z5158334
Yuanzhi Shang

Question 1

Firstly, we need to figure out the maximum possible level number. The maximum level number M is limited by the minimum occurrence's times of these five letters S, N, A, K, E. The binary search could be applied on this problem: we set $L = M$. If the DNA sequence does not contain a subsequence matching this level, we set L = M//2. Therefore, the algorithm could be described as following steps.

1. Get the possible maximum level

   Pseudo code

```
/* find the maximum possible level */
int s,n,a,k,e = 0       /* counter of each letter */
s = string.count('S');
n = string.count('N');
a = string.count('A');
k = string.count('K');
e = string.count('E');
return min(s,n,a,k,e)
```

2. Using binary search to determine if the DNA sequence contain a subsequence matching level L. The binary search would check the middle point of M first. Every time we need to form a string which match the current required level. Binary search run in $O(logn)$

   Pseudo code

```
/* min: minium possible ans
 * should be 1 at frist */
while min < max_level:

    set a position p which is the middle of min and max_level

    if string contain a subsequence matching level p:
        /* it may has other possible level */
        min = p;
    else:
        max_level = p - 1
```

3. Inside the binary search, we need to check if the DNA contain a subsequence that matching a level. We only need to go through these two string and compare them. The algorithm would run in $O(n)$

   Psedu code

```
while j < len(original_string) and i < len(substring):
    if original_string[i] == substring[j]:
        i ++;
    j ++;
```

```
/* i is the number of matching */
if i = len(subtring):
    matching success
```

Overall, we have a $O(\log n)$ algorithm that contain a $O(n)$ algorithm. Therefore, the overall time complexity would be $O(n\log n)$