Skill development
Assignment – 1

AIM:-

To create ADT that implement the "set" concept.

a)Add (newElement) -Place a value into the set
b)Remove (element)
c)Contains (element) Return true if element is in collection
d)Size () Return number of values in collection
e)Intersection of two sets
f)Union of two sets
g)Difference between two sets
h)Subset

OBJECTIVE:

To get the thorough understanding of the concepts of sets and the various operations performed on it

THEORY:-

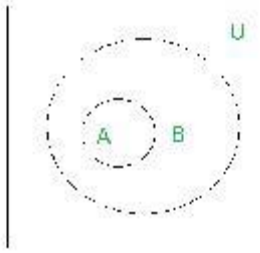
• SET:-

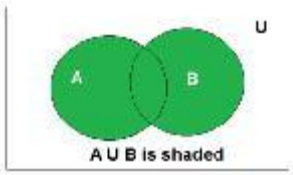
A **Set** is an unordered collection of objects, known as elements or members of the set.
An element 'a' belong to a set A can be written as 'a ∈ A', 'a ∉A' denotes that a is not an element of the set A.

• EQUAL SETS:-

Two sets are said to be equal if both have same elements. For example A = {1, 3, 9, 7} and B = {3, 1, 7, 9} are equal sets.

Name: shivam parve **Batch**: B1 **Roll no**: 222020 **Gr no** :17u113





SKILL DEVELOPTMENT LAB-II 2018-19

• SUBSET:-

A set A is said to be **subset** of another set B if and only if every element of set A is also a part of other set B.
Denoted by '⊆'. 'A ⊆ B ' denotes A is a subset of B.

• SIZE OF A SET:-

Size of a set can be finite or infinite.

Size of the set S is known as Cardinality number, denoted as |S|.

Example: Let A be a set of odd positive integers less than 10.

Solution : A = {1,3,5,7,9}, Cardinality of the set is 5, i.e.,|A| = 5.

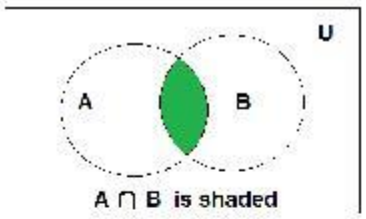
•UNION :

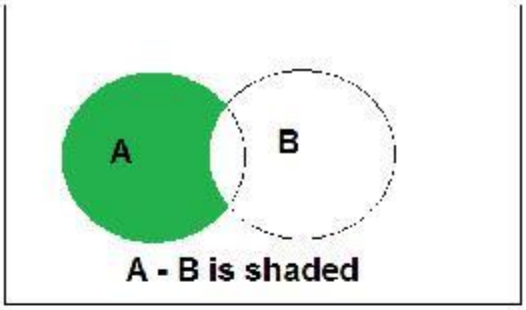
Union of the sets A and B, denoted by A ∪ B, is the set of distinct element belongs to set A or set B, or both.

•INTERSECTION:

The intersection of the sets A and B, denoted by A of elements belongs to both A and B i.e. set of the element in A and B

∩B, is the set common


U
A
B
A ∩ B is shaded


A
B
A - B is shaded

• SET DIFFERENCE:-

Difference between sets is denoted by 'A – B', is the set containing elements of set A but not in B. i.e all elements of A except the element of B.

ALGORITHM:

• FOR INTERSECTION:

Step 1: Take an empty set (intersection set)
Step 2: pass each element of set-2 and the entire set-1 to the function member()
Step 3: if it returns true,
Add that element to the intersection set

• FOR UNION:

Step 1: Take an empty set (union set)
Step 2: copy all the elements of set1 to this new set
Step 3: traverse through the set2 and pass each element of set-2 along with the entire set-1 to the function member(),

and if it returns false then add that specified element to the union set

• FOR CONTAINS:

Step1 : take the number as input which you want to search
Step 2 : enter 1 for searching in set-1 or 2 for searching in set-2
Step 3 : initialise i=0
Step 4: traverse the set-1 or set-2 till the end depending on whether the input was 1 or 2 after passing the element and that set to the function member()
Step 5: if element found then display element is present

• FOR SUBSET:

Step 1: Enter 1 if you want to check if set 2 is subset of 1, or enter 2 if you want to check if set-1 is subset of set-2.

Step 2: Depending on input we will traverse the set(which has to be the subset) until its end, by passing each element of this set and other set to the function member().

Step 3: If member() returns true, then continue else return false

Step 4: If false, then display "it is a subset" else display "it is not"

• FOR DIFFERENCE :

Step 1: Initialise the difference set to 0, difference set contains all the element. which are in set-1 but not in set-2

Step 2: Traverse the entire set-1 and, pass each element of this set and the set-2 to the function member()

Step 3: if it returns false then add this element to the difference set

• FOR REMOVE:

Step 1: enter 1 or 2 for removing element from set-1 or set-2 respectively
Step 2: enter the index from which you want to remove the element
Step 3: if entered position is less than the size of the set then move all the elements to their left from the position at which you want to remove the element and just decrease the size of the set else, entered position is equal to the size of the set then just decrease the size

• FOR SIZE:

step 1: show the 0th index of the set which contains the size of our set

Program:

```
#include<iostream>
using namespace std;


void create(int a[], int n)
 {

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

                }
 }

void display( int a[],int no)
{
cout<<"{";
for (int i=0; i<no; i++)
{
cout<<a[i];
cout<<"\t ";
```

```
}
cout<<"}";
}


int union1(int a[], int b[] , int c[], int no1,int no2)
{
int i,j,count=0,k=0;
for( i=0; i<no1; i++)
{
c[i]=a[i];
k++;
}

for( j=0 ;j<no2 ;j++)
{
count=0;

for (i=0; i<no1 ;i++)
  {
        if(a[i]== b[j])
        {
        count++;

        }
  }

        if(count==0)
        {

        c[k]=b[j];
        k++;
```

```
        }


 }

return k;

}

int difference(int a[], int b[] , int c[], int no1,int no2)
{

int i,j,k=0,count=0;

for(i=0; i<no1;i++)

{
count=0;

for(j=0; j<no2;j++)


  {

        if(a[i]==b[j])
        {

        count++;
        }
```

```
  }
if(count==0)
        {
        c[k]=a[i];
        k++;

        }


}

return k;

}


int intersection(int a[], int b[] , int c[], int no1,int no2)
{

int i,j,k=0,count=0;

for(i=0; i<no1;i++)

{
count=0;

for(j=0; j<no2;j++)


  {

        if(a[i]==b[j])
```

```
        {

        count++;
        }


  }
if(count!=0)
        {
        c[k]=a[i];
        k++;

        }


}

return k;

}

int subset(int a[], int b[] , int no1,int no2)
{
int count=0;

                for(int i=0; i<no1; i++)
{
        for(int j=0; j<no2;j++)

        {
                if(a[i]==b[j])
                {
```

```
                count++;
            }

        }

}

cout<<" \n\n the count is ( no of elements in set 2 )"<<count;
if(count==no2)
{
        return 1;
        }
        else
        {
            return 1;
        }


}

void remove(int a[],int no)
{

            int pos,i;
        cout<<" which position you wnat to remove   "; cin>>pos;

        for( i=pos-1; i<no;i++)
        {
            a[i]=a[i+1];

        }
        a[i]=0;
```

```
}

void modify(int a[],int no)
{
        int pos,i,ele;
    cout<<" which position you wnat to modify   "; cin>>pos;

    cout<<"\n enter the new element   ";
    a[pos]=ele;

}


int main()
{
int no_ele,choise;
int result[100];
int no1,no2,ch1;
cout<<"\n enter the no of elemnts you want to insert in set 1  "; cin>>no1;

int arr1[no1];

cout<<"\n enter the no of elemnts you want to insert in set 2  "; cin>>no2;

int arr2[no2];

cout<<"enter the elements in 1st set  ";
create(arr1,no1);
cout<<"enter the elements in 2nd set  ";
create(arr2,no2);
```

```
do
{
cout<<"\n 1. union\n 2.difference \n 3. intersection\n 4. subset\n 5.remove \n
6.display\n 0.exit";
cin>>ch1;

switch(ch1)
{
case 1:
no_ele=union1(arr1,arr2,result,no1,no2);
cout<<"\n the unoin is   ";
 display(result,no_ele);
break;



case 2:

        {
                cout<<"\n\n  ";
                cout<<"\n1.a-b \n 2.b-a";
                cin>>choise;

                switch(choise)
                {
                        case 1:
                        no_ele=difference(arr1,arr2,result,no1,no2);
          cout<<"\n the difference  is   ";
```

```
        display(result,no_ele);
        break;

                case 2:
                        no_ele=difference(arr2,arr1,result,no1,no2);
        cout<<"\n the difference  is   ";
        display(result,no_ele);
        break;


            }


            break;
        }

case 3:
no_ele=intersection(arr1,arr2,result,no1,no2);
cout<<"\n the intersection  is   ";
 display(result,no_ele);
break;

case 4:
        {

        no_ele=subset(arr2,arr1,no1,no2);
 if(no_ele==1)
 {
        cout<<" yes!!! it is subset  ";
 }
        else
                {
```

```
            cout<<"  not a subset  ";
              }
break;
}

case 5:
        {
              cout<<"\n which set you want to change 1st or second   ";
              cin>>choise;

              if(choise==1)
              {
                    remove(arr1,no1);
              }
              else
              {
                    remove(arr2,no2);
              }

              break;
        }

        case 6:
              {
                    cout<<"\n which set you want to display 1st or second   ";
              cin>>choise;

              if(choise==1)
              {
                    display(arr1,no1);
              }
              else
```

```
                {
                        display(arr2,no2);
                }

        break;
        }case 7:
        {
        cout<<"\n which set you want to modify 1st or second   ";
                cin>>choise;
                if(choise==1)
                {modify(arr1,no1);
                }
                else
                {
                        modify(arr2,no2);
                }
                break;
}
}while(ch1!=0);



return 0;




}
```

College : VIIT pune **branch:** computer science **batch**: 2017(pattern)

Name: shivam parve **Batch**: B1 **Roll no**: 222020 **Gr no** :17u113

OUTPUT:

```
enter the no of elemnts you want to insert in set 1  4

enter the no of elemnts you want to insert in set 2  4

enter the elements in 1st set  1

2

3

4

enter the elements in 2nd set  54

2
```

3

67

1. union

2.difference

3. intersection

4. subset

5.remove

6.display

0.exit1

the unoin is {1 2 3 4 54 67 }

1. union

2.difference

3. intersection

4. subset

5.remove

6.display

0.exit2

1.a-b

2.b-a1

the difference is {1 4 }

1. union

2.difference

3. intersection

4. subset

5.remove

6.display

0.exit2

1.a-b

2.b-a2

the difference is {54 67 }

1. union

2.difference

3. intersection

4. subset

5.remove

6.display

0.exit3

the intersection is {2 3 }

1. union

2.difference

3. intersection

4. subset

5.remove

6.display

0.exit4

the count is (no of elements in set 2)2 yes!!! it is subset

1. union

2.difference

3. intersection

4. subset

5.remove

6.display

0.exit5

which set you want to change 1st or second 1

which position you wnat to remove 3

1. union

2.difference

3. intersection

4. subset

5.remove

6.display

0.exit6

which set you want to display 1st or second 1

{1 2 4 4782944 }

1. union

2.difference

3. intersection

4. subset

5.remove

6.display

0.exit6

which set you want to display 1st or second 2

{54 2 3 67 }

1. union

2.difference

3. intersection

4. subset

5.remove

6.display

CONCLUSION:

We understood the concepts of sets and the various operations performed on them, and were able to apply those concepts through programming.

Skill development

Assignment – 2

Aim:

Construct a threaded binary search tree by inserting values in the given order and traverse it in inorder traversal using threads.

Objective:

To understand the following Concepts of Threaded Binary Search Tree (TBT):

i.Creating a TBT using tree data structure.
ii.Inorder traversal using threads.

THEORY:

Threaded Binary Tree:

Inorder traversal of a Binary tree can either be done using recursion or with the use of a auxiliary stack. The idea of threaded binary trees is to make inorder traversal faster and do it without stack and without recursion. A binary tree is made threaded by making all right child pointers that would normally be NULL point to the inorder successor of the node (if it exists).

There are two types of threaded binary trees.

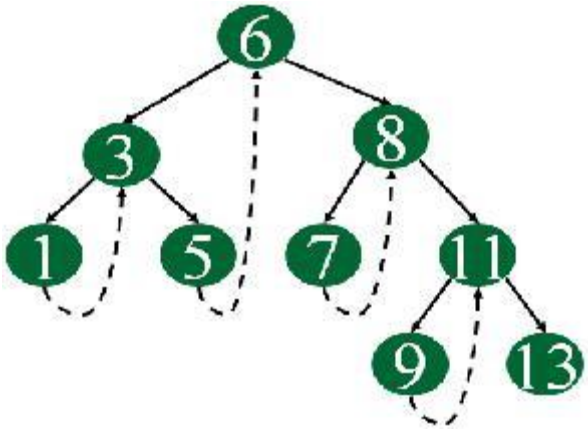
Single Threaded: Where a NULL right pointers is made to point to the inorder successor (if successor exists)

Double Threaded: Where both left and right NULL pointers are made to point to inorder predecessor and inorder successor respectively. The predecessor threads are useful for reverse inorder traversal and postorder traversal.

The threads are also useful for fast accessing ancestors of a node.

Following diagram shows an example Single Threaded Binary Tree. The dotted lines represent threads.

1



ALGORITHM:

Non recursive Inorder traversal for a Threaded Binary Tree

1.curr-node node leftmost (root)

2.While (curr_node != Null)

a.print (curr_node)

b.If (curr_node.RTag == 0) then

curr_node <- curr_node.right go to step 2.

c. else curr_node <- leftmost(curr_node.right) go to step 2.

CODE:

```
#include<iostream>
using namespace std;

class node
{
        node *left;
        node *right;
        int data,lth,rth;
        public:
        friend class BST;

};

class BST
{
        node *top, *dummy;

        public:

                BST()
                {
                        top=NULL;
                        dummy=NULL;
                }

                void insert(node *top,node *newnode)
                {
```

```
        if(newnode->data<top->data)
        {
                if(top->lth==0)
                {
                        newnode->left=top->left;
                        top->left=newnode;
                                newnode->right=top;
                        top->lth=1;

                }
                else
                insert(top->left,newnode);
        }
        else if(newnode->data>top->data)
        {
                if(top->rth==0)
                {
                newnode->right=top->right;
                top->right=newnode;
                newnode->left=top;
                top->rth=1;

                }
                else
                insert(top->right,newnode);

        }

else
cout<<"duplicate data insertded  ";

        }
```

```
void create()
{
    node *newnode= new node;
        newnode->left=NULL;
        newnode->right=NULL;
        cout<<" enter the data  ";
        cin>>newnode->data;
        newnode->lth=0;
        newnode->rth=0;

    if(top==NULL)
    {
        top=newnode;
        dummy= new node;
        dummy->data=-999;
        dummy->lth=dummy->rth=0;
        dummy->left=NULL;
        dummy->right=NULL;
        top->left=top->right=dummy;
    }
    else
    {

insert(top,newnode);
}
}


    node *returntop()
```

```
        {
                return top;
        }

        void display(node *top)
        {
                while(top!=dummy)
                {
                        while(top->lth==1)
                        {
                                top=top->left;
                        }
                        cout<<top->data<<"\t";
                        while(top->rth==0)
                        {
                                top=top->right;
                        if(top==dummy)
                        return;
                        cout<<top->data<<"\t";

                        }
                        top=top->right;

                }

        }


};

int main()
{
```

```
        int h,c;
        node *top;
        top=NULL;
        BST b1;
        for(int i=0;i<5;i++)
        {
                b1.create();
        }

        node *temp=b1.returntop();
        b1.display(temp);
        return 0;
}

// 2    7      10     15     20      no of nodes  5 height of the tree  3
```

OUTPUT

:

```
enter the data  34
 enter the data  12
 enter the data  67
 enter the data  43
 enter the data  11
11      12      34      43      67
--------------------------------
Process exited after 10.84 seconds with return value 0
Press any key to continue . . .
```


CONCLUSION:

Study and operations like creating TBT and Inorder traversal using threads was performed successfully.

Skill development

Assignment – 3

Aim :

There are flight paths between cities. If there is a flight between city A and city B then there is an edge between the cities. The cost of the edge can be the time that flight takes to reach city B from A, or the amount of fuel used for the journey. Represent this as a graph. The node can be represented by airport name or name of the city. Use adjacency list representation of the graph or use adjacency matrix representation of the graph. Justify the storage representations used.

Objectives:

To understand the various operations on graphs.

Theory:

Dijkstra's algorithm is very similar to Prim's algorithm for minimum spanning tree. Like Prim's MST, we generate a shortest path tree with given source as root. We maintain two sets, one set contains vertices included in shortest path tree, and other set includes vertices not yet included in shortest path tree.

For Example:

At every step of the algorithm, we find a vertex which is in the other set and has a minimum distance from the source. Below are the detailed steps used in Dijkstra's algorithm to find the shortest path from a single source vertex to all other vertices in the given graph.

Algorithm:

1.

2.

3.

Create priority queue pq

Enqueue(pq,s)

For(i=1;i<=g->v;i++)

Distance[i]=-1

4.Distance[s]=0

..

5.while(!isemptyqueue(pq))

{

5.1v=deletemin(pq);

5.2 for all adjacent vertices w to v

{

Compute new distance d=distance[v]+weight[v][w];

If(Distance[w]==-1)

{

Distance[w]=new distance d;

Insert w in priorityqueue with priority d

Path[w]=v}

If(Distance[w]>newdistance d)

{distance[w]=new disance d; Update priority 0f vertex w to be d; Path[w]=v;

}

}}}

Program:

```
#include<iostream>
using namespace std;

class cost
{
        int n_v,n_e,u,v,value;
        int G[100][100];
        public:
        cost()
        {
                cout<<"enter the no of cities in path ";
                cin>>n_v;

                for(int i=0;i<n_v;i++)
                {
                for(int j=0;j<n_v;j++)
                {
                        G[i][j]=0;
                        }

                }

        }

        void assign()
        {
```

```
            cout<<"enter the no of paths in between cites ";
            cin>>n_e;

            for(int i=0;i<n_e;i++)
            {
                  cout<<"enter the statring city ending city and the cost of fuel";
                  cin>>u>>v>>value;

                  G[u][v]= G[v][u]=value;

            }

      }

      void display()
      {

                  for(int i=0;i<n_v;i++)
            {
                  cout<<endl;
            for(int j=0;j<n_v;j++)
            {
            cout<<      G[i][j]<<"\t";
                  }

            }

      }


};

int main()
{
      cost t;

      t.assign();
```

```
        t.display();
        return 0;



        }
```

/*Output:

```
enter the no of cities in path 4
enter the no of paths in between cites 5
enter the statring city ending city and the cost of fuel0
1
12
enter the statring city ending city and the cost of fuel1
2
23
enter the statring city ending city and the cost of fuel3
4
56
enter the statring city ending city and the cost of fuel1
4
57
enter the statring city ending city and the cost of fuel0
3
89

0       12      0       89
12      0       23      0
0       23      0       0
89      0       0       0
--------------------------
Process exited after 24.81 seconds with return value 0
Press any key to continue . . .
```

Conclusion:

From above experiment we learnt how to use shortest path algorithm using graph operation.

..

Skill development

Assignment – 4

Aim:

For a weighted graph G, find the minimum spanning tree using Prims Algorithm.

Objective:

Understand the concepts of prims algorithm

Theory:

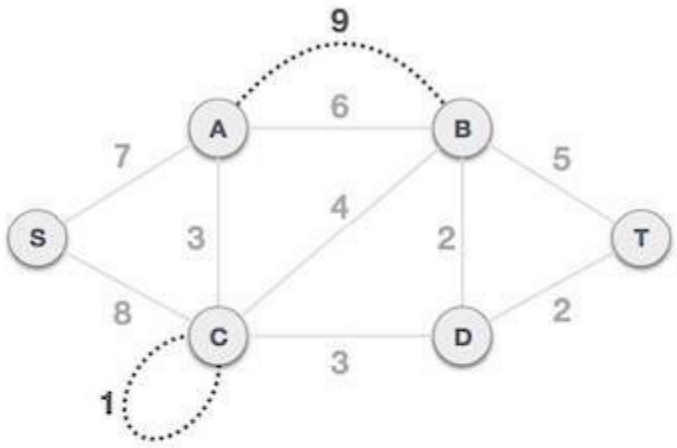
Prim's algorithm to find minimum cost spanning tree (as Kruskal's algorithm) uses the greedy approach. Prim's algorithm shares a similarity with the **shortest path first** algorithms.

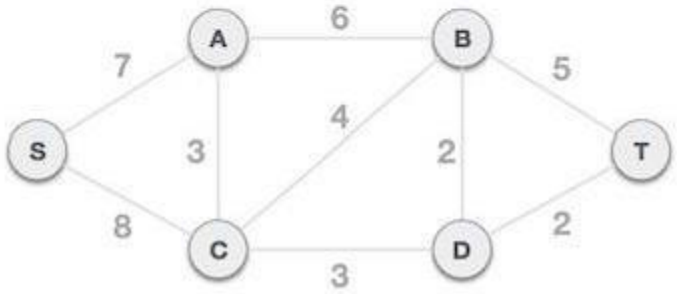
Prim's algorithm, in contrast with Kruskal's algorithm, treats the nodes as a single tree and keeps on adding new nodes to the spanning tree from the given graph.

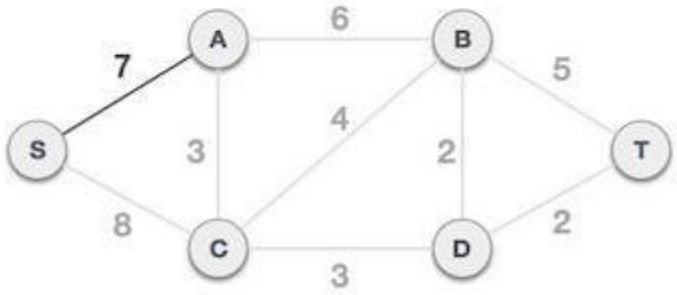
To contrast with Kruskal's algorithm and to understand Prim's algorithm better, we shall use the same example −

Step 1 - Remove all loops and parallel edges

..


9
A
B
6
7
5
4
S
3
2
T
8
C
D
2
3
1


A
B
6
7
5
4
S
3
2
T
8
C
D
2
3


A
B
6
7
5
4
S
3
2
T
8
C
D
2
3

Remove all loops and parallel edges from the given graph. In case of parallel edges, keep the one which has the least cost associated and remove all others.

Step 2 - Choose any arbitrary node as root node

In this case, we choose **S** node as the root node of Prim's spanning tree. This node is arbitrarily chosen, so any node can be the root node. One may wonder why any video can be a root node. So the answer is, in the spanning tree all the nodes of a graph are included and because it is connected then there must be at least one edge, which will join it to the rest of the tree.

