Appendix B. Algorithms

Algorithm 2. *bit-masked k-differences matching algorithm with quality values aware*[1]

Preprocessing

```
 1: for a in [A,C,G,T,N] do
 2:    calculate a bit vector misBits[a] in comparison with P
 3: end for
```

Alignment

```
 4: legalBits = 0
 5: penalK = k × (($P_{min}$ + $P_{max}$) / 2)
 6: for i=1 to $\lfloor penalK/delta \rfloor$ do
 7:    Q.pushBack(idx=-i,dif=delta×i)
 8:    legalBits = (legalBits << 1) | 1
 9: end for
10: for j=0 to n-1 do
11:    d = ($P_0 \neq S_j$)? qual2penalty($qv_j$):0
12:    Q.pushFront(idx=j,dif=d)
13:    legalBits = (legalBits << 1) | 1
14:    for i=1 to Q.size()-1 && (legalBits(i) & misBits[$S_j$](i)) do
15:       Q[i].dif = Q[i].dif + (qual2penalty($qv_j$) - delta)
16:       $q = argmin_{r=i-1}^{i+1}(Q[r].dif)$
          {r≠i+1 for the last iteration}
17:       Q[i].dif = Q[q].dif + delta
18:       Q[i].idx = Q[q].idx
19:       if Q[i].dif > penalK then
20:          legalBits &= ~(1 << i)
21:       end if
22:    end for
23:    while Q.back().dif > penalK do
24:       Q.popBack()
25:    end while
26:    if Q.size() == m then
27:       report Q.back().idx;
28:       Q.popBack()
29:    end if
30: end for
31: function qual2penalty(qual)
32:    return $P_{min}$ + qual/$quality_{max}$ × ($P_{max}$ - $P_{min}$)
33: end
```

[1] P_{min}: minimum penalty for a mismatch; P_{max}: maximum penalty for a mismatch; delta=P_{max}: penalty for an indel; $quality_{max}$=40: maximum possible quality value; *qv*: quality sequence corresponding to *S*