Design and Analysis of Algorithms
Lab File
ITD03

Submitted by : Shiv Kumar
IT-2
5th Semester
2016UIT2563

Index

Practical 1
Mergesort

```
#include<bits/stdc++.h>

using namespace std;

void mergeSort(int input[], int size){
   // Write your code here
    if(size<=0 || size==1)
    {
        return;
    }

    int middle=size/2;
    mergeSort(input,middle);
    if(size%2==0)
    {
        mergeSort(input+middle,middle);
    }

    else mergeSort(input+middle,middle+1);

    int third[size];
    int i=0;
    int j=middle;
    int count=0;
    while(i<size/2 && j<size)
    {
        if(input[i]<input[j])
        {
            third[count]=input[i];
            i++;
        }

        else {
            third[count]=input[j];
            j++;
        }
```

```
        count++;

    }

    while(i<size/2)
    {
        third[count]=input[i];
        i++;
        count++;
    }

    while(j<size)
    {
        third[count]=input[j];
        j++;
        count++;
    }

    for(int m=0;m<size;m++)
    {
        input[m]=third[m];
    }
}

int main()
{
    int size;
    cout<<"Please enter the size of the array:";
    cin>>size;
    int * arr = new int[size];

    for(int i=0;i<size;i++)
    {
        cin>>arr[i];
    }

    mergeSort(arr,size);

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

```
        cout<<arr[i]<<" ";
    }

    cout<<endl;
}
```

```
[Shivs-MacBook-Air:designalgo championballer$ g++ merge_sort.cpp -o run
[Shivs-MacBook-Air:designalgo championballer$ ./run
Please enter the size of the array:5
5 3 4 2 1
1 2 3 4 5
```

Practical 2
Quicksort

```
#include<bits/stdc++.h>

using namespace std;

int partition(int arr[],int size)
{
    int x=arr[0];
    int countSmall=0;

    for(int i=1;i<size;i++)
    {
        if(arr[i]<arr[0])
        {
            countSmall++;
        }
    }

    arr[0]=arr[countSmall];
    arr[countSmall]=x;

    int z=0;
    int j=size-1;

    while(z<countSmall && j>countSmall)
    {
        if(arr[z]<x)
        {
            z++;
        }

        else if(arr[j]>=x)
        {
            j--;
        }

        else {
            int temp=arr[z];
```

```
            arr[z]=arr[j];
            arr[j]=temp;
            z++;
            j--;
        }

    }

    return countSmall;
}


void quickSort(int input[], int size) {
  /* Don't write main().
     Don't read input, it is passed as function argument.
     Change in the given array itself.
     Taking input and printing output is handled automatically.
  */
    if(size==0 || size==1)
    {
        return;
    }

    int pos=partition(input,size);

    quickSort(input,pos);
    quickSort(input+pos+1,size-pos-1);
}

int main()
{
    int size;
    cout<<"Please enter the size of the array:";
    cin>>size;
    int * arr = new int[size];

    for(int i=0;i<size;i++)
    {
        cin>>arr[i];
    }
```

```
    quickSort(arr,size);

    for(int i=0;i<size;i++)
    {
        cout<<arr[i]<<" ";
    }

    cout<<endl;
}
```

```
[Shivs-MacBook-Air:designalgo championballer$ g++ quick_sort.cpp -o run
[Shivs-MacBook-Air:designalgo championballer$ ./run
Please enter the size of the array:5
3 4 2 5 1
1 2 3 4 5
```

Practical 3
Maximum and Minimum using divide and conquer

```
#include<bits/stdc++.h>

using namespace std;

pair<int,int> max_min(int * arr, int n)
{
   pair<int,int> ans;
   if(n==0)
   {
         ans.first = INT_MIN;
         ans.second = INT_MAX;

         return ans;
   }

   if(n==1)
   {
         ans.first = arr[0];
         ans.second = arr[0];

         return ans;
   }

   int middle = n/2;

   pair<int,int> fsans = max_min(arr,middle);
   pair<int,int> ssans = max_min(arr+middle,n-middle);

   ans.first = max(fsans.first,ssans.first);
   ans.second = min(fsans.second,ssans.second);

   return ans;
}

int main()
{
   int n;
```

```
    cout<<"Enter size of array:";
    cin>>n;

    int * arr = new int[n];

