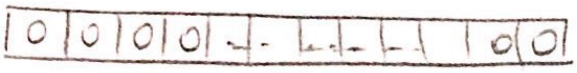
HOMEWORK 4

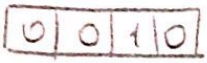
Miray YILDIZ

(1)

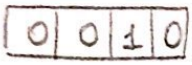
Text is : $\underbrace{0 \ldots 0}_{n}$ length → n

Pattern = 0010 → length → $m=4$

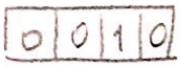




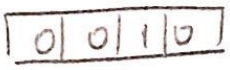
→ 3 character comparison



→ 3 character comparison



→ 3 character comparison



→ 3 char comparison

↷ $(n-m+1)$

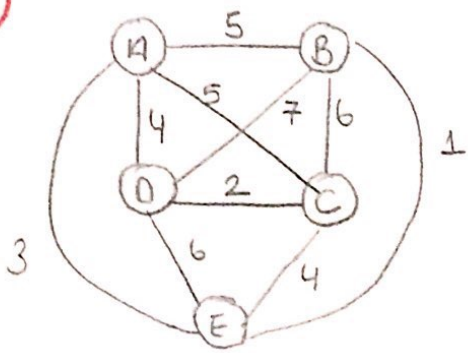
$n-3$ time string comparison

3 times character comparison (each) → 3

Total character comparison → $3(n-3)$

Pattern should be 001 → because first two digit should be same.

(2)



All combinations:

A-B-C-D-E-A => 5+6+2+6+3=22
A-B-C-E-D-A => 5+6+4+6+4=25
A-B-D-C-E-A => 5+7+2+4+3=21
A-B-D-E-C-A => 5+7+6+4+5=27
A-B-E-C-D-A => 5+1+4+2+4=16
A-B-E-D-C-A => 5+1+6+2+5=19
A-C-B-D-E-A => 5+6+7+6+3=27
A-C-B-E-D-A => 5+6+1+6+4=22
A-C-D-B-E-A => 5+2+7+1+3=18
A-C-D-E-B-A => 5+2+6+1+5=19
A-C-E-B-D-A => 5+4+1+7+4=21
A-C-E-D-B-A => 5+4+6+7+5=27
A-D-B-C-E-A => 4+7+6+4+3=24
A-D-B-E-C-A => 4+7+1+4+5=21
A-D-C-B-E-A => 4+2+6+1+3=16
A-D-C-E-B-A => 4+2+4+1+5=16
A-D-E-B-C-A => 4+6+1+6+5=22
A-D-E-C-B-A => 4+6+4+6+5=25
A-E-B-C-D-A => 3+1+6+2+4=16
A-E-B-D-C-A => 3+1+7+2+5=18
A-E-C-B-D-A => 3+4+6+7+4=24
A-E-C-D-B-A => 3+4+2+7+5=21
A-E-D-B-C-A => 3+6+7+6+5=27
A-E-D-C-B-A => 3+6+2+6+5=22

Shourt Routes:

A-E-B-C-D-A
A-D-C-E-B-A
A-D-C-B-E-A
A-B-E-C-D-A

} All of them are 16.

(3) Algorithm:

```
LogFloor(n)
    //Input: Positive Integer n
    //Output: Returns [log2 n]
     if n=1
          return 0
     else
          return LogFloor([n/2])+1
```

The recurrence relation for the number of additions is

$$A(n) = \begin{cases} 0 & , \text{for } n=1 \\ 1 + A\left(\left[\frac{n}{2}\right]\right) & , \text{for } n>1 \end{cases}$$

(4) In decrease and conquer algorithm, in order to find solutions to the given proplems, a solution is sought for an example of a smaller size than problem. The solution for its smaller size is applied to the main problem.

In this question, we found bottles that has different weight. Apply decrease and conquer algorithm.

1- Calculate length of array. → n
2- Divide array's 2 parts. If n is even, each part has n/2 element.
3- If n is odd, (n+1)/2 and (n-1)/2.
4- Calculate both parts length.
5- Choose lighter part.
6- Continue. (until incorrect bottle)

Best case: If the array found incorrect bottle in first comparison, it is O(1).
Worst case: If it found last stage $O(\log_2 n)$
Average case: $O(\frac{1}{2}\log_2 n)$

(5) First we should sort unsorted array. So, I use Mergesort od sort two arrays.

```
Mergesort(arr1)
Merge sort (arr2)

XthElement (arr1, arr2, x)
    if not arr1
        return arr2[x]
    if not arr2
        return arr1[x]
    midIndex1 ← len(arr1)/2
    midIndex2 ← len(arr2)/2
    midElem1 ← arr1[midIndex1]
    midElem2 ← arr2[midIndex2]

    if midIndex1 + midIndex2 < x
        if midElem1 > midElem2
            return XthElement(arr1, arr2[midIndex2+1:], x-midIndex2-1)
        else
            return XthElement(arr1[midIndex1+1:], arr2, x-midIndex1-1)
    else
        if midElem1 > midElem2
            return XthElement(arr1[:midIndex1], arr2, x)
        else
            return XthElement(arr1, arr2[:midIndex2], x)
```

In this code, first I find median elements and median indexes of arr1 and arr2. If x is bigger than the sum of median indices of arr1 and arr2 and if median element of arr1 is bigger than median element of arr2, I make recursive call and this calling first half of arr2 doesnot include x. So, I extract first half of arr2. This process continued for both such arrays. Actually in each step, problem is divide into 2 sub problems by recursively.

So by these recursive callings according to explained conditions, xth elemet of the merged array of these sorted arrays is found without merge first after find xth element later.