project2.pdf

Dario Maddaloni (233187)

March 2022

1 Task 1

The algorithm is dived in two different part:

- Data collection
- Data organization

1.1 Data collection

The core of this part is the "for cycle" that reads all the char of S. This cycle has $n-k$ rounds, because we are interested in the k-mers. At each round, we update the kmer we are interested in (in constant time) and then we check if the kmer is already a key of our dictionary (we can approximate with k computations):

yes: we update the frequency of this kmer adding one in the item aimed by the key (in constant time) and check if the max frequency is no more max_freq.

  yes: we update max_freq (in constant time)

  no: we do nothing

no: we set to 1 our value with key the kmer we found (this is our first time we see it). This cost O(k).

Finally, if we take in to account all the computations we have, then the algorithm has at most an asymptotic complexity of

$$O(3nk - 4k) = O(k(n-1)).$$

1.2 Data organization

First of all, we initialize h with a sequence with *max_freq* 0, because it holds all the frequencies of the frequencies of the kmers (this costs *max_freq* that is at most $n-k$). Moreover, my_second_dict is a dictionary where the keys are the frequencies and the item is the first kmer seen. The core of this subsection is the "for cycle" that takes in to account all the items of my_dict. In particular we have #my_dict.items() rounds: we can state that the upper bound is n. Inside the "for cycle" we have an update of the sequence h, which cost is O(#h), and an "if statement". The control of this statement cost #h=max_freq checks: we can state that the upper bound is $n - k$.

yes: we met an element with frequency= freq the first time. All this state cost at most k

no: this costs k

Hence, this section cost less then the first. We can exclude it from the complete computation of asymptotic complexity of the algorithm. So the algorithm has asymptotic complexity of

$$O(k(n-1)).$$

2 Task 2

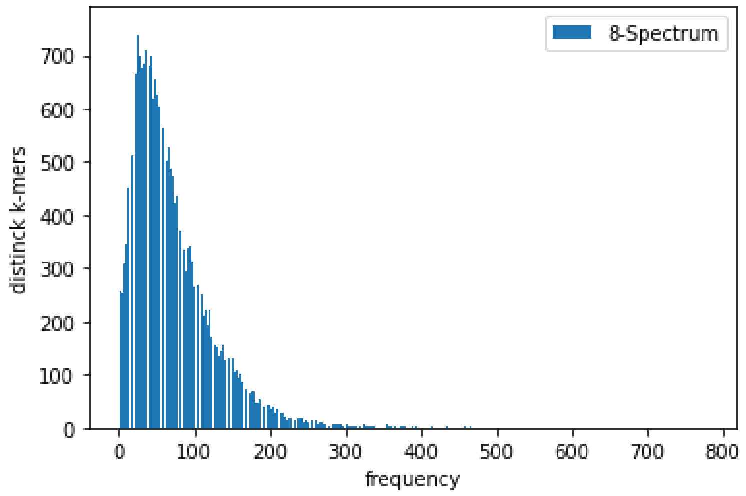

Figure 1: Escherichia Coli: 8-Spectrum

k	mfkmer	frequency
3	['CGC']	115734
8	['CGCTGGCG']	778
15	['ACGCCGCATCCGGCA']	71
25	['GGATAAGGCGTTCACGCCGCATCCG']	39
35	['GTAGGCCGGATAAGGCGTTTACGCCGCATCCGGCA']	22

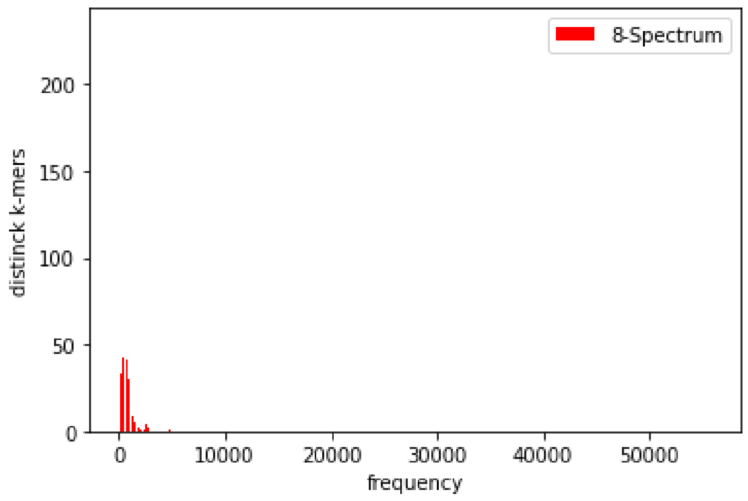

Figure 2: Homo Sapiens: 8-Spectrum

k	mfkmer	frequency
3	['TTT']	1565770
8	['TTTTTTTT']	55795
15	['TTTTTTTTTTTTTTT']	14580
25	['TGTGTGTGTGTGTGTGTGTGTGTGT']	2380
35	['TGGATATTTGGATAGCTTGGAGGATTTCGTTGGAA']	1043

3 Task 3

In this task we have to find out in S all the kmer given in L.
We initialize seq_pos (the sequence of all the positions we meet for the first time for a specific kmer) and seq_freq (the sequence of the frequency of the kmer in the posizion i in the sequence L) This algorithm is divided into 2 different parts

- Subsection1
- maximum_pos and maximum_seq

3.1 Subsection1

This subsection will cost a lot of time because of the fact that we have two nested "for cycle".
In particular, the first one has #len(S)-$k-1$ rounds and the second one has #Len(L). Inside the more nested "for cycle" we have an "if statement" that checks if the string of S is equal to the i-th kmer.

- yes: we found it, hence we add one at the frequency and then we have another "if statement" that checks if we have already found that particular kmer
    - yes: update the position in the position where we found it
    - no: do nothing
- no: do nothing

This subsection has asymptotic complexity of

$$\text{O((\#len(S)-k-1)(\#Len(L))(seq\_freq))=O}((n-k-1)(k)(k))$$

3.2 maximum_pos and maximum_seq

Now, I want to find the maximum of maximum_freq (moreover, the maximum_pos will be the one of the index maximum_freq). These "for cycles" have cost #seq_pos that is negligible.