1. **Design and implement C/C++ Program to find Minimum Cost Spanning Key of a givenconnected undirected graph using Kruskal's Algorithm.**

```
#include<stdio.h>
#define INFI 99
#define MAX 10
int a[MAX][MAX],b[MAX][MAX],n,cost=0;
void findmin(int * v1,int *v2) //finding the edge having minimum weight.
{
int edge=INFI,i,j;
for(i=1;i<=n;i++)
for(j=i+1;j<=n;j++)
if(a[i][j]>0 && a[i][j]<edge)
{
edge=a[i][j];
*v1=i;
*v2=j;
}
}
void update(int root[],int v1,int v2)
{
int temp,i;
temp=root[v2];
for(i=1;i<=n;i++)
{
if(root[i]==temp)
root[i]=root[v1];
}
}
void kruskal()
{
int i ,v1,v2,root[MAX],edge,count=0;
for(i=1;i<=n;i++)
root[i]=i;
i=0;
```

```
while(i!=n-1)
{
findmin(&v1,&v2);
edge=a[v1][v2];
a[v1][v2]=a[v2][v1]=0; //do not select the same edge on next time.
if(root[v1]!=root[v2])
{
printf("(%d,%d)\n",v1,v2);
update(root,v1,v2);
cost+=edge;
i++;
}
}
}
int main()
{
int i,j;
printf("\n Enter the number of vertices : ");
scanf("%d",&n);
printf("\n Enter the weighted graph : \n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d",&a[i][j]);
printf("\n Edges of spanning tree are:\n");
kruskal();
printf("\n Minimum cost=%d:",cost);
return(0);

 }
```

Output:

```
aiml-admin@aimladmin-HP:~/ada$ ./a.out

 Enter the number of vertices : 4

 Enter the weighted graph :
0 2 99 2
2 0 3 1
99 3 0 5
2 1 5 0

 Edges of spanning tree are:
(2,4)
(1,2)
(2,3)

 Minimum cost=6:aiml-admin@aimladmin-HP:~/ada$
```

```
aiml-admin@aimladmin-HP:~/ada$ ./a.out

 Enter the number of vertices : 4

 Enter the weighted graph :
0 1 99 2
1 0 6 2
99 6 0 5
2 2 5 0

 Edges of spanning tree are:
(1,2)
(1,4)
(3,4)

 Minimum cost=8:aiml-admin@aimladmin-HP:~/ada$
```

Review Questions:

1. **What is Kruskal's algorithm used for?**
2. **What is the main idea behind Kruskal's algorithm?**
3. **What is a minimum spanning tree (MST)?**

2. Design and implement C/C++ Program to find Minimum Cost Spanning Key of a givenconnected undirected graph using Prim's Algorithm.

```
#include<stdio.h>
void prims( );
int nearest[10], cost[10][10], t[10][2], i, j, n, k, min, u, mincost = 0;
void main( )
{
printf("\n\n************** PRIMS ALGORITHM **************\n\n");
printf("Enter the number of nodes\n");
scanf("%d", &n);
printf("\n\nEnter the cost matrix\n");
for(i=1; i<=n; i++)
{
for(j=1; j<=n; j++)
{
scanf("%d", &cost[i][j]);
}
}
printf("\n\nThe entered cost matrix is\n");
for(i=1; i<=n; i++)
{
for(j=1; j<=n; j++)
{
printf("%d\t", cost[i][j]);
}
printf("\n");
}
printf("\n\nMinimum Spanning Tree Edges and their costs are\n");
prims( );
printf("\n\nThe minimum spanning tree cost is %d", mincost);
printf("\n\n******* ******************************** *******");
}
void prims()
{
for(i=2; i<=n; i++)
nearest[i]=1;  //Initializing nearest nodes to all the nodes other than first node as 1
nearest[1]=0;  //Marking the first node as visited
for(i=1; i<n; i++)
{
min=99;
for(j=1;j<=n;j++)  //find a node j which can be visited with minimum cost
{
/*if the node j is not visited and if the cost of j to its nearest node is less than minimum*/
if(nearest[j]!=0 && cost[j][nearest[j]]<min)
{
min=cost[j][nearest[j]];
u=j;
}

t[i][1] = u; // marking the node j
```

```
t[i][2] = nearest[u];  //marking the node from where we visited j
mincost += min; //updating the minimum cost
nearest[u] = 0; //indicating that node u=j has been visited
/* if any of the remaining unvisited node can be reached from newly visited node with minimum cost
then the newly visited node becomes its nearest node*/
for(k=1; k<=n; k++)
{
if(nearest[k] != 0 && cost[k][nearest[k]] > cost[k][u])
nearest[k] = u;
}
printf("%d) edge (%d,%d) , cost %d\n", i, t[i][1], t[i][2], min);
}
}
```

