VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

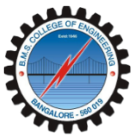
LAB REPORT
on

Analysis and Design of Algorithms

Submitted by

Akash M (1BM20CS006)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING



B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
May-2022 to July-2022

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "**Analysis and Design of Algorithms**" carried out by **Akash M(1BM20CS006),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of **Analysis and Design of Algorithms - (19CS4PCADA)** work prescribed for the said degree.

Dr. Prasad G R
Professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

`

Index Sheet

Sl. No.	Experiment Title	Page No.
1	Write a recursive program to Solve **a)** Towers-of-Hanoi problem **b)** To find GCD	
2	Implement Recursive Binary search and Linear search and determine the time required to search an element. Repeat the experiment for different values of N and plot a graph of the time taken versus N.	
3	Sort a given set of N integer elements using Selection Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.	
4	Write program to do the following: **a)** Print all the nodes reachable from a given starting node in a digraph using BFS method. **b)** Check whether a given graph is connected or not using DFS method.	
5	Sort a given set of N integer elements using Insertion Sort technique and compute its time taken.	
6	Write program to obtain the Topological ordering of vertices in a given digraph.	
7	Implement Johnson Trotter algorithm to generate permutations.	
8	Sort a given set of N integer elements using Merge Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.	
9	Sort a given set of N integer elements using Quick Sort technique and compute its time taken.	
10	Sort a given set of N integer elements using Heap Sort technique and compute its time taken.	
11	Implement Warshall's algorithm using dynamic programming	
12	Implement 0/1 Knapsack problem using dynamic programming.	
13	Implement All Pair Shortest paths problem using Floyd's algorithm.	
14	Find Minimum Cost Spanning Tree of a given undirected graph using Prim's algorithm.	
15	Find Minimum Cost Spanning Tree of a given undirected graph using Kruskals algorithm.	
16	From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.	
17	Implement "Sum of Subsets" using Backtracking. "Sum of Subsets" problem: Find a subset of a given set S = {s1,s2,……,sn} of n positive integers whose sum is equal to a given positive integer d. For example, if S = {1,2,5,6,8} and d = 9 there are two solutions	

	{1,2,6} and {1,8}. A suitable message is to be displayed if the given problem instance doesn't have a solution.	
18	Implement "N-Queens Problem" using Backtracking.	

Course Outcome

CO1	Ability to **analyze** time complexity of Recursive and Non-Recursive algorithms using asymptotic notations.
CO2	Ability to **design** efficient algorithms using various design techniques.
CO3	Ability to **apply** the knowledge of complexity classes P, NP, and NP-Complete and prove certain problems are NP-Complete
CO4	Ability to **conduct** practical experiments to solve problems using an appropriate designing method and find time efficiency.

```c
//1.Write a recursive program to
  //a. Solve Towers-of-Hanoi problem

#include<stdio.h>

void toh(int n,char s,char d,char a){
  if(n==1){
    printf("Moving from %c to %c\n",s,d);
    return;
  }
  toh(n-1,s,a,d);
  printf("Moving from %c to %c\n",s,d);
  toh(n-1,a,d,s);
}

int main(){
  int n;
  char s,d,a,temp;
  printf("Number of Discs :");
  scanf("%d",&n);
  printf("Enter the Names of Source Destination and Aux towers :");
  scanf("%c",&temp);
  scanf("%c %c %c",&s,&d,&a);
  toh(n,s,d,a);
}
```

Output:

```
Number of Discs :5
Enter the Names of Source Destination and Aux towers :a c
Moving from a to c
Moving from a to b
Moving from c to b
Moving from a to c
Moving from b to a
Moving from b to c
Moving from a to c
Moving from a to b
Moving from c to b
Moving from c to a
Moving from b to a
Moving from c to b
Moving from a to c
Moving from a to b
Moving from c to b
Moving from a to c
Moving from b to a
Moving from b to c
Moving from a to c
Moving from b to a
Moving from c to b
Moving from c to a
Moving from b to a
Moving from b to c
Moving from a to c
Moving from a to b
Moving from c to b
Moving from a to c
Moving from b to a
Moving from b to c
Moving from a to c
```

```c
//1.Write a recursive program to
  //b. T o find GCD
#include<stdio.h>

int gcd(int n, int m){
  if(m!=0){
    return gcd(m,n%m);
  }
  else{
    return n;
  }
}


int main(){
  int n,m;
  printf("Enter number for which GCD to be found :");
  scanf("%d %d",&n,&m);
  int res=gcd(n,m);
  printf("GCD of %d and %d is %d\n",n,m,res);
}
```

