CODE FOR FORD FULKERSON ALGORITHM:

#include <iostream>

#include <climits>

#include <cstring>

#include <queue>

#define n 7

using namespace std;

bool bfs(int g[n][n], int s, int t, int par[])

{

   bool visit[n];

   memset(visit, 0, sizeof(visit));

   queue <int> q;

   q.push(s);

   visit[s] = true;

   par[s] = -1;

   while (!q.empty())

   {

      int u = q.front();

      q.pop();

      for (int v=0; v<n; v++)

      {

         if (visit[v]==false && g[u][v] > 0)

         {

            q.push(v);

            par[v] = u;

            visit[v] = true;

         }

      }

   }

   return (visit[t] == true);

}

int fordFulkerson(int G[n][n], int s, int t)

{

   int u, v;

   int g[n][n];

   for (u = 0; u < n; u++)

   {

      for (v = 0; v < n; v++)

      g[u][v] = G[u][v];

   }

   int par[n];

   int max\_flow = 0;

   while (bfs(g, s, t,par))

   {

      int path\_flow = INT\_MAX;

      for (v=t; v!=s; v=par[v])

      {

         u = par[v];

         path\_flow = min(path\_flow, g[u][v]);

      }

      for (v = t; v != s; v = par[v])

      {

         u = par[v];

         g[u][v] -= path\_flow;

         g[v][u] += path\_flow;

      }

      max\_flow += path\_flow;

   }

   return max\_flow;

}

int main()

{

   int g[n][n] = {{0, 6, 7, 1},

      {0, 0, 4, 2},

      {0, 5, 0, 0},

      {0, 0, 19, 12},

      {0, 0, 0, 17},

      {0, 0, 0, 0}};

   cout << "The maximum possible flow is " << fordFulkerson(g, 0, 3);

   return 0;

}

OUTPUT:

The maximum possible flow is 3

TIME COMPLEXITY:

In the first step that is finding an augmented path the time complexity is O(E).

Where E = no.of edges . when ever we have multiple no.of edges then finding an augmenting path suitable for that particular instants of that network will also increase, that will depend upon no.of edges.

To calculate an augmenting path, bottleneck capacity, each edge and total flow the complexity will be in order of O(F). Where F = Maximum flow

The total complexity of this algorithm will be O(E\*F).