Output:

```
aiml-admin@aimladmin-HP:~/ada$ ./a.out

 Enter the number of vertices: 4

 Enter the cost matrix?
0 2 99 2
2 0 5 3
99 5 0 4
2 3 4 0


 The edges of this minimum cost spanning tree are
 (1,2)
 (1,4)
 (4,3)

 Minimum cost Spanning Tree is:8aiml-admin@aimladmin-HP:~/ada$
```

Review Questions:

1. **What is Prim's algorithm used for?**

3. a) Design and implement C/C++ Program to solve All-Pairs Shortest Paths Problem using Floyd's algorithm.

b) Design and implement C/C++ Program to find the transitive closure using Warshall'salgorithm

```
#include<stdio.h>
#include <stdlib.h>
#define MAX 10
#define min(c,d) (c<d?c:d)
int dist[MAX][MAX],n;
void floyd()
{
 int i,j,k;
for(k=1;k<=n;k++) // record the lengths of shortest path
for(i=1;i<=n;i++)
 for(j=1;j<=n;j++)
 dist[i][j]=min(dist[i][j],dist[i][k]+dist[k][j]);
}
void main()
{
 int i, j;
 printf("Enter the number of vertices :\n");
 scanf("%d",&n);
 printf("Enter the distance matrix\n"); //read distance matrix
 for(i=1;i<=n;i++)
 for( j=1;j<=n;j++)
 scanf("%d",&dist[i][j]);
 floyd();
 printf("\nAll pairs shortest path matrix is :\n");
 for( i=1;i<=n;i++)
 {
 for( j=1;j<=n;j++)
 {
    printf("%d\t",dist[i][j]);
}
printf("\n");
```

```
 }
 }
```

b. Warshall's Algorithm

```
#include<stdio.h>
 #define MAX 10
int D[MAX][MAX],n;
void warshall()
{
int i,j,k;
for(k=1;k<=n;k++)
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
D[i][j] = D[i][j] || ( D[i][k] && D[k][j] );
}
void main()
{
int i, j;
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++)
scanf("%d",&D[i][j]);
warshall();
printf("Trasitive closure of digraph is :\n");for(
i=1;i<=n;i++)
{
for( j=1;j<=n;j++)
printf("%d\t",D[i][j]);
printf("\n") ;
}
}
```

Output:

```
aiml-admin@aimladmin-HP:~/ada$ gcc -fopenmp program3a.c
aiml-admin@aimladmin-HP:~/ada$ ./a.out
Enter the number of nodes: 4
Enter the cost adjacency matrix:
0 99 3 99
2 0 99 99
99 7 0 1
6 99 99 0

Total Threads Used are: 12
All-Pairs Shortest Paths is as follows:
0       10      3       4
2       0       5       6
7       7       0       1
6       16      9       0

The time taken to perform Floyd's Algorithm is: 0.035606
aiml-admin@aimladmin-HP:~/ada$
```

```
aiml-admin@aimladmin-HP:~$ gcc w.c
aiml-admin@aimladmin-HP:~$ ./a.out
Enter the number of vertices :
4
Enter the adjacency matrix
0 1 1 0
0 0 0 0
1 0 0 1
1 0 0 0
Trasitive closure of digraph is :
1       1       1       1
0       0       0       0
1       1       1       1
1       1       1       1
aiml-admin@aimladmin-HP:~$
```

Review Questions:

1. **What is Floyd/Warshall's algorithm?**
2. **What is the time complexity of Floyd/Warshall's algorithm?**
3. **What is the output of Floyd's/Warshall's algorithm?**
4. **What is the transitive closure of a graph?**

4. Design and implement C/C++ Program to find From a given vertex in a weighted connectedgraph to other vertices using Dijkstra's algorithm.

```
#include<stdio.h>
#include<stdio.h>
void dij( );
void printpath( );
int sv, i, j, n, w, v=0, t, min, count, dist[10], visited[10],cost[10][10], path[10];
void main( )
{
printf("\n\n********* DIJKSTRA'S ALGORITHM *********\n\n");
printf("Enter the number of nodes: ");
scanf("%d", &n);
printf("\n\nEnter the cost matrix\n");
for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
                scanf("%d", &cost[i][j]);
printf("\n\nThe entered cost matrix is\n");
for(i=1;i<=n;i++)
{
        for(j=1;j<=n;j++)
        {
                printf("%d\t", cost[i][j]);
        }
        printf("\n");
        }
printf("\n\nEnter the source vertex: ");
scanf("%d", &sv);
dij( );
printpath( );
printf("\n\n********* ************************ *********");
}
```

```
void dij( )
{
for(i=1; i<=n; i++)
{
        visited[i]=0;
        dist[i] = cost[sv][i];
        if(cost[sv][i] == 999)
                path[i] = 0;
        else
                path[i] = sv;
}
visited[sv]=1;
count = 1;
while(count<=n-1)
{
        min = 99;
        for(w=1; w<=n; w++)
        if(dist[w] < min && !visited[w])
        {
                min = dist[w];
                v = w;
        }
        visited[v] = 1;
        count++;
        for(w=1; w<=n; w++)
        {
                if(dist[w] > dist[v] + cost[v][w])
                {
                        dist[w] = dist[v] + cost[v][w];
                        path[w] = v;
                }
        }
```

```
  }
    }
void printpath( )
{
for(w=1; w<=n; w++)
{
if(visited[w] == 1 && w != sv)
{
printf("\n\nThe shortest distance between %d->%d = %d", sv, w, dist[w]);
t=path[w];
printf("\nThe path is:\n");
printf("%d", w);
while(t != sv)
{
printf("<-->%d", t);
t=path[t];
}
printf("<-->%d", sv);
}
}
        }
```

Output:

```
aiml-admin@aimladmin-HP:~/ada$ gcc p5.c
aiml-admin@aimladmin-HP:~/ada$ ./a.out

 Enter the number of vertices in a graph : 4

 Enter the weight matrix?
0 2 5 1
2 0 1 2
5 1 0 3
1 2 3 0

 enter the source vertex: 2


 Shortest paths from vertex 2 are

vertext->1 length:2    path: 1<--2
vertext->3 length:1    path: 3<--2
vertext->4 length:2    path: 4<--2
aiml-admin@aimladmin-HP:~/ada$
```

Review Questions:

1. **What is Dijkstra's algorithm?**
2. **What is the time complexity of Dijkstra's algorithm?**
3. **What is the purpose of the visited set in Dijkstra's algorithm?**
4. **What are some real-world applications of Dijkstra's algorithm?**

5. Design and implement C/C++ Program to obtain the Topological ordering of vertices in agiven digraph.

```
#include<stdio.h>
void topo( );
int ad[10][10], i, j, n, k;
void main( )
{
printf("\n\n********** TOPOLOGICAL SORTING *********\n\n");
printf("Enter the number of vertices\n");
scanf("%d", &n);
printf("\n\nEnter the adjacency matrix\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
scanf("%d", &ad[i][j]);
}
}
printf("\n\nThe entered adjacency matrix is\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
printf("%d\t", ad[i][j]);
}
printf("\n");
}
topo( );
}
void topo( ) //function definition
{
```

```
int v[10], in=1, flag=0, count=0, f=1;
while(f) //checking for all possibilities
{
count++;
for(i=1;i<=n;i++)
{
flag = 0;
for(j=1; j<=n; j++)
{
if(ad[j][i] != 0 || v[j]==i) //if there is no incoming edge or if the node is already visited
{
flag = 1;
break;
}
}
if(flag != 1)
{
v[in++] = i;
for(k=1; k<=n; k++)
ad[i][k] = 0;
}
}
if(count == n)
f = 0;
}
if(in < n)
printf("\n\nTopological ordering is not possible\n");
else
{
printf("\n\nTopological ordering is possible\n");
printf("\nOrdering is:\t");
for(i=1; i<=n; i++)
```

```
printf("%d\t", v[i]);
}
printf("\n*********************************************");
}
```

Output:

```
aiml-admin@aimladmin-HP:~$ ./a.out

 Enter the number of vertices: 3

 Enter the matrix of the digraph :
0 1 0
0 0 1
1 0 0

 No topological orderaiml-admin@aimladmin-HP:~$ ./a.out

 Enter the number of vertices: 4

 Enter the matrix of the digraph :
0 0 0 1
1 0 0 1
0 1 0 1
0 0 0 0

 Topological sequence is...
3       2       1       4       aiml-admin@aimladmin-HP:~$
```

Review Questions:

1. **What is a topological sort?**
2. **Why can't a graph with cycles be topologically sorted?**
3. **What types of graphs can have a topological ordering?**
4. **What are the common algorithms used for topological sorting?**

6. Design and implement C/C++ Program to solve 0/1 Knapsack problem using DynamicProgramming method.

```
#include <stdio.h>
int my_max(int a, int b)
{
return (a > b) ? a : b;
}
int val[20],wt[20],n,c,v[20][20];
//build a matrix(v[i][j]) with the weight bounds as the columns and the number of items as the
rows.
int knap()
{
 int i,j;
for(i=0;i<=n;i++)
for(j=0;j<=c;j++)
if(i==0||j==0)
v[i][j] = 0;
else
if(wt[i]>j)
v[i][j] = v[i-1][j];
else
v[i][j] =my_max(v[i-1][j],(v[i-1][j-wt[i]]+val[i]));
return v[n][c];
}
int main()
{
int opt,i,j;
printf("\nEnter the no of items in Knapsack : ");
scanf("%d",&n);
printf("\nEnter values (profit) of %d elements : ",n);
for(i=1;i<=n;i++)
scanf("%d",&val[i]);
printf("\nEnter weight of %d elements : ",n);
for(i=1;i<=n;i++)
```

```
scanf("%d",&wt[i]);
printf("\n Enter the capacity of Knapsack : ");
scanf("%d",&c);
opt = knap();
printf("\n\nCapacity");
for(j=0;j<=c;j++)
printf("%4d",j);
printf("\n");
for(i=0;i<=n;i++)
{
printf("\nItem-%2d:",i);
for(j=0;j<=c;j++)
printf("%4d",v[i][j]);
}
printf("\n\nOptimal solution is : %d",opt);
printf("\n\n The selected items are : ");
while(n>0) //best subset with weight at most knapsack size
{
if(v[n][c] != v[n-1][c])
{
printf("%d\t",n);c
= c - wt[n];
}n --;
}
return(0);
}
```

Output:

```
aiml-admin@aimladmin-HP:~$ ./a.out

Enter the no of items in Knapsack : 4

Enter values (profit) of 4 elements : 12 10 20 15

Enter weight of  4 elements : 2 1 3 2

 Enter the capacity of Knapsack : 5


Capacity   0   1   2   3   4   5

Item- 0:   0   0   0   0   0   0
Item- 1:   0   0  12  12  12  12
Item- 2:   0  10  12  22  22  22
Item- 3:   0  10  12  22  30  32
Item- 4:   0  10  15  25  30  37

Optimal solution is : 37

 The selected items are : 4     2       1       aiml-admin@aimladmin-HP:~$
```

Review Questions:

1. **What is the 0/1 Knapsack problem?**
2. **Why is it called the 0/1 Knapsack problem?**
3. **What is the objective of the 0/1 Knapsack problem?**
4. **What are some real-world applications of the 0/1 Knapsack problem?**

