네트워크 플로우

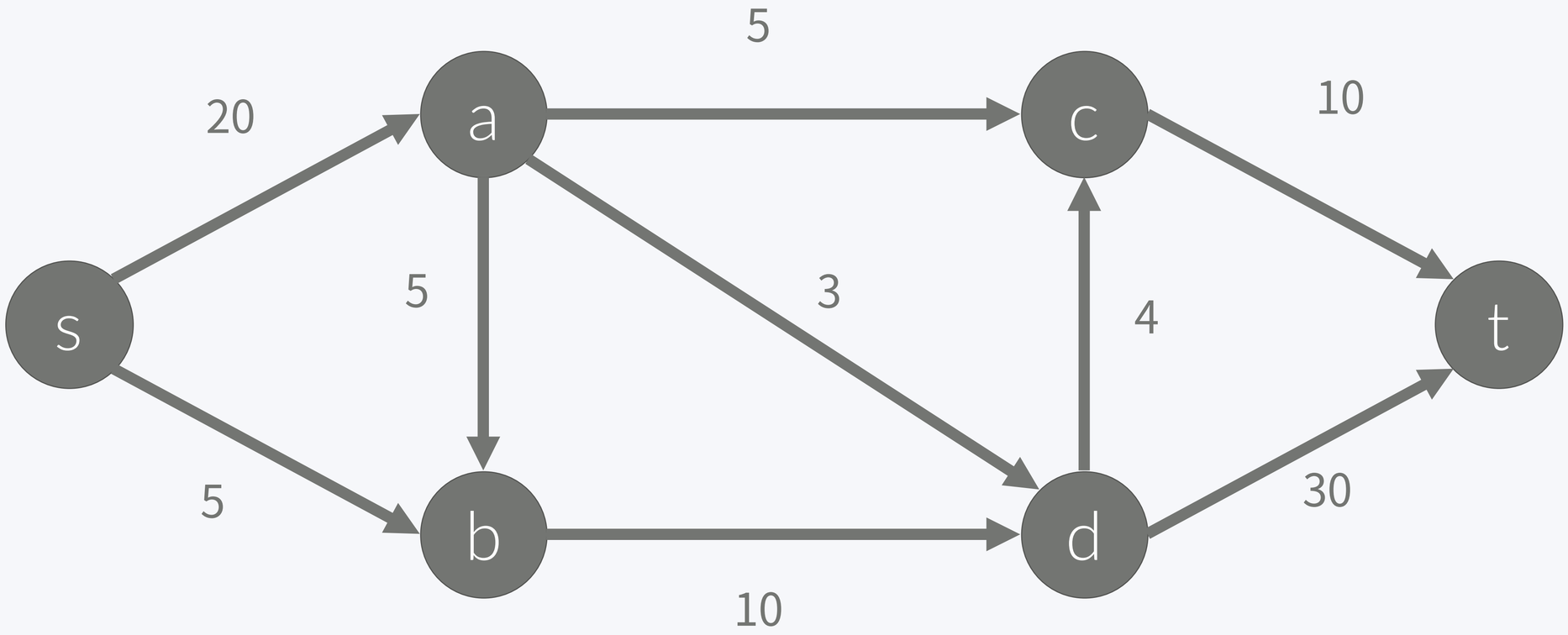
최백준 choi@startlink.io

최대 유량

최대 유량

Maximum Flow

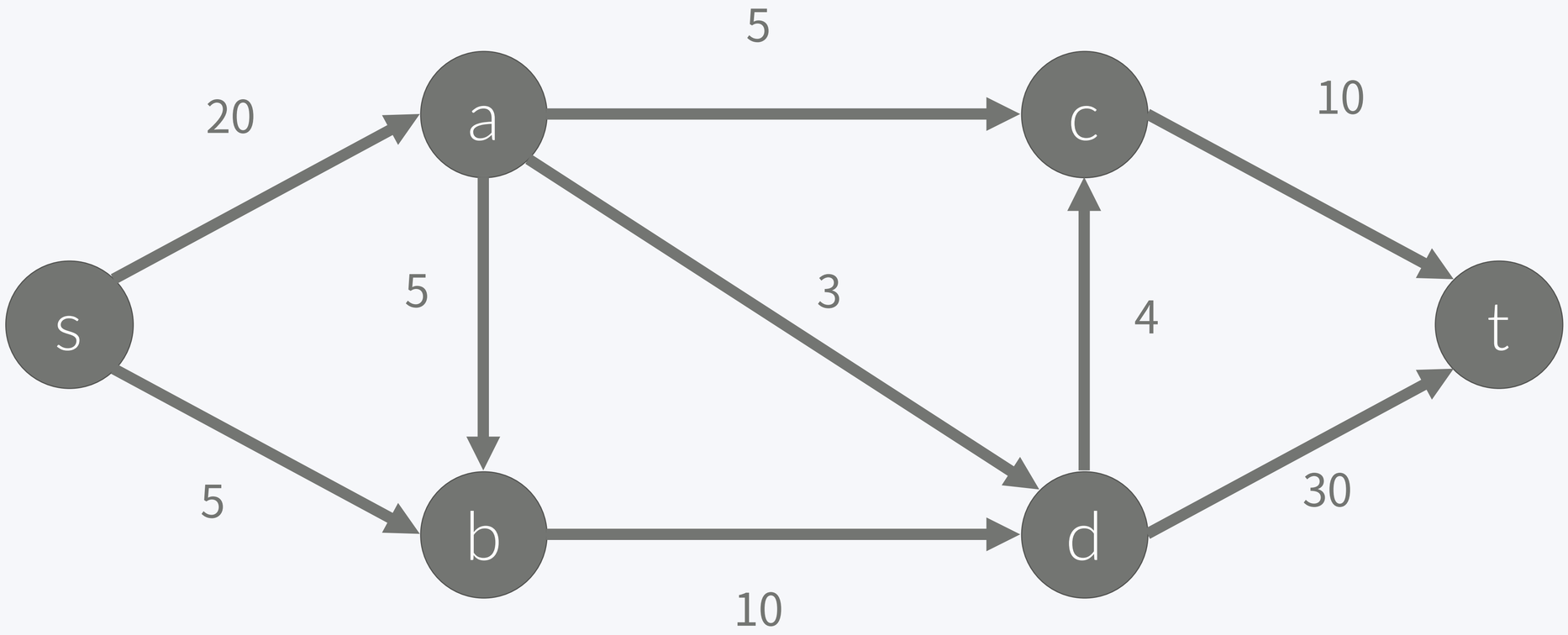
- 각 간선이 나타내는 것은 흐를 수 있는 양
- s에서 t로 최대 얼마나 흐를 수 있는가?



최대 유량

Maximum Flow

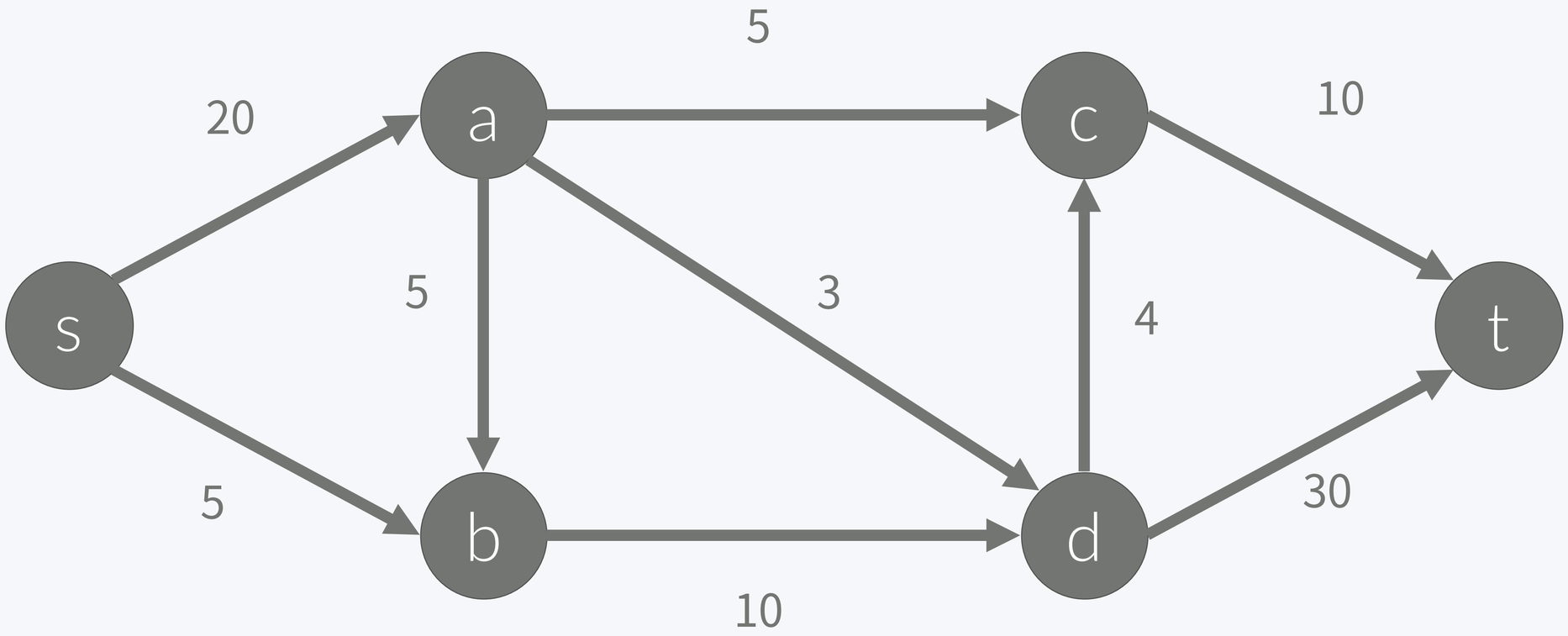
- s -> a로 최대 20만큼만 흐를 수 있다.
- a -> c로 최대 5만큼 흐를 수 있다.



