**EXPERIMENT DETAILS**

**1.Design and implement a Program to sort a given set of n integer elements using Selection Sort method and compute its time complexity. Run the program for varied values of n> 5000 and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.**

**………………………………………………………………………………………………………..**

PROGRAM:

#include<stdio.h>

#include<time.h>

#include<stdlib.h>

void selectionSort(int arr[], int n)

{

int i, j, minIndex, temp;

// Traverse through all array elements

for (i = 0; i < n-1; i++)

{

// Find the minimum element in unsorted array

minIndex = i;

for (j = i+1; j < n; j++)

{

if (arr[j] < arr[minIndex])

minIndex = j;

}

// Swap the found minimum element with the first element

temp = arr[minIndex];

arr[minIndex] = arr[i];

arr[i] = temp;

}

}

// Function to print an array

void printArray\_beforesort(int arr[], int size)

{

int i;

for (i=0; i < size; i++)

printf("%d ", arr[i]);

printf("\n");

}

void printArray\_aftersort(int arr[], int size)

{

int i;

for (i=0; i < size; i++)

printf("%d ", arr[i]);

printf("\n");

}

int main()

{

int arr[10000],n,i;

double time\_taken;

clock\_t start,end;

printf("Enter the number of elements \n");

scanf("%d",&n);

for(i=0;i<n;i++)

arr[i]=rand()%10000;

printf("elements before sorting");

printArray\_beforesort( arr, n);

start=clock();

selectionSort(arr, n );

end=clock();

printf("elements after sorting\n");

printArray\_aftersort( arr, n);

time\_taken=((double)(end-start))/CLOCKS\_PER\_SEC\*1000;

printf("time taken to sort elements=%lf",time\_taken);

return 0;

}

OUTPUT:

Enter the number of elements

100

elements before sorting9383 886 2777 6915 7793 8335 5386 492 6649 1421 2362 27 8690 59 7763 3926 540 3426 9172 5736 5211 5368 2567 6429 5782 1530 2862 5123 4067 3135 3929 9802 4022 3058 3069 8167 1393 8456 5011 8042 6229 7373 4421 4919 3784 8537 5198 4324 8315 4370 6413 3526 6091 8980 9956 1873 6862 9170 6996 7281 2305 925 7084 6327 336 6505 846 1729 1313 5857 6124 3895 9582 545 8814 3367 5434 364 4043 3750 1087 6808 7276 7178 5788 3584 5403 2651 2754 2399 9932 5060 9676 3368 7739 12 6226 8586 8094 7539

elements after sorting

12 27 59 336 364 492 540 545 846 886 925 1087 1313 1393 1421 1530 1729 1873 2305 2362 2399 2567 2651 2754 2777 2862 3058 3069 3135 3367 3368 3426 3526 3584 3750 3784 3895 3926 3929 4022 4043 4067 4324 4370 4421 4919 5011 5060 5123 5198 5211 5368 5386 5403 5434 5736 5782 5788 5857 6091 6124 6226 6229 6327 6413 6429 6505 6649 6808 6862 6915 6996 7084 7178 7276 7281 7373 7539 7739 7763 7793 8042 8094 8167 8315 8335 8456 8537 8586 8690 8814 8980 9170 9172 9383 9582 9676 9802 9932 9956

time taken to sort elements=0.022000

……………………………………………………………………………………………………….

**2.Design and implement a Program to sort a given set of n integer elements using Merge Sort method and compute its time complexity. Run the program for varied values of n> 5000, and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.**

**………………………………………………………………………………………………………..**

PROGRAM :

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define MAX 10000

void merge(int array[], int low, int mid, int high)

{

int i = low;

int j = mid + 1; int k = low;

int resarray[MAX];

while (i <= mid && j <= high)

{

if (array[i] < array[j])

{

resarray[k++] = array[i++];

}

else

{

resarray[k++] = array[j++];

}

}

while (i <= mid)

{

resarray[k++] = array[i++];

}

while (j <= high)

{

resarray[k++] = array[j++];

}

for (int m = low; m <= high; m++)

{

array[m] = resarray[m];

}

}

void sort(int array[], int low, int high)

{

if (low < high)

{

int mid = (low + high) / 2;

sort(array, low, mid);

sort(array, mid + 1, high);

merge(array, low, mid, high);

}

}

int main()

{

int array[MAX];

int i;

printf("Enter the array size: ");

int n;

scanf("%d", &n);

srand(time(NULL));

for (i = 0; i < n; i++)

{

array[i] = rand() % 10000;

}

printf("Array before sorting:\n");

for (i = 0; i < n; i++)

{

printf("%d ", array[i]);

}

printf("\n");

clock\_t start = clock();

sort(array, 0, n - 1);

clock\_t end = clock();

printf("Sorted array:\n");

for (i = 0; i < n; i++)

{

printf("%d ", array[i]);

}

printf("\n");

double elapsedTime = (double)(end - start) / CLOCKS\_PER\_SEC \* 1000;

printf("Time taken to sort array is: %lf miliseconds\n", elapsedTime);

return 0;

}

OUTPUT:Enter the array size: 100

Array before sorting:

3010 8057 2487 4556 1827 4748 2025 206 3200 1155 9757 8999 5688 9379 7305 7330 102 8786 758 6324 1249 5131 6928 3517 1070 2429 8768 2540 7130 7428 1874 6492 5485 4362 7401 3664 9110 9426 223 8663 6933 9980 7662 2622 9360 1319 9952 5814 105 7062 2138 7706 2193 5418 1224 3263 7847 9992 5803 1329 7420 4030 7822 9258 8392 5223 9274 3854 4649 9497 8869 1583 5830 2883 557 1542 4202 6861 7356 660 3923 9495 8366 6116 1265 9590 5732 5465 5934 7887 6794 3355 1917 968 8965 6661 2543 4591 6868 7193

Sorted array:

102 105 206 223 557 660 758 968 1070 1155 1224 1249 1265 1319 1329 1542 1583 1827 1874 1917 2025 2138 2193 2429 2487 2540 2543 2622 2883 3010 3200 3263 3355 3517 3664 3854 3923 4030 4202 4362 4556 4591 4649 4748 5131 5223 5418 5465 5485 5688 5732 5803 5814 5830 5934 6116 6324 6492 6661 6794 6861 6868 6928 6933 7062 7130 7193 7305 7330 7356 7401 7420 7428 7662 7706 7822 7847 7887 8057 8366 8392 8663 8768 8786 8869 8965 8999 9110 9258 9274 9360 9379 9426 9495 9497 9590 9757 9952 9980 9992

Time taken to sort array is: 0.038000 miliseconds.

……………………………………………………………………………………………………..

**3.Design and implement a Program to solve discrete Knapsack and continuous Knapsack problems using**

**greedy approximation method.**

**…………………………………………………………………………………………………………………….**

PROGRAM :

#include <stdio.h>

#define MAX 50

int p[MAX], w[MAX];

float x[MAX];

double maxprofit;

int n, m, i;

void greedyKnapsack(int n, int w[], int p[], int m) {

double ratio[MAX];

// Calculate the ratio of profit to weight for each item

for (i = 0; i < n; i++)

{

ratio[i] = (double)p[i] / w[i];

}

// Sort items based on the ratio in non-increasing order

for (i = 0; i < n - 1; i++)

{

for (int j = i + 1; j < n; j++)

{

if (ratio[i] < ratio[j])

{

double temp = ratio[i];

ratio[i] = ratio[j];

ratio[j] = temp;

int temp2 = w[i];

w[i] = w[j];

w[j] = temp2;

temp2 = p[i];

p[i] = p[j];

p[j] = temp2;

}

}

}

int currentWeight = 0;

maxprofit = 0.0;

// Fill the knapsack with items

for (i = 0; i < n; i++)

{

if (currentWeight + w[i] <= m)

{

x[i] = 1; // Item i is selected

currentWeight += w[i];

maxprofit += p[i];

}

else

{

// Fractional part of item i is selected

x[i] = (m - currentWeight) / (double)w[i];

maxprofit += x[i] \* p[i];

break;

}

}

printf("Optimal solution for greedy method: %.1f\n", maxprofit);

printf("Solution vector for greedy method: ");

for (i = 0; i < n; i++)

printf("%f\t", x[i]);

}

int main()

{

printf("Enter the number of objects: ");

scanf("%d", &n);

printf("Enter the objects' weights: ");

for (i = 0; i < n; i++)

scanf("%d", &w[i]);

printf("Enter the objects' profits: ");

for (i = 0; i < n; i++)

scanf("%d", &p[i]);

printf("Enter the maximum capacity: ");

scanf("%d", &m);

greedyKnapsack(n, w, p, m);

return 0;

}

OUTPUT:

Enter the number of objects: 7

Enter the objects' weights: 2 3 5 7 1 4 1

Enter the objects' profits: 10 5 15 7 6 18 3

Enter the maximum capacity: 15

Optimal solution for greedy method: 55.33

Solution vector for greedy method: 1.000000 1.000000 1.000000 1.000000 1.000000 0.666667 0.000000

………………………………………………………………………………………………………

**4.Design an application for a thermal power station and electrical lines that are connected among various power stations. The costs of electrification involved appear as weights on the edges. Obtain the minimum possible connection among the thermal stations so that any two thermal stations can be linked with the minimum cost involved.**

**………………………………………………………………………………………………………………………**

PROGRAM :

