```cpp
/*
    | Dinitz's Max Flow algorithm |
    Desc: Calculating Max flow in O(V^2*E).
    Source: KawakiMeido
    State: Untested lmao (But that one problem in ECNA practice works sooooo
*/

struct Node{
    int u,v,flow,cap;
    Node(int _u, int _v, int _cap): u(_u), v(_v), cap(_cap){
        flow = 0;
    }
};

int s,t,edgecnt;
vector<Node> edge;
vector<int> adj[N];
int level[N],ptr[N];

void AddEdge(int u, int v, int cap){
    edge.emplace_back(u,v,cap);
    edge.emplace_back(v,u,0);
    adj[u].push_back(edgecnt);
    adj[v].push_back(edgecnt+1);
    edgecnt+=2;
}

bool BFS(){
    queue<int> q;
    memset(level,-1,sizeof(level));
    level[s] = 0;
    q.push(0);

    while (!q.empty()){
        int u = q.front();
        q.pop();

        for (auto id:adj[u]){
            int v = edge[id].v;
            if (level[v] == -1 && edge[id].flow!=edge[id].cap){
                level[v] = level[u]+1;
                q.push(v);
            }
        }
    }

    return (level[t]!=-1);
}

```

```cpp
int DFS(int u, int pushed){
    if (pushed == 0) return 0;
    if (u==t) return pushed;

    int res = 0;

    for (int &pos = ptr[u]; pos<(int)adj[u].size(); pos++){
        int id = adj[u][pos];
        int v = edge[id].v;
        if (level[v] == level[u]+1 && edge[id].flow!=edge[id].cap){
            if ((res = DFS(v,min(pushed,edge[id].cap-edge[id].flow)))){
                edge[id].flow+=res;
                edge[id^1].flow-=res;
                return res;
            }
        }
    }

    return 0;
}

int Dinitz(){
    int max_flow = 0;
    while (BFS()){
        memset(ptr,0,sizeof(ptr));
        int flow;
        while ((flow = DFS(s,INF))){
            max_flow+=flow;
        }
    }
    return max_flow;
}
```