Supplementary Table 1. DIDA algorithm pseudo code.

Input: Set T of target sequences, set Q of query sequences
procedure DISTRIBUTE
    $comp \leftarrow$ dfs(T) ▷ identifying connected components
    $comp_{sorted} \leftarrow$ qsort($comp$) ▷ sorting connected components
    $P \leftarrow$ best-fit-decreasing($comp_{sorted}$) ▷ partitioning sorted connected components
procedure INDEX
    for all $p \in P$ **in parallel do**
        $index_p \leftarrow$ build-index(p) ▷ constructing index for each partition
        store-index($index_p$) ▷ storing index for alignment step
procedure DISPATCH
    for all $p \in P$ **in parallel do** ▷ loading Bloom filter for each partition
        for all $t \in p$ **do**
            for all b-mer $\in t$ **do**
                insert(b-mer,$BF[p]$)
    for all $q \in Q$ **do** ▷ *flowing* queries through partitions
        for all $p \in P$ **in parallel do**
            if contain(b-mer $\in q$,$BF[p]$) **then**
                dispatch(q,$node_p$)
procedure ALIGN
    for all $p \in P$ **in parallel do** ▷ aligning queries against targets on all partitions
        receive($q, node_p$)
        $s \leftarrow$ align($q, index_p$)
        send($s, node_{merger}$)
procedure MERGE
    while receive($s, node_i$) **in parallel do** ▷ merging results from all partitions
        insert(s, priority-queue)
        $s \leftarrow$ priority-queue.pop()
        write($s, samFile$)
Output: File $samFile$