

2022-1 HI-ARC 초급스터디

8주차. 그래프 탐색

이지은 (leeju1013)

목차

1. 그래프
2. 그래프 탐색
 - 1) DFS
 - 2) BFS

바이러스

성공



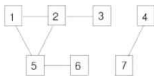
3 실버 III

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	128 MB	98051	46574	31449	45.811%

문제

신종 바이러스인 웜 바이러스는 네트워크를 통해 전파된다. 한 컴퓨터가 웜 바이러스에 걸리면 그 컴퓨터와 네트워크 상에서 연결되어 있는 모든 컴퓨터는 웜 바이러스에 걸리게 된다.

예를 들어 7대의 컴퓨터가 <그림 1>과 같이 네트워크 상에서 연결되어 있다고 하자. 1번 컴퓨터가 웜 바이러스에 걸리면 웜 바이러스는 2번과 5번 컴퓨터를 거쳐 3번과 6번 컴퓨터까지 전파되어 2, 3, 5, 6 네 대의 컴퓨터는 웜 바이러스에 걸리게 된다. 하지만 4번과 7번 컴퓨터는 1번 컴퓨터와 네트워크 상에서 연결되어 있지 않기 때문에 영향을 받지 않는다.



< 그림 1 >

예제 입력 1 복사

```

7
6
1 2
2 3
1 5
5 2
5 6
4 7

```

예제 출력 1 복사

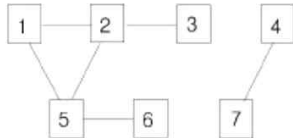
4

어느 날 1번 컴퓨터가 웜 바이러스에 걸렸다. 컴퓨터의 수와 네트워크 상에서 서로 연결되어 있는 정보가 주어질 때, 1번 컴퓨터를 통해 웜 바이러스에 걸리게 되는 컴퓨터의 수를 출력하는 프로그램을 작성하시오.

1. 그래프

- What?

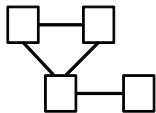
- 정점(Vertex)과 간선(Edge)의 집합으로 이루어진 자료구조



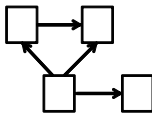
< 그림 1 >

- 정점 : 1, 2, 3, 4, 5, 6, 7
- 간선 : (1, 2), (2, 3), (1, 5), (5, 2), (5, 6), (4, 7)
- 인접(adjacent) : 간선으로 이어진 두 정점
- 차수(degree) : 정점에 연결된 간선의 수

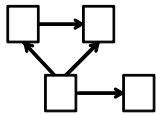
• 방향성 : 무방향 그래프



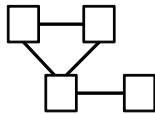
방향 그래프



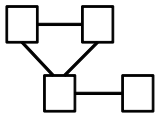
• 사이클 : 비순환 그래프



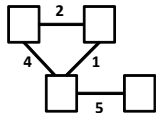
순환 그래프



• 가중치 : 비가중 그래프

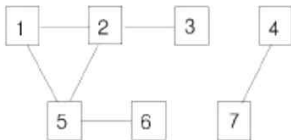


가중 그래프



1. 그래프

- How? _ 인접 행렬 `int adj[8][8]`



< 그림 1 >

	1	2	3	4	5	6	7
1	0	1	0	0	1	0	0
2	1	0	1	0	1	0	0
3	0	1	0	0	0	0	0
4	0	0	0	0	0	0	1
5	1	1	0	0	0	1	0
6	0	0	0	0	1	0	0
7	0	0	0	1	0	0	0

- 인접하면 1, 인접하지 않으면 0

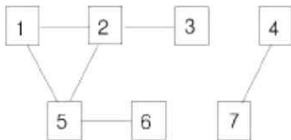
- 장점: 간선 여부를 $O(1)$ 에 확인

- 단점: 공간 복잡도 $O(|V|^2)$

-> 간선 수가 많은 경우 사용

1. 그래프

- How? _ 인접 리스트 `vector<int> adj[8]`

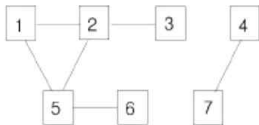


< 그림 1 >

1	2	5	
2	1	3	5
3	2		
4	7		
5	1	2	6
6	5		
7	4		

- 해당 점점에서 나가는 간선 저장
- 장점: 공간 복잡도 $O(|V|+|E|)$
- 단점: 간선 여부를 $O(\text{degree})$ 에 확인
- > 정점 수가 많은 경우 사용

* 대부분 문제들이 간선에 비해 정점 개수가 많기도 하고,
특정 노드에 연결된 모든 노드를 찾는 경우가 많음
-> 공간도 적게 사용하며, 탐색시간도 빠른 인접 리스트를 주로 사용!



< 그림 1 >

예제 입력 1 복사

```
7
6
1 2
2 3
1 5
5 2
5 6
4 7
```

```
1 int adj[8][8]={};
2 int v,e;
3 cin>> v >> e;
4 while(e--){
5     int a,b;
6     cin>> a >> b;
7     adj[a][b]=1;
8     adj[b][a]=1;
9 }
```

	1	2	3	4	5	6	7
1	0	1	0	0	1	0	0
2	1	0	1	0	1	0	0
3	0	1	0	0	0	0	0
4	0	0	0	0	0	0	1
5	1	1	0	0	0	1	0
6	0	0	0	0	1	0	0
7	0	0	0	1	0	0	0

<인접 행렬>

입력

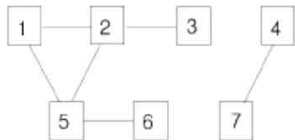
첫째 줄에는 컴퓨터의 수가 주어진다. 컴퓨터의 수는 100 이하이고 각 컴퓨터에는 1번 부터 차례대로 번호가 매겨진다. 둘째 줄에는 네트워크 상에서 직접 연결되어 있는 컴퓨터 쌍의 수가 주어진다. 이어서 그 수만큼 한 줄에 한 쌍씩 네트워크 상에서 직접 연결되어 있는 컴퓨터의 번호 쌍이 주어진다.

```
1 vector<int> adj[8];
2 int v,e;
3 cin>> v >> e;
4 while(e--){
5     int a,b;
6     cin>> a >> b;
7     adj[a].push_back(b);
8     adj[b].push_back(a);
9 }
```

1	2	5	
2	1	3	5
3	2		
4	7		
5	1	2	6
6	5		
7	4		

<인접 리스트>

2. 그래프 탐색

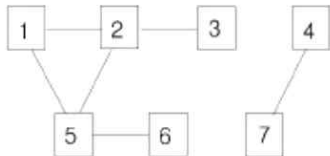


< 그림 1 >

- DFS (깊이 우선 탐색)
: 1 -> 2 -> 3 -> 5 -> 6
- BFS (너비 우선 탐색)
: 1 -> 2 -> 5 -> 3 -> 6

2-1. DFS (Depth First Search)

- What?
 - '깊이'를 우선적으로 탐색하는 알고리즘
- How?
 - 스택
 - 재귀함수



- DFS (깊이 우선 탐색)
: 1 -> 2 -> 3 -> 5 -> 6

1

1번 정점에서 시작!, visited[1]=1, push(1)

1번 정점 방문하고 pop(),

1번 정점과 인접하고 visited[x]==0 이면 visited[x]=1, push(x)

5 2

2번 정점 방문하고 pop(),

2번 정점과 인접하고 visited[x]==0 이면 visited[x]=1, push(x)

5 3

3번 정점 방문하고 pop(),

3번 정점과 인접하고 visited[x]==0 인 정점이 없으므로 그냥 넘어감

5

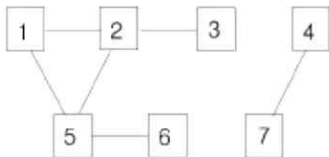
5번 정점 방문하고 pop(),

5번 정점과 인접하고 visited[x]==0 이면 visited[x]=1, push(x)

6

6번 정점 방문하고 pop(),

6번 정점과 인접하고 visited[x]==0 인 정점이 없으므로 그냥 넘어감. 탐색 끝!



- DFS(깊이 우선 탐색)
: 1 -> 2 -> 3 -> 5 -> 6

1

5 2

5 3

5

6

```

1 vector<int> adj[8];
2 int visited[8]={};
3
4 void dfs(int start){
5     stack<int> st;
6     st.push(start);
7     visited[start]=1;
8     while(!st.empty()){
9         int cur = st.top(); st.pop();
10        for(int next : adj[cur]){
11            if(!visited[next]){
12                visited[next] = 1;
13                st.push(next);
14            }
15        }
16    }
17 }
```

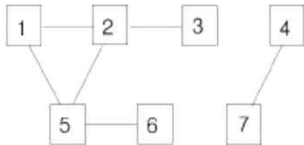
```

1 vector<int> adj[8];
2 int visited[8]={};
3
4 void dfs(int cur){
5     visited[cur]=1;
6     for(auto &next : adj[cur])
7         if(!visited[next]) dfs(next);
8 }
```

<재귀함수>

1	2	5	
2	1	3	5
3	2		
4	7		
5	1	2	6
6	5		
7	4		

<스택>



입력

첫째 줄에는 컴퓨터의 수가 주어진다. 컴퓨터의 수는 100 이하이고 각 컴퓨터에는 1번 부터 차례대로 번호가 매겨진다. 둘째 줄에는 네트워크 상에서 직접 연결되어 있는 컴퓨터 쌍의 수가 주어진다. 이어서 그 수만큼 한 줄에 한 쌍씩 네트워크 상에서 직접 연결되어 있는 컴퓨터의 번호 쌍이 주어진다.

출력

1번 컴퓨터가 웜 바이러스에 걸렸을 때, 1번 컴퓨터를 통해 웜 바이러스에 걸리게 되는 컴퓨터의 수를 첫째 줄에 출력한다.

예제 입력 1 복사

```

7
6
1 2
2 3
1 5
5 2
5 6
4 7
  
```

예제 출력 1 복사

```

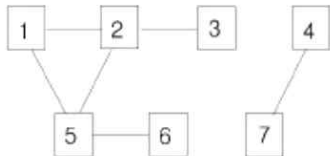
4
  
```

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 vector<int> adj[101];
6 int visited[101];
7 int ans;
8
9 void dfs(int cur){
10     visited[cur]=1;
11     ans++;
12     for(auto &next:adj[cur]){
13         if(!visited[next]) dfs(next);
14     }
15 }
16 int main(){
17     ios_base::sync_with_stdio(false); cin.tie(NULL);
18
19     int v,e; cin>> v >> e;
20     while(e--){
21         int a,b; cin>> a >> b;
22         adj[a].push_back(b);
23         adj[b].push_back(a);
24     }
25     dfs(1);
26     cout<<ans-1; //1번 컴퓨터는 제외
27     return 0;
28 }
  
```

2-2. BFS (Breadth First Search)

- What?
 - '너비'를 우선적으로 탐색하는 알고리즘
- How?
 - 큐



• BFS(너비 우선 탐색)

: 1 -> 2 -> 5 -> 3 -> 6

1

2 5

5 3

3 6

6

1번 정점에서 시작!, visited[1]=1, push(1)

1번 정점 방문하고 pop(),

1번 정점과 인접하고 visited[x]==0 이면 visited[x]=1, push(x)

2번 정점 방문하고 pop(),

2번 정점과 인접하고 visited[x]==0 이면 visited[x]=1, push(x)

5번 정점 방문하고 pop(),

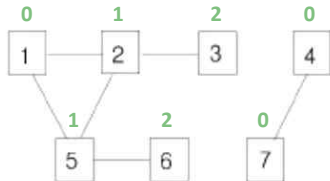
5번 정점과 인접하고 visited[x]==0 이면 visited[x]=1, push(x)

3번 정점 방문하고 pop(),

3번 정점과 인접하고 visited[x]==0 인 정점이 없으므로 그냥 넘어감

6번 정점 방문하고 pop(),

6번 정점과 인접하고 visited[x]==0 인 정점이 없으므로 그냥 넘어감. 탐색 끝!



1

2 5

5 3

3 6

6

- BFS (너비 우선 탐색)
: 1 -> 2 -> 5 -> 3 -> 6

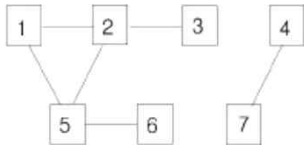
```

1 vector<int> adj[8];
2 int visited[8]={};
3 int dist[8]={};
4
5 void bfs(int start){
6     queue<int> q;
7     q.push(start);
8     visited[start]=1;
9     while(!q.empty()){
10         int cur = q.top(); q.pop();
11         for(int next : adj[cur]){
12             if(!visited[next]){
13                 visited[next] = 1;
14                 dist[next] = dist[cur]+1; //최단거리
15                 q.push(next);
16             }
17         }
18     }
19 }

```

1	2	5	
2	1	3	5
3	2		
4	7		
5	1	2	6
6	5		
7	4		

<큐>



입력

첫째 줄에는 컴퓨터의 수가 주어진다. 컴퓨터의 수는 100 이하이고 각 컴퓨터에는 1번 부터 차례대로 번호가 매겨진다. 둘째 줄에는 네트워크 상에서 직접 연결되어 있는 컴퓨터 쌍의 수가 주어진다. 이어서 그 수만큼 한 줄에 한 쌍씩 네트워크 상에서 직접 연결되어 있는 컴퓨터의 번호 쌍이 주어진다.

출력

1번 컴퓨터가 원 바이러스에 걸렸을 때, 1번 컴퓨터를 통해 원 바이러스에 걸리게 되는 컴퓨터의 수를 첫째 줄에 출력한다.

예제 입력 1 복사

```

7
6
1 2
2 3
1 5
5 2
5 6
4 7
  
```

예제 출력 1 복사

```

4
  
```

```

6 vector<int> adj[101];
7 int visited[101];
8 int ans;
9
10 void bfs(int start){
11     queue<int> q;
12     q.push(start);
13     visited[start]=1;
14     while(!q.empty()){
15         int cur = q.front(); q.pop();
16         for(int next:adj[cur]){
17             if(!visited[next]){
18                 visited[next] = 1;
19                 ans++;
20                 q.push(next);
21             }
22         }
23     }
24 }
25 int main(){
26     ios_base::sync_with_stdio(false); cin.tie(NULL);
27
28     int v,e; cin>> v >> e;
29     while(e--){
30         int a,b; cin>> a >> b;
31         adj[a].push_back(b);
32         adj[b].push_back(a);
33     }
34     bfs(1);
35     cout<<ans;
36     return 0;
37 }
  
```

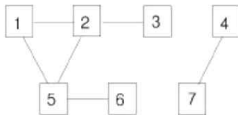
DFS

VS

BFS

- 깊이 우선 탐색
- 스택 (재귀함수)
- 사이클 검출
- 1 -> 2 -> 3 -> 5 -> 6

```
1 vector<int> adj[8];
2 int visited[8]={};
3
4 void dfs(int cur){
5     visited[cur]=1;
6     for(auto &next : adj[cur])
7         if(!visited[next]) dfs(next);
8 }
```



- 너비 우선 탐색
- 큐
- 최단거리 찾기
- 1 -> 2 -> 5 -> 3 -> 6

```
1 vector<int> adj[8];
2 int visited[8]={};
3 int dist[8]={};
4
5 void bfs(int start){
6     queue<int> q;
7     q.push(start);
8     visited[start]=1;
9     while(!q.empty()){
10         int cur = q.top(); q.pop();
11         for(int next : adj[cur]){
12             if(!visited[next]){
13                 visited[next] = 1;
14                 dist[next] = dist[cur]+1; //최단거리
15                 q.push(next);
16             }
17         }
18     }
19 }
```

그림

성공

1 실버 I

시간 제한	메모리 제한	제출	정답	맞
2 초	128 MB	14777	6355	44

문제

어떤 큰 도화지에 그림이 그려져 있을 때, 그 그림의 개수와, 그 그림 중 넓이가 가장 넓은 것의 넓이를 출력하여라. 단, 그림이라는 것은 1로 연결된 것을 한 그림이라고 정의하자. 가로나 세로로 연결된 것은 연결이 된 것이고 대각선으로 연결이 된 것은 떨어진 그림이다. 그림의 넓이란 그림에 포함된 1의 개수이다.

입력

첫째 줄에 도화지의 세로 크기 $n(1 \leq n \leq 500)$ 과 가로 크기 $m(1 \leq m \leq 500)$ 이 차례로 주어진다. 두 번째 줄부터 $n+1$ 줄 까지 그림의 정보가 주어진다. (단 그림의 정보는 0과 1이 공백을 두고 주어지며, 0은 색칠이 안된 부분, 1은 색칠이 된 부분을 의미한다)

출력

첫째 줄에는 그림의 개수, 둘째 줄에는 그 중 가장 넓은 그림의 넓이를 출력하여라. 단, 그림이 하나도 없는 경우에는 가장 넓은 그림의 넓이는 0이다.

예제 입력 1 복사



```
6 5
1 1 0 1 1
0 1 1 0 0
0 0 0 0 0
1 0 1 1 1
0 0 1 1 1
0 0 1 1 1
```

예제 출력 1 복사

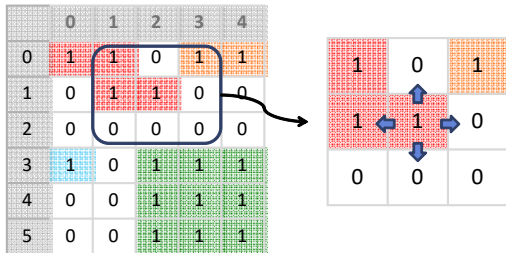
```
4
9
```

예제 입력 1 복사

```
6 5
1 1 0 1 1
0 1 1 0 0
0 0 0 0 0
1 0 1 1 1
0 0 1 1 1
0 0 1 1 1
```

예제 출력 1 복사

```
4      그림의 개수
9      그 중 가장 넓은 그림의 넓이
```



```
1 int matrix[501][501]; //인접하면 1, 인접하지 않으면 0
2 int visited[501][501];
3 int dx[4] = {-1, 0, 1, 0};
4 int dy[4] = {0, 1, 0, -1};
5 int ans;
6
7 void bfs(int x, int y){
8     queue<pair<int,int>> q;
9     q.push({x,y});
10    visited[x][y]=1;
11
12    while(!q.empty()){
13        int cur_x=q.front().first;
14        int cur_y=q.front().second;
15        q.pop();
16
17        for(int i=0; i<4; i++){
18            int nx=cur_x+dx[i], ny=cur_y+dy[i];
19            if(nx<0 || ny<0 || nx>=n || ny>=m) continue;
20            if(matrix[nx][ny] && !visited[nx][ny]){
21                visited[nx][ny]=1;
22                q.push({nx,ny});
23            }
24        }
25    }
26 }
```

```

1 #include <iostream>
2 #include <algorithm>
3 #include <vector>
4 #include <queue>
5
6 using namespace std;
7 int n, m, ans, cnt; //행, 열, 그림 개수, 그림 넓이
8 int matrix[501][501]; // 0 : 그림 없음, 1 : 그림 있음, 2 : 그림 방문했음
9 int dx[4] = {-1, 0, 1, 0};
10 int dy[4] = {0, 1, 0, -1};
11 vector<int> space; //그림들의 넓이 보관
12
13 void bfs(int x, int y){
14     queue<pair<int,int>> q;
15     matrix[x][y]=2; //방문 체크
16     cnt=1; //현재 그림의 넓이 초기화
17     q.push({x,y});
18
19     while(!q.empty()){
20         int cur_x=q.front().first;
21         int cur_y=q.front().second;
22         q.pop();
23
24         for(int i=0; i<4; i++){
25             int nx=cur_x+dx[i], ny=cur_y+dy[i];
26             if(nx<0 || ny<0 || nx>=n || ny>=m) continue;
27             if(matrix[nx][ny]==1){ //그림 있고 아직 방문 전이면
28                 matrix[nx][ny] = 2; //방문 체크
29                 cnt++; //현재 그림 넓이 계산
30                 q.push({nx, ny});
31             }
32         }
33     }
34     space.push_back(cnt); //현재 그림 넓이 보관
35 }

```

예제 입력 1 복사

```

6 5
1 1 0 1 1
0 1 1 0 0
0 0 0 0 0
1 0 1 1 1
0 0 1 1 1
0 0 1 1 1

```

예제 출력 1 복사

```

4
9

```

```

36 int main(){
37     ios_base::sync_with_stdio(false); ci
38
39     cin>>n>>m;
40     for(int i=0; i<n; i++){
41         for(int j=0; j<m; j++){
42             cin>>matrix[i][j];
43         }
44     }
45
46     for(int i=0; i<n; i++){
47         for(int j=0; j<m; j++){
48             if(matrix[i][j]==1){ //그림 있고 아직 방문 전이면
49                 bfs(i,j);
50                 ans++; // 그림의 개수
51             }
52         }
53     }
54     cout<< ans <<'\n';
55
56     if(!ans) cout<<0; // 그림이 하나도 없는 경우에는 가장 넓은 그림의 넓이는 0
57     else {
58         sort(space.begin(), space.end());
59         cout << space[ans - 1]; // 가장 넓은 그림의 넓이
60     }
61     return 0;
62 }

```

```

1 #include <iostream>
2 #include <algorithm>
3 #include <vector>
4
5 using namespace std;
6 int n, m, ans, cnt; //행, 열, 그림 개수, 그림 넓이
7 int matrix[501][501]; // 0 : 그림 없음, 1 : 그림 있음, 2 : 그림 방문
8 int dx[4] = {-1, 0, 1, 0};
9 int dy[4] = {0, 1, 0, -1};
10 vector<int> space; //그림들의 넓이 보관
11
12 void dfs(int x, int y){
13     matrix[x][y]=2; //방문 체크
14     cnt++; //현재 그림 넓이 계산
15
16     for(int i=0; i<4; i++){
17         int nx=x+dx[i], ny=y+dy[i];
18         if(nx<0 || ny<0 || nx>=n || ny>=m) continue;
19         if(matrix[nx][ny]==1){ //그림 있고 아직 방문 전이면
20             dfs(nx, ny);
21         }
22     }
23 }

```

예제 입력 1 복사

```

24 int main(){
25     ios_base::sync_with_stdio(false)
26
27     cin>>n>>m;
28     for(int i=0; i<n; i++){
29         for(int j=0; j<m; j++){
30             cin>>matrix[i][j];
31         }
32     }
33
34     for(int i=0; i<n; i++){
35         for(int j=0; j<m; j++){
36             if(matrix[i][j]==1){ //그림 있고 아직 방문 전이면
37                 cnt=0; //현재 그림의 넓이 초기화
38                 dfs(i,j);
39                 space.push_back(cnt); //현재 그림 넓이 보관
40                 ans++; // 그림의 개수
41             }
42         }
43     }
44     cout<< ans <<'\n';
45
46     if(!ans) cout<<0; // 그림이 하나도 없는 경우에는 가장 넓은 그림의 넓이는 0
47     else {
48         sort(space.begin(), space.end());
49         cout << space[ans - 1]; // 가장 넓은 그림의 넓이
50     }
51     return 0;
52 }

```

예제 출력 1 복사

```

6 5
1 1 0 1 1
0 1 1 0 0
0 0 0 0 0
1 0 1 1 1
0 0 1 1 1
0 0 1 1 1
4
9

```

감사합니다

- 필수 문제

2 1260번

DFS와 BFS

3 2606번

바이러스

- 연습 문제

2 11724번

연결 요소의 개수

2 18352번

특정 거리의 도시 찾기

1 1325번

효율적인 해킹

1 1926번

그림

2 1012번

유기농 배추

1 2667번

단지번호붙이기