최대 유량

Maximum Flow

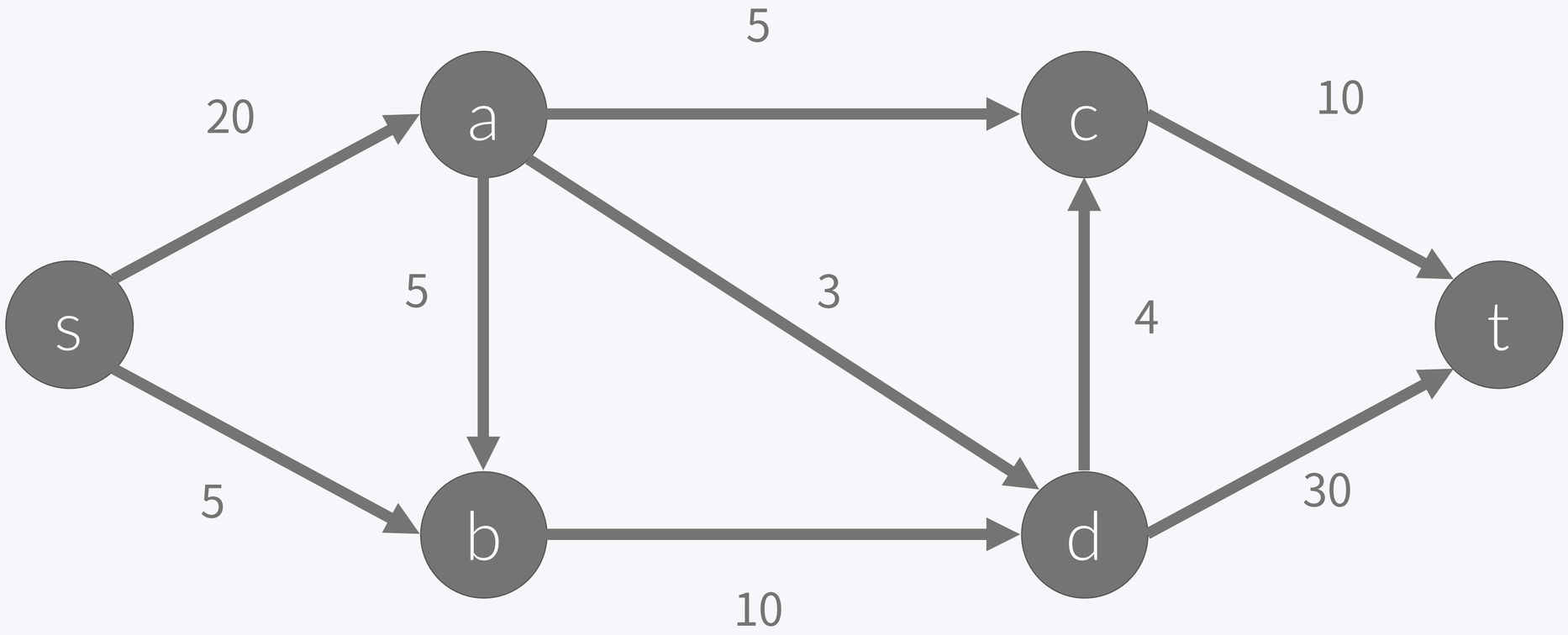
- s -> a로 20을 보냈다고 하더라도
- a -> c, a -> d, a -> b는 5+3+5 = 13이기 때문에, 13 이상을 보낼 수 없다



최대 유량

Maximum Flow

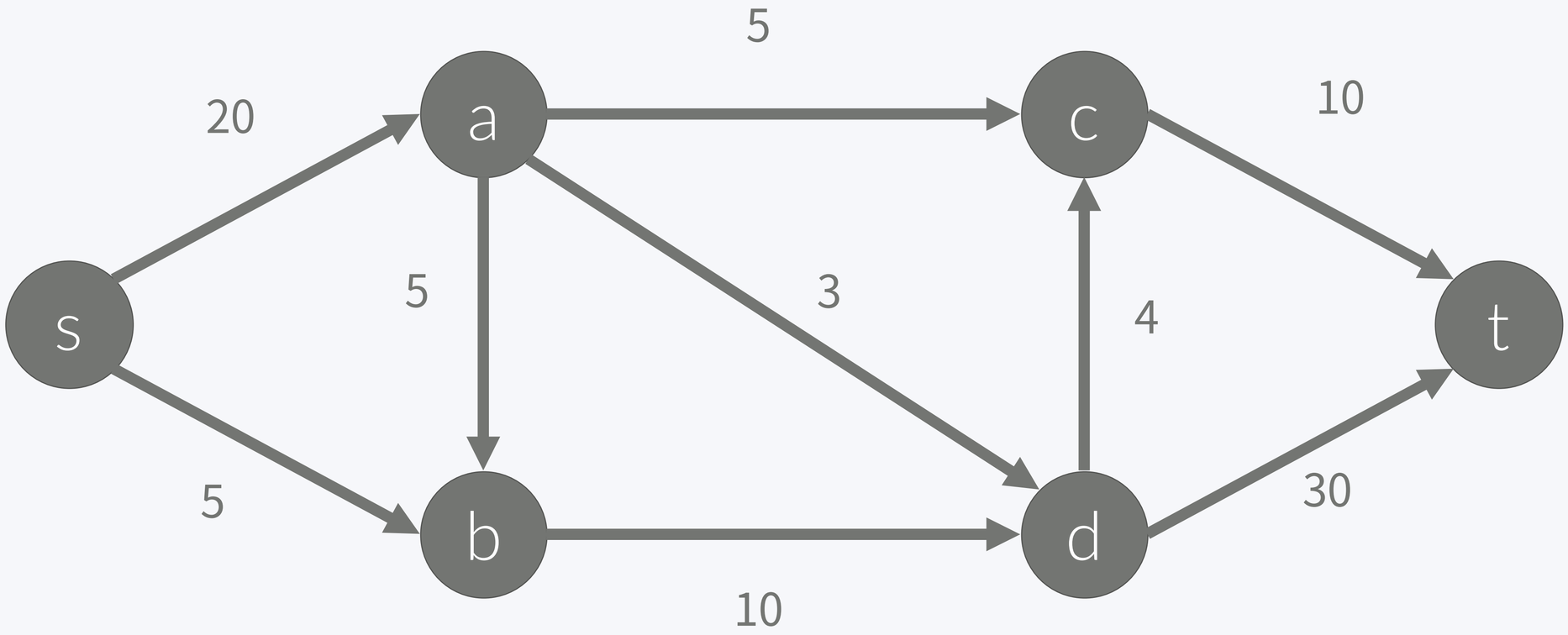
- s에서 t로 최대 얼마나 보낼 수 있는지를 구하는 문제



최대 유량

Maximum Flow

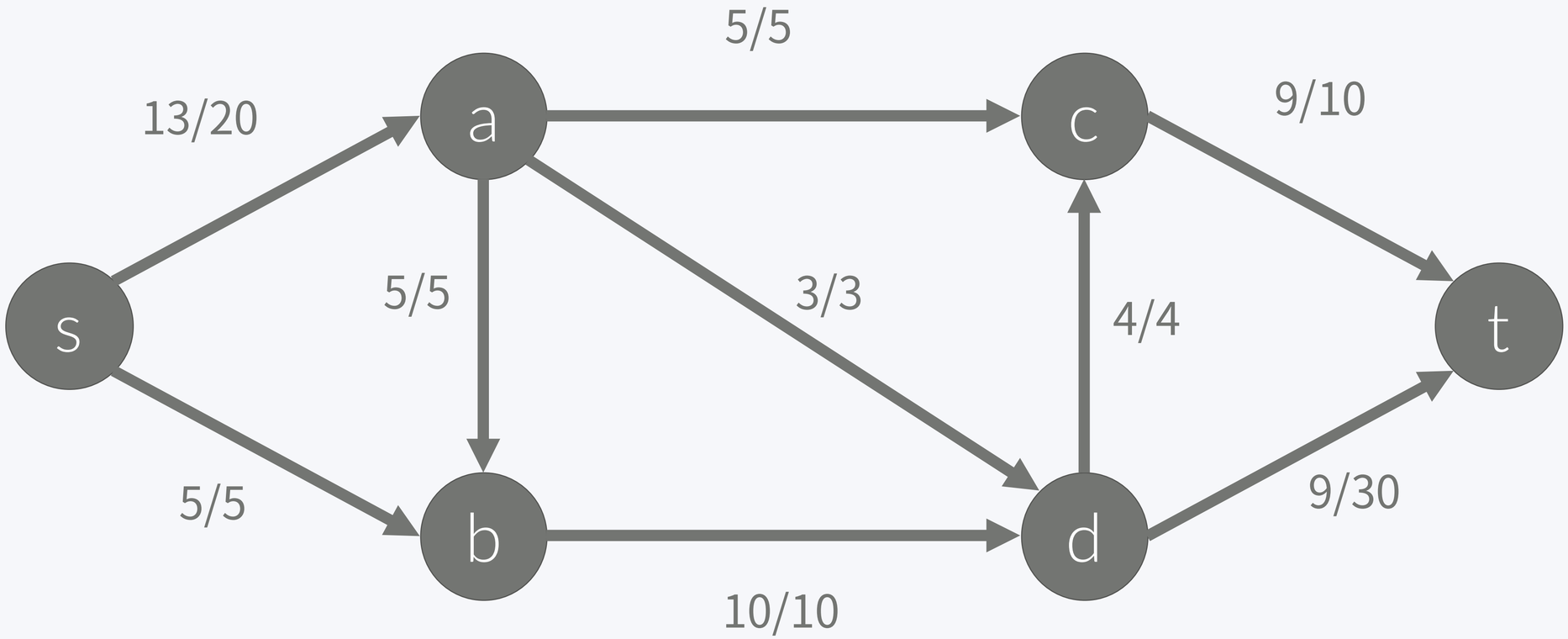
- 간선에 나타나있는 것: capacity
- 실제 그 간선을 따라서 흐른 양: flow



최대 유량

Maximum Flow

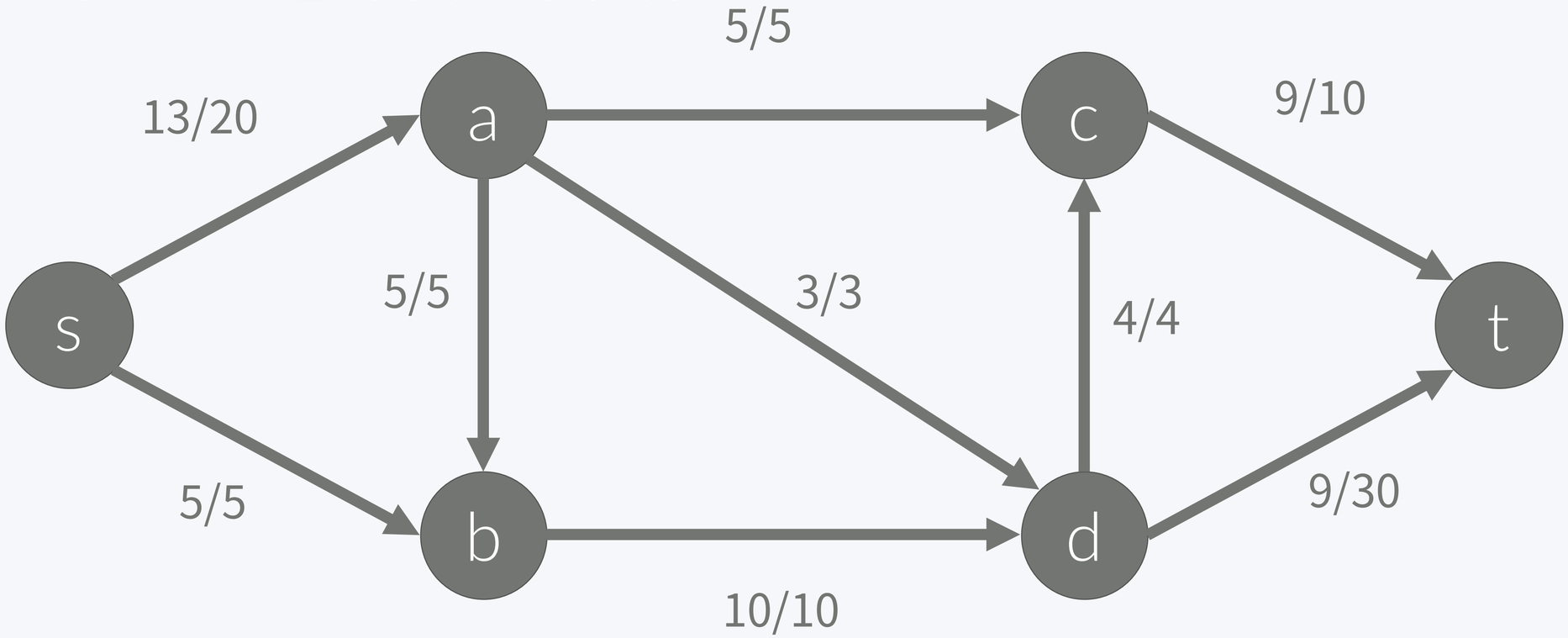
- flow/capacity



최대 유량

Maximum Flow

- 다른 예시 vertex: 교차로, edge: 도로
- capacity: 1분동안 그 도로를 지나갈 수 있는 차의 개수
- flow: 1분동안 그 도로를 지나가는 차의 개수



최대 유량

Maximum Flow

- 그래프 G에서
- u -> v 간선의 용량: c(u, v)
- u -> v 간선의 흐른 양: f(u, v)

최대 유량

Maximum Flow

- 세가지 속성을 만족해야 한다
- Capacity constraint
  - $f(u, v) \leq c(u, v)$
- Skew symmetry
  - $f(u, v) = -f(v, u)$
  - u->v로 x를 보냈으면, v->u로는 –x를 보낸다고 한다
- Flow conservation
  - $\Sigma v \in V f(u,v) = 0$
  - u와 연결된 모든 간선의 흐른양의 합은 0이다
  - Skew symmetry에 의해서 음수를 저장했기 때문