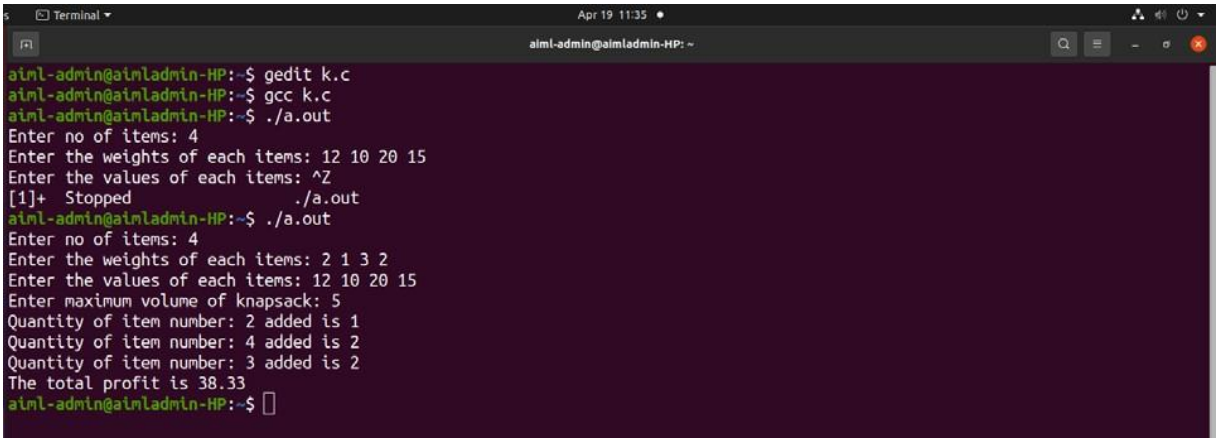
7.Design and implement C/C++ Program to solve discrete Knapsack and continuous Knapsackproblem using greedy approximation method.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define max(a,b) ((a>b)?a:b)
void knapsack( );
void optimal( );
int i, j, x[10], n, m, v[10][10], w[10], p[10], item=0;
void main( )
{
printf("\n**** KNAPSACK PROBLEM ****\n\n");
printf("Enter the total number of items: ");
scanf("%d", &n);
printf("\n\nEnter the weight of each item: ");
for(i=1;i<=n;i++)
scanf("%d", &w[i]);
printf("\n\nEnter the profit of each item: ");
for(i=1;i<=n;i++)
scanf("%d", &p[i]);
printf("\n\nEnter the knapsack capacity: ");
scanf("%d", &m);
knapsack( );
printf("\n\nThe contents of the knapsack table are\n");
for(i=0; i<=n; i++)
{
for(j=0; j<=m; j++)
{
printf("%d\t", v[i][j]);
}
```

```
  printf("\n");
  }
  optimal( ); //call optimal function
  }
  void knapsack( )  /*function to prepare the knapsack table*/
  {
  for(i=0; i<=n; i++)    // every individual item i
  {
  for(j=0; j<=m; j++)    // for the available knapsack capacity j
  {
  if(i==0 | |j==0) v[i][j]=0;
  else if(j < w[i]) v[i][j]=v[i-1][j];
  else v[i][j]=max(v[i-1][j], v[i-1][j-w[i]]+p[i]);
  }
}
}
void optimal( ) /*function to find the optimal solution*/
{
int i = n, j = m;
while( i != 0 && j != 0)
{
if(v[i][j] != v[i-1][j])
x[i] = 1; j = j-w[i];
i = i-1;
}
printf("\n\nOptimal solution is %d\n\n", v[n][m]);
printf("Selected items are: ");
for(i=1; i<= n;i++)
if(x[i] == 1)
{
printf("%d, ", i);
```

```
item=1; // flag item is set if there are any items that can be placed in Knapsack
}
printf("\b\b ");
if(item == 0)
printf("NIL\n\t Sorry ! No item can be placed in Knapsack\n");
printf("\n******** ****************************** *************");
}
```

Output:



Review Questions:

1. What is the difference between the Discrete Knapsack problem and the Continuous Knapsack problem?
2. What is a greedy algorithm?

8. Design and implement C/C++ Program to find a subset of a given set S = {sl, s2,. ,sn} of n positive integers whose sum is equal to a given positive integer d.

```
#include <stdio.h>
int main()
{
int i, selItem = 0, max_qty, availCap, n;
float sum = 0, unitProfit;
int weight[20],profit[20];
printf("Enter no of items: ");
scanf("%d", &n);
printf("Enter the weights of each items: ");
for (i = 0; i < n; i++)
{
        scanf("%d", &weight[i]);
}
printf("Enter the values of each items: ");
for (i = 0; i < n; i++) {
        scanf("%d", &profit[i]);
}
printf("Enter maximum volume of knapsack: ");
scanf("%d", &max_qty);
availCap = max_qty;
while (availCap > 0)
 {
        unitProfit = 0;
        for (i = 0; i < n; i++)
         {
                if (((float)profit[i]) / ((float)weight[i]) > unitProfit) {
                unitProfit = ((float)profit[i]) / ((float)weight[i]);
                selItem = i;
        }
}
        if (weight[selItem] > availCap)
         {
                printf("Quantity of item number: %d added is %d\n", (selItem + 1), availCap);
                sum += availCap * unitProfit;
                availCap = -1;
        }
                Else
        {
                printf("Quantity of item number: %d added is %d\n", (selItem + 1), weight[selItem]);
                availCap -= weight[selItem];
                sum += (float)profit[ selItem];
                profit[selItem] = 0;
        }
}
printf("The total profit is %.2f\n", sum);
return 0;
}
```

Output:

```
aiml-admin@aimladmin-HP:~$ gcc Proghram8.c
aiml-admin@aimladmin-HP:~$ ./a.out

Enter setize of the set:5

Enter set elements in increasing order
1 2 5 6 8

Enter maximum limit:9

The subsets with sum=9 are:
{1,2,6}
{1,8}
aiml-admin@aimladmin-HP:~$
```

Review Questions:

1. What is backtracking?
2. What is the time complexity of the subset sum problem using backtracking?

9. Design and implement C/C++ Program to sort a given set of n integer elements using SelectionSort method and compute its time Complexity. Run the program for varied values of n>5000 andrecord the time 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.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int n;
int *array;
void input()
{
printf("Enter the total numbers: ");
scanf("%d", &n);
array = (int *)malloc(n * sizeof(int));
for (int i = 0; i < n; i++) {
array[i] = rand() % 1000; // Generates random numbers 0-999
}
printf("Unsorted array\n");
for (int i = 0; i < n; i++) {
printf("%d ", array[i]);
}
printf("\n");
}
void selectionSort()
{
for (int step = 0; step < n - 1; step++)
 {
int min_idx = step;
for (int i = step + 1; i < n; i++)
{
if (array[i] < array[min_idx])
{
min_idx = i;
}
```

```
}
int temp = array[step];
array[step] = array[min_idx];
array[min_idx] = temp;
}
}
int main()
{
input();
clock_t start = clock();
selectionSort();
clock_t end = clock();
double duration = ((double)(end - start)) / CLOCKS_PER_SEC * 1000000000;
printf("\nTime for sorting is %.2f nano seconds\n", duration);
printf("Sorted Array in Ascending Order:\n");
for (int i = 0; i < n; i++)
 {
    printf("%d ", array[i]);
 }
printf("\n");
free(array);
return 0;
}
```

Output:

```
aiml-admin@aimladmin-HP:~$ gcc sel.c
aiml-admin@aimladmin-HP:~$ ./a.out
Enter the total numbers: 10
Unsorted array
383 886 777 915 793 335 386 492 649 421

Time for sorting is 4000.00 nano seconds
Sorted Array in Ascending Order:
335 383 386 421 492 649 777 793 886 915
aiml-admin@aimladmin-HP:~$
```

Review Questions:

1. What is Selection Sort?
2. What is the time/space complexity of the Selection Sort algorithm?
3. What are the characteristics of Selection Sort?
4. What are the advantages/disadvantages of Selection Sort?

