Data Analytics:Assignment-5

Kishalay Das (SR Number-15938)

December 11, 2019

Color Blindness Detection :

Task 1: Problem Statement:

We are give the following:

1. Approx 3 million genome reads.
2. The reference sequence of chromosome X.
3. The BWT last column and the pointers back to the reference for chromosome X.
4. The locations of the exons of the red and green genes in chromosome X.

Our goal is to align the reads to the reference sequence with up to two mismatches, and then count reads mapping to exons of the red and green genes, counting 1 for each read that unambiguously maps to one of the two genes, and 1/2 for each gene for a read that maps ambiguously.Also For each of the possible red-green gene configurations, determine the probability of generating these counts given that configuration, and determine the most likely configuration that leads to colour-blindness.

1.A. Methodology :

I followed the approach covered in the class.We are provided a reference sequence along with the reads.Also we are given the last column data of BWT as well as its mapping with the rows of actual reference string.
Firstly create four binary strings from BWT last column data for each of A,C,G,T characters.Let size of BWT last column data is n.Then each binary string length will be n.So overall storage requirement for these 4 binary strings would be 4n bits.
Using these strings, we have taken some Δ window range and store cumulative rank count in a separate integer array for each of the above characters.That is we stored total count of characters in binary string upto that window.So the storage requirement of each of such array will be $\frac{n}{\Delta}$.Hence total it is $\frac{4n}{\Delta}$.So finding rank for a character given a particular index will be $\mathcal{O}(\Delta)$.The Δ parameter balances the time-space trade-off in this problem.
Now given a particular read in read file,we have to find recursively a range of indices in BWT column(one start/top indices and other end/bottom indices) in which the suffix has exact

match.We recursively find rank query and add it with start or end of the first column to find this.Then using mapping of the indices in the BWT column to the reference sequence we can find the matching indices in the reference sequence.This is how we are finding the exact matching patterns in reads.
After getting the exact matching patterns, we found out how many of these matched indices falls in range of red or green exons given in the README file.The logic we applied to find the exon count is :- In case of unambiguous match, i.e if a read matches with only red or green exon, we increment the respective count by 1. In case of ambiguous match, the count is incremented by 0.5 for both red and green exons.

Now in case of mismatched scenario,we are allowing only max two mismatches.In that case the whole read is segmented into three equal sized parts and each segment is checked for exact matches.It will give a set of indices and for each index we need to check in the original reference sequence the count of mismatches. For this problem up to two mismatches are allowed.

1.B. Find Most probable Configuration :

☑

Given following 4 configurations $\longrightarrow$

config 1: 50% 50% 50% 50%

config 2: 100% 100% ~~100%~~ 0% 0%

config 3: 33% 33% 100% 100%

config 4: 33% 33% 33% 100%

Let $R_c = \{ r_i \mid i \in \{1,2,3,4,5,6\} \} \longrightarrow$ Exon count for red Genes

$G_c = \{ g_i \mid i \in \{1,2,3,4,5,6\} \} \longrightarrow$ " " " Green "

$C_K = \{ p_{Kj} \mid j \in \{2,3,4,5\} \} \rightarrow$ for config K, probability of Generating these counts

So using Bionomial distribution $\longrightarrow$

$$P\left(\{R_c, G_c\} \mid C_K\right) = \prod_{i=2}^{5} {}^{r_i+g_i}C_{r_i} \; p_{Ki}^{r_i} \left(1-p_{Ki}\right)^{g_i}$$

Where $p_{Ki} \rightarrow$ probability of ith exon is red for config K

$(1-p_{Ki}) \rightarrow$ " " ith " " green " " K

So Given the exon counts — we need to calculate this probability for each configuration & conclude the most probable one as the configuration which has highest probability.

Results :

we have observed following results on exon counts:

RedExon-1	RedExon-2	RedExon-3	RedExon-4	RedExon-5	RedExon-6
67.5	56.5	50.0	105.5	216.5	179.0
GreenExon-1	**GreenExon-2**	**GreenExon-3**	**GreenExon-4**	**GreenExon-5**	**GreenExon-6**
67.5	180.5	58.0	100.5	269.5	179.0

We know count of 1st and 6th exons are same for both red and green genes.Here we observed the same from the results.We findout the probability using the formula described last page.And we got following results:

Configuration	Probability of Generation
1	2.98691 exp-22
2	0
3	2.48569 exp-12
4	2.15752 exp-26

Hence we conclude that configuration 3 is the most probable configuration and the reference string has this configuration.

Task 2: Problem Statement:

Show how to answer Select queries on a binary array of size n, allowing cn/Δ a extra space, in $\mathcal{O}(\Delta)$ time. Make c as small as you can.

So for the binary string of size n of a particular character ('A'/'C'/'G'/'T') we will keep record of Δ-rank counts/milestones.This rank array will have storage requirement $\frac{n}{\Delta}$ and it will contain indices of corresponding 1 entry in original binary string for that particular string.So for four characters total size of this additional data structure in $\frac{4n}{\Delta}$.
So to select a given query, we will be given a indev value i and with that we can straight away go to that index in the original binary string and start traversing from there till we reached ith 1.In this approach the extra space allowed is $\frac{n}{\Delta}$ per character.But running time is not as we desired.In the worst case if the array is very sparse and non-uniformly distributed, the running time could be $\mathcal{O}(n)$.But, in amortized case, it will take $\mathcal{O}(\Delta)$ time.
To improve the non-uniform situation,we need an extra data structure of size $\frac{n}{\Delta}$ to point to the next 1, which you store at Δ milestones.This way we can improve the search time and hence the storage requirement will be $\frac{8n}{\Delta}$($\frac{2n}{\Delta}$ per character) and runtime $\mathcal{O}(\Delta)$ time.