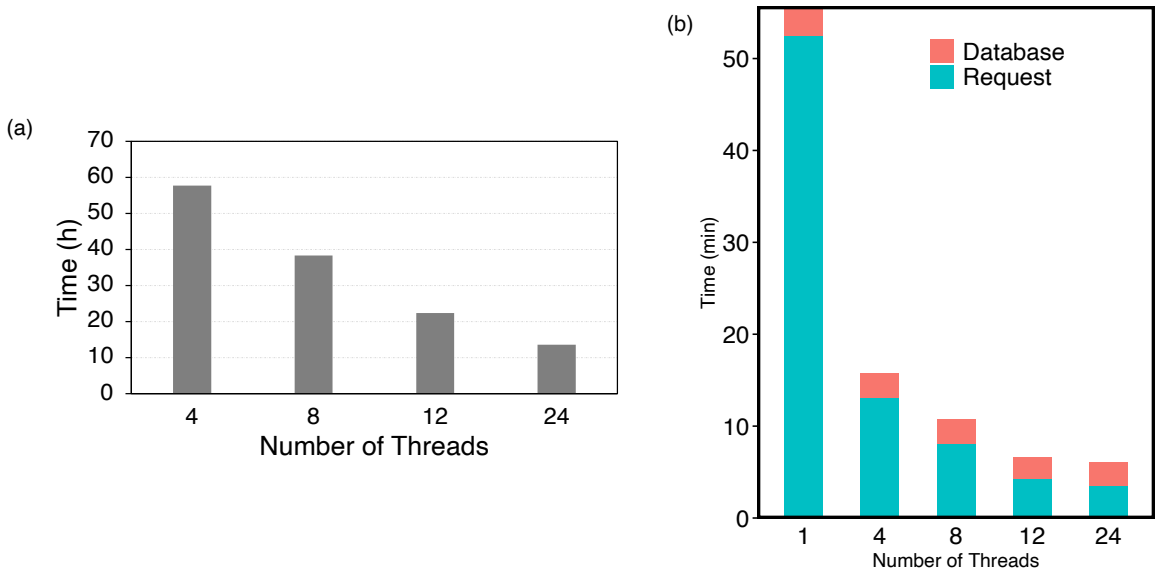

Figure S6. Database build time for the IMGVR phage species database (935,122) at the amino acid level (a), and total search (request) time for 10,000 query phages (b) against this database. Database size was about 15.8 GB, and thus loading the database takes a substantial fraction of the total searching time.

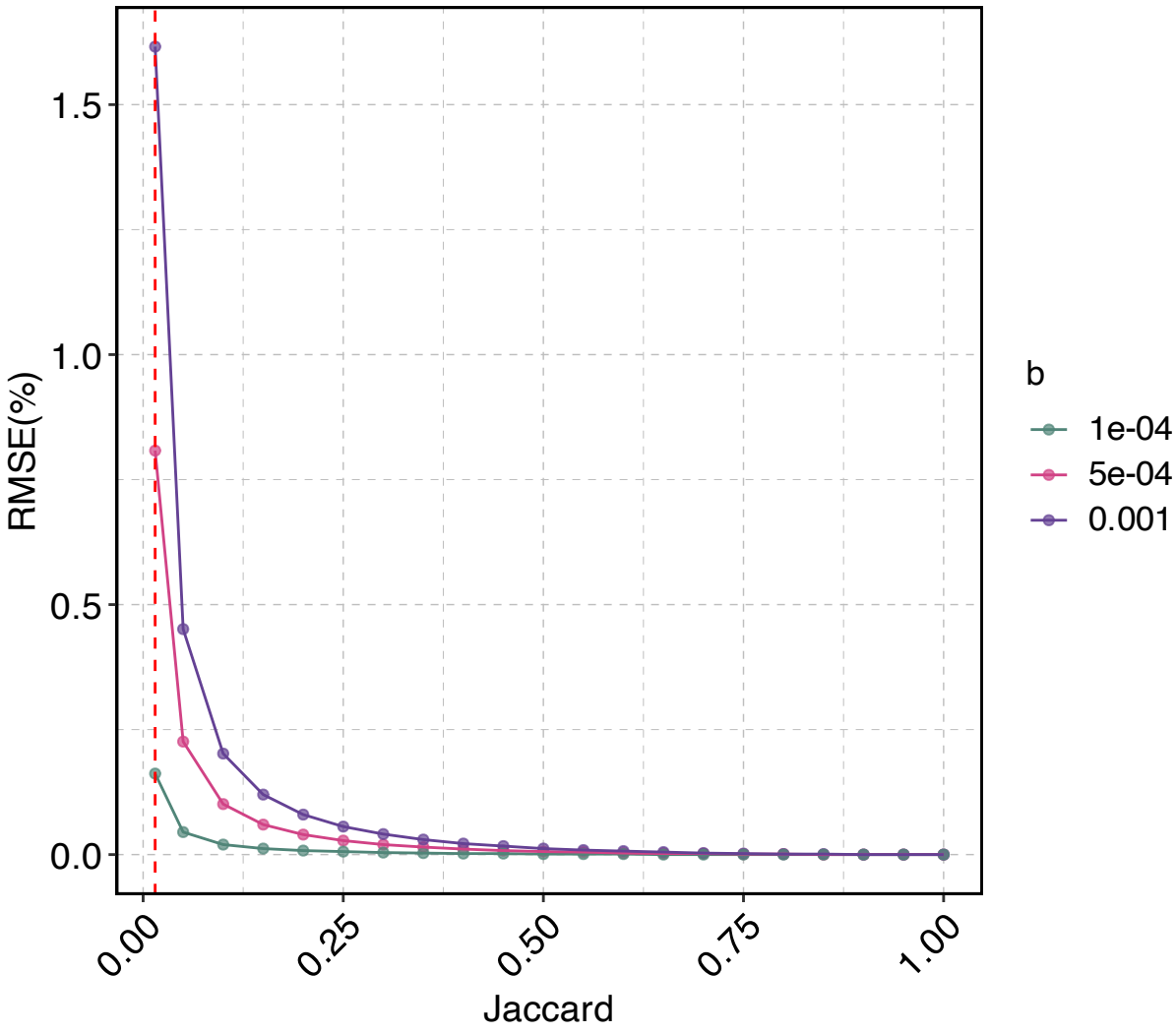

Figure S7. Rooted Mean Squared Error (RMSE) of SetSektch LSH algorithm for estimation of Jaccard index with respect to true Jaccard index for different values of parameter b. Note that changing b requires changing m in SetSektch. Three b values were used, 0.001, 0.0005, 0.0001 corresponding to m=4096, 6144 and 8192 respectively. The red dashed line indicates J=0.015 with RMSE 0.81%, corresponding to ANI 77.99%, with variation of ANI from 77.94% to 78.04% for the case that b=0.0005 and m=6144.

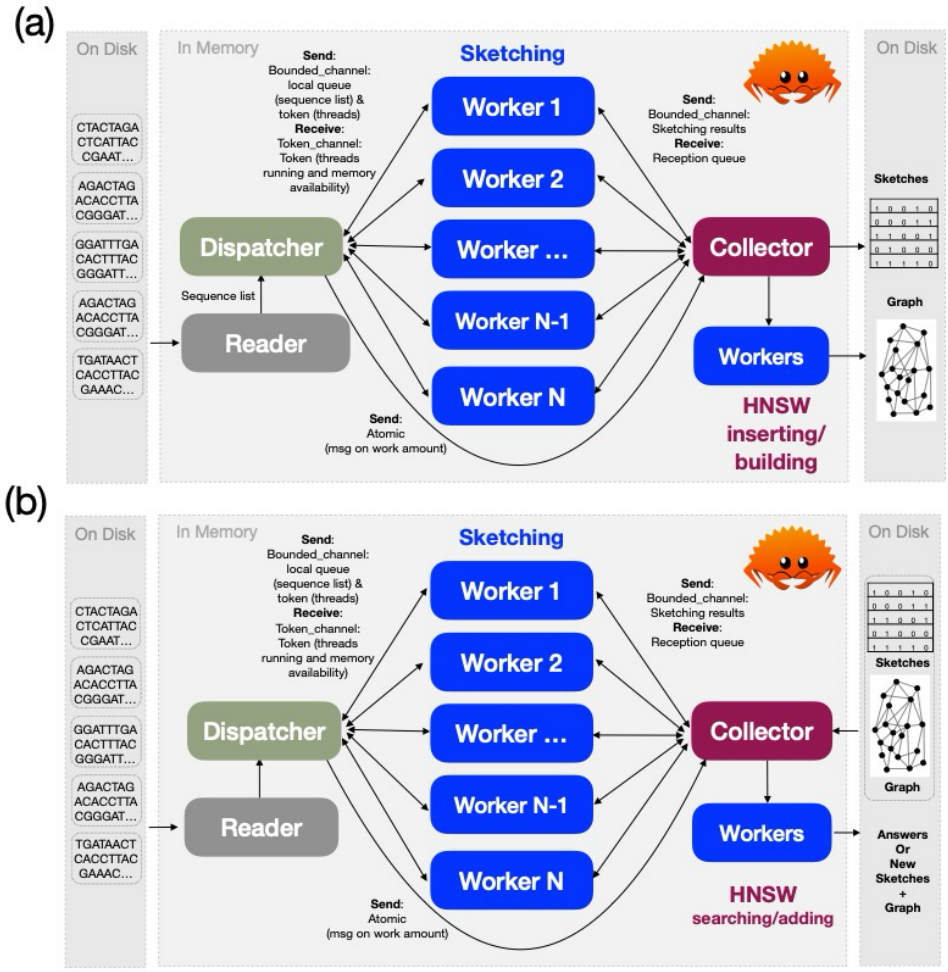

Figure S8. Parallel and concurrent computing model of GSearch tohnsw (a) and request or add (b) modules. Standard dispatcher and worker model was used with Reader and Collector to handle file input, collecting results from sketching workers, and initializing HNSW graph building/searching workers. Tohnsw/add module will dump sketching results and graph to disk while searching module will load prebuilt sketching and graph files and then dump answers for the query genomes to disk. Note that the number of Workers for HNSW building/searching/adding is to use all available computing cores while the number of Workers for sketching can be controlled to balance memory consumption and sketch speed. The concurrent model was based on Rayon library, and communication among threads was based on Crossbeam library.

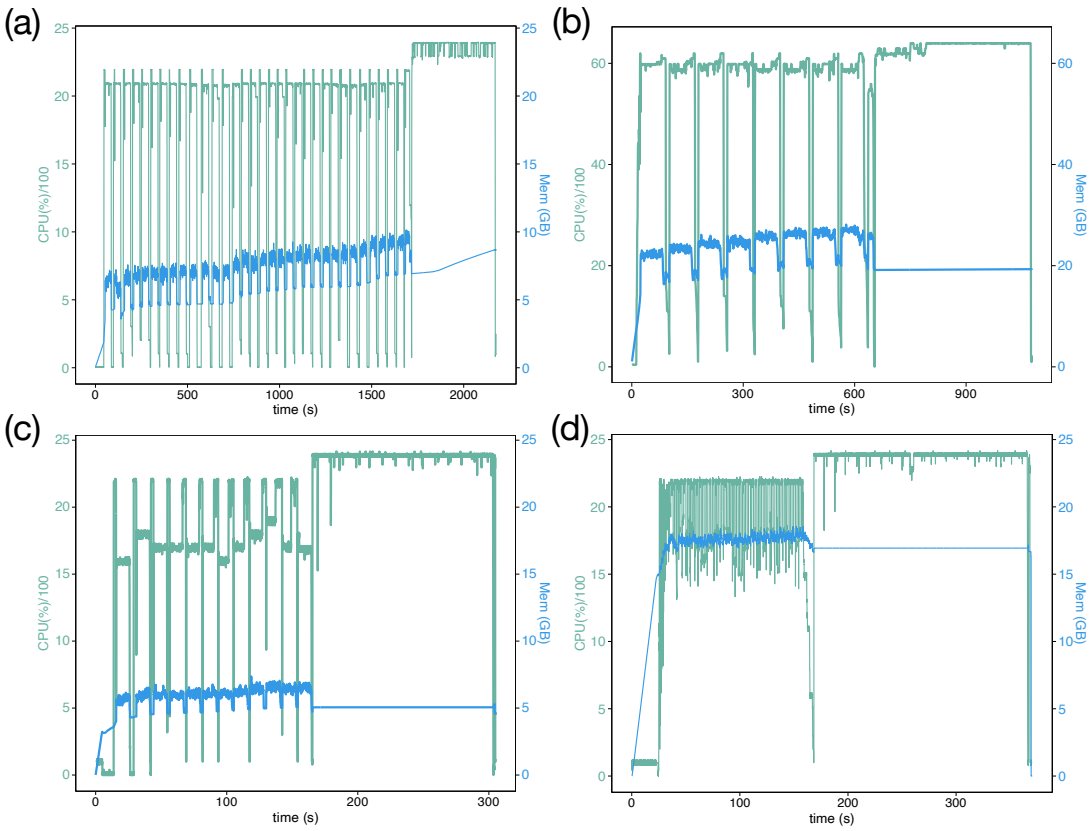

Figure S9. Parallel efficiency and memory consumption of GSearch tohnsw and request modules (ProbMinHash) at the nucleotide level. (a) tohnsw build module real-time CPU usage (left axis) and memory consumption (right axis) for the GTDB v207 (65,703 genomes) database using 24-thread node. (b) The same with (a) but on a 64-thread node. (c) Real-time CPU usage and memory consumption for searching 8,466 genomes against the GTDB prebuilt database in (a) on a 24-thread node. (d) Real-time CPU usage and memory consumption for searching 8,466 genomes against entire NCBI/Ref_Seq genomes (~318K). The two phases in all figures are sketching and graph building/searching, respectively.

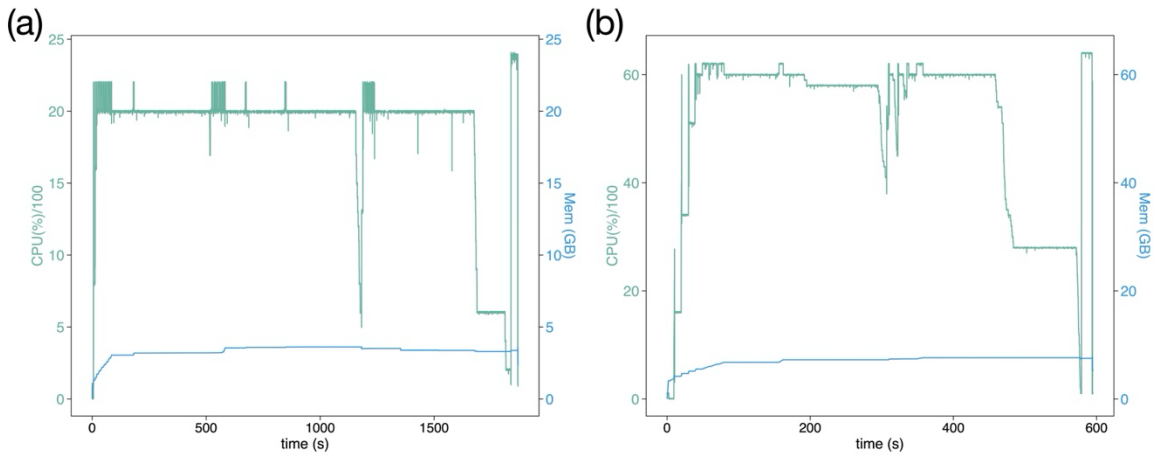

Figure S10. Parallel efficiency and memory consumption of GSearch request modules (SetSketch) at nucleotide level. (a) 8,466 query genomes against the GTDB v207 database (65,703 genomes) using 24 threads. (b) 8,466 query genomes against NCBI's RefSeq database (~318K genomes) using 64 threads.

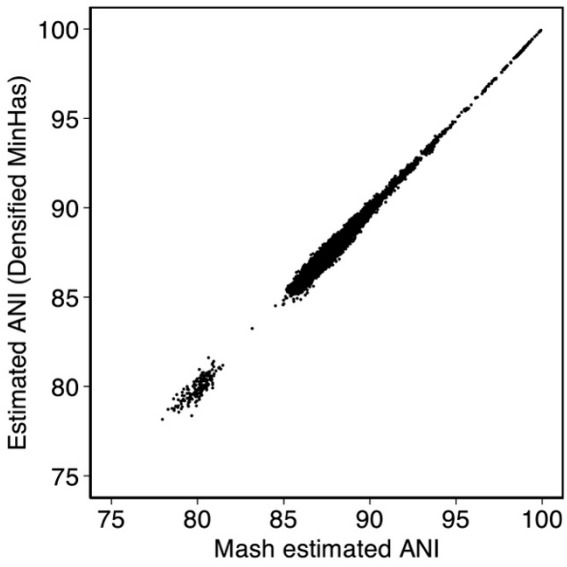

Figure S11. Correlation between ANI estimates from our implementation of Densified MinHash (optimal densification) and Mash bottom-m implementation. A sketch size m=12,000 was used for improved accuracy in densified MinHash while 15,000 was

used in Mash. Our implementation, as also reported in the BinDash paper, is at least 10 times faster than Mash.

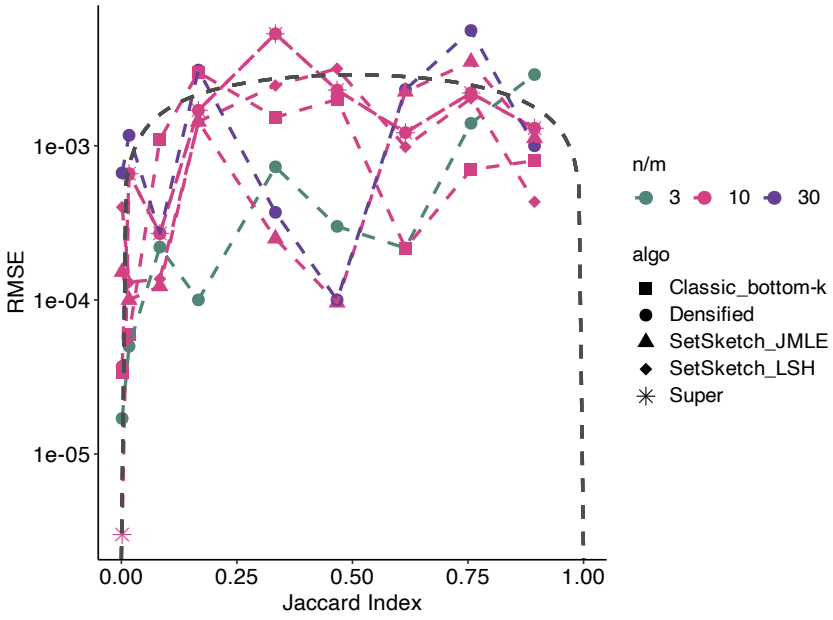

Figure S12. Root Mean Squared Error (RMSE) of Densified MinHash and Classic MinHash theoretical RMSE ($RMSE = \frac{\sqrt{J*(1-J)}}{\sqrt{m}}$) for Jaccard similarity, where m is the sketch size. We use set size (number of elements in the set) n=300,000 for all cases, m is 10^5 for classic MinHash.

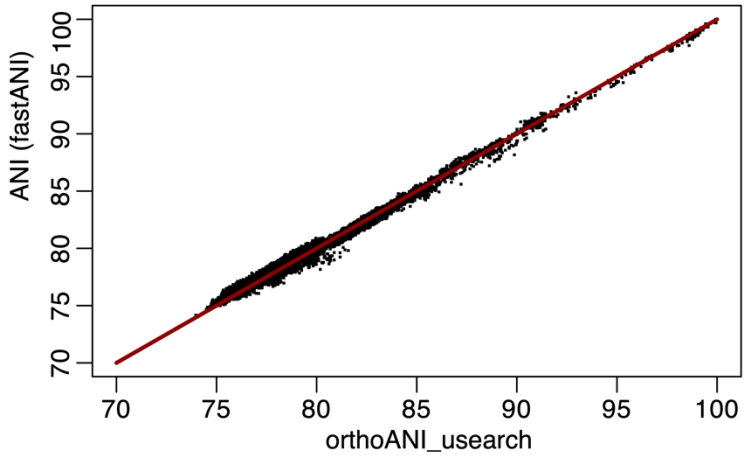

Figure S13. FastANI (y-axis) versus orthoANI (x-axis). Results are based on a testing dataset of 3000 genomes randomly subsampled from the large collection pf genome used in Figure S1. MAE is 1.4%.

Supplementary Tables

Table S1. Comparisons of MinHash-like and HyperLogLog algorithms. In bold face are results obtained as part of this study (via gsearch --algo prob/super/hll option); remaining results shown are from the literature. J_p is closer to real Jaccard index than J_w despite both algorithms considering k-mer weights.

	Weighted element	Set Size bias	Speed (More is better)	Space (Less is better)	Mergeability[10]	Approximated Jaccard-like index
Classic MinHash[1]	✗	✓	★★★★	☆☆☆☆	✓	J
Densifed MinHash (with OD[2,11])	✗	✓	★★★★★★	☆☆☆	✗	J
SuperMinHash[3]	✗	✓	★★★	☆☆☆	✓	**J**
ProbMinHash (default)[4]	✓	✗	★★★★★	☆☆☆	✓	**J_p**
BagMinhash[5]	✓	✓	★★	☆☆☆☆☆	✓	J_w
DartMinHash[6]	✓	✓	★★★★	☆☆☆☆☆	✓	J_w
HyperLogLog[7]	✗	✗	★★★	☆☆	✓	J
HyperLogLogLog[8]	✗	✗	★★	☆	✓	J
SetSketch[9]	✗ (✓)[12]	✗	★★★	☆☆	✓	**J**
UltraLogLog	✗	✗	★★	☆	✓	J

[1]Broder (1997), [2]Shrivastava (2017), [3]Ertl (2017), [4]Ertl (2020), [5]Ertl (2018), [6]Christiani (2020), [7]Flajolet et al. (2007), [8]Karppa and Pagh (2022), [9]Ertl (2021)
[10] which means that adding an element or taking the union of multiple subsets can be performed in sketch space, important feature for large scale application when new data need to be added to existing sketches or distributed environment.
[11]OD indicates optimal densification. Since it is not mergeable, it is difficult to be applied in distributed environment.
[12]SetSketch can be applied in a weighted fashion but not in this paper

Table S2. Theoretical comparison of estimator variance (relative mean standard error) of MinHash-like algorithms for Jaccard index, where m is the number of registers (sketch size). The same m was used for all tools expect for SetSketch, which requires a smaller m to have similar accuracy. J or J_p represents Jaccard index or J_p, respectively.

	RMSE
Classic MinHash	Sqrt(J*(1-J)/m)
ProbMinHash	sqrt(J_p*(1-J_p)/m)
SuperMinHash[1]	$sqrt(\frac{J(1-J)}{m}(1-\frac{\sum_{l=1}^{m-1} l^u((l+1)^u+(l-1)^u)-2l^u}{(m-1)^u m^u(u-1)}))$
SetSketch[2]	[sqrt(1/m), sqrt(1.08/m)]
Dashing (original&Improved)[3]	sqrt(1.079/m) + union/intersection error > sqrt(1.079/m)
Dashing (MLE)[4]	sqrt(1.074/m)
Densifed MinHash[5]	sqrt(1/m)
τ-GRA (τ=0.8898)[6]	sqrt(1.075/m)

[1]The second long term in the variance is always smaller than 1, thus SuperMinHash RMSE is always smaller than classic MinHash, the smallest MinHash RMSE known until today; [2]As b is close to 1, RMSE is close to lower bound; [3]Both the original and improved estimator need correction at small cardinality. Dashing2 has the same variance with SetSketch because it reimplemented SetSketch; [4]MLE is Maximum Likelihood Estimator, theoretically optimal for HyperLogLog sketches but much slower; [5]One Permutation MinHash with Optimal Densification, variance goes to 0 much faster than classic MinHash/SuperMinHash/ProbMinHash as m increases, see Shrivastava (2017). Note that the variance is larger than classic MinHash or ProbMinHash for any J because J(1-J) is always smaller than 0.25; [6]Generalized Remaining Area with τ=0.8898 has the lowest variance. Detailed analysis of RSME for Tau-GRA can be found in Pettie and Wang (2022)

Table S3. GSearch's performance on major CPU platforms for searching 8,466 query genomes against the RefSeq reference database (~318K genomes). ProbMinHash was used as the genomic distance algorithm for this benchmark, with 50 neighbors requested.

CPU	Number of threads	Clock speed (GHz)	**Request time for nt (min)**	Gene Prediction-FGSrs (min)[c]	**Request time for proteome (min)**	hmmsearch time (min)[d]	**Request time for USCG (min)[d]**
Intel (R) Xeon (R) Gold 6226[a]	24	2.70	**9.329**	3.348	**6.234**	3.524	**0.445**
Intel (R) Core i9-8950HK[b]	16	2.30	**13.654**	6.764	**7.91**	6.214	**0.613**
AMD EPYC 7513a[a]	32(24 used)	2.60	**10.245**	4.452	**6.021**	3.345	**0.402**
Apple M1 Pro[b]	10	3.22	**9.669**	4.31	**6.166**	4.603	**0.568**

[a] RHEL v7.9, Linux v3.10.0-1160, all threads used. 512G RAM available.
[b]MacOS v12.3, Darwin 21.4.0 on MacBook Pro laptops, all threads used. Database was split into two pieces to have smaller memory requirement on those 16GB RAM laptops since database size was ~15GB for entire NCBI/RefSeq.

[c]Parallel package was used to run multiprocess at the same time. FGSrs stands for FragGeneScanRs (comparisons with Prodigal can be found in Table S11). Note that in practice only those genomes failing in the request step at the nucleotide (nt) level (best match found was less than 78% ANI) will be used in this amino-acid step.

[d]Only 1000 genomes were used for testing hmmsearch and USCG request in GSearch because this step was for assessing a few genomes that were novel at the order level or above. Parallel Packages was used to run multiple processes of hmmsearch, one thread per process for hmmsearch.

Table S4. GSearch 3-step classification pipeline accuracy using GTDB-tk as reference for 1,000 test genomes of various degrees of novelty relative to the database genomes.

	GSearch	GTDB
Species level (>95% ANI)	100% (699/699)	100% (reference)
Genus to family level or above (<62% AAI)	87.1%(266/301)	100% (reference)

Table S5. GSearch search (request) performance on major CPU platforms using the GTDB v207 database for graph building and 1000 genome queries. Default ProbMinHash option was used.

CPU	Number of threads	Clock speed (GHz)	**Request time for nt (min)**	Gene Prediction-FGSrs (min)[c]	**Request time for proteome (min)**	hmmsearch time (min)[d]	**Request time for USCG (min)**
Intel (R) Xeon (R) Gold 6226[a]	24	2.70	**2.329**	1.348	**1.334**	0.524	**0.117**
Intel (R) Core i7-7770HQ[b]	8	2.80	**8.654**	6.764	**2.041**	1.534	**0.510**
AMD EPYC 7513a[a]	32(24 used)	2.60	**1.937**	1.120	**1.021**	0.345	**0.102**
Apple M1 Pro[b]	10	3.22	**2.369**	2.12	**0.866**	0.498	**0.168**

[a] RHEL v7.9, Linux v3.10.0-1160, all threads used.

[b]MacOS v12.3, Darwin 21.4.0, all threads used.

[c]Parallel package was used to run multiprocess at the same time. FGSrs stands for FragGeneScanRs. Note that in practice only those genomes failing at the Request step for nucleotide-level search (best match found is <78% ANI) will be used at this step.

[d]Only 100 genomes were used for testing hmmsearch because this step is for very novel (deep-branching) genomes at order level or above, which are not very common in real-world dataset. The Parallel Package was used to run multiple processes of hmmsearch, one thread per process for hmmsearch.

Table S6. GSearch performance and accuracy for the viral database using the BLAST-based AAI as the reference standard. MASH dist command was run using 24 threads for this analysis.

	GSearch (aa)	Recall (top 5)	MASH (aa)
1000 phage	4 min 24 s	98.32%	2.12 h
10,000 Phage	39 min 58 s	96.04%	17.4 h

Table S7. Benchmarking of hnswlib-rs library using the MNIST fashion (70,000) dataset.

	M[a]	ef_construct	recall	Build time (s)	Query time[b] (s)	Speed[c] (queries/s)
Run #1	32	200	0.9849	4.88	0.27	36569.63
Run #2	64	400	0.9959	9.09	0.46	21699.18
Run #3	128	400	0.9981	9.66	0.78	12740.01
Run #4	128	800	0.9984	18.00	0.86	11574.46
Run #5	128	1600	0.9986	34.53	0.93	10747.78
Run #6	256	1200	0.9995	30.00	1.61	6209.15
Run #7	256	1600	0.9996	39.18	1.67	5972.75
Run #8	256	2400	0.99999	56.33	1.72	5804.61

[a] ef_search=M, the width of search for querying equals to M

[b] Large M and ef_construct take more time for both build and query search. In the MNIST fashion dataset, Euclidean distance is pre-computed and not considered in this test. The distance used is the same for Table S8, Table S9 and table S10.

[c] Tests were run on a 24 threads Intel (R) Xeon (R) Gold 6226 CPU@2.70 Ghz. Speed is tested using 10,000 queries.

Table S8. Benchmarking of hnswlib-rs library using the SIFT1M (1,000,000) dataset.

	M	ef_construct	recall	Build time (min)	Query time (s)	Speed [a](queries/s)
Run #1	128	1600	0.9980	21.18	1.81	5528.93
Run #2	256	3200	0.9986	40.83	3.82	2618.20

[a] Tests were run on a 24 threads Intel (R) Xeon (R) Gold 6226 CPU@2.70 Ghz. Speed is tested using 10,000 queries.

Table S9. Benchmarking of hnswlib-rs library for scalability using the SIFT1M dataset.

	Number of threads	M	ef_construct	Build time (min)	Query time (s)	Speed[a] (queries/s)
Run #1	8	256	3200	87.97	7.90	1265.27
Run #2	12	256	3200	67.02	6.26	1596.35
Run #3	24	256	3200	40.83	3.82	2618.20

[a] Tests were run on an Intel (R) Xeon (R) Gold 6226 CPU@2.70 Ghz. For each run, the number of threads shown in column #2 were requested. Speed is tested using 10,000 queries.

Table S10. Comparison of hnswlib-rs with C++ hnswlib using the MNIST fashion (70,000) dataset. For this test, 60,000 genomes were used for building as database and 10,000 as query, requesting 50 best neighbors.

	Number of threads	M	ef_construct	Build time (s)	Query time (s)	Speed[a] (queries/s)
Run #1 (hnswlib-rs)	24	128	1600	34.53	0.93	11225
Run #2 (hnswlib-c++[1])	24	128	1600	31.02	1.05	10204
Run #3 (hnswlib-rs)	24	256	3200	92.83	2.32	4250
Run #4 (hnswlib-c++)	24	256	3200	51.83	1.85	5235

[1] https://github.com/nmslib/hnswlib

Table S11. The table shows benchmark results of FGSrs against truth (full version v0.0.1) and comparison with other tools. True positives were denoted as the number of base pair in a prediction on the correct strand, false positives the number of base pair in a prediction outside gene annotations or on the wrong strand, true negatives the number of bp outside

gene annotations that weren't in any prediction, and false negatives the number of bp inside gene annotations that weren't in any prediction. Prec denotes precision, Sens denotes sensitivity, Spec denotes specificity. Negative Predictive Value and Matthew's Correlation Coefficient are provided as percentages.

tool	TP	FP	TN	FN	prec	sens	F1 score	spec	NPV	MCC
FGS	2494195	231257	656879	283010	91.51	89.81	0.90	73.96	69.89	62.58
prodigal	2488137	117535	745673	313996	**95.49**	**88.79**	**0.92**	**86.38**	70.37	70.36
FGSrs	2494195	231257	656879	283010	**91.51**	**89.81**	**0.90**	**73.96**	69.89	62.58

Table S12. Effect of genome completeness on GSearch recall. Genomes of different degrees of completeness were obtained by randomly sampling the gene sequences from the predicted gene collection of the complete genome. *Hydrogenimonas urashimensis*, an H_2 dependent chemolithoautotrophic bacteria isolated from deep sea hydrothermal vent (Mino et al., 2021), was used as query. There is no identical genome to *H. urashimensis* in the GTDB database.

Completeness	Top Hits found/10	Top 5 hits found/5	Recall (top 10)	Recall (top 10)
0.95	10/10	5/5	100%	100%
0.90	10/10	5/5	100%	100%
0.85	9/10	5/5	90%	100%
0.80	8/10	5/5	80%	100%
0.75	8/10	5/5	80%	100%
0.70	8/10	5/5	80%	100%
0.65	8/10	5/5	80%	100%
0.60	8/10	5/5	80%	100%
0.55	8/10	5/5	80%	100%
0.50	7/10	5/5	70%	100%
0.45	5/10	5/5	50%	100%
0.40	4/10	4/5	40%	80%
0.35	4/10	4/5	40%	80%
0.30	4/10	4/5	40%	80%

0.20	2/10	2/5	20%	40%
0.10	1/10	2/5	10%	40%

Table S13. GSearch, Sourmash,Dashing and BinDash benchmark against blastn-ANI at nucleotide level. Query genomes is "OceanDNA-b42278.fa" from (Nishimura and Yoshizawa, 2022). Database genomes were all NCBI/RefSeq genomes. Mash, FastANI and Blastn-based ANI give the same top 10 for this query genomes. For each column, matches are ranked by the respective distance, taken directly from the software output. Top 10 found by blastn-ANI were about 80% to 97% ANI.

Blastn-ANI (top10), ground truth	GSearch (top10, ProbMinHash), recall 100%, 2s	Sourmash (top10), recall 90%, 13min	Dashing (top10), recall 90%, 26s	BinDash (top 10), recall 100%, 3.7s
GCA_902591925.1_genomic.fna	GCA_902591925.1_genomic.fna	GCA_902591925.1_genomic.fna	GCA_902591925.1_genomic.fna	GCA_902591925.1_genomic.fna
GCA_902617045.1_genomic.fna	GCA_902617045.1_genomic.fna	GCA_902617045.1_genomic.fna	GCA_902617045.1_genomic.fna	GCA_902617045.1_genomic.fna
GCA_902541175.1_genomic.fna	GCA_902547295.1_genomic.fna	GCA_902541175.1_genomic.fna	GCA_902612915.1_genomic.fna	GCA_902541175.1_genomic.fna
GCA_902612915.1_genomic.fna	GCA_902612915.1_genomic.fna	GCA_902612915.1_genomic.fna	GCA_902541175.1_genomic.fna	GCA_902612915.1_genomic.fna
GCA_004212975.1_genomic.fna	GCA_004212975.1_genomic.fna	GCA_902547295.1_genomic.fna	GCA_902547295.1_genomic.fna	GCA_902547295.1_genomic.fna
GCA_902547295.1_genomic.fna	GCA_902541175.1_genomic.fna	GCA_004212975.1_genomic.fna	GCA_004212975.1_genomic.fna	GCA_004212975.1_genomic.fna
GCA_902630885.1_genomic.fna	GCA_902586925.1_genomic.fna	GCA_902586925.1_genomic.fna	GCA_000252525.1_genomic.fna	GCA_902630885.1_genomic.fna
GCA_902586925.1_genomic.fna	GCA_902630885.1_genomic.fna	GCA_902631785.1_genomic.fna	GCA_902630885.1_genomic.fna	GCA_902586925.1_genomic.fna
GCA_902631785.1_genomic.fna	GCA_902631785.1_genomic.fna	GCA_000252525.1_genomic.fna	GCA_902631785.1_genomic.fna	GCA_902631785.1_genomic.fna
GCA_000252525.1_genomic.fna	GCA_000252525.1_genomic.fna	GCA_902582355.1_genomic.fna	GCA_902582355.1_genomic.fna	GCA_000252525.1_genomic.fna

Table S14. GSearch (ProbMinHash), Sourmash, Dashing and Sourmash benchmark against Blastn-ANI at nucleotide level. Query genome is "OceanDNA-b42278.fa" from (Nishimura and Yoshizawa, 2022). Database genomes are all NCBI/RefSeq genomes (318k) after removing the top 10 ground truth genomes found in **Table S13** above. For each column, matches are ranked by the respective distance provided in the output of each tool. Top 10 matches found by blastn-ANI showed between 75% to 80% to the query genome, corresponding to Jaccard index 0.009 to 0.015 respectively. Boldface denotes the genomes found by each method compared to the ground truth in column 1.

Blastn-ANI (top10), ground truth	GSearch (Prob) (top10), recall 50%, 2s	Sourmash (top10), recall 60%, 14min	Dashing (default Ertl-MLE estimator) (top10), recall 10%, 26s	BinDash recall (90%). 3.7s
GCA_902582355.1_genomic.fna (79.24%)	**GCA_002690725.1_genomic.fna**	**GCA_002690725.1_genomic.fna**	**GCA_902582355.1_genomic.fna**	**GCA_902582355.1_genomic.fna**
GCA_002690725.1_genomic.fna (78.79%)	**GCA_902560315.1_genomic.fna**	**GCA_902560315.1_genomic.fna**	GCA_003282945.1_genomic.fna	**GCA_902560315.1_genomic.fna**
GCA_902560315.1_genomic.fna (77.92%)	**GCA_902556105.1_genomic.fna**	**GCA_902556105.1_genomic.fna**	GCA_012735275.1_genomic.fna	**GCA_002690725.1_genomic.fna**
GCA_902517505.1_genomic.fna (77.57%)	GCA_902558095.1_genomic.fna	**GCA_902582355.1_genomic.fna**	GCF_000165465.1_genomic.fna	**GCA_902517505.1_genomic.fna**
GCA_002169625.2_genomic.fna (76.41%)	**GCA_902517505.1_genomic.fna**	GCA_902558095.1_genomic.fna	GCF_003111605.1_genomic.fna	**GCA_902583575.1_genomic.fna**
GCA_902556105.1_genomic.fna (76.37%)	GCF_017873235.1_genomic.fna	**GCA_003213495.1_genomic.fna**	GCA_017532025.1_genomic.fna	**GCA_902579825.1_genomic.fna**
GCA_902583575.1_genomic.fna (75.30%)	**GCA_002169625.2_genomic.fna**	GCA_002721465.1_genomic.fna	GCA_902592715.1_genomic.fna	**GCA_003213495.1_genomic.fna**
GCA_003213495.1_genomic.fna (75.24%)	GCA_002704625.1_genomic.fna	**GCA_902517505.1_genomic.fna**	GCA_014653355.1_genomic.fna	**GCA_902556105.1_genomic.fna**
GCA_902579825.1_genomic.fna (75.15%)	GCA_902559345.1_genomic.fna	GCA_902557965.1_genomic.fna	GCA_902626385.1_genomic.fna	GCA_902558095.1_genomic.fna
GCA_902563835.1_genomic.fna (75.00%)	GCA_902557965.1_genomic.fna	GCA_902559345.1_genomic.fna	GCA_016288795.1_genomic.fna	**GCA_902563835.1_genomic.fna**

Table S15. GSearch (SetSketch) and Dashing (Ertl's Joint MLE) benchmarking against blastn-ANI. The table is similar to **Table S14** above but using the SetSketch algorithm.

Blastn-ANI (top10), ground truth	GSearch with SetSketch HLL (top10), recall 60%, 2s	GSearch with SuperMinHash (top10), recall 60%, 2s	Dashing (JMLE methods, ~10x slower) (top 10), recall 10%, 17.4 min
GCA_902582355.1_genomic.fna (79.24%)	**GCA_002690725.1_genomic.fna**	**GCA_902560315.1_genomic.fna**	**GCA_902582355.1_genomic.fna**
GCA_002690725.1_genomic.fna (78.79%)	**GCA_902560315.1_genomic.fna**	**GCA_002690725.1_genomic.fna**	GCA_902586925.1_genomic.fna
GCA_902560315.1_genomic.fna (77.92%)	**GCA_902517505.1_genomic.fna**	**GCA_902583575.1_genomic.fna**	GCA_902626385.1_genomic.fna
GCA_902517505.1_genomic.fna (77.57%)	**GCA_902583575.1_genomic.fna**	**GCA_902517505.1_genomic.fna**	GCA_003282945.1_genomic.fna
GCA_002169625.2_genomic.fna (76.41%)	**GCA_902579825.1_genomic.fna**	GCA_902559345.1_genomic.fna	GCA_012735275.1_genomic.fna
GCA_902556105.1_genomic.fna (76.37%)	GCA_902559345.1_genomic.fna	**GCA_902579825.1_genomic.fna**	GCF_000165465.1_genomic.fna

GCA_902583575.1_genomic.fna (75.30%)	GCA_902558095.1_genomic.fna	**GCA_902556105.1_genomic.fna**	GCF_003111605.1_genomic.fna
GCA_003213495.1_genomic.fna (75.24%)	**GCA_902556105.1_genomic.fna**	GCA_902592715.1_genomic.fna	GCA_017532025.1_genomic.fna
GCA_902579825.1_genomic.fna (75.15%)	GCA_902528875.1_genomic.fna	GCA_014653355.1_genomic.fna	GCA_902592715.1_genomic.fna
GCA_902563835.1_genomic.fna (75.00%)	GCA_902551185.1_genomic.fna	GCF_000165465.1_genomic.fna	GCA_014653355.1_genomic.fna

Table S16. Recall (top 10) of ProbMinHash, Densified MinHash, SetSketch and SuperMinHash against the BLAST-ANI results. Same query genome as **Table S13** was used.

Blastn-ANI (top10), ground truth	GSearch ProbMinHash (top10), recall 100%, 2s	GSearch Densified Minhash recall 100%, 2s	GSearch setsketch recall 100%, 5s	GSearch SuperMinHash recall 100%, 8s
GCA_902591925.1_genomic.fna	GCA_902591925.1_genomic.fna	GCA_902591925.1_genomic.fna	GCA_902591925.1_genomic.fna	GCA_902591925.1_genomic.fna
GCA_902617045.1_genomic.fna	GCA_902617045.1_genomic.fna	GCA_902617045.1_genomic.fna	GCA_902617045.1_genomic.fna	GCA_902617045.1_genomic.fna
GCA_902541175.1_genomic.fna	GCA_902547295.1_genomic.fna	GCA_902541175.1_genomic.fna	GCA_902612915.1_genomic.fna	GCA_902541175.1_genomic.fna
GCA_902612915.1_genomic.fna	GCA_902612915.1_genomic.fna	GCA_902612915.1_genomic.fna	GCA_902541175.1_genomic.fna	GCA_902612915.1_genomic.fna
GCA_004212975.1_genomic.fna	GCA_004212975.1_genomic.fna	GCA_902547295.1_genomic.fna	GCA_004212975.1_genomic.fna	GCA_004212975.1_genomic.fna
GCA_902547295.1_genomic.fna	GCA_902541175.1_genomic.fna	GCA_004212975.1_genomic.fna	GCA_902547295.1_genomic.fna	GCA_902547295.1_genomic.fna
GCA_902630885.1_genomic.fna	GCA_902586925.1_genomic.fna	GCA_902630885.1_genomic.fna	GCA_902630885.1_genomic.fna	GCA_902630885.1_genomic.fna
GCA_902586925.1_genomic.fna	GCA_902630885.1_genomic.fna	GCA_902586925.1_genomic.fna	GCA_902586925.1_genomic.fna	GCA_902586925.1_genomic.fna
GCA_902631785.1_genomic.fna	GCA_902631785.1_genomic.fna	GCA_902631785.1_genomic.fna	GCA_902631785.1_genomic.fna	GCA_902631785.1_genomic.fna
GCA_000252525.1_genomic.fna	GCA_000252525.1_genomic.fna	GCA_000252525.1_genomic.fna	GCA_902582355.1_genomic.fna	GCA_000252525.1_genomic.fna

Table S17. Recall (top 10) of GSearch (ProbMinHash option), Mash and Sourmash against the BLAST-AAI results. The same query genome as shown in **Table S14** was used but at the proteome level after gene prediction. Dashing and BinDash were not included because these tools do not provide an amino acid level search option. Same query genome as Table S13.

Blastp-AAI (top10), ground truth, AAI	GSearch ProbMinHash (top10), recall 90%, 2s	Mash, recall 90%, 13 min	Sourmash, recall 80%, 3 min
GCA_902591925.1_genomic.fna (83.2%)	**GCA_902591925.1_genomic.fna**	**GCA_902591925.1_genomic.fna**	**GCA_902591925.1_genomic.fna**
GCA_902617045.1_genomic.fna (83.16%)	**GCA_902617045.1_genomic.fna**	**GCA_902617045.1_genomic.fna**	**GCA_902617045.1_genomic.fna**
GCA_902541175.1_genomic.fna (82.57%)	**GCA_004212975.1_genomic.fna**	**GCA_902612915.1_genomic.fna**	**GCA_902541175.1_genomic.fna**
GCA_902612915.1_genomic.fna (82.15%)	**GCA_902612915.1_genomic.fna**	**GCA_902541175.1_genomic.fna**	**GCA_902612915.1_genomic.fna**
GCA_004212975.1_genomic.fna (81.86%)	**GCA_902547295.1_genomic.fna**	**GCA_902547295.1_genomic.fna**	GCA_902558095.1_genomic.fna
GCA_902630885.1_genomic.fna (80.99%)	**GCA_902541175.1_genomic.fna**	**GCA_004212975.1_genomic.fna**	**GCA_902547295.1_genomic.fna**
GCA_902547295.1_genomic.fna (80.73%)	**GCA_902586925.1_genomic.fna**	GCA_902559345.1_genomic.fna	**GCA_902630885.1_genomic.fna**
GCA_902586925.1_genomic.fna (80.50%)	GCA_902558095.1_genomic.fna	**GCA_902586925.1_genomic.fna**	**GCA_902586925.1_genomic.fna**
GCA_000252525.1_genomic.fna (79.96%)	**GCA_902630885.1_genomic.fna**	**GCA_902631785.1_genomic.fna**	**GCA_902631785.1_genomic.fna**
GCA_902631785.1_genomic.fna (79.48%)	**GCA_000252525.1_genomic.fna**	**GCA_902582355.1_genomic.fna**	GCA_902559345.1_genomic.fna

Table S18. Database build time, maximum memory and database size for GSearch (ProbMinHash option), Densified MinHash, SuperMinHash and SetSketch. GTDB v207 was used (65,703 genomes). 24 threads are used.

	Build time	Max Memory	Database size
ProbMinHash (default)	1.3h	15.3G	3.0G
Densified MinHash	0.74h	9.4G	1.9G
SuperMinHash	4.7h	13.2G	3.0G
SetSketch	2.4h	3.7G	0.6G

Table S19. Database build time, maximum memory and database size for GSearch (ProbMinHash option), Densified MinHash, SuperMinHash and SetSketch. NCBI/RefSeq genomes (~318K) were used. 24 threads are used.

	Build time	Max Memory	Database size
ProbMinHash (default)	4.1h	21G	15G
Densified MinHash	1.4h	14G	11G
SuperMinHash	27h	16G	15G
SetSketch	13h	3.7G	3.0G

Table S20. GSearch accuracy benchmark using orthoANI as an additional ground truth. Query genomes is "OceanDNA-b42278.fa" from (Nishimura and Yoshizawa, 2022). Database genomes were all NCBI/RefSeq genomes. Showing top 10 nearest genomes, ANI from ~94% to ~81% for the top 10.

Ortho-ANI (top10), ground truth	Blastn-ANI (ANI calculator), top 10	GSearch (top10, ProbMinHash), recall 100%, 2s
GCA_902591925.1_genomic.fna (95.33%)	GCA_902591925.1_genomic.fna (95.27%)	GCA_902591925.1_genomic.fna (95.38%)
GCA_902617045.1_genomic.fna (94.12%)	GCA_902617045.1_genomic.fna (94.18%)	GCA_902617045.1_genomic.fna (94.29%)
GCA_902541175.1_genomic.fna (93.35%)	GCA_902541175.1_genomic.fna (93.22%)	GCA_902547295.1_genomic.fna (93.31%)
GCA_902612915.1_genomic.fna (92.43%)	GCA_902612915.1_genomic.fna (92.46%)	GCA_902612915.1_genomic.fna (92.83%)
GCA_004212975.1_genomic.fna (92.06%)	GCA_004212975.1_genomic.fna (92.10%)	GCA_004212975.1_genomic.fna (91.91%)
GCA_902547295.1_genomic.fna (90.15%)	GCA_902547295.1_genomic.fna (90.09%)	GCA_902541175.1_genomic.fna (90.35%)
GCA_902630885.1_genomic.fna (88.56%)	GCA_902630885.1_genomic.fna (88.51%)	GCA_902586925.1_genomic.fna (88.12%)
GCA_902586925.1_genomic.fna (85.23%)	GCA_902586925.1_genomic.fna (85.49%)	GCA_902630885.1_genomic.fna (85.37%)
GCA_902631785.1_genomic.fna (83.91%)	GCA_902631785.1_genomic.fna (83.88%)	GCA_902631785.1_genomic.fna (83.48%)
GCA_000252525.1_genomic.fna (81.08%)	GCA_000252525.1_genomic.fna (81.32%)	GCA_000252525.1_genomic.fna (81.51%)

Supplementary Table References

Broder, A.Z. (1997) On the resemblance and containment of documents. In *Proceedings Compression and Complexity of SEQUENCES 1997 (Cat No 97TB100171)*: IEEE, pp. 21-29.
Christiani, T. (2020) DartMinHash: Fast Sketching for Weighted Sets. *arXiv preprint arXiv:200511547*.
Ertl, O. (2017) Superminhash-A new minwise hashing algorithm for jaccard similarity estimation. *arXiv preprint arXiv:170605698*.
Ertl, O. (2018) BagMinHash - Minwise Hashing Algorithm for Weighted Sets. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*: 1368–1377.
Ertl, O. (2020) ProbMinHash – A Class of Locality-Sensitive Hash Algorithms for the (Probability) Jaccard Similarity. *IEEE Transactions on Knowledge and Data Engineering*: 1-1.
Ertl, O. (2021) SetSketch: filling the gap between MinHash and HyperLogLog. *Proc VLDB Endow* **14**: 2244–2257.
Flajolet, P., Fusy, É., Gandouet, O., and Meunier, F. (2007) Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. *Discrete Mathematics and Theoretical Computer Science*: 137-156.
Karppa, M., and Pagh, R. (2022) HyperLogLogLog: Cardinality Estimation With One Log More. *arXiv preprint arXiv:220511327*.
Mino, S., Shiotani, T., Nakagawa, S., Takai, K., and Sawabe, T. (2021) Hydrogenimonas urashimensis sp. nov., a hydrogen-oxidizing chemolithoautotroph isolated from a deep-sea hydrothermal vent in the Southern Mariana Trough. *Systematic and Applied Microbiology* **44**: 126170.
Nishimura, Y., and Yoshizawa, S. (2022) The OceanDNA MAG catalog contains over 50,000 prokaryotic genomes originated from various marine environments. *Scientific Data* **9**: 305.
Pettie, S., and Wang, D. (2022) Simpler and Better Cardinality Estimators for HyperLogLog and PCSA. *arXiv preprint arXiv:220810578*.
Shrivastava, A. (2017) Optimal densification for fast and accurate minwise hashing. *International Conference on Machine Learning*: 3154-3163.

Supplementary Methods & Materials

ProbMinHash vs. traditional MinHash

Mash is a hashing-based algorithm based on MinHash [1], which is very efficient for comparing genome/metagenome overall similarity [2]. Mash distances represent a k-mer-based overall overlap between sequences according to a minimal evolutionary model. Essentially, Mash distance is the Jaccard similarity value of kmer shared between k-mer sets A and B extracted from genome A and B. However, Mash, and similar MinHash-based tools, have several limitations; most notably, the loss of k-mer frequency information (only presence/absence of kmer is counted) and the impact of relative set size (e.g., completeness level of a genome or genome size) on the Jaccard similarity estimates (for example, Mash distance, where MinHash based estimation of Jaccard index is biased by different set size, or total k-mer count of the genome due to bottom-k sketches; HyperLogLog is not affected by different set size) [2, 3] Although some recent MinHash implementations address the relative set size limitation, e.g., the over-sketching and track-abundance methods of the MinHash-based tools ‘finch , ‘sourmash’ or FracMinHash (sketching a subset of k-mers according to size of set) and HyperLogLog [4-7], they do not utilize the frequencies of all observed k-mers in generating the k-mer-profile (sketch) for a given sequence set. More recently, in the HULK software, consistent weighted sampling (D^2histosketch which is the same with P-MinHash algorithm and it was proposed for J_p, we called it P-MinHash because it was invented before D^2histosketch and is equivalent to P-MinHash[8]) [9] was utilized to incorporate k-mer frequency information when estimating weighted and standard Jaccard similarity, which effectively addresses these limitations mentioned above [10]. Notably, the hash algorithm (P-MinHash) used in D^2histosketch could be further optimized to achieve a time complexity below *O(nm)* (where m denotes the signature size and n is the number of elements with nonzero weight in two sequence sets), further improving the performance of applications

such as HULK. Motivated by the SuperMinHash for conventional Jaccard similarity estimation [11] and BagMinHash algorithm for weighted Jaccard similarity estimation [12], ProbMinHash (probminhash 3(a) and 4 algorithm) is orders of magnitude faster than the original algorithm P-MinHash proposed in D^{2}histosketch [14]. Probminhash estimates the Jaccard probability J_p similarity, and 1- Jp is indeed a metric on the probability distributions and is Pareto optimal (Supplementary Note 1) [8, 14]. Densified MinHash, or One Permutation MinHash with Optimal/faster Densification, is the fastest MinHash algorithm due to theoretical breakthrough (average case O(n+m)) despite large variance than classic MinHash (Table S2). It uses only one hash function but copy values from empty bins to non-empty bins ("densified") in the sketch vector either mapping forward or backward or both [13, 15, 16]. We choose Sourmash, Mash, Dashing 1/2 and BinDash for benchmark because all provide estimation of Jaccard index, which correlates very well with ANI after transformation, and had been all previously benchmarked against BLAST-based ANI and fastANI [2, 19].

Comparison with other genome/sequence search algorithm

There are many other data structures designed for general purposes sequence search problems such as Sequence Bloom Tree (SBT) and its variants [20-22], COBS/BIGSI [23, 24], Layered LSH (approximating ungapped alignment, not applicable to ANI like distance, which is gapped alignment) [25] and RAMBO (Repeated and Merged Bloom Filter) [26] that can achieve linear or sometimes even sub-linear genomic database search performance alone. However, those tools have never been benchmarked against BLAST-based ANI/AAI in the context of microbial genomic search (that is, whether the best hits/genomes found by the mentioned tools are the same with ANI/AAI comparison best hits/genomes), which is a popular reference standard for measuring microbial genomic distance/identity, and thus infer microbial taxonomy [27, 28]. MinHash-based tool such as Mash and FastANI have been benchmarked against BLAST-based ANI, and were shown

to be the most accurate k-mer-based sub-linear algorithms for ANI comparisons and/or searching [2, 19].

HNSW in Rust benchmark against testing dataset

To benchmark our reimplementation of hnswlib, we followed standard ANN benchmark procedures using two popular testing datasets (MINST and SIFT1M) based on their Euclidean distance[29]. Our results showed that, for the MINST fashion dataset (784 dimensions, 60,000 vectors), recall for top 100 neighbors of 10,000 query vectors is greater than 98% for a smaller number of M and ef_construct, and even higher recall rate (99.86%) for a medium M and ef_construct while query speed is not compromised (Supplemental Table S7). For the SIFT1M dataset (128 dimensions, 1,000,000 vectors), recall for top 100 neighbors of 10,000 query vectors was 99.77% for a medium M and ef_construct (Supplemental Table S7 and S8). In terms of speed, we compare it with hnswlib using the MINST-fashion dataset and it is as fast as hnswlib: it took 18.06s and 0.89s for database building and searching for hnswlib-rs while it took 18.47s and 1.07s for database building and searching for the C++ hnswlib) (Supplementary Table S9). The Rust package hnswlib-rs can be found at: https://github.com/jean-pierreBoth/hnswlib-rs. For each genomic database, we chose M and ef_construct experimentally, by gradually increasing M and ef_construct while monitoring query speed and recall, similar to what is shown in Supplementary Table S2 for MNIST dataset. We stopped the assessment when there was only a marginal increase in accuracy but decent decrease in speed. To leverage between recall and speed, we use M=128 and ef_search=1600 for graph building for GTDB database fungal database while M=128, ef_search=3200 for phage database.

Details of program implementation in Rust

Tohnsw starts by reading database genomes and generating k-mer profile and sketches using the ProbMinHash3a algorithm (or SuperMinHash, SetSketch, Densified MinHash) for distance calculation (Figure 1a and c). Next, tohnsw selects, at random, the first batch of genomes to insert into the graph (Figure 1a (1)), following HNSW constructing rules mentioned above and taking into account the computed ProbMinHash distance or SetSketch approximated Jaccard distances (1-J) between genomes to connect genomes based on their relatedness (Figure 1b) until all genomes in database have been inserted (Figure 1a (2), (3) and (4)). Finding nearest genomes for the genome to be inserted is essentially a search process but search in a partially built graph, which is similar to the request/search module: whenever a genome is going to be searched against the existing graph, each genome in the graph is associated with a list that stores the M closest neighbors/genomes to the genome and the distance to these neighbors. Then, the distances of this genome with the nearest neighbors (M) of entry genome in each layer will be computed (ef_construct times) using the Probminhash3a algorithm or the SetSketch, and the smallest distance of the neighbor genomes will be the new entry genome (Figure 1d and e). This process will be repeated until the nearest genomes (<=M) in the layer are found and subsequently, the program will go to the layer below, using the genome that was represented by the nearest genome in the above layer as new entry genome in the new layer. The search layer algorithm is repeated until the bottom layer is reached/analyzed (Figure 1c). In contrast to the default settings in the original hnswlib, we allow the two parameters of neighbor selecting heuristics, *extendCandidates* to be true and *keepPrunedConnections* to be false because our genomic data is extremely clustered and there is no need to fix the number of connections per element considering the maximum connection allowed. The “Add” module is to add new genomes to pre-built HNSW database using default parameters loaded in the pre-built database files. Request module will load the graph database and then search query genomes against it to return the best

neighbors of each query, following exactly the same procedure with building step but without updating the database.

Details of the SetSketch implementation

We implemented SetSketch algorithm 1 locality sensitivity section (LSH) according to lower and upper bound of Jaccard ($J_{low} = max(0, \frac{b^{\frac{D_0/m+1}{2}}-1}{b-1} - 1), J_{up} = \frac{b^{D_0/m}-1}{b-1}$) where D_0 is the number of registers in the sketch of genome A that are equal to those in the sketch of genome B. In practice, J_{low} is closer to true Jaccard for small J thus we use it. We use parameter m = 6144, b = 1.0005, a = 20, and q = 65534 instead of the default ones to have smaller RMSE (<0.8%) around J=0.015 (corresponding to ANI 77.99% according to Mash equation) and acceptable running time (Figure S7), which is equivalent to HyperLogLog for estimating Jaccard index in terms of space but with smaller variance (with b close to 2, SetSketch will then be equivalent to SuperMinHash). A SetSketch using this configuration is suitable to represent any set with up to 10^19 distinct elements/k-mers (much larger than total-number of k-mers from microbial genomes). The expected error of cardinality estimates or LSH is very small ($\sqrt{\frac{1}{m}(\frac{b+1}{b-1}log(b) - 1)}$) as b approximates 1($\frac{1}{\sqrt{m}}$) [30], close to that of MinHash for large m, like 10^4 or above but use much smaller space. We also implemented the Joint Maximum Likelihood Estimator (JMLE): the ML estimate for Jaccard was found by standard univariate optimization algorithm called Brent's method, based on the argmin Rust package since the ML function is strictly concave for the parameter mentioned above [30]. The RMSE of JMLE based on its Fisher information can be found in Supplementary Note 7. Since RMSE of JMLE is smaller than LSH, it is much slower in practice, but we only use it for filtering false positives after top k best neighbors were found for each query by LSH and HNSW, thus it is not a problem for overall speed.

Details of the Densified MinHash implementation

We reimplemented both the One Permutation MinHash with Optimal Densification [15] and also the Faster Densification [16] in Rust in the probminhash package. Specifically, one predefined hash function is applied to all elements/k-mers in the set. Then, one-permutation MinHash deterministically partitions the hash values into a predefined maximum number B of buckets, extracts the smallest hash value in each bucket, and extracts the b lowest bits of each smallest hash value. These B*b bits are used as the signature of the set. Usually, a hash value v is assigned to the $\lceil v/(M/B) \rceil$ bucket, where M is the maximum possible value for v. Although fast at both constructing and comparing sketches, one-permutation MinHash may produce a bucket that contains hash values for one set but no hash values for another set. All buckets are totally ordered in some way, each empty bucket uses the smallest hash value in the next non-empty bucket as its own hash value (densified), and an additional bucket containing a special value is ordered after all other buckets. We either map non-empty bins to empty bins (faster densification) or the other way around to copy values (optimal densification). The densified sketch vectors for 2 sets were then used for calculating collision probability/Jaccard index (See densminhash.rs in prominhash package). We can use 2 times larger sketch size m for densified MinHash to achieve a smaller variance compared to original MinHash without increasing running time.

Supplementary Methods & Materials References

1. Broder, A.Z. in Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171) 21-29 (IEEE, 1997).
2. Ondov, B.D. et al. Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biology* **17**, 132 (2016).
3. Koslicki, D. & Zabeti, H. Improving MinHash via the containment index with applications to metagenomic analysis. *Applied Mathematics and Computation* **354**, 206-215 (2019).
4. Brown, C.T. & Irber, L. sourmash: a library for MinHash sketching of DNA. *Journal of Open Source Software* **1**, 27 (2016).
5. Bovee, R. & Greenfield, N. Finch: a tool adding dynamic abundance filtering to genomic MinHashing. *Journal of Open Source Software* **3**, 505 (2018).

6. Irber, L. et al. Lightweight compositional analysis of metagenomes with FracMinHash and minimum metagenome covers. *bioRxiv*, 2022.2001.2011.475838 (2022).
7. Baker, D.N. & Langmead, B. Dashing: fast and accurate genomic distances with HyperLogLog. *Genome Biology* **20**, 265 (2019).
8. Moulton, R. & Jiang, Y. Maximally Consistent Sampling and the Jaccard Index of Probability Distributions. *2018 IEEE International Conference on Data Mining (ICDM)*, 347-356 (2018).
9. Yang, D., Li, B., Rettig, L. & Cudré-Mauroux, P. D²histoSketch: Discriminative and Dynamic Similarity-Preserving Sketching of Streaming Histograms. *IEEE Transactions on Knowledge and Data Engineering* **31**, 1898-1911 (2019).
10. Rowe, W.P.M. et al. Streaming histogram sketching for rapid microbiome analytics. *Microbiome* **7**, 40 (2019).
11. Ertl, O. Superminhash-A new minwise hashing algorithm for jaccard similarity estimation. *arXiv preprint arXiv:1706.05698* (2017).
12. Ertl, O. BagMinHash - Minwise Hashing Algorithm for Weighted Sets. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1368–1377 (2018).
13. Jia, P. et al. in Proceedings of the 2021 International Conference on Management of Data 830-842 (2021).
14. Ertl, O. ProbMinHash – A Class of Locality-Sensitive Hash Algorithms for the (Probability) Jaccard Similarity. *IEEE Transactions on Knowledge and Data Engineering*, 1-1 (2020).
15. Shrivastava, A. Optimal densification for fast and accurate minwise hashing. *International Conference on Machine Learning*, 3154-3163 (2017).
16. Mai, T. et al. in Uncertainty in Artificial Intelligence 831-840 (PMLR, 2020).
17. Baker, D.N. & Langmead, B. Dashing 2: genomic sketching with multiplicities and locality-sensitive hashing. *bioRxiv* (2022).
18. Agret, C., Cazaux, B. & Limasset, A. Toward optimal fingerprint indexing for large scale genomics. *bioRxiv*, 2021.2011.2004.467355 (2022).
19. Jain, C., Rodriguez-R, L.M., Phillippy, A.M., Konstantinidis, K.T. & Aluru, S. High throughput ANI analysis of 90K prokaryotic genomes reveals clear species boundaries. *Nature Communications* **9**, 5114 (2018).
20. Solomon, B. & Kingsford, C. Fast search of thousands of short-read sequencing experiments. *Nature biotechnology* **34**, 300-302 (2016).
21. Solomon, B. & Kingsford, C. Improved search of large transcriptomic sequencing databases using split sequence bloom trees. *Journal of Computational Biology* **25**, 755-765 (2018).
22. Sun, C., Harris, R.S., Chikhi, R. & Medvedev, P. Allsome sequence bloom trees. *Journal of Computational Biology* **25**, 467-479 (2018).
23. Bingmann, T., Bradley, P., Gauger, F. & Iqbal, Z. in String Processing and Information Retrieval: 26th International Symposium, SPIRE 2019, Segovia, Spain, October 7–9, 2019, Proceedings 26 285-303 (Springer, 2019).
24. Bradley, P., Den Bakker, H.C., Rocha, E.P., McVean, G. & Iqbal, Z. Ultrafast search of all deposited bacterial and viral genomic data. *Nature biotechnology* **37**, 152-159 (2019).
25. Chakraborty, A. & Bandyopadhyay, S. in 2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT) 1-4 (IEEE, 2018).
26. Gupta, G. et al. in Proceedings of the 2021 International Conference on Management of Data 2226-2234 (2021).

27. Konstantinidis, K.T. & Tiedje, J.M. Towards a Genome-Based Taxonomy for Prokaryotes. *Journal of Bacteriology* **187**, 6258-6264 (2005).
28. Goris, J. et al. DNA–DNA hybridization values and their relationship to whole-genome sequence similarities. *International Journal of Systematic and Evolutionary Microbiology* **57**, 81-91 (2007).
29. Aumüller, M., Bernhardsson, E. & Faithfull, A. ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems* **87**, 101374 (2020).
30. Ertl, O. SetSketch: filling the gap between MinHash and HyperLogLog. *Proc. VLDB Endow.* **14**, 2244–2257 (2021).

Supplementary Notes

Note 1. **Calculation of metric (Weighted) Jaccard similarity indices**. $J_W = \frac{\sum_{d\in D} min(\omega_A(d),\omega_B(d))}{\sum_{d\in D} max(\omega_A(d),\omega_B(d))}$ (where ω is the weight function of each element d) and total k-mer count in genomes in biasing the Jaccard index estimation (or set size) [1]. K-mer-weighted hashing approaches (taking into account the abundance of k-mers, not just presence/absence) are more space-expensive (e.g. higher memory requirement) for large dataset [2], but are advantageous for genomes with frequent repeats. They have not been widely adopted yet [3, 4]. To consider multiplicity of k-mers in the k-mer set of genomes, traditional MinHash algorithms will not be a good choice since they assume unique set element (k-mer). New MinHash algorithms such as ICWS, BagMinHash and DartMinHash were designed for weighted set to address this limitation, with DartMinHash being the fastest [5-8]. Still, those weighted MinHash algorithms do not solve the problem of different genome size (set size) in biasing estimation of weighted Jaccard index [9]. A possible solution is to normalize the abundance of k-mer by the total k-mer count of each dataset, thereby providing a probability distribution of each k-mer. This then leads to the normalized weighted Jaccard index $J_N = \frac{\sum_{d\in D} min(\frac{\omega_A(d)}{\sum_{d'\in D}\omega_A(d')}, \frac{\omega_B(d)}{\sum_{d'\in D}\omega_B(d')})}{\sum_{d\in D} max(\frac{\omega_A(d)}{\sum_{d'\in D}\omega_A(d')}, \frac{\omega_B(d)}{\sum_{d'\in D}\omega_B(d')})}$, where ω_A and ω_B describe the weight of each k-mer. Also, those weighted MinHash algorithm could be further optimized computationally to be orders of magnitude faster, like it was done in BagMinHash and DartMinHash [10], similar to the computational optimizations implemented in MinHash in SuperMinHash [11]. Recently, ProbMinHash was proposed to take into account both weighted set (k-mer multiplicity) and total set size (total k-mer count, or genome size) [12]. Accordingly, new Locality Sensitive Hashing algorithms (P-MinHash) considering weighted set and different set size was proposed to estimate weighted and normalized (to account for set size difference) Jaccard-like index $J_P =$

$\sum_{d\in D} \frac{1}{\sum_{d'\in D} max(\frac{\omega_A(d')}{\omega_A(d)}, \frac{\omega_B(d')}{\omega_B(d)})}$. P-MinHash is considered a more general case for Jaccard Index estimation and is more close to true Jaccard than J_W and J_N [13, 14]. More importantly, J_p is Pareto optimal and 1-J_p is a proper metric [13]. It has been shown that ANI estimated from J_p, which was computed by ProbMinHash (ProbMinHash2 algorithm) in Dashing 2, is slight better for bacterial genomes than traditional MinHash like Mash [15]. ProbMinHash was built upon the new MinHash algorithm with further computational optimization for speed [12]. Hence, ProbMinHash is the currently the default option in GSearch. HyperLogLog can also be used to estimate Jaccard index: distinct element count of k-mers in set/genome A and B via HyperLogLog sketch (memory efficiency) and then use the inclusion-exclusion rule to have Jaccard index ($Jaccard(A,B) = \frac{|A|+|B|-|A\cup B|}{|A\cup B|}$) as implemented in Dashing, despite being overall less accurate for small cardinalities (e.g. virus genomes) [16, 17]. The Maximum Likelihood estimator (MLE) [16], in addition to the original estimator in HyperLogLog and improved estimator in [16] for distinct element counting, was thought to have the smallest variance, which met the Cramér-Rao lower bound, despite slower and difficult to compute and update [18]. This is the idea behind Dashing, which we benchmark against GSearch.

Note 2. Kmer selecting rationale for nucleotide (nt) and amino-acid (aa) level searches for MinHash/probminhash tools. Assume amino acid/nucleotide sequences evolve at a constant rate and alphabets $|\Sigma|$ for AA is 20 and 4 for nucleotide. Consider two sequences x , y $\in A^N$ drawn randomly over the alphabet of size #A = 20, N is the genome length, and let $v_x, v_y \in R^{20\wedge k}$ denote the k-mer frequency profile (multiplicity of each k-mer divided by total number of k-mers) for each sequence. For long sequences with $N \gg 20^k$, k-mer frequencies will converge to their mean, that

is 20^{-k} for all their components, which implies that $||v_x - v_y||_1 \rightarrow 0$. Therefore, any k-mer profile-based method will severely underestimate distance between these two random sequences. In order to avoid this, k has to be restricted to large values $k' \geq log_{|\Sigma|}(N)$. In practice, since genomes sequences are not completely random, k can be slightly smaller than $\log_4(N)$ without underestimating genomic distance. Ondov et al., in the MASH paper, came up with a probability term (1-q)/q to take into account the probability of having a random kmer: $k' \geq log_{|\Sigma|}(N(1-q)/q)$ [19]. Assume a probability q=0.01, for a typical bacterial proteome, k>=$\log_{20}$(1000000*0.99/0.01) =6.146, so at least a kmer of size 6 or above should be used. For a typical bacterial genome at nucleotide level, N is about 4*10^6, k >= $\log_4$(4*10^6*0.99/0.01) =14.280, so at least a kmer of size 14 or above should be used. For universal genes only, N is about 350 AA * 120=42000, so k >= $\log_{20}$(42000*0.99/0.01) = 5.087, so at least a kmer of size 5 or above should be used. Since universal gene alphabets are not randomly evolving (e.g., some regions in genes that encode key metabolic functions such as RNA processing units rarely mutate), k=5 should be appropriate. To optimize between sensitivity and specificity for kmer, we followed the practice suggested by Ondov et al., and Jain et al. [19, 20]; that is, use k=16 for bacterial nucleotide genome sequences and a k=7 for bacterial proteome sequences. For fungal genomes, genome size is 10 to 20 times larger than tht of bacteria but the same formula applies and so, we have k >= $\log_4$(20*4*10^6*0.99/0.01) =16.441, and accordingly we use kmer=21. For fungal proteome, coding density is much smaller than that of bacteria (we use an empirical 0.5 coding density) thus, we have k >= $\log_{20}$(10*1*10^6*0.99/0.01) =6.914. We use k=11 for fungal proteome to increase the specificity of the searches. For virus/bacteriophage genomes, we use k=11 for nt and k=7 for aa, considering also the jumbo/giant phage genomes that were found recently.

Note 3. Assessment of the time complexity of ProbMinHash and HNSW. The search time complexity is estimated based on the equation (or big O annotation) $O(vdlog(n))$, where n is the dataset size while v and d are maximum out-degree of the graph to be built and the number of dimensions (kmer features in a genome) of the dataset, respectively [21, 22]. In nearest neighbor search studies, L2 distance (Euclidean distance) is often used and is related to dataset dimension, which is very large for k-mer features of genomes (~10^6). In the case of ProbMinHash distance, which is a MinHash-based method to sample the k-mer space of genomes and serve as a dimensionality reduction in approximating genomic distance, the sampled number of k-mers (or meanwise hashes) is always a constant and much smaller number (sketch size, normally around 10,000) than the total number of k-mers of a genome (number of dimensions of dataset). Therefore, d can be ignored (MinHash samples only a small fraction of k-mer space/dimension). Also, v is a small constant considering the graph structure to be built (normally smaller than 100). Therefore, the time complexity, as it was also explained in the main text, can be safely written as $O(log(n))$ for low dimension datasets [22]. Similar rules applied for the time complexity of the graph build step, which is $O(dnlog(n))$ and thus, $O(nlog(n))$ in our case, where d is dataset dimension. Time complexity $O(n + mlog(m))$ of the Probminhash3a algorithm, where n is the set size (total sampled k-mer number) while m is the number of weighted elements (k-mers), is also a constant in the context of high-quality genomes given also that m is very small for prokaryotic and bacteriophage genomes (normally about 5% genomic sequences are multi-copy). Therefore, ProbMinHash time complexity is very close to $O(n)$ in practice. For a given genome, for example, bacterial genome or viral genome, n is the sampled k-mer number of the genome, and thus also a constant number.

Note 4. Kmer-based genomic distance estimation is less accurate for distantly related genomes. For a random mutation rate $r \in (0,1)$, the probability that a k-mer (k ls length of the k-

mer) belonging to sequence X and Y is not mutated is $1-(1-r)^k \approx 1-exp(-\frac{r}{k})$, indicating that $k \lesssim r^{-1}$, so k must be small enough to capture the mutation and also big enough to avoid underestimation (**Note 2**). This means that for close related bacterial genomes (r<0.01 for example), k could easily meet both conditions mentioned above while for distantly related genomes -for example- r=0.125 (corresponding to an ANI value of 87.5%) k <= 8, which is contradictory with $k>\log_4(N)=14.28$ (Note 2). Thus, k-mer based method will lose accuracy for distantly related genomes. In practice, we observed that this r threshold is around 0.215 (ANI=78.5%) because mutation is not completely random as assumed above.

Note 5. Theoretical guarantee of graph based NNS search algorithms. Until recently, a theoretical analysis based on a dataset evenly distributed on an d-dimension Euclidean sphere ($d \ll \log(n)$, n is the dataset size) showed that under certain conditions, there is a guarantee that the best neighbors could be found compare to brute-force distance metric comparisons [23]. However, a theoretical analysis under more general conditions, e.g., other metric space, or d not being much smaller than log(n) is still not available, to the best of our knowledge. Despite the lack of theoretical analysis under more general conditions, graph-based algorithms work well in practice, as also reflected by the large number of graph-based NNS libraries available [21, 24].

Note 6. Parallelism of probminhash and HNSW. Due to ownership mechanisms of Rust (so called memory and thread safety), fearless concurrency (e.g., no data competition/race, memory and thread safety) is made possible. The crossbeam crate package was used in GSearch for communication of data among threads for task level parallelism while Rayon crate was used to do data level parallelism [25]. Accordingly, by default, GSearch uses all available processors/threads to make full use of multi-processor CPUs.

Note 7. Details of Joint Maximum Likelihood estimator in SetSketch algorithm. For cardinalities n_U and n_v of set U and V estimated by SetSketch cardinality estimation section according to equation 12 in [26], the maximum likelihood method can be used to estimate J. Specifically, the log-likelihood function as a function of J is:

$$log\mathcal{L}(J) = D_+ log(p_b(u - vJ)) + D_- log(p_b(v - uJ)) + D_0 log(1 - p_b(u - vJ) - p_b(v - uJ)),$$

where $D_+ = |\{i: K_{Ui} > K_{Vi}\}|, D_- = |\{i: K_{Ui} < K_{Vi}\}|, D_0 = |\{i: K_{Ui} = K_{Vi}\}|$ are number of registers in the sketch of U that are greater than, less than, or equal to those in the sketch of V, respectively; $K_{Ui}\ and\ K_{Vi}$ are registers of set U and V, respectively while v and u is relative cardinalities $u = \frac{n_U}{n_U + n_V}$ and $v = \frac{n_v}{n_U + n_V}$, respectively. $p_b(x)$ is defined as $p_b(x) = -log_b(1 - x\frac{b-1}{b})$. The RMSE of the ML estimate is expected to be $I^{-1/2}(J)$, where I denotes the Fisher information with respect to J for n_U and n_v:

$$I(J) = \frac{m(b-1)^2}{b^2 log^2(b)}\left(\frac{(vb^{p_b(u-vJ)})^2}{p_b(u-vJ)} + \frac{(ub^{p_b(v-uJ)})^2}{p_b(v-uJ)} + \frac{(vb^{p_b(u-vJ)} + ub^{p_b(v-uJ)})^2}{1 - p_b(u-vJ) - p_b(v-uJ)}\right)$$

The SetSketch paper showed that the estimation error (RMSE) for J will be almost the same for MinHash with the same number of registers m.

Supplementary Notes References

1. Rowe, W.P.M. et al. Streaming histogram sketching for rapid microbiome analytics. *Microbiome* **7**, 40 (2019).
2. Marchet, C., Iqbal, Z., Gautheret, D., Salson, M. & Chikhi, R. REINDEER: efficient indexing of k-mer presence and abundance in sequencing datasets. *Bioinformatics* **36**, i177-i185 (2020).
3. Brown, C.T. & Irber, L. sourmash: a library for MinHash sketching of DNA. *Journal of Open Source Software* **1**, 27 (2016).
4. Bovee, R. & Greenfield, N. Finch: a tool adding dynamic abundance filtering to genomic MinHashing. *Journal of Open Source Software* **3**, 505 (2018).
5. Ioffe, S. in 2010 IEEE International Conference on Data Mining 246-255 (2010).

6. Shrivastava, A. Simple and efficient weighted minwise hashing. *Advances in Neural Information Processing Systems* **29** (2016).
7. Christiani, T. DartMinHash: Fast Sketching for Weighted Sets. *arXiv preprint arXiv:2005.11547* (2020).
8. Wu, W., Li, B., Chen, L., Gao, J. & Zhang, C. A review for weighted minhash algorithms. *IEEE Transactions on Knowledge and Data Engineering* **34**, 2553-2573 (2020).
9. Koslicki, D. & Zabeti, H. Improving MinHash via the containment index with applications to metagenomic analysis. *Applied Mathematics and Computation* **354**, 206-215 (2019).
10. Ertl, O. in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining 1368–1377 (Association for Computing Machinery, London, United Kingdom; 2018).
11. Ertl, O. Superminhash-A new minwise hashing algorithm for jaccard similarity estimation. *arXiv preprint arXiv:1706.05698* (2017).
12. Ertl, O. ProbMinHash – A Class of Locality-Sensitive Hash Algorithms for the (Probability) Jaccard Similarity. *IEEE Transactions on Knowledge and Data Engineering*, 1-1 (2020).
13. Moulton, R. & Jiang, Y. in 2018 IEEE International Conference on Data Mining (ICDM) 347-356 (2018).
14. Yang, D., Li, B., Rettig, L. & Cudré-Mauroux, P. D²histoSketch: Discriminative and Dynamic Similarity-Preserving Sketching of Streaming Histograms. *IEEE Transactions on Knowledge and Data Engineering* **31**, 1898-1911 (2019).
15. Baker, D.N. & Langmead, B. Dashing 2: genomic sketching with multiplicities and locality-sensitive hashing. *bioRxiv* (2022).
16. Ertl, O. New cardinality estimation algorithms for HyperLogLog sketches. *arXiv preprint arXiv:1702.01284* (2017).
17. Baker, D.N. & Langmead, B. Dashing: fast and accurate genomic distances with HyperLogLog. *Genome Biology* **20**, 265 (2019).
18. Pettie, S. & Wang, D. Simpler and Better Cardinality Estimators for HyperLogLog and PCSA. *arXiv preprint arXiv:2208.10578* (2022).
19. Ondov, B.D. et al. Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biology* **17**, 132 (2016).
20. Jain, C., Rodriguez-R, L.M., Phillippy, A.M., Konstantinidis, K.T. & Aluru, S. High throughput ANI analysis of 90K prokaryotic genomes reveals clear species boundaries. *Nature Communications* **9**, 5114 (2018).
21. Lu, K., Kudo, M., Xiao, C. & Ishikawa, Y. HVS: hierarchical graph structure based on voronoi diagrams for solving approximate nearest neighbor search. *Proceedings of the VLDB Endowment* **15**, 246-258 (2021).
22. Malkov, Y.A. & Yashunin, D.A. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **42**, 824-836 (2020).
23. Prokhorenkova, L. & Shekhovtsov, A. Graph-based Nearest Neighbor Search: From Practice to Theory. *Proceedings of the 37th International Conference on Machine Learning* **119**, 7803--7813 (2020).
24. Fu, C., Xiang, C., Wang, C. & Cai, D. Fast approximate nearest neighbor search with the navigating spreading-out graph. *arXiv preprint arXiv:1707.00143* (2017).
25. Moraza, I.E. Rust High Performance. (Packt Publishing, Birmingham, UK; 2018).
26. Ertl, O. SetSketch: filling the gap between MinHash and HyperLogLog. *Proc. VLDB Endow.* **14**, 2244–2257 (2021).

Supplementary File 1

user_genome	FastANI/GTDB-Tk best hit	GSearch best hit
GWMC127_GWMC127.002.fasta.gz	GCF_000738435.1	GCF_000738435.1
GWMC127_GWMC127.023.fasta.gz	GCA_002685195.1	GCA_002685195.1
GWMC127_GWMC127.045.fasta.gz	GCA_002389265.1	GCA_002389265.1
GWMC127_GWMC127.14.fasta.gz	GCA_002711735.1	GCA_002711735.1
GWMC127_GWMC127.25.fasta.gz	GCF_001642945.1	GCF_001642945.1
GWMC127_GWMC127.37.fasta.gz	GCF_900197625.1	GCF_900197625.1
GWMC127_GWMC127.47_sub.fasta.gz	GCA_003045825.1	GCA_003045825.1
GWMC127_GWMC127.51.fasta.gz	GCA_003483155.1	GCA_003483155.1
GWMC127_GWMC127.64.fasta.gz	GCA_002387615.1	GCA_002387615.1
GWMC131_GWMC131.002.fasta.gz	GCF_000738435.1	GCF_000738435.1
GWMC131_GWMC131.024.fasta.gz	GCA_003045825.1	GCA_003045825.1
GWMC131_GWMC131.067.fasta.gz	GCA_003482475.1	GCA_003482475.1
GWMC131_GWMC131.18.fasta.gz	GCA_002711735.1	GCA_002711735.1
GWMC132_GWMC132.003.fasta.gz	GCF_000738435.1	GCF_000738435.1
GWMC132_GWMC132.019.fasta.gz	GCA_002711735.1	GCA_002711735.1
GWMC132_GWMC132.049.fasta.gz	GCA_003483155.1	GCA_003483155.1
GWMC132_GWMC132.3.fasta.gz	GCA_002684605.1	GCA_002684605.1
GWMC132_GWMC132.53.fasta.gz	GCA_003483505.1	GCA_003483505.1
GWMC132_GWMC132.55.fasta.gz	GCA_003045825.1	GCA_003045825.1
GWMC132_GWMC132.80.fasta.gz	GCF_000315525.1	GCF_000315525.1
GWMC244_GWMC244.021.fasta.gz	GCA_003486095.1	GCA_003486095.1
GWMC244_GWMC244.22.fasta.gz	GCA_003023665.1	GCA_003023665.1
GWMC244_GWMC244.28.fasta.gz	GCA_008081045.1	GCA_008081045.1
GWMC244_GWMC244.31.fasta.gz	GCA_013911165.1	GCA_013911165.1
GWMC245_GWMC245.104.fasta.gz	GCA_009921445.1	GCA_009921445.1
GWMC245_GWMC245.24.fasta.gz	GCF_003335675.1	GCF_003335675.1
GWMC245_GWMC245.34.fasta.gz	GCA_011525065.1	GCA_011525065.1
GWMC245_GWMC245.36.fasta.gz	GCA_003023665.1	GCA_003023665.1
GWMC245_GWMC245.8.fasta.gz	GCA_008081045.1	GCA_008081045.1
GWMC247_GWMC247.32.fasta.gz	GCA_003023665.1	GCA_003023665.1
GWMC247_GWMC247.38.fasta.gz	GCA_002700385.1	GCA_002700385.1
GWMC247_GWMC247.42.fasta.gz	GCA_008081045.1	GCA_008081045.1
GWMC247_GWMC247.55_sub.fasta.gz	GCA_009919335.1	GCA_009919335.1
GWMC264_GWMC264.0007.fasta.gz	GCA_003482475.1	GCA_003482475.1
GWMC264_GWMC264.10.fasta.gz	GCF_900197625.1	GCF_900197625.1
GWMC264_GWMC264.35.fasta.gz	GCA_009919335.1	GCA_009919335.1
GWMC264_GWMC264.45.fasta.gz	GCF_000738435.1	GCF_000738435.1
GWMC264_GWMC264.5.fasta.gz	GCA_002390525.1	GCA_002390525.1
GWMC265_GWMC265.0031_sub.fasta.gz	GCF_001642945.1	GCF_001642945.1
GWMC265_GWMC265.26.fasta.gz	GCF_900197625.1	GCF_900197625.1
GWMC265_GWMC265.47.fasta.gz	GCF_000738435.1	GCF_000738435.1
GWMC265_GWMC265.51.fasta.gz	GCA_002390525.1	GCA_002390525.1
GWMC266_GWMC266.021_sub.fasta.gz	GCA_003045825.1	GCA_003045825.1
GWMC266_GWMC266.12.fasta.gz	GCA_009919335.1	GCA_009919335.1
GWMC266_GWMC266.18.fasta.gz	GCF_900197625.1	GCF_900197625.1
GWMC266_GWMC266.24.fasta.gz	GCF_000738435.1	GCF_000738435.1
GWMC266_GWMC266.27.fasta.gz	GCA_003482475.1	GCA_003482475.1
GWMC266_GWMC266.3.fasta.gz	GCA_002390525.1	GCA_002390525.1
GWMC281_GWMC281.019.fasta.gz	GCA_003486095.1	GCA_003486095.1
GWMC281_GWMC281.021.fasta.gz	GCA_009919335.1	GCA_009919335.1
GWMC281_GWMC281.049.fasta.gz	GCA_003045935.1	GCA_003045935.1
GWMC281_GWMC281.066_sub.fasta.gz	GCA_003482475.1	GCA_003482475.1
GWMC281_GWMC281.22_sub.fasta.gz	GCA_003045825.1	GCA_003045825.1
GWMC281_GWMC281.48_sub.fasta.gz	GCF_000738435.1	GCF_000738435.1
GWMC282_GWMC282.004.fasta.gz	GCA_003486095.1	GCA_003486095.1
GWMC282_GWMC282.006.fasta.gz	GCA_002691565.1	GCA_002691565.1
GWMC282_GWMC282.020_sub.fasta.gz	GCA_010023085.1	GCA_010023085.1
GWMC282_GWMC282.20.fasta.gz	GCA_011525015.1	GCA_011525015.1
GWMC282_GWMC282.22.fasta.gz	GCF_000738435.1	GCF_000738435.1
GWMC282_GWMC282.27.fasta.gz	GCA_002457055.1	GCA_002457055.1
GWMC282_GWMC282.34.fasta.gz	GCF_900197625.1	GCF_900197625.1
GWMC282_GWMC282.37.fasta.gz	GCA_009886815.1	GCA_009886815.1
GWMC282_GWMC282.44.fasta.gz	GCA_003045935.1	GCA_003045935.1
GWMC282_GWMC282.53.fasta.gz	GCF_003335675.1	GCF_003335675.1
GWMC282_GWMC282.56.fasta.gz	GCA_003023665.1	GCA_003023665.1
GWMC282_GWMC282.57.fasta.gz	GCA_002480045.1	GCA_002480045.1
GWMC284_GWMC284.010_sub.fasta.gz	GCA_003486095.1	GCA_003486095.1
GWMC284_GWMC284.039_sub.fasta.gz	GCA_009919335.1	GCA_009919335.1
GWMC284_GWMC284.28.fasta.gz	GCA_003023665.1	GCA_003023665.1

GWMC284_GWMC284.3.fasta.gz	GCF_900197605.1	GCF_900197605.1
GWMC284_GWMC284.84.fasta.gz	GCA_009923785.1	GCA_009923785.1
GWMC304_GWMC304.038.fasta.gz	GCA_009921445.1	GCA_009921445.1
GWMC304_GWMC304.24.fasta.gz	GCA_008081045.1	GCA_008081045.1
GWMC497_GWMC497.11.fasta.gz	GCA_002390485.1	GCA_002390485.1
GWMC497_GWMC497.24.fasta.gz	GCA_002457245.1	GCA_002457245.1
GWMC497_GWMC497.29.fasta.gz	GCA_009919335.1	GCA_009919335.1
GWMC497_GWMC497.34.fasta.gz	GCA_002389265.1	GCA_002389265.1
GWMC497_GWMC497.41.fasta.gz	GCF_001735715.1	GCF_001735715.1
GWMC497_GWMC497.45.fasta.gz	GCA_009936895.1	GCA_009936895.1
GWMC497_GWMC497.58.fasta.gz	GCF_900197625.1	GCF_900197625.1
GWMC497_GWMC497.6.fasta.gz	GCF_000738435.1	GCF_000738435.1
GWMC497_GWMC497.62.fasta.gz	GCA_003485335.1	GCA_003485335.1
GWMC539_GWMC539.009.fasta.gz	GCA_002691565.1	GCA_002691565.1
GWMC539_GWMC539.16.fasta.gz	GCF_000384415.1	GCF_000384415.1
GWMC539_GWMC539.23.fasta.gz	GCA_003023665.1	GCA_003023665.1
GWMC539_GWMC539.2_sub.fasta.gz	GCF_000738435.1	GCF_000738435.1
GWMC539_GWMC539.35.fasta.gz	GCA_003486095.1	GCA_003486095.1
GWMC539_GWMC539.8.fasta.gz	GCA_003045935.1	GCA_003045935.1
GWMC540_GWMC540.003.fasta.gz	GCA_003486095.1	GCA_003486095.1
GWMC540_GWMC540.012.fasta.gz	GCA_002691565.1	GCA_002691565.1
GWMC540_GWMC540.18.fasta.gz	GCA_011525015.1	GCA_011525015.1
GWMC540_GWMC540.28.fasta.gz	GCA_003023665.1	GCA_003023665.1
GWMC540_GWMC540.38.fasta.gz	GCA_003045935.1	GCA_003045935.1
GWMC550_GWMC550.037.fasta.gz	GCF_000179255.1	GCF_000179255.1
GWMC550_GWMC550.14.fasta.gz	GCF_003335675.1	GCF_003335675.1
GWMC550_GWMC550.28.fasta.gz	GCA_008081045.1	GCA_008081045.1
GWMC550_GWMC550.45.fasta.gz	GCA_003023665.1	GCA_003023665.1
GWMC551_GWMC551.14.fasta.gz	GCA_008081045.1	GCA_008081045.1
GWMC551_GWMC551.4.fasta.gz	GCF_003335675.1	GCF_003335675.1
GWMC552_GWMC552.021.fasta.gz	GCA_008081045.1	GCA_008081045.1

Supplementary File 2

The same query genome BRRP3A.13 for the following

Top 10 from split database 1

0 path ../../GWMC_1000_HQ_faa_aai_635/BRRP3A.13.faa, fasta_id total sequence, len : 800987

distance : 9.156E-1 answer fasta id ../database/database_1/GB_GCA_016699615.1_protein.faa
answer fasta id total sequence, seq len : 906076
distance : 9.452E-1 answer fasta id ../database/database_1/GB_GCA_017984135.1_protein.faa
answer fasta id total sequence, seq len : 797872
distance : 9.822E-1 answer fasta id ../database/database_1/GB_GCA_001897535.1_protein.faa
answer fasta id total sequence, seq len : 957913
distance : 9.835E-1 answer fasta id ../database/database_1/GB_GCA_018268055.1_protein.faa
answer fasta id total sequence, seq len : 974493
distance : 9.839E-1 answer fasta id ../database/database_1/GB_GCA_016787045.1_protein.faa
answer fasta id total sequence, seq len : 1127633
distance : 9.839E-1 answer fasta id ../database/database_1/GB_GCA_018268135.1_protein.faa
answer fasta id total sequence, seq len : 960055
distance : 9.843E-1 answer fasta id ../database/database_1/GB_GCA_018267655.1_protein.faa
answer fasta id total sequence, seq len : 1045076
distance : 9.843E-1 answer fasta id ../database/database_1/GB_GCA_001800905.1_protein.faa
answer fasta id total sequence, seq len : 646701
distance : 9.843E-1 answer fasta id ../database/database_1/GB_GCA_017303895.1_protein.faa
answer fasta id total sequence, seq len : 1121215
distance : 9.844E-1 answer fasta id ../database/database_1/GB_GCA_002281875.1_protein.faa
answer fasta id total sequence, seq len : 962695

Top 10 from split database 2

0 path ../../GWMC_1000_HQ_faa_aai_635/BRRP3A.13.faa, fasta_id total sequence, len : 800987

distance : 9.551E-1 answer fasta id ../database/database_2/GB_GCA_016711605.1_protein.faa
answer fasta id total sequence, seq len : 1152198
distance : 9.832E-1 answer fasta id ../database/database_2/GB_GCA_018267475.1_protein.faa
answer fasta id total sequence, seq len : 858052
distance : 9.846E-1 answer fasta id ../database/database_2/GB_GCA_903920455.1_protein.faa
answer fasta id total sequence, seq len : 710803
distance : 9.850E-1 answer fasta id ../database/database_2/RS_GCF_003384935.1_protein.faa
answer fasta id total sequence, seq len : 1036633
distance : 9.850E-1 answer fasta id ../database/database_2/RS_GCF_004342685.1_protein.faa
answer fasta id total sequence, seq len : 1066355
distance : 9.851E-1 answer fasta id ../database/database_2/GB_GCA_002307135.1_protein.faa
answer fasta id total sequence, seq len : 1018135
distance : 9.852E-1 answer fasta id ../database/database_2/GB_GCA_018267625.1_protein.faa
answer fasta id total sequence, seq len : 901729
distance : 9.852E-1 answer fasta id ../database/database_2/RS_GCF_000348685.1_protein.faa
answer fasta id total sequence, seq len : 920194
distance : 9.852E-1 answer fasta id ../database/database_2/GB_GCA_018267875.1_protein.faa
answer fasta id total sequence, seq len : 822214
distance : 9.853E-1 answer fasta id ../database/database_2/GB_GCA_018267925.1_protein.faa
answer fasta id total sequence, seq len : 1037058

Top 10 from database 3

0 path ../../GWMC_1000_HQ_faa_aai_635/BRRP3A.13.faa, fasta_id total sequence, len : 800987

distance : 9.711E-1 answer fasta id ../database/database_3/GB_GCA_016705125.1_protein.faa
answer fasta id total sequence, seq len : 973431
distance : 9.843E-1 answer fasta id ../database/database_3/GB_GCA_903840505.1_protein.faa
answer fasta id total sequence, seq len : 884513
distance : 9.847E-1 answer fasta id ../database/database_3/GB_GCA_001567275.1_protein.faa
answer fasta id total sequence, seq len : 906415
distance : 9.847E-1 answer fasta id ../database/database_3/GB_GCA_903857255.1_protein.faa
answer fasta id total sequence, seq len : 903318
distance : 9.847E-1 answer fasta id ../database/database_3/GB_GCA_019136655.1_protein.faa
answer fasta id total sequence, seq len : 909617
distance : 9.849E-1 answer fasta id ../database/database_3/RS_GCF_900167075.1_protein.faa
answer fasta id total sequence, seq len : 1236890
distance : 9.851E-1 answer fasta id ../database/database_3/GB_GCA_016786925.1_protein.faa
answer fasta id total sequence, seq len : 1087491
distance : 9.851E-1 answer fasta id ../database/database_3/GB_GCA_016715625.1_protein.faa
answer fasta id total sequence, seq len : 1063770
distance : 9.851E-1 answer fasta id ../database/database_3/GB_GCA_903828245.1_protein.faa
answer fasta id total sequence, seq len : 615471
distance : 9.853E-1 answer fasta id ../database/database_3/GB_GCA_017303835.1_protein.faa
answer fasta id total sequence, seq len : 1008010

Top 10 from database 4

0 path ../../GWMC_1000_HQ_faa_aai_635/BRRP3A.13.faa, fasta_id total sequence, len : 800987

distance : 9.540E-1 answer fasta id ../database/database_4/GB_GCA_016721245.1_protein.faa
answer fasta id total sequence, seq len : 1111343
distance : 9.829E-1 answer fasta id ../database/database_4/GB_GCA_018262795.1_protein.faa
answer fasta id total sequence, seq len : 1041576
distance : 9.838E-1 answer fasta id ../database/database_4/GB_GCA_002352045.1_protein.faa
answer fasta id total sequence, seq len : 1030255
distance : 9.843E-1 answer fasta id ../database/database_4/GB_GCA_018268195.1_protein.faa
answer fasta id total sequence, seq len : 923024
distance : 9.843E-1 answer fasta id ../database/database_4/GB_GCA_903861055.1_protein.faa
answer fasta id total sequence, seq len : 1018870
distance : 9.846E-1 answer fasta id ../database/database_4/GB_GCA_016183765.1_protein.faa

answer fasta id total sequence, seq len : 1094501
distance : 9.847E-1 answer fasta id ../database/database_4/GB_GCA_004295015.1_protein.faa
answer fasta id total sequence, seq len : 1160623
distance : 9.847E-1 answer fasta id ../database/database_4/RS_GCF_009363055.1_protein.faa
answer fasta id total sequence, seq len : 862789
distance : 9.847E-1 answer fasta id ../database/database_4/GB_GCA_018267725.1_protein.faa
answer fasta id total sequence, seq len : 822915
distance : 9.849E-1 answer fasta id ../database/database_4/GB_GCA_016788165.1_protein.faa
answer fasta id total sequence, seq len : 1025440

Top 10 from split database 5

0 path ../../GWMC_1000_HQ_faa_aai_635/BRRP3A.13.faa, fasta_id total sequence, len : 800987

distance : 9.539E-1 answer fasta id ../database/database_5/GB_GCA_016719635.1_protein.faa
answer fasta id total sequence, seq len : 1064205
distance : 9.841E-1 answer fasta id ../database/database_5/GB_GCA_001567165.1_protein.faa
answer fasta id total sequence, seq len : 638956
distance : 9.842E-1 answer fasta id ../database/database_5/RS_GCF_900100625.1_protein.faa
answer fasta id total sequence, seq len : 872517
distance : 9.842E-1 answer fasta id ../database/database_5/GB_GCA_016722375.1_protein.faa
answer fasta id total sequence, seq len : 1037108
distance : 9.844E-1 answer fasta id ../database/database_5/GB_GCA_016787165.1_protein.faa
answer fasta id total sequence, seq len : 849407
distance : 9.844E-1 answer fasta id ../database/database_5/GB_GCA_002455375.1_protein.faa
answer fasta id total sequence, seq len : 948379
distance : 9.845E-1 answer fasta id ../database/database_5/GB_GCA_903883945.1_protein.faa
answer fasta id total sequence, seq len : 1157790
distance : 9.845E-1 answer fasta id ../database/database_5/GB_GCA_015655125.1_protein.faa
answer fasta id total sequence, seq len : 1100101
distance : 9.847E-1 answer fasta id ../database/database_5/GB_GCA_017495115.1_protein.faa
answer fasta id total sequence, seq len : 763790
distance : 9.847E-1 answer fasta id ../database/database_5/RS_GCF_002217405.1_protein.faa
answer fasta id total sequence, seq len : 770090

Top 10 from one large database

0 path ../GWMC_1000_HQ_faa_aai/BRRP3A.13.faa.gz, fasta_id total sequence, len : 800987

distance : 9.156E-1 answer fasta id ../protein_faa_reps_r207/all/GB_GCA_016699615.1_protein.faa
answer fasta id total sequence, seq len : 906076
distance : 9.452E-1 answer fasta id ../protein_faa_reps_r207/all/GB_GCA_017984135.1_protein.faa
answer fasta id total sequence, seq len : 797872
distance : 9.539E-1 answer fasta id ../protein_faa_reps_r207/all/GB_GCA_016719635.1_protein.faa
answer fasta id total sequence, seq len : 1064205
distance : 9.540E-1 answer fasta id ../protein_faa_reps_r207/all/GB_GCA_016721245.1_protein.faa
answer fasta id total sequence, seq len : 1111343
distance : 9.551E-1 answer fasta id ../protein_faa_reps_r207/all/GB_GCA_016711605.1_protein.faa
answer fasta id total sequence, seq len : 1152198
distance : 9.711E-1 answer fasta id ../protein_faa_reps_r207/all/GB_GCA_016705125.1_protein.faa
answer fasta id total sequence, seq len : 973431
distance : 9.822E-1 answer fasta id ../protein_faa_reps_r207/all/GB_GCA_001897535.1_protein.faa
answer fasta id total sequence, seq len : 957913
distance : 9.829E-1 answer fasta id ../protein_faa_reps_r207/all/GB_GCA_018262795.1_protein.faa
answer fasta id total sequence, seq len : 1041576
distance : 9.832E-1 answer fasta id ../protein_faa_reps_r207/all/GB_GCA_018267475.1_protein.faa
answer fasta id total sequence, seq len : 858052
distance : 9.835E-1 answer fasta id ../protein_faa_reps_r207/all/GB_GCA_018268055.1_protein.faa
answer fasta id total sequence, seq len : 974493

