1.Sort a given set of N integer elements using Quick Sort technique and compute its time taken.

#include<time.h>

#include<stdio.h>

#define MAX 5000

void quicksort(int[],int,int);

int partition(int[],int,int);

void main()

{

int i,n,a[MAX],ch;

clock\_t start,end;

clrscr();

while(ch)

{

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

scanf("%d",&n);

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

a[i]=rand()%200;

printf("The random generated array is\n");

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

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

start=clock();

quicksort(a,0,n-1);

end=clock();

printf("\n\nthe sorted array elements are\n\n");

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

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

printf("time taken = %f",(end-start)/CLK\_TCK);

printf("\n\n do u wish to continue (0/1)\n");

scanf("%d",&ch);

}

}

void quicksort(int a[],int low,int high)

{

int mid;

delay(500);

if(low<high)

{

mid=partition(a,low,high);

quicksort(a,low,mid-1);

quicksort(a,mid+1,high);

}

}

int partition(int a[],int low,int high)

{

int key,i,j,temp,k;

key=a[low];

i=low+1;

j=high;

while(i<=j)

{

while(i<=high && key>=a[i])

i=i+1;

while(key<a[j])

j=j-1;

if(i<j)

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

else

{

k=a[j];

a[j]=a[low];

a[low]=k;

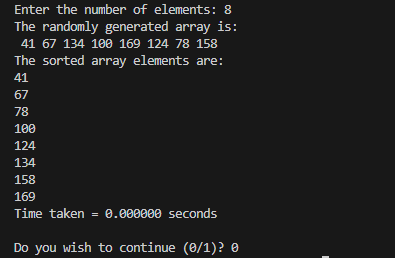
}

}

return j;

}

Output:



2. Find Minimum Cost Spanning Tree of a given undirected graph using Prim’s algorithm.

#include <stdio.h>

int cost[10][10], n, t[10][2], sum;

void prims(int cost[10][10], int n);

int main() {

int i, j;

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

scanf("%d", &n);

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

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

for (j = 0; j < n; j++) {

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

}

}

prims(cost, n);

printf("Edges of the minimal spanning tree:\n");

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

printf("(%d, %d)" , t[i][0], t[i][1]);

}

printf("\nSum of minimal spanning tree: %d\n", sum);

return 0;

}

void prims(int cost[10][10], int n) {

int i, j, u, v;

int min, source;

int p[10], d[10], s[10];

min = 999;

source = 0;

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

d[i] = cost[source][i];

s[i] = 0;

p[i] = source;

}

s[source] = 1;

sum = 0;

int k = 0;

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

min = 999;

u = -1;

for (j = 0; j < n; j++) {

if (s[j] == 0 && d[j] < min) {

min = d[j];

u = j;

}

}

if (u != -1) {

t[k][0] = u;

t[k][1] = p[u];

k++;

sum += cost[u][p[u]];

s[u] = 1;

for (v = 0; v < n; v++) {

if (s[v] == 0 && cost[u][v] < d[v]) {

d[v] = cost[u][v];

p[v] = u;

}

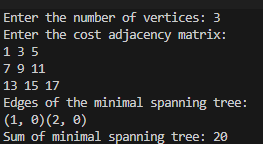
}

}

}

}

Output:



3. Find Minimum Cost Spanning Tree of a given undirected graph using Kruskal’s algorithm.

#include <stdio.h>

int cost[10][10], n, t[10][2], sum;

void kruskal(int cost[10][10], int n);

int find(int parent[10], int i);

int main() {

int i, j;

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

scanf("%d", &n);

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

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

for (j = 0; j < n; j++) {

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

}

}

kruskal(cost, n);

printf("Edges of the minimal spanning tree:\n");

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

printf("(%d, %d) ", t[i][0], t[i][1]);

}

printf("\nSum of minimal spanning tree: %d\n", sum);

return 0;

}

void kruskal(int cost[10][10], int n) {

int min, u, v, count, k;

int parent[10];

k = 0;

sum = 0;

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

parent[i] = i;

}

count = 0;

while (count < n - 1) {

min = 999;

u = -1;

v = -1;

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

for (int j = 0; j < n; j++) {

if (find(parent, i) != find(parent, j) & cost[i][j] < min) {

min = cost[i][j];

u = i;

v = j;

}

}

}

int root\_u = find(parent, u);

int root\_v = find(parent, v);

if (root\_u != root\_v) {

parent[root\_u] = root\_v;

t[k][0] = u;

t[k][1] = v;

sum += min;

k++;

count++;

}

}

}

int find(int parent[10], int i) {

while (parent[i] != i) {

i = parent[i];

}

return i;

}

Output:

