#9주차 고급반

FLOW ALOGRITHM 2

T. 강건

수업의 목표

1. Dinic 알고리즘을 이해하고 구현할 수 있다.
2. MCMF와 관련된 문제들을 이해하고 해결할 수 있다.

Dinic Algorithm

어려워...

최대 유량 문제_용어정리

유량 네트워크(flow network) : 간선의 가중치가 “용량”인 단방향 그래프

소스(source) : 유량이 시작하는 정점.

싱크(sink) : 유량이 도착하는 정점.

용량(capacity) : 각 간선에 대하여 흐를 수 있는 최대 유량(용량)

유량(flow) : 각 간선에 대하여 실제로 흐른 유량

최대 유량 문제_용량의 제한

유량(flow)는 그 간선의 용량을 초과할 수 없다.

flow <= capacity

$$f_{uv} \leq c_{uv}$$

최대 유량 문제_여유용량

여유 용량 : 흐를 수 있는 상태

flow < capacity

$$f_{uv} < c_{uv}$$

최대 유량 문제_ 유량의 보존

소스와 싱크를 제외한 정점에는 들어오는 유량과 나가는 유량이 같다.

u에 들어오는 유량 = u에서 나가는 유량

$$\sum_{v:(u,v)\in E} f_{vu} = \sum_{v:(u,v)\in E} f_{uv}$$

(v는 u와 연결된 정점)

최대 유량 문제_유량의 대칭성

u->v로 f만큼의 유량이 흐르면 v->u로 −f만큼의 유량이 흐른다

$$f_{uv} = -f_{vu}$$

최대 유량 문제_최대유량

유량 네트워크의 최대유량은 소스에서 싱크로 흘러가는 유량의 크기이다.

소스에서 나오는 유량 = 싱크로 흘러 들어가는 유량

$$F = \sum_{u:(s,u)\in E} f_{su} = \sum_{v:(v,t)\in E} f_{vt}$$

s:source, t:sink

이걸 구하는게 문제의 답입니다.

Dinic_level

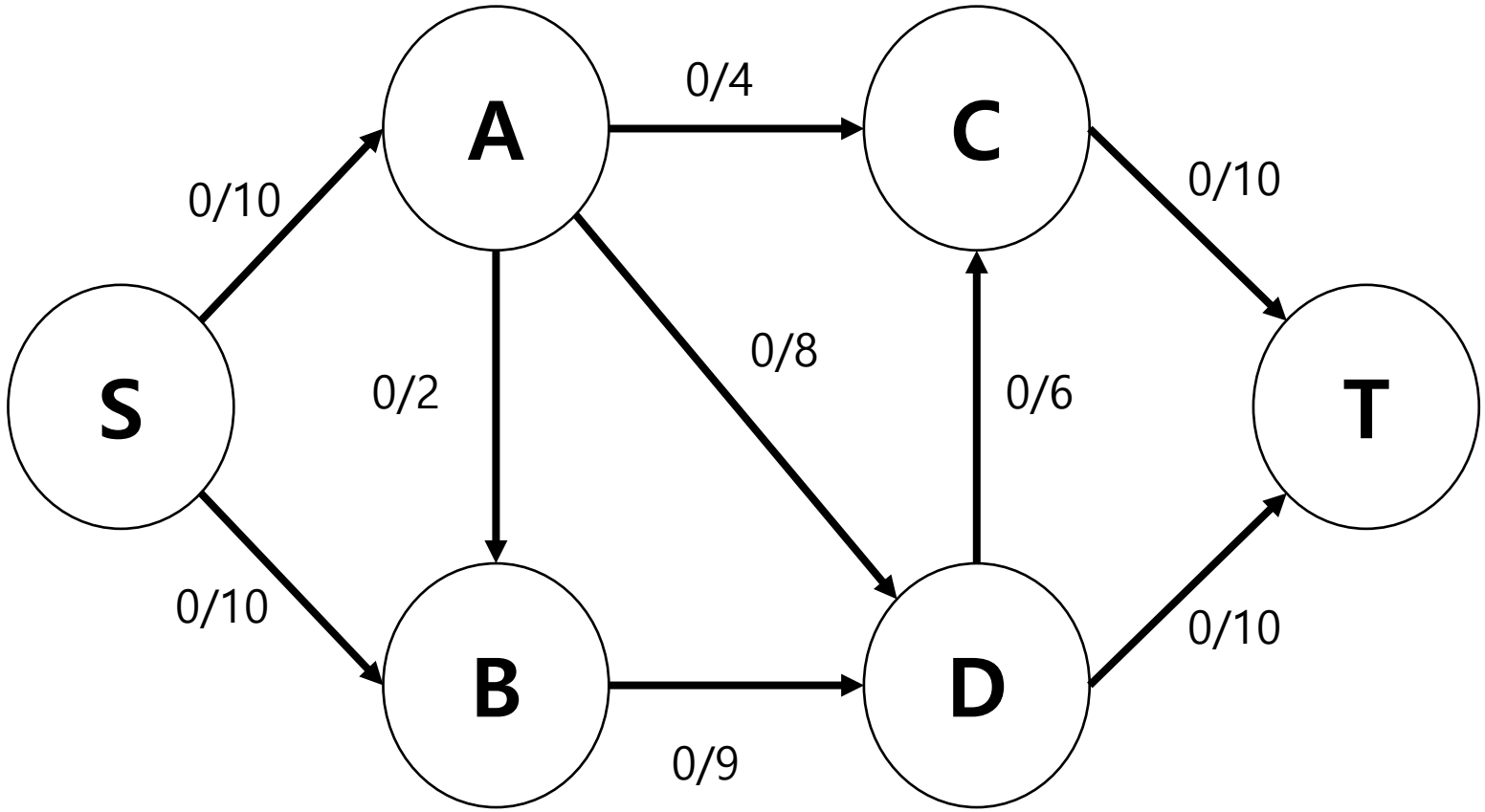
Level :
Source에서부터 한 정점까지 몇 개의 간선을 거쳐서 갈 수 있는지(최단거리)를 나타낸 값(여유 용량이 없는 간선은 제외)

Level Graph :
각 정점들에 level을 매긴 그래프

Dinic_level

이런 그래프가 있습니다.

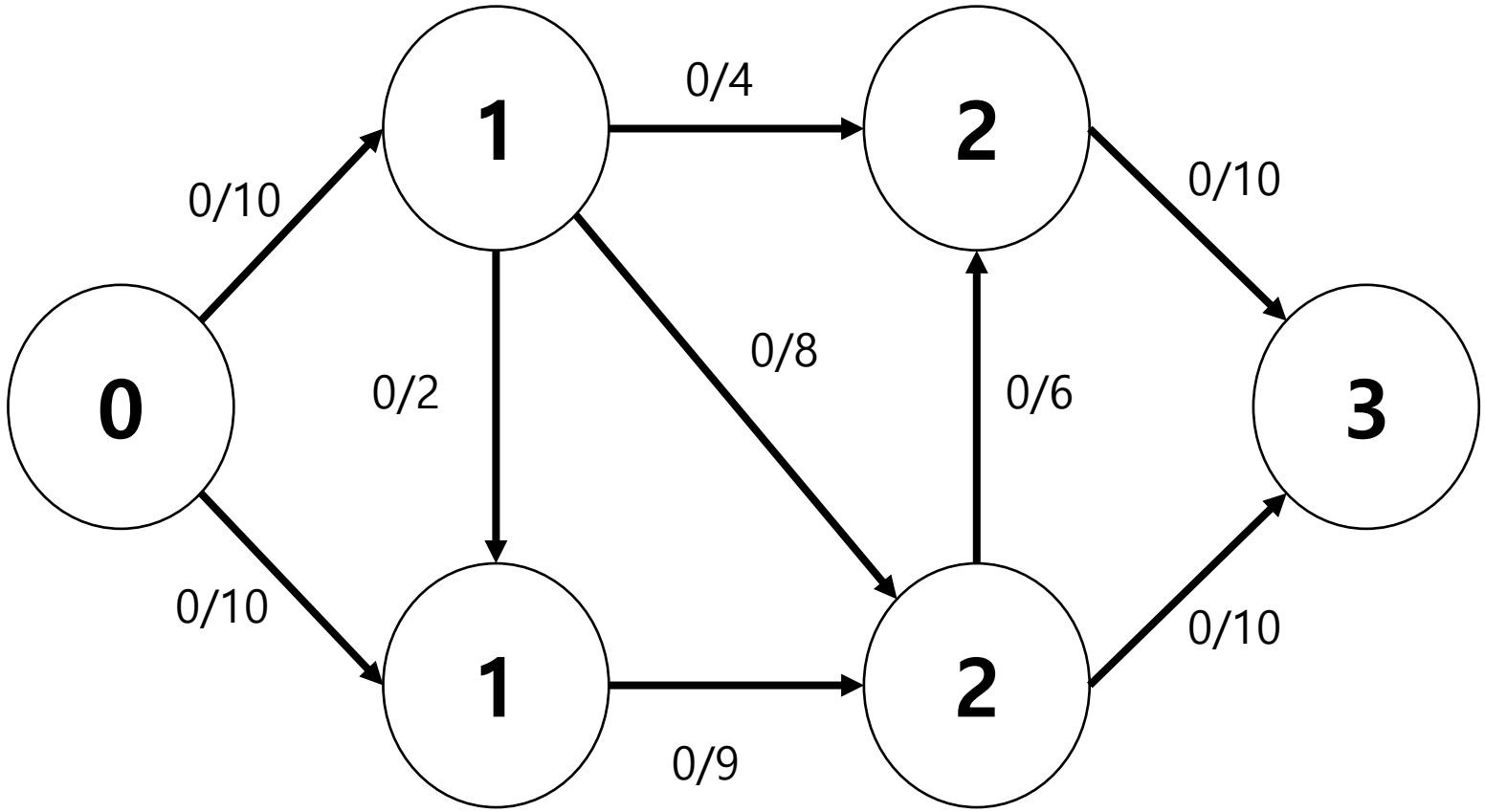
레벨을 한번 붙여 볼게요.



Dinic_level

참 쉽죠??

bfs하면 간단히 끝나요.



Dinic_level

Level Graph에서는

간선 uv에서 v가 u의 레벨보다 1 클 때만 이동 가능하다.

즉 자신과 인접하면서 자신보다 레벨이 1 큰 정점으로만 흐를 수 있다.

blcoking flow(차단유량) :

이런 상태에서 더 이상 소스에서 링크로 흘릴 수 있는 유량이 없게 만드는 유량 상태(최대유량)

Dinic Algorithm

이제 진짜 시작

Ford Fulkerson Algorithm_개념

1. 증가경로를 찾는다.

2. 증가 경로로 흘릴 수 있는 만큼의 유량을 흘려준다
   -> 경로 상의 모든 f(u,v)를 증가시킨다

3. f(u,v)를 증가시킨 만큼 **역방향 간선**에 감소시킨다.

4. 더 이상 흐를 수 없을 때까지(증가 경로가 없을 때가지) 위의 과정을 반복한다.

디닉 아님!!!

Dinic Algorithm_개념

1. 레벨 그래프를 만든다.

2. 차단유량을 찾아 그만큼 흘려준다.

3. 더 이상 흐를 수 없을 때까지 위의 과정을 반복한다.

말은 참 쉬워요.

~~원래 말 쉬운게 더 어려운데~~

Dinic Algorithm_개념

1. 레벨 그래프를 만든다.

2. 차단유량을 찾아 그만큼 흘려준다.

3. 더 이상 흐를 수 없을 때까지 위의 과정을 반복한다.

그래 이건 그냥 bfs니까 쉽지

쉽지??

Dinic Algorithm_개념

1. 레벨 그래프를 만든다.

2. 차단유량을 찾아 그만큼 흘려준다.

3. 더 이상 흐를 수 없을 때까지 위의 과정을 반복한다.

이건 어떻게 할까요???

Dinic Algorithm_개념

Ford Fulkerson Algorithm_개념

1. 증가경로를 찾는다.

2. 증가 경로로 흘릴 수 있는 만큼의 유량을 흘려준다
   -> 경로 상의 모든 f(u,v)를 증가시킨다

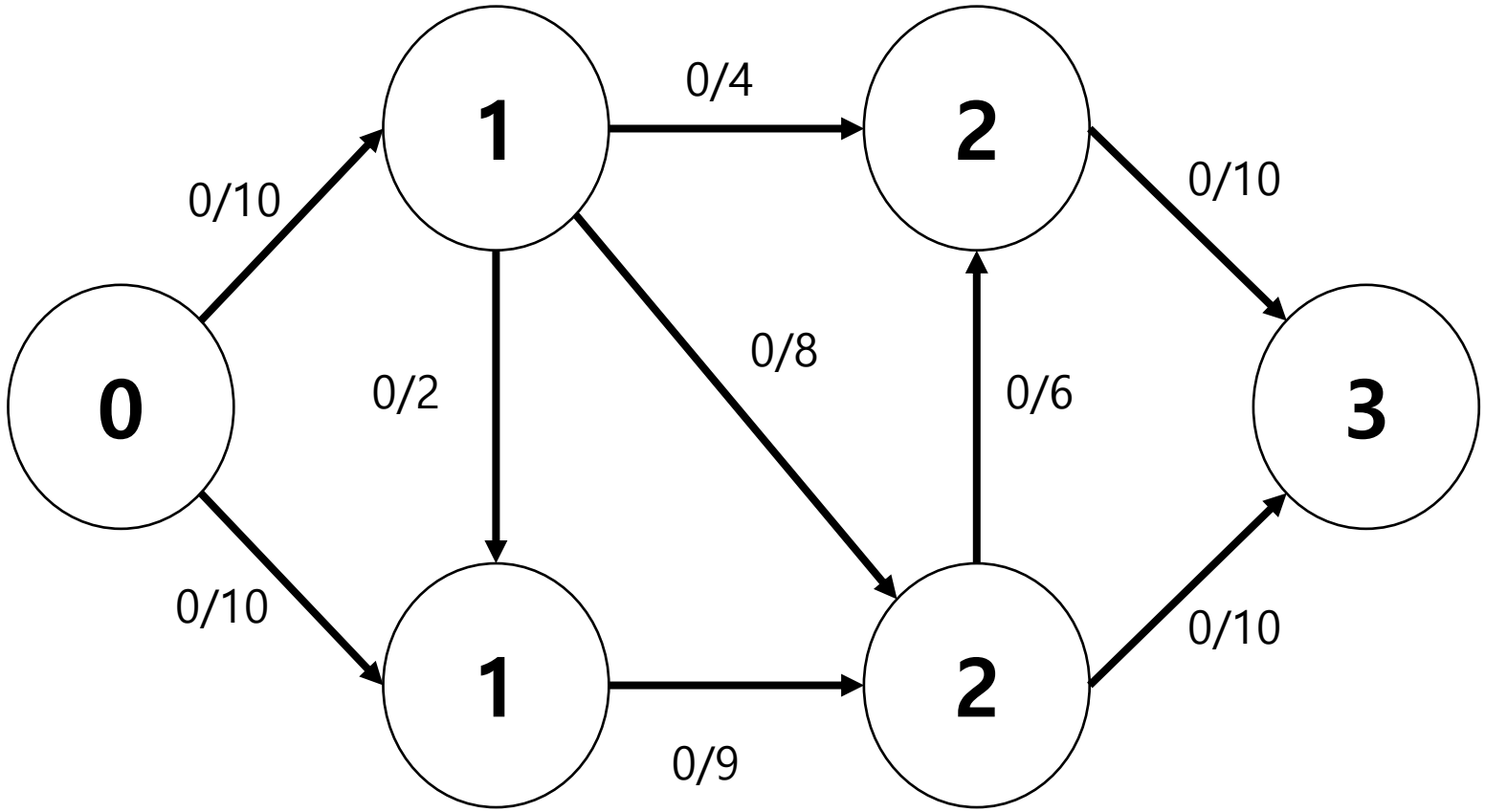
3. f(u,v)를 증가시킨 만큼 **역방향 간선**에 감소시킨다.

4. 더 이상 흐를 수 없을 때까지(증가 경로가 없을 때가지) 위의 과정을 반복한다.

여기 있네요???

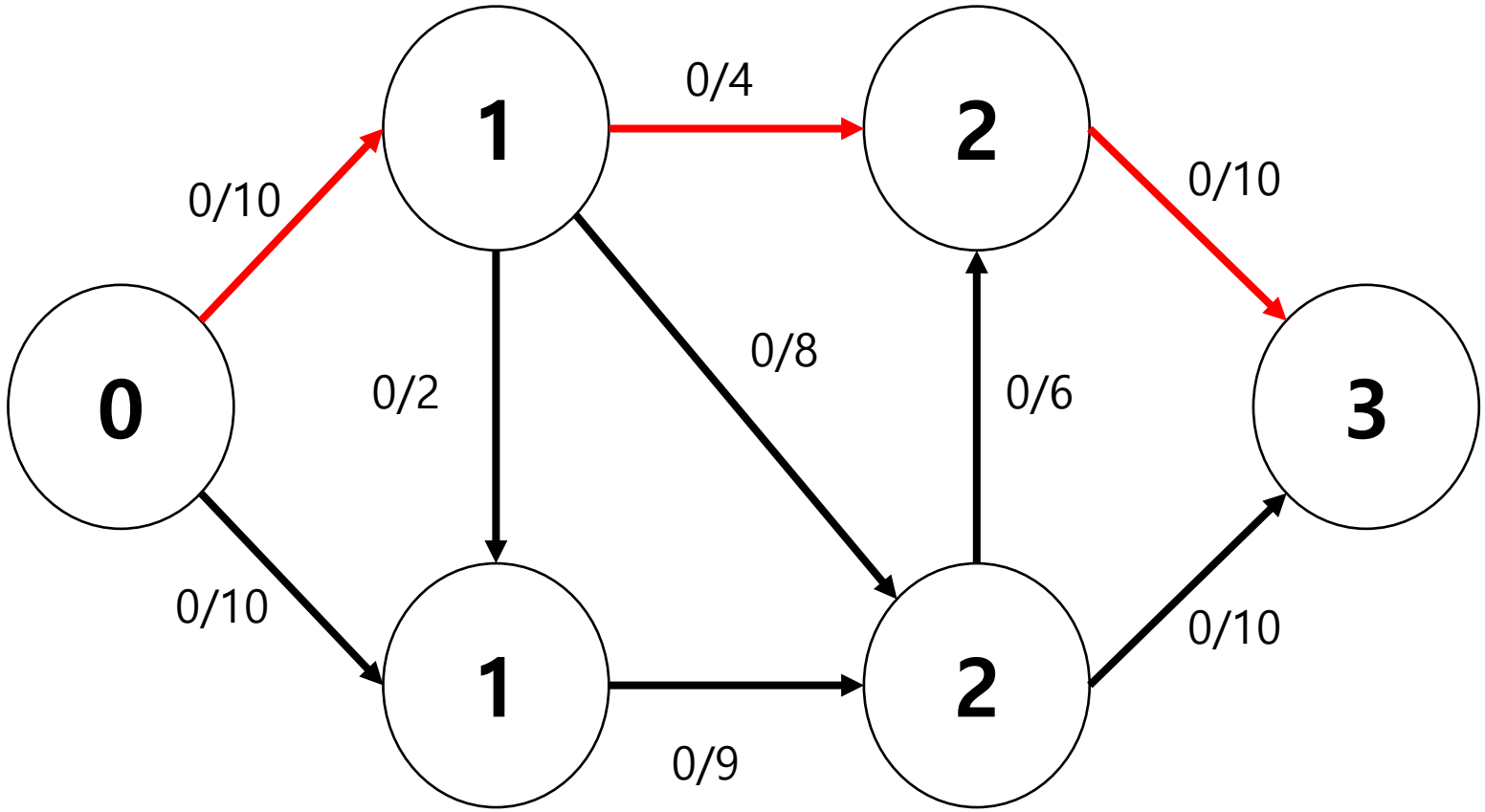
Dinic_예시

자 이렇게 level graph를 그렸어요.



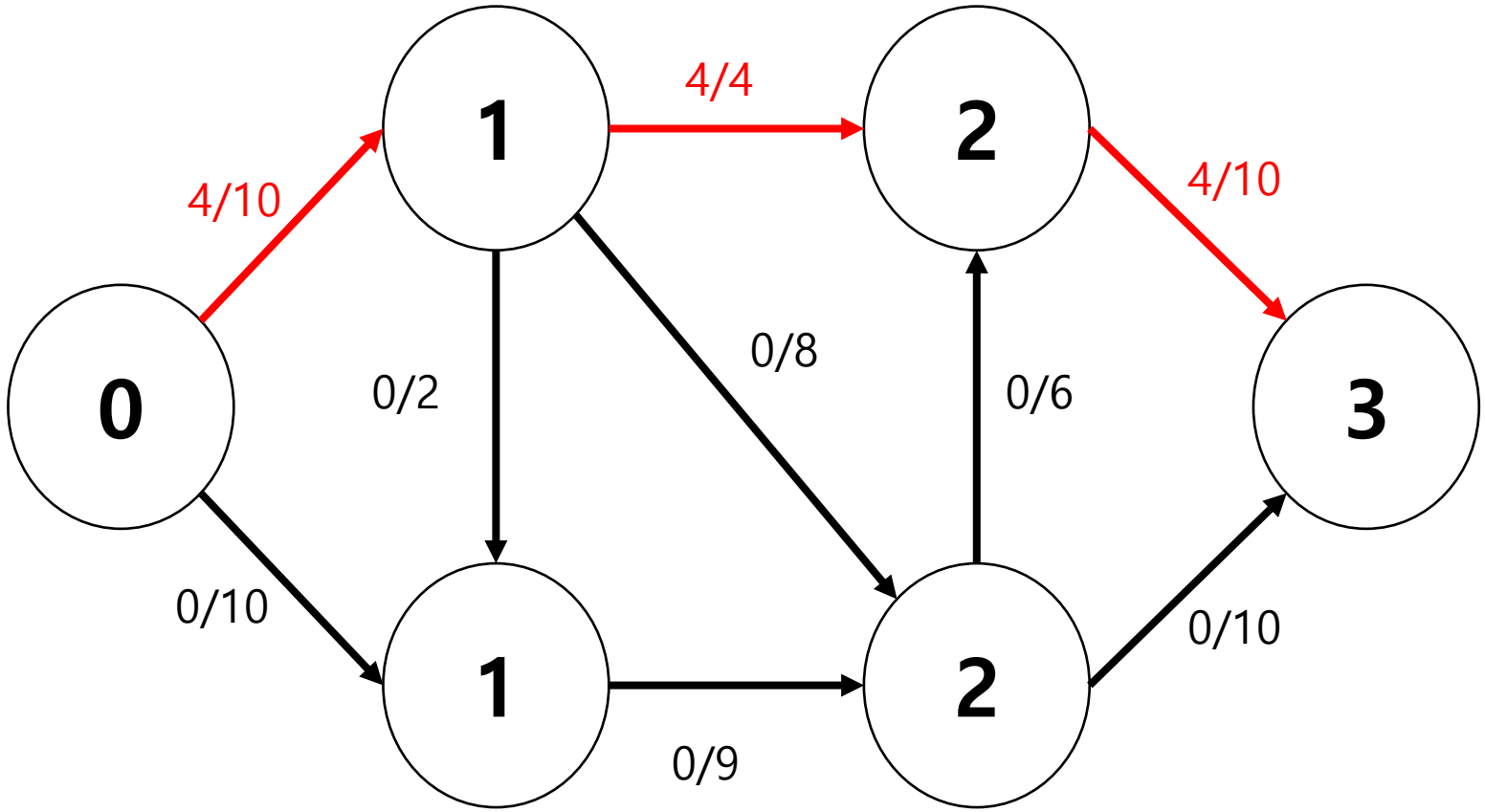
Dinic_예시

여기 증가 경로가 있네요.



Dinic_예시

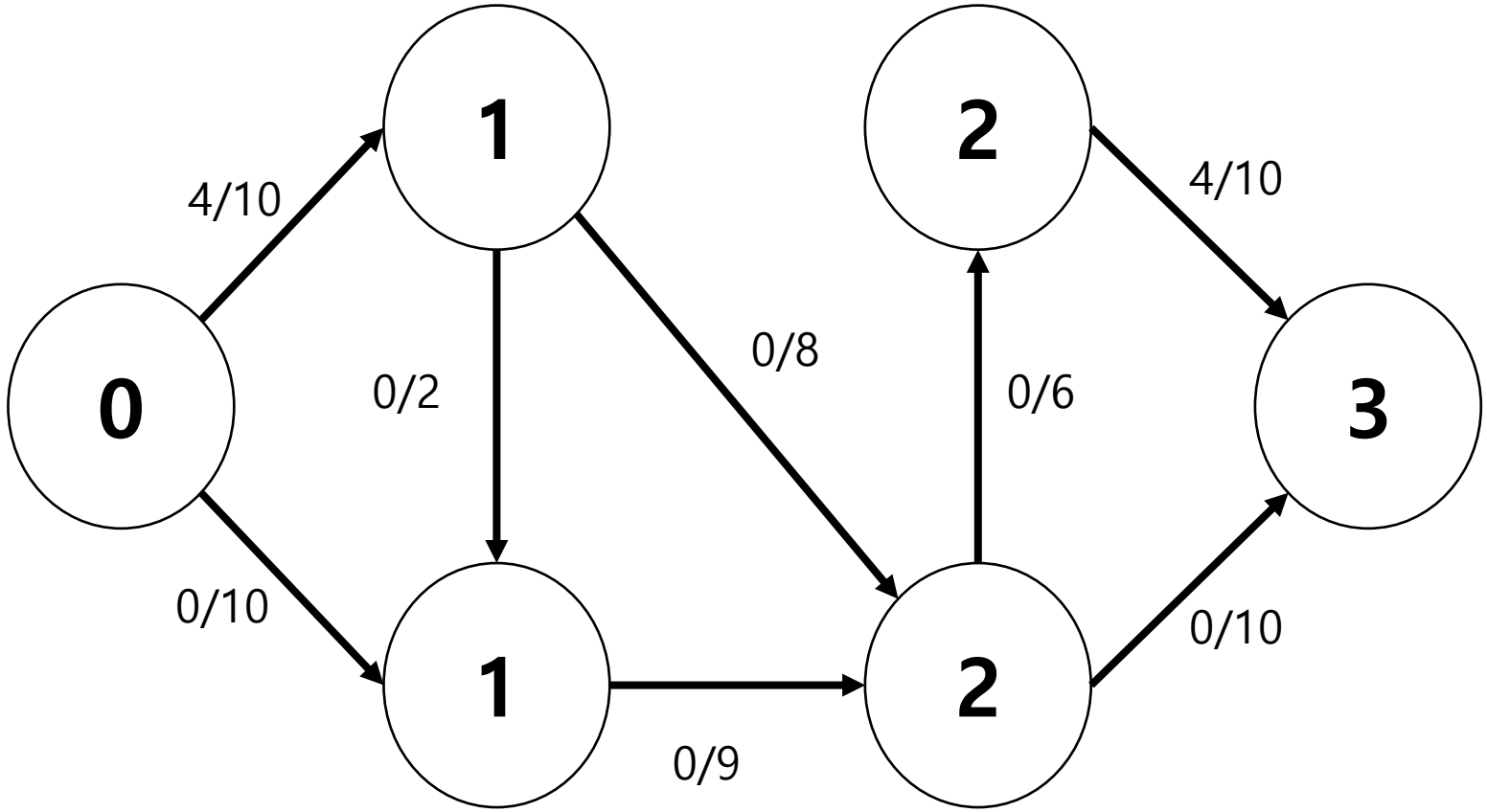
갱신 해줍시다.



Dinic_예시

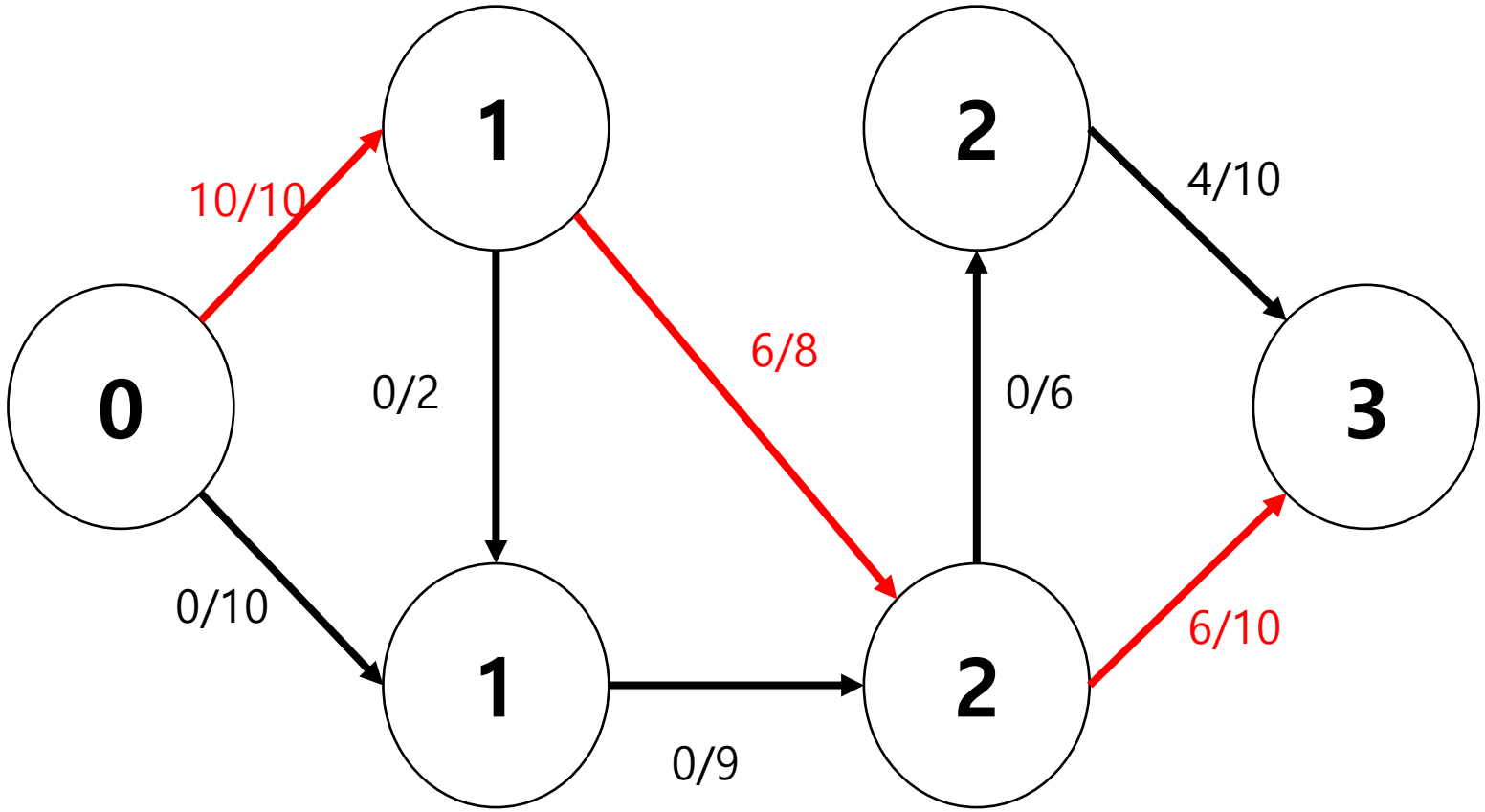
역방향 간선은 생략할게요.~~(안해도 다 알잖아)~~

여유 용량이 없는 간선도 지우도록 할게요.

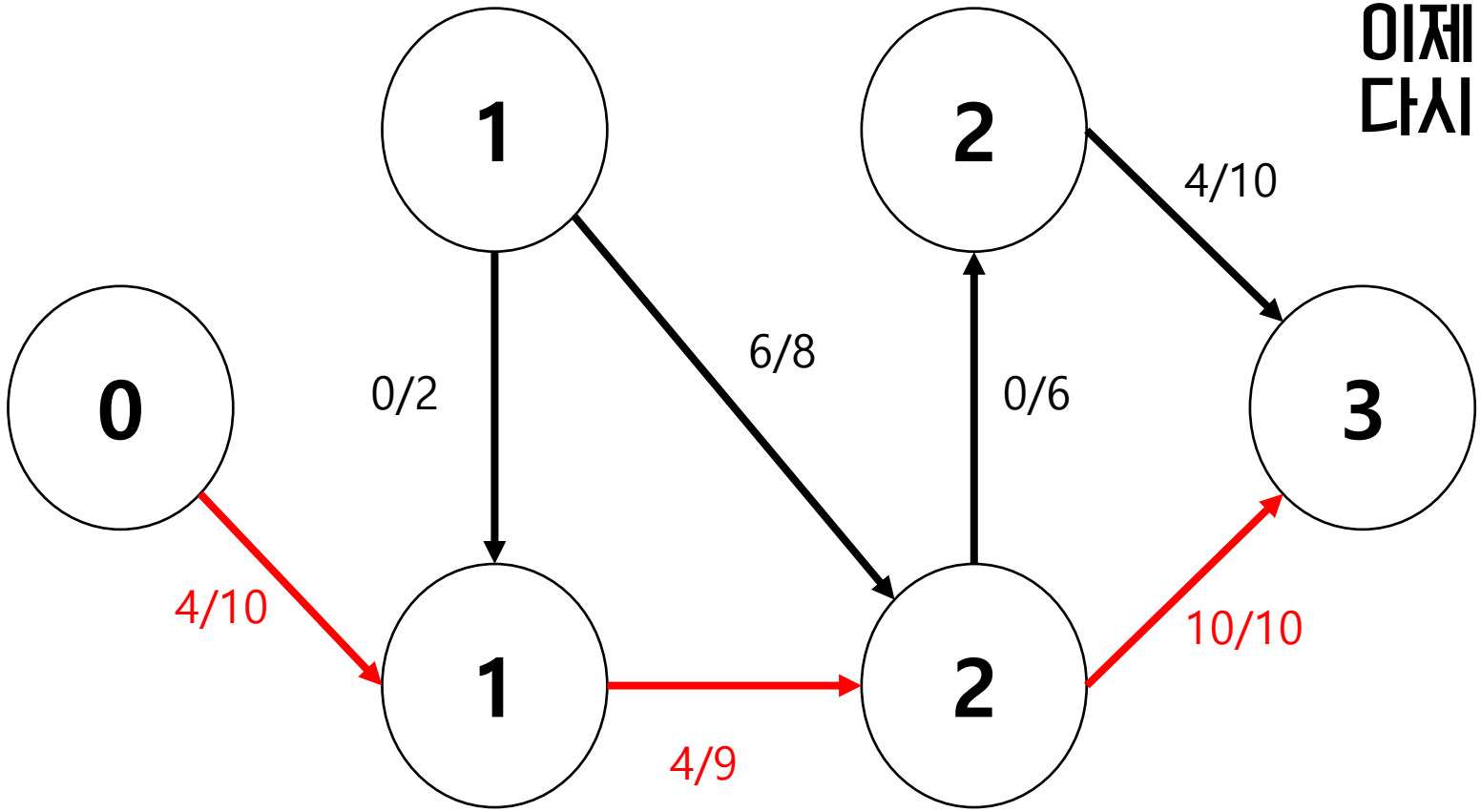


Dinic_예시

여기도 됐고



Dinic_예시



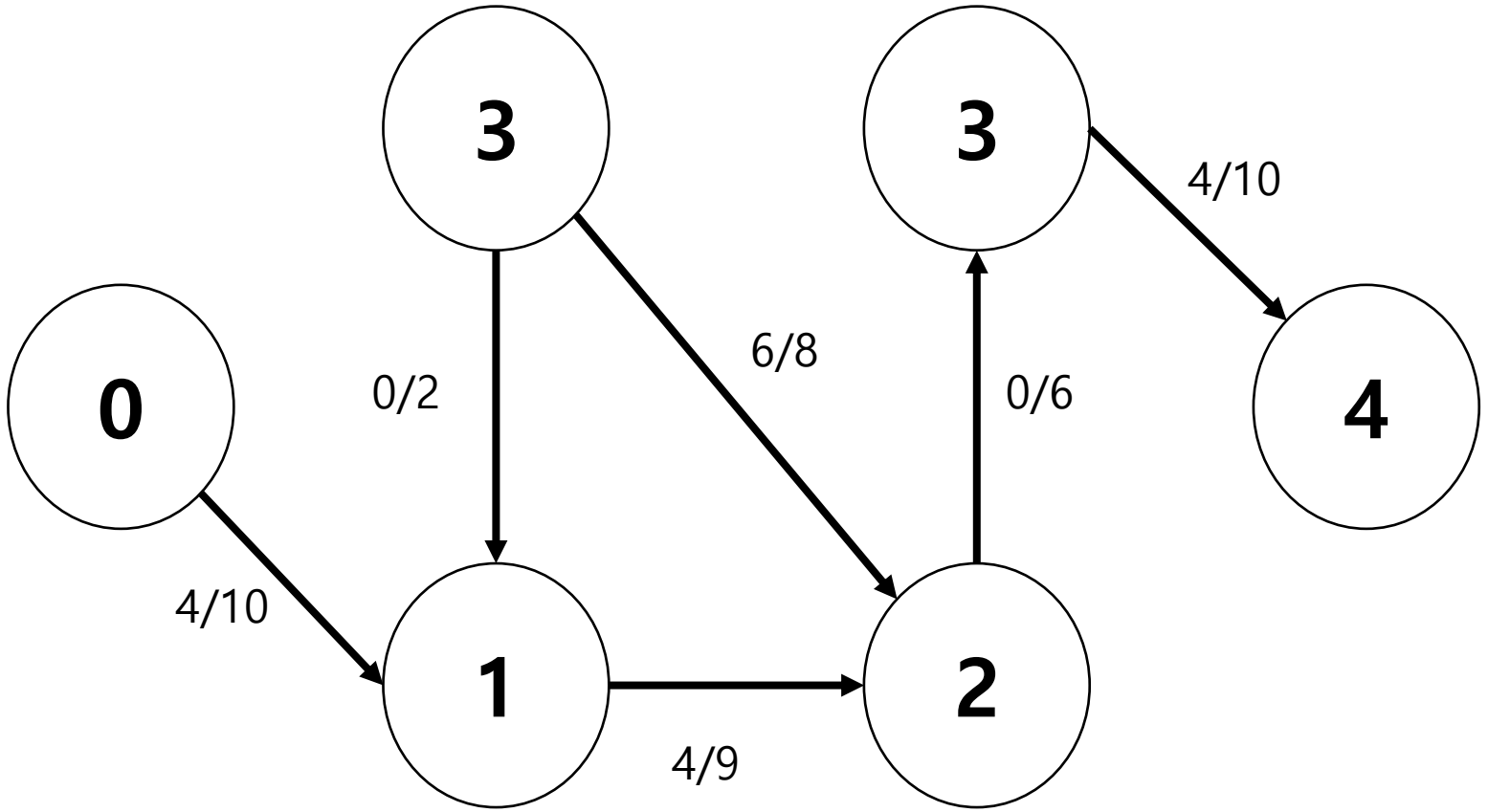
여기까지 이해 되시죠?

지난주를 잘 했으면 쉬울거에요.

이제 더 이상 흐를 수 없으니
다시 level graph를 그려야 해요.

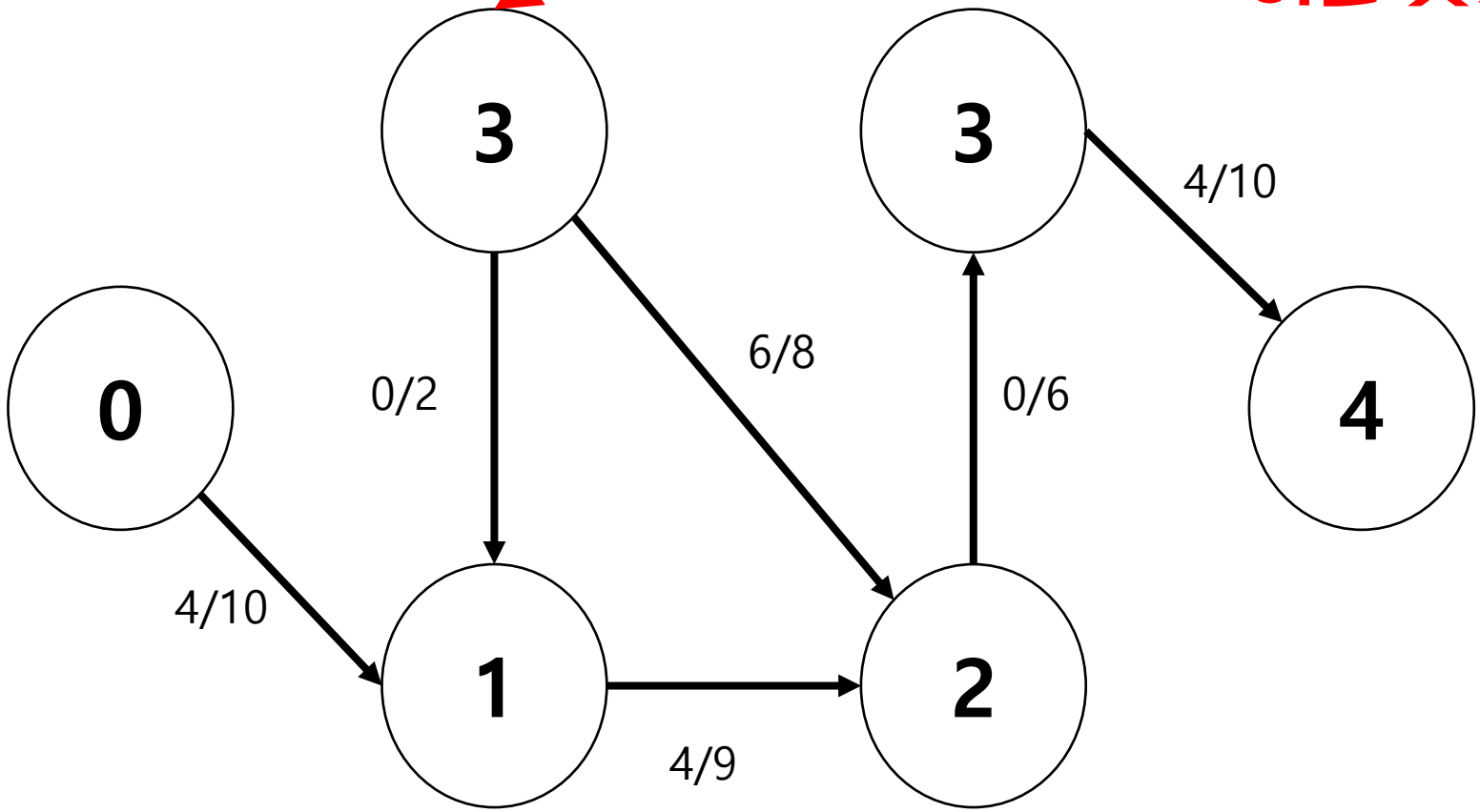
Dinic_예시

자 다시 그렸습니다.

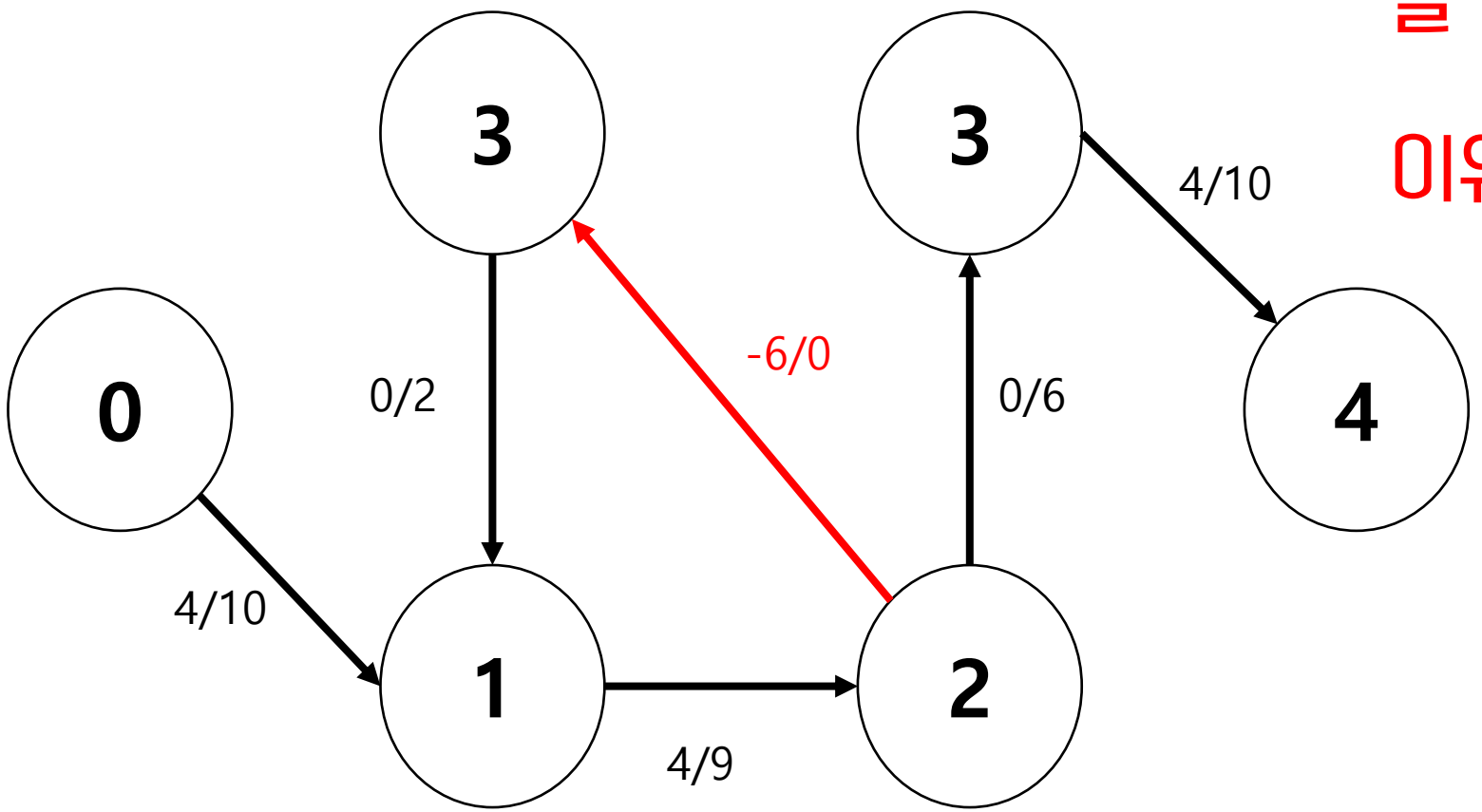


Dinic_예시

아니 얘가 왜 3이냐고요!!!
여긴 못가는데???



Dinic_예시

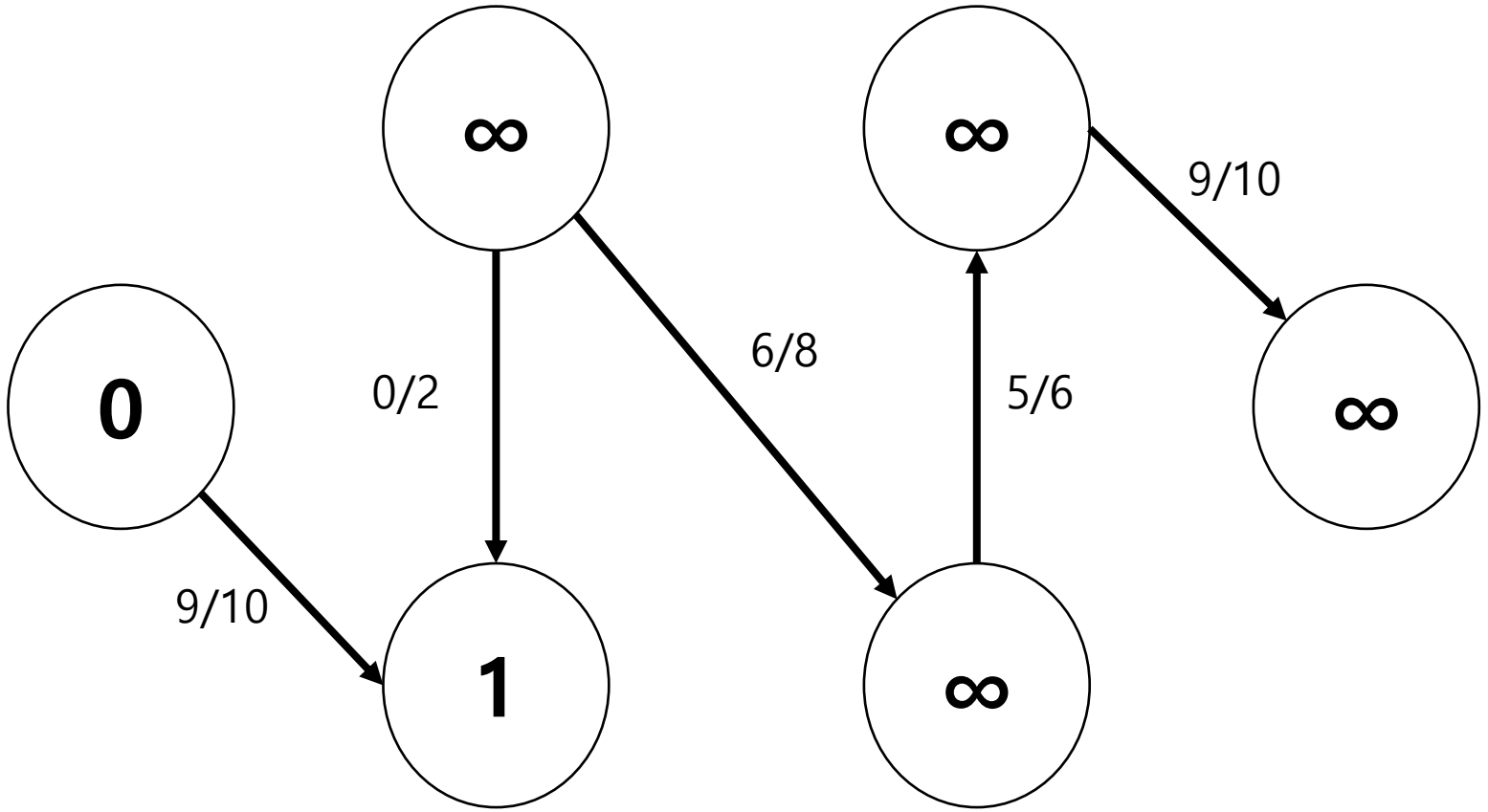


갈 수 있습니다.

이유는 잘 생각해 보세요.

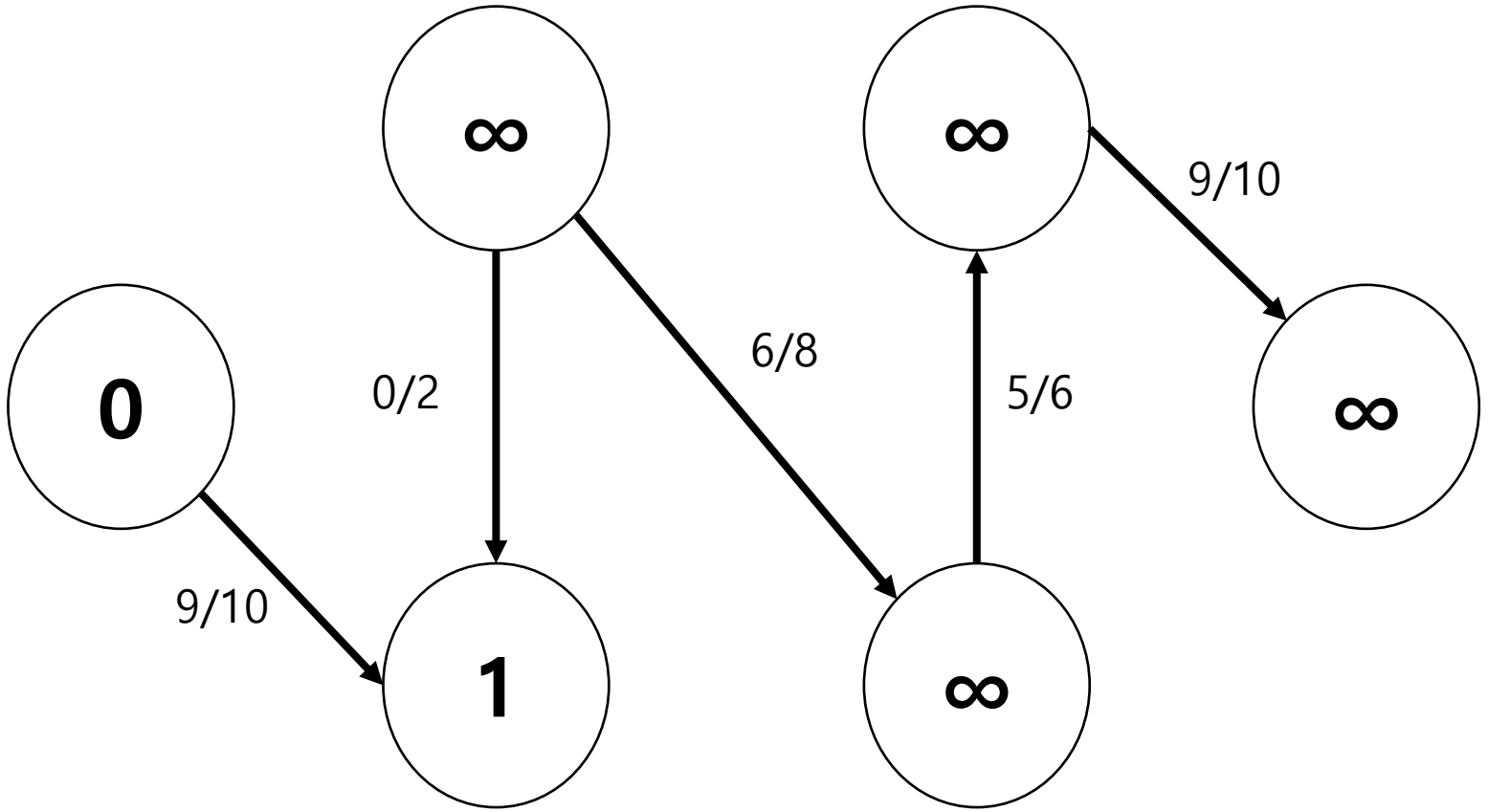
Dinic_예시

이제 더 이상 흐를 수 없네요.



Dinic_예시

이제 더 이상 흐를 수 없네요.



Dinic

시간 복잡도 : $O(V^2E)$

이걸 설명할라면 길어지니까... 각자 찾아보는걸로!

어려우니까 알고만 있으면 됩니다. 디닉이 무조건 빨라요!

~~아 비대면 수업하고싶다.~~

예제

6086 – 최대 유량

Dinic으로 풀어봅시다.

6086_최대유량

- 다음 장의 코드를 보면서 풀어보아요.

~~자신이 있다면 넘기지 마시오.~~

6086_최대유량_자료구조

Ford-Fulkerson에서 수정한다고 생각하시면 됩니다!

```
31      int N, source, sink;          // 정점의 갯수, 소스, 싱크
32      vector<vector<Edge*>> Graph;
33      // 주어진 그래프; Graph[i] : i에서 출발하는 Edge들, 인접리스트 형식이라고 생각하면 됨.
34      vector<int> level;
35      queue<int> bfs_queue;
36
37      MaximumFlow(void) :
38          N(0), source(0), sink(0), Graph(), level(), bfs_queue()
39      {
40      }
41      MaximumFlow(const MaximumFlow& mf) :
42          N(mf.N), source(mf.source), sink(mf.sink), Graph(mf.Graph), level(mf.level), bfs_queue(mf.bfs_queue)
43      {
44      }
45      MaximumFlow(int N, int source, int sink) :
46          N(N), source(source), sink(sink), Graph(N), level(N), bfs_queue()
47      {
48          // class말고 struct로 하면 이 생성자만 있으면 됩니다.
49      }
```

6086_최대유량_자료구조

Level graph그리는 부분 추가해 주시고

```
62      bool bfs(void)
63      {
64          fill(level.begin(), level.end(), -1); // -1 : 무한대
65          level[source] = 0;
66          bfs_queue.push(source);
67
68          while ( !bfs_queue.empty() )
69          {
70              int now = bfs_queue.front();
71              bfs_queue.pop();
72
73              for ( auto e : Graph[now] )
74              {
75                  if ( level[e->to] == -1 && e->capacity - e->flow > 0 )
76                  {
77                      level[e->to] = level[now] + 1;
78                      bfs_queue.push(e->to);
79                  }
80              }
81          }
82
83          return level[sink] != -1;
84      }
```

6086_최대유량_자료구조

Ford Fulkerson이랑 뭐가 다를까요? 비교해보세요.

```
86      int dfs(int now, int mf)
87      {
88          // now : 현재 정점
89          // mf : 현재까지 흐를 수 있는 최대 유량
90          if ( now == sink )
91          { // sink까지 도달하면 흐를 수 있는 가장 큰 값을 리턴한다.
92              return mf;
93          }
94
95          for ( auto e : Graph[now] ) // 현재에서 출발하는 모든 간선에 대해
96          {
97              if ( level[e->to] == level[now] + 1 && e->capacity - e->flow > 0 ) // 흐를 수 있다면
98              {
99                  int flow = dfs(e->to, min(mf, e->capacity - e->flow));
100                 // dfs를 통해  흐를 수 있는 유량을 찾는다.
101                 if ( flow > 0 )                                    // 찾았다면
102                 {
103                     e->flow += flow;          // 정방향으로 흘려준다
104                     e->reverse->flow -= flow; // 역방향으로 흘려준다
105                     return flow;              // 얼마나 흘렀는지 리턴해준다.
106                 }
107             }
108         }
109
110         return 0;
111     }
```

6086_최대유량_자료구조

디닉 알고리즘 두줄설명 구현 부분입니다.

main 함수는 직접 해보세요 ㅎㅎ

```
113     int maximum_flow(void) // 진짜 답을 구하는 함수
114     {
115         int ret = 0;
116         // sink까지 증가 경로가 있을 때까지
117         while ( bfs() )
118         {
119             int flow = dfs(source, INT_MAX);
120 
121             if ( flow == 0 ) // 증가 경로를 못찾으면 break
122             {
123                 break;
124             }
125             ret += flow;
126         }
127         return ret;
128     }
129 };
```

예제

6086 – 최대 유량

잘 따라 왔다면 다들 “맞았습니다!!”를 받았을 거에요.
이제 스스로의 힘으로 해결해 볼까요?

어떠한 방법도 좋아요.
자신만의 스타일로 “맞았습니다!!”를 받아봅시다!

과제안내

- 알고리즘 설명이 빈약한 부분 인정합니다.
  하지만 강의자료만으로 설명하기엔 부족함이 많고 또 너무 tmi가 많아지면 좀 그래서..

- 과제 해설은 강사에게 요청하면 힌트 혹은 코드를 보내드립니다.

과제

11405 – 책 구매하기

과제

11377 – 열혈강호 3

과제

11376 – 열혈강호2

과제

1867 – 돌멩이 제거

과제

2414 – 게시판 구멍 막기

과제

1420 – 학교 가지마!

과제

9169 – 나는 999번 문제를 풀 수 있다

과제

13161 – 분단의 슬픔

1년동안 고생하셨습니다~

"이게 아마 끝일거야.."

고생하셨습니다!!

"나도 고생했어……."