Supplementary File 3

Commands used for the pieces of software compared to GSearch

Sourmash (v4.2.3):
(1) create sketches:
find ./gtdb_v207/ -name '*.fna.gz' | parallel -j 24 "sourmash sketch dna -p k=16,noabund {} -o {.}.sourmash"

sourmash sketch dna -p k=16,noabund OceanDNA-b42278.fa -o OceanDNA-b42278.fa.sourmash
(2) Create SBT index from the database sketches in (1):
sourmash index -k 16 gtdb_sourmash_index --from-file gtdb_v207_sourmash_sketch.sig.txt -q

Note: here gtdb_v207_sourmash_sketch.sig.txt is a list of *.sourmash file from step (1)
(3) Search sketch against (2) SBT index:
sourmash search --threshold 0.01 --num-results 50 OceanDNA-b42278.fa.sourmash gtdb_sourmash_index.sbt.zip -o OceanDNA-b42278.sourmash.dist.txt

Note: here OceanDNA-b42278.fa.sourmash is obtained by similar commands in (1) but only for this one query genome file. This step takes about 15 minutes for just one query genome (only one thread can be used) in sourmash. We use GNU parallel to run 24 queries at a time.

find ./Tara_query/ -name '*.fna.sourmash' | parallel -j 24 “sourmash search --threshold 0.01 --num-results 50 {} gtdb_sourmash_index.sbt.zip -o {.}.sourmash.dist.txt”

Mash (v2.3):
(1) Create sketches:
For nucleotide:
mash sketch -p 24 -s 12000 -k 16 -o ./query_genome -l query_name.txt
mash sketch -p 24 -s 12000 -k 16 -o ./reference_genome -l reference_name.txt

For Amino Acid:
mash sketch -p 24 -s 12000 -k 7 -a -o ./query_genome_aa -l query_name_aa.txt
mash sketch -p 24 -s 12000 -k 7 -a -o ./reference_genome_aa -l reference_name_aa.txt
(2) Compute distance
mash dist -p 24 query_genome.msh reference_genome.msh > dist.txt

mash dist -p 24 query_genome_aa.msh reference_genome_aa.msh > dist_aa.txt
Dashing (v1.0.2-4-g0635): default HyperLogLog option with Ertl’s Joint MLE estimator (--ertl-jmle**) or default estimator**
(1) create sketches:
dashing sketch -k 16 --nthreads 24 -S 14 --suffix dashing_hll -F ./GTDB_nt_name.txt

dashing sketch -k 16 --nthreads 24 -S 14 --suffix dashing_hll -F ./Query_name.txt
Note: here GPD_test10000_nt_name.txt is a list of all bacteria database genomes, -S 14 (14*64) is similar to (or better) -s 12000 in Mash above
(2) Compute distance:
dashing dist -F ./GTDB_nt_dashing_sketch.txt -Q Query_nt_name.txt --nthreads 24 --presketched -O dist.txt
Note: here GTDB_nt_dashing_sketch.txt is a list of .hll sketches from step (1) while Query_nt_name.txt is the list of sketches of query genomes generated in similar way to (1).
Dashing 2 (version v2.1.11):
dashing2 dashing2 sketch -k 16 -S 12000 --threads 24 --pminhash -Q Tara_Ocean_MAGs_name.txt -F name.txt --cmpout Tara_Ocean_MAGs_Dashing2_pminhash_NCBI.txt

To make fair comparison with GSearch (default probminhash), we use the probminhash (--pminhash) option.
BinDash (v0.2.1):
bindash sketch --kmerlen=16 --outfname=Tara_bindash --sketchsize64=250 --listfname=Tara_Ocean_MAGs_name.txt --dens=1 --bbits=32 --minhashtype=2 --nthreads=24
bindash dist --nthreads=24 --outfname=Tara_refseq_dist.txt Tara_bindash refseq_ncbi_all_bindash_optdens

FastANI (v1.33):
fastANI --ql query_name.txt --rl reference_name.txt -t 24 -o ANI.txt

Note: query_name.txt and reference_name.txt are files storing genome path.