#include<stdio.h>

int a,b,u,n,i,j,ne=1,v;

int visited[10],min,mincost=0,cost[10][10];

int main()

{

printf("Enter the number of vertices and graph data\n");

scanf("%d",&n);

for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

{

printf("(%d,%d):",i,j);

scanf("%d",&cost[i][j]);

if(cost[i][j]==0)

cost[i][j]=999;

}

for(i=1;i<=n;i++)

visited[i]=0;

printf("The edges of spanning tree are\n");

visited[1]=1;

while(ne<n)

{

for(i=1,min=999;i<=n;i++)

{

for(j=1;j<=n;j++)

{

if(cost[i][j]<min)

{

if(visited[i]==0)

continue;

else

{

min=cost[i][j];

a=u=i;

b=v=j;

}

}

}

}

if(visited[u]==0||visited[v]==0)

{

printf("%d:edge(%d,%d)=%d\t",ne++,a,b,min);

mincost=mincost+min;

visited[b]=1;

}

cost[a][b]=cost[b][a]=999;

}

printf("\nMinimum cost is %d\n",mincost);

return 0;

}

OUTPUT:

Enter the number of vertices and graph data

4

(1,1):999

(1,2):1

(1,3):5

(1,4):2

(2,1):1

(2,2):999

(2,3):999

(2,4):999

(3,1):5

(3,2):999

(3,3):999

(3,4):3

(4,1):2

(4,2):999

(4,3):3

(4,4):999

The edges of spanning tree are

1:edge(1,2)=1 2:edge(1,4)=2 3:edge(4,3)=3

Minimum cost is 6

……………………………………………………………………………………………………...

**5.Develop an optimal route for a scenario where a person wants to buy a ticket to a baseball game. Along**

**the way from house to reaching the destination, the person using it is a toll road, and has to pay a certain**

**amount of money.**

……………………………………………………………………………………………………….

PROGRAM :

#include<stdio.h>

int n,cost[10][10],dist[10];

void readmatrix(int);

int min(int,int);

void shortest\_path(int,int);

int main()

{

int i,s;

printf("Enter the number of vertices\n");

scanf("%d",&n);

printf("Enter the cost adjecency matrix(999 for infinite)\n");

readmatrix(n);

printf("Enter the source vertex\n");

scanf("%d",&s);

shortest\_path(n,s);

for(i=1;i<=n;i++)

printf("The shortest path between vertex %d to %d is %d\n",s,i,dist[i]);

return 0;

}

int min(int a,int b)

{

return a>b?b:a;

}

void readmatrix(int n)

{

int i,j;

for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

{

printf("(%d,%d):",i,j);

scanf("%d",&cost[i][j]);

if(cost[i][j]==0)

cost[i][j]=999;

}

}

void shortest\_path(int n,int s)

{

int vis[10],c,u,i,k;

for(i=1;i<=n;i++)

{

vis[i]=0;

dist[i]=cost[s][i];

}

dist[s]=0;

vis[s]=1;

for(k=1;k<=n;k++)

{

c=999;

for(i=1;i<=n;i++)

if(vis[i]==0)

{

if(dist[i]<=c)

{

c=dist[i];

u=i;

}

}

vis[u]=1;for(i=1;i<=n;i++)

dist[i]=min(dist[i],dist[u]+cost[u][i]);

}

}

OUTPUT :Enter the number of vertices

5

Enter the cost adjecency matrix(999 for infinite)

(1,1):999

(1,2):1

(1,3):2

(1,4):999

(1,5):999

(2,1):1

(2,2):999

(2,3):3

(2,4):4

(2,5):999

(3,1):2

(3,2):3

(3,3):999

(3,4):5

(3,5):6

(4,1):999

(4,2):4

(4,3):5

(4,4):999

(4,5):6

(5,1):999

(5,2):999

(5,3):6

(5,4):6

(5,5):999

Enter the source vertex

1

The shortest path between vertex 1 to 1 is 0

The shortest path between vertex 1 to 2 is 1

The shortest path between vertex 1 to 3 is 2

The shortest path between vertex 1 to 4 is 5

The shortest path between vertex 1 to 5 is 8

………………………………………………………………………………………………………..

**6a. Design and implement a Program to solve All-Pairs Shortest Paths problem using Floyd's algorithm.**

**b. Design and implement a Program to find the transitive closure using Warshal's algorithm.**

……………………………………………………………………………………………………………………..

