Department of Computer Science and Engineering

PES UNIVERSITY

UE19CS251: Design and Analysis of Algorithms (4-0-0-4-4)
Unit 2: Questions and Answers

Q.1 What are the advantages and disadvantages of BruteForce approach?
Solution Brute-force approach is simple to implement, and will always find a Solution if it exists. However, its cost is proportional to the number of candidate Solutions, which, in many practical problems, tends to grow very quickly as the size of the problem increases. Therefore, brute-force search is typically used when the problem size is limited, or when there are problem-specific heuristics that can be used to reduce the set of candidate Solutions to a manageable size. The method is also used when the simplicity of implementation is more important than speed. This is the case, for example, in critical applications where any errors in the algorithm would have very serious consequences; or when using a computer to prove a mathematical theorem. Brute-force search is also useful as "baseline" method when benchmarking other algorithms

Q.2 Design an efficient brute-force algorithm for computing the value of a polynomial $p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdot\cdot\cdot + a_1 x + a_0$ at a given point x_0 and determine its worst-case efficiency class.
Solution

```
Algorithm BruteForcePolynomialEvaluation(P[0..n], x)
//The algorithm computes the value of polynomial P at a given point x
//by the “lowest-to-highest term” algorithm
//Input: Array P[0..n] of the coefficients of a polynomial of degree n,
// from the lowest to the highest, and a number x
//Output: The value of the polynomial at the point x
p ← P[0]; power ← 1
for i ← 1 to n do
        power ← power ∗ x
        p ← p + P[i] ∗ power
return p
```

$$M(n) = \sum_{i=1}^{n} 2 = 2n$$

Q.3 Sort the list E, X, A, M, P, L, E in alphabetical order by bubble sort

Solution

$$
\begin{array}{ccccccccccccc}
E & \overset{?}{\leftrightarrow} & X & \overset{?}{\leftrightarrow} & A & & M & & P & & L & & E \\
E & & A & & X & \overset{?}{\leftrightarrow} & M & & P & & L & & E \\
E & & A & & M & & X & \overset{?}{\leftrightarrow} & P & & L & & E \\
E & & A & & M & & P & & X & \overset{?}{\leftrightarrow} & L & & E \\
E & & A & & M & & P & & L & & X & \overset{?}{\leftrightarrow} & E \\
E & & A & & M & & P & & L & & E & & |X \\
E & \overset{?}{\leftrightarrow} & A & & M & & P & & L & & E & & \\
A & & E & \overset{?}{\leftrightarrow} & M & \overset{?}{\leftrightarrow} & P & \overset{?}{\leftrightarrow} & L & & E & & \\
A & & E & & M & & L & & P & \overset{?}{\leftrightarrow} & E & & \\
A & & E & & M & & L & & E & & |P & & \\
A & \overset{?}{\leftrightarrow} & E & \overset{?}{\leftrightarrow} & M & \overset{?}{\leftrightarrow} & L & & E & & & & \\
A & & E & & L & & M & \overset{?}{\leftrightarrow} & E & & & & \\
A & & E & & L & & E & & |M & & & & \\
A & \overset{?}{\leftrightarrow} & E & \overset{?}{\leftrightarrow} & L & \overset{?}{\leftrightarrow} & E & & & & & & \\
A & & E & & E & & |L & & & & & & \\
A & \overset{?}{\leftrightarrow} & E & \overset{?}{\leftrightarrow} & E & \overset{?}{\leftrightarrow} & L & & & & & &
\end{array}
$$

Q.4 Find the number of character comparisons that will be made by 'straight forward string matching' for the pattern ABABC in the following text: BAABABABCCA

Solution: number of character comparisons required are 14

Q.5 Write a brute-force algorithm for counting the number of vowels in a given Text.

Solution:

```
A [1….n], cont =0;
For (i=0; i<=n; i++)
{
        If (a[i] ==”a”||”e”||”i”||”o”||”u”)
                Count++;
}
```

Q.6 Give an example of a text of length n and a pattern of length m that constitutes a worst-case input for the brute-force string-matching algorithm. Exactly how many character comparisons will be made for such input?

Solution:

The text composed of n zeros and the pattern 0………… 0(m-1 times)1 is an example of the worst-case input. The algorithm will make m(n − m + 1) character comparisons on such input.

Q.7 Find the optimal solution for the assignment problem given below

	Job 1	Job 2	Job 3	Job 4
Person 1	4	3	8	6
Person 2	5	7	2	4
Person 3	16	9	3	1
Person 4	2	5	3	7

Solution:

Person1 → Job 2

Person2 → Job 3

Person3 → Job 4

Person4 → Job 1

Total cost=8

Q.8 Write pseudocode for divide-and-conquer algorithm for the exponentiation problem of computing an where a > 0 and n is a positive integer

Solution:

Algorithm DivConqPower(a, n)
//Computes an by a divide-and-conquer algorithm
//Input: A positive number a and a positive integer n
//Output: The value of an
if n = 1
    return a
else
    return DivConqPower(a,rounddown(n/2)∗DivConqPower(a,roundup(n/2))

Q.9 Find the order of growth for solutions of the following recurrences.

a. T(n) = 4T(n/2) + n, T(1) = 1
b. T(n) = 4T(n/2) + n^2, T(1) = 1
c. T(n) = 4T(n/2) + n^3, T(1) = 1