10. Design and implement C/C++ Program to sort a given set of n integer elements using Quick Sort method and compute its time Complexity. Run the program for varied values of n>5000 andrecord the time 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.

```
#include <stdio.h>
#include<stdlib.h>
#include <time.h>

void swap(int *a,int *b)
{
int temp;
temp=*a;
*a=*b;
*b=temp;
}
int partition(int a[],int left,int right)
{
int p,i,j;
p=a[left];
i=left;
j=right+1;
while(i<=j)
{
do
   i++;
   while(a[i]<=p) ; //find elt>p
do
  j--;
while(a[j]>p); //find elt<p
if(i<j)
 swap(&a[i],&a[j]);
}
swap(&a[left],&a[j]); //swap pivot and a[j]
return j;
}
void quicksort(int a[],int left,int right)
{
int s;
if(left<right)
{
s=partition(a,left,right);
quicksort(a,left,s-1);
quicksort(a,s+1,right);
}
}
int main()
{
int n;
int *a;
printf("Enter the number of elements:");
```

```
scanf("%d", &n);
a = (int *)malloc(n * sizeof(int));
// Generates random numbers 0-999
for (int i = 0; i < n; i++)
{
a[i] = rand() % 1000;
}
printf("randomly generated elements are:\nArray is:");
for (int i = 0; i < n; i++)
{
printf("%d\t ", a[i]);
}
printf("\n");
a[n]=9999;

clock_t start = clock();
quicksort(a,0,n-1);
clock_t end = clock();

double duration = ((double)(end - start)) / CLOCKS_PER_SEC*1000;

printf("Sorted array is:\n");
for (int i = 0; i < n; i++)
{
printf("%d\t ", a[i]);
}
printf("\n");
printf("\nTime for sorting is %f milli seconds\n", duration);
free(a);
return 0;
  }
```

Output:

```
aiml-admin@aimladmin-HP:~$ gcc quick.c
aiml-admin@aimladmin-HP:~$ ./a.out
Enter the number of elements:
10
randomly generated elements are:
Array is
83      86      77      15      93      35      86      92      49      21
Sorted array is
15      21      35      49      77      83      86      86      92      93      Total time = 3000.00
0000 nanoseconds
aiml-admin@aimladmin-HP:~$
```

Review Questions:

1. What is the time and space complexity of Quick Sort?
2. What are the advantages/disadvantages of Quick Sort?

11. Design and implement C/C++ 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 recordthe time to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can begenerated using the random number generator.

```
#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() % 20;
}
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();
double elapsedTime = (double)(end - start) / CLOCKS_PER_SEC * 1000000000;
printf("Time taken to sort array is: %.0f nanoseconds\n", elapsedTime);
printf("Sorted array:\n");
for (i = 0; i < n; i++)
 {
printf("%d ", array[i]);
}
printf("\n");
return 0;
}
```

Output:

```
aiml-admin@aimladmin-HP:~$ gcc mergesortnew.c
aiml-admin@aimladmin-HP:~$ ./a.out
Enter the array size: 10
Array before sorting:
0 10 14 3 1 10 5 5 9 5
Time taken to sort array is: 42000 nanoseconds
Sorted array:
0 1 3 5 5 5 9 10 10 14
aiml-admin@aimladmin-HP:~$
```

Review Questions:

1. What is the time and space complexity of Merge Sort?
2. What are the advantages/disadvantages of Merge Sort?

12. Design and implement C/C++ Program for N Queen's problem using Back Tracking.

```
#include<stdio.h>
#include<stdlib.h>
int place(int);
int x[10];
void main()
{
int i, j, n, k, count = 0;
printf("\n\n********** N-QUEEN PROBLEM ***********\n\n");
printf("Enter the number of elements: ");
scanf("%d", &n);
if(n==0||n==2||n==3)
{
printf("\n\nNo solution\n");
printf("\n\n********** ******************* ***********");
exit(0);
}
k=1;
x[k]=0;
while(k)
{
x[k] = x[k] + 1;
while(x[k] <= n && !place(k))
x[k] = x[k] + 1;
if(x[k] <= n)
{
if(k == n)
{
printf("\nSolution %d\n\n", ++count);
for(i=1; i<=n; i++)
{
for(j=1; j<x[i]; j++)
```

```
printf("*\t");
printf("Q\t");
for(j=x[i]+1; j<=n; j++)
printf("*\t");
printf("\n");
}
}
else
{
k = k + 1;
x[k] = 0;
}
}
else
k = k-1;
}
printf("\n\n********** ******************* ************");
}
int place(int p)
{
int i;
for(i=1; i<=p-1; i++)
{
if(x[i] == x[p]||abs(i-p) == abs(x[i]-x[p]))
return 0;
}
return 1;
}
```

Output:

```
********** N-QUEEN PROBLEM ***********

Enter the number of elements: 4

Solution 1

*       Q       *       *
*       *       *       Q
Q       *       *       *
*       *       Q       *

Solution 2

*       *       Q       *
Q       *       *       *
*       *       *       Q
*       Q       *       *


********** ******************* ***********
Process returned 44 (0x2C)   execution time : 14.469 s
Press any key to continue.
```

```
********** N-QUEEN PROBLEM ***********

Enter the number of elements: 2


No solution


********** ******************* ***********
Process returned 0 (0x0)   execution time : 2.074 s
Press any key to continue.
```

Review Questions:

1. What is the N-Queens problem?
2. What does it mean for a position to be "safe" in the N-Queens problem?
3. What is the base case for solving the N-Queens problem using backtracking?
4. What is the time complexity of the N-Queens problem?

Viva Questions:

1. What is Kruskal's algorithm used for?
2. What is the main idea behind Kruskal's algorithm?
3. What is a minimum spanning tree (MST)?
4. How does Kruskal's algorithm ensure that the resulting tree is a spanning tree?
5. Given a graph with vertices {A, B, C, D} and edges {(A-B, 1), (B-C, 4), (A-C, 3), (C-D, 2)}, what is the MST using Kruskal's algorithm?
6. What is Prim's algorithm used for?
7. Explain the basic idea behind Prim's algorithm.
8. Can Prim's algorithm handle graphs with negative edge weights?
9. How does Prim's algorithm handle edge weights that are equal?
10. Given a graph with vertices {A, B, C, D} and edges {(A-B, 1), (B-C, 4), (A-C, 3), (C-D, 2)}, what is the MST using Prim's algorithm starting from vertex A?
11. What is Floyd/Warshall's algorithm?
12. What is the time complexity of Floyd/Warshall's algorithm?
13. Can Floyd/Warshall's algorithm be used for directed graphs?
14. What is the output of Floyd's/Warshall's algorithm?
15. What is the transitive closure of a graph?
16. What is Dijkstra's algorithm?
17. What is the time complexity of Dijkstra's algorithm?
18. Can Dijkstra's algorithm handle graphs with cycles?
19. What is the purpose of the visited set in Dijkstra's algorithm?
20. What are some real-world applications of Dijkstra's algorithm?
21. What is a topological sort?
22. Why can't a graph with cycles be topologically sorted?
23. What types of graphs can have a topological ordering?
24. What are the common algorithms used for topological sorting?
25. Can there be more than one topological ordering for a given DAG?
26. Why is it called the 0/1 Knapsack problem?
27. What is the objective of the 0/1 Knapsack problem?
28. How would you retrieve the items included in the optimal solution after solving the problem using DP?
29. What are some real-world applications of the 0/1 Knapsack problem?
30. What is the difference between the Discrete Knapsack problem and the Continuous Knapsack problem?
31. What is a greedy algorithm?

32. How does the Greedy Approximation Method work for the Continuous Knapsack problem?
33. Can the Greedy Algorithm guarantee an optimal solution for the Continuous Knapsack problem?
34. How would you apply the Greedy Algorithm to the following set of items for the Continuous Knapsack problem? Items: (Weight: 10, Profit: 60), (Weight: 20, Profit: 100), (Weight: 30, Profit: 120), Knapsack capacity: 50.
35. What is backtracking?
36. What is the time complexity of the subset sum problem using backtracking?
37. How does the program find a subset with the given sum?
38. Why do we need to enter the set elements in increasing order?
39. How does the program handle the case when no subset can achieve the target sum?
40. What is Selection Sort?
41. What is the time/space complexity of the Selection Sort algorithm?
42. What are the characteristics of Selection Sort?
43. How can the efficiency of this program be improved?
44. What are the advantages/disadvantages of Selection Sort?
45. How does Quick Sort work?
46. What is the time and space complexity of Quick Sort?
47. What are the advantages/disadvantages of Quick Sort?
48. Is Quick Sort stable
49. Explain the partitioning step in Quick Sort.
50. How does Merge Sort work?
51. What is the time and space complexity of Merge Sort?
52. What are the advantages/disadvantages of Merge Sort?
53. How does the merge step work in Merge Sort?
54. Compare Merge Sort and Quick Sort.
55. What is the N-Queens problem?
56. What does it mean for a position to be "safe" in the N-Queens problem?
57. How do you check if a position is safe for a queen?
58. What is the base case for solving the N-Queens problem using backtracking?
59. What is the time complexity of the N-Queens problem?