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

    pair<int,int> mm = max_min(arr,n);

    cout<<"max:"<<mm.first<<" "<<"min:"<<mm.second<<endl;
}
```

```
[Shivs-MacBook-Air:designalgo championballer$ g++ max_min.cpp -o run
[Shivs-MacBook-Air:designalgo championballer$ ./run
Enter size of array:7
1 2 3 4 5 6 7
max:7 min:1
```

Practical 4
Binary Search

```
#include<bits/stdc++.h>

using namespace std;

int binarySearch(int input[], int size, int element) {
    // Write your code here
    if(size==0)
    {
        return -1;
    }
    int middle=0;
    if(size%2==0)
    {
        middle=size/2;
        middle-=1;
    }

    else{
      middle=size/2;
    }



    if(input[middle]==element){
        return middle;
    }

    else if(input[middle]>element)
    {
        int x=binarySearch(input,middle,element);
        return x;
    }

    else{

        int x=binarySearch(input+middle+1,size-middle-1,element);
        if(x==-1)
        {
```

```
            return x;
        }

        else return x+middle+1;
    }



}

int main()
{
    int size;
    cout<<"Please enter the size of the array:";
    cin>>size;
    int * arr = new int[size];

    for(int i=0;i<size;i++)
    {
        cin>>arr[i];
    }

    int element;
    cout<<"Please enter the element to find:";
    cin>>element;

    cout<<binarySearch(arr,size,element)<<endl;
}
```

```
[Shivs-MacBook-Air:designalgo championballer$ g++ binary_search.cpp -o run
[Shivs-MacBook-Air:designalgo championballer$ ./run
Please enter the size of the array:5
1 2 3 4 5
Please enter the element to find:2
1
```

Practical 5
Heap Sort

```
#include<bits/stdc++.h>

using namespace std;

void inplaceHeapSort(int input[], int n){

   for(int i=0;i<n;i++)
    {
      int ci=i;
      int pi=(ci-1)/2;
      while(ci!=0)
      {
        if(input[ci]<input[pi])
        {
          int temp=input[ci];
          input[ci]=input[pi];
          input[pi]=temp;
          ci=pi;
          pi=(ci-1)/2;
        }

        else break;
      }
    }

   for(int i=n-1;i>=0;i--)
   {
     int temp=input[0];
     input[0]=input[i];
     input[i]=temp;

     int pi=0;
     int ci1=(2*pi)+1;
     int ci2=(2*pi)+2;
     int min=0;
         while(ci1<i)
         {
```

```
        if(ci1<i && ci2>=i)
        {
                min=ci1;
        }

        else if(input[ci1]<input[ci2])
        {
                min=ci1;
        }

        else min=ci2;

        if(input[min]<input[pi])
        {
        int tempi=input[min];
        input[min]=input[pi];
        input[pi]=tempi;
        pi=min;
        ci1=(2*pi)+1;
        ci2=(2*pi)+2;
        }

        else break;
        }

    }

}

int main()
{
    int size;
    cout<<"Please enter the size of the array:";
    cin>>size;
    int * arr = new int[size];

    for(int i=0;i<size;i++)
    {
        cin>>arr[i];
    }
```

```
    inplaceHeapSort(arr,size);

    for(int i=0;i<size;i++)
    {
        cout<<arr[i]<<" ";
    }

    cout<<endl;
}
```

```
[Shivs-MacBook-Air:designalgo championballer$ g++ heap_sort.cpp -o run
[Shivs-MacBook-Air:designalgo championballer$ ./run
Please enter the size of the array:5
9 5 6 1 2
9 6 5 2 1
```

Pratical 6
Kruskal’s Algorithm

```
#include<bits/stdc++.h>

using namespace std;

class edge{
   public:
   int source;
   int destination;
   int weight;
};

bool compare(edge e1,edge e2)
{
  return e1.weight<e2.weight;
}

int findParent(int* parent,int i)
{
  if(parent[i]==i)
  {
    return i;
  }

  else return findParent(parent,parent[i]);
}
int main()
{
  int n,e;
  cout<<"Please enter the number of nodes and edges:";
  cin>>n>>e;

  edge* input=new edge[e];
  cout<<"Please enter source, destination and corresponding weights for each edge:";
  for(int i=0;i<e;i++)
  {

    int src, des, w;
```

```
  cin>>src>>des>>w;
  input[i].source=src;
  input[i].destination=des;
  input[i].weight=w;
}