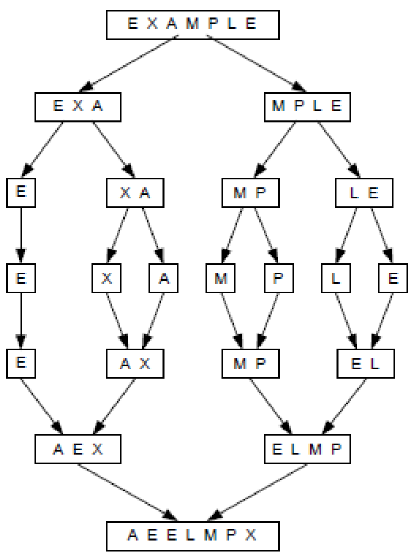
Solution: a. Theta(n^2)

b. Theta(n^2logn)

c. Theta(n^3)

Q.10 Apply mergesort to sort the list E, X, A, M, P, L, E in alphabetical order

Solution:



Q.11 Write a pseudocode for a divide-and-conquer algorithm for finding values of both the largest and smallest elements in an array of n numbers.

Solution:

```
Algorithm MinMax (A[l..r], minval, maxval )
//Finds the values of the smallest and largest elements in a given subarray
//Input: A portion of array A[0..n − 1] between indices l and r (l ≤ r)
//Output: The values of the smallest and largest elements in A[l..r]
//assigned to minval and maxval, respectively
if r = l
        minval ← A[l]; maxval ← A[l]
else if r − l = 1
        if A[l] ≤ A[r]
                minval ← A[l]; maxval ← A[r]
        else
                 minval ← A[r]; maxval ← A[l]
else //r − l > 1
        MinMax(A[l..rounddown((l + r)/2)], minval, maxval )
        MinMax(A[roundup((l + r)/2) + 1..r], minval2, maxval2 )
        if minval2 < minval
                minval ← minval2
        if maxval2 > maxval
                maxval ← maxval2
```

Q.12 Apply quicksort to sort the list M, E, R, G, E, S, O, R, T in alphabetical order. Find the element whose position is unchanged in the sorted list.
Solution: After applying quick sort to the given list M,E,R,G,E,S,O,R,T the element T ‘s position is unchanged for in the sorted list

Q.13 Design a divide-and-conquer algorithm for computing the number of levels in a binary tree. (In particular, the algorithm must return 0 and 1 for the empty and single-node trees, respectively.) What is the efficiency class of your algorithm?
Solution:

```
Algorithm Levels(T)
//Computes recursively the number of levels in a binary tree
//Input: Binary tree T
//Output: Number of levels in T
        if T = ∅
                return 0
```

else

return max{Levels(TL), Levels(TR)} + 1

Time complexity Θ(n)

Q.14 Compute 2101 ∗ 1130 by applying the divide-and-conquer algorithm outlined in the text

Solution:

For 2101 ∗ 1130:

c2 = 21∗ 11

c0 = 01∗ 30

c1 = (21 + 01) ∗ (11 + 30) − (c2 + c0) = 22 ∗ 41 − 21 ∗ 11 − 01 ∗ 30.

For 21 ∗ 11:

c2 = 2∗1 = 2

c0 = 1∗1 = 1

c1 = (2 + 1) ∗ (1 + 1) − (2 + 1) = 3 ∗ 2 −3 = 3.

So, 21 ∗ 11 = $2 \cdot 10^2 + 3 \cdot 10^1 + 1 = 231$.

For 01 ∗ 30:

c2 = 0∗3 = 0

c0 = 1∗0 = 0

c1 = (0 + 1) ∗ (3 + 0) − (0 + 0) = 1 ∗ 3 −0 = 3.

So, 01 ∗ 30 = $0 \cdot 10^2 + 3 \cdot 10^1 + 0 = 30$.

For 22 ∗ 41:

c2 = 2∗4 = 8

c0 = 2∗1 = 2

c1 = (2 + 2) ∗ (4 + 1) − (8 + 2) = 4 ∗ 5 − 10 = 10.

So, 22 ∗ 41 = $8 \cdot 10^2 + 10 \cdot 10^1 + 2 = 902$.

Hence

2101 ∗ 1130 = $231 \cdot 10^4 + (902 - 231 - 30) \cdot 10^2 + 30$ = 2, 374, 130

Q.15 Write an algorithm for Mergesort. Mention its time complexity for Best, Worst and Average case.

Solution

ALGORITHM Mergesort(A[0..n-1])

//Sorts array A[0..n-1] by recursive mergesort

//Input: An array A[0..n-1] of orderable elements

//Output: Array A[0..n-1] sorted in nondecreasing order

if n > 1

//divide

copy A[0..   n/2 – 1] to B[0..   n/2 – 1]

copy A[n/2 .. n – 1] to C[0..   n/2 – 1]

//conquer

Mergesort(B[0..   n/2 – 1)

Mergesort(C[0..   n/2 – 1)

//combine

Merge(B, C, A)

ALGORITHM Merge(B[0..p-1], C[0..q-1], A[0..p+q-1])

//Merges two sorted arrays into one sorted array

//Input: Arrays B[0..p-1] and C[0..q-1] both sorted

//Output: Sorted Array A[0..p+q-1] of the elements of B and C

i ← 0; j ← 0; k ← 0

while i < p and j < q do //while both B and C are not exhausted

if B[i] <= C[j]   //put the smaller element into A

A[k] ← B[i]; i ← i + 1

else A[k] ← C[j]; j ← j + 1

k ← k + 1

if i = p   //if list B is exhausted first

copy C[j..q-1] to A[k..p+q-1]

else   //list C is exhausted first

copy B[i..p-1] to A[k..p+q-1]

Time complexity: worst, Best and Average Case: nlogn