

시트링크 플로우

소스코드

최백준 choi@startlink.io



C++14

Ford-Fulkerson

$A[u][v] = A[v][u]$

인접 리스트

→ to



$u \xrightarrow{cap} v$

$u \rightarrow v$
 $v \rightarrow u$

X

간선의 최소
sink로 가는 경로를 찾지 못함

② sink로 찾음

$X \xrightarrow{cap} Y$

bfs

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 #include <cstring>
5 #include <string>
6 #include <map>
7 #include <queue>
8 using namespace std;
9 struct MaximumFlow {
10     struct Edge {
11         int to;
12         int capacity;
13         Edge *rev;
14         Edge(int to, int capacity) : to(to), capacity(capacity) {}
15     };
16 };
17 int n;
18 int source, sink;
19 vector<vector<Edge *>> graph;
20 vector<bool> check;
21 MaximumFlow(int n, int source, int sink) : n(n), source(source), sink(sink) {
22     graph.resize(n);
23     check.resize(n);
24 };
25 void add_edge(int u, int v, int cap) {
26     Edge *ori = new Edge(v, cap);
27     Edge *rev = new Edge(u, 0);
28     ori->rev = rev;
29     rev->rev = ori;
30     graph[u].push_back(ori);
31     graph[v].push_back(rev);
32 };
33 int dfs(int x, int c) {
34     if (check[x]) return 0;
35     check[x] = true;
36     if (x == sink) {
37         return c;
38     }
39     for (int i=0; i<graph[x].size(); i++) {
40         if (graph[x][i]->capacity > 0) {
41             int nc = graph[x][i]->capacity;
42             if (c != 0 && c <= nc) {
43                 nc = c;
44             }
45             int f = dfs(graph[x][i]->to, nc);
46             if (f) {
47                 graph[x][i]->capacity -= f;
48                 graph[x][i]->rev->capacity += f;
49                 return f;
50             }
51         }
52     }
53     return 0;
54 };
55 int flow() {
56     int ans = 0;
57     while (true) {
58         fill(check.begin(), check.end(), false);
59         int f = dfs(source, 0);
60         if (f == 0) break;
61         ans += f;
62     }
63     return ans;
64 };
65 };
66 int node(string s) {
67     if (s[0] >= 'A' && s[0] <= 'Z') {
68         return s[0] - 'A';
69     } else {
70         return s[0] - 'a' + 26;
71     }
72 };
73 int main() {
74     int m;
75     cin >> m;
76     MaximumFlow mf(52, 0, 'Z'-'A');
77     for (int i=0; i<m; i++) {
78         string us, vs;
79         int f;
80         cin >> us >> vs >> f;
81         int u = node(us);
82         int v = node(vs);
83         mf.add_edge(u, v, f);
84         mf.add_edge(v, u, f);
85     }
86     cout << mf.flow() << '\n';
87     return 0;
88 }
```

결과

메모리

시간

코드 길이

맞았습니다!!

1996 KB

0 ms

2145 B

C++14

Edmond-Karp

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 #include <cstring>
5 #include <string>
6 #include <map>
7 #include <queue>
8 using namespace std;
9 struct MaximumFlow {
10     struct Edge {
11         int to;
12         int capacity;
13         Edge *rev;
14         Edge(int to, int capacity) : to(to), capacity(capacity) {
15         }
16     };
17     int n;
18     int source, sink;
19     vector<vector<Edge *>> graph;
20     MaximumFlow(int n, int source, int sink) : n(n), source(source), sink(sink) {
21         graph.resize(n);
22     };
23     void add_edge(int u, int v, int cap) {
24         Edge *ori = new Edge(v, cap);
25         Edge *rev = new Edge(u, 0);
26         ori->rev = rev;
27         rev->rev = ori;
28         graph[u].push_back(ori);
29         graph[v].push_back(rev);
30     }
31     int bfs() {
32         vector<bool> check(n, false);
33         vector<pair<int, int>> from(n, make_pair(-1, -1));
34         queue<int> q;
35         q.push(source);
36         check[source] = true;
37         while (!q.empty()) {
38             int x = q.front();
39             q.pop();
40             for (int i = 0; i < graph[x].size(); i++) {
41                 if (graph[x][i]->capacity > 0 && !check[graph[x][i]->to]) {
42                     q.push(graph[x][i]->to);
43                     check[graph[x][i]->to] = true;
44                     from[graph[x][i]->to] = make_pair(x, i);
45                 }
46             }
47         }
48         if (!check[sink]) {
49             return 0;
50         }
51         int x = sink;
52         int c = graph[from[x].first][from[x].second]->capacity;
53         while (from[x].first != -1) {
54             if (c > graph[from[x].first][from[x].second]->capacity) {
55                 c = graph[from[x].first][from[x].second]->capacity;
56             }
57             x = from[x].first;
58         }
59         x = sink;
60         while (from[x].first != -1) {
61             Edge *e = graph[from[x].first][from[x].second];
62             e->capacity -= c;
63             e->rev->capacity += c;
64             x = from[x].first;
65         }
66         return c;
67     }
68     int flow() {
69         int ans = 0;
70         while (true) {
71             int f = bfs();
72             if (f == 0) break;
73             ans += f;
74         }
75         return ans;
76     }
77 };
78 int node(string s) {
79     if (s[0] >= 'A' && s[0] <= 'Z') {
80         return s[0] - 'A';
81     } else {
82         return s[0] - 'a' + 26;
83     }
84 }
85 int main() {
86     int m;
87     cin >> m;
88     MaximumFlow mf(52, 0, 'Z'-'A');
89     for (int i=0; i<m; i++) {
90         string us, vs;
91         int f;
92         cin >> us >> vs >> f;
93         int u = node(us);
94         int v = node(vs);
95         mf.add_edge(u, v, f);
96         mf.add_edge(v, u, f);
97     }
98     cout << mf.flow() << '\n';
99     return 0;
100 }
```

인장리소트
ACW 간소=1

영구적
u -> v
from[x].first = u
.second =
first: 정점번호
Second: u -> v 간소의
ACW의 몇 번째?

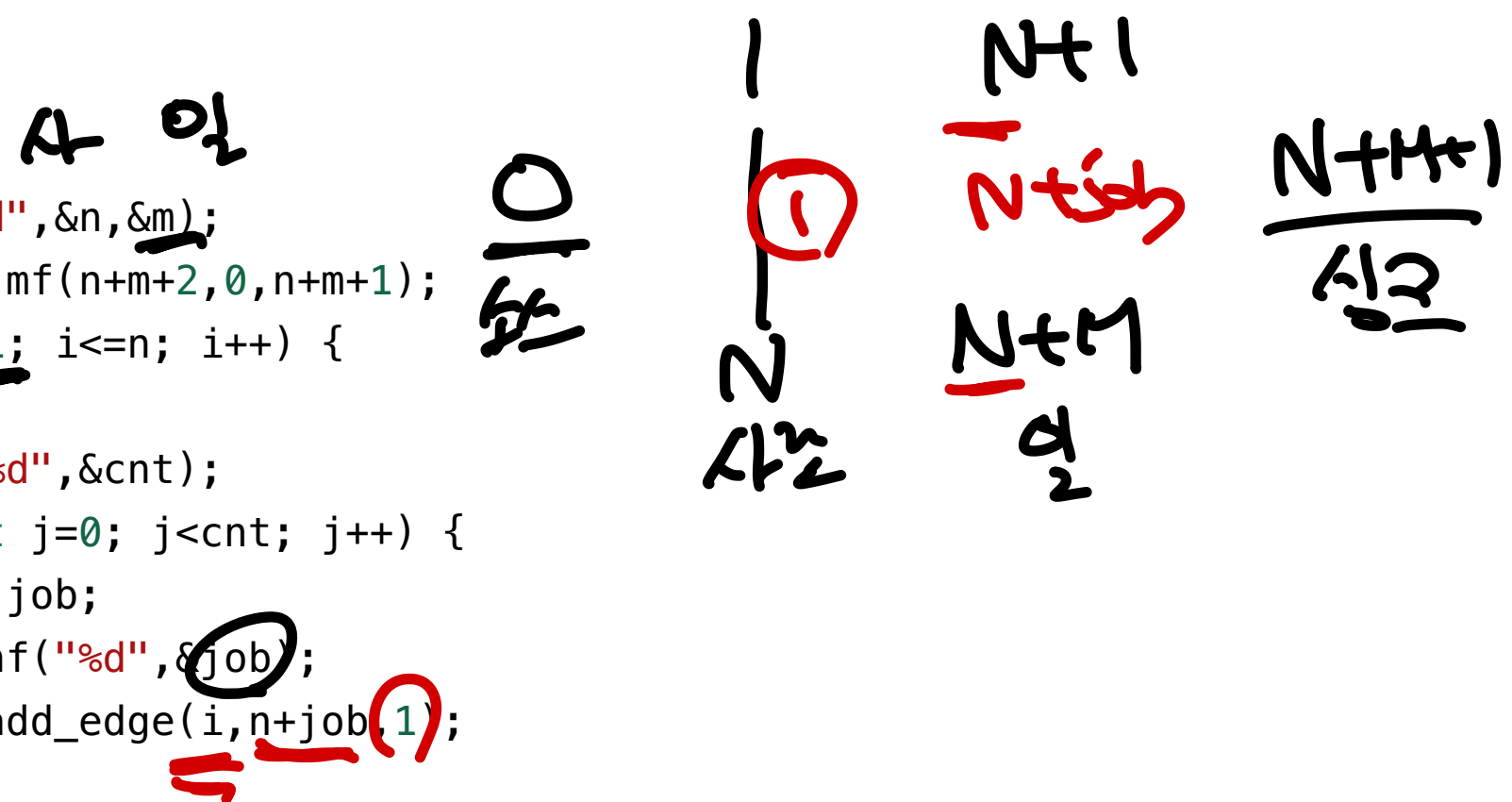
최소값

e->capacity -= c;
e->rev->capacity += c;

bfs()

C++14

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 #include <cstring>
5 #include <string>
6 #include <map>
7 #include <queue>
8 using namespace std;
9 struct MaximumFlow {
10     struct Edge {
11         int to;
12         int capacity;
13         Edge *rev;
14         Edge(int to, int capacity) : to(to), capacity(capacity) {
15             }
16     };
17     int n;
18     int source, sink;
19     vector<vector<Edge *>> graph;
20     vector<bool> check;
21     MaximumFlow(int n, int source, int sink) : n(n), source(source), sink(sink) {
22         graph.resize(n);
23         check.resize(n);
24     };
25     void add_edge(int u, int v, int cap) {
26         Edge *ori = new Edge(v, cap);
27         Edge *rev = new Edge(u, 0);
28         ori->rev = rev;
29         rev->rev = ori;
30         graph[u].push_back(ori);
31         graph[v].push_back(rev);
32     }
33     void add_edge_from_source(int v, int cap) {
34         add_edge(source, v, cap);
35     }
36     void add_edge_to_sink(int u, int cap) {
37         add_edge(u, sink, cap);
38     }
39     int dfs(int x, int c) {
40         if (check[x]) return 0;
41         check[x] = true;
42         if (x == sink) {
43             return c;
44         }
45         for (int i=0; i<graph[x].size(); i++) {
46             if (graph[x][i]->capacity > 0) {
47                 int nc = graph[x][i]->capacity;
48                 if (c != 0 && c < nc) {
49                     nc = c;
50                 }
51                 int f = dfs(graph[x][i]->to, nc);
52                 if (f) {
53                     graph[x][i]->capacity -= f;
54                     graph[x][i]->rev->capacity += f;
55                     return f;
56                 }
57             }
58         }
59         return 0;
60     }
61     int flow() {
62         int ans = 0;
63         while (true) {
64             fill(check.begin(), check.end(), false);
65             int f = dfs(source, 0);
66             if (f == 0) break;
67             ans += f;
68         }
69         return ans;
70     }
71 };
72
73 int main() {
74     int n, m;
75     scanf("%d %d", &n, &m);
76     MaximumFlow mf(n+m+2, 0, n+m+1);
77     for (int i=1; i<=n; i++) {
78         int cnt;
79         scanf("%d", &cnt);
80         for (int j=0; j<cnt; j++) {
81             int job;
82             scanf("%d", &job);
83             mf.add_edge(i, n+job, 1);
84         }
85     }
86     for (int i=1; i<=n; i++) {
87         mf.add_edge_from_source(i, 1);
88     }
89     for (int i=1; i<=m; i++) {
90         mf.add_edge_to_sink(n+i, 1);
91     }
92     printf("%d\n", mf.flow());
93     return 0;
94 }
```



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

C++14

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 #include <cstring>
5 #include <string>
6 #include <map>
7 #include <queue>
8 using namespace std;
9 struct MaximumFlow {
10     int n;
11     int source, sink;
12     vector<vector<int>>> graph;
13     vector<bool> check;
14     vector<int> pred;
15     MaximumFlow(int n) : n(n) {
16         graph.resize(n);
17         check.resize(n);
18         pred.resize(n,-1);
19     };
20     void add_edge(int u, int v) {
21         graph[u].push_back(v);
22     }
23     bool dfs(int x) {
24         if (x == -1) return true;
25         for (int next : graph[x]) {
26             if (check[next]) continue;
27             check[next] = true;
28             if (dfs(pred[next])) {
29                 pred[next] = x;
30                 return true;
31             }
32         }
33         return false;
34     }
35     int flow() {
36         int ans = 0;
37         for (int i=0; i<n; i++) {
38             fill(check.begin(),check.end(),false);
39             if (dfs(i)) {
40                 ans += 1;
41             }
42         }
43         return ans;
44     }
45 };
46
47 int main() {
48     int n,m;
49     scanf("%d %d",&n,&m);
50     MaximumFlow mf(max(n,m));
51     for (int i=0; i<n; i++) {
52         int cnt;
53         scanf("%d",&cnt);
54         for (int j=0; j<cnt; j++) {
55             int job;
56             scanf("%d",&job);
57             mf.add_edge(i,job-1);
58         }
59     }
60     printf("%d\n",mf.flow());
61     return 0;
62 }
```

결과

메모리

시간

코드 길이

맞았습니다!!

5948 KB

344 ms

1344 B

C++14

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 #include <cstring>
5 #include <string>
6 #include <map>
7 #include <queue>
8 using namespace std;
9 struct MaximumFlow {
10     int n;
11     int source, sink;
12     vector<vector<int>> graph;
13     vector<bool> check;
14     vector<int> pred;
15     MaximumFlow(int n) : n(n) {
16         graph.resize(n);
17         check.resize(n);
18         pred.resize(n,-1);
19     };
20     void add_edge(int u, int v) {
21         graph[u].push_back(v);
22     }
23     bool dfs(int x) {
24         if (x == -1) return true;
25         for (int next : graph[x]) {
26             if (check[next]) continue;
27             check[next] = true;
28             if (dfs(pred[next])) {
29                 pred[next] = x;
30                 return true;
31             }
32         }
33         return false;
34     }
35     int flow() {
36         int ans = 0;
37         for (int i=0; i<n; i++) {
38             for (int j=0; j<2; j++) {
39                 fill(check.begin(),check.end(),false);
40                 if (dfs(i)) {
41                     ans += 1;
42                 }
43             }
44         }
45         return ans;
46     }
47 };
48
49 int main() {
50     int n,m;
51     scanf("%d %d",&n,&m);
52     MaximumFlow mf(max(n,m));
53     for (int i=0; i<n; i++) {
54         int cnt;
55         scanf("%d",&cnt);
56         for (int j=0; j<cnt; j++) {
57             int job;
58             scanf("%d",&job);
59             mf.add_edge(i,job-1);
60         }
61     }
62     printf("%d\n",mf.flow());
63     return 0;
64 }
```

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

C++14

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 #include <cstring>
5 #include <string>
6 #include <map>
7 #include <queue>
8 using namespace std;
9 struct MaximumFlow {
10     struct Edge {
11         int to;
12         int capacity;
13         Edge *rev;
14         Edge(int to, int capacity) : to(to), capacity(capacity) {
15             }
16     };
17     int n;
18     int source, sink;
19     vector<vector<Edge *>> graph;
20     MaximumFlow(int n, int source, int sink) : n(n), source(source), sink(sink) {
21         graph.resize(n);
22     };
23     void add_edge(int u, int v, int cap) {
24         Edge *ori = new Edge(v, cap);
25         Edge *rev = new Edge(u, 0);
26         ori->rev = rev;
27         rev->rev = ori;
28         graph[u].push_back(ori);
29         graph[v].push_back(rev);
30     }
31     void add_edge_from_source(int v, int cap) {
32         add_edge(source, v, cap);
33     }
34     void add_edge_to_sink(int u, int cap) {
35         add_edge(u, sink, cap);
36     }
37     int bfs() {
38         vector<bool> check(n, false);
39         vector<pair<int, int>> from(n, make_pair(-1, -1));
40         queue<int> q;
41         q.push(source);
42         check[source] = true;
43         while (!q.empty()) {
44             int x = q.front();
45             q.pop();
46             for (int i=0; i<graph[x].size(); i++) {
47                 if (graph[x][i]->capacity > 0 && !check[graph[x][i]->to]) {
48                     q.push(graph[x][i]->to);
49                     check[graph[x][i]->to] = true;
50                     from[graph[x][i]->to] = make_pair(x, i);
51                 }
52             }
53         }
54         if (!check[sink]) {
55             return 0;
56         }
57         int x = sink;
58         int c = graph[from[x].first][from[x].second]->capacity;
59         while (from[x].first != -1) {
60             if (c > graph[from[x].first][from[x].second]->capacity) {
61                 c = graph[from[x].first][from[x].second]->capacity;
62             }
63             x = from[x].first;
64         }
65         x = sink;
66         while (from[x].first != -1) {
67             Edge *e = graph[from[x].first][from[x].second];
68             e->capacity -= c;
69             e->rev->capacity += c;
70             x = from[x].first;
71         }
72         return c;
73     }
74     int flow() {
75         int ans = 0;
76         while (true) {
77             int f = bfs();
78             if (f == 0) break;
79             ans += f;
80         }
81         return ans;
82     }
83 };
84 int dx[] = {0, 0, 1, -1};
85 int dy[] = {1, -1, 0, 0};
86 int in_node(int x, int y, int m) {
87     return 2*(x*m+y);
88 }
89 int out_node(int x, int y, int m) {
90     return in_node(x, y, m)+1;
91 }
92 int main() {
93     int n, m;
94     cin >> n >> m;
95     vector<string> a(n);
96     for (int i=0; i<n; i++) {
97         cin >> a[i];
98     }
99     int sx, sy;
100     int ex, ey;
101     for (int i=0; i<n; i++) {
102         for (int j=0; j<m; j++) {
103             if (a[i][j] == 'K') {
104                 sx=i;
105                 sy=j;
106             } else if (a[i][j] == 'H') {
107                 ex=i;
108                 ey=j;
109             }
110         }
111     }
112     int inf = 1000000;
113     MaximumFlow mf(2*(n*m), out_node(sx, sy, m), in_node(ex, ey, m));
114     for (int i=0; i<n; i++) {
115         for (int j=0; j<m; j++) {
116             if (a[i][j] == '#') continue;
117             mf.add_edge(in_node(i, j, m), out_node(i, j, m), 1);
118             for (int k=0; k<4; k++) {
119                 int nx = i+dx[k];
120                 int ny = j+dy[k];
121                 if (0 <= nx && nx < n && 0 <= ny && ny < m) {
122                     if (a[nx][ny] != '#') {
123                         mf.add_edge(out_node(i, j, m), in_node(nx, ny, m), inf);
124                     }
125                 }
126             }
127         }
128     }
129     int ans = mf.flow();
130     if (ans >= inf) {
131         ans = -1;
132     }
133     printf("%d\n", ans);
134     return 0;
135 }
```

결과

메모리

시간

코드 길이

맞았습니다!!

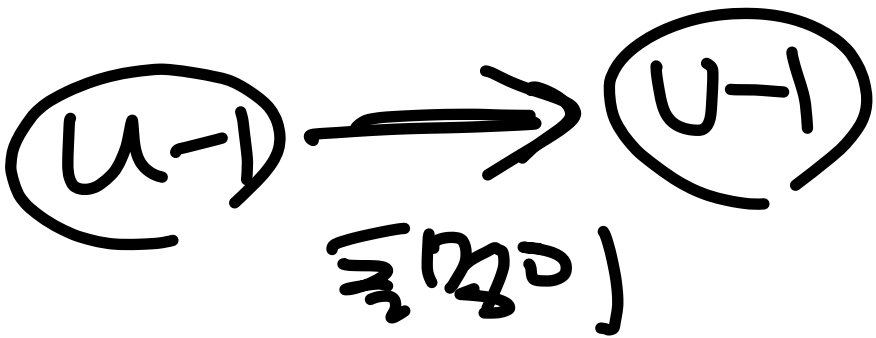
4084 KB

4 ms

3622 B

C++14

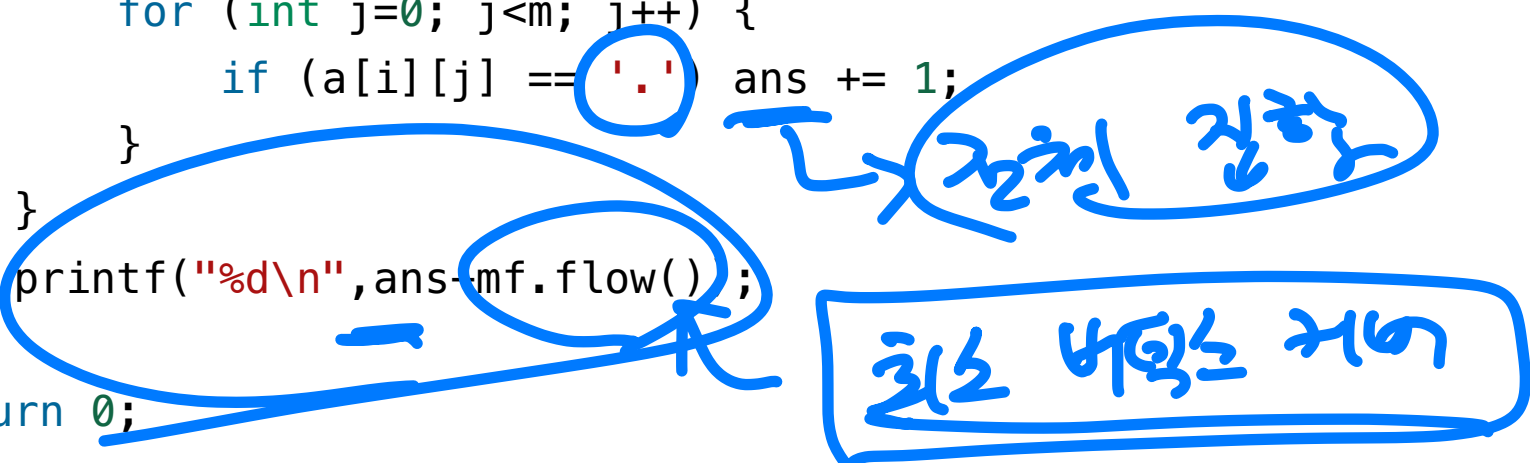
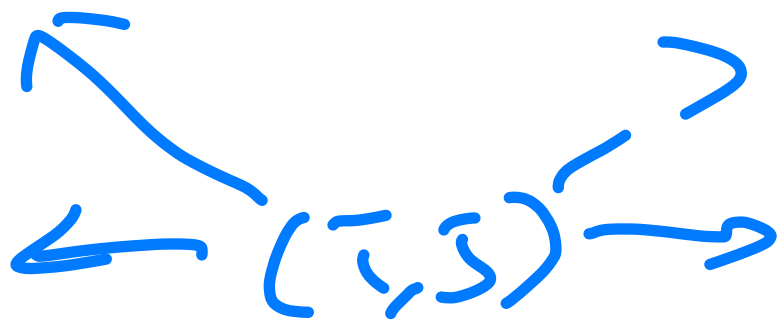
```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 #include <cstring>
5 #include <string>
6 #include <map>
7 #include <queue>
8 using namespace std;
9 struct MaximumFlow {
10     int n;
11     int source, sink;
12     vector<vector<int>> graph;
13     vector<bool> check;
14     vector<int> pred;
15     MaximumFlow(int n) : n(n) {
16         graph.resize(n);
17         check.resize(n);
18         pred.resize(n,-1);
19     };
20     void add_edge(int u, int v) {
21         graph[u].push_back(v);
22     }
23     bool dfs(int x) {
24         if (x == -1) return true;
25         for (int next : graph[x]) {
26             if (check[next]) continue;
27             check[next] = true;
28             if (dfs(pred[next])) {
29                 pred[next] = x;
30                 return true;
31             }
32         }
33         return false;
34     }
35     int flow() {
36         int ans = 0;
37         for (int i=0; i<n; i++) {
38             fill(check.begin(),check.end(),false);
39             if (dfs(i)) {
40                 ans += 1;
41             }
42         }
43         return ans;
44     }
45 };
46
47 int main() {
48     int n,m;
49     scanf("%d %d",&n,&m);
50     MaximumFlow mf(n);
51     for (int i=0; i<m; i++) {
52         int u,v;
53         scanf("%d %d",&u,&v);
54         mf.add_edge(u-1,v-1);
55     }
56     printf("%d\n",mf.flow());
57     return 0;
58 }
```



최소 배속스 커

C++14

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 #include <cstring>
5 #include <string>
6 #include <map>
7 #include <queue>
8 using namespace std;
9 struct MaximumFlow {
10     int n;
11     int source, sink;
12     vector<vector<int>> graph;
13     vector<bool> check;
14     vector<int> pred;
15     MaximumFlow(int n) : n(n) {
16         graph.resize(n);
17         check.resize(n);
18         pred.resize(n,-1);
19     };
20     void add_edge(int u, int v) {
21         graph[u].push_back(v);
22     }
23     bool dfs(int x) {
24         if (x == -1) return true;
25         for (int next : graph[x]) {
26             if (check[next]) continue;
27             check[next] = true;
28             if (dfs(pred[next])) {
29                 pred[next] = x;
30                 return true;
31             }
32         }
33         return false;
34     }
35     int flow() {
36         int ans = 0;
37         for (int i=0; i<n; i++) {
38             fill(check.begin(),check.end(),false);
39             if (dfs(i)) {
40                 ans += 1;
41             }
42         }
43         return ans;
44     }
45 };
46 int node(int x, int y, int m) {
47     return x*m+y;
48 }
49 int main() {
50     int t;
51     cin >> t;
52     while (t--) {
53         int n,m;
54         cin >> n >> m;
55         vector<string> a(n);
56         for (int i=0; i<n; i++) {
57             cin >> a[i];
58         }
59         MaximumFlow mf(n*m);
60         for (int i=0; i<n; i++) {
61             for (int j=0; j<m; j++) {
62                 if (a[i][j] == 'x' continue;
63                 if (i > 0) {
64                     if (j > 0) {
65                         if (a[i-1][j-1] == '.') {
66                             if (j%2 == 0) {
67                                 mf.add_edge(node(i,j,m),node(i-1,j-1,m));
68                             } else {
69                                 mf.add_edge(node(i-1,j-1,m),node(i,j,m));
70                             }
71                         }
72                     }
73                     if (j+1 < m) {
74                         if (a[i-1][j+1] == '.') {
75                             if (j%2 == 0) {
76                                 mf.add_edge(node(i,j,m),node(i-1,j+1,m));
77                             } else {
78                                 mf.add_edge(node(i-1,j+1,m),node(i,j,m));
79                             }
80                         }
81                     }
82                 }
83                 if (j > 0) {
84                     if (a[i][j-1] == '.') {
85                         if (j%2 == 0) {
86                             mf.add_edge(node(i,j,m),node(i,j-1,m));
87                         } else {
88                             mf.add_edge(node(i,j-1,m),node(i,j,m));
89                         }
90                     }
91                 }
92             }
93         }
94         int ans = 0;
95         for (int i=0; i<n; i++) {
96             for (int j=0; j<m; j++) {
97                 if (a[i][j] == '.' ans += 1;
98             }
99         }
100         printf("%d\n",ans-mf.flow());
101     }
102     return 0;
103 }
```



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.