PROGRAM :

#include <stdio.h>

int min(int,int);

void floyd(int [][10],int);

int main()

{

int a[10][10],n,i,j;

printf("\nEnter the number of vertices\n");

scanf("%d",&n);

printf("Enter the cost matrix\n");

for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

scanf("%d",&a[i][j]);

floyd(a,n);

printf("Shortest path matrix is \n");

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

printf("%d\t",a[i][j]);

printf("\n");

}

return 0;

}

int min(int a,int b)

{

return (a<b?a:b);

}

void floyd(int w[10][10],int n)

{

int i,j,k;

#pragma omp parallel for private(i,j,k) shared(w)

for(k=1;k<=n;k++)

for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

w[i][j]=min(w[i][j],w[i][k]+w[k][j]);

}

OUTPUT :

Enter the number of vertices

4

Enter the cost matrix

0 1 3 6

1 0 2 3

3 2 0 1

6 3 1 0

Shortest path matrix is

0 1 3 4

1 0 2 3

3 2 0 1

4 3 1 0

………………………………………………………………………………………………………

**6b.Design and implement a Program to find the transitive closure using Warshal's algorithm.**

**…………………………………………………………………………………………………………………………**

#include<stdio.h>

void warshall(int a[10][10],int);

int main()

{

int i,j,n,a[10][10];

printf("Enter the number of vertices\n");

scanf("%d",&n);

printf("Enter the adjacency matrix\n");

for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

{

printf("(%d,%d):",i,j);

scanf("%d",&a[i][j]);

}

warshall(a,n);

printf("All pair distance matrix\n");

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

printf("%d\t",a[i][j]);

printf("\n");

}

return 0;

}

void warshall(int a[10][10],int n)

{

int i,j,k;for(k=1;k<=n;k++)

for(j=1;j<=n;j++)

for(i=1;i<=n;i++)

a[i][j]=a[i][j] || (a[i][k] && a[k][j]);

}

OUTPUT :

Enter the number of vertices

4

Enter the adjacency matrix

(1,1):0

(1,2):1

(1,3):0

(1,4):0

(2,1):0

(2,2):0

(2,3):0

(2,4):1

(3,1):1

(3,2):0

(3,3):1

(3,4):0

(4,1):0

(4,2):0

(4,3):0

(4,4):0

All pair distance matrix

0 1 0 1

0 0 0 1

1 1 1 1

0 0 0 0

………………………………………………………………………………………………………..

**7. The owner of a gourmet coffee shop wishes to mix a 10-pound bag of coffee using various types of coffee**

**beans in such a way to produce the coffee blend at the maximum cost. The weights of the objects in the**

**problem correspond to the quantity in pounds available of each type of coffee bean. The value of each**

**quantity of coffee beans is the total cost of that quantity in rupees. Apply the Knapsack algorithm to**

**maximize the profit.**

**…………………………………………………………………………………………………………………………**

PROGRAM :

#include<stdio.h>

int v[20][20];

int max\_value(int,int);

int main()

{

int i,j,p[20],w[20],n,max;

printf("\nEnter the number of items\n");

scanf("%d",&n);

for(i=1;i<=n;i++)

{

printf("Enter the weight and profit of the item %d:",i);

scanf("%d %d",&w[i],&p[i]);

}

printf("\nEnter the capacity of knapsack\n");

scanf("%d",&max);

for(i=0;i<=n;i++)

v[i][0]=0;

for(i=0;i<=max;i++)

v[0][i]=0;

for(i=1;i<=max;i++)

for(j=1;j<=max;j++)

{

if(w[i]>j)

v[i][j]=v[i-1][j];

else

v[i][j]=max\_value(v[i-1][j],v[i-1][j-w[i]]+p[i]);

}

printf("The table is \n");

for(i=0;i<=n;i++)

{

for(j=0;j<=max;j++)

printf("%d\t",v[i][j]);

printf("\n");

}

printf("The max profit is %d",v[n][max]);

printf("most profitable subset is :{");

j=max;

for(i=n;i>=1;i--)

if(v[i][j]!=v[i-1][j])

{

printf("item%d:1\t",i);

j=j-w[i];

}

else

printf("item%d:0\t",i);

printf("}\n");

return 0;

}

int max\_value(int a,int b)

{

return (a>b?a:b);

}

OUTPUT:

Enter the number of items

4

Enter the weight and profit of the item 1:2 12

Enter the weight and profit of the item 2:1 10

Enter the weight and profit of the item 3:3 20

Enter the weight and profit of the item 4:2 15

Enter the capacity of knapsack

5

The table is

0 0 0 0 0 0

0 0 12 12 12 12

0 10 12 22 22 22

0 10 12 22 30 32

0 10 15 25 30 37

The max profit is 37most profitable subset is :{item4:1 item3:0 item2:1 item1:1}

…………………………………………………………………………………………………………

**8.Design an application for drilling an optimal printed circuit board. To drill two holes of different diameters consecutively, the head of the machine has to move to a toolbox and change the drilling equipment. This is quite time consuming. Thus, it is clear that one has to choose some diameter, drill all holes of the same diameter, change the drill, drill the holes of the next diameter, etc. Thus, this drilling problem has to minimize the travel time for the machine head. Find the optimal time to drill the circuit board.**

