National Institute of Technology Delhi

Department of Computer Science Engineering

Practical File

Subject: Design and Analysis of Algorithm

CSB 252

Name- Arpit Saharawat

Roll No.- 161210011

CSE 2nd Year

Index

S.No	Algorithm Name	Date	Teacher's Sign
1	Linear Search		
2	Binary Search		
3	Insertion Sort		
4	Quick Sort		
5	Min Heap		
6	Breadth First Search		
7	Depth First Search		
8	Fractional Knapsack		
9	0/1 Knapsack		
10	Travelling Salesman Problem		
11	Longest Common Subsequence		

Linear Search

```
#include<iostream>
using namespace std;
int
 main()
{
	int arr[10], i, num, n, c=0, pos;
	cout<<"Enter the array size : ";
	cin>>n;
	cout<<"Enter Array Elements : ";
	for(i=0; i<n; i++)
	{
		cin>>arr[i];
	}
	cout<<"Enter the number to be search : ";
	cin>>num;
	for(i=0; i<n; i++)
	{
		if(arr[i]==num)
		{
			c=1;
			pos=i+1;
			break;
		}
	}
	if(c==0)
	{
		cout<<"Number not found..!!";
	}
	else
	{
		cout<<num<<" found at position "<<pos;
	}
}
```

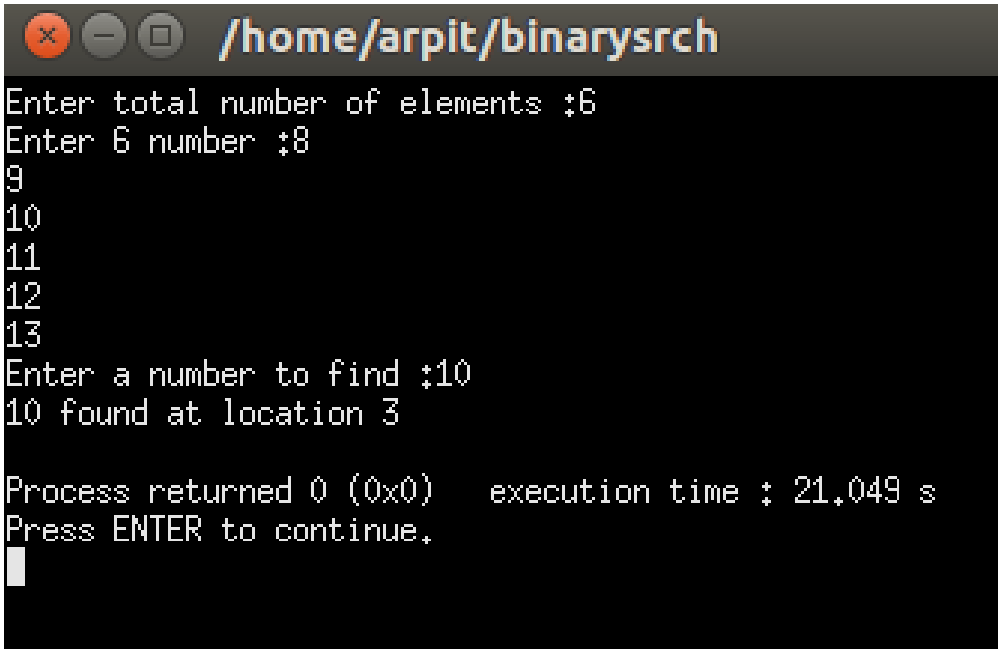
Output:

```
/home/arpit/linearsrch
Enter the array size : 5
Enter Array Elements : 5
6
7
2
8
Enter the number to be search : 2
2 found at position 4
Process returned 0 (0x0)   execution time : 15.469 s
Press ENTER to continue.
```

Binary Search

```
#include<iostream>
using namespace std;
int main()
{
	int n, i, arr[50], search, first, last, middle;
	cout<<"Enter total number of elements :";
	cin>>n;
	cout<<"Enter "<<n<<" number :";
	for (i=0; i<n; i++)
	{
		cin>>arr[i];
	}
	cout<<"Enter a number to find :";
	cin>>search;
	first = 0;
	last = n-1;
	middle = (first+last)/2;
	while (first <= last)
	{
		if(arr[middle] < search)
		{
			first = middle + 1;
		}
		else if(arr[middle] == search)
		{
			cout<<search<<" found at location "<<middle+1<<"\n";
			break;
		}
		else
		{
			 last = middle - 1;
		}
		middle = (first + last)/2;
	}
	if(first > last)
	{
		cout<<"Not found! "<<search<<" is not present in the list.";
}	}
```

Output:


/home/arpit/binarysrch
Enter total number of elements :6
Enter 6 number :8
9
10
11
12
13
Enter a number to find :10
10 found at location 3
Process returned 0 (0x0) execution time : 21.049 s
Press ENTER to continue.

Insertion Sort

```
#include<iostream>

using namespace std;

int main()
{
    int i,j,n,temp,a[30];
    cout<<"Enter the number of elements:";
    cin>>n;
    cout<<"\nEnter the elements\n";

    for(i=0;i<n;i++)
    {
        cin>>a[i];
    }

    for(i=1;i<=n-1;i++)
    {
        temp=a[i];
        j=i-1;

        while((temp<a[j])&&(j>=0))
        {
            a[j+1]=a[j];    //moves element forward
            j=j-1;
        }

        a[j+1]=temp;    //insert element in proper place
    }

    cout<<"\nSorted list is as follows\n";
    for(i=0;i<n;i++)
    {
        cout<<a[i]<<" ";
    }

    return 0;
}
```

Output:

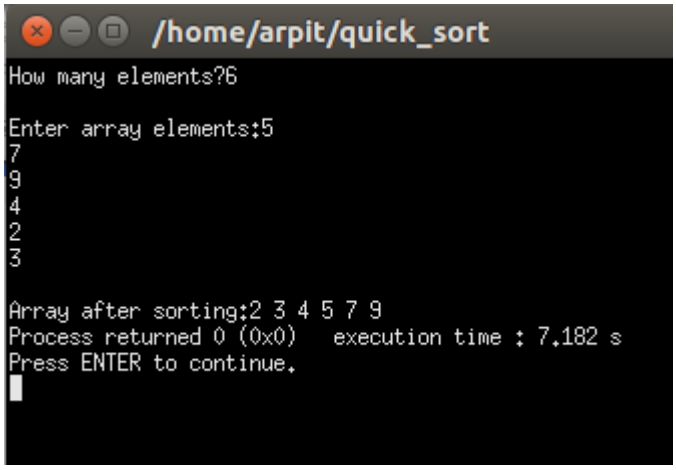
```
/home/arpit/insertion_sort
Enter the number of elements:5

Enter the elements
2
4
3
1
9

Sorted list is as follows
1 2 3 4 9
Process returned 0 (0x0)   execution time : 9.003 s
Press ENTER to continue.
```

Quick Sort

```
#include <iostream>

using namespace std;

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

int main()
{
    int a[50],n,i;
    cout<<"How many elements?";
    cin>>n;
    cout<<"\nEnter array elements:";

    for(i=0;i<n;i++)
        cin>>a[i];

    quick_sort(a,0,n-1);
    cout<<"\nArray after sorting:";

    for(i=0;i<n;i++)
        cout<<a[i]<<" ";

    return 0;
}

void quick_sort(int a[],int l,int u)
{
    int j;
    if(l<u)
    {
        j=partition(a,l,u);
        quick_sort(a,l,j-1);
        quick_sort(a,j+1,u);
    }
}

int partition(int a[],int l,int u)
{
```

```
    int v,i,j,temp;
    v=a[l];
    i=l;
    j=u+1;

    do
    {
        do
            i++;

        while(a[i]<v&&i<=u);

        do
            j--;
        while(v<a[j]);

        if(i<j)
        {
            temp=a[i];
            a[i]=a[j];
            a[j]=temp;
        }
    }while(i<j);

    a[l]=a[j];
    a[j]=v;

    return(j);
}
```

Output:



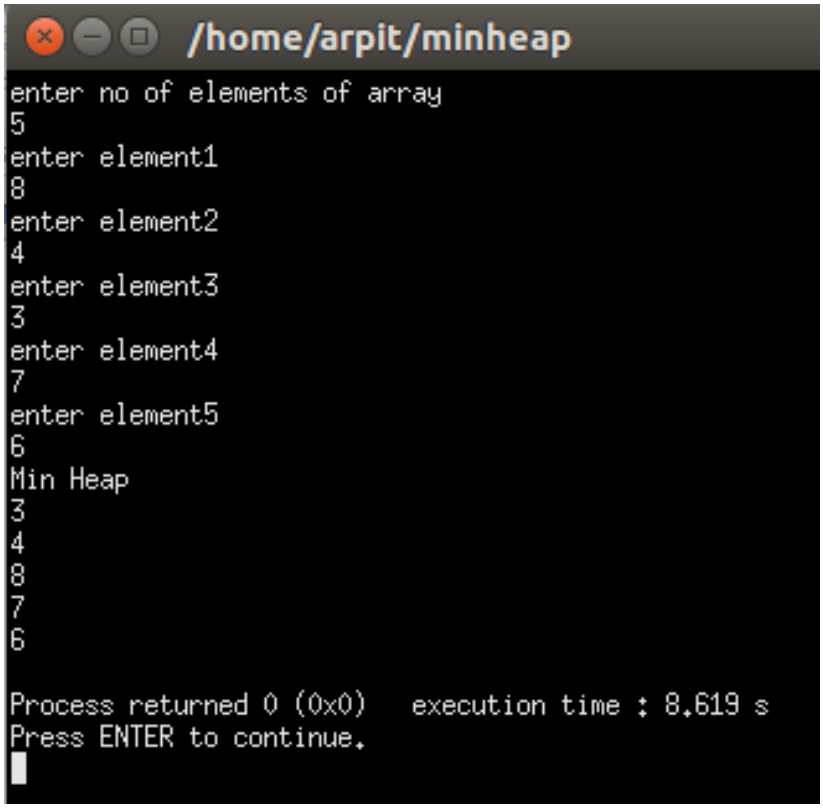
Min Heap

```
#include <iostream>
using namespace std;
void min_heapify(int *a,int i,int n)

{
    int j, temp;

    temp = a[i];

    j = 2 * i;

    while (j <= n)

    {

        if (j < n && a[j+1] < a[j])

            j = j + 1;

        if (temp < a[j])

            break;

        else if (temp >= a[j])

        {

            a[j/2] = a[j];

            j = 2 * j;

        }

    }

    a[j/2] = temp;

    return;
```

```
}
void build_minheap(int *a, int n)
{
    int i;
    for(i = n/2; i >= 1; i--)
    {
        min_heapify(a,i,n);
    }
}
int main()
{
    int n, i, x;
    cout<<"enter no of elements of array\n";
    cin>>n;
    int a[20];
    for (i = 1; i <= n; i++)
    {
        cout<<"enter element"<<(i)<<endl;
        cin>>a[i];
    }
    build_minheap(a, n);
```

```
    cout<<"Min Heap\n";

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

    {

        cout<<a[i]<<endl;

    }

}
```

Output:



Breadth First Search

```
#include<iostream>
using namespace std;
int cost[10][10],i,j,k,n,qu[10],front,rare,v,visit[10],visited[10];
int main()
{
int m;
cout <<"Enter no of vertices:";
cin >> n;
cout <<"Enter no of edges:";
cin >> m;
cout <<"\nEDGES \n";
for(k=1; k<=m; k++)
{
cin >>i>>j;
cost[i][j]=1;
}
cout <<"Enter initial vertex to traverse from:";
cin >>v;
cout <<"Visitied vertices:";
cout <<v<<" ";
visited[v]=1;
k=1;
while(k<n)
{
for(j=1; j<=n; j++)
if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1)
{
visit[j]=1;
qu[rare++]=j;}
v=qu[front++];
cout<<v <<" ";
k++;
visit[v]=0;
visited[v]=1;
}
return 0;
}
```

Output:

```
/home/arpit/bfs
Enter no of vertices:5
Enter no of edges:6

EDGES
1 2
2 3
1 3
2 4
3 4
4 5
Enter initial vertex to traverse from:1
Visitied vertices:1 2 3 4 5
Process returned 0 (0x0)   execution time : 25.028 s
Press ENTER to continue.
```

Depth First Search

```
#include<iostream>
using namespace std;
int cost[10][10],i,j,k,n,stk[10],top,v,visit[10],visited[10];
int main()
{
int m;
cout <<"enterno of vertices";
cin >> n;
cout <<"ente no of edges";
cin >> m;
cout <<"\nEDGES \n";
for(k=1;k<=m;k++)
{
cin >>i>>j;
cost[i][j]=1;
}
cout <<"enter initial vertex";
cin >>v;
cout <<"ORDER OF VISITED VERTICES"<<endl;
cout << v <<" ";
visited[v]=1;
k=1;
while(k<n)
{
for(j=n;j>=1;j--)
if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1)
{visit[j]=1;
stk[top]=j;
top++;
}
v=stk[--top];
cout<<v << " ";
k++;
visit[v]=0; visited[v]=1;
}
return 0;
}
```

Output:

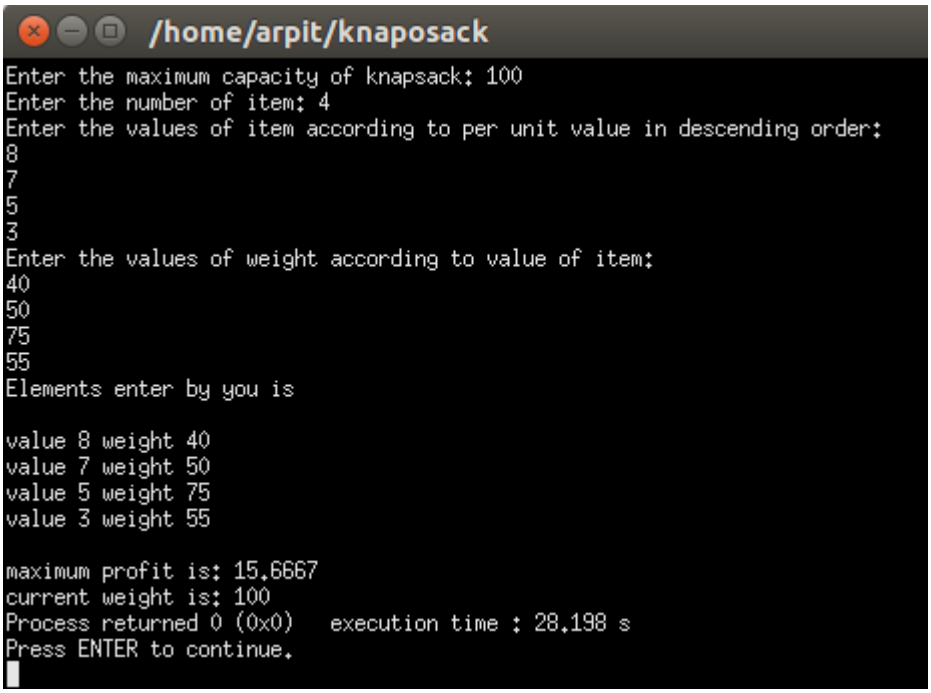
```
/home/arpit/dfs
enterno of vertices5
ente no of edges7

EDGES
1 2
2 3
2 4
4 5
3
5
4 3
4 2
enter initial vertex1
ORDER OF VISITED VERTICES
1 2 3 5 4
Process returned 0 (0x0)   execution time : 37.079 s
Press ENTER to continue.
```

Fractional Knapsack

```
#include<iostream>
using namespace std;
void knapsack(float*,float*,float,float);
int main()
{
float value[50],weight[50],max;
int i,n;
cout<<"Enter the maximum capacity of knapsack: ";
cin>>max;
cout<<"Enter the number of item: ";
cin>>n;
cout<<"Enter the values of item according to per unit value in descending
order:\n";
for( i=0;i<n;i++)
{cin>>value[i];
}
cout<<"Enter the values of weight according to value of item:\n";
for( i=0;i<n;i++)
{cin>>weight[i];
}
cout<<"Elements enter by you is \n";
for( i=0;i<n;i++)
{cout<<"\nvalue "<<value[i]<<" weight "<<weight[i];
}
knapsack(value,weight,max,n);
}
void knapsack(float value[],float weight[],float max,float n)
{ float current=0,rem;int i;
float currentval=0;
while((current<=max)&& (weight[i]<=max-current))
{ current=current+weight[i];
currentval=currentval+value[i];
i++;
}
rem=max-current;
if(current<max)
{current =current+rem;
currentval=currentval+((rem * value[i])/weight[i] );
}
```

```
cout<<"\n\nmaximum profit is: "<<currentval;
cout<<"\ncurrent weight is: "<<current;
}
```

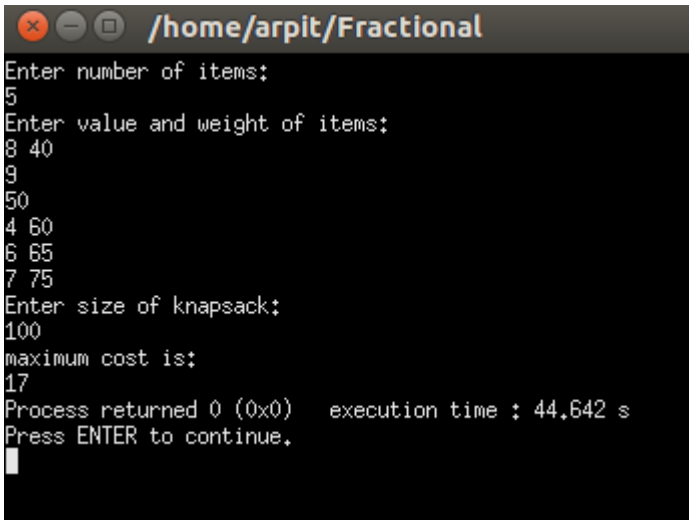
Output:



0/1 Knapsack

```
#include<iostream>
using namespace std;
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 i, n, val[20], wt[20], W;
cout<<"Enter number of items:"<<endl;
cin>>n;cout<<"Enter value and weight of items:\n";
for(i = 0;i < n; ++i){
cin>>val[i]>>wt[i];
}
cout<<"Enter size of knapsack:"<<endl;
cin>>W;
cout<<"maximum cost is:"<<endl;
cout<< knapSack(W, wt, val, n);
return 0;
}
```

Output:


/home/arpit/Fractional
Enter number of items:
5
Enter value and weight of items:
8 40
9
50
4 60
6 65
7 75
Enter size of knapsack:
100
maximum cost is:
17
Process returned 0 (0x0) execution time : 44.642 s
Press ENTER to continue.

Travelling Salesman Problem

```
#include<iostream>
using namespace std;
int ary[10][10],completed[10],n,cost=0;
void takeInput()
{
int i,j;
cout<<"Enter the number of villages: ";
cin>>n;
cout<<"\nEnter the Cost Matrix\n";
for(i=0;i < n;i++)
{
cout<<"\nEnter Elements of Row: "<<i+1<<"\n";
for( j=0;j < n;j++)
cin>>ary[i][j];
completed[i]=0;
}
cout<<"\n\nThe cost list is:";
for( i=0;i < n;i++)
{cout<<"\n";
for(j=0;j < n;j++)
cout<<"\t"<<ary[i][j];
}
}
int least(int c)
{
int i,nc=999;
int min=999,kmin;
for(i=0;i < n;i++)
{
if((ary[c][i]!=0)&&(completed[i]==0))
if(ary[c][i]+ary[i][c] < min)
{
min=ary[i][0]+ary[c][i];
kmin=ary[c][i];
nc=i;
}
}
```

```
if(min!=999)
cost+=kmin;
return nc;
}
void mincost(int city)
{
int i,ncity;completed[city]=1;
cout<<city+1<<"--->";
ncity=least(city);
if(ncity==999)
{
ncity=0;
cout<<ncity+1;
cost+=ary[city][ncity];
return;
}
mincost(ncity);
}
int main()
{
takeInput();
cout<<"\n\nThe Path is:\n";
mincost(0); //passing 0 because starting vertex
cout<<"\n\nMinimum cost is "<<cost;
return 0;
}
```

Output:

```
Enter the number of villages: 6

Enter the Cost Matrix

Enter Elements of Row: 1
1
5
4
8
9
3

Enter Elements of Row: 2
5

7
8
9
45
6

Enter Elements of Row: 3
2
4
9
8
7
6

Enter Elements of Row: 4
3
6
5
4
7
8

Enter Elements of Row: 5
9
4
8
7
6
2

Enter Elements of Row: 6
1
9
8
7
3
4


The cost list is:
        1       5       4       8       9       3
```

```
The cost list is:
        1       5       4       8       9       3
        5       7       8       9       45      6
        2       4       9       8       7       6
        3       6       5       4       7       8
        9       4       8       7       6       2
        1       9       8       7       3       4

The Path is:
1--->6--->5--->2--->3--->4--->1

Minimum cost is 29
Process returned 0 (0x0)   execution time : 31.909 s
Press ENTER to continue.
```

Longest Common Subsequence

```
#include<iostream>
#include<string.h>
using namespace std;
int i,j,m,n,c[20][20];
char x[20],y[20],b[20][20];
void print(int i,int j)
{
if(i==0 || j==0)
return;
if(b[i][j]=='c')
{
print(i-1,j-1);
cout<<x[i-1];
}
else if(b[i][j]=='u')
print(i-1,j);
else
print(i,j-1);
}
void lcs()
{
m=strlen(x);
n=strlen(y);
for(i=0;i<=m;i++)
c[i][0]=0;
for(i=0;i<=n;i++)
c[0][i]=0;
for(i=1;i<=m;i++)for(j=1;j<=n;j++)
{
if(x[i-1]==y[j-1])
{
c[i][j]=c[i-1][j-1]+1;
b[i][j]='c';
}
else if(c[i-1][j]>=c[i][j-1])
```

```
{
c[i][j]=c[i-1][j];
b[i][j]='u';
}
else
{
c[i][j]=c[i][j-1];
b[i][j]='l';
}
}
}
int main()
{
cout<<"Enter 1st sequence:"<<endl;
cin>>x;
cout<<"Enter 2nd sequence:"<<endl;
cin>>y;
cout<<"\nThe Longest Common Subsequence is "<<endl;
lcs();
print(m,n);
return 0;
}
```

Output: