 max flow dinic

struct Edge

{

    int v ;

    int flow ; // flow of data in edge

    int C;

    int rev ;

};

// Residual Graph

class Graph

{

    int V; // number of vertex

    int \*level ; // stores level of a node

    vector< Edge > \*adj;

public :

    Graph(int V)

    {

        adj = new vector<Edge>[V];

        this->V = V;

        level = new int[V];

    }

    // add edge to the graph

    void addEdge(int u, int v, int C)

    {

        Edge a{v, 0, C, adj[v].size()};

        Edge b{u, 0, 0, adj[u].size()};

        adj[u].push\_back(a);

        adj[v].push\_back(b); // reverse edge

    }

    bool BFS(int s, int t);

    int sendFlow(int s, int flow, int t, int ptr[]);

    int DinicMaxflow(int s, int t);

};

bool Graph::BFS(int s, int t)

{

    for (int i = 0 ; i < V ; i++)

        level[i] = -1;

    level[s] = 0;  // Level of source vertex

    list< int > q;

    q.push\_back(s);

    vector<Edge>::iterator i ;

    while (!q.empty())

    {

        int u = q.front();

        q.pop\_front();

        for (i = adj[u].begin(); i != adj[u].end(); i++)

        {

            Edge &e = \*i;

            if (level[e.v] < 0  && e.flow < e.C)

            {

                level[e.v] = level[u] + 1;

                q.push\_back(e.v);

            }

        }

    }

    return level[t] < 0 ? false : true ;

}

int Graph::sendFlow(int u, int flow, int t, int start[])

{

    if (u == t)

        return flow;

    for (  ; start[u] < adj[u].size(); start[u]++)

    {

        Edge &e = adj[u][start[u]];

        if (level[e.v] == level[u]+1 && e.flow < e.C)

        {

            int curr\_flow = min(flow, e.C - e.flow);

            int temp\_flow = sendFlow(e.v, curr\_flow, t, start);

            if (temp\_flow > 0)

            {

                e.flow += temp\_flow;

                adj[e.v][e.rev].flow -= temp\_flow;

                return temp\_flow;

            }

        }

    }

    return 0;

}

int Graph::DinicMaxflow(int s, int t)

{

    if (s == t)

        return -1;

    int total = 0;  // Initialize result

    while (BFS(s, t) == true)

    {

        int \*start = new int[V+1];

        while (int flow = sendFlow(s, INT\_MAX, t, start))

            total += flow;

    }

    return total;

}