ANALYSIS AND DESIGN OF ALGORITHMS LABORATORY PROGRAMS

PROGRAM – 01 (KRUSKAL'S ALGORITHM)

Design and implement C/C++ Program to find Minimum Cost Spanning Tree of a given connected undirected graph using Kruskal's algorithm.

```
#include<stdio.h>
#define INF 999
#define MAX 100
int p[MAX],c[MAX][MAX],t[MAX][2];
int find(int v)
{
        while(p[v])
        v=p[v];
        return v;
}
void union1(int i,int j)
{
        p[j]=i;
}
void kruskal(int n)
{
        int i,j,k,u,v,min,res1,res2,sum=0;
        for(k=1;k<n;k++)
        {
                min=INF;
                for(i=1;i<n;i++)
                {
                    for(j=1;j<=n;j++)
                    {
                    if(i==j)
                    continue;
                    if(c[i][j]<min)
                    {
                          u=find(i);
                          v=find(j);
                          if(u!=v)
                          {
                          res1=i;
                          res2=j;
                          min=c[i][j];
                          }
                    }
                    }
                }
        union1(res1,find(res2));
        t[k][1]=res1;
        t[k][2]=res2;
        sum=sum+min;
        }
printf("\n Cost of spanning tree is =%d",sum);
printf("\n Edge of spanning tree are : \n");
for(i=1;i<n;i++)
```

```
printf("%d->%d\n",t[i][1],t[i][2]);
}
int main()
{
        int i,j,n;
        printf("\nEnter the n value : ");
        scanf("%d",&n);
        for(i=1;i<n;i++)
        p[i]=0;
        printf("\nEnter the graph data : \n");
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        scanf("%d",&c[i][j]);
        kruskal(n);
        return 0;
}
```

OUTPUT :

```
Enter the n value : 5

Enter the graph data :
0 5 999 6 999
5 0 1 3 999
999 1 0 4 6
6 3 4 0 2
999 999 6 2 0

 Cost of spanning tree is =11
 Edge of spanning tree are :
2->3
4->5
2->4
1->2
```

PROGRAM – 02 (PRIMS'S ALGORITHM)

Design and implement C/C++ Program to find Minimum Cost Spanning Tree of a given connected undirected graph using Prim's algorithm.

```
#include<stdio.h>
int ne=1,min_cost=0;
void main() {
int n,i,j,min,cost[20][20],a,u,b,v,source,visited[20];
printf("Enter the no. of nodes:");
scanf("%d",&n);
printf("Enter the cost matrix:\n");
for(i=1;i<=n;i++) {
for(j=1;j<=n;j++) {
scanf("%d",&cost[i][j]);
}
}
for(i=1;i<=n;i++)
visited[i]=0;
printf("Enter the root node:");
scanf("%d",&source);
visited[source]=1;
printf("\nMinimum cost spanning tree is\n");
while(ne<n)
{
min=999;
for(i=1;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("\nEdge %d\t(%d->%d)=%d\n",ne++,a,b,min);
min_cost=min_cost+min;
visited[b]=1;
}
cost[a][b]=cost[b][a]=999;
}
printf("\nMinimum cost=%d\n",min_cost);
}
```

OUTPUT :

```
Enter the no. of nodes:5
Enter the cost matrix:
0 5 999 6 999
5 0 1 3 999
999 1 0 4 6
6 3 4 0 2
999 999 6 2 0
Enter the root node:1

Minimum cost spanning tree is

Edge 1   (1->2)=5

Edge 2   (2->3)=1

Edge 3   (2->4)=3

Edge 4   (4->5)=2

Minimum cost=11
```

PROGRAM – 03 A

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

```
#include<stdio.h>
#define INF 99
int min(int a , int b)
{
        return(a<b)?a:b;
}

void floyd(int p[][10],int n)
{
        int i,j,k;
        for(k=1;k<=n;k++)
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        p[i][j]=min(p[i][j],p[i][k]+p[k][j]);
}

void main()
{
        int a[10][10],n,i,j;
        printf("\nEnter the n value : ");
        scanf("%d",&n);
        printf("\nEnter the graph data : \n");
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        scanf("%d",&a[i][j]);
        floyd(a,n);
        printf("\nShortest path matrix\n");
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                printf("%d",a[i][j]);
                printf("\n");
        }
}
```

OUTPUT :

```
Enter the n value : 4

Enter the graph data :
0 99 3 99
2 0 99 99
99 7 0 1
6 99 99 0

Shortest path matrix
01034
2056
7701
61690
```

PROGRAM – 03 B

Design and implement C/C++ Program to find the transitive closure using Warshal's algorithm.

```
#include<stdio.h>
void warsh(int p[][10],int n)
{
        int i,j,k;
        for(k=1;k<=n;k++)
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        p[i][j]=p[i][j]||p[i][k]&&p[k][j];
}

int main()
{
        int a[10][10],n,i,j;
        printf("\nEnter the n value : ");
        scanf("%d",&n);
        printf("\nEnter the graph data : \n");
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        scanf("%d",&a[i][j]);
        warsh(a,n);
        printf("\nResultant path matrix\n");
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j++)
                printf("%d",a[i][j]);
                printf("\n");
        }
        return 0;
}
```

OUTPUT :

```
Enter the n value : 4

Enter the graph data :
0 1 0 0
0 0 0 1
0 0 0 0
1 0 1 0

Resultant path matrix
1111
1111
0000
1111
```

PROGRAM – 04

Design and implement C/C++ Program to find shortest paths from a given vertex in a weighted connected graph to other vertices using Dijkstra's algorithm.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_VERTICES 10
#define INF 99
int d[MAX_VERTICES];
int p[MAX_VERTICES];
int visited[MAX_VERTICES];
void dijk(int a[MAX_VERTICES][MAX_VERTICES], int s, int n) {
int u, v, i, j, min;
for (v = 0; v < n; v++) {
d[v] = INF;
p[v] = -1;
visited[v] = 0;
}
d[s] = 0;
for (i = 0; i < n; i++) {
min = INF;
for (j = 0; j < n; j++) {
if (d[j] < min && visited[j] == 0) {
min = d[j];
u = j;
}
}
visited[u] = 1;
for (v = 0; v < n; v++) {
if ((d[u] + a[u][v] < d[v]) && (u != v) && visited[v] == 0) {
d[v] = d[u] + a[u][v];
p[v] = u;
}
}
}
}

void path(int v, int s)
{
if (p[v] != -1)
path(p[v], s);
if (v != s)
printf("->%d ", v);
}
void display(int s, int n) {
int i;
for (i = 0; i < n; i++) {
if (i != s) {
printf("%d ", s);
path(i, s);
}
if (i != s)
printf("=%d ", d[i]);
printf("\n");
```

```
}
}
int main() {
int a[MAX_VERTICES][MAX_VERTICES];
int i, j, n, s;
printf("Enter the number of vertices: ");
scanf("%d", &n);
printf("Enter the weighted matrix:\n");
for (i = 0; i < n; i++)
for (j = 0; j < n; j++)
scanf("%d", &a[i][j]);
printf("Enter the source vertex: ");
scanf("%d", &s);
dijk(a, s, n);
printf("The shortest path between source %d to remaining vertices are:\n", s);
display(s, n);
return 0;
}
```

OUTPUT :

```
Enter the number of vertices: 5
Enter the weighted matrix:
999 3 999 7 999
3 999 4 2 999
999 4 999 5 6
7 2 5 999 4
999 999 6 4 999
Enter the source vertex: 0
The shortest path between source 0 to remaining vertices are:

0 ->1 =3
0 ->1 ->2 =7
0 ->1 ->3 =5
0 ->1 ->3 ->4 =9
```

PROGRAM – 05

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

```
#include<stdio.h>
int temp[10],k=0;
void sort(int a[][10],int id[],int n)
{
        int i,j;
        for(i=1;i<=n;i++)
        {
        if(id[i]==0)
        {
        id[i]=-1;
        temp[++k]=i;
        for(j=1;j<=n;j++)
        {
        if(a[i][j]==1 && id[j]!=-1)
        id[j]--;
        }
        i=0;
        }
        }
}
void main()
{
        int a[10][10],id[10],n,i,j;
        printf("\nEnter the n value : ");
        scanf("%d",&n);
        for(i=1;i<=n;i++)
        id[i]=0;
        printf("\n Enter the graph data : \n");
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        {
                scanf("%d",&a[i][j]);
                if(a[i][j]==1)
                id[j]++;
        }
        sort(a,id,n);
        if(k!=n)
        printf("\nTopological ordering not possible ");
        else
        {
        printf("\nTopological ordering is : ");
        for(i=1;i<=k;i++)
        printf("%d",temp[i]);
        }
}
```

OUTPUT :

```
Enter the n value : 5

 Enter the graph data :
0 0 1 1 0
1 0 0 1 0
0 0 0 0 1
0 0 1 0 1
0 0 0 0 0

Topological ordering is : 2     1       4       3       5
```

PROGRAM – 06

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

```
#include<stdio.h>
int max(int a,int b)
{
        return(a>b)?a:b;
}
int knapsack(int W , int wt[],int val[], int n)
{
int i, w;
int K[n + 1][W + 1];
for (i = 0; i <= n; i++) {
for (w = 0; w <= W; w++) {
if (i == 0 || w == 0)
K[i][w] = 0;
else if (wt[i - 1] <= w)
K[i][w] = max(val[i - 1] + K[i - 1][w - wt[i - 1]], K[i - 1][w]);
else
K[i][w] = K[i - 1][w];
}
}
return K[n][W];
}
int main() {
int val[100], wt[100];
int W,n;
printf("Enter the number of items: ");
scanf("%d", &n);
printf("Enter the values and weights of %d items:\n", n);
for (int i = 0; i < n; i++) {
printf("Enter value and weight for item %d: ", i + 1);
scanf("%d %d", &val[i], &wt[i]);
}
printf("Enter the knapsack capacity: ");
scanf("%d", &W);
printf("Maximum value that can be obtained: %d\n", knapsack(W, wt, val, n));
return 0;
}
```

OUTPUT:

```
Enter the number of items: 3
Enter the values and weights of 3 items:
Enter value and weight for item 1: 20 5
Enter value and weight for item 2: 25 10
Enter value and weight for item 3: 15 8
Enter the knapsack capacity: 16
Maximum value that can be obtained: 45
```

PROGRAM -07

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

```
#include<stdio.h>
int main()
{
float weight[50],profit[50],ratio[50],Totalvalue,temp,capacity,amount;
int n,i,j;
printf("Enter the number of items :");
scanf("%d",&n);
for (i = 0; i < n; i++)
{
printf("Enter Weight and Profit for item[%d] :\n",i);
scanf("%f %f", &weight[i], &profit[i]);
}
printf("Enter the capacity of knapsack :\n");
scanf("%f",&capacity);
for(i=0;i<n;i++)
ratio[i]=profit[i]/weight[i];
for (i = 0; i < n; i++)
for (j = i + 1; j < n; j++)
if (ratio[i] < ratio[j])
{
temp = ratio[j];
ratio[j] = ratio[i];
ratio[i] = temp;
temp = weight[j];
weight[j] = weight[i];
weight[i] = temp;
temp = profit[j];
profit[j] = profit[i];
profit[i] = temp;
}
printf("Knapsack problems using Greedy Algorithm:\n");
for (i = 0; i < n; i++)
{
if(weight[i] > capacity) break;
else
{
Totalvalue = Totalvalue + profit[i];
capacity = capacity - weight[i];
}
}
if (i < n)
Totalvalue = Totalvalue + (ratio[i]*capacity);
printf("\nThe maximum value is :%f\n",Totalvalue);
return 0;
}
```

OUTPUT:

```
Enter the number of items :3
Enter Weight and Profit for item[0] :
5 20
Enter Weight and Profit for item[1] :
10 25
Enter Weight and Profit for item[2] :
15 8
Enter the capacity of knapsack :
16
Knapsack problems using Greedy Algorithm:

The maximum value is :45.533333
```

PROGRAM -08

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>
#define MAX 10
int s[MAX],x[MAX],d;
void sumofsub(int p,int k,int r)
{
int i;
x[k]=1;
if((p+s[k])==d)
{
for(i=1;i<=k;i++)
if(x[i]==1)
printf("%d ",s[i]);
printf("\n");
}
else
if(p+s[k]+s[k+1]<=d)
sumofsub(p+s[k],k+1,r-s[k]);
if((p+r-s[k]>=d) && (p+s[k+1]<=d))
{
x[k]=0;
sumofsub(p,k+1,r-s[k]);
}
}
int main()
{
int i,n,sum=0;
printf("\nEnter the n value:");
scanf("%d",&n);
printf("\nEnter the set in increasing order:");
for(i=1;i<=n;i++)
scanf("%d",&s[i]);
printf("\nEnter the max subset value:");
scanf("%d",&d);
for(i=1;i<=n;i++)
sum=sum+s[i];
if(sum<d || s[1]>d)
printf("\nNo subset possible");
else
sumofsub(0,1,sum);
return 0;
}
```

OUTPUT:

```
Enter the n value:5

Enter the set in increasing order:1 2 5 6 8

Enter the max subset value:9
1 2 6
1 8
```

PROGRAM -09

Design and implement C/C++ 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.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
// Swap utility
void swap(long int* a, long int* b)
{
int tmp = *a;
*a = *b;
*b = tmp;
}
// Selection sort
void selectionSort(long int arr[], long int n)
{
long int i, j, min;
for (i = 0; i < n - 1; i++) {
// Find the minimum element in unsorted array
min = i;
for (j = i + 1; j < n; j++)
if (arr[j] < arr[min])
min = j;
// Swap the found minimum element
// with the first element
swap(&arr[min], &arr[i]);
}
}
// Driver code
int main()
{
long int n = 5000;
int iteration = 0;
// Arrays to store time duration
// of sorting algorithms
double time[10];
printf("A_size, Selection\n");
// Performs 10 iterations
while (iteration++ < 10) {
long int a[n];
// generating n random numbers
// storing them in array a
for (int i = 0; i < n; i++) {
long int num = rand() % n + 1;
a[i] = num;
}
// using clock_t to store time
clock_t start, end;
// Selection sort
```

```
start = clock();
selectionSort(a, n);
end = clock();
time[iteration] = ((double)(end - start));
// type conversion to long int
// for plotting graph with integer values
printf("%li, %li\n",
n, (long int)time[iteration]);
// increases the size of array by 500
n += 500;
}
return 0;
}
```

OUTPUT:

```
A_size, Selection
5000, 23831
5500, 10139
6000, 12150
6500, 14147
7000, 16186
7500, 18710
8000, 21153
8500, 24010
9000, 26886
9500, 29942
```

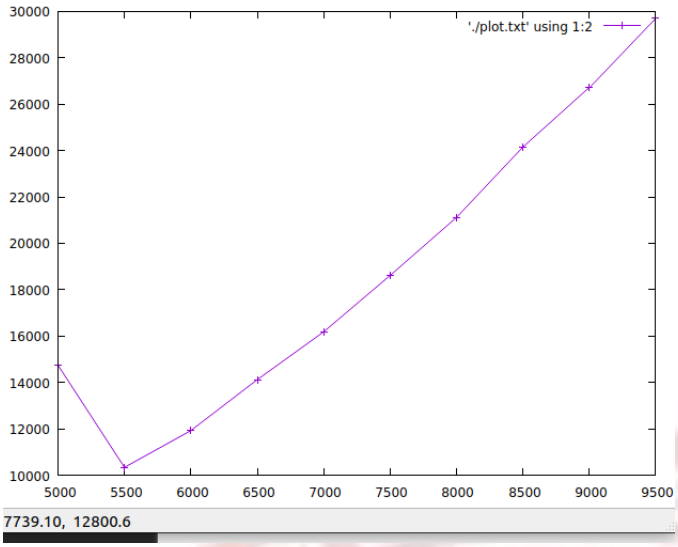
STEPS TO PLOT THE GRAPH

To Install gnuplot to get graph
sudo-apt-get install gnuplot

To get the graph

./a.out>plot.txt
gnuplot

gnuplot> plot './plot.txt' using 1:2 with linespoints
If you use “,” in the output give data separator as “,” or else “ “.
gnuplot> set datafile separator ","
gnuplot> plot './plot.txt' using 1:2 with linespoints


'./plot.txt' using 1:2
30000
28000
26000
24000
22000
20000
18000
16000
14000
12000
10000
5000
5500
6000
6500
7000
7500
8000
8500
9000
9500
7739.10, 12800.6

PROGRAM -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 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.

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
void swap(long int *a,long int *b)
{
long int tmp=*a;
*a=*b;
*b=tmp;
}
long int partition(long int arr[],long int low,long int high)
{
long int pivot=arr[high];
long int i=low-1;
for(long int j=low;j<=high-1;j++)
{
if (arr[j]<=pivot)
{
i++;
swap(&arr[i],&arr[j]);
}
}
swap(&arr[i+1],&arr[high]);
return (i+1);
}
void quicksort(long int arr[],long int low,long int high)
{
if (low<high)
{
long int pi=partition(arr,low,high);
quicksort(arr,low,pi-1);
quicksort(arr,pi+1,high);
}
}
int main()
{
long int n=10000;
int it=0;
double time [10];
printf("a_size,quick\n");
while(it++<10)
{
long int a[n];
for(int i=0;i<n;i++)
{
long int num=rand()%n+1;
a[i]=num;
}
```

```
clock_t start,end;
start=clock();
quicksort(a,0,n-1);
end=clock();
time[it]=((double)(end-start) / CLOCKS_PER_SEC);
printf("%li %f\n",n,time[it]);
n+=10000;
}
return 0;
}
```

