171044036 - Baran Hasan BozDUMAN

1)

$$\underbrace{000000000 \text{-----} 0000}_{}\overset{n}{0}$$

n-3

```
0010
 0010
  0010
   0010
    0010
     0010
                  0010
                   0010
```

It will compare until n-3 th elementh after that the algorithm doesn't need to control because the number of pattein doesn't match with rest of choracter.

$3(n-3)$ ↓ foreach compoisions; number of character in the text

The 3 bit pattern for the worst case is "001". It can be used with text which has n number of zeros

n-2, n

```
00000 -----0000
00 1
 0 0 1
  0 0 1 ....001
              0 0 1
```

3 comparision for each character

$3(n-2)$

2) Since they ask for burute-force algorithm we need to try each combination, but also there will be somany way to find shortest path. So I will show the possibleways then I will point the shortest path that I found. and I assume sales person starts "A" city.

A E D C B A -> 22
A B C E D A -> 25
A D E C B A -> 25
A E B C D A -> 16 ✓
A C E D B A -> 27
A D C E B A -> 16 ✓
A E C D B A -> 21
A C D B E A -> 18
A D C E B A -> 16 ✓
A C D E B A -> 19

A E D B C A -> 27
A D E B C A -> 22
A B E D C A -> 19
A E B D C A -> 18
A D B E C A -> 21
A B D E C A -> 27
A B C E D A -> 25
A C B E D A -> 22
A B E C D A -> 22

A C E B D A -> 21
A E C B D A -> 24
A C D B E A -> 18
A B D C E A -> 21
A D B C E A -> 24
A C B D E A -> 27
A B C D E A -> 22

3) As input, a positive integer n; Output returns $\lfloor \log_2 n \rfloor$

```
If n = 1
    return 0
else
    return logfloor(⌊n/2⌋) + 1
```

$T(n) = 0$ for $n=1$

If the number greater than 1 It calls recursive by half the number than add 1.

$T(n) = 1 + T(\lfloor n/2 \rfloor)$ for $n>1$

$$T(n) = \begin{cases} 0 \rightarrow \text{for } n=1 \\ 1+T(\lfloor n/2 \rfloor) \rightarrow \text{for } n>1 \end{cases}$$

Using master theorem

$T(n) = aT(n/b) + \theta(n^d)$ $a>0, b>1, d>,0$

$$T(n) \begin{cases} \theta(n^d) & \text{if } d>\log_b a \\ \theta(n^d \log n) & \text{if } d=\log_b a \\ \theta(n^{\log_b a}) & \text{if } d<\log_b a \end{cases}$$

$d=0$ $b=2$ $a=1$ $\log_2 1 = 0$

$\theta(n^0 . \log n) = T(n) = \theta(\log_2 n)$

4) Its similar to fake-coin problem which was showed in lecture. Firstly I need to indicate even we chose incorrect bottle light or heavy it doesn't change the complexities.

Firstly we split the bottles two section if there's even number of bottle. Otherwise if The number of bottles is odd we split them two again and if both side has some weight the incorrect bottle is the middle one and we will continue accordig to incorrect bottle light or left

For the best case we can find incorrect bottle at first (when the number of bottles odd when we split them and we find left side and right side equal that means, The incorrect bottle is in the middle). So that gives Best case $\Theta(1)$

If we analyze algorithm

$$\left.\begin{array}{l} T(n) = 1 + T(\lfloor n/2 \rfloor) \\ T(n) = 0 \end{array}\right\}$$ according to master theorem it gives. $\Theta(\log n)$ (*Check 3th question) for Worst and AverageCases we can say its $\Theta(\log n)$

5) Firstly we need to sort to array without merging two array. We can use mergesort to sort two arrays, after that. We can use divide and conquer approach

```
mergeSort(arr1);  }  It takes Θ(nlogn + mlogm)
mergeSort(arr2);  }
xth elementh (arr1, begin1, end1, arr2, begin2, end2, x) ⟶ this takes Θ(logn+logm)
    len1 = arr1.length
    len2 = arr2.length
    if (len1 is equal to "0"   //if one of array has no elementh
        return arr2[begin2+x]
    if (len2 is equal to "0" //if one of array has no elementh
        return arr1[begin1+x]
    if x == 1  //if it asks firstelementh
        return  arr1[begin1] > arr2[begin2] ? arr2[begin2] : arr1[begin1]
    if x >(len1+len2) or x < 1; //if the given number is out of bound
        return -1;
    i = min(len1, x/2) / to determine middle elementh
    J = min(len2, x/2) //to determine middle elementh
    If ( arr1[begin1+i] > arr2[begin2+j])  ⟶ arr1's middle elementh greater than other ones sorts in arr1's first part or arr2's second part
        x = x - (J+1)
        end1 = begin1+i
        begin2 = begin2 + J + 1
    else ->otherwise the elementh between arr second part or arr2 first part
        x = x - (i+1)
        begin1 = begin1 + i + 1;
        end2 = begin2 + J
    return xthelementh
        (arr1, begin1, end1, arr2, begin2,
         end2, x)
```

Totally

$\Theta(n\log n + m\log m) + \Theta(\log n + \log m$

thatsmeans the sorting part determines the complexi