최대 유량

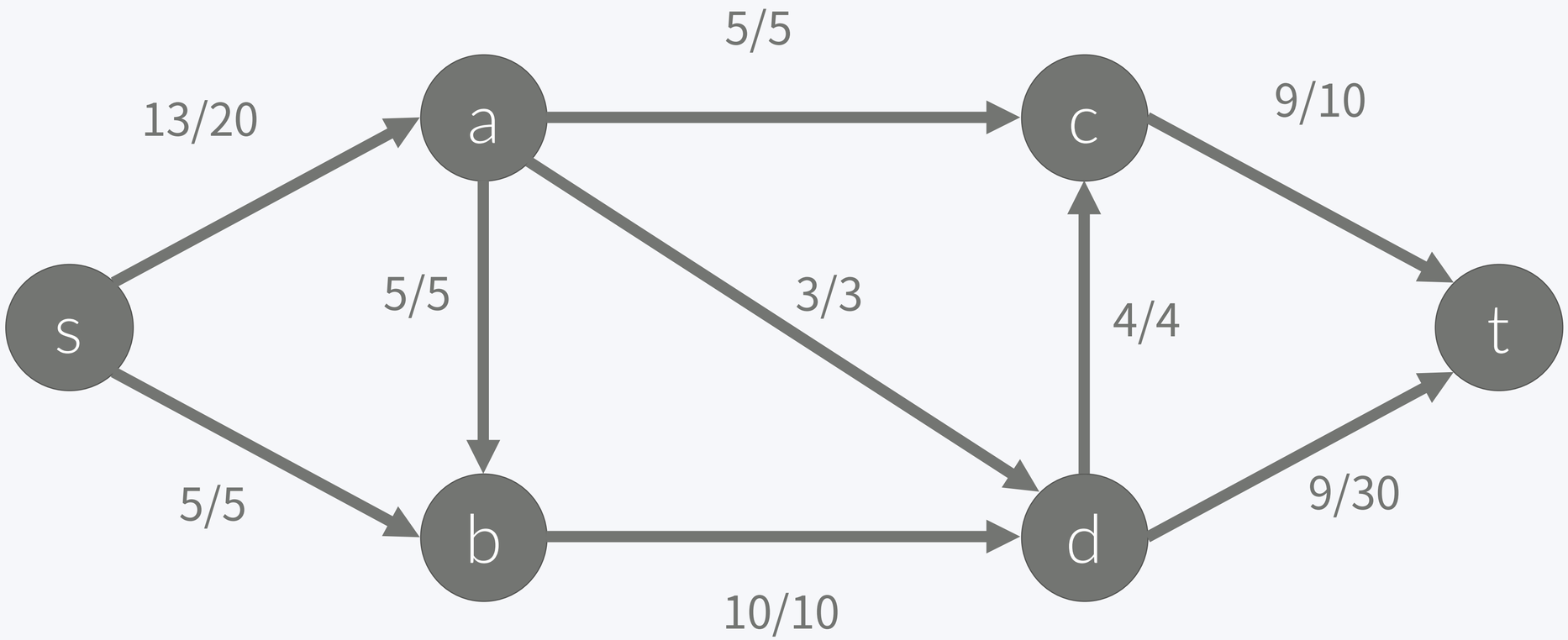
Maximum Flow

- Source (s): 시작
- Sink (t): 끝
- 총 흐른 양
  - Source에서 나간 양
  - 또는
  - Sink로 들어온 양

Residual Capacity

Maximum Flow

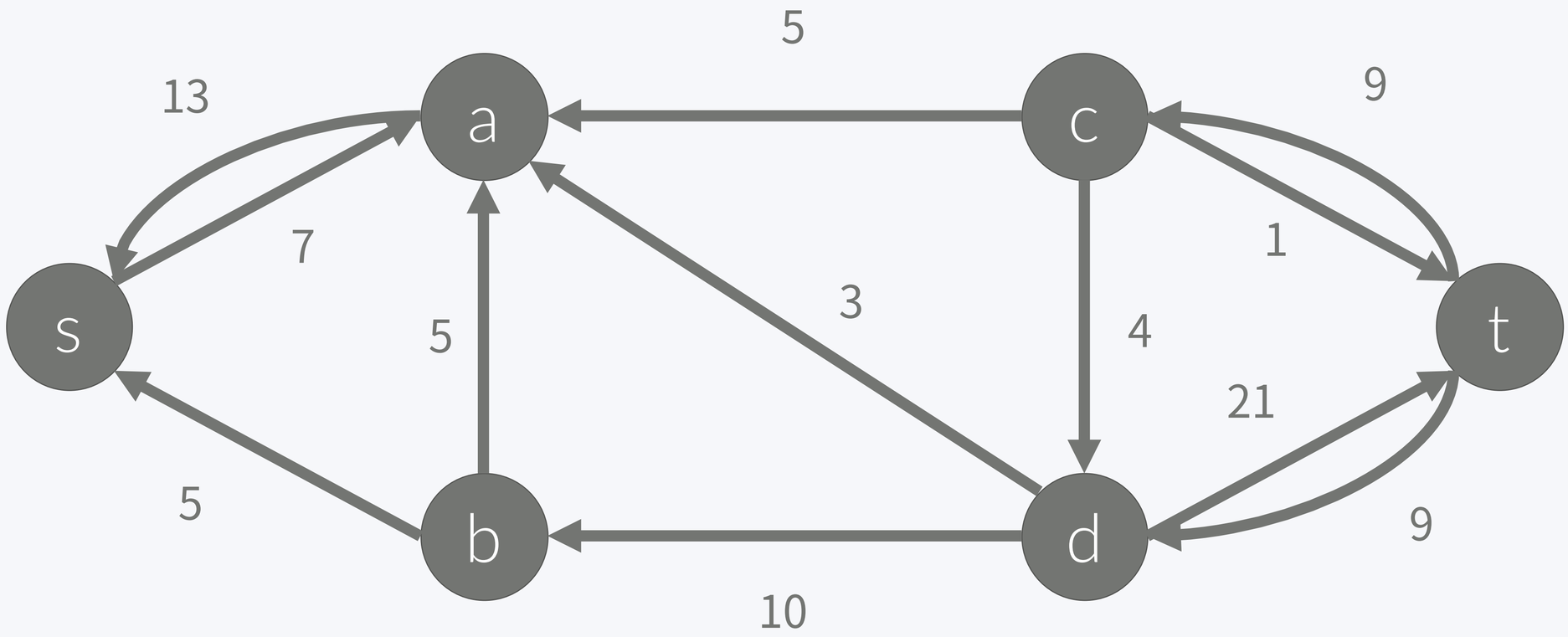
- cf(u, v) = c(u, v) – f(u v)
- Available Capacity



Residual Capacity

Maximum Flow

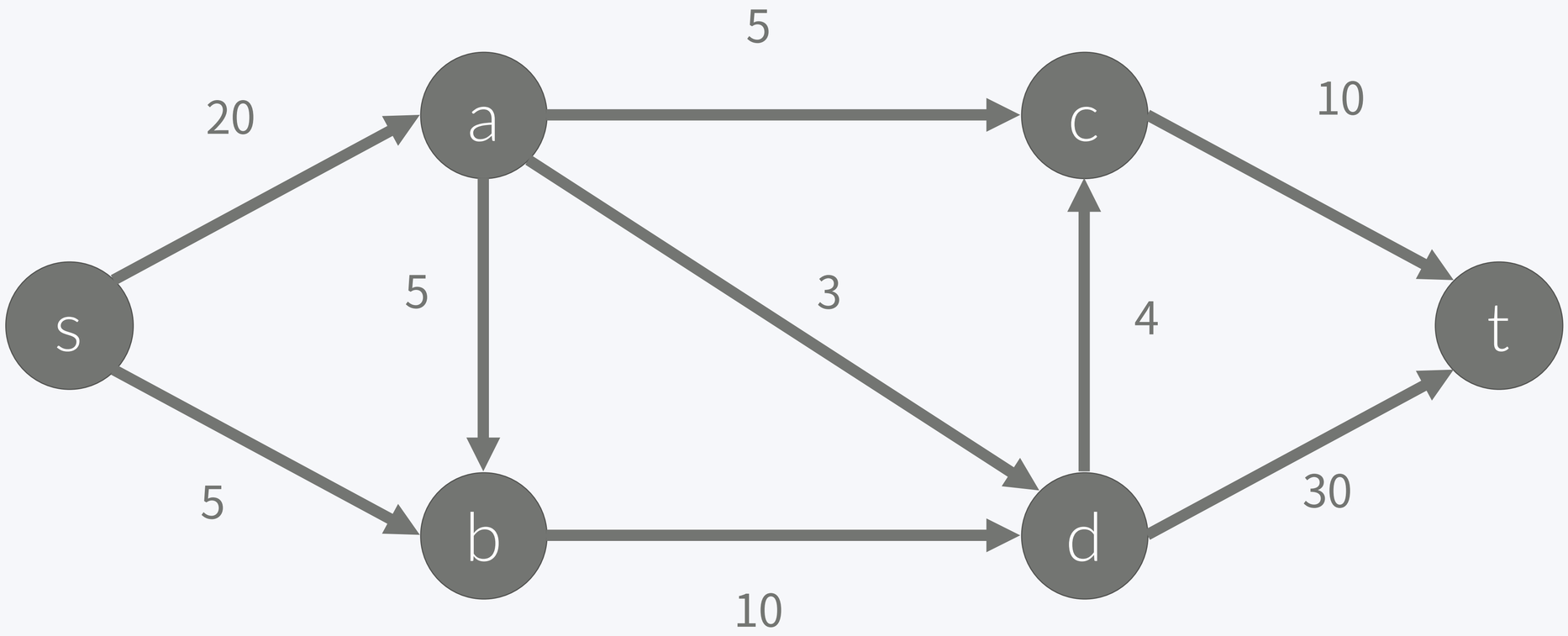
- cf(u, v) = c(u, v) – f(u v)



Augmenting Path

Maximum Flow

- Residual network 에서 구한다
- $u_1, u_2, \cdots, u_k$ (u_1 = Source, u_k = Sink), $cf(u_i, u_{i+1}) > 0$



Ford-Fulkerson

Ford-Fulkerson

Maximum Flow

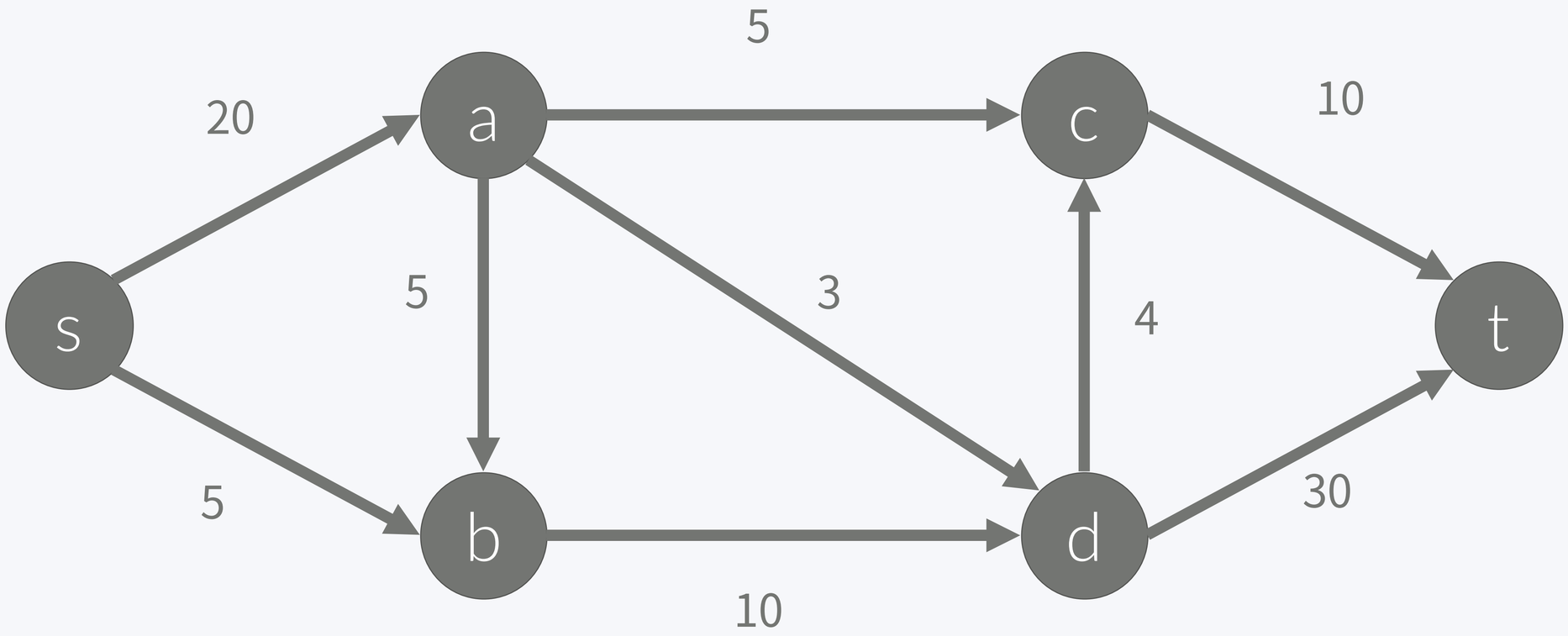
- 최대 유량을 구하는 알고리즘

1. Augmenting Path를 DFS를 이용해서 구한다.
2. m = Augmenting Paht 상에서의 최소값을 구한다
3. (u_i, u_{i+1}) 방향의 Residual Capacity에서 m을 뺀다
4. (u_{i+1}, u_i) 방향의 Residual Capacity에 m을 더한다.
5. 위의 과정을 Augmenting Path를 못 구할때 까지 계속 한다

Ford-Fulkerson

Maximum Flow

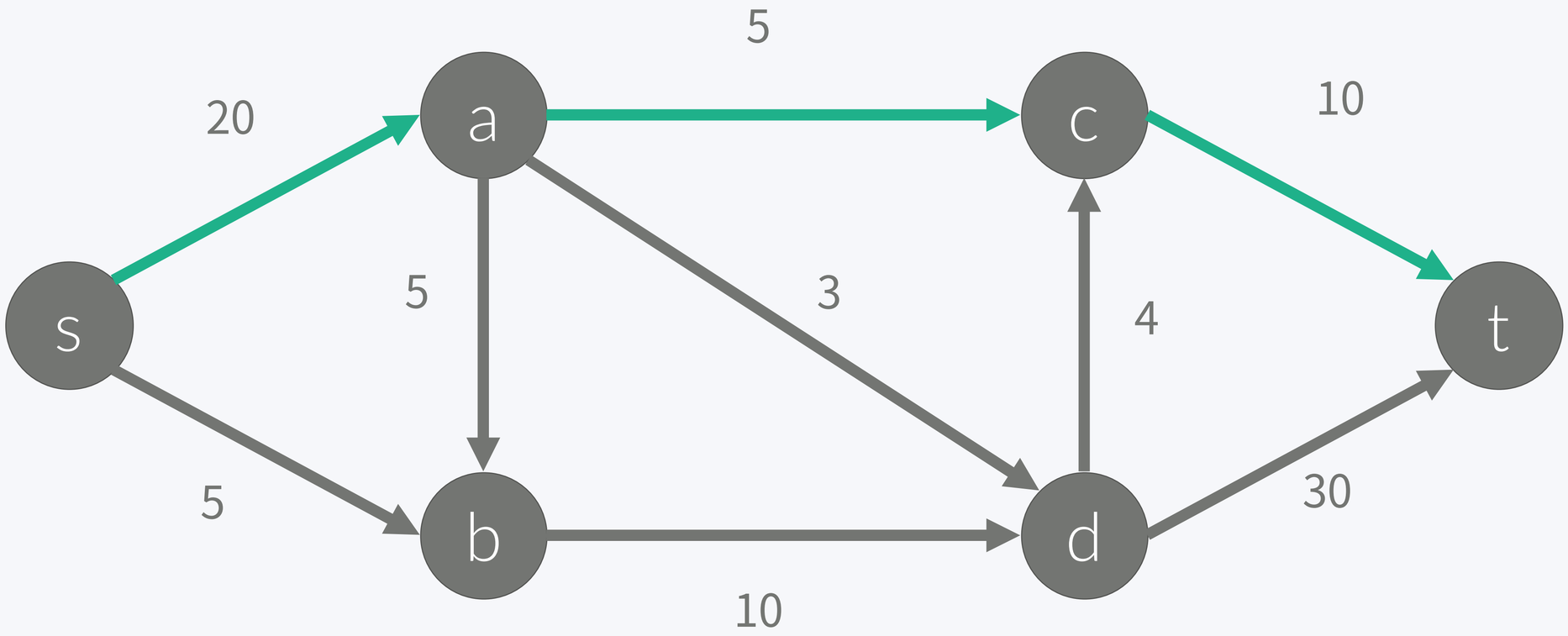
- 그래프



Ford-Fulkerson

Maximum Flow

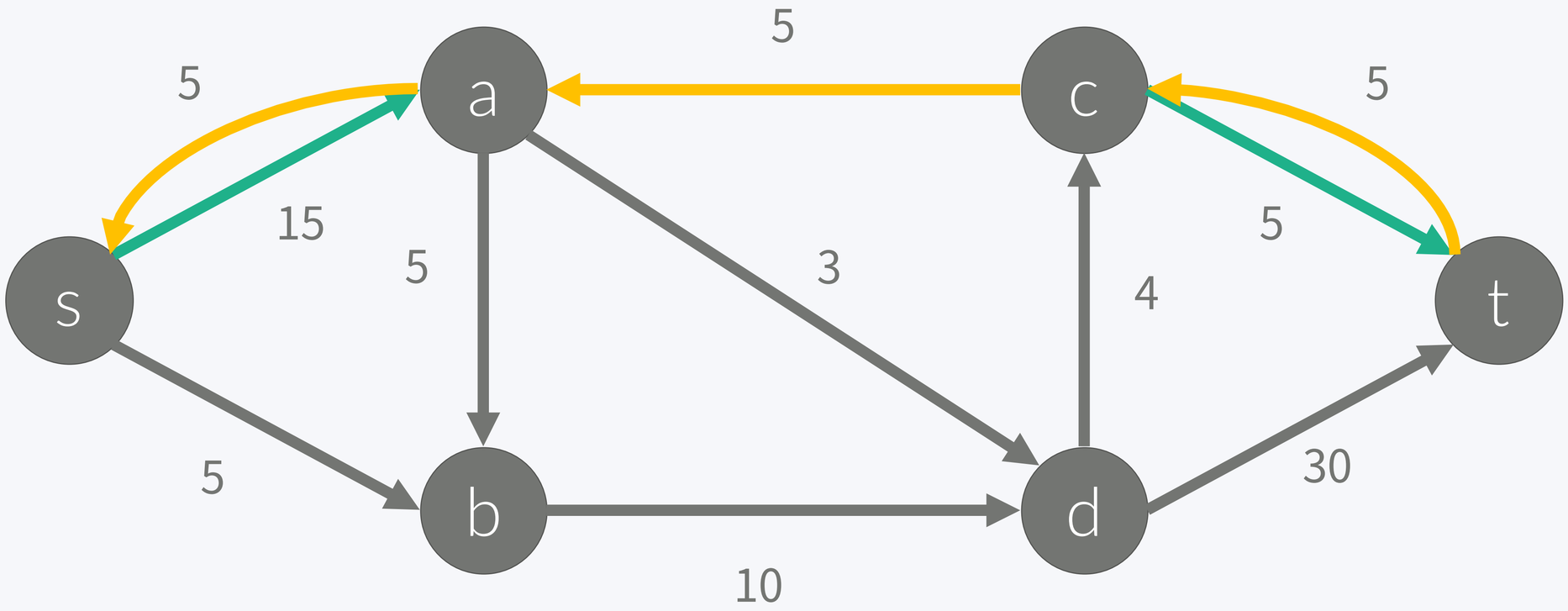
- Augmenting Path: s -> a -> c -> t
- m = 5



Ford-Fulkerson

Maximum Flow

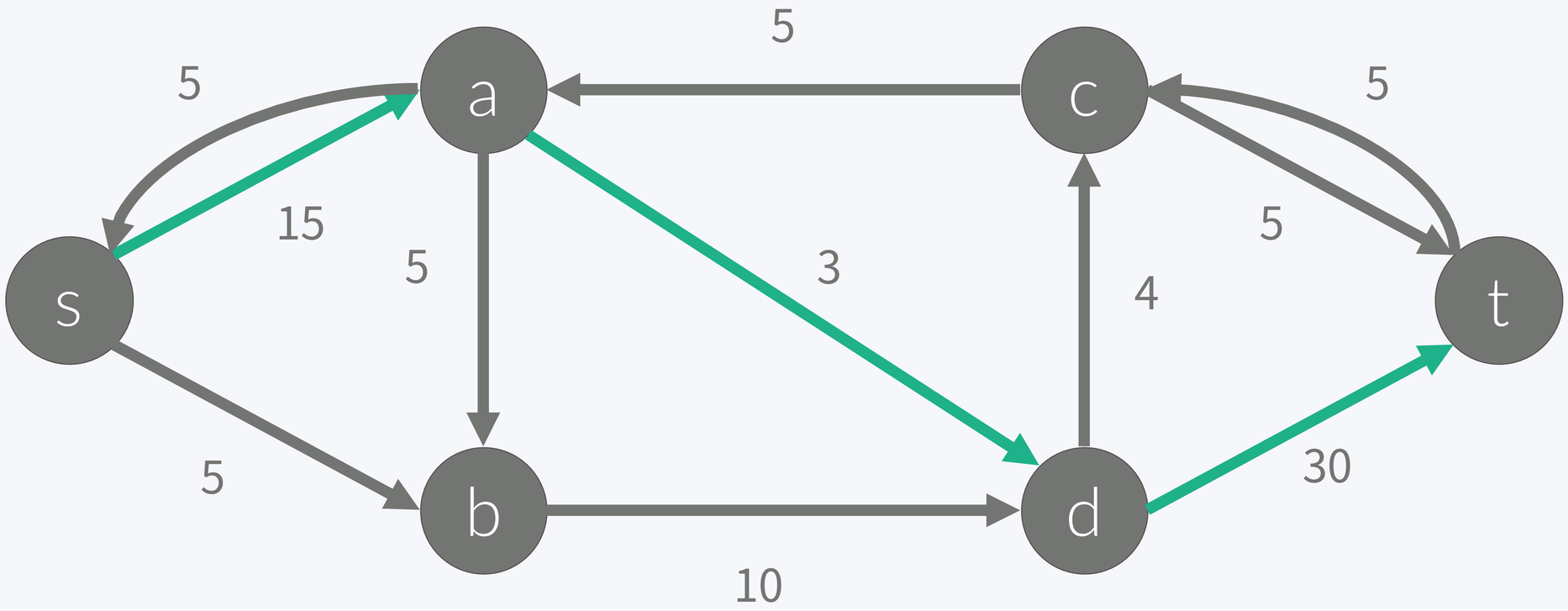
- Augmenting Path: s -> a -> c -> t
- m = 5



Ford-Fulkerson

Maximum Flow

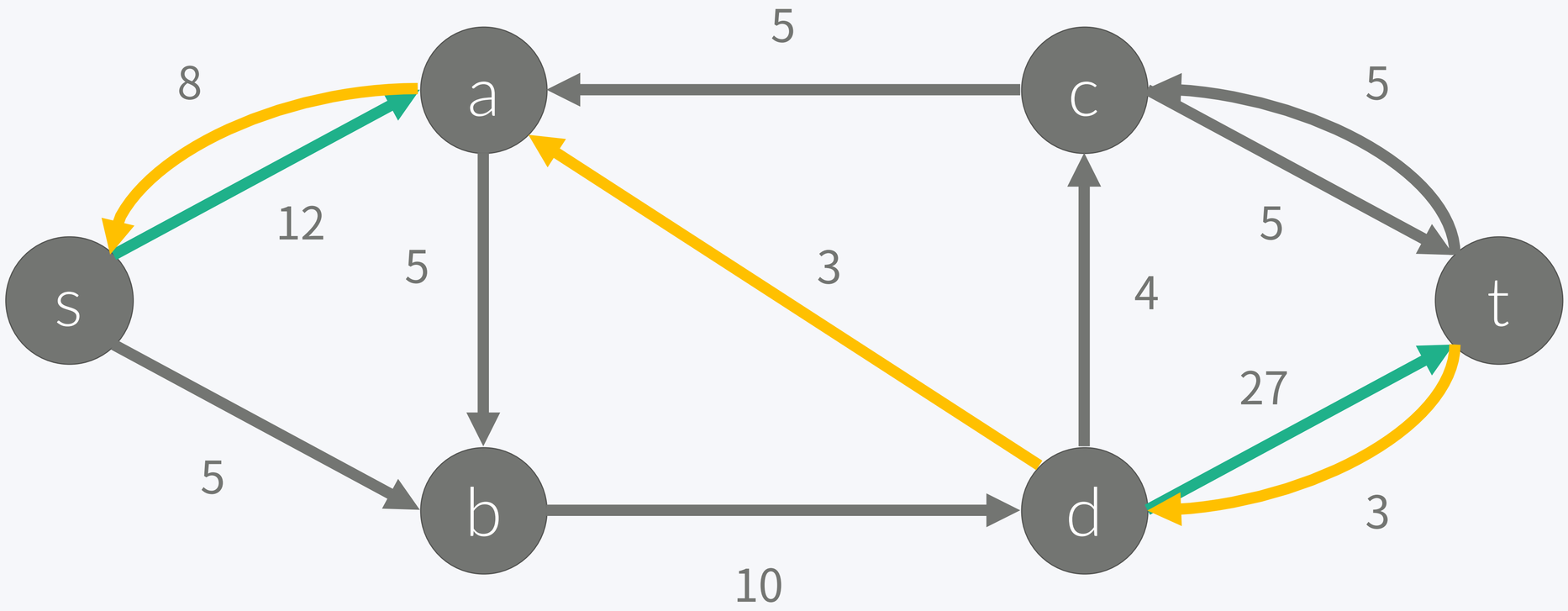
- Augmenting Path: s -> a -> d -> t
- m = 3



Ford-Fulkerson

Maximum Flow

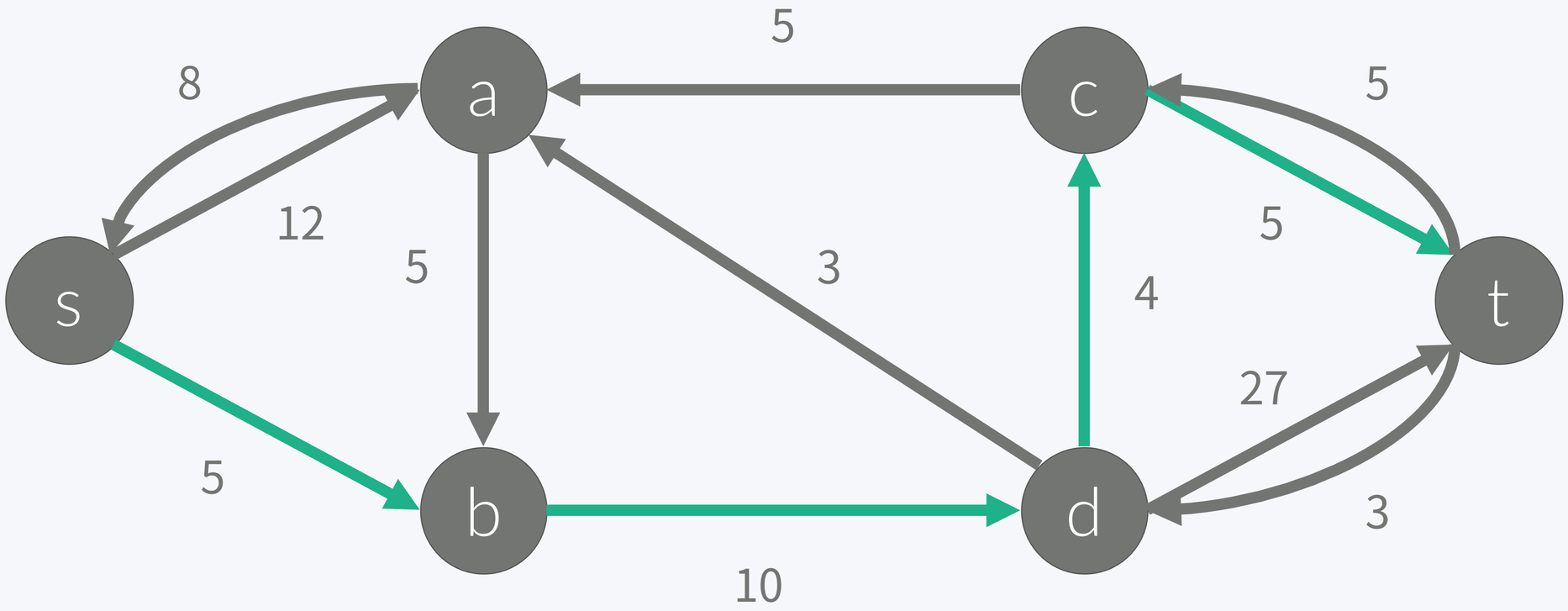
- Augmenting Path: s -> a -> d -> t
- m = 3



Ford-Fulkerson

Maximum Flow

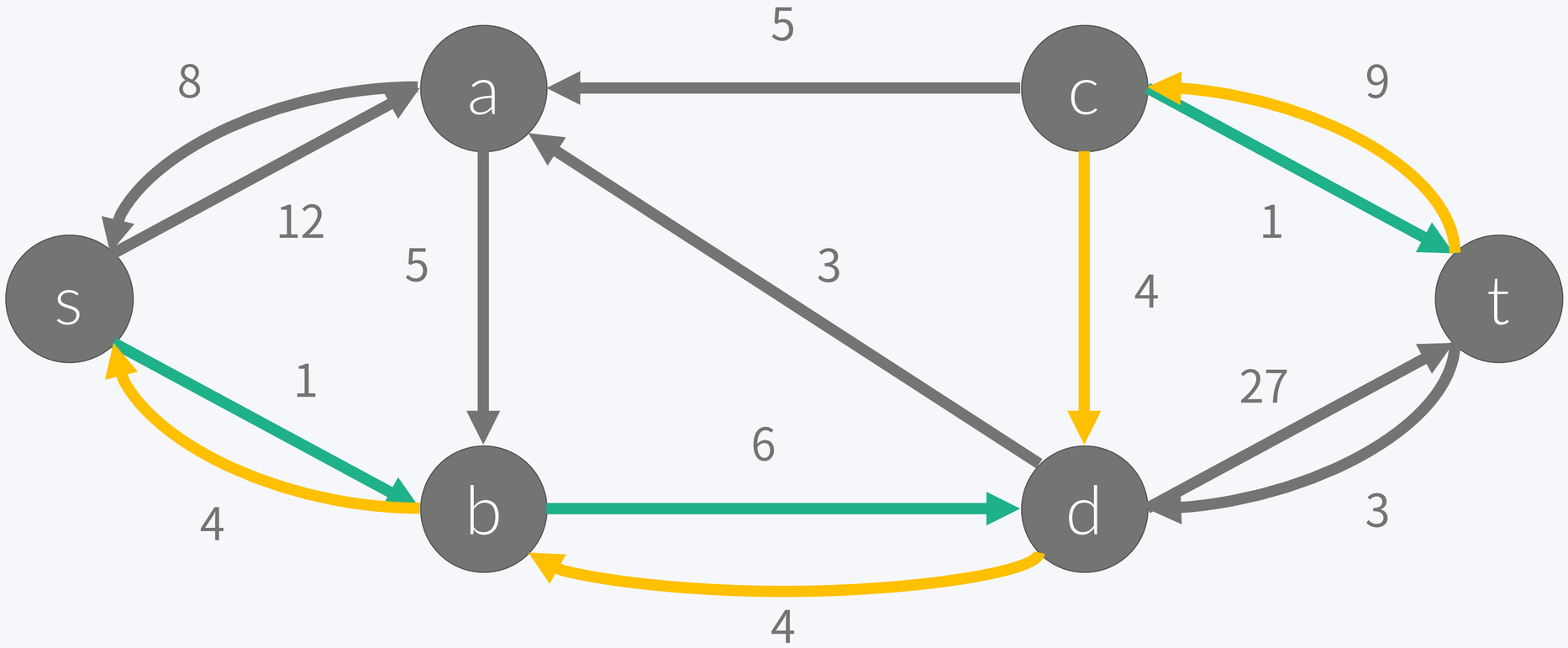
- Augmenting Path: s -> b -> d -> c -> t
- m = 4



Ford-Fulkerson

Maximum Flow

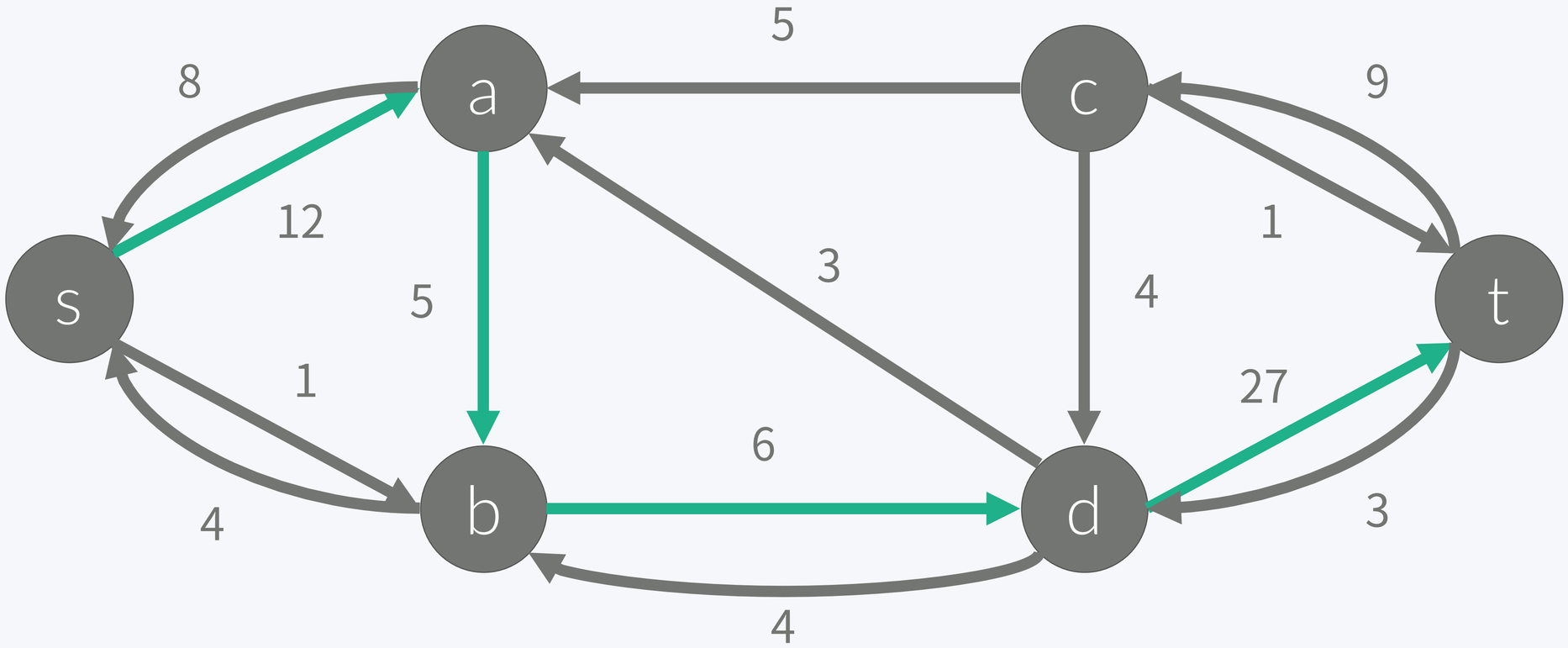
- Augmenting Path: s -> b -> d -> c -> t
- m = 4



Ford-Fulkerson

Maximum Flow

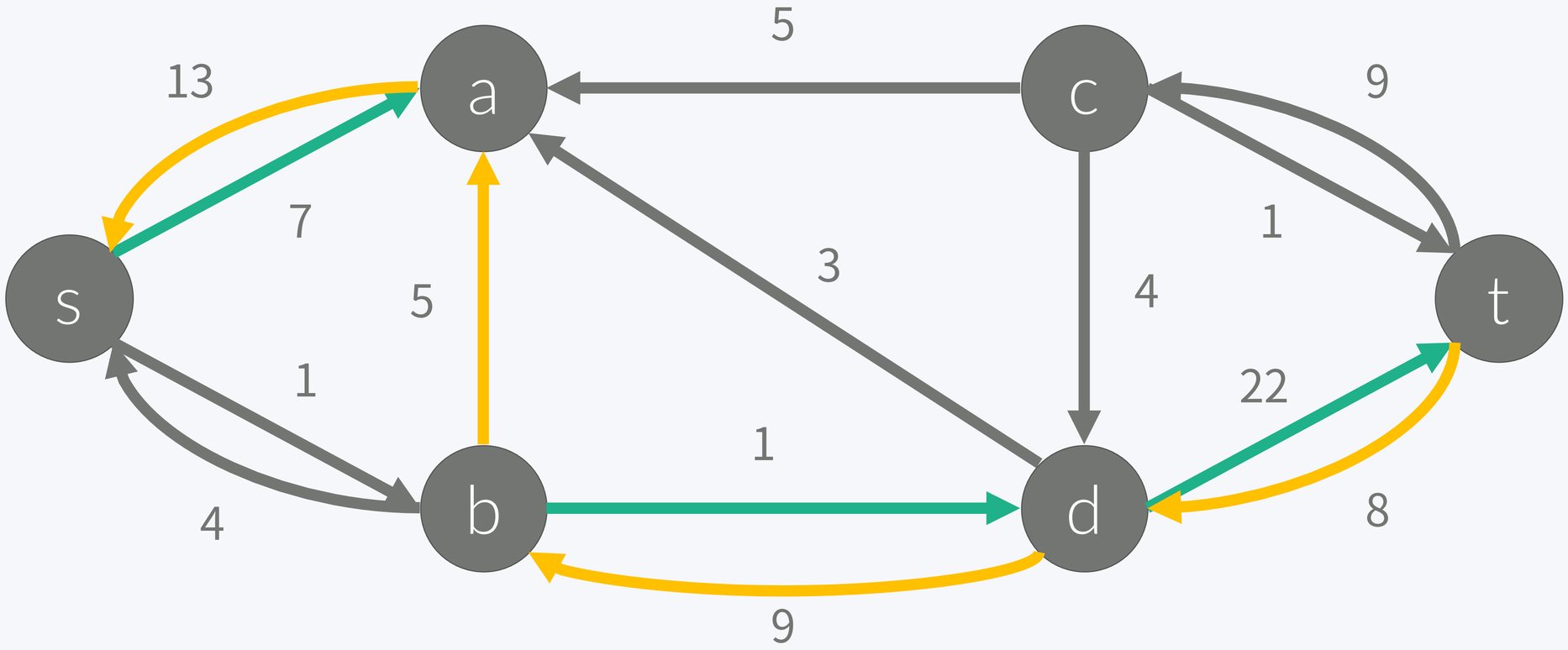
- Augmenting Path: s-> a-> b-> d-> t
- m = 5



Ford-Fulkerson

Maximum Flow

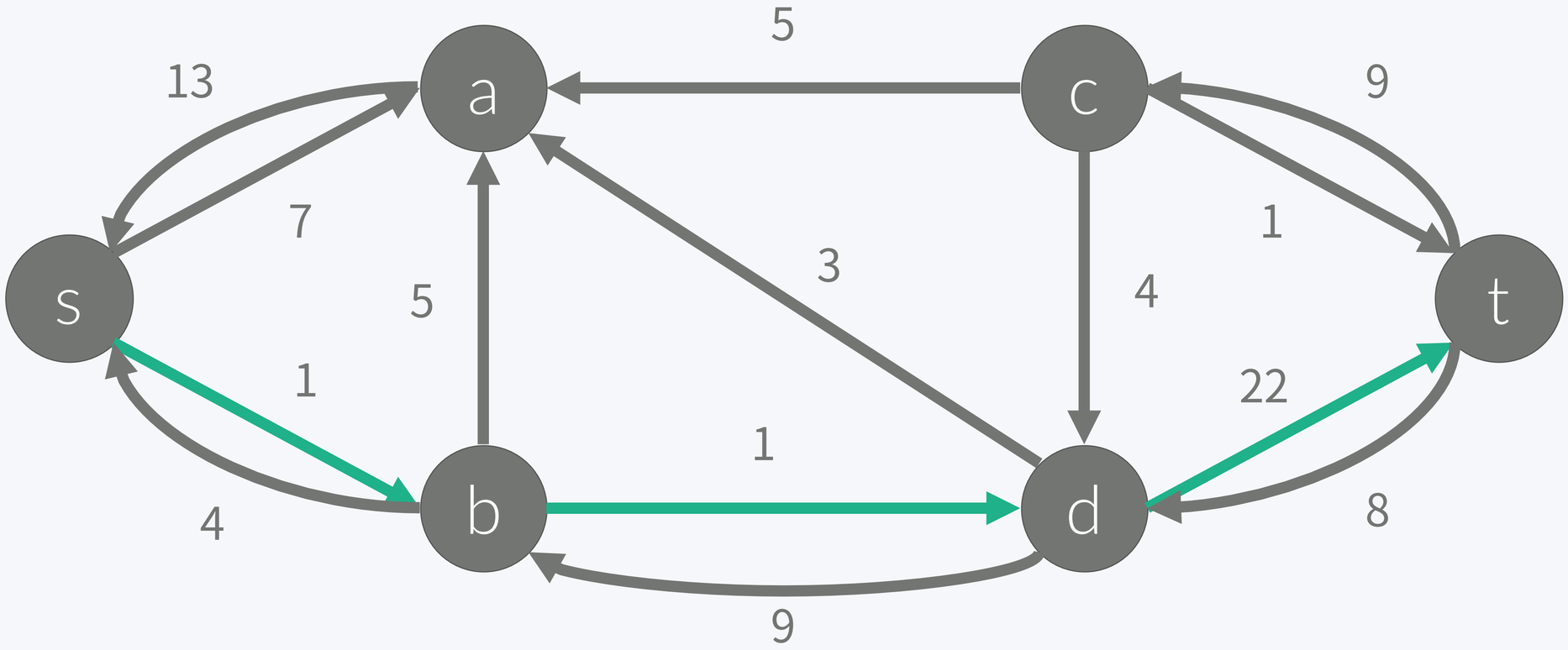
- Augmenting Path: s-> a-> b-> d-> t
- m = 5



Ford-Fulkerson

Maximum Flow

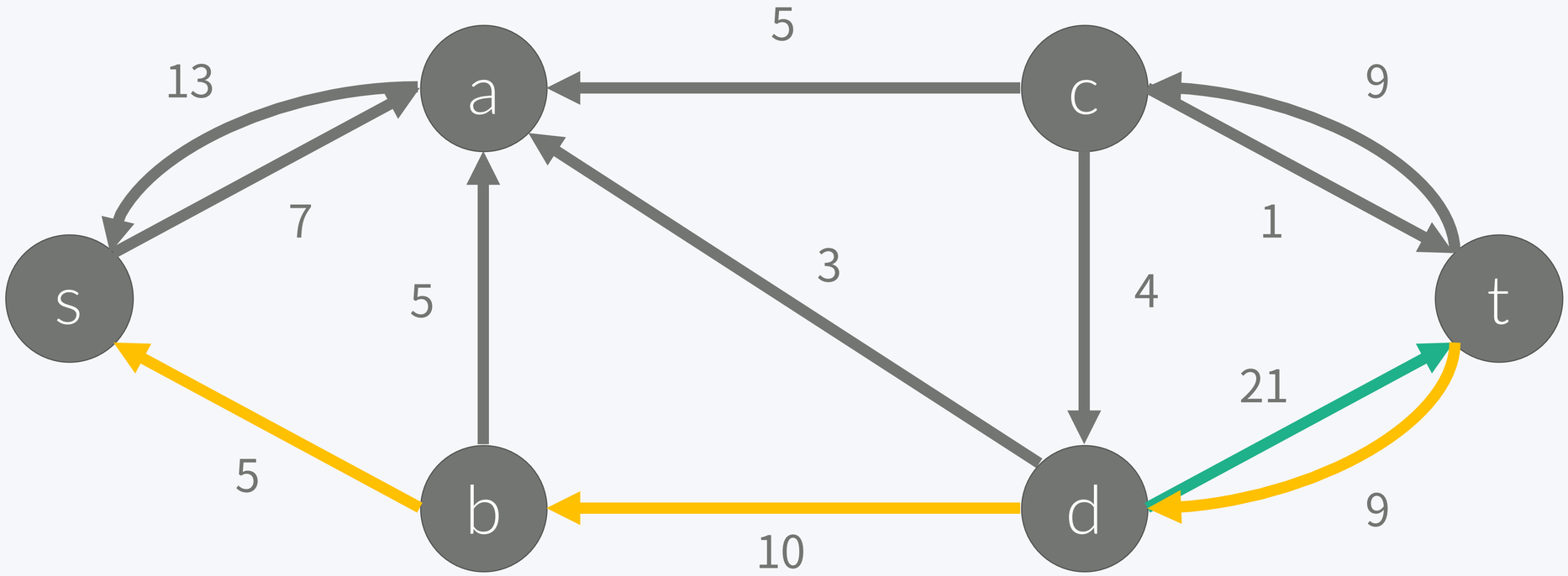
- Augmenting Path: s ->b -> d -> t
- m = 1



Ford-Fulkerson

Maximum Flow

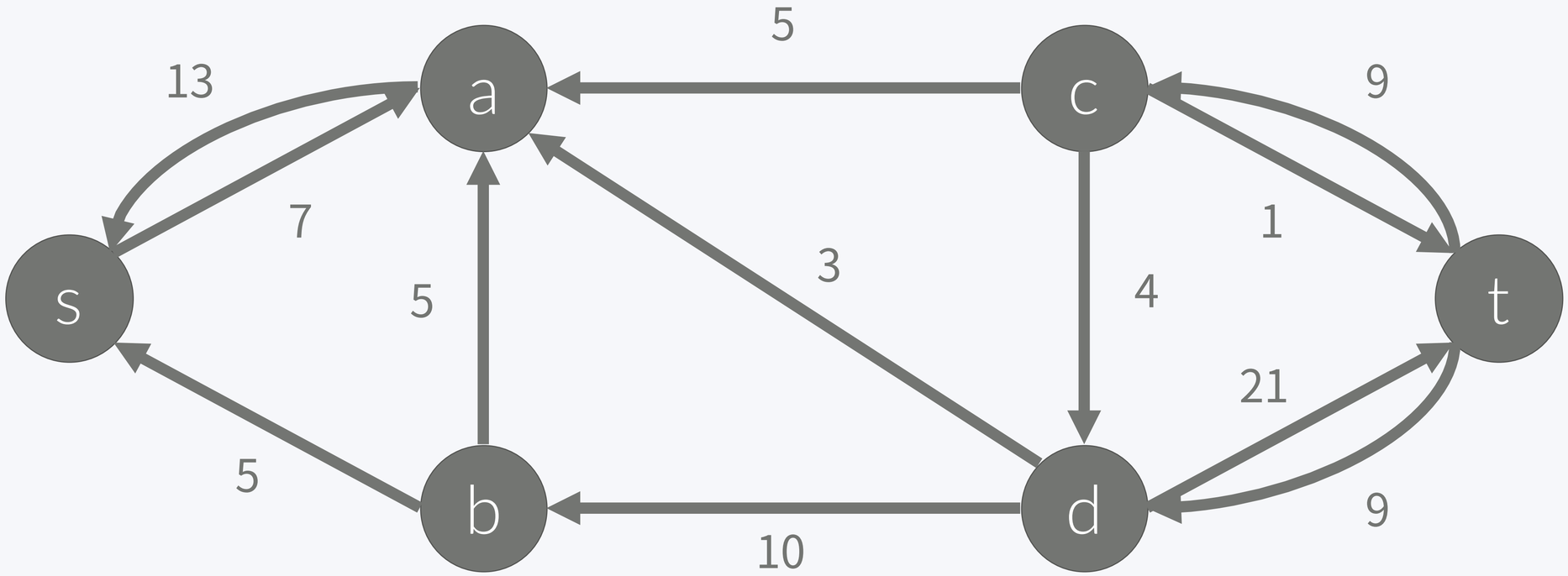
- Augmenting Path: s ->b -> d -> t
- m = 1



Ford-Fulkerson

Maximum Flow

