ADA

Section C

Q3 Algorithm to compute Transitive Closure

```
for (i=0 to n-1) do
    for (j=0 to n-1) do
        P[i][j] = A[i][j]
    end for
end for

for (k=0 to n-1) do
    for (i=0 to n-1) do
        for (j=0 to n-1) do
            if (P[i][j]=0 and P[i,k]=1 &&
                                P[k,j]=1
                then
                    P[i,j] = 1
            end if
        end for
    end for
end for
```

Time efficiency of Warshall's Algorithm

$$f(n) = \sum_{k=0}^{n-1} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} 1$$

$$= \sum_{k=0}^{n-1} \sum_{i=0}^{n-1} (n-\cancel{1}-0+\cancel{1})$$

$$= \sum_{k=0}^{n-1} n \times \sum_{i=0}^{n-1} 1$$

$$= \sum_{k=0}^{n-1} n \times (n-\cancel{1}-0+\cancel{1})$$

$$= \sum_{k=0}^{n-1} n^2$$

$$= n^2 \sum_{k=0}^{n-1} 1$$

$$= n^2 (n-\cancel{1}-0+\cancel{1})$$

$$= n^3$$

So, Time complexity $= O(n^3)$

Q4 Prim's Program

```
#include <stdio.h>
#include <conio.h>
#define INFINITY 999
int prim(int cost[10][10], int source, int n)
{
    int i,j, visited[10], verten[10], cmp[10];
    int min, u, v, sum=0;
    for(i=1; i<=n; i++)
    {
        verten[i]= source;
```

```
visited[i]=0;
cmp[i] = cost [source][i];
}
visited[source] = 1;
for(i=1; i<=n-1; i++)
{
    min = INFINITY;
    for(j=1; j<=n; j++)
    if( ! visited[j] && cmp[j] < min)
    {
        min= cmp[j];
        u = j;
    }
    visited[u]=1;
    sum += cmp[u];
    printf("\n%d→%d sum=%d",
                vertex[u], u, sum);
    for(v=1; v<=n; v++)
    if(! visited[v] && cost[u][v] < cmp[v])
    {
        cmp[v] = cost[u][v];
        vertex[v]=u;
    }
}

void main()
{
    int a[10][10], n, i, j, m, source, k;
```

```
printf(" Enter no. of vertices");
scanf("%d" , &n);
printf("Enter cost matrin);
for (i=1 ; i<=n ; i++)
 for (j=1 ; j<=n ;j++)
 scanf(" %d" , & a[i][j];
printf(" Enter the source");
scanf("%d", & source );
m = prim (a, source, n);
printf(" Cost = %d", m);
getch();
}
```

Section B

Q1

Difference

Divide & Conquer Method

(1) It deals three steps at each level of recursion : Divide, Conquer , Combine.

(2) It does more work on subproblems & hence has more time consumption.

(3) It is a topdown approach.

(4) In this subproblem are independent of each other.

(5) It is Recursive.

(6) Example : Merge Sort

Dynamic Programming

(1.) It involves sequence of four steps.
  <i> Characterize the structure.
  <ii> Recursively define the values
  <iii> Compute the value.
  <iv> Construct the optimal solution.

(2) It solves subproblems only once & then stores in the table.

(3) It is a Bottom-up approach.

(4) In this subproblems are solved dependently.

(5) It is non recursive.

(6) Example : Fibnacci series.

Q2 Bionomial Cofficient of ${}^{6}C_{4}$

$c(6,4) = c(5,3) + c(5,4)$
$c(5,3) = c(4,2) + c(4,3)$
$c(5,4) = c(4,3) + c(4,4)$
$c(4,2) = c(3,1) + c(3,2)$
$c(4,3) = c(3,2) + c(3,3)$
$c(3,1) = c(2,0) + c(2,1)$
$c(3,2) = c(2,1) + c(2,2)$
$c(2,1) = c(1,0) + (1,1)$

$\longrightarrow k$

n \ k	0	1	2	3	4
0	1				
1	1	1			
2	1	?	1		
3	1	?	?	1	
4	1	2	?	?	1
5	1	?	?	?	?
6	1	?	?	?	?

So,

$C(2,1) = 1 + 1 = 2$
$C(3,2) = 2 + 1 = 3$
$C(3,1) = 1 + 2 = 3$
$C(4,3) = 3 + 1 = 4$
$C(4,2) = 3 + 3 = 6$
$C(5,4) = 4 + 1 = 5$
$C(5,3) = 6 + 4 = 10$
$C(6,4) = 10 + 5 = 15$

n \ k →	0	1	2	3	4
0	1				
1	1	1			
2	1	2	1		
3	1	3	3	1	
4	1	5	6	4	1
5	1			10	5
6	1				15

So, $C(6,4) = 15$

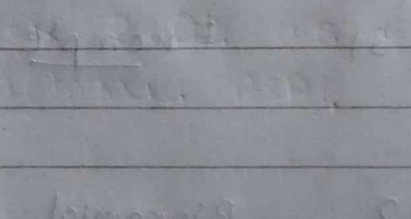
Section A

Q1 Transitive Closure

If there is a path from (i, k) & (k, j) then there exist a path from (i, j)

Example: Input:

1 1 0 1
0 1 1 0
0 0 1 1
0 0 0 1

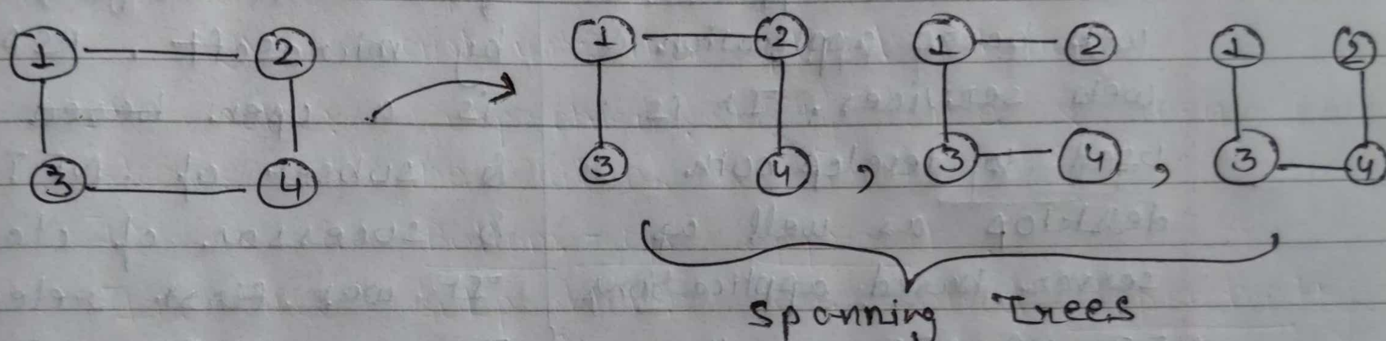


Output:

1 1 1 1
0 1 1 1
0 0 1 1
0 0 0 1

Q2 Sponning Tree

Sponning Tree is a tree where all nodes should be connected & graph should be acyclic.



Q3 Optimal Solution

An optimal solution is the solution where the objective function reaches the manimum or minimum value.

For example : The most profit or the least cost.

Q5 Two functions of kruskal's Algorithm

(1) find ()
(2) union ()

Q6 Problems that can be solved using dynamic :

(1) Partition problem
(2) Matrin Chain Multiplication
(3) Coin change Problem
(4) Word break Problem