**…………………………………………………………………………………………………………………………**

PROGRAM:

#include<stdio.h>

int a[10][10],visited[10],n,cost=0;

int least(int c);

void mincost(int city)

{

int i,ncity;

visited[city]=1;

printf("%d-->",city);

ncity=least(city);

if(ncity==999)

{

ncity=1;

printf("%d",ncity);

cost+=a[city][ncity];

return;

}

mincost(ncity);

}

int least(int c)

{

int i,nc=999;

int min=999,kmin;

for(i=1;i<=n;i++)

{

if((a[c][i]!=0)&&(visited[i]==0))

if(a[c][i]<min)

{

min=a[i][1]+a[c][i];

kmin=a[c][i];

nc=i;

}

}

if(min!=999)

cost+=kmin;

return nc;

}

void main()

{

int i,j;

printf("enter No. of cities:\n");

scanf("%d",&n);

printf("enter the cost matrix\n");

for(i=1;i<=n;i++)

{

printf("enter elements of row:%d\n",i );

for(j=1;j<=n;j++)

scanf("%d",&a[i][j]);

visited[i]=0;

}

printf("the cost list is\n");

for(i=1;i<=n;i++)

{

printf("\n\n");

for(j=1;j<=n;j++)

printf("\t%d",a[i][j]);

}

printf("\n\n the path is :\n\n");

mincost(1);

printf("\n\n minimum cost:");

printf("%d",cost);

}

OUTPUT :

enter No. of cities:

4

enter the cost matrix

enter elements of row:1

0 1 3 6

enter elements of row:2

1 0 2 3

enter elements of row:3

3 2 0 1

enter elements of row:4

6 3 1 0

the cost list is

0 1 3 6

1 0 2 3

3 2 0 1

6 3 1 0

the path is :

1-->2-->4-->3-->1

minimum cost:8

………………………………………………………………………………………………………….

**9.Design and implement for a given chess board having N×N cells, place *N* queens on the board in such a way that no queen attacks any other queen. If it is possible to place all the *N* queens in such a way that no queen attacks another queen, then print *N* lines having *N* Queens. If there is more than one solution of placing the queens, print all of them. If it is not possible to place all *N* queens in the desired way, then print "Not possible".**

**…………………………………………………………………………………………………………………………**

PROGRAM :

#include<stdio.h>

#include<stdlib.h>

int count=0,x[5];

void n\_queen(int,int);

int place(int,int);

int main()

{

int n;

printf("Enter the number of queens\n");

scanf("%d",&n);

n\_queen(1,n);

if(count==0)

printf("No solutions are found\n");

else

printf("Number of solutions found are %d \n",count);

return 0;

}

int place(int k,int i)

{

int j;

for(j=1;j<k;j++)

if((x[j]==i)||((abs(x[j]-i))==(abs(j-k))))

return 0;

return 1;

}

void n\_queen(int k,int n)

{

int i,j,p;

for(i=1;i<=n;i++){

if(place(k,i))

{

x[k]=i;

if(k==n)

{

count++;

printf("solution %d\n",count);

for(j=1;j<=n;j++)

{

for(p=1;p<=n;p++)

if(x[j]==p)

printf("$\t");

else

printf("0\t");

printf("\n");

}

printf("\n");

}

else

n\_queen(k+1,n);

}

}

}

OUTPUT:

Enter the number of queens

4

solution 1

0 $ 0 0

0 0 0 $

$ 0 0 0

0 0 $ 0

solution 2

0 0 $ 0

$ 0 0 0

0 0 0 $

0 $ 0 0

………………………………………………………………………………………………………………………

**VIVA QUESTIONS**

1. **What is an algorithm?**

An algorithm is a sequence of unambiguous instructions for solving a problem. i.e., for obtaining a required output for any legitimate input in a finite amount of time.

**2. What are important problem types? (or) Enumerate some important types of problems.**

1. Sorting 2. Searching 3. Numerical problems 4. Geometric problems 5. Combinatorial Problem 6. Graph Problems 7. String processing Problems

**3. Name some basic Efficiency classes**

1. Constant 2. Logarithmic 3. Linear 4. Nlogn 5. Quadratic 6. Cubic 7. Exponential 8. Factorial

**4. What are algorithm design techniques?**

Algorithm design techniques ( or strategies or paradigms) are general approaches to solving problems algorithmatically, applicable to a variety of problems from different areas of computing. General design techniques are: (i) Brute force (ii) divide and conquer (iii) decrease and conquer (iv) transform and concquer (v) greedy technique (vi) dynamic programming (vii) backtracking (viii) branch and bound

**5. How is an algorithm’s time efficiency measured?**

Time efficiency indicates how fast the algorithm runs. An algorithm’s time efficiency is measured as a function of its input size by counting the number of times its basic operation(running time) is executed. Basic operation is the most time consuming operation in the algorithm’s innermost loop.

