Syn

Sardar Patel Institute of Technology

Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous Institute Affiliated to University of Mumbai)

Mid Semester Examination
March-2018
Synoptic

Max. Marks: 30 | Duration: 1.5 Hrs
Class: S.Y. | Semester: IV
Course Code: **MCA43** | Branch: M.C.A.
Name of the Course: **Design and Analysis of Algorithm**

Q1) What is Recursion? Write a routine to calculate Fibonacci series using it.
Answer:
Definition:----------2 Marks
A recursive algorithm is an algorithm which calls itself with "smaller (or simpler)" input values, and which obtains the result for the current input by applying simple operations to the returned value for the smaller
(or simpler) input. More generally if a problem can be solved utilizing solutions to smaller versions of the same problem,
and the smaller versions reduce to easily solvable cases, then one can use a recursive algorithm to solve that problem.
For example, the elements of a recursively defined set, or the value of a recursively defined function can be obtained by a recursive algorithm.

Algorithm:-----3 marks

```
int Fibonacci(int);

int main()
{
  int n, i = 0, c;

  scanf("%d",&n);

  printf("Fibonacci series\n");

  for ( c = 1 ; c <= n ; c++ )
  {
    printf("%d\n", Fibonacci(i));
    i++;
  }

  return 0;
}

int Fibonacci(int n)
{
  if ( n == 0 )
    return 0;
  else if ( n == 1 )
```

```
    return 1;
  else
    return ( Fibonacci(n-1) + Fibonacci(n-2) );
}
```

Q2)Compare and contrast P and NP problems.
Answer:
Definition :- of P & NP 2Marks
Example of each:2Marks
One comparision of each:1Marks
P- Polynomial time solving . Problems which can be solved in polynomial time, which take time like O(n), O(n2), O(n3).
Eg: finding maximum element in an array or to check whether a string is palindrome or not.
so there are many problems which can be solved in polynomial time.
P is a complexity class that represents the set of all decision problems that can be solved in polynomial time.
That is, given an instance of the problem, the answer yes or no can be decided in polynomial time.

Example
Given a connected graph G, can its vertices be coloured using two colours so that no edge is monochromatic?
Algorithm: start with an arbitrary vertex, color it red and all of its neighbours blue and continue. Stop when you run out of vertices or you are forced to make an edge have both of its endpoints be the same color.

NP- Non deterministic Polynomial time solving. Problem which can't be solved in polynomial time like
TSP(travelling salesman problem) or An easy example of this is subset
sum: given a set of numbers, does there exist a subset whose sum is zero?.
NP is the class of problems which have solutions that can be efficiently verified.
As usual, efficiently means polynomial in size of input.
NP stands for nondeterministic polynomial time.
coloring is in NP. Given a proposed
coloring, we can quickly check if it works

NP
NP is a complexity class that represents the set of all decision problems for which the instances where the answer is "yes" have proofs that can be verified in polynomial time.
This means that if someone gives us an instance of the problem and a certificate (sometimes called a witness) to the answer being yes, we can check that it is correct in polynomial time.
Example
Integer factorisation is in NP. This is the problem that given integers n and m, is there an integer f with $1 < f < m$, such that f divides n (f is a small factor of n)?

This is a decision problem because the answers are yes or no.
If someone hands us an instance of the problem (so they hand us integers n and m) and an integer f with $1 < f < m$, and claim that f is a factor of n (the certificate), we can check the answer in polynomial time by performing the division n / f.

Q3) Role of Algorithm in computing: point wise explanation is expected for (at least 5) following points

i) problem solving
ii) Data stuctures
iii) Techniques
iv) Hard problems
v) parallelism
vi) efficency

one mark for each point. answer should relate the above points.

Q4)
Analysis of Time Complexity of MergeSort Algorithm

Algorithm:

```
mergesort( int [] a, int left, int right)
{
        if (right > left)
        {
                middle = left + (right - left)/2;
                mergesort(a, left, middle);
                mergesort(a, middle+1, right);
                merge(a, left, middle, right);
        }
}
```

Assumption: N is a power of two.
For N = 1: time is a constant (denoted by 1)
Otherwise: time to mergesort N elements = time to mergesort N/2 elements plus time to merge two arrays each N/2 elements.
Time to merge two arrays each N/2 elements is linear, i.e. N
Thus we have:
(1) $T(1) = 1$
(2) $T(N) = 2T(N/2) + N$
Next we will solve this recurrence relation. First we divide (2) by N:
(3) $T(N) / N = T(N/2) / (N/2) + 1$
N is a power of two, so we can write
(4) $T(N/2) / (N/2) = T(N/4) / (N/4) + 1$
(5) $T(N/4) / (N/4) = T(N/8) / (N/8) + 1$
(6) $T(N/8) / (N/8) = T(N/16) / (N/16) + 1$
(7) .....
(8) $T(2) / 2 = T(1) / 1 + 1$

Now we add equations (3) through (8) : the sum of their left-hand sides
will be equal to the sum of their right-hand sides:
T(N) / N + T(N/2) / (N/2) + T(N/4) / (N/4) + ... + T(2)/2 =
T(N/2) / (N/2) + T(N/4) / (N/4) + ...+ T(2) / 2 + T(1) / 1 + LogN
(LogN is the sum of 1s in the right-hand sides)
After crossing the equal term, we get
(9) T(N)/N = T(1)/1 + LogN
T(1) is 1, hence we obtain
(10) T(N) = N + NlogN = O(NlogN)
Hence the complexity of the MergeSort algorithm is O(NlogN).
For every two steps 1 Mark

OR

Q.4) Consider the string1: ABCDMZ, String2: ACABDZ. Find Longest Common Subsequence with its length.
LCS length=4 and LCS= 2 possibilities
Ans1: ACDZ
Ans2:ABDZ
Table for correct enteries in LCS: 3 Marks
interpretation of path for answer1 --1Mark
interpretation of path for answer2 --1Mark

(2) A B D Z

Length of LCS = 4

Q.5. Total Weight of knapsack = 60

I	w_i	V_i	V_i/w_i
1	5	30	6
2	10	20	2
3	20	100	5
4	30	90	3
5	40	160	4

descending order of V_i/w_i

I	w_i	V_i	V_i/w_i
1	5	30	6
2	20	100	5
3	40	160	4
4	30	90	3
5	10	20	2

(i) 

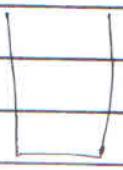
$W = 60 - 5 = 55$

$P_i = 0 + 30 = 30$

(ii) 

$W = 55 - 20 = 35$

$P = 30 + 100 = 130$

(iii) 

Q-6 - Solution.

Chain of matrices = [2 3 6 4]

∴ matrices are [2×3] · [3×6] · [6×4]

total 3 matrices. [A·B·C] or [0 · 1 · 2]

	0	1	2
0	0	36	84 (01)×2
1	×	0	72
2	×	×	0

for L = 1 self multiplication, diagonal elements are zero.

L = 2 case: [0 1] = 2×3×6 = 36

case: [1 2] = 3×6×4 = 72

L = 3
- case 1 - [0 1] · 2 = cost of [01] + cost of 2
  = 36 + 2×6×4
  = 36 + 48 = 84
- case 2 - [1 2] = cost of [12] + cost of 0
  = 72 + 2×3×4
  = 72 + 24
  = 96

min

= 84

cost of multiplication = 84

order of multiplication = [01] × 2