네트워크 플로우 소스코드

최백준 choi@startlink.io

C++14

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <cstring>
#include <string>
#include <map>
#include <queue>
using namespace std;
struct MaximumFlow {
    struct Edge {
        int to;
        int capacity;
        Edge *rev;
        Edge(int to, int capacity) : to(to), capacity(capacity) {
        }
    };
    int n;
    int source, sink;
    vector<vector<Edge *>> graph;
    vector<bool> check;
    MaximumFlow(int n, int source, int sink) : n(n), source(source), sink(sink) {
        graph.resize(n);
        check.resize(n);
    };
    void add_edge(int u, int v, int cap) {
        Edge *ori = new Edge(v,cap);
        Edge *rev = new Edge(u,0);
        ori->rev = rev;
        rev->rev = ori;
        graph[u].push_back(ori);
        graph[v].push_back(rev);
    }
    int dfs(int x, int c) {
        if (check[x]) return 0;
        check[x] = true;
        if (x == sink) {
            return c;
        }
        for (int i=0; i<graph[x].size(); i++) {
            if (graph[x][i]->capacity > 0) {
                int nc = graph[x][i]->capacity;
                if (c != 0 && c < nc) {
                    nc = c;
                }
                int f = dfs(graph[x][i]->to, nc);
                if (f) {
                    graph[x][i]->capacity -= f;
                    graph[x][i]->rev->capacity += f;
                    return f;
                }
            }
        }
        return 0;
    }
    int flow() {
        int ans = 0;
        while (true) {
            fill(check.begin(),check.end(),false);
            int f = dfs(source, 0);
            if (f == 0) break;
            ans += f;
        }
        return ans;
    }
};
int node(string s) {
    if (s[0] >= 'A' && s[0] <= 'Z') {
        return s[0] - 'A';
    } else {
        return s[0] - 'a' + 26;
    }
}
int main() {
    int m;
    cin >> m;
    MaximumFlow mf(52, 0, 'Z'-'A');
    for (int i=0; i<m; i++) {
        string us, vs;
        int f;
        cin >> us >> vs >> f;
        int u = node(us);
        int v = node(vs);
        mf.add_edge(u,v,f);
        mf.add_edge(v,u,f);
    }
    cout << mf.flow() << '\n';
    return 0;
}
```

결과	메모리	시간	코드 길이
맞았습니다!!	1996 KB	0 ms	2145 B

C++14

Edmond-karp

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <cstring>
#include <string>
#include <map>
#include <queue>
using namespace std;
struct MaximumFlow {
    struct Edge {
        int to;
        int capacity;
        Edge *rev;
        Edge(int to, int capacity) : to(to), capacity(capacity) {
        }
    };
    int n;
    int source, sink;
    vector<vector<Edge *>> graph;
    MaximumFlow(int n, int source, int sink) : n(n), source(source), sink(sink) {
        graph.resize(n);
    };
    void add_edge(int u, int v, int cap) {
        Edge *ori = new Edge(v,cap);
        Edge *rev = new Edge(u,0);
        ori->rev = rev;
        rev->rev = ori;
        graph[u].push_back(ori);
        graph[v].push_back(rev);
    }
    int bfs() {
        vector<bool> check(n, false);
        vector<pair<int,int>> from(n, make_pair(-1,-1));
        queue<int> q;
        q.push(source);
        check[source] = true;
        while (!q.empty()) {
            int x = q.front();
            q.pop();
            for (int i=0; i<graph[x].size(); i++) {
                if (graph[x][i]->capacity > 0 && !check[graph[x][i]->to]) {
                    q.push(graph[x][i]->to);
                    check[graph[x][i]->to] = true;
                    from[graph[x][i]->to] = make_pair(x,i);
                }
            }
        }
        if (!check[sink]) {
            return 0;
        }
        int x = sink;
        int c = graph[from[x].first][from[x].second]->capacity;
        while (from[x].first != -1) {
            if (c > graph[from[x].first][from[x].second]->capacity) {
                c = graph[from[x].first][from[x].second]->capacity;
            }
            x = from[x].first;
        }
        x = sink;
        while (from[x].first != -1) {
            Edge *e = graph[from[x].first][from[x].second];
            e->capacity -= c;
            e->rev->capacity += c;
            x = from[x].first;
        }
        return c;
    }
    int flow() {
        int ans = 0;
        while (true) {
            int f = bfs();
            if (f == 0) break;
            ans += f;
        }
        return ans;
    }
};
int node(string s) {
    if (s[0] >= 'A' && s[0] <= 'Z') {
        return s[0] - 'A';
    } else {
        return s[0] - 'a' + 26;
    }
}
int main() {
    int m;
    cin >> m;
    MaximumFlow mf(52, 0, 'Z'-'A');
    for (int i=0; i<m; i++) {
        string us, vs;
        int f;
        cin >> us >> vs >> f;
        int u = node(us);
        int v = node(vs);
        mf.add_edge(u,v,f);
        mf.add_edge(v,u,f);
    }
    cout << mf.flow() << '\n';
    return 0;
}
```

결과	메모리	시간	코드 길이
맞았습니다!!	1992 KB	0 ms	2612 B

C++14

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <cstring>
#include <string>
#include <map>
#include <queue>
using namespace std;
struct MaximumFlow {
    struct Edge {
        int to;
        int capacity;
        Edge *rev;
        Edge(int to, int capacity) : to(to), capacity(capacity) {
        }
    };
    int n;
    int source, sink;
    vector<vector<Edge *>> graph;
    vector<bool> check;
    MaximumFlow(int n, int source, int sink) : n(n), source(source), sink(sink) {
        graph.resize(n);
        check.resize(n);
    };
    void add_edge(int u, int v, int cap) {
        Edge *ori = new Edge(v,cap);
        Edge *rev = new Edge(u,0);
        ori->rev = rev;
        rev->rev = ori;
        graph[u].push_back(ori);
        graph[v].push_back(rev);
    }
    void add_edge_from_source(int v, int cap) {
        add_edge(source, v, cap);
    }
    void add_edge_to_sink(int u, int cap) {
        add_edge(u, sink, cap);
    }
    int dfs(int x, int c) {
        if (check[x]) return 0;
        check[x] = true;
        if (x == sink) {
            return c;
        }
        for (int i=0; i<graph[x].size(); i++) {
            if (graph[x][i]->capacity > 0) {
                int nc = graph[x][i]->capacity;
                if (c != 0 && c < nc) {
                    nc = c;
                }
                int f = dfs(graph[x][i]->to, nc);
                if (f) {
                    graph[x][i]->capacity -= f;
                    graph[x][i]->rev->capacity += f;
                    return f;
                }
            }
        }
        return 0;
    }
    int flow() {
        int ans = 0;
        while (true) {
            fill(check.begin(),check.end(),false);
            int f = dfs(source, 0);
            if (f == 0) break;
            ans += f;
        }
        return ans;
    }
};

int main() {
    int n,m;
    scanf("%d %d",&n,&m);
    MaximumFlow mf(n+m+2,0,n+m+1);
    for (int i=1; i<=n; i++) {
        int cnt;
        scanf("%d",&cnt);
        for (int j=0; j<cnt; j++) {
            int job;
            scanf("%d",&job);
            mf.add_edge(i,n+job,1);
        }
    }
    for (int i=1; i<=n; i++) {
        mf.add_edge_from_source(i,1);
    }
    for (int i=1; i<=m; i++) {
        mf.add_edge_to_sink(n+i,1);
    }
    printf("%d\n",mf.flow());
    return 0;
}
```

결과	메모리	시간	코드 길이
시간 초과	80880 KB	3792 ms	2338 B

C++14

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <cstring>
#include <string>
#include <map>
#include <queue>
using namespace std;
struct MaximumFlow {
    int n;
    int source, sink;
    vector<vector<int>> graph;
    vector<bool> check;
    vector<int> pred;
    MaximumFlow(int n) : n(n) {
        graph.resize(n);
        check.resize(n);
        pred.resize(n,-1);
    };
    void add_edge(int u, int v) {
        graph[u].push_back(v);
    }
    bool dfs(int x) {
        if (x == -1) return true;
        for (int next : graph[x]) {
            if (check[next]) continue;
            check[next] = true;
            if (dfs(pred[next])) {
                pred[next] = x;
                return true;
            }
        }
        return false;
    }
    int flow() {
        int ans = 0;
        for (int i=0; i<n; i++) {
            fill(check.begin(),check.end(),false);
            if (dfs(i)) {
                ans += 1;
            }
        }
        return ans;
    }
};

int main() {
    int n,m;
    scanf("%d %d",&n,&m);
    MaximumFlow mf(max(n,m));
    for (int i=0; i<n; i++) {
        int cnt;
        scanf("%d",&cnt);
        for (int j=0; j<cnt; j++) {
            int job;
            scanf("%d",&job);
            mf.add_edge(i,job-1);
        }
    }
    printf("%d\n",mf.flow());
    return 0;
}
```

결과	메모리	시간	코드 길이
맞았습니다!!	5948 KB	344 ms	1344 B

C++14

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <cstring>
#include <string>
#include <map>
#include <queue>
using namespace std;
struct MaximumFlow {
    int n;
    int source, sink;
    vector<vector<int>> graph;
    vector<bool> check;
    vector<int> pred;
    MaximumFlow(int n) : n(n) {
        graph.resize(n);
        check.resize(n);
        pred.resize(n,-1);
    };
    void add_edge(int u, int v) {
        graph[u].push_back(v);
    }
    bool dfs(int x) {
        if (x == -1) return true;
        for (int next : graph[x]) {
            if (check[next]) continue;
            check[next] = true;
            if (dfs(pred[next])) {
                pred[next] = x;
                return true;
            }
        }
        return false;
    }
    int flow() {
        int ans = 0;
        for (int i=0; i<n; i++) {
            for (int j=0; j<2; j++) {
                fill(check.begin(),check.end(),false);
                if (dfs(i)) {
                    ans += 1;
                }
            }
        }
        return ans;
    }
};

int main() {
    int n,m;
    scanf("%d %d",&n,&m);
    MaximumFlow mf(max(n,m));
    for (int i=0; i<n; i++) {
        int cnt;
        scanf("%d",&cnt);
        for (int j=0; j<cnt; j++) {
            int job;
            scanf("%d",&job);
            mf.add_edge(i,job-1);
        }
    }
    printf("%d\n",mf.flow());
    return 0;
}
```

결과	메모리	시간	코드 길이
맞았습니다!!	5948 KB	1636 ms	1412 B

C++14

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <cstring>
#include <string>
#include <map>
#include <queue>
using namespace std;
struct MaximumFlow {
    struct Edge {
        int to;
        int capacity;
        Edge *rev;
        Edge(int to, int capacity) : to(to), capacity(capacity) {
        }
    };
    int n;
    int source, sink;
    vector<vector<Edge *>> graph;
    MaximumFlow(int n, int source, int sink) : n(n), source(source), sink(sink) {
        graph.resize(n);
    };
    void add_edge(int u, int v, int cap) {
        Edge *ori = new Edge(v,cap);
        Edge *rev = new Edge(u,0);
        ori->rev = rev;
        rev->rev = ori;
        graph[u].push_back(ori);
        graph[v].push_back(rev);
    }
    void add_edge_from_source(int v, int cap) {
        add_edge(source, v, cap);
    }
    void add_edge_to_sink(int u, int cap) {
        add_edge(u, sink, cap);
    }
    int bfs() {
        vector<bool> check(n, false);
        vector<pair<int,int>> from(n, make_pair(-1,-1));
        queue<int> q;
        q.push(source);
        check[source] = true;
        while (!q.empty()) {
            int x = q.front();
            q.pop();
            for (int i=0; i<graph[x].size(); i++) {
                if (graph[x][i]->capacity > 0 && !check[graph[x][i]->to]) {
                    q.push(graph[x][i]->to);
                    check[graph[x][i]->to] = true;
                    from[graph[x][i]->to] = make_pair(x,i);
                }
            }
        }
        if (!check[sink]) {
            return 0;
        }
        int x = sink;
        int c = graph[from[x].first][from[x].second]->capacity;
        while (from[x].first != -1) {
            if (c > graph[from[x].first][from[x].second]->capacity) {
                c = graph[from[x].first][from[x].second]->capacity;
            }
            x = from[x].first;
        }
        x = sink;
        while (from[x].first != -1) {
            Edge *e = graph[from[x].first][from[x].second];
            e->capacity -= c;
            e->rev->capacity += c;
            x = from[x].first;
        }
        return c;
    }
    int flow() {
        int ans = 0;
        while (true) {
            int f = bfs();
            if (f == 0) break;
            ans += f;
        }
        return ans;
    }
};
int dx[] = {0,0,1,-1};
int dy[] = {1,-1,0,0};
int in_node(int x, int y, int m) {
    return 2*(x*m+y);
}
int out_node(int x, int y, int m) {
    return in_node(x,y,m)+1;
}
int main() {
    int n,m;
    cin >> n >> m;
    vector<string> a(n);
    for (int i=0; i<n; i++) {
        cin >> a[i];
    }
    int sx,sy;
    int ex,ey;
    for (int i=0; i<n; i++) {
        for (int j=0; j<m; j++) {
            if (a[i][j] == 'K') {
                sx=i;
                sy=j;
            } else if (a[i][j] == 'H') {
                ex=i;
                ey=j;
            }
        }
    }
    int inf = 1000000;
    MaximumFlow mf(2*(n*m), out_node(sx,sy,m), in_node(ex,ey,m));
    for (int i=0; i<n; i++) {
        for (int j=0; j<m; j++) {
            if (a[i][j] == '#') continue;
            mf.add_edge(in_node(i,j,m), out_node(i,j,m), 1);
            for (int k=0; k<4; k++) {
                int nx = i+dx[k];
                int ny = j+dy[k];
                if (0 <= nx && nx < n && 0 <= ny && ny < m) {
                    if (a[nx][ny] != '#') {
                        mf.add_edge(out_node(i,j,m), in_node(nx,ny,m), inf);
                    }
                }
            }
        }
    }
    int ans = mf.flow();
    if (ans >= inf) {
        ans = -1;
    }
    printf("%d\n",ans);
    return 0;
}
```

결과	메모리	시간	코드 길이
맞았습니다!!	4084 KB	4 ms	3622 B

C++14

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <cstring>
#include <string>
#include <map>
#include <queue>
using namespace std;
struct MaximumFlow {
    int n;
    int source, sink;
    vector<vector<int>> graph;
    vector<bool> check;
    vector<int> pred;
    MaximumFlow(int n) : n(n) {
        graph.resize(n);
        check.resize(n);
        pred.resize(n,-1);
    };
    void add_edge(int u, int v) {
        graph[u].push_back(v);
    }
    bool dfs(int x) {
        if (x == -1) return true;
        for (int next : graph[x]) {
            if (check[next]) continue;
            check[next] = true;
            if (dfs(pred[next])) {
                pred[next] = x;
                return true;
            }
        }
        return false;
    }
    int flow() {
        int ans = 0;
        for (int i=0; i<n; i++) {
            fill(check.begin(),check.end(),false);
            if (dfs(i)) {
                ans += 1;
            }
        }
        return ans;
    }
};

int main() {
    int n,m;
    scanf("%d %d",&n,&m);
    MaximumFlow mf(n);
    for (int i=0; i<m; i++) {
        int u,v;
        scanf("%d %d",&u,&v);
        mf.add_edge(u-1,v-1);
    }
    printf("%d\n",mf.flow());
    return 0;
}
```


결과	메모리	시간	코드 길이
맞았습니다!!	1992 KB	0 ms	1240 B

C++14

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <cstring>
#include <string>
#include <map>
#include <queue>
using namespace std;
struct MaximumFlow {
    int n;
    int source, sink;
    vector<vector<int>> graph;
    vector<bool> check;
    vector<int> pred;
    MaximumFlow(int n) : n(n) {
        graph.resize(n);
        check.resize(n);
        pred.resize(n,-1);
    };
    void add_edge(int u, int v) {
        graph[u].push_back(v);
    }
    bool dfs(int x) {
        if (x == -1) return true;
        for (int next : graph[x]) {
            if (check[next]) continue;
            check[next] = true;
            if (dfs(pred[next])) {
                pred[next] = x;
                return true;
            }
        }
        return false;
    }
    int flow() {
        int ans = 0;
        for (int i=0; i<n; i++) {
            fill(check.begin(),check.end(),false);
            if (dfs(i)) {
                ans += 1;
            }
        }
        return ans;
    }
};
int node(int x, int y, int m) {
    return x*m+y;
}
int main() {
    int t;
    cin >> t;
    while (t--) {
        int n,m;
        cin >> n >> m;
        vector<string> a(n);
        for (int i=0; i<n; i++) {
            cin >> a[i];
        }
        MaximumFlow mf(n*m);
        for (int i=0; i<n; i++) {
            for (int j=0; j<m; j++) {
                if (a[i][j] == 'x') continue;
                if (i > 0) {
                    if (j > 0) {
                        if (a[i-1][j-1] == '.') {
                            if (j%2 == 0) {
                                mf.add_edge(node(i,j,m),node(i-1,j-1,m));
                            } else {
                                mf.add_edge(node(i-1,j-1,m),node(i,j,m));
                            }
                        }
                    }
                    if (j+1 < m) {
                        if (a[i-1][j+1] == '.') {
                            if (j%2 == 0) {
                                mf.add_edge(node(i,j,m),node(i-1,j+1,m));
                            } else {
                                mf.add_edge(node(i-1,j+1,m),node(i,j,m));
                            }
                        }
                    }
                }
                if (j > 0) {
                    if (a[i][j-1] == '.') {
                        if (j%2 == 0) {
                            mf.add_edge(node(i,j,m),node(i,j-1,m));
                        } else {
                            mf.add_edge(node(i,j-1,m),node(i,j,m));
                        }
                    }
                }
            }
        }
        int ans = 0;
        for (int i=0; i<n; i++) {
            for (int j=0; j<m; j++) {
                if (a[i][j] == '.') ans += 1;
            }
        }
        printf("%d\n",ans-mf.flow());
    }
    return 0;
}
```

결과	메모리	시간	코드 길이
맞았습니다!!	2220 KB	64 ms	2822 B

끝

코드 플러스

https://code.plus

- 슬라이드에 포함된 소스 코드를 보려면 "정보 수정 > 백준 온라인 저지 연동"을 통해 연동한 다음, "백준 온라인 저지"에 로그인해야 합니다.
- 강의 내용에 대한 질문은 코드 플러스의 "질문 게시판"에서 할 수 있습니다.
- 문제와 소스 코드는 슬라이드에 첨부된 링크를 통해서 볼 수 있으며, "백준 온라인 저지"에서 서비스됩니다.
- 슬라이드와 동영상 강의는 코드 플러스 사이트를 통해서만 볼 수 있으며, 동영상 강의의 녹화와 다운로드, 배포와 유통은 저작권법에 의해서 금지되어 있습니다.
- 다른 경로로 이 슬라이드나 동영상 강의를 본 경우에는 codeplus@startlink.io 로 이메일 보내주세요.
- 강의 내용, 동영상 강의, 슬라이드, 첨부되어 있는 소스 코드의 저작권은 스타트링크와 최백준에게 있습니다.