sort(input,input+e,compare);
edge* output=new edge[n-1];
int* parent=new int[n];
for(int i=0;i<n;i++)
{
  parent[i]=i;
}

int count=0,current=0;
while(count<n-1)
{

  int sourceParent=findParent(parent,input[current].source);
  int destinationParent=findParent(parent,input[current].destination);

  if(sourceParent==destinationParent)
  {
    current++;
    continue;
  }

  else{
    output[count]=input[current];

        parent[findParent(parent,input[current].source)]=parent[findParent(parent,input[curr
        ent].destination)];
    count++;
    current++;
  }
}

for(int i=0;i<n-1;i++)
{
  if(output[i].destination<output[i].source)
```

```
    {
      cout<<output[i].destination<<" "<<output[i].source<<" "<<output[i].weight<<endl;
    }

    else cout<<output[i].source<<" "<<output[i].destination<<" "<<output[i].weight<<endl;
  }


}
```

```
[Shivs-MacBook-Air:designalgo championballer$ g++ kruskals.cpp -o run
[Shivs-MacBook-Air:designalgo championballer$ ./run
Please enter the number of nodes and edges:4 4
Please enter source, destination and corresponding weights for each edge:0 1 3
0 3 5
1 2 1
2 3 8
1 2 1
0 1 3
0 3 5
```

Practical 7
Prim's Algorithm

```
#include <bits/stdc++.h>
using namespace std;

class edge{
  public:
  int source;
  int destination;
  int weight;
};

int leastWeight(int* weight,bool* visited,int n)
{
  int i=0;
  int min=INT_MAX;
  int index=-1;
  while(i<n)
  {
    if(weight[i]<min && !visited[i])
    {
      min=weight[i];
      index=i;
    }
    i++;
  }

  return index;
}

void PrimAlgo(int** edges, int n, bool* visited,int* parent,int* weightArr)
{
  int outer=0;
  while(outer<n)
  {
   int nextVertex=leastWeight(weightArr,visited,n);
    if(nextVertex==-1)
    {
      return;
```

```
    }
   visited[nextVertex]=true;

   for(int i=0;i<n;i++)
   {
          if(i==nextVertex)
          {
          continue;
          }

           if(!visited[i] && edges[nextVertex][i]!=0)
           {
                    if(weightArr[i]>edges[nextVertex][i])
                    {
          parent[i]=nextVertex;
                    weightArr[i]=edges[nextVertex][i];
                    }
        }
    }
     outer++;
  }
}

edge* makeMST(int* parent, int* weight, int n)
{
  edge* output=new edge[n-1];
  for(int i=1;i<n;i++)
  {
    output[i-1].source=parent[i];
    output[i-1].destination=i;
    output[i-1].weight=weight[i];
  }

  return output;
}

int main()
{

  //taking Input;
```

```
int n,e;
cout<<"Please enter the number of nodes and edges:";
cin>>n>>e;

int** edges=new int*[n];
cout<<"Please enter source, destination and corresponding weights for each edge:";
for(int i=0;i<n;i++)
{
  edges[i]=new int[n];
  for(int j=0;j<n;j++)
  {
    edges[i][j]=0;
  }
}

for(int i=0;i<e;i++)
{
  int src, des, weight;
  cin>>src>>des>>weight;
  edges[src][des]=weight;
  edges[des][src]=weight;
}


bool* visited=new bool[n];
for(int i=0;i<n;i++)
{
  visited[i]=false;
}

int* parent=new int[n];
parent[0]=-1;
int* weightArr=new int[n];
weightArr[0]=0;
for(int i=1;i<n;i++)
{
  weightArr[i]=INT_MAX;
}

PrimAlgo(edges,n,visited,parent,weightArr);
```

```
  edge* output=makeMST(parent,weightArr,n);
  for(int i=0;i<n-1;i++)
  {
    if(output[i].source<output[i].destination)
    {
      cout<<output[i].source<<" "<<output[i].destination<<" "<<output[i].weight<<endl;
    }

    else cout<<output[i].destination<<" "<<output[i].source<<" "<<output[i].weight<<endl;
  }

  return 0;
}
```

```
[Shivs-MacBook-Air:designalgo championballer$ g++ prims.cpp -o run
[Shivs-MacBook-Air:designalgo championballer$ ./run
Please enter the number of nodes and edges:4 4
Please enter source, destination and corresponding weights for each edge:0 1 3
0 3 5
1 2 1
2 3 8
0 1 3
1 2 1
0 3 5
```

Pratical 8
Matrix chain multiplication