**6. How is the efficiency of the algorithm defined?**

The efficiency of an algorithm is defined with the components.

(i) Time efficiency -indicates how fast the algorithm runs

(ii) Space efficiency -indicates how much extra memory the algorithm needs.

**7. What are the characteristics of an algorithm?**

Every algorithm should have the following five characteristics

(i) Input

(ii) Output

(iii) Definiteness

(iv) Effectiveness

(v) Termination

Therefore, an algorithm can be defined as a sequence of definite and effective instructions, which

terminates with the production of correct output from the given input.

**8. Write general plan for analyzing non-recursive algorithms.**

i. Decide on parameter indicating an input’s size.

ii. Identify the algorithm’s basic operation

iii. Checking the no.of times basic operation executed depends on size of input.

iv. Set up sum expressing the no.of times the basic operation is executed. depends on some additional property,then best,worst,avg.cases need to be investigated (establishing order of growth)

**9. Define the terms: pseudocode, flow chart**

A pseudocode is a mixture of a natural language and programming language like constructs. A pseudocode is usually more precise than natural language. A flowchart is a method of expressing an algorithm by a collection of connected geometric shapes containing descriptions of the algorithm’s steps.

**10. write general plan for analyzing recursive algorithms.**

i. Decide on parameter indicating an input’s size.

ii. Identify the algorithm’s basic operation

iii. Checking the no.of times basic operation executed depends on size of input.if it depends on some additional property,then best,worst,avg.cases need to be investigated of times the basic operation is executed

iv. Set up the recurrence relation,with an appropriate initial condition,for the number

v. Solve recurrence (establishing order of growth)

**11. Define the divide an conquer method.**

Given a function to compute on ‘n’ inputs the divide-and-comquer strategy suggests splitting the inputs in to’k’ distinct susbsets, 1<k <n, yielding ‘k’ subproblems. The subproblems must be solved recursively, and then a method must be found to combine subsolutions into a solution of the whole.

**12. What is Merge sort?**

Merge sort is an O (n log n) comparison-based sorting algorithm. Where the given array is divided into two equal parts. The left part of the array as well as the right part of the array is sorted recursively. Later, both the left sorted part and right sorted part are merged into a single sorted array.

**13. What is general divide and conquer recurrence?**

Time efficiency T(n)of many divide and conquer algorithms satisfies the equation T(n)=a.T(n/b)+f(n).This is the general recurrence relation.

**14. Describe the recurrence relation for merge sort?**

If the time for the merging operation is proportional to n, then the computing time of merge sort is described by the recurrence relationT(n) = a n = 1, a a constant 2T (n/2) + n n >1, c a constant

**15. The relation between order of growth of functions**

O(1) < O(log n) < O(n) < O(n \* log n) < O(n2) < O(n3) < O(2n)

**16. Asymptotic notations**

Big Oh(Worst Case), Big Theta (Average Case), Big Omega(Best Case)

**17) What is the time complexity of linear search?**

**Θ(n)**

**18) What is the time complexity of binary search?**

Θ(log2n)

**19) What is the major requirement for binary search?**

The given list should be sorted.

**20) What is binary search?**

It is an efficient method of finding out a required item from a given list, provided the list is in order.

The process is:

1. First the middle item of the sorted list is found.

2. Compare the item with this element.

3. If they are equal search is complete.

4. If the middle element is greater than the item being searched, this process is repeated in the upper half of the list.

5. If the middle element is lesser than the item being searched, this process is repeated in the lower half of the list.

**21) What is parental dominance?**

The key at each node is greater than or equal to the keys at its children.

**22) What is heap?**

A heap can be defined as a binary tree with keys assigned to its nodes ( one key per node) provided the following two conditions are met:

1 The tree’s shape requirement – The binary tree is essentially complete ( or simply complete), that is, all its levels are full except possibly the last level, where only some rightmost leaves may

be missing.

2. The parental dominance requirement – The key at each node is greater than or equal to the

keys at its children

**23) What is height of Binary tree?**

It is the longest path from the root to any leaf.

**24) What is the time complexity of heap sort?**

Θ( n logn)

**25) Who invented Merge Sort?**

John Von Neumann in 1945.

**26) On which paradigm is it based?**

Merge-sort is based on the divide-and-conquer paradigm.

**27) What is the time complexity of merge sort?**

n log n.

**28) What is the space requirement of merge sort?**

Space requirement: Θ(n) (not in-place).