Output:

```
akashm@Akashs-MacBook-Pro ADAPrograms % ./GCD
Enter number for which GCD to be found :366 60
GCD of 366 and 60 is 6
akashm@Akashs-MacBook-Pro ADAPrograms % ./GCD
Enter number for which GCD to be found :6 3
GCD of 6 and 3 is 3
akashm@Akashs-MacBook-Pro ADAPrograms % ./GCD
Enter number for which GCD to be found :66 6
GCD of 66 and 6 is 6
akashm@Akashs-MacBook-Pro ADAPrograms % ./GCD
Enter number for which GCD to be found :78 6
GCD of 78 and 6 is 6
akashm@Akashs-MacBook-Pro ADAPrograms % ./GCD
Enter number for which GCD to be found :78 12
GCD of 78 and 12 is 6
```

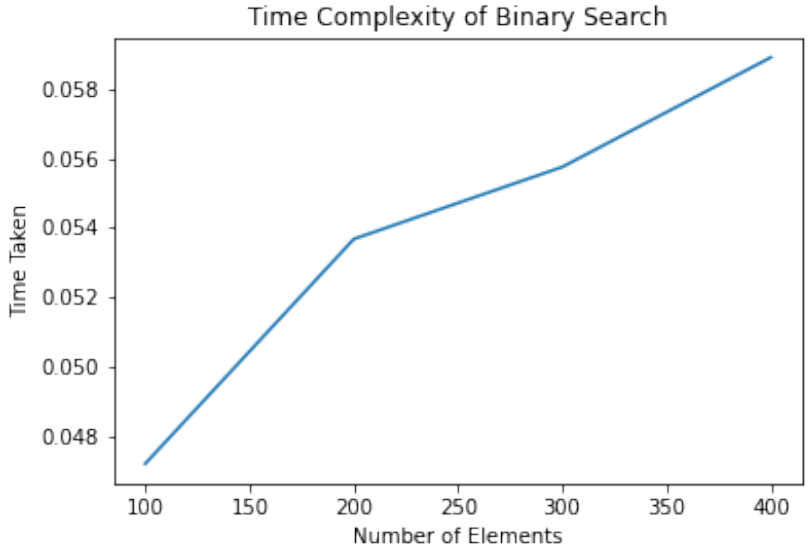
```c
//2.a. Implement Recursive Binary search and determine the time required to search an element. Repeat the experiment for different values of N and plot a graph of the time taken versus N.

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

void delay(){
  for (int i = 0; i < 5000000; ++i)
  {
    int d=i/33;
  }
}

int binarySearch(int arr[], int k, int si, int ei){
  if(si>ei){
    return -1;
  }
  delay();
  int mid=(si+ei)/2;
  if(arr[mid]==k){
    return mid;
  }
  else if(arr[mid]>k){
    return binarySearch(arr,k,si,mid-1);
  }
  else{
    return binarySearch(arr,k,mid+1,ei);
  }
}

void swap(int arr[], int i, int j){
  if(i==j){
    return;
  }
  int temp=arr[i];
  arr[i]=arr[j];
  arr[j]=temp;
}

void bubbleSort(int arr[], int n){
  for(int i=n-1;i>=0;i--){
    for(int j=0;j<i;j++){
      if(arr[j]>arr[j+1]){
        swap(arr,j,j+1);
      }
    }
  }
}

int main(){
  int arr[5000],n;
  printf("Enter the number of elements in the Array :");
  scanf("%d",&n);
```

```c
  for(int i=0;i<n;i++){
    arr[i]=(rand() % (1 - 5000 + 1)) + 1;
  }
  bubbleSort(arr,n);
  clock_t start=clock();
  int j=binarySearch(arr,-1,0,n-1);
  clock_t end=clock();

  printf("Time taken by Binary Search is %f",(double)(end-start)/CLOCKS_PER_SEC);
}
```

Output:

```
akashm@Akashs-MacBook-Pro ADAPrograms % ./BinarySearch
Enter the number of elements in the Array :100
Time taken by Binary Search is 0.047197%
akashm@Akashs-MacBook-Pro ADAPrograms % ./BinarySearch
Enter the number of elements in the Array :200
Time taken by Binary Search is 0.053674%
akashm@Akashs-MacBook-Pro ADAPrograms % ./BinarySearch
Enter the number of elements in the Array :300
Time taken by Binary Search is 0.055753%
akashm@Akashs-MacBook-Pro ADAPrograms % ./BinarySearch
Enter the number of elements in the Array :400
Time taken by Binary Search is 0.058912%
```

Graph:

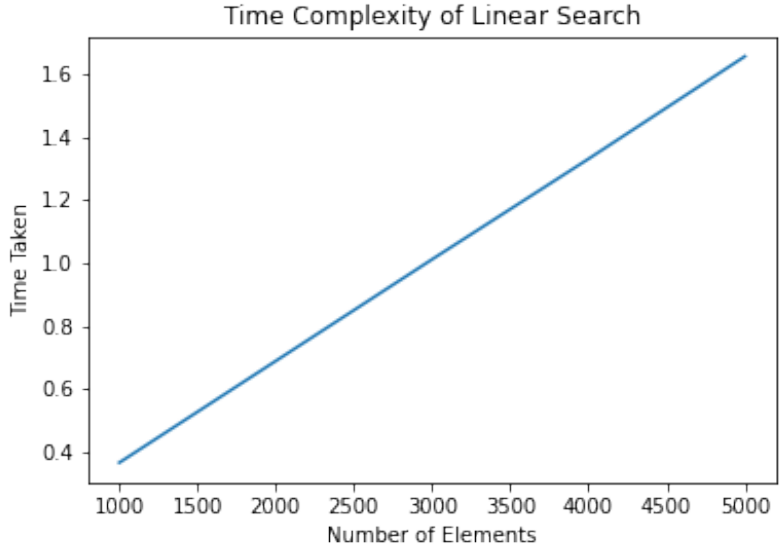


```c
//2.b. Implement Recursive Linear search and determine the time required to search an element. Repeat the experiment for different values of N and plot a graph of the time taken versus N.

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

void delay(){
  for (int i = 0; i < 500000; ++i)
  {
    int d=i/33;
  }
}


int linearSearch(int arr[], int k,int si,int ei){
  if(si>ei){
    return -1;
  }
  if(arr[si]==k){
    return si;
  }
  delay();
  return linearSearch(arr,k,si+1,ei);
}


int main(){
  int arr[5000],n;
  printf("Enter the number of elements in the Array :");
  scanf("%d",&n);
  for(int i=0;i<n;i++){
    arr[i]=(rand() % (1 - 5000 + 1)) + 1;
  }
  clock_t start=clock();
  int i=linearSearch(arr,-1,0,n-1);
  clock_t end=clock();

  printf("Time taken by Linear Search is %f",(double)(end-start)/CLOCKS_PER_SEC);
}
```

Output:

```
akashm@Akashs-MacBook-Pro ADAPrograms % make Lin
cc     LinearSearch.c   -o LinearSearch
akashm@Akashs-MacBook-Pro ADAPrograms % ./Linear
Enter the number of elements in the Array :1000
Time taken by Linear Search is 0.364711%
akashm@Akashs-MacBook-Pro ADAPrograms % ./Linear
Enter the number of elements in the Array :2000
Time taken by Linear Search is 0.687920%
akashm@Akashs-MacBook-Pro ADAPrograms % ./Linear
Enter the number of elements in the Array :3000
Time taken by Linear Search is 1.011175%
akashm@Akashs-MacBook-Pro ADAPrograms % ./Linear
Enter the number of elements in the Array :4000
Time taken by Linear Search is 1.330706%
akashm@Akashs-MacBook-Pro ADAPrograms % ./Linear
Enter the number of elements in the Array :5000
Time taken by Linear Search is 1.656666%
akashm@Akashs-MacBook-Pro ADAPrograms %
```

Graph:

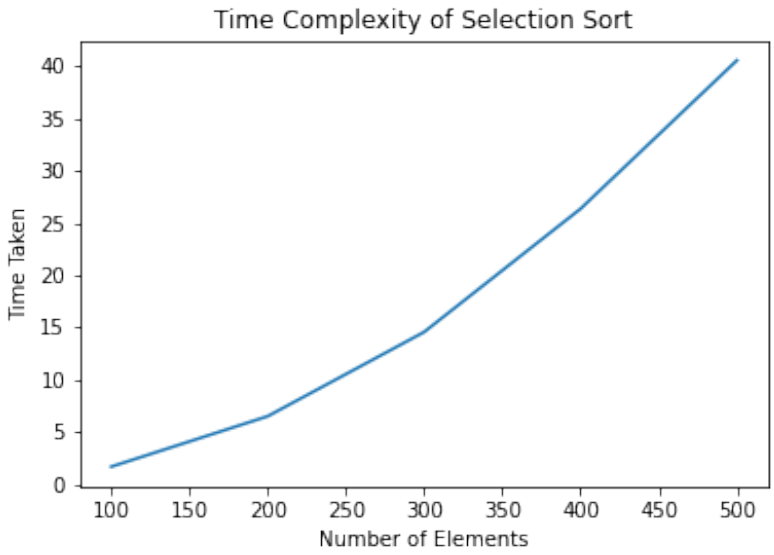


```c
//3.Sort a given set of N integer elements using Selection Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.

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

void delay(){
  for (int i = 0; i < 500000; ++i)
  {
    int d=i/33;
  }
}

void swap(int arr[],int i,int j){
  if(i==j){
    return;
  }
  int temp=arr[i];
  arr[i]=arr[j];
  arr[j]=temp;
}

void selectionSort(int arr[],int n){
  for(int i=n-1;i>0;i--){
    int max=-999;
    int maxInd=0;
    for (int j = 0; j <= i; ++j)
    {
      delay();
      if(arr[j] > max){
        max=arr[j];
        maxInd=j;
      }
    }
    swap(arr,maxInd,i);
  }
}

int main(){
  int arr[5000],n;
  printf("Enter the number of elements in the Array :");
  scanf("%d",&n);
  for(int i=0;i<n;i++){
    arr[i]=(rand() % (1 - 5000 + 1)) + 1;
  }
  clock_t start=clock();
  selectionSort(arr,n);
  clock_t end=clock();

  printf("Time taken by Selection Sort is %f",(double)(end-start)/CLOCKS_PER_SEC);
}
```

Output:

```
akashm@Akashs-MacBook-Pro ADAPrograms % ./SelectionSort
Enter the number of elements in the Array :100
Time taken by Selection Sort is 1.663294%
akashm@Akashs-MacBook-Pro ADAPrograms % ./SelectionSort
Enter the number of elements in the Array :200
Time taken by Selection Sort is 6.492463%
akashm@Akashs-MacBook-Pro ADAPrograms % ./SelectionSort
Enter the number of elements in the Array :300
Time taken by Selection Sort is 14.554929%
akashm@Akashs-MacBook-Pro ADAPrograms % ./SelectionSort
Enter the number of elements in the Array :400
Time taken by Selection Sort is 26.356178%
akashm@Akashs-MacBook-Pro ADAPrograms % ./SelectionSort
Enter the number of elements in the Array :500
Time taken by Selection Sort is 40.571646%
akashm@Akashs-MacBook-Pro ADAPrograms %
```

Graph:



```c
//4.Write program to do the following:
  //.Print all the nodes reachable from a given starting node in a digraph using BFS method.

#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>

int q[30];
int f=-1,r=-1;


bool isEmpty(){
  if(f==-1 && r==-1){
    return true;
  }
  return false;
}

void enQueue(int i){
  if(f==-1 && r==-1){
    f=0;
  }
  q[++r]=i;
  return;
}

int deQueue(){
  if(isEmpty()){
    return -1;
  }
  if(f==r){
    int temp=q[f];
    f=-1;
    r=-1;
    return temp;
  }
  int temp=q[f];
  f++;
  return temp;

}

void bfs(int arr[10][10], int start, int visited[], int n, int *flag){
  visited[start]=1;
  enQueue(start);

  while(!isEmpty()){
    int pending=deQueue();
    printf("%d ",pending);
    for(int i=0;i<n;i++){
      if(arr[pending][i]==1 && visited[i]==0){
        enQueue(i);
        visited[i]=1;
        *flag=1;
```

```
        *flag = 1;
      }
    }
  }
}

int main(){
  int arr[10][10];
  int n,m;
  int visited[10];
  printf("Enter number of Vertices :");
  scanf("%d",&n);
  printf("Enter number of Edges :");
  scanf("%d",&m);
  for(int i = 0; i < n; ++i){
    for (int j = 0; j < n; ++j)
    {
      arr[i][j]=0;
    }
  }
  for (int i = 0; i < m; ++i)
  {
    int a,b;
    printf("Enter the Edge %d :",(i+1));
    scanf("%d %d",&a,&b);
    arr[a][b]=1;
    arr[b][a]=1;
  }
  for (int i = 0; i < n; ++i)
  {
    visited[i]=0;
  }
  int start;
  int flag=0;
  printf("Enter the starting vertex :");
  scanf("%d",&start);

  bfs(arr,start,visited,n,&flag);

  if(flag==0){
    printf("\nNo Path Found\n");
  }
}
```

Output:

```
akashm@Akashs-MacBook-Pro ADAPrograms % ./BFS
Enter number of Vertices :5
Enter number of Edges :6
Enter the Edge 1 :1 2
Enter the Edge 2 :2 3
Enter the Edge 3 :3 4
Enter the Edge 4 :1 3
Enter the Edge 5 :0 4
Enter the Edge 6 :0 1
Enter the starting vertex :0
0 1 4 2 3 %
akashm@Akashs-MacBook-Pro ADAPrograms % ./BFS
Enter number of Vertices :5
Enter number of Edges :4
Enter the Edge 1 :1 2
Enter the Edge 2 :2 3
Enter the Edge 3 :3 4
Enter the Edge 4 :1 3
Enter the starting vertex :0
0
No Path Found
```

```c
//4.Write program to do the following:
  //b. Check whether a given graph is connectedor not using DFS method.
#include<stdio.h>
#include<stdlib.h>

void dfs(int arr[10][10],int start,int visited[], int n){
  visited[start]=1;
  // printf("%d ",start);
  for(int i=0;i<n;i++){
    if(arr[start][i]==1 && visited[i]==0){
      dfs(arr,i,visited,n);
    }
  }
}

int main(){
  int arr[10][10];
  int n,m;
  int visited[10];
  printf("Enter number of Vertices :");
  scanf("%d",&n);
  printf("Enter number of Edges :");
  scanf("%d",&m);
  for(int i = 0; i < n; ++i){
    for (int j = 0; j < n; ++j)
    {
      arr[i][j]=0;
    }
  }
  for (int i = 0; i < m; ++i)
  {
    int a,b;
    printf("Enter the Edge %d :",(i+1));
    scanf("%d %d",&a,&b);
    arr[a][b]=1;
    arr[b][a]=1;
  }
  for (int i = 0; i < n; ++i)
  {
    visited[i]=0;
  }

  for (int i = 0; i < n; ++i)
  {
    if(i!=0 && visited[i]==0){
      printf("\nDisconnected Graph\n");
      return 0;
    }
    if(visited[i]==0)
      dfs(arr,i,visited,n);
  }
  printf("\nConnected Graph\n");
}
```

Output:

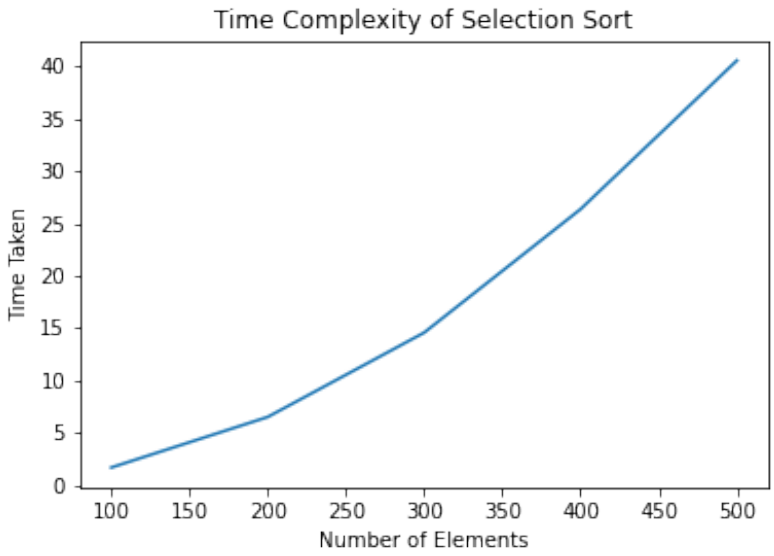
```
akashm@Akashs-MacBook-Pro ADAPrograms % ./DFS
Enter number of Vertices :5
Enter number of Edges :3
Enter the Edge 1 :0 4
Enter the Edge 2 :3 4
Enter the Edge 3 :2 3

Disconnected Graph
akashm@Akashs-MacBook-Pro ADAPrograms % ./DFS
Enter number of Vertices :5
Enter number of Edges :6
Enter the Edge 1 :0 1
Enter the Edge 2 :0 4
Enter the Edge 3 :1 2
Enter the Edge 4 :2 3
Enter the Edge 5 :3 4
Enter the Edge 6 :1 3

Connected Graph
```

```c
//5. Sort a given set of N integer elements using Insertion Sort technique and compute its time taken.

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


void delay(){
  for (int i = 0; i < 500000; ++i)
  {
    int d=i/33;
  }
}

void insertionSort(int arr[],int n){
  for (int i = 1; i < n ; ++i)
  {
    int temp=arr[i];
    int j;
    for(j=i-1;j>=0 && temp<arr[j]; j--){
      delay();
      arr[j+1]=arr[j];
    }
    arr[j+1]=temp;
  }
}

int main(){
  int arr[5000],n;
  printf("Enter the number of elements in the Array :");
  scanf("%d",&n);
  for(int i=0;i<n;i++){
    arr[i]=(rand() % (1 - 5000 + 1)) + 1;
  }

  clock_t start=clock();
  insertionSort(arr,n);
  clock_t end=clock();

  printf("Time taken by Insertion Sort is %f",(double)(end-start)/CLOCKS_PER_SEC);
}
```

Output:

```
akashm@Akashs-MacBook-Pro ADAPrograms % ./InsertionSort
Enter the number of elements in the Array :100
Time taken by Insertion Sort is 0.896186%
akashm@Akashs-MacBook-Pro ADAPrograms % ./InsertionSort
Enter the number of elements in the Array :200
Time taken by Insertion Sort is 3.658319%
akashm@Akashs-MacBook-Pro ADAPrograms % ./InsertionSort
Enter the number of elements in the Array :300
Time taken by Insertion Sort is 8.061177%
akashm@Akashs-MacBook-Pro ADAPrograms % ./InsertionSort
Enter the number of elements in the Array :400
Time taken by Insertion Sort is 13.283337%
akashm@Akashs-MacBook-Pro ADAPrograms % ./InsertionSort
Enter the number of elements in the Array :500
Time taken by Insertion Sort is 20.731952%
```

Graph:



```c
//6. Write program to obtain the Topological ordering of vertices in a given digraph.

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

int q[30];
int f=-1,r=-1;

bool isEmpty(){
  if(f==-1 && r==-1){
    return true;
  }
  return false;
}

void enQueue(int i){
  if(f==-1 && r==-1){
    f=0;
  }
  q[++r]=i;
  return;
}

int deQueue(){
  if(isEmpty()){
    return -1;
  }
  if(f==r){
    int temp=q[f];
    f=-1;
    r=-1;
    return temp;
  }
  int temp=q[f];
  f++;
  return temp;

}

void topoLogicalSorting(int arr[10][10],int n){
  int sorted[10];
  int k=0;
  int indegrees[10];
  for (int i = 0; i < n; ++i)
  {
    indegrees[i]=0;
  }
  for(int i=0;i<n;i++){
    for(int j=0;j<n;j++){
      if(arr[i][j]==1)
        indegrees[j]++;
    }
```

```c
    }
  }
  for (int i = 0; i < n; ++i)
  {
    if(indegrees[i]==0){
      enQueue(i);
    }
  }
  int visitedNodes=0;
  while(!isEmpty()){
    int snode=deQueue();
    sorted[k++]=snode;
    for(int i=0;i<n;i++){
      if(arr[snode][i]==1){
        if(--indegrees[i]==0){
          enQueue(i);
        }
      }
    }
    visitedNodes++;
  }
  if(visitedNodes!=n){
    printf("\nError!! Graph has a cycle\n");
    return;
  }

  for (int i = 0; i < n; ++i)
  {
    printf("%d ",sorted[i]);
  }
}

int main(){
  int arr[10][10];
  int n,m;
  int visited[10];
  printf("Enter number of Vertices :");
  scanf("%d",&n);
  printf("Enter number of Edges :");
  scanf("%d",&m);
  for(int i = 0; i < n; ++i){
    for (int j = 0; j < n; ++j)
    {
      arr[i][j]=0;
    }
  }
  for (int i = 0; i < m; ++i)
  {
    int a,b;
    printf("Enter the Edge %d :",(i+1));
    scanf("%d %d",&a,&b);
    arr[a][b]=1;
  }
  topoLogicalSorting(arr,n);
}
```

