

## Practical-10: Implementation of Edmonds-Karp algorithm

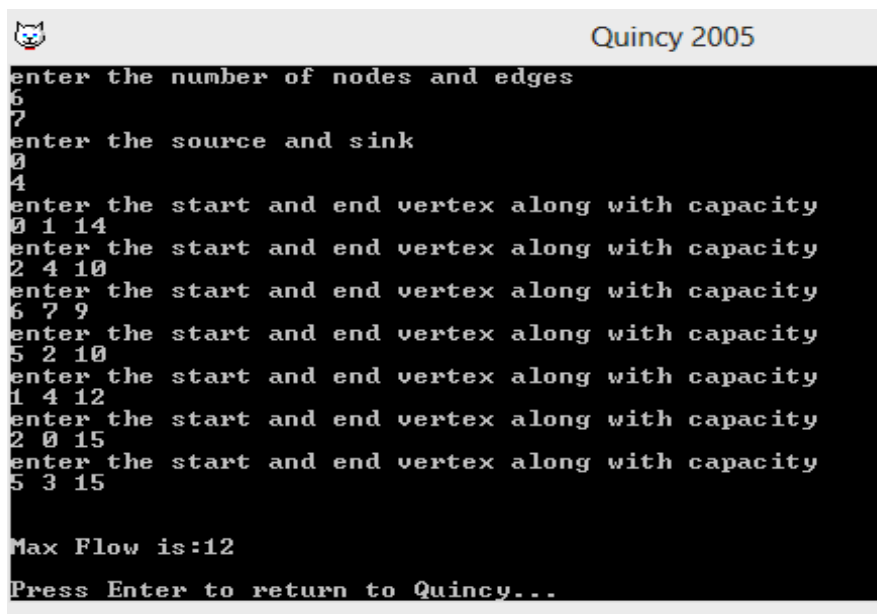
```
#include<cstdio>
#include<queue>
#include<cstring>
#include<vector>
#include<iostream>
using namespace std;
int c[10][10]; int flowPassed[10][10]; vector<int> g[10]; int parList[10]; int currentPathC[10];
int bfs(int sNode, int eNode)
{
    memset(parList, -1, sizeof(parList));
    memset(currentPathC, 0, sizeof(currentPathC));
    queue<int> q;
    q.push(sNode);
    parList[sNode] = -1;
    currentPathC[sNode] = 999;
    while(!q.empty())
    {
        int currNode = q.front();
        q.pop();
        for(int i=0; i<g[currNode].size(); i++)
        {
            int to = g[currNode][i];
            if(parList[to] == -1)
            {
                if(c[currNode][to] - flowPassed[currNode][to] > 0)
                {
                    parList[to] = currNode;
                    currentPathC[to] = min(currentPathC[currNode],
                    c[currNode][to] - flowPassed[currNode][to]);
                    if(to == eNode)
                    {
                        return currentPathC[eNode];
                    }
                }
                q.push(to);
            }
        }
    }
}
return 0;
}
int edmondsKarp(int sNode, int eNode)
{
    int maxFlow = 0;
    while(true)
    {
        int flow = bfs(sNode, eNode);
        if (flow == 0)
        {
            break;
        }
    }
}
```

```

    maxFlow += flow;
    int currNode = eNode;
    while(currNode != sNode)
    {
        int prevNode = parList[currNode];
        flowPassed[prevNode][currNode] += flow;
        flowPassed[currNode][prevNode] -= flow;
        currNode = prevNode;
    }
}
return maxFlow;
}
int main()
{
    int nodCount, edCount;
    cout<<"enter the number of nodes and edges\n";
    cin>>nodCount>>edCount;
    int source, sink;
    cout<<"enter the source and sink\n";
    cin>>source>>sink;
    for(int ed = 0; ed < edCount; ed++)
    {
        cout<<"enter the start and end vertex along with capacity\n";
        int from, to, cap;
        cin>>from>>to>>cap;
        c[from][to] = cap;
        g[from].push_back(to);
        g[to].push_back(from);
    }
    int maxFlow = edmondsKarp(source, sink);
    cout<<endl<<endl<<"Max Flow is:"<<maxFlow<<endl;
}

```

## Output:



```

Quincy 2005
enter the number of nodes and edges
6
7
enter the source and sink
0
4
enter the start and end vertex along with capacity
0 1 14
enter the start and end vertex along with capacity
2 4 10
enter the start and end vertex along with capacity
6 7 9
enter the start and end vertex along with capacity
5 2 10
enter the start and end vertex along with capacity
1 4 12
enter the start and end vertex along with capacity
2 0 15
enter the start and end vertex along with capacity
5 3 15

Max Flow is:12
Press Enter to return to Quincy...

```