

Sample title

Anonymous

Overleaf

2021

Seed extension

Algorithm Standard LIS construction $O(n \log n)$

Require: $n \geq 0$

$lis_len \leftarrow 0$

▷Length of LIS

$parent \leftarrow \{\infty, \infty, \infty, \dots, \infty\}_{n+1}$

▷Mapping to reconstruct LIS

$sub \leftarrow \{\infty, \infty, \infty, \dots, \infty\}_{n+1}$

▷Array with indices for matches that form LIS

$i \leftarrow 0$

while $i < n$ **do**

▷Iterate over all elements $i = 0, 1, 2, \dots, n - 1$

$start \leftarrow 1$

$end \leftarrow lis_len$

while $start \leq end$ **do**

 ▷Binary search over existing longest sequence

$middle \leftarrow \left\lfloor \frac{start+end}{2} \right\rfloor$

if $matches_q[sub[middle]] < matches_q[i]$ **then**

$start \leftarrow middle + 1$

else

$start \leftarrow middle - 1$

$parent[i] \leftarrow sub[start - 1]$

▷We pin current value to the found parent

$sub[start] \leftarrow i$

if $start > lis_len$ **then**

$lis_len = start$

$i \leftarrow i + 1$

Seed extension

Algorithm Reconstruct LIS by following parent array $O(n)$

$current_node \leftarrow sub[lis_len]$

$result \leftarrow \{0, 0, 0, \dots, 0\}_{lis_len}$

$result[lis_len - 1] \leftarrow current_node$ \triangleright Will contain all indices from matches describing the output subsequence

$j \leftarrow lis_len - 1$

while $1 \leq j$ **do**

$current_node \leftarrow parent[current_node]$

$result[j - 1] \leftarrow current_node$

$j \leftarrow j - 1$

Seed extension

Algorithm Segmented-LIS heuristic $O(n \log n)$

Require: $n \geq 0$

$lis_len \leftarrow 0$

▷ Length of LIS

$parent \leftarrow \{\infty, \infty, \infty, \dots, \infty\}_{n+1}$

▷ Mapping to reconstruct LIS

$sub \leftarrow \{\infty, \infty, \infty, \dots, \infty\}_{n+1}$

▷ Array with indices for matches that form LIS

$i \leftarrow 0$

while $i < n$ **do**

▷ Iterate over all elements $i = 0, 1, 2, \dots, n - 1$

$start \leftarrow 1$

$end \leftarrow lis_len$

while $start \leq end$ **do**

▷ Binary search-like

$middle \leftarrow \lfloor \frac{start+end}{2} \rfloor$

if $matches_T[sub[middle]] > matches_T[i] - max_diff$ **then**

▷ Encountered old entry

$end \leftarrow start - 1$

▷ Breaks loop

else if $matches_Q[sub[middle]] < matches_Q[i]$ **then**

$start \leftarrow middle + 1$

else

$start \leftarrow middle - 1$

$parent[i] \leftarrow sub[start - 1]$

▷ We pin current value to the found parent

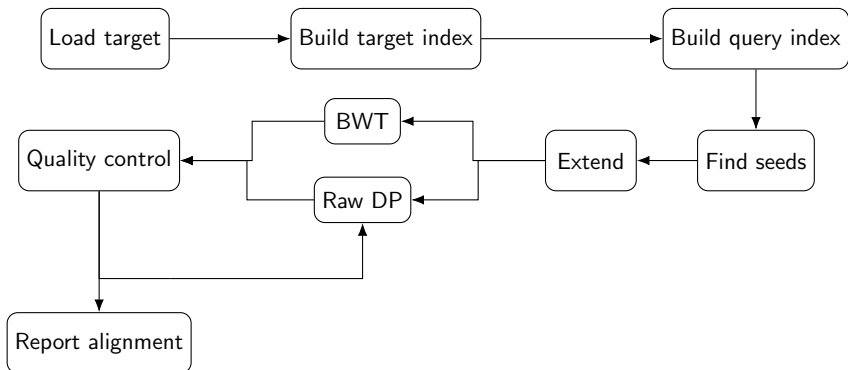
$sub[start] \leftarrow i$

if $start > lis_len$ **then**

$lis_len = start$

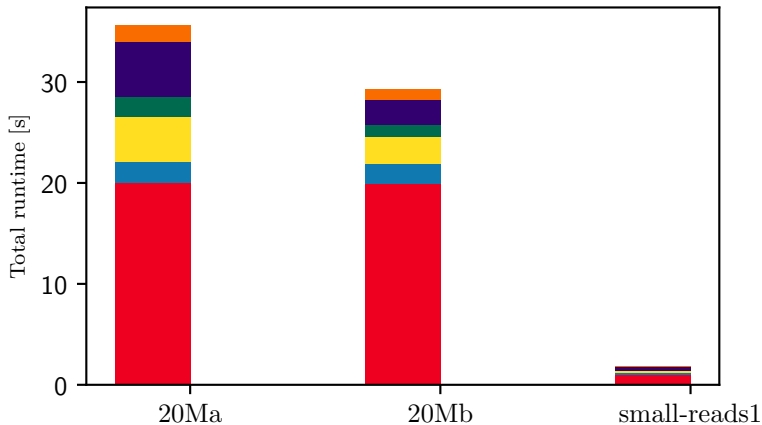
$i \leftarrow i + 1$

Seed and Extend approach



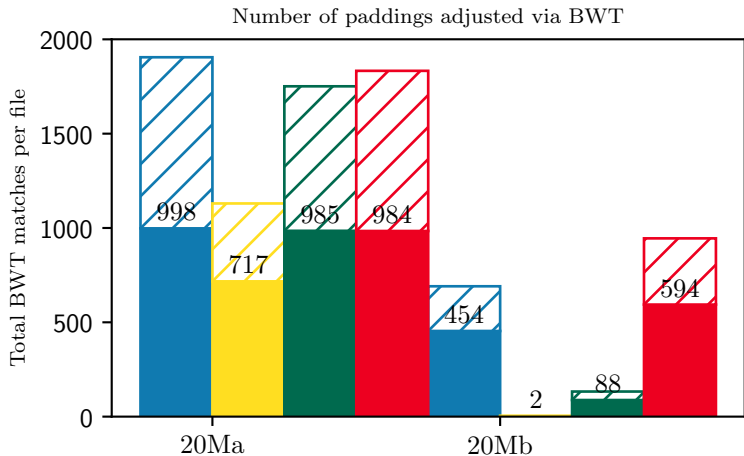
Execution times

Total runtime by category



- solution: Index target
- solution: Index query
- solution: Read LIS
- solution: Find best LIS region
- solution: BWT Alignment
- solution: DP Alignment

Aligner routine effectiveness



BWT routine implementation

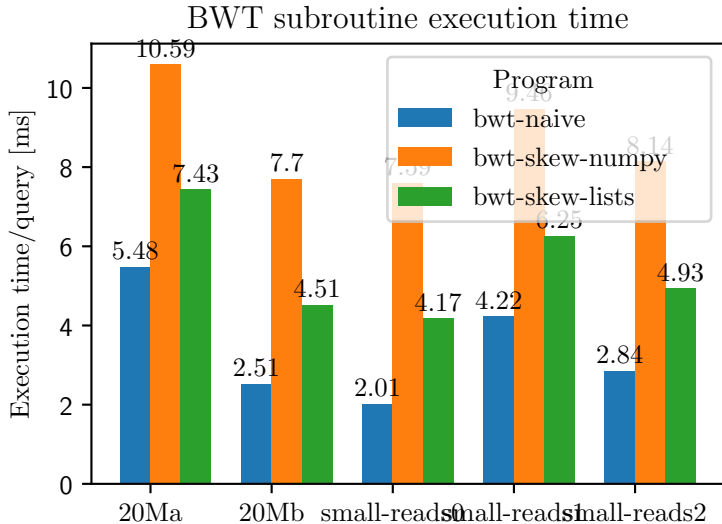


Figure: Test

Raw DP routine implementation

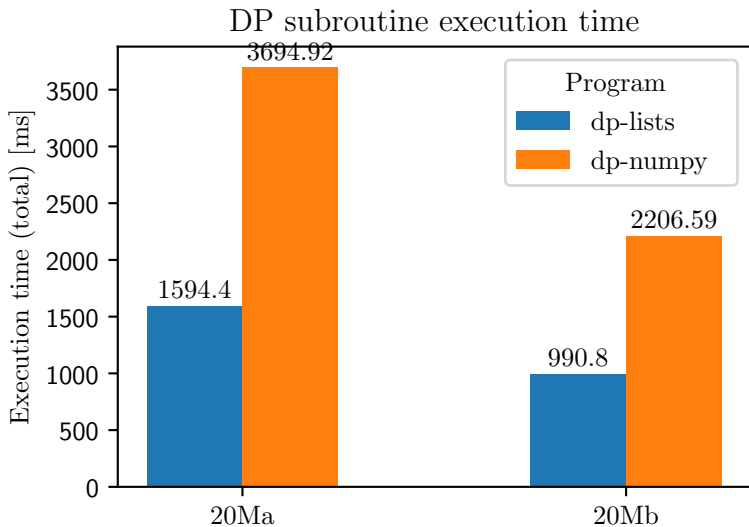


Figure: Test

Sample frame title

$$T = (t_1, t_2, \dots, t_m)$$

