Mubarra Khan Z5161249 Thursday, 9 July 2020

Q1

After the success of your latest research project in mythical DNA, you have gained the attention of a most diabolical creature: Medusa. Medusa has snakes instead of hair. Each of her snakes' DNA is represented by an uppercase string of letters. Each letter is one of S, N, A, K or E. Your extensive research shows that a snake's venom level depends on its DNA. A snake has venom level x if its DNA:

- has exactly 5x letters • begins with x copies of the letter S
- then has x copies of the letter N • then has x copies of the letter A
- then has x copies of the letter K • ends with x copies of the letter E.
- For example, a snake with venom level 1 has DNA SNAKE, while a snake that has venom level 3 has DNA SSSNNNAAAKKKEEE. If a snake's DNA does not fit the format described above, it has a venom level of 0. Medusa would like your help making her snakes venomous, by deleting zero or more letters from their DNA. Given a snake's DNA, can you work out the maximum venom level this snake could have? Your algorithm should run in time *O(n log n)*

Answer:

Main idea : We know that for a snake of venom level x there is the same number of S,N,A,K and E which is x. So given a string of length n, we note the snakes venom level can be any value between $0 < x \leq n/5$, 0 being the minimum venom level and n/5 being the maximum venom level of the snake. Once determining the maximum venom level possible for a given string we can then delete the extra letters to return the most venomous snake.

Note : Brute force method can be used to check each of these values and eliminate which one does not work to finally return the largest one that does. However this will prove quite slow for large values of n and instead better approach would be to use *Binary Search* which runs in $log(n)$ time.

Binary search (function X):

- Begin by setting the Lo and Hi variables to be 0 and n/5 respectively. Next we call our binary search function to get the middle term which in the first iteration will be n/10.
- We then run a search on the middle term (in a separate function Y) to check if each letter S,N,A,K and E occurs that many times i.e. n/10 number of times for the first iteration.
    - If this returns true then it must mean the max venom level of the snake will be $n/10 \leq maxV \leq n/5.$

        - For the next iteration set the Lo and Hi variables to n/10 +1 and n/5 respectively.

    - Else it will mean the max venom level will be $0 \leq maxV < n/10$.

        - For the next iteration set the Lo and Hi variables to 0 and n/10 -1 respectively.

- We will then recursively call our program until we ascertain a max value that succeeds, which becomes our maximum venom level.
- In this manner the binary search will always choose the more optimal choice.
- **Greedy search (Function Y)** - The greedy search at each iteration will be as follows:
    - We will create the desired subsequence that we need to find in the DNS string. i.e. for the first iteration the subsequence will be a string with n/10 S's , n/10 N's , n/10 A's , n/10 K's and n/10 E's. We will call this - sequence B and the DNA string will be sequence A.
    - Now to check if B is a subsequence of A we will iterate through both strings simultaneously and we will find and mark the earliest occurrence of the first letter of B in A. Then, for each subsequent letter of B, find and mark the earliest occurrence of that letter in A which is after the last marked letter. If you reach the end of B before or at the same time as you reach the end of A, then B is a subsequence of A.
    - This will take time O(n)
- **Analysis :** in each iteration we are making one subsequence check which runs O(n) time since we are running it through the entire original string.
    - We will be making log(n) iterations since we will be using binary search which effectively halves our search space each time.
    - So in total we have $n + n \ldots + n$ many operations from log(n) many calls. Thus in total the algorithm will run in O (n logn).