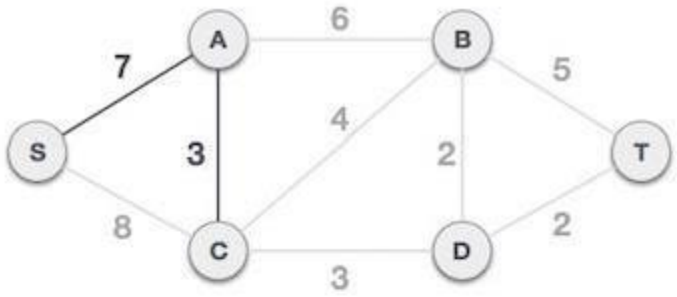
Step 3 - Check outgoing edges and select the one with less cost

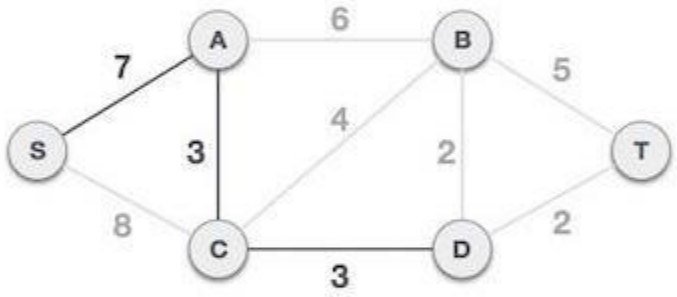
After choosing the root node **S**, we see that S,A and S,C are two edges with weight 7 and 8, respectively. We choose the edge S,A as it is lesser than the other.

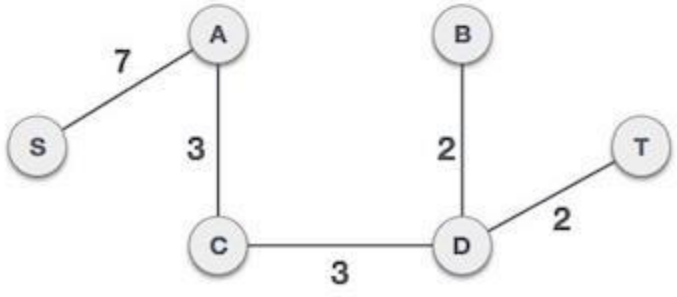
Now, the tree S-7-A is treated as one node and we check for all edges going out from it. We select the one which has the lowest cost and include it in the tree.

2

..







After this step, S-7-A-3-C tree is formed. Now we'll again treat it as a node and will check all the edges again. However, we will choose only the least cost edge. In this case, C-3-D is the new edge, which is less than other edges' cost 8, 6, 4, etc.

After adding node **D** to the spanning tree, we now have two edges going out of it having the same cost, i.e. D-2-T and D-2-B. Thus, we can add either one. But the next step will again yield edge 2 as the least cost. Hence, we are showing a spanning tree with both edges included.

We may find that the output spanning tree of the same graph using two different algorithms is same.

Algorithm:

Algorithm Prims(E,cost,n,t)

3

..

{1.Let (k,l) be the edge of minimum cost 2.mincost=cost(k,l)

3.t[1,1]=k;t[1,2]=l; 4.for i=1 to n do

If(cost[i,l]<cost[i,k] then near[i]=l

Else near[i]=k;

5.near[k]=near[l]=0 6.for i=2 to n-1 do

6.1Let j be the index such that near[j]!=0 and Cost[j,near[j]] is minimum

6.2t[i,1]=j ;t[i ,2]=near[j]

6.3mincost=mincost+cost[j,near[j]];

6.4near[j]=0

6.5for k=1 to n do

if ((near[k]!=0) and (cost[k,near[k]]>cost[k,j])) then near[k]=j}

Return mincost

}

C++ Code:

```
#include<iostream>
using namespace std;

int minimum(int *v,int *d, int n)
{

        int index;
        int min=9999;

        for(int i=0;i<n;i++)
          {
              if(d[i]<min && v[i]==0)
              {
                    min= d[i];
                    index=i;

                }
```

```
        }
        return index;

}



int main()
{
        int n_v,n_e,u,v,value;

        cout<<"enter the no of vertices and no of edges";
        cin>>n_v>>n_e;

        int g[n_v][n_v];
        int parent[n_v];
        int visited[n_v];
        int distance[n_v];

        for(int i=0;i<n_v;i++)
        {
                distance[i]=9999;
                parent[i]=0;
                visited[i]=0;

        }

for(int i=0;i<n_v;i++)
        for(int j=0;j<n_v;j++)
             g[i][j]=0;

        distance[0]=0;
```

```
        for(int i=0;i<n_e;i++)
         {
                cout<<"enter the u,v,value";
                cin>>u>>v>>value;

        g[u][v]=g[v][u]=value;
         }

cout<<"the g matrix is";
  for(int i=0;i<n_v;i++)
  {
        cout<<endl;
        for(int j=0;j<n_v;j++)
        {
        cout<<g[i][j]<<"\t";
        }
  }
```

```
for(int j=0;j<n_v-1;j++)
{
        int v= minimum(visited, distance,n_v);
             visited[v]=1;

             cout<<"the "<<j<<"loop run"<<endl;
             cout<<"the minimum value is"<<v<<endl;


        for(int i=0;i<n_v;i++)
        {
```

```
                if(g[v][i]!=0 && (distance[i]>g[v][i]) && visited[i]==0)
                {
                        distance[i]=g[v][i];
                        parent[i]=v;

                }


        }

/*      cout<<"the distance matrix is"<<endl;
        for(int i=0;i<n_v;i++)
  cout<<"0 ->"<<i<<" "<<"= "<<distance[i]<<endl;

        cout<<"the parent matrix is "<<endl;
        for(int i=0;i<n_v;i++)
  cout<<i<<" "<<"= "<<parent[i]<<endl;*/

}
cout<<"**************************************0";
cout<<"\n\n\n\n";
cout<<"the path summary is";
for(int i=0;i<n_v;i++)
  {
        cout<<" vertex1  "<<" vertex2 "<< " distance ";
        cout<<i<<" --->"<<parent[i]<<" == "<<distance[i];
        cout<<endl;
  }


int p,sum;
cout<<"enter -1 to close";
do
{
        sum=0;
        cout<<"enter the destination";
cin>>p;
```

```
cout<<"required path"<<p;
sum=sum+distance[p];
        while(p!=0)
{
        p=parent[p];
        sum+=distance[p];
        cout<<"<--"<<p;

}
cout<<"total path length "<<sum<<endl;

}while(p!=-1);


        return 0;
```

}**Output:**

```
enter the no of vertices and no of edges5
6
enter the u,v,value0
1
10
enter the u,v,value0
3
5
enter the u,v,value1
2
3
enter the u,v,value2
3
31
enter the u,v,value1
4
20
enter the u,v,value2
4
22
the g matrix is
0       10      0       5       0
```

```
10      0       3       0       20
0       3       0       31      22
5       0       31      0       0
0       20      22      0       0       the 0loop run
the minimum value is0
the 1loop run
the minimum value is3
the 2loop run
the minimum value is1
the 3loop run
the minimum value is2

the path summary is
 vertex1   vertex2  distance
 0 --->0 == 0
 vertex1   vertex2  distance
  1 --->0 == 10
 vertex1   vertex2  distance
  2 --->1 == 3
 vertex1   vertex2  distance
  3 --->0 == 5
 vertex1   vertex2  distance
  4 --->1 == 20

enter -1 to closeenter the destination4
required path4<--1<--0
total path length 30

enter the destination2
required path2<--1<--0
total path length 13

enter the destination3
required path3<--0
total path length 5
enter the destination
```

Conclusion:

This assignment is used how prims algorithm is used in solving the example using vertex edge .

.

Skill development

Assignment – 5

Aim :

You have a business with several offices; you want to lease phone lines to connect them up with each other; and the phone company charges different amounts of money to connect different pairs of cities. You want a set of lines that connects all your offices with a minimum total cost. Solve the problem by suggesting appropriate data structures

Objective:

To understand the application of Prims algorithm to find the minimum spanning tree.

Theory:

Prim's algorithm to find minimum cost spanning tree (as Kruskal's algorithm) uses the greedy approach. Prim's algorithm shares a similarity with the **shortest path first** algorithms.

Prim's algorithm, in contrast with Kruskal's algorithm, treats the nodes as a single tree and keeps on adding new nodes to the spanning tree from the given graph.

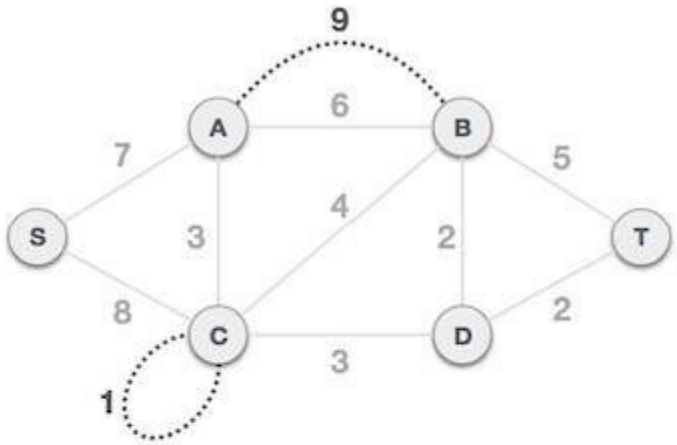
To contrast with Kruskal's algorithm and to understand Prim's algorithm better, we

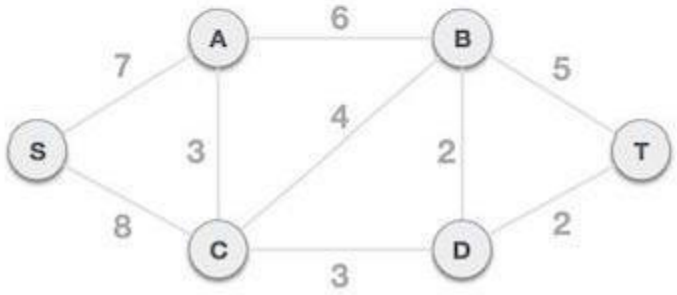
shall use the same example −

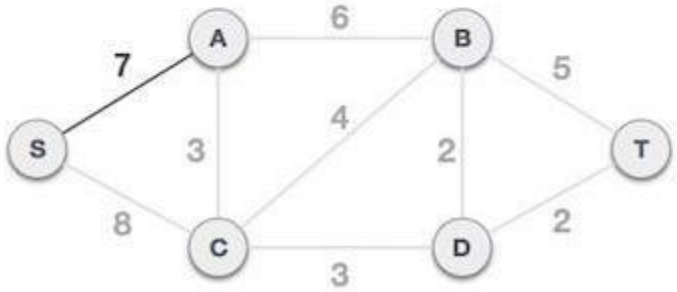
Step 1 - Remove all loops and parallel edges

1

..


9
A
B
6
7
5
4
S
3
2
T
8
C
D
2
3
1


A
B
6
7
5
4
S
3
2
T
8
C
D
2
3


A
B
6
7
5
4
S
3
2
T
8
C
D
2
3

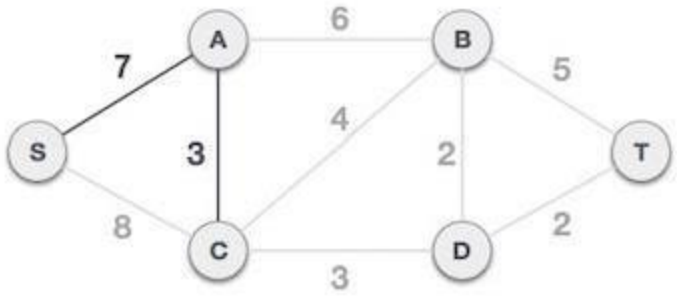
Remove all loops and parallel edges from the given graph. In case of parallel edges, keep the one which has the least cost associated and remove all others.

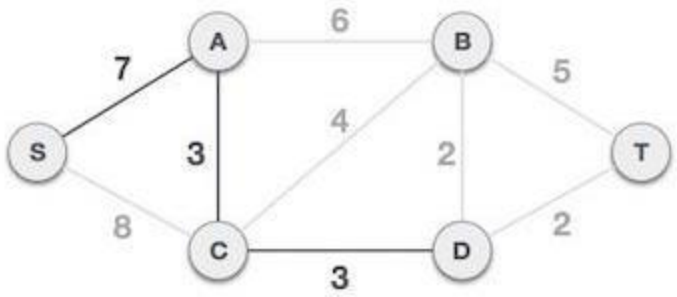
Step 2 - Choose any arbitrary node as root node

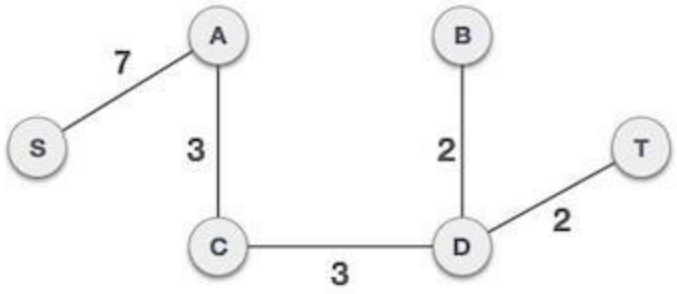
In this case, we choose **S** node as the root node of Prim's spanning tree. This node is arbitrarily chosen, so any node can be the root node. One may wonder why any video can be a root node. So the answer is, in the spanning tree all the nodes of a graph are included and because it is connected then there must be at least one edge, which will join it to the rest of the tree.

Step 3 - Check outgoing edges and select the one with less cost

After choosing the root node **S**, we see that S,A and S,C are two edges with weight 7 and 8, respectively. We choose the edge S,A as it is lesser than the other.







Now, the tree S-7-A is treated as one node and we check for all edges going out from it. We select the one which has the lowest cost and include it in the tree.

After this step, S-7-A-3-C tree is formed. Now we'll again treat it as a node and will check all the edges again. However, we will choose only the least cost edge. In this case, C-3-D is the new edge, which is less than other edges' cost 8, 6, 4, etc.

After adding node **D** to the spanning tree, we now have two edges going out of it having the same cost, i.e. D-2-T and D-2-B. Thus, we can add either one. But the next step will again yield edge 2 as the least cost. Hence, we are showing a spanning tree with both edges included.

We may find that the output spanning tree of the same graph using two different algorithms is same.

..

Algorithm:

Algorithm Prims(E,cost,n,t)

{1.Let (k,l) be the edge of minimum cost 2.mincost=cost(k,l)

3.t[1,1]=k;t[1,2]=l; 4.for i=1 to n do

If(cost[i,l]<cost[i,k] then near[i]=l

Else near[i]=k;

5.near[k]=near[l]=0 6.for i=2 to n-1 do

6.1Let j be the index such that near[j]!=0 and Cost[j,near[j]] is minimum

6.2t[i,1]=j ;t[i ,2]=near[j]

6.3mincost=mincost+cost[j,near[j]];

6.4near[j]=0

6.5for k=1 to n do

if ((near[k]!=0) and (cost[k,near[k]]>cost[k,j])) then near[k]=j}

Return mincost

}

Code:

```
#include<iostream>
using namespace std;

int minimum(int *v,int *d, int n)
{

        int index;
        int min=9999;

        for(int i=0;i<n;i++)
          {
              if(d[i]<min && v[i]==0)
              {
                      min= d[i];
                      index=i;

                }
```

```
        }
        return index;

}



int main()
{
        int n_v,n_e,u,v,value;

        cout<<"enter the no of CITIES and no of PATHS between them";
        cin>>n_v>>n_e;

        int g[n_v][n_v];
        int parent[n_v];
        int visited[n_v];
        int distance[n_v];

        for(int i=0;i<n_v;i++)
        {
                distance[i]=9999;
                parent[i]=0;
                visited[i]=0;

        }

for(int i=0;i<n_v;i++)
        for(int j=0;j<n_v;j++)
             g[i][j]=0;

        distance[0]=0;
```

```
        for(int i=0;i<n_e;i++)
         {
                cout<<"enter the stating city- destination- charge by phone company
";
                cin>>u>>v>>value;

        g[u][v]=g[v][u]=value;
         }

cout<<"the cost matrix is";
  for(int i=0;i<n_v;i++)
  {
        cout<<endl;
        for(int j=0;j<n_v;j++)
        {
        cout<<g[i][j]<<"\t";
        }
  }




for(int j=0;j<n_v-1;j++)
{
        int v= minimum(visited, distance,n_v);
             visited[v]=1;

             cout<<"the "<<j<<"loop run"<<endl;
             cout<<"the minimum value is"<<v<<endl;


        for(int i=0;i<n_v;i++)
        {
                if(g[v][i]!=0 && (distance[i]>g[v][i]) && visited[i]==0)
                {
                        distance[i]=g[v][i];
```

```
                        parent[i]=v;

                    }


            }

/*      cout<<"the distance matrix is"<<endl;
        for(int i=0;i<n_v;i++)
  cout<<"0 ->"<<i<<" "<<"= "<<distance[i]<<endl;

        cout<<"the parent matrix is "<<endl;
        for(int i=0;i<n_v;i++)
  cout<<i<<" "<<"= "<<parent[i]<<endl;*/

}
cout<<"****************************************0";
cout<<"\n\n\n\n";
cout<<"the path summary is";
for(int i=0;i<n_v;i++)
  {
        cout<<" vertex1  "<<" vertex2 "<< " distance ";
        cout<<i<<" --->"<<parent[i]<<" == "<<distance[i];
        cout<<endl;
  }

int sum1=0;
for(int i=0;i<n_v;i++)
    sum1+=distance[i];

cout<<"total cost for all telephone line setup "<<sum1<<endl;

int p,sum;
cout<<"enter -1 to close";
do
{
        sum=0;
        cout<<"enter the destination";
```

```
cin>>p;

cout<<"required path"<<p;
sum=sum+distance[p];
        while(p!=0)
{
        p=parent[p];
        sum+=distance[p];
        cout<<"<--"<<p;

}
cout<<"total path length "<<sum<<endl;

}while(p!=-1);


        return 0;
}
```

Output:

..

Conclusion:

We understood the implementation of Prims algorithm in real life problems.

Skill development

Assignment – 6

Aim :

Read the marks obtained by students of second year in an online examination of particular subject. Find out maximum and minimum marks obtained in that subject using heap data structure.

Objective :

To find maximum and minimum marks obtained by the students in second year in a particular subject using a binary heap (either max heap or min heap) and then sorting the heap using heap sort algorithm for desired output.

Theory :

A binary heap is a complete binary tree which satisfies the heap ordering property.

The ordering can be one of two types:

- the min-heap property: the value of each node is greater than or equal to the value of its parent, with the minimum-value element at the root.
- the max-heap property: the value of each node is less than or equal to the value of its parent, with the maximum-valueelement at the root.

We create a heap by adding numbers from left to right level by level. Heap can be implemented using an array or a priority queue.

For sorting the heap, after it's creation, the first position of the array would contain either the smallest or the largest element depending on whether max

heap or min heap is created ,heap sort algorithm swaps the first element in the heap with the last one and heapify the heap excluding the last element and reduce

..

the size of the array by one. Repeat the steps until the complete heap is sorted.

Code :

```
#include <iostream>


using namespace std;

void max_heapify(int *arr, int i, int n)

{

   int largest=i;

   int l=2*i+1;

   int r=2*i+2;


   if(l<n && arr[l]> arr[largest])

      largest=l;




   if(r<n && arr[r]> arr[largest])
```

```
        largest=r;


    if(largest!=i)
    {
          swap(arr[i],arr[largest]);


          max_heapify(arr,largest,n);


          }



}


void min_heapify(int *arr, int i, int n)
{
    int minimum=i;
    int l=2*i+1;
    int r=2*i+2;


    if(l<n && arr[l]< arr[minimum])
        minimum=l;
```

```
    if(r<n && arr[r]< arr[minimum])
        minimum=r;

    if(minimum!=i)
    {
         swap(arr[i],arr[minimum]);

         min_heapify(arr,minimum,n);

         }

}

void swap(int *a, int *b)
```

```
{
        int temp=*a;
        *a=*b;
        *b=temp;




}

void build_minheap(int *a,int n)
{
   int i;
   for(i = n/2-1; i >= 0; i--)
   {
      min_heapify(a,i,n);
   }

}


void build_maxheap(int *a,int n)
{
```

```
    int i;
    for(i = n/2-1; i >= 0; i--)
    {
        max_heapify(a,i,n);
    }



}



void sort( int * a, int n)
{
        int size= n;

        for(int i=n-1;i>=0;i--)
        {
                int last= a[i];
        int     first=a[0];
                a[i]=a[0];
                a[0]=last;

                size--;
```

```
            max_heapify(a,0,i);


        }



}
int main()
{
   int n, i, x;
   cout<<"enter no of students in class\n";
   cin>>n;
   int a[20];
   int b[20];
   for (i = 0; i < n; i++)
   {
      cout<<"enter marks obtained by students"<<(i)<<endl;
      cin>>a[i];
      b[i]=a[i];
   }
   build_maxheap(a,n);
   cout<<"Max Heap\n";
```

```
    for (i = 0; i < n; i++)
    {
        cout<<a[i]<<endl;
    }

   cout<<"the maximun marks obtained by students "<<a[0]<<endl;

    cout<<"min heap";
    build_minheap(b,n);
    for (i = 0; i < n; i++)
    {
        cout<<b[i]<<endl;
    }
    cout<<"the minimum marks obtained by students "<<b[0]<<endl;
```

}Output:

enter no of students in class

5

enter marks obtained by students0

34

enter marks obtained by students1

67

enter marks obtained by students2

4

enter marks obtained by students3

12

enter marks obtained by students4

56

Max Heap

67

56

4

12

34

the maximun marks obtained by students 67

min heap4

12

34

67

56

the minimum marks obtained by students 4

Process exited after 8.014 seconds with return value 0

Press any key to continue . . .

Conclusion-:

The objective of creating a heap from an array was completed. The heap was then sorted to give out the maximum and minimum marks obtained by students using heap sort algorithm.

..

Skill development

Assignment – 7

Aim:

Insert the keys into a hash table of length m using open addressing using double hashing with h(k)=1+(k mod (m-1)).

Objective:

To understand :

1.How keys can be mapped to the corresponding values , in a hash table, in order to have the lowest time complexity.

2.How collisions can be resolved , in a hash table , using a second hash function.

Theory:

Double hashing is a computer programming technique , used in hash tables to resolve hash collisions, in cases when two different values to be searched for produce the same hash key. It is a popular collision -resolution technique in open-addressedhash tables. Double hashing is implemented in many popular libraries.

Like linear probing, it uses one hash value as a starting point and then repeatedly steps forward an interval until the desired value is located, an empty location is reached, or the entire table has been searched; but this interval is decided using a second, independent hash function (hence the name double hashing). Unlike linear probing and quadratic probing , the interval depends on the data, so that even values mapping to the same location have different bucket sequences; this minimizes repeated collisions and the effects of clustering.

First hash function is typically hash1(key) = key % TABLE_SIZE

A popular second hash function is : **hash2(key) = PRIME – (key % PRIME)** where PRIME is a prime smaller than the TABLE_SIZE.
A good second Hash function is:

- It must never evaluate to zero
- Must make sure that all cells can be probed

Example:

1

..

Algorithm:

1.Start.
2.Accept the size of the table.
3.Initialize the hash table array to any negative integer value say "-111"(Providednegative keys are not accepted in the table).
4.Map the key to it's value, using first hash function: hash1(key) = key % Table_size.
5.If collision occurs use the second hash function: hash2(key) = 1+(key mod (size-1)).
6.Do: Hi(key)=((Hash(key) + i * hash2(key)) mod size) , using a for loop, for i from 1 to (size-1), untill the key gets mapped to it's appropriate value.
7.Stop.

Code:

2

..

```
#include<iostream>
using namespace std;


class hashTable
{
```

```
public:
        int data[10],occ[10];
        int key,index=0,index2=0,n;

hashTable()
{

        for(int i=0;i<10;i++)
        {
                occ[i]=0;
                data[i]=0;
        }


}

void insert();
void calIndex();
void display();
void search();
void delet();


};
```

```
void hashTable::insert()
{

        cout<<"\n\n\tHow many Keys u Want To Enter?? ";
        cin>>n;

for(int i=0;i<n;i++)
{
        cout<<"\n\n\tEnter Key Value";
        cin>>key;

        index = (key % 10);
        calIndex();
}

}
```

```
void hashTable::calIndex()
{
```

```
        if(occ[index]==0)
        {
        data[index] = key;
        occ[index] = 1;
        }
        else if(occ[index] == 1)
        {
        for(int j=0;j<10;j++)
        {
                index2 = 7 - (key % 7);

                index = (index + j*index2)%10;

                if(occ[index] == 0)
                        break;
        }
        data[index] = key;
        occ[index] = 1;

        }
}
```

```
void hashTable::display()
{
	cout<<"\t\t\tIndex "<<"\t\tKey\n";
	for(int i=0;i<10;i++)
	cout<<"\t\t\t"<<i<<"\t\t"<<data[i]<<"\n";

}

/*
void hashTable::delet()
{
	int del;
	cout<<"\n\n\tEnter Key to be Deleted ";
	cin>>del;

	for(int i=0;i<10;i++)
	{
	if(data[i]==del)
	{
		cout<<"\n\t\t"<<del<<" Deleted from Index "<<i<<"\n";
		data[i]=0;
		occ[i]=0;
	}
	}
```

```
}
*/

void hashTable::search()
{
        int search;
        cout<<"\n\n\tEnter Key to be Searched ";
        cin>>search;

        for(int i=0;i<10;i++)
        {
        if(data[i]==search)
        {
                cout<<"\n\t\t"<<search<<" Found at Index "<<i<<"\n";
        }
        }
}



int main()
{
        int ch;
```

```
        hashTable h1;

        do{

        cout<<"Enter Ur Choice\n1.Insert\n2.Display\n3.Search\n0.Exit\n";
        cin>>ch;

        switch(ch)
        {
                case 1: h1.insert();
                                break;
                case 2: h1.display();
                                break;
                case 3: h1.search();
                                break;

        }
}while(ch!=0);

}
```

OUTPUT:

```
Enter Ur Choice
1.Insert
2.Display
3.Search
0.Exit
1

How many Keys u Want To
Enter?? 5

Enter Key Value11

Enter Key Value23

Enter Key Value21

Enter Key Value45

Enter Key Value67
Enter Ur Choice
1.Insert
2.Display
3.Search
0.Exit
2
Index    Key
0        0
1        11
2        0
3        23
4        0
5        45
6        0
7        67
8        21
```

9 0
Enter Ur Choice
1.Insert
2.Display
3.Search
0.Exit
3

Enter Key to be Searched 67

67 Found at Index 7
Enter Ur Choice
1.Insert
2.Display
3.Search
0.Exit7

..

Conclusion:

Hence Double Hashing can be used in this way to solve problem of collision.

..

Skill development

Assignment – 8

Aim:

Department maintains a student information. The file contains roll number, name, division and address.

Allow user to add, delete information of student. Display information of particular employee. If record of

student does not exist an appropriate message is displayed. If it is, then the system displays the student details Use Sequential file to maintain data

Objective:

Understand the concepts of sequential file handling

Theory:

A file is a collection of related data stored in a particular area on the disk

A File can be opened in the following ways

File mode para meter	Meaning

ios::app	Append to end of file
ios::ate	go to end of file on opening
ios::binary	file open in binary mode
ios::in	open file for reading only
ios::out	open file for writing only
ios::nocreate	open fails if the file does not e xist

1

..

ios::noreplace

ios::trunc
open fails if the file already exi st

delete the contents of the file i f it exist
When we want to move file pointer to desired position then use these function to manage the file pointers.

Seekg () = moves get pointer (input) to a
specified location

Seekp () = moves put pointer (output) to a
specified location

tellg () = gives the current position of the get pointer

tellp () = gives the current position of the put pointer

file . read ((char *)&V , sizeof (V)); file . Write ((char *)&V , sizeof (V));

These function take two arguments. The first is the address of the variable V , and the second is the length of that variable in bytes . The address of variable must be cast to type char * (i.e pointer to character type) .

Algorithm:

1. Take the count of number of students from the user

2

..

2.Make an array of object of the student class which stores the information of the students

3.Open a file by using the ofstream object

4.Take the information of the student from the user and write it to the file

5.User can perform 1.Search 2.Delete 3.Display operations

6.For Search

1.Input the Roll number to be searched

2.Open the file using Ifstream object in input mode

3.Read the contents of the file in an object sequentially and check it with the roll number to be searched if found Display found message and the details of the students

4.If not found continue till end of file

5.If eof is reached display the message Not found

7.For Delete

1.Input the roll number to be deleted

2. Open the Main file in input mode and a temporary file in output mode

3.Sequentially search through the main file and copy the contents to the temp file except the roll number to be deleted

4.Delete the contents of the Main file

5.Rename the temp file with the name of the main file

8.For Display

1.Open the file in input mode and display the details of all the students sequentially

C++ Code:

```
#include<iostream>
#include<fstream>
using namespace std;
class student
{
int roll_num;
char div;
```

3

..

```
string name; string address;
public:
void getdata()
{
cout<<"\n Enter the Roll Number"; cin>>roll_num;
cout<<"\n Enter the division "; cin>>div;
```

```
cout<<"\n Enter the Name"; fflush(stdin); getline(cin,name); cout<<"\n Enter the
Address"; fflush(stdin); getline(cin,address);
}
void putdata(int n)
{
   student st[n]; ifstream infile; infile.open("student.dat",ios::binary|ios::in); for(int
i=0;i<n;i++)
{
   infile.read((char *)&st[i],sizeof(st[i])); cout<<"\n Roll Number: "<<st[i].roll_num;
cout<<"\n Division: "<<st[i].div; fflush(stdin);
cout<<"\n Name: "<<st[i].name; fflush(stdin);
cout<<"\n Address: "<<st[i].address;
```



```
..

cout<<"\n
_____ \n";
}
infile.close();
}

void search_(int n)
{
student st[n]; ifstream infile;
cout<<"\n Enter the Roll Number to be searched"; int r;
```

```
cin>>r;
infile.open("student.dat",ios::in|ios::binary); for(int i=0;i<n;i++)
{
infile.read((char *)&st[i],sizeof(st[i])); if(st[i].roll_num==r)
{
cout<<"\n Found"; cout<<"\n Details: "<<endl;
   cout<<"\n Roll Number: "<<st[i].roll_num; cout<<"\n Division: "<<st[i].div;
fflush(stdin);

cout<<"\n Name: "<<st[i].name; fflush(stdin);
cout<<"\n Address: "<<st[i].address;
cout<<"\n
_____ \n";
infile.close();
```



```
    ..

return;
}
}
cout<<"\n Not Found"; infile.close();
}
void del(int n)
{
student st[n]; int r;
cout<<"\n Enter the roll number to be deleted "; cin>>r;
```

```
ifstream infile; ofstream outfile; infile.open("student.dat",ios::binary|ios::in);
outfile.open("temp.dat",ios::binary|ios::out); for(int i=0;i<n;i++)
{
infile.read((char *)&st[i],sizeof(st[i])); if(st[i].roll_num==r)
{
continue;
}
else
{
outfile.write((char *)&st[i],sizeof(st[i]));

}
}
outfile.close();
```


..

```
infile.close();
remove("student.dat");
int re=rename("temp.dat","student.dat");

}


};
int main()
```

```
{
int n;
cout<<"\n Enter the Number of Students"; cin>>n;
student s[n]; ofstream outfile; outfile.open("student.dat",ios::out|ios::binary);
for(int i=0;i<n;i++)

{cout<<"\n Enter the Number of Students"; s[i].getdata();
outfile.write((char *)&s[i],sizeof(s[i]));
}
outfile.close();

int c; student d; do
{
cout<<"\n 1.Search";
cout<<"\n 2.Delete";
cout<<"\n 3.Display";
```

7

..

```
cout<<"\n 4.Exit";
cout<<"\n Enter Your Choice"; cin>>c;
switch(c)
{
case 1:d.search_(n);break; case 2:d.del(n);n=n-1;break;case 3:d.putdata(n);break;
case 4:break;
}
}
```

```
while(c!=4);

}
```

Output:

Enter the Number of Students2

Enter the Number of Students

Enter the Roll Number1

Enter the division C

Enter the Name Hrishi

Enter the Address Pune

Enter the Number of Students

8

..

Enter the Roll Number2

Enter the division C

Enter the Name XYZ

Enter the Address PUNE

1.Search

2.Delete

3.Display

4.Exit

Enter Your Choice1

Enter the Roll Number to be searched1

Found

Details:

Roll Number: 1

Division: C

Name: Hrishi

Address: Pune

1.Search

2.Delete

3.Display

4.Exit

Enter Your Choice2

Enter the roll number to be deleted 2

1.Search

2.Delete

3.Display

4.Exit

Enter Your Choice3

Roll Number: 1

Division: C

Name: Hrishi

Address: Pune

1.Search

2.Delete

3.Display

4.Exit

Conclusion:

Through this assignment ,we learned and performed how to store information in a sequential file and perform various operations on the file like search,delete etc.

..

Skill development

Assignment – 9

Aim:

Department maintains a employee information. The file contains employee ID, name, designation and salary . Allow user to add, delete information of employee. Display information of particular employee. If employee does not exist an appropriate message is displayed. If it is, then the system displays the employee details. Use index sequential file to main the data.

Objective:

To make use of index sequential files to maintain and operation on data.

Theory:

Index Sequential File:

This is basically a mixture of sequential and indexed file organisation techniques. Records are held in sequential order and can be accessed randomly through an index. Thus, these files share the merits of both systems enabling sequential or direct access to the data.

The index to these files operates by storing the highest record key in given cylinders and tracks. Note how this organisation gives the index a tree structure. Obviously this type of file organisation will require a direct access device, such as a hard disk.

Indexed sequential file organisation is very useful where records are often retrieved randomly and are also processed in (sequential) key order. Banks may use this organisation for their auto-bank machines i.e. customers randomly access their accounts throughout the day and at the end of the day the banks can update the whole file sequentially.

Advantages of Indexed Sequential Files:

1.Allows records to be accessed directly or sequentially.

2.Direct access ability provides vastly superior (average)

acces s times.

Disadvantages of Indexed Sequential Files:

1

..

1.The fact that several tables must be stored for the index makes for a considerable storage overhead.

2.As the items are stored in a sequential fashion this adds complexity to the

addition/deletion of records. Because frequent updating can be very inefficient, especially for large files, batch updates are often performed.

Code:

```
/*Rajas Mateti

   17u187

   222010

   B1*/
```

```
#include <iostream>

#include<fstream>

#include<string>

using namespace std;

typedef struct seq_file

{

   int id;

   char name[20],desg[20];

   long int sal;

}record;

typedef struct ind_file

{
```

```
    int id;

}index;

class file

{

    record data;

    index info;

public:

    void get_data()

    {

        cout<<"Enter id: ";

        cin>>data.id;

        cout<<"Enter name: ";

        cin>>data.name;
```

```
    cout<<"Enter designation: ";

    cin>>data.desg;

    cout<<"Enter salary: ";

    cin>>data.sal;

    info.id=data.id;

}

void add()

{

    fstream out1;

    fstream out2;

    out1.open("pos.txt",ios::app);

    out2.open("rec.txt",ios::app);

    get_data();
```

```
    out2.write((char*)&data,sizeof(data));

    out1.write((char*)&info,sizeof(info));

    out1.close();

    out2.close();

}

void search_rec(int id)

{

    int pos=0,loc=-1;

    fstream out1;

    fstream out2;

    out1.open("pos.txt");

    out2.open("rec.txt");

    loc=sizeof(info)*pos;
```

```
out2.seekg(loc,ios::beg);

for(pos=0;out2.read((char*)&info,sizeof(info));pos++)

{

   loc=sizeof(info)*pos;

   out2.seekg(loc,ios::beg);

   out2.read((char*)&info,sizeof(info));

   if(info.id==id)

   {

      break;

   }

}

if(loc==-1)

{
```

```
        cout<<"Record not found\n";

    }

    else

    {

        pos--;

        pos=sizeof(data)*pos;

        out1.seekg(pos,ios::beg);

        out1.read((char*)&data,sizeof(data));

        cout<<"Record found\n";

        cout<<data.id<<"\t"<<data.name<<"\t"<<data.desg<<"\t"<<data.sal<<endl;

    }

    out1.close();

    out2.close();
```

```
    }

};

int main()

{

    char r;

    do

    {

        char op;

        file f;

        do

        {

            int c;
```

```
cout<<"\n======================Menu======================\n";

        cout<<"1] Add record\n2] Search record\n3] Delete record\n";

cout<<"_____________________________________________________\n";

        cout<<"Enter your choice: ";

        cin>>c;

        switch(c)

        {

           case 1: {

                    f.add();

                }

                break;

           case 2: {
```

```
            int id;

            cout<<"Enter id to search: ";

            cin>>id;

            f.search_rec(id);

        }

        break;

    case 3: {



        }

        break;

    case 4: {



        }
```

```
                break;

            default:cout<<"Error 404.....page not found\n";

        }

        cout<<"Do you wish to continue(y/n): ";

        cin>>op;

    }while(op=='y' || op=='Y');

    cout<<"Test pass(y/n): ";

    cin>>r;

}while(r=='n' || r=='N');

cout<<"**********\n";

cout<<"*  Thank You!  *\n";

cout<<"**********\n";

return 0;
```

}

Output:

==========Menu==========

1] Add record

2] Search record

3] Delete record

Enter your choice: 1

Enter id: 11

Enter name: shivam

Enter designation: student

Enter salary: 60000

Do you wish to continue(y/n): y

==========Menu==========

1] Add record

2] Search record

3] Delete record

Enter your choice: 1

Enter id: 12

Enter name: rajas

Enter designation: manager

Enter salary: 235000

Do you wish to continue(y/n): y

==========Menu==========

1] Add record

2] Search record

3] Delete record

Enter your choice: 2

Enter id to search: 11

Record found

12 rajas manager 235000

Do you wish to continue(y/n): y

==========Menu==========

1] Add record

2] Search record

3] Delete record

_____

Enter your choice: 3

Do you wish to continue(y/n): y

==========Menu==========

1] Add record

2] Search record

3] Delete record

_____

Enter your choice: 3

Do you wish to continue(y/n): y

==========Menu==========

1] Add record

2] Search record

3] Delete record

Enter your choice:

Conclusion:

In above assignment, we made the use of index sequential files to operate on employee data.

2

7