Nevzat
Sefaoğlu
171044024

1)

a) there are n zeros so;

000 0000
010

$\underbrace{\quad\quad\quad}$ n-3 comparision for three digits

Total → (n-3) . 3 = 3n - 9

b) The case that will be worst case is <u>001</u> for 3 bits. Because other than last bit, all other bits has to be same for the worst case.

$\underbrace{000 \quad . \quad . \quad . \quad 0}$ 000
<u>001</u> → it will take the pattern and shift through end of the text. therefore pattern should be <u>001</u>

2) I supposed the salesman start at point A: then come back to A after visit all cities.

* There will be 4! different route. According to bruce force we need to consider all the way.

ABCDEA → 5+6+2+6+3 → 22
ACBDEA → 5+6+7+6+3 → 27
ADBCEA → 4+7+6+4+3 → 24
ABDCEA → 5+7+2+4+3 → 21
ACDBEA → 5+2+7+1+3 → 18
R1 - ADCBEA → 4+2+6+1+3 → 16
AECBDA → 3+4+6+7+4 → 24
ACEBDA → 5+4+1+7+4 → 21
ABECDA → 5+7+4+2+4 → 22
R2 - AEBCDA → 3+1+6+2+4 → 16
ACBEDA → 5+6+1+6+4 → 22
ABCEDA → 5+6+4+6+4 → 25
ABDECA → 5+7+6+4+5 → 27
ADBECA → 4+7+1+4+5 → 21
AEBDCA → 3+1+7+2+5 → 18
ABEDCA → 5+1+6+2+5 → 19
ADEBCA → 4+6+1+6+5 → 22
AEDBCA → 3+6+7+6+5 → 27
AEDCBA → 3+6+2+6+5 → 22
ADECBA → 4+6+4+6+5 → 25
ACEDBA → 5+4+6+7+5 → 27
AECDBA → 3+4+2+7+5 → 21
R3 - ADCEBA → 4+2+4+1+5 → 16
ACDEBA → 5+2+6+1+5 → 19

$\left.\begin{array}{l}\text{ADCBEA}\\ \text{AEBCDA}\\ \text{ADCEBA}\end{array}\right\}$ 16 units

There are 3 different routes which are more less than the other ways.

* In brute-force we need to consider all possible ways.

3) In decrease and conquer algorithm, we need to;

* reduce problem instance to smaller instance of the same problem.
* solve smaller instance
* extend the solution of smaller istance to obtain solution to obtain solution

Lets calculate $\log_2 x$ by decreasing the problem size for each call.

if n is not 1:

  return $1 + \log_2\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$

else

  return 0

For all value which is n greater than 1 we are calculating a sum by decreasing the n as half of itself. To handle the precision, we are taking floor of value.

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

$T(n) = 1 + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$ if $n > 1$

$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ 1 + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) & \text{if } n > 1 \end{cases}$$

We will use master teorem for solving the above problem. Here $a = 1$ and $b = 2$

$n^{\log_2 a} = n^{\log_2 1}$

$a = n^0$ $\quad \log_2 1 = 0$

$a = 1$

Therefore, $T(n) = \Theta(n^{\log_b a})$, the second rule of master teorem can be applicable.

$T(n) = \Theta(\log_2 n)$

* As a result, the total efficiency of the algorithm will be $T(n) = \Theta(\log_2 n)$

Scanned with CamScanner

4) In this problem, there are four different situation according to given text. We will consider each problem seperately and showing the worst case, best case and average case is same for all of them.

Case 1

* There are even number of bottles and incorrect weight is less than others. If we seperate the bottles at two parts until reach the incorrect bottle by checking both both side.

worst case

n bottle

$\left(\frac{n}{2} \middle| \frac{n}{2}\right)$ } $\Theta(\log n)$

incorrect bottle here!

{ because each problem size will decrease by half.

* Also this problem is a type of decrease and conquer problem. So master teorem can be applied to proof.

$T(n) = 1 + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$
$T(n) = 0$

} also give logaritmic time for both average and, there fore average case = $\Theta(\log n)$

* In the best case, in algorithm we are splitting sequence in to two part and checking both side if same recursively. But if the second division middle element is already different one, best case will be $\Theta(1)$ constant.

Case 2

There are even number of bottles and incorrect weight is greater than other.

* Results will not change because we just change the side of the incorrect bottle.

Case 3

There are odd number of bottles and incorrect weight is less than others.

* Even we have change the size of bottles as odd, results will never change because after the first division of sequence, odd will be even after just 1 call. Asymtotic notation cannot be affected by constant.

Case 4

There are odd number of bottles and incorrect weight is greater than others.

* As occured in previous case, we just change the side of the incorrect bottle in sequence. It does not change the asymtotic notation.

5) That's a typical k'th element of two sorted array problem. But our arrays are unsorted. Firstly we need to sort both array which does not broke the rule of divide and conquer methods.

* sorting method should be merge-sort because it works according to divide to problem into sub problem and conquer each part.

```
function findKthElement (arr1, arr2, k):

        arr1 = mergeSort (arr1)   } works Θ(nlogn) in
        arr2 = mergeSort (arr2)   } worst case

        helper (arr1, arr2, len(arr1), len(arr2), k-1)
end function


function helper (arr1, arr2, i, j, k)

    if i is zero:
        return arr2[k]
    if j is zero:
        return arr1[k]

    mid1 = i/2
    mid2 = j/2

    if mid1 + mid2 < k

        if arr1[mid1] > arr2[mid2]:
            return helper (arr1, arr2+mid2+1, i, j, k-mid2-1)
        else
            return helper (arr1[mid1+1:], arr2, i, j, k-mid1-1)

    else

        if arr1[mid1] > arr2[mid2]
            return helper (arr1, arr2, i-mid1, j, k)
        else
            return helper (arr1, arr2, i, j-mid2, k)
```

* The approach is quite similar to binary search. We are checking the comparision of middle element in each recursive call. According to result, queried part is changing until the assumption is correct.

* In worst case, like in binary search, because the decisions made according to middle element comparision of each array. Therefore, $\Theta(\log(size1 + size2))$

Result: our design works in;

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