Output:

```
akashm@Akashs-MacBook-Pro ADAPrograms % ./TopoLogicalSort
Enter number of Vertices :4
Enter number of Edges :4
Enter the Edge 1 :2 0
Enter the Edge 2 :2 1
Enter the Edge 3 :1 0
Enter the Edge 4 :1 3
2 1 0 3 %
akashm@Akashs-MacBook-Pro ADAPrograms % ./TopoLogicalSort
Enter number of Vertices :4
Enter number of Edges :5
Enter the Edge 1 :2 0
Enter the Edge 2 :2 1
Enter the Edge 3 :1 0
Enter the Edge 4 :1 3
Enter the Edge 5 :3 0
2 1 3 0 %
akashm@Akashs-MacBook-Pro ADAPrograms % ./TopoLogicalSort
Enter number of Vertices :4
Enter number of Edges :5
Enter the Edge 1 :0 2
Enter the Edge 2 :2 1
Enter the Edge 3 :1 3
Enter the Edge 4 :1 0
Enter the Edge 5 :3 0

Error!! Graph has a cycle
```

```c
//7. Implement Johnson Trotter algorithm to generate permutations

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


int searchArray(int arr[], int n, int mobile){
  for(int i = 0; i < n;i++){
    if(arr[i]==mobile){
      return i+1;
    }
  }
  return 0;
}

int fact(int n){
  if(n==0 || n==1){
    return 1;
  }
  return n*fact(n-1);
}

int mobileSearch(int arr[], int n, int dir[]){
  int mobile=0, mobile_prev=0;
  for(int i=0;i<n;i++){
    if(dir[arr[i]-1]==0 && i!=0){
      if(arr[i]>arr[i-1] && arr[i]>mobile_prev){
        mobile=arr[i];
        mobile_prev=mobile;
      }
    }
  }

  for(int i = 0; i < n;i++){
    if(dir[arr[i]-1]==1 && i!=n-1){
      if(arr[i]>arr[i+1] && arr[i]>mobile_prev){
        mobile=arr[i];
        mobile_prev=mobile;
      }
    }
  }

  if(mobile != 0)
    return mobile;
  else
    return 0;
}

void permutationPrint(int arr[], int n, int dir[]){
  int mobile=mobileSearch(arr,n,dir);
  int getPos=searchArray(arr,n,mobile);

  if(dir[arr[getPos-1]-1]==0){
    int temp=arr[getPos-1];
    arr[getPos-1]=arr[getPos-2];
```

```
    arr[getPos-2]=temp;
  }
  if(dir[arr[getPos-1]-1]==1){
    int temp=arr[getPos-1];
    arr[getPos-1]=arr[getPos];
    arr[getPos]=temp;
  }


  for (int i = 0; i < n; ++i)
  {
    if(arr[i]>mobile && dir[arr[i]-1]==0){
      dir[arr[i]-1]=1;
    }

    else if(arr[i]>mobile && dir[arr[i]-1]==1){
      dir[arr[i]-1]=0;
    }
  }

  for (int i = 0; i < n; ++i)
  {
    printf("%d",arr[i]);
  }

  printf(" ");
}

void johnsonTrotter(int n){
  int dir[300];
  int arr[300];
  for(int i=0;i<n;i++){
    arr[i]=i+1;
  }

  for (int i = 0; i < n; ++i)
  {
    printf("%d",arr[i]);
  }

  printf(" ");

  for (int i = 0; i < n; ++i)
  {
    dir[i]=0;
  }

  for (int i = 1; i < fact(n); ++i)
  {
    permutationPrint(arr, n, dir);
  }

}
```

```c
int main(){
  int n;
  printf("Enter the value of N : ");
  scanf("%d",&n);
  printf("Permutations of 5 digit number is \n");
  johnsonTrotter(n);
}
```

Output:

```
akashm@Akashs-MacBook-Pro ADAPrograms % ./Johnsc
Enter the value of N : 4
Permutations of 5 digit number is
1234 1243 1423 4123 4132 1432 1342 1324 3124 314
2134 %
akashm@Akashs-MacBook-Pro ADAPrograms % ./Johnsc
Enter the value of N : 5
Permutations of 5 digit number is
12345 12354 12534 15234 51234 51243 15243 12543
1235 41325 41352 41532 45132 51432 51342 15342 1
252 31225 32125 32152 32512 35212 52312 52132 25
54 21245 21425 21452 21542 25142 52142 52412 254
2 24152 24125 21425 21452 21524 25124 52124 2512
 24152 24125 42125 42152 42512 45212 54212 52412
21254 21245 21425 21452 21542 %
```

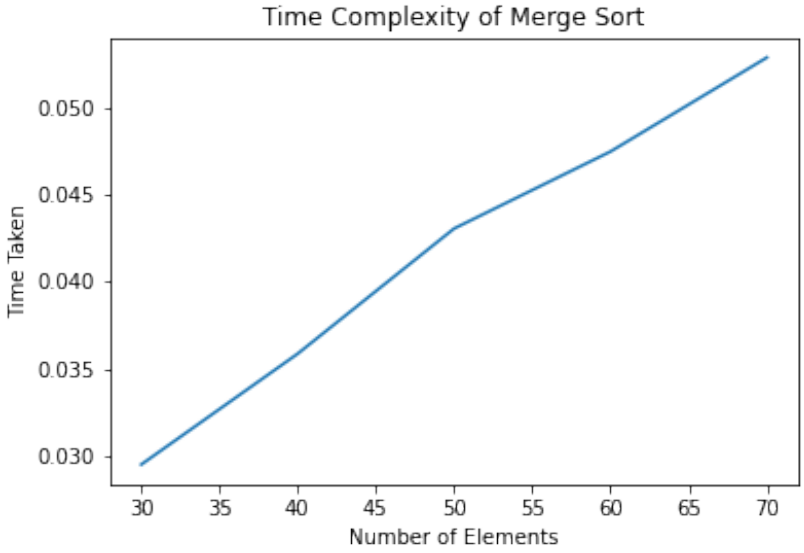
```c
//8. Sort a given set of N integer elements using Merge Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.

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


void delay(){
  for (int i = 0; i < 500000; ++i)
  {
    int d=i/33;
  }
}

void merge(int arr[],int si,int ei,int mid){
  int i=si,j=mid+1,k=si;
  int res[ei+1];
  while(i<=mid && j<=ei){
    if(arr[i]<arr[j]){
      res[k]=arr[i];
      k++;
      i++;
    }
    else if(arr[i]>arr[j]){
      res[k]=arr[j];
      k++;
      j++;
    }
  }
  while(i<=mid){
    res[k]=arr[i];
    i++;
    k++;
  }
  while(j<=ei){
    res[k]=arr[j];
    j++;
    k++;
  }
  for(int i=si;i<=ei;i++){
    arr[i]=res[i];
  }
}

void mergeSort(int arr[],int si,int ei){
  if(si>=ei){
    return;
  }
  delay();
  int mid=(si+ei)/2;

  mergeSort(arr,si,mid);
  mergeSort(arr,mid+1,ei);
```

```
  mergeSort(arr,mid+1,ei);

  merge(arr,si,ei,mid);

}

int main(){
  int arr[5000],n;
  printf("Enter the number of elements in the Array :");
  scanf("%d",&n);
  for(int i=0;i<n;i++){
    arr[i]=(rand() % (1 - 5000 + 1)) + 1;
  }
  clock_t start=clock();
  mergeSort(arr,0,n-1);
  clock_t end=clock();

  printf("Time taken by Merge Sort is %f",(double)(end-start)/CLOCKS_PER_SEC);
}
```

Output:

```
akashm@Akashs-MacBook-Pro ADAPrograms % ./mergeSort
Enter the number of elements in the Array :30
Time taken by Merge Sort is 0.029481%
akashm@Akashs-MacBook-Pro ADAPrograms % ./mergeSort
Enter the number of elements in the Array :40
Time taken by Merge Sort is 0.035862%
akashm@Akashs-MacBook-Pro ADAPrograms % ./mergeSort
Enter the number of elements in the Array :50
Time taken by Merge Sort is 0.043071%
akashm@Akashs-MacBook-Pro ADAPrograms % ./mergeSort
Enter the number of elements in the Array :60
Time taken by Merge Sort is 0.047507%
akashm@Akashs-MacBook-Pro ADAPrograms % ./mergeSort
Enter the number of elements in the Array :70
Time taken by Merge Sort is 0.052922%
```

Graph:



```c
//9. Sort a given set of N integer elements using Quick Sort technique and compute its time taken

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

void swap(int arr[],int i,int j){
  if(i==j){
    return;
  }
  int temp = arr[i];
  arr[i] = arr[j];
  arr[j] = temp;

}

int partition(int arr[],int si,int ei){
  int temp=arr[si];
  int count=0;
  for (int i = si+1; i <= ei; ++i)
  {
    if(temp>arr[i]){
      count++;
    }
  }
  swap(arr,si,si+count);

  int i=si, j=ei;

  while(i<j){
    if(arr[i]<temp){
      i++;
    }
    else if(arr[j]>=temp){
      j--;
    }
    else{
      swap(arr,i,j);
      i++;
      j--;
    }
  }
  return si+count;

}


void quickSort(int arr[],int si,int ei){
  if(si>=ei){
    return;
  }

  int i=partition(arr,si,ei);
  quickSort(arr,si,i-1);
```

```c
  quickSort(arr,i+1,ei);
}


int main(){
  int arr[5000],n;
  printf("Enter the number of elements in the Array :");
  scanf("%d",&n);
  for(int i=0;i<n;i++){
    arr[i]=(rand() % (1 - 5000 + 1)) + 1;
  }

  clock_t start=clock();
  quickSort(arr,0,n-1);
  clock_t end=clock();

  printf("Time taken by Quick Sort is %f",(double)(end-start)/CLOCKS_PER_SEC);
}
```

Output:

```
akashm@Akashs-MacBook-Pro ADAPrograms % ./QuickSort
Enter the number of elements in the Array :1000
Time taken by Quick Sort is 0.000348%
akashm@Akashs-MacBook-Pro ADAPrograms % ./QuickSort
Enter the number of elements in the Array :1500
Time taken by Quick Sort is 0.000720%
akashm@Akashs-MacBook-Pro ADAPrograms % ./QuickSort
Enter the number of elements in the Array :2000
Time taken by Quick Sort is 0.000978%
akashm@Akashs-MacBook-Pro ADAPrograms % ./QuickSort
Enter the number of elements in the Array :2500
Time taken by Quick Sort is 0.001283%
akashm@Akashs-MacBook-Pro ADAPrograms % ./QuickSort
Enter the number of elements in the Array :3000
Time taken by Quick Sort is 0.001402%
akashm@Akashs-MacBook-Pro ADAPrograms % ./QuickSort
Enter the number of elements in the Array :3500
Time taken by Quick Sort is 0.001820%
```

Graph: