Appendix B. Algorithms

Algorithm 2: bit-masked k-differences matching algorithm with quality values aware

Preprocessing
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 [penalK/delta] 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=argmin_{r\neq i+1}(Q[r].dif)
16: Q[i].dif = Q[i].dif + (qual2penalty(qv_j) - delta)
17: Q[i].idx = Q[q].idx
18: if Q[i].dif > penalK then
19: legalBits &= ~(1 << i)
20: end if
21: end for
22: while Q.back().dif > penalK do
23: Q.popBack()
24: end while
25: if Q.size() == m then
26: report Q.back().idx;
27: Q.popBack()
28: end if
29: end for
30: function qual2penalty(qual)
31: return P_{min} + qual/quality_{max} × (P_{max} - P_{min})
32: 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