OUTPUT:

```
a_size,quick
10000 0.003403
20000 0.001267
30000 0.001990
40000 0.002807
50000 0.003603
60000 0.006285
70000 0.006191
80000 0.005993
90000 0.006813
100000 0.007273
```

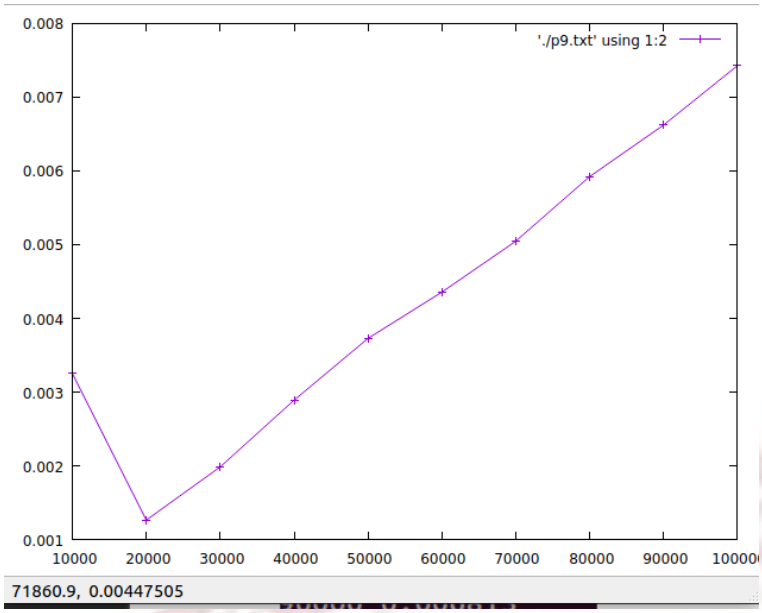
To get the graph

./a.out>plot.txt
gnuplot

gnuplot> plot './p9.txt' using 1:2 with linespoints

If you use “,” in the output give data separator as “,” or else “ “.

gnuplot> set datafile separator " "
gnuplot> plot './p9.txt' using 1:2 with linespoints


'./p9.txt' using 1:2
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
10000
20000
30000
40000
50000
60000
70000
80000
90000
71860.9, 0.00447505

PROGRAM -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 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.

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
void merge(long int arr[],long int low,long int mid,long int high)
{
        long int k=low;
        long int i=low;
        long int j=mid+1;
        long int temp[100000];
        while(i<=mid && j<=high)
        {
                if(arr[i]<=arr[j])
                {
                        temp[k]=arr[i];
                        i++;
                        k++;
                }
        else
        {
                temp[k]=arr[j];
                j++;
                k++;
        }
        }
while(i<=mid)
{
        temp[k]=arr[i];
        i++;
        k++;
}
while(j<=high)
{
        temp[k]=arr[j];
        j++;
        k++;
}
for(i=low;i<=high;i++)
arr[i]=temp[i];
}

void mergesort(long int arr[],long int low,long int high)
{
        if(low<high)
        {
                long int mid=(low+high)/2;
                mergesort(arr,low,mid);
                mergesort(arr,mid+1,high);
```

```
                merge(arr,low,mid,high);
        }
}

int main()
{
        long int n=10000;
        int it=0;
        double time[10];
        printf("A_size,Merge\n");
        while(it++<10)
        {
                long int arr[n];
                for(int i=0;i<n;i++)
                {
                        long int num=rand()%n+1;
                        arr[i]=num;
                }
                clock_t start,end;
                start=clock();
                mergesort(arr,0,n-1);
                end=clock();
                time[it]=((double)(end-start))/CLOCKS_PER_SEC;
                printf("%li,%f\n",n,time[it]);
                n+=10000;
        }
return 0;
}
```

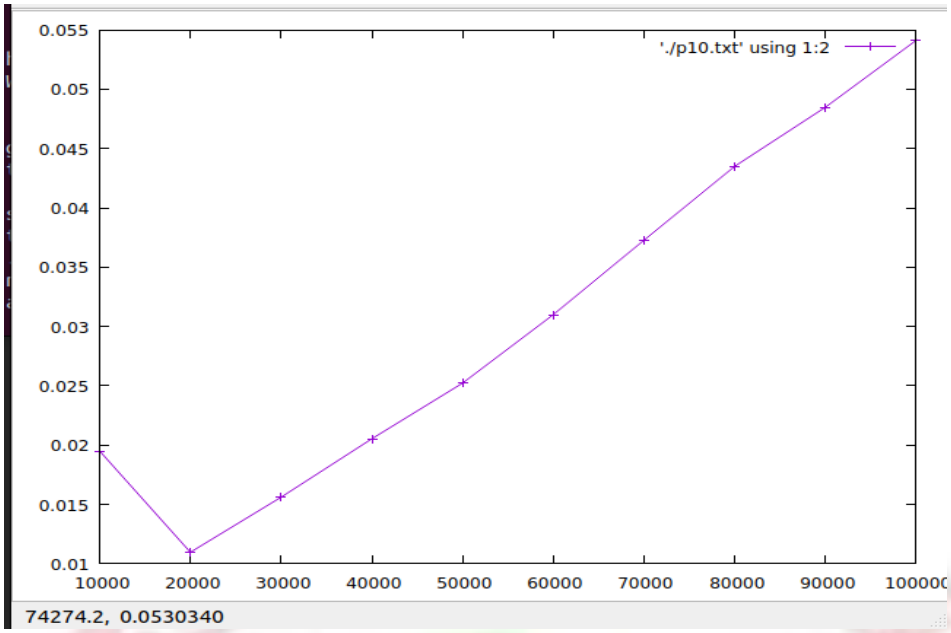
OUTPUT:

```
A_size,Merge
10000,0.009012
20000,0.013463
30000,0.017047
40000,0.021064
50000,0.026017
60000,0.032097
70000,0.037430
80000,0.043230
90000,0.048793
100000,0.052670
```

To get the graph

./a.out>plot.txt
gnuplot

gnuplot> plot './p10.txt' using 1:2 with linespoints
If you use “,” in the output give data separator as “,” or else “ “.
gnuplot> set datafile separator ","
gnuplot> plot './p10.txt' using 1:2 with linespoints


'./p10.txt' using 1:2
0.055
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
74274.2, 0.0530340

PROGRAM -12

Design and implement C/C++ Program for N Queen's problem using Backtracking.

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 50
int can_place(int c[],int r)
{
int i;
for(i=0;i<r;i++)
if(c[i]==c[r] || (abs(c[i]-c[r])==abs(i-r)))
return 0;
return 1;
}
void display(int c[],int n)
{
int i,j;
char cb[10][10];
for(i=0;i<n;i++)
for(j=0;j<n;j++)
cb[i][j]='-';
for(i=0;i<n;i++)
cb[i][c[i]]='Q';
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
printf("%c",cb[i][j]);
printf("\n");
}
}
void n_queens(int n)
{
int r;
int c[MAX];
c[0]=-1;
r=0;
while(r>=0)
{ c[r]++;
while(c[r]<n && !can_place(c,r))
c[r]++;
if(c[r]<n)
{
if(r==n-1)
{ display(c,n);
printf("\n\n");
}
else
{ r++;
c[r]=-1;
}
}
else
r--;
}
```

```
}
void main()
{
int n;
printf("\nEnter the no. of queens:");
scanf("%d",&n);
n_queens(n);
}
```

OUTPUT:

```
Enter the no. of queens:4
-Q--
---Q
Q---
--Q-


--Q-
Q---
---Q
-Q--
```