**30) When do you say an algorithm is stable and in place?**

Stable – if it retains the relative ordering. In – place if it does not use extra memory.

**31) Who invented Dijkstra’s Algorithm?**

Edsger Dijkstra invented this algorithm.

**32) What is the other name for Dijkstra’s Algorithm?**

Single Source Shortest Path Algorithm.

**33) Which technique the Dijkstra’s algorithm is based on?**

Greedy Technique.

**34) What is the purpose of Dijkstra’s Algorithm?**

To find the shortest path from source vertex to all other remaining vertices

**35) Name the different algorithms based on Greedy technique.**

Prim’s Algorithm, Kruskal’s algorithm

**36) What is the constraint on Dijkstra’s Algorithm?**

This algorithm is applicable to graphs with non-negative weights only.

**37) What is a Weighted Graph?**

A weighted graph is a graph with numbers assigned to its edges. These numbers are called weights or costs.

**38) What is a Connected Graph?**

A graph is said to be connected if for every pair of its vertices u and v there is a path from u to v.

**39) What is the time complexity of Dijkstra’s algorithm?**

For adjacency matrix, It is O(IVI\*IVI)

For adjacency list with edges stored as min heap, it is O(IEIlogIVI)

**40) Differentiate b/w Traveling Salesman Problem(TSP) and Dijkstra’s Algorithm.**

In TSP, given n cities with known distances b/w each pair , find the shortest tour that passes through all the cities exactly once before returning to the starting city.

In Dijkstra’s Algorithm, find the shortest path from source vertex to all other remaining vertices

**41) Differentiate b/w Prim’s Algorithm and Dijkstra’s Algorithm?**

Prim’s algorithm is to find minimum cost spanning tree.

Dijkstra’s algorithm is to find the shortest path from source vertex to all other remaining vertices.

**42) What are the applications of Dijkstra’s Algorithm?**

It finds its application even in our day to day life while travelling. They are helpful in routing applications.

**43) Who was the inventor of the Quicksort?**

C.A.R.HOARE, a British Scientist.

**44) Which design strategy does Quicksort uses?**

Divide and Conquer

**45) What is Divide and Conquer Technique?**

(I) Divide the instance of a problem into two or more smaller instances

(II) Solve the smaller instances recursively

(III) Obtain solution to original instances by combining these solutions

**46) What is the another name for Quicksort?**

Partition Exchange Sort

**47)Is Quicksort Stable as well as In place?**

Not Stable but In place.

**48) When do you say that a Quick sort having best case complexity?**

When the pivot exactly divides the array into equal half

**49) When do you say that Quick sort having worst case complexity?**

When any one of the subarray is empty

**50) How many more comparisons are needed for average case compared to best case?**

38% more

**51) when do you say that the pivot element is in its final position?**

When all the elements towards the left of pivot are <= pivot and all the elements towards the right of pivot are >= pivot.

**52) What technique does kruskal’s algorithm follow.**

Greedy technique

**53) What is the time complexity of kruskal algorithm.**

With an efficient sorting algorithm, the time efficiency of an kruskal’s algorithm will be inO(|E|log|E|).

**54) Who is the inventor of kruskal’s algorithm.**

Joseph Kruskal.

**55) Which technique is used to solve BFS & DFS problems?**

Decrease and Conquer

**56) What is DFS traversal?**

Consider an arbitrary vertex v and mark it as visited on each iteration, proceed to an unvisited vertex w adjacent to v. we can explore this new vertex depending upon its adjacent information. This process continues until dead end is encountered.(no adjacent unvisited vertex is available). At the dead end the algorithm backs up by one edge and continues the process of visiting unvisited vertices. This process continues until we reach the starting vertex.

**57) What are the various applications of BFS & DFS tree traversal technique?**

Application of BFS

Checking connectivity and checking acyclicity of a graph.

Check whether there is only one root in the BFS forest or not.

If there is only one root in the BFS forest, then it is connected graph otherwise disconnected graph.

For checking cycle presence, we can take advantage of the graph's representation in the form of a BFS forest, if the latter vertex does not have cross edge, then the graph is acyclic.

Application of DFS

To check whether given graph is connected or not.

To check whether the graph is cyclic or not.

To find the spanning tree.

Topological sorting.

**58) Which data structures are used in BFS & DFS tree traversal technique?**

We use Stack data structure to traverse the graph in DFS traversal. We use queue data structure to traverse the graph in BFS traversal.

**59) What is Dynamic Programming?**

It is a technique for solving problems with overlapping sub problems.

**60) What does Dynamic Programming have in common with Divideand-Conquer?**

Both solve the problems by dividing the problem into sub problems. Using the solutions of sub problems, the solutions for larger instance of the problem can be obtained.

**61) What is a principle difference between the two techniques?**

Only one instance of the sub problem is computed & stored. If the same instance of sub problem

is encountered, the solution is retrieved from the table and never recomputed. Very precisely re -

computations are not preformed.

**63) What is a spanning tree ?**

A spanning tree of a weighted connected graph is a connected acyclic(no cycles) subgraph (i.e. a tree)that contains all the vertices of a graph and number of edges is one less than number of vertices.

**64) What is a minimum spanning tree ?**

Minimum spanning tree of a connected graph is its spanning tree of smallestweight.

**65) Prim’s algorithm is based on which technique.**

Greedy technique.

**66) Name some of the application greedy technique**

They are helpful in routing applications: In the case of network where large number of computers are connected, to pass the data from one node to another node we can find the shortest distance easily by this algorithm. Communication Networks: Suppose there are 5 different cities and we want to establish computer connectivity among them, then we take into the account of cost factor for communication links. Building a road network that joins n cities with minimum cost.

**67) Does Prim’s Algorithm always yield a minimum spanning tree ?**

Yes

**68) What is the purose of Floyd’s algoreithm?**

- To find the all pair shortest path.

**69) What is the time efficiency of floyd’s algorithm?**

- It is same as warshall’s algorithm that is θ(n3).

**70) What is shortest path?**

- It is the path which is having shortest length among all possible paths.

**71) warshall’s algorithm is optimization problem or non-optimization problem ?**

Non-optimization problem

**72) For what purpose we use Warshall’s algorithm?**

Warshall’s algorithm for computing the transitive closure of a directed graph

**73) Warshall’s algorithm depends on which property?**

Transitive property

**74) what is the time complexity of Warshall’s algorithm?**

Time complexity of warshall’s algorithm is θ(n3).

**75) what is state space tree?**

Constructing a tree of choices being made and processing is known as state-spacetree. Root represents an initial state before the search for solution begins.

**76) what are promising and non-promising state-space-tree?**

Node which may leads to solution is called promising. Node which doesn’t leads to solution is called non-promising.

**77) What are the requirements that are needed for performing Backtracking?**

Complex set of constraints. They are:

i. Explicit constraints.

ii. Implicit constraints.

**GREEDY METHOD**

**1. Explain the greedy method.**

Greedy method is the most important design technique, which makes a choice that looks best atthat moment. A given ‘n’ inputs are required us to obtain a subset that satisfies some constraints that is the feasible solution. A greedy method suggests that one candevice an algorithm that works in stages considering one input at a time.

**2. Define feasible and optimal solution.**

Given n inputs and we are required to form a subset such that it satisfies some given constraints then such a subset is called feasible solution. A feasible solution either maximizes or minimizes the given objective function is called as optimal solution.

**3. What are the constraints of knapsack problem?**

To maximize ∑pixi

The constraint is : ∑wixi ≥ m and 0 ≤ xi ≤ 1 1≤ i ≤ nwhere m is the bag capacity, n is the number of objects and for each object i wi and pi are the weight and profit of object respectively.

**4. Specify the algorithms used for constructing Minimum cost spanning tree.**

a) Prim’s Algorithm

b) Kruskal’s Algorithm

**5. State single source shortest path algorithm (Dijkstra’s algorithm).**

For a given vertex called the source in a weigted connected graph,find shotrtest paths to all its other vertices.Dijikstra’s algorithm applies to graph with non-negative weights only.

**6. State efficiency of prim’s algorithm.**

O(|v|2) (WEIGHT MATRIX AND PRIORITY QUEUE AS UNORDERED ARRAY)

O(|E| LOG|V|) (ADJACENCY LIST AND PRIORITY QUEUE AS MIN-HEAP)

**7. State Kruskal Algorithm.**

The algorithm looks at a MST for a weighted connected graph as an acyclic subgraph with |v|-1 edges for which the sum of edge weights is the smallest.

**8. State efficiency of Dijkstra’s algorithm.**

O(|v|2) (WEIGHT MATRIX AND PRIORITY QUEUE AS UNORDERED ARRAY)

O(|E| LOG|V|) (ADJACENCY LIST AND PRIORITY QUEUE AS MIN-HEAP)

**DYNAMIC PROGRAMMING**

**1. Define multistage graph**

A multistage graph G =(V,E) is a directed graph in which the vertices are partitioned into K>=2 disjoint sets Vi,1<=i<=k.The multi stage graph problem is to find a minimum cost paths from s(source ) to t(sink)

Two approach(forward and backward)

**2. Define All pair shortest path problem**

Given a weighted connected graph, all pair shortest path problem asks to find the lengths of shortest paths from each vertex to all other vertices.

**3.Define floyd’s algorithm**

To find all pair shortest path

**4. State the time efficiency of floyd’s algorithm**

O(n3)

It is cubic