Assignment -4 : Color Blindness

Harsh Gupta (Sr. No : 20961)

Oct 9, 2022

Problem Statement: Find Best Configuration for given Reference

Here, we have used the pattern matching using BWT(Burrows-Wheeler Transform)

Input:

- Last Column for Reference Sequence for BWT
- Contains mapping of indexes in BWT with index in reference Sequence
- Reads that we have to match to corresponding location in Reference Sequence

Functions:

There are mainly three functions to be performed which are as follows:

- ***MatchRead***:

  This function uses **RankMatrix** ,**firstOccurence** and read as input to match the pattern.
  FirstOcccurence is the first location of symbol in first Column
  Basically, for **RankMatrix** can do three things:

  a) We can search for the ith place of the symbol in the final Column using linear search instead of RankMatrix, however that computationally expensive. Space is optimised in this situation, but time will increase significantly.
  b) Has the ability to construct the Rank Matrix for each symbol's location in the last column. In contrast to the previous situation, the space need is considerable and the time complexity is low in this case.
  c) We are able to create a milestone matrix where we can apply linear search to fill in a little gap after counting at various checkpoints..It optimize both space and time and represent a kind of space-time trade off

  Here , I have used 2nd approach ,as it takes lesser time, but we can do any of them according to our requirements.

  ### ***Algorithm***:

MatchReadToLoc(read):

top =0,bottom= len(LastCol)-1,mismatch =0

positions = ***matchRead****(read,top,bottom,mismatch)*

positions+= ***matchRead****(rev_comp_read,top,bottom, mismatch)*

return positions

matchRead(read,top,bottom,mismatch):

while(top<=bottom):

IF Read is nonEmpty:

symbol = Read[-1]

remove last Letter from Read

IF *CheckLetter match any location b/w top and bottom*

Update top and bottom

Else*:*

If mismatch==2: return []

Else:

Mismatch++

matchRead*(read,top,bottom,other_symbol_in_b/w)*

ELSE:

return list of item in range(top+1,bottom+1)

//If no match after two mismatch too:

Return []

- ***WhichExon:***
  a) In this function, we have to match the corresponding location of matching read that it lies in given range for red exon and green exon
  b) If single read match more than one exon than we have to update it by ½ for both exon.
  c) ***Result***:

[R0 R1 R2 R3 R4 R5 G0 G1 G2 G3 G4 G5]

[74.5 66.5 55. 127. 243. 198.5 74.5 210.5 110. 112. 310. 198.5]

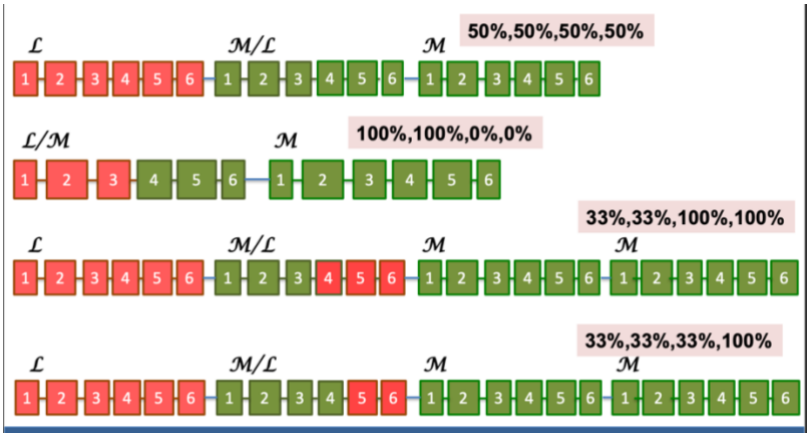
- ***Compute probability:***

In this , we have used binomial probability distribution for each exon to calculate probability for 2,3,4,5 exon of red and green exon correspondingly.

```
P2 = RExons[2]/GExons[2]
P3 = RExons[3]/GExons[3]
P4 = RExons[4]/GExons[4]
P5 = RExons[5]/GExons[5]
```

Probability [P0,P1,P2,P3]:
[0.3159144893111639, 0.5, 1.1339285714285714, 0.7838709677419354]

- ***Best Match:***
  a) It matches the best Configuration from given configurations
  b) For that , I have used ecludian distance of given configurations probability distance to find best match
  c) For best match, I have taken the configuration which have minimum distance from computed probability



Result: ***Configuration 2 is best Match***