Supplementary File 4

query_genome_name	GSearch_taxonomy	query_genome_name	Sourmash_lca_taxonomy
AUBR2B.34.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium OLB8	AUBR2B.34.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium OLB8,"
AUBR2C.63.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium OLB8	AUBR2C.63.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium OLB8,"
AUCR2B.2.fasta.gz	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__Thiotrichales; f__Thiotrichaceae; g__Thiothrix; s__Thiothrix eikelboomii	AUCR2B.2.fasta.gz	"found,Bacteria,Proteobacteria,Gammaproteobacteria,Thiotrichales,Thiotrichaceae,Thiothrix,Thiothrix eikelboomii,"
AUEP2BC.56.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium ADurb.BinA245	AUEP2BC.56.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium ADurb.BinA245,"
AUXP4AB.87.fasta.gz	d__Bacteria; p__Proteobacteria; c__Deltaproteobacteria; o__Myxococcales; f__Anaeromyxobacteraceae; g__Anaeromyxobacter; s__unassigned	AUXP4AB.87.fasta.gz	"found,Bacteria,Proteobacteria,Deltaproteobacteria,Myxococcales,Anaeromyxobacteraceae,Anaeromyxobacter,unassigned,"
AUXP4AB.163.fasta.gz	d__Bacteria; p__Proteobacteria; c__Alphaproteobacteria; o__Rhizobiales; f__Rhizobiaceae; g__Kaistia; s__Kaistia sp. SCN 65-12	AUXP4AB.163.fasta.gz	"found,Bacteria,Proteobacteria,Alphaproteobacteria,Rhizobiales,Rhizobiaceae,Kaistia,Kaistia sp. SCN 65-12,"
AUXP4AB.12.fasta.gz	d__Bacteria; p__Actinobacteria; c__Actinobacteria; o__Corynebacteriales; f__Gordoniaceae; g__Skermania; s__Skermania piniformis	AUXP4AB.12.fasta.gz	"found,Bacteria,Actinobacteria,Actinobacteria,Corynebacteriales,Gordoniaceae,Skermania,Skermania piniformis,"
BRBH3C.63.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium ADurb.BinA245	BRBH3C.63.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium ADurb.BinA245,"
BRBH1C.68.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium ADurb.BinA245	BRBH1C.68.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium ADurb.BinA245,"
BRBH4C.97.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium ADurb.BinA245	BRBH4C.97.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium ADurb.BinA245,"
BROS2C.21.fasta.gz	d__Bacteria; p__Spirochaetes; c__Spirochaetia; o__Spirochaetales; f__Spirochaetaceae; g__Spirochaeta; s__Spirochaeta sp. EBM-43	BROS2C.21.fasta.gz	"found,Bacteria,Spirochaetes,Spirochaetia,Spirochaetales,Spirochaetaceae,Spirochaeta,Spirochaeta sp. EBM-43,"
BROS2C.42.fasta.gz	d__Bacteria; p__Proteobacteria; c__Deltaproteobacteria; o__Desulfobacterales; f__Desulfobulbaceae; g__Desulfobulbus; s__Desulfobulbus propionicus	BROS2C.42.fasta.gz	"found,Bacteria,Proteobacteria,Deltaproteobacteria,Desulfobacterales,Desulfobulbaceae,Desulfobulbus,Desulfobulbus propionicus,"
BRRP2B.69.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium OLB8	BRRP2B.69.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium OLB8,"
BRRP3A.53.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium OLB8	BRRP3A.53.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium OLB8,"
BRRP3A.37.fasta.gz	d__Bacteria; p__Chlorobi; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Chlorobi bacterium OLB7	BRRP3A.37.fasta.gz	"found,Bacteria,Chlorobi,unassigned,unassigned,unassigned,unassigned,Chlorobi bacterium OLB7,"
BRRP3A.47.fasta.gz	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__Xanthomonadales; f__unassigned; g__unassigned; s__unassigned	BRRP3A.47.fasta.gz	"found,Bacteria,Proteobacteria,Gammaproteobacteria,Xanthomonadales,unassigned,unassigned,unassigned,"
CLVPM1.1.fasta.gz	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__Thiotrichales; f__Thiotrichaceae; g__Thiothrix; s__Thiothrix eikelboomii	CLVPM1.1.fasta.gz	"found,Bacteria,Proteobacteria,Gammaproteobacteria,Thiotrichales,Thiotrichaceae,Thiothrix,Thiothrix eikelboomii,"
CLVPM1.2.fasta.gz	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__Thiotrichales; f__Thiotrichaceae; g__Thiothrix; s__Thiothrix eikelboomii	CLVPM1.2.fasta.gz	"found,Bacteria,Proteobacteria,Gammaproteobacteria,Thiotrichales,Thiotrichaceae,Thiothrix,Thiothrix eikelboomii,"
CLVPM2.2.fasta.gz	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__Thiotrichales; f__Thiotrichaceae; g__Thiothrix; s__Thiothrix eikelboomii	CLVPM2.2.fasta.gz	"found,Bacteria,Proteobacteria,Gammaproteobacteria,Thiotrichales,Thiotrichaceae,Thiothrix,Thiothrix eikelboomii,"
CLVPM3.2.fasta.gz	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__Thiotrichales; f__Thiotrichaceae; g__Thiothrix; s__Thiothrix eikelboomii	CLVPM3.2.fasta.gz	"found,Bacteria,Proteobacteria,Gammaproteobacteria,Thiotrichales,Thiotrichaceae,Thiothrix,Thiothrix eikelboomii,"
CLVPM3.64.fasta.gz	d__Bacteria; p__Proteobacteria; c__Alphaproteobacteria; o__unassigned; f__unassigned; g__unassigned; s__Alphaproteobacteria bacterium ADurb.BinA280	CLVPM3.64.fasta.gz	"found,Bacteria,Proteobacteria,Alphaproteobacteria,unassigned,unassigned,unassigned,Alphaproteobacteria bacterium ADurb.BinA280,"
CLVPT1.2.fasta.gz	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__Thiotrichales; f__Thiotrichaceae; g__Thiothrix; s__Thiothrix eikelboomii	CLVPT1.2.fasta.gz	"found,Bacteria,Proteobacteria,Gammaproteobacteria,Thiotrichales,Thiotrichaceae,Thiothrix,Thiothrix eikelboomii,"
CLVPT1.1.fasta.gz	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__Thiotrichales; f__Thiotrichaceae; g__Thiothrix; s__Thiothrix eikelboomii	CLVPT1.1.fasta.gz	"found,Bacteria,Proteobacteria,Gammaproteobacteria,Thiotrichales,Thiotrichaceae,Thiothrix,Thiothrix eikelboomii,"
CLVPT2.2.fasta.gz	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__Thiotrichales; f__Thiotrichaceae; g__Thiothrix; s__Thiothrix eikelboomii	CLVPT2.2.fasta.gz	"found,Bacteria,Proteobacteria,Gammaproteobacteria,Thiotrichales,Thiotrichaceae,Thiothrix,Thiothrix eikelboomii,"
CLVPM3.1.fasta.gz	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__Thiotrichales; f__Thiotrichaceae; g__Thiothrix; s__Thiothrix eikelboomii	CLVPM3.1.fasta.gz	"found,Bacteria,Proteobacteria,Gammaproteobacteria,Thiotrichales,Thiotrichaceae,Thiothrix,Thiothrix eikelboomii,"
CNBJ6BC.12.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium ADurb.BinA245	CNBJ6BC.12.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium ADurb.BinA245,"
CNCD4C.3.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium ADurb.Bin141	CNCD4C.3.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium ADurb.Bin141,"
CNCQ4C.33.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium ADurb.Bin141	CNCQ4C.33.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium ADurb.Bin141,"
CNDL1AC.13.fasta.gz	d__Bacteria; p__Actinobacteria; c__Actinobacteria; o__Corynebacteriales; f__Mycobacteriaceae; g__Mycobacterium; s__unassigned	CNDL1AC.13.fasta.gz	"found,Bacteria,Actinobacteria,Actinobacteria,Corynebacteriales,Mycobacteriaceae,Mycobacterium,unassigned,"
CNDL1BC.25.fasta.gz	d__Bacteria; p__Actinobacteria; c__Actinobacteria; o__Corynebacteriales; f__Mycobacteriaceae; g__Mycobacterium; s__unassigned	CNDL1BC.25.fasta.gz	"found,Bacteria,Actinobacteria,Actinobacteria,Corynebacteriales,Mycobacteriaceae,Mycobacterium,unassigned,"
CNDL2AC.49.fasta.gz	d__Bacteria; p__Bacteroidetes; c__Flavobacteriia; o__Flavobacteriales; f__Flavobacteriaceae; g__Flavobacterium; s__Flavobacterium cauense	CNDL2AC.49.fasta.gz	"found,Bacteria,Bacteroidetes,Flavobacteriia,Flavobacteriales,Flavobacteriaceae,Flavobacterium,Flavobacterium cauense,"
CNDL2AC.4.fasta.gz	d__Bacteria; p__Bacteroidetes; c__Sphingobacteriia; o__Sphingobacteriales; f__unassigned; g__unassigned; s__Sphingobacteriales bacterium 39-19	CNDL2AC.4.fasta.gz	"found,Bacteria,Bacteroidetes,Sphingobacteriia,Sphingobacteriales,unassigned,unassigned,Sphingobacteriales bacterium 39-19,"
CNJN1C.21.fasta.gz	d__Bacteria; p__Actinobacteria; c__Acidimicrobiia; o__Acidimicrobiales; f__Microthrixaceae; g__Candidatus Microthrix; s__Candidatus Microthrix parvicella	CNJN1C.21.fasta.gz	"found,Bacteria,Actinobacteria,Acidimicrobiia,Acidimicrobiales,Microthrixaceae,Candidatus Microthrix,Candidatus Microthrix parvicella,"
CNJN1C.22.fasta.gz	d__Bacteria; p__Proteobacteria; c__Alphaproteobacteria; o__Sphingomonadales; f__Sphingomonadaceae; g__unassigned; s__unassigned	CNJN1C.22.fasta.gz	"found,Bacteria,Proteobacteria,Alphaproteobacteria,Sphingomonadales,Sphingomonadaceae,unassigned,unassigned,"
CNJN2C.1.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium ADurb.Bin141	CNJN2C.1.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium ADurb.Bin141,"
CNJN4C.36.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium ADurb.Bin141	CNJN4C.36.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium ADurb.Bin141,"
CNWX2C.18.fasta.gz	d__Bacteria; p__Proteobacteria; c__Alphaproteobacteria; o__Rhodobacterales; f__Rhodobacteraceae; g__unassigned; s__unassigned	CNWX2C.18.fasta.gz	"found,Bacteria,Proteobacteria,Alphaproteobacteria,Rhodobacterales,Rhodobacteraceae,unassigned,unassigned,"
CNXA3C.36.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium ADurb.BinA245	CNXA3C.36.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium ADurb.BinA245,"
CNXM4C.17.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium OLB8	CNXM4C.17.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium OLB8,"
CNXM3C.28.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium OLB8	CNXM3C.28.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium OLB8,"
COCL1B.006_sub.fasta.gz	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__unassigned; f__unassigned; g__unassigned; s__unassigned	COCL1B.006_sub.fasta.gz	"found,Bacteria,Proteobacteria,Gammaproteobacteria,unassigned,unassigned,unassigned,unassigned,"
COCL1B.11.fasta.gz	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__Xanthomonadales; f__unassigned; g__unassigned; s__unassigned	COCL1B.11.fasta.gz	"found,Bacteria,Proteobacteria,Gammaproteobacteria,Xanthomonadales,unassigned,unassigned,unassigned,"
COCL1B.33.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__Burkholderiales; f__unassigned; g__unassigned; s__unassigned	COCL1B.33.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,Burkholderiales,unassigned,unassigned,unassigned,"
COCL2B.15.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium ADurb.BinA395	COCL2B.15.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium ADurb.BinA395,"
COCL2B.60.fasta.gz	d__Bacteria; p__Proteobacteria; c__Deltaproteobacteria; o__Desulfobacterales; f__Desulfobulbaceae; g__Desulfobulbus; s__Desulfobulbus propionicus	COCL2B.60.fasta.gz	"found,Bacteria,Proteobacteria,Deltaproteobacteria,Desulfobacterales,Desulfobulbaceae,Desulfobulbus,Desulfobulbus propionicus,"
COCL3C.005_sub.fasta.gz	d__Bacteria; p__Proteobacteria; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__unassigned	COCL3C.005_sub.fasta.gz	"found,Bacteria,Proteobacteria,unassigned,unassigned,unassigned,unassigned,unassigned,"
COCL3C.12.fasta.gz	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__Xanthomonadales; f__unassigned; g__unassigned; s__unassigned	COCL3C.12.fasta.gz	"found,Bacteria,Proteobacteria,Gammaproteobacteria,Xanthomonadales,unassigned,unassigned,unassigned,"
COCL3C.3.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__Burkholderiales; f__unassigned; g__unassigned; s__unassigned	COCL3C.3.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,Burkholderiales,unassigned,unassigned,unassigned,"
DEHE1C.2.fasta.gz	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__Thiotrichales; f__Thiotrichaceae; g__Thiothrix; s__Thiothrix eikelboomii	DEHE1C.2.fasta.gz	"found,Bacteria,Proteobacteria,Gammaproteobacteria,Thiotrichales,Thiotrichaceae,Thiothrix,Thiothrix eikelboomii,"
COCL3C.13.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__Burkholderiales; f__unassigned; g__unassigned; s__unassigned	COCL3C.13.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,Burkholderiales,unassigned,unassigned,unassigned,"
DERA1C.41.fasta.gz	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__unassigned; f__Competibacteraceae; g__Candidatus Contendobacter; s__Candidatus Contendobacter odensis	DERA1C.41.fasta.gz	"found,Bacteria,Proteobacteria,Gammaproteobacteria,unassigned,Competibacteraceae,Candidatus Contendobacter,Candidatus Contendobacter odensis,"
ITLF2C.7.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__unassigned; f__unassigned; g__unassigned; s__unassigned	ITLF2C.7.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,unassigned,unassigned,unassigned,unassigned,"
NZBL2.101.fasta.gz	d__Bacteria; p__Bacteroidetes; c__Bacteroidia; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidia bacterium Ga0077558	NZBL2.101.fasta.gz	"found,Bacteria,Bacteroidetes,Bacteroidia,unassigned,unassigned,unassigned,Bacteroidia bacterium Ga0077558,"
ITLF2B.6.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__unassigned; f__unassigned; g__unassigned; s__unassigned	ITLF2B.6.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,unassigned,unassigned,unassigned,unassigned,"
ITMC3C.16.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__unassigned; f__unassigned; g__unassigned; s__unassigned	ITMC3C.16.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,unassigned,unassigned,unassigned,unassigned,"
ITRB1C.37.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__Rhodocyclales; f__Rhodocyclaceae; g__Thauera; s__Thauera phenylacetica	ITRB1C.37.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,Rhodocyclales,Rhodocyclaceae,Thauera,Thauera phenylacetica,"
ITRB1C.24.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium ADurb.Bin141	ITRB1C.24.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium ADurb.Bin141,"
NZDM2.40.fasta.gz	d__Bacteria; p__Proteobacteria; c__Alphaproteobacteria; o__unassigned; f__unassigned; g__unassigned; s__unassigned	NZDM2.40.fasta.gz	"found,Bacteria,Proteobacteria,Alphaproteobacteria,unassigned,unassigned,unassigned,unassigned,"
NZDM2.131.fasta.gz	d__Bacteria; p__Proteobacteria; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__unassigned	NZDM2.131.fasta.gz	"found,Bacteria,Proteobacteria,unassigned,unassigned,unassigned,unassigned,unassigned,"
SADP3.8.fasta.gz	d__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__unassigned; f__Competibacteraceae; g__Candidatus Competibacter; s__Candidatus Competibacter denitrificans	SADP3.8.fasta.gz	"found,Bacteria,Proteobacteria,Gammaproteobacteria,unassigned,Competibacteraceae,Candidatus Competibacter,Candidatus Competibacter denitrificans,"
SAKB5.19.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__Rhodocyclales; f__Rhodocyclaceae; g__unassigned; s__Rhodocyclaceae bacterium UTPRO2	SAKB5.19.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,Rhodocyclales,Rhodocyclaceae,unassigned,Rhodocyclaceae bacterium UTPRO2,"
SAKR2.109.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__unassigned; f__unassigned; g__Candidatus Accumulibacter; s__Candidatus Accumulibacter sp. BA-91	SAKR2.109.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,unassigned,unassigned,Candidatus Accumulibacter,Candidatus Accumulibacter sp. BA-91,"
SAZK3.32.fasta.gz	d__Bacteria; p__Proteobacteria; c__Deltaproteobacteria; o__unassigned; f__unassigned; g__unassigned; s__Deltaproteobacteria bacterium Ga0077550	SAZK3.32.fasta.gz	"found,Bacteria,Proteobacteria,Deltaproteobacteria,unassigned,unassigned,unassigned,Deltaproteobacteria bacterium Ga0077550,"
SEGL2C.1.fasta.gz	d__Bacteria; p__Actinobacteria; c__Acidimicrobiia; o__Acidimicrobiales; f__Microthrixaceae; g__Candidatus Microthrix; s__Candidatus Microthrix parvicella	SEGL2C.1.fasta.gz	"found,Bacteria,Actinobacteria,Acidimicrobiia,Acidimicrobiales,Microthrixaceae,Candidatus Microthrix,Candidatus Microthrix parvicella,"
SESD1C.50.fasta.gz	d__Bacteria; p__Actinobacteria; c__Acidimicrobiia; o__Acidimicrobiales; f__Microthrixaceae; g__Candidatus Microthrix; s__Candidatus Microthrix parvicella	SESD1C.50.fasta.gz	"found,Bacteria,Actinobacteria,Acidimicrobiia,Acidimicrobiales,Microthrixaceae,Candidatus Microthrix,Candidatus Microthrix parvicella,"
SESD2C.72.fasta.gz	d__Bacteria; p__Proteobacteria; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__unassigned	SESD2C.72.fasta.gz	"found,Bacteria,Proteobacteria,unassigned,unassigned,unassigned,unassigned,unassigned,"
SGEA1C.73.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium OLB8	SGEA1C.73.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium OLB8,"
SGWE1C.37.fasta.gz	d__Bacteria; p__Cyanobacteria; c__unassigned; o__Nostocales; f__Tolypothrichaceae; g__Hassallia; s__Hassallia byssoidea	SGWE1C.37.fasta.gz	"found,Bacteria,Cyanobacteria,unassigned,Nostocales,Tolypothrichaceae,Hassallia,Hassallia byssoidea,"
SGWE1C.14.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium OLB8	SGWE1C.14.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium OLB8,"
SGJR1C.8.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium OLB8	SGJR1C.8.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium OLB8,"
TWKS2C.44.fasta.gz	d__Bacteria; p__Proteobacteria; c__Alphaproteobacteria; o__Caulobacterales; f__Caulobacteraceae; g__Phenylobacterium; s__unassigned	TWKS2C.44.fasta.gz	"found,Bacteria,Proteobacteria,Alphaproteobacteria,Caulobacterales,Caulobacteraceae,Phenylobacterium,unassigned,"
TWKS2C.51.fasta.gz	d__Bacteria; p__unassigned; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__unassigned	TWKS2C.51.fasta.gz	"found,Bacteria,unassigned,unassigned,unassigned,unassigned,unassigned,unassigned,"
TWKS2C.7.fasta.gz	d__Bacteria; p__Actinobacteria; c__Actinobacteria; o__Corynebacteriales; f__Mycobacteriaceae; g__Mycobacterium; s__unassigned	TWKS2C.7.fasta.gz	"found,Bacteria,Actinobacteria,Actinobacteria,Corynebacteriales,Mycobacteriaceae,Mycobacterium,unassigned,"
TWTN1C.66.fasta.gz	d__Bacteria; p__Actinobacteria; c__Actinobacteria; o__Corynebacteriales; f__Mycobacteriaceae; g__Mycobacterium; s__unassigned	TWTN1C.66.fasta.gz	"found,Bacteria,Actinobacteria,Actinobacteria,Corynebacteriales,Mycobacteriaceae,Mycobacterium,unassigned,"
TWTN1C.22.fasta.gz	d__Bacteria; p__Proteobacteria; c__Alphaproteobacteria; o__Caulobacterales; f__Caulobacteraceae; g__Phenylobacterium; s__unassigned	TWTN1C.22.fasta.gz	"found,Bacteria,Proteobacteria,Alphaproteobacteria,Caulobacterales,Caulobacteraceae,Phenylobacterium,unassigned,"
TWTN2C.25.fasta.gz	d__Bacteria; p__Proteobacteria; c__Alphaproteobacteria; o__Sphingomonadales; f__Sphingomonadaceae; g__unassigned; s__	TWTN2C.25.fasta.gz	"found,Bacteria,Proteobacteria,Alphaproteobacteria,Sphingomonadales,Sphingomonadaceae,unassigned,"
TWTN2C.2.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium ADurb.Bin141	TWTN2C.2.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium ADurb.Bin141,"
USAG1C.3.fasta.gz	d__Bacteria; p__Proteobacteria; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__unassigned	USAG1C.3.fasta.gz	"found,Bacteria,Proteobacteria,unassigned,unassigned,unassigned,unassigned,unassigned,"
USAG1C.85.fasta.gz	d__Bacteria; p__Actinobacteria; c__Actinobacteria; o__Corynebacteriales; f__Mycobacteriaceae; g__Mycobacterium; s__unassigned	USAG1C.85.fasta.gz	"found,Bacteria,Actinobacteria,Actinobacteria,Corynebacteriales,Mycobacteriaceae,Mycobacterium,unassigned,"
USBT2C.006_sub.fasta.gz	d__Bacteria; p__Nitrospirae; c__Nitrospira; o__Nitrospirales; f__Nitrospiraceae; g__Nitrospira; s__Candidatus Nitrospira nitrosa	USBT2C.006_sub.fasta.gz	"found,Bacteria,Nitrospirae,Nitrospira,Nitrospirales,Nitrospiraceae,Nitrospira,Candidatus Nitrospira nitrosa,"
USCB1AC.10.fasta.gz	d__Bacteria; p__Proteobacteria; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__unassigned	USCB1AC.10.fasta.gz	"found,Bacteria,Proteobacteria,unassigned,unassigned,unassigned,unassigned,unassigned,"
USCB1DC.10.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium ADurb.BinA245	USCB1DC.10.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium ADurb.BinA245,"
USCB1CC.21.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__Rhodocyclales; f__Rhodocyclaceae; g__Thauera; s__Thauera selenatis	USCB1CC.21.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,Rhodocyclales,Rhodocyclaceae,Thauera,Thauera selenatis,"
USFT4C.26.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__unassigned; f__unassigned; g__unassigned; s__unassigned	USFT4C.26.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,unassigned,unassigned,unassigned,unassigned,"
USHI1C.30.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__Burkholderiales; f__unassigned; g__unassigned; s__unassigned	USHI1C.30.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,Burkholderiales,unassigned,unassigned,unassigned,"
USHI4C.42.fasta.gz	d__Bacteria; p__Proteobacteria; c__Alphaproteobacteria; o__unassigned; f__unassigned; g__unassigned; s__unassigned	USHI4C.42.fasta.gz	"found,Bacteria,Proteobacteria,Alphaproteobacteria,unassigned,unassigned,unassigned,unassigned,"
USKP1O.64.fasta.gz	d__Bacteria; p__Proteobacteria; c__Alphaproteobacteria; o__unassigned; f__unassigned; g__unassigned; s__unassigned	USKP1O.64.fasta.gz	"found,Bacteria,Proteobacteria,Alphaproteobacteria,unassigned,unassigned,unassigned,unassigned,"
USKP1O.71.fasta.gz	d__Bacteria; p__Proteobacteria; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__unassigned	USKP1O.71.fasta.gz	"found,Bacteria,Proteobacteria,unassigned,unassigned,unassigned,unassigned,unassigned,"
USKP2O.22.fasta.gz	d__Bacteria; p__Proteobacteria; c__Alphaproteobacteria; o__unassigned; f__unassigned; g__unassigned; s__unassigned	USKP2O.22.fasta.gz	"found,Bacteria,Proteobacteria,Alphaproteobacteria,unassigned,unassigned,unassigned,unassigned,"
USMI1C.53.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__Burkholderiales; f__unassigned; g__unassigned; s__unassigned	USMI1C.53.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,Burkholderiales,unassigned,unassigned,unassigned,"
USMD4C.62_sub.fasta.gz	d__Bacteria; p__Proteobacteria; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__unassigned	USMD4C.62_sub.fasta.gz	"found,Bacteria,Proteobacteria,unassigned,unassigned,unassigned,unassigned,unassigned,"
USMD4C.17.fasta.gz	d__Bacteria; p__Actinobacteria; c__Actinobacteria; o__Corynebacteriales; f__Mycobacteriaceae; g__Mycobacterium; s__unassigned	USMD4C.17.fasta.gz	"found,Bacteria,Actinobacteria,Actinobacteria,Corynebacteriales,Mycobacteriaceae,Mycobacterium,unassigned,"
USNO1C.50.fasta.gz	d__Bacteria; p__Actinobacteria; c__Actinobacteria; o__Corynebacteriales; f__unassigned; g__unassigned; s__unassigned	USNO1C.50.fasta.gz	"found,Bacteria,Actinobacteria,Actinobacteria,Corynebacteriales,unassigned,unassigned,unassigned,"
USOP1AC.36.fasta.gz	d__Bacteria; p__Proteobacteria; c__Alphaproteobacteria; o__unassigned; f__unassigned; g__unassigned; s__unassigned	USOP1AC.36.fasta.gz	"found,Bacteria,Proteobacteria,Alphaproteobacteria,unassigned,unassigned,unassigned,unassigned,"
USOP1CC.12.fasta.gz	d__Bacteria; p__Proteobacteria; c__Alphaproteobacteria; o__Sphingomonadales; f__Sphingomonadaceae; g__Sphingomonas; s__unassigned	USOP1CC.12.fasta.gz	"found,Bacteria,Proteobacteria,Alphaproteobacteria,Sphingomonadales,Sphingomonadaceae,Sphingomonas,unassigned,"
USPT2C.38.fasta.gz	d__Bacteria; p__Actinobacteria; c__Acidimicrobiia; o__Acidimicrobiales; f__Microthrixaceae; g__Candidatus Microthrix; s__Candidatus Microthrix parvicella	USPT2C.38.fasta.gz	"found,Bacteria,Actinobacteria,Acidimicrobiia,Acidimicrobiales,Microthrixaceae,Candidatus Microthrix,Candidatus Microthrix parvicella,"
USRE3C.8.fasta.gz	d__Bacteria; p__Proteobacteria; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__unassigned	USRE3C.8.fasta.gz	"found,Bacteria,Proteobacteria,unassigned,unassigned,unassigned,unassigned,unassigned,"
USRE8C.18.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__unassigned; f__unassigned; g__unassigned; s__unassigned	USRE8C.18.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,unassigned,unassigned,unassigned,unassigned,"
USSD2BB.2.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__Burkholderiales; f__unassigned; g__unassigned; s__unassigned	USSD2BB.2.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,Burkholderiales,unassigned,unassigned,unassigned,"
USTE3B.12.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__Burkholderiales; f__unassigned; g__unassigned; s__unassigned	USTE3B.12.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,Burkholderiales,unassigned,unassigned,unassigned,"
USSF4C.36.fasta.gz	d__Bacteria; p__Nitrospirae; c__Nitrospira; o__Nitrospirales; f__Nitrospiraceae; g__Nitrospira; s__Candidatus Nitrospira nitrosa	USSF4C.36.fasta.gz	"found,Bacteria,Nitrospirae,Nitrospira,Nitrospirales,Nitrospiraceae,Nitrospira,Candidatus Nitrospira nitrosa,"
USSF3C.36.fasta.gz	d__Bacteria; p__Actinobacteria; c__Actinobacteria; o__Corynebacteriales; f__Gordoniaceae; g__Gordonia; s__Gordonia amarae	USSF3C.36.fasta.gz	"found,Bacteria,Actinobacteria,Actinobacteria,Corynebacteriales,Gordoniaceae,Gordonia,Gordonia amarae,"
USTE3C.50.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__unassigned	USTE3C.50.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,unassigned,"
USTE3B.7.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__unassigned	USTE3B.7.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,unassigned,"
USTE3A.7.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__unassigned	USTE3A.7.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,unassigned,"
USVA2C.2.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__unassigned; f__unassigned; g__unassigned; s__unassigned	USVA2C.2.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,unassigned,unassigned,unassigned,unassigned,"
USVA6C.47.fasta.gz	d__Bacteria; p__Bacteroidetes; c__unassigned; o__unassigned; f__unassigned; g__unassigned; s__Bacteroidetes bacterium OLB8	USVA6C.47.fasta.gz	"found,Bacteria,Bacteroidetes,unassigned,unassigned,unassigned,unassigned,Bacteroidetes bacterium OLB8,"
USVA9C.1.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__Rhodocyclales; f__Rhodocyclaceae; g__Azonexus; s__Azonexus hydrophilus	USVA9C.1.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,Rhodocyclales,Rhodocyclaceae,Azonexus,Azonexus hydrophilus,"
USVD1DC.47_sub.fasta.gz	d__Bacteria; p__Cyanobacteria; c__unassigned; o__Nostocales; f__Tolypothrichaceae; g__Hassallia; s__Hassallia byssoidea	USVD1DC.47_sub.fasta.gz	"found,Bacteria,Cyanobacteria,unassigned,Nostocales,Tolypothrichaceae,Hassallia,Hassallia byssoidea,"
USWR1AC.27.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__Burkholderiales; f__unassigned; g__unassigned; s__unassigned	USWR1AC.27.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,Burkholderiales,unassigned,unassigned,unassigned,"
USWR1BC.53.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__Burkholderiales; f__unassigned; g__unassigned; s__unassigned	USWR1BC.53.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,Burkholderiales,unassigned,unassigned,unassigned,"
USWR1AC.37.fasta.gz	d__Bacteria; p__Proteobacteria; c__Betaproteobacteria; o__Burkholderiales; f__unassigned; g__unassigned; s__unassigned	USWR1AC.37.fasta.gz	"found,Bacteria,Proteobacteria,Betaproteobacteria,Burkholderiales,unassigned,unassigned,unassigned,"