```
#include <bits/stdc++.h>
using namespace std;
void printParenthesis(int i, int j, int n, int *bracket, char &name)
{
   if (i == j)
   {
         cout << name++;
   return;
   }

   cout << "(";
   printParenthesis(i, *((bracket + i * n) + j), n, bracket, name);
   printParenthesis(*((bracket + i * n) + j) + 1, j, n, bracket, name);
   cout << ")";
}

void matrixChainOrder(int p[], int n)
{
   int m[n][n];
   int bracket[n][n];
   for (int i = 1; i < n; i++)
         m[i][i] = 0;
   for (int L = 2; L < n; L++)
   {
         for (int i = 1; i < n - L + 1; i++)
         {
                  int j = i + L - 1;
                  m[i][j] = INT_MAX;
                  for (int k = i; k <= j - 1; k++)
                  {
                           int q = m[i][k] + m[k + 1][j] + p[i - 1] * p[k] * p[j];
                           if (q < m[i][j])
                           {
                           m[i][j] = q;
                           bracket[i][j] = k;
                           }
                  }
```

```
        }
}

    char name = 'A';
    cout << "Optimal Parenthesization is : ";
    printParenthesis(1, n - 1, n, (int *)bracket, name);
    cout << "\nOptimal Cost is : " << m[1][n - 1];
}

int main()
{
    int n;
    cout << "Enter the size of the set of dimensions \n";
    cin >> n;
    int arr[n];
    cout << "Enter the dimensions one by one\n";
    for (int i = 0; i < n; i++)
        cin >> arr[i];
    matrixChainOrder(arr, n);
    cout<<endl;
    return 0;
}
```

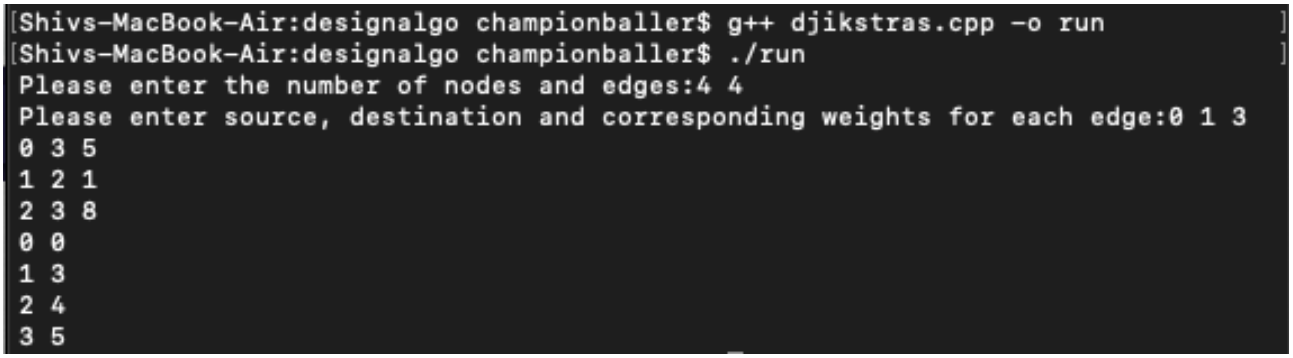
```
[Shivs-MacBook-Air:designalgo championballer$ g++ matrix_dp.cpp -o run
[Shivs-MacBook-Air:designalgo championballer$ ./run
Enter the size of the set of dimensions
5
Enter the dimensions one by one
3 4 5 6 7
Optimal Parenthesization is : (((AB)C)D)
Optimal Cost is : 276
```

Practical 9
Djikstra's Algorithm

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

int minDis(int* distance, bool* visited,int n)
{
  int i=0;
  int index=-1;
  int min=INT_MAX;
  while(i<n)
  {
    if(!visited[i] && distance[i]<min)
    {
      min=distance[i];
      index=i;
    }
    i++;
  }
  return index;
}

void djikstra(int** edges,int n,bool* visited,int* distance)
{
  int outer=0;
  while(outer<n){
    int nextVertex=minDis(distance,visited,n);
    if(nextVertex==-1)
    {
      return;
    }
    visited[nextVertex]=true;
   for(int i=0;i<n;i++)
    {
      if(i==nextVertex)continue;
      if(!visited[i] && edges[nextVertex][i]!=0)
      {
        int cdis=distance[nextVertex]+edges[nextVertex][i];
```

```
        if(cdis<distance[i])
        {
          distance[i]=cdis;
        }
       }
     }
     outer++;
   }
}

int main()
{
  int n,e;
  cout<<"Please enter the number of nodes and edges:";
  cin>>n>>e;

  int** edges=new int*[n];
  cout<<"Please enter source, destination and corresponding weights for each edge:";
  for(int i=0;i<n;i++)
  {
    edges[i]=new int[n];
    for(int j=0;j<n;j++)
    {
      edges[i][j]=0;
    }
  }

  for(int i=0;i<e;i++)
  {
    int src,des,weight;
    cin>>src>>des>>weight;
    edges[src][des]=weight;
    edges[des][src]=weight;
  }

  bool* visited=new bool[n];
  for(int i=0;i<n;i++)visited[n]=false;

  int* distance=new int[n];
  distance[0]=0;
```

```
  for(int i=1;i<n;i++)
  {
    distance[i]=INT_MAX;
  }

  djikstra(edges,n,visited,distance);
  for(int i=0;i<n;i++)
  {
    cout<<i<<" "<<distance[i]<<endl;
  }
  return 0;
}
```



Practical 10
Bellman-Ford Algorithm

```
#include <bits/stdc++.h>

using namespace std;

struct Edge
{
    int src, dest, weight;
};

struct Graph
{
    int V, E;
    struct Edge *edge;
};

struct Graph *createGraph(int V, int E)
{
    struct Graph *graph = new Graph;
    graph->V = V;
    graph->E = E;
    graph->edge = new Edge[E];
    return graph;
}

void printArr(int dist[], int n)
{
    printf("Vertex Distance from Source\n");
    for (int i = 0; i < n; ++i)
        printf("%d \t\t %d\n", i, dist[i]);
}

void BellmanFord(struct Graph *graph, int src)
{
    int V = graph->V;
    int E = graph->E;
    int dist[V];
    for (int i = 0; i < V; i++)
```

```
        dist[i] = INT_MAX;

    dist[src] = 0;
    for (int i = 1; i <= V - 1; i++)
    {
        for (int j = 0; j < E; j++)
        {
            int u = graph->edge[j].src;
            int v = graph->edge[j].dest;
            int weight = graph->edge[j].weight;
            if (dist[u] != INT_MAX && dist[u] + weight < dist[v])
            dist[v] = dist[u] + weight;
        }
    }

    for (int i = 0; i < E; i++)
    {
        int u = graph->edge[i].src;
        int v = graph->edge[i].dest;
        int weight = graph->edge[i].weight;
        if (dist[u] != INT_MAX && dist[u] + weight < dist[v])
            printf("Graph contains negative weight cycle");
    }

    printArr(dist, V);
    return;
}

int main()
{
    int V;
    int E;
    cout << "enter the number of vetices: \n";
    cin >> V;
    cout << "enter the number of edges: \n";
    cin >> E;

    struct Graph *graph = createGraph(V, E);
    cout << "Enter the weight source and destination for each node \n";
    for (int i = 0; i < E; i++)
```

```
    {
        cin >> graph->edge[i].weight >> graph->edge[i].src >> graph->edge[i].dest; //
          >>temp.weight;
    }

    BellmanFord(graph, 0);
    return 0;
}
```

```
[Shivs-MacBook-Air:designalgo championballer$ g++ bellman_ford.cpp -o run
[Shivs-MacBook-Air:designalgo championballer$ ./run
enter the number of vetices:
3
enter the number of edges:
3
Enter the weight source and destination for each node
3 0 1
3 0 2
4 1 2
Vertex Distance from Source
0               0
1               3
2               3
```

Practical 11
Longest Common Substring

```
#include<bits/stdc++.h>
#include<cstring>
using namespace std;

int lcs(string s1, string s2){

    vector<vector<int> > dp;
   for(int i=0;i<=s1.length();i++)
   {
     vector<int> small(s2.length()+1);
     dp.push_back(small);
   }
   dp[0][0] = 0;
   for(int i=1;i<=s2.length();i++)
   {
         dp[0][i] = 0;
   }

   for(int j=1;j<=s1.length();j++)
   {
         dp[j][0] = 0;
   }

   for(int i=1;i<=s1.length();i++)
   {
     for(int j=1;j<=s2.length();j++)
     {
       if(s1[s1.length()-i]==s2[s2.length()-j])
       {
         dp[i][j] = dp[i-1][j-1]+1;
       }

       else dp[i][j] = max(dp[i-1][j],dp[i][j-1]);
     }
   }

   return dp[s1.length()][s2.length()];
```

```
}

int main()
{
  string s1,s2;

  cout<<"Please enter strings to compare:";
  cin>>s1>>s2;

  cout<<lcs(s1,s2)<<endl;
}
```

```
[Shivs-MacBook-Air:designalgo championballer$ g++ lcs.cpp -o run
[Shivs-MacBook-Air:designalgo championballer$ ./run
Please enter strings to compare:adebc dcadb
3
```

Practical 12
Matrix Chain Multiplication using dynamic programming

```
#include <bits/stdc++.h>
using namespace std;

void printParenthesis(int i, int j, int n,int *bracket, char &name)
{
   if (i == j)
   {
   cout << name++;
   return;
   }
   cout << "(";
   printParenthesis(i, *((bracket + i * n) + j), n,
   bracket, name);
   printParenthesis(*((bracket + i * n) + j) + 1, j,
   n, bracket, name);
   cout << ")";
}

void matrixChainOrder(int p[], int n)
{
   int m[n][n];
   int bracket[n][n];
   for (int i = 1; i < n; i++)
   m[i][i] = 0;
   for (int L = 2; L < n; L++)
   {
         for (int i = 1; i < n - L + 1; i++)
         {
         int j = i + L - 1;
         m[i][j] = INT_MAX;
                  for (int k = i; k <= j - 1; k++)
                  {
                           int q = m[i][k] + m[k + 1][j] + p[i - 1] * p[k] * p[j];
                           if (q < m[i][j])
                           {
                                    m[i][j] = q;
                                    bracket[i][j] = k;
```

```
                    }
                }
            }
        }
    char name = 'A';
    cout << "Optimal Parenthesization is : ";
    printParenthesis(1, n - 1, n, (int *)bracket, name);
    cout << "\nOptimal Cost is : " << m[1][n - 1];
}

int main()
{
    int n;
    cout << "Enter the size of the set of dimensions \n";
    cin >> n;
    int arr[n];
    cout << "Enter the dimensions one by one\n";
    for (int i = 0; i < n; i++)
    cin >> arr[i];
    matrixChainOrder(arr, n);
    cout<<endl;
    return 0;
}
```

```
[Shivs-MacBook-Air:designalgo championballer$ g++ matrix_dp.cpp -o run
[Shivs-MacBook-Air:designalgo championballer$ ./run
Enter the size of the set of dimensions
5
Enter the dimensions one by one
3 4 5 6 7
Optimal Parenthesization is : (((AB)C)D)
Optimal Cost is : 276
```

Practical 13
Minimum spanning tree

```
#include <bits/stdc++.h>
using namespace std;

class edge{
  public:
  int source;
  int destination;
  int weight;
};

int leastWeight(int* weight,bool* visited,int n)
{
  int i=0;
  int min=INT_MAX;
  int index=-1;
  while(i<n)
  {
    if(weight[i]<min && !visited[i])
    {
      min=weight[i];
      index=i;
    }
    i++;
  }

  return index;
}

void PrimAlgo(int** edges, int n, bool* visited,int* parent,int* weightArr)
{
  int outer=0;
  while(outer<n)
  {
   int nextVertex=leastWeight(weightArr,visited,n);
    if(nextVertex==-1)
    {
      return;
```

```
    }
   visited[nextVertex]=true;

   for(int i=0;i<n;i++)
   {
          if(i==nextVertex)
          {
          continue;
          }

           if(!visited[i] && edges[nextVertex][i]!=0)
           {
                    if(weightArr[i]>edges[nextVertex][i])
                    {
          parent[i]=nextVertex;
                    weightArr[i]=edges[nextVertex][i];
                    }
        }
    }
     outer++;
  }
}

edge* makeMST(int* parent, int* weight, int n)
{
  edge* output=new edge[n-1];
  for(int i=1;i<n;i++)
  {
    output[i-1].source=parent[i];
    output[i-1].destination=i;
    output[i-1].weight=weight[i];
  }

  return output;
}

int main()
{

  //taking Input;
```

```
int n,e;
cout<<"Please enter the number of nodes and edges:";
cin>>n>>e;

int** edges=new int*[n];
cout<<"Please enter source, destination and corresponding weights for each edge:";
for(int i=0;i<n;i++)
{
  edges[i]=new int[n];
  for(int j=0;j<n;j++)
  {
    edges[i][j]=0;
  }
}

for(int i=0;i<e;i++)
{
  int src, des, weight;
  cin>>src>>des>>weight;
  edges[src][des]=weight;
  edges[des][src]=weight;
}


bool* visited=new bool[n];
for(int i=0;i<n;i++)
{
  visited[i]=false;
}

int* parent=new int[n];
parent[0]=-1;
int* weightArr=new int[n];
weightArr[0]=0;
for(int i=1;i<n;i++)
{
  weightArr[i]=INT_MAX;
}

PrimAlgo(edges,n,visited,parent,weightArr);
```

```
  edge* output=makeMST(parent,weightArr,n);
  for(int i=0;i<n-1;i++)
  {
    if(output[i].source<output[i].destination)
    {
      cout<<output[i].source<<" "<<output[i].destination<<" "<<output[i].weight<<endl;
    }

    else cout<<output[i].destination<<" "<<output[i].source<<" "<<output[i].weight<<endl;
  }

  return 0;
}
```

```
[Shivs-MacBook-Air:designalgo championballer$ g++ prims.cpp -o run
[Shivs-MacBook-Air:designalgo championballer$ ./run
Please enter the number of nodes and edges:4 4
Please enter source, destination and corresponding weights for each edge:0 1 3
0 3 5
1 2 1
2 3 8
0 1 3
1 2 1
0 3 5
```

Practical 14
Depth First Search

```
#include <bits/stdc++.h>
using namespace std;
void dfs(vector<vector<int> > &a, int s, vector<bool> &vis)
{
   vis[s] = true;
   cout << s + 1 << " ";
   for (int i = 0; i < a[s].size(); i++)
   if (!vis[a[s][i]])
   dfs(a, a[s][i], vis);
}

int main()
{
   int n, e;
   cout<<"Please enter the number of nodes and edges:";
   cin >> n >>e;

   cout<<"Please enter source, destination for each edge:";
   vector<vector<int> > a(n, vector<int>(1, 0));
   for (int i = 0; i < e; i++)
   {
         int x, y;

         cin >> x >> y;
         a[x - 1].push_back(y - 1);
         a[y - 1].push_back(x - 1);
   }
   int s;
   cout << "Enter the source node: ";
   cin >> s;
   vector<bool> vis(n, false);
   dfs(a, s - 1, vis);
   cout << endl;
   return 0;
}
```

```
[Shivs-MacBook-Air:designalgo championballer$ g++ dfs.cpp -o run
[Shivs-MacBook-Air:designalgo championballer$ ./run
Please enter the number of nodes and edges:4 4
Please enter source, destination for each edge:2 3
3 1
4 2
3 4
Enter the source node: 1
1 3 2 4
```

Practical 15
Breadth First Search

```
#include <bits/stdc++.h>
using namespace std;

void bfs(vector<vector<int> > &a, int s)
{
   int n = a.size();
   vector<bool> vis(n, false);
   queue<int> q;
   q.push(s);
   vis[s] = true;
   while (!q.empty())
   {
         int x = q.front();
         q.pop();
         for (int i = 0; i < a[x].size(); i++)
         {
                  if (!vis[a[x][i]])
                  {
                           q.push(a[x][i]);
                           vis[a[x][i]] = true;
                  }
         }
         cout << x + 1 << " ";
   }
}

int main()
{
   int n, e;
   cout<<"Please enter the number of nodes and edges:";
   cin >> n>>e;

   cout<<"Please enter source, destination for each edge:";
   vector<vector<int> > a(n, vector<int>(1, 0));
   for (int i = 0; i < e; i++)
   {
         int x, y;
```

```
        cin >> x >> y;
        a[x - 1].push_back(y - 1);
        a[y - 1].push_back(x - 1);
    }
    int s;
    cout << "Enter the source node: ";
    cin >> s;
    bfs(a, s - 1);
    cout << endl;
    return 0;
}
```

```
[Shivs-MacBook-Air:designalgo championballer$ g++ bfs.cpp -o run
[Shivs-MacBook-Air:designalgo championballer$ ./run
Please enter the number of nodes and edges:4 4
Please enter source, destination for each edge:1 2
2 3
3 4
4 1
Enter the source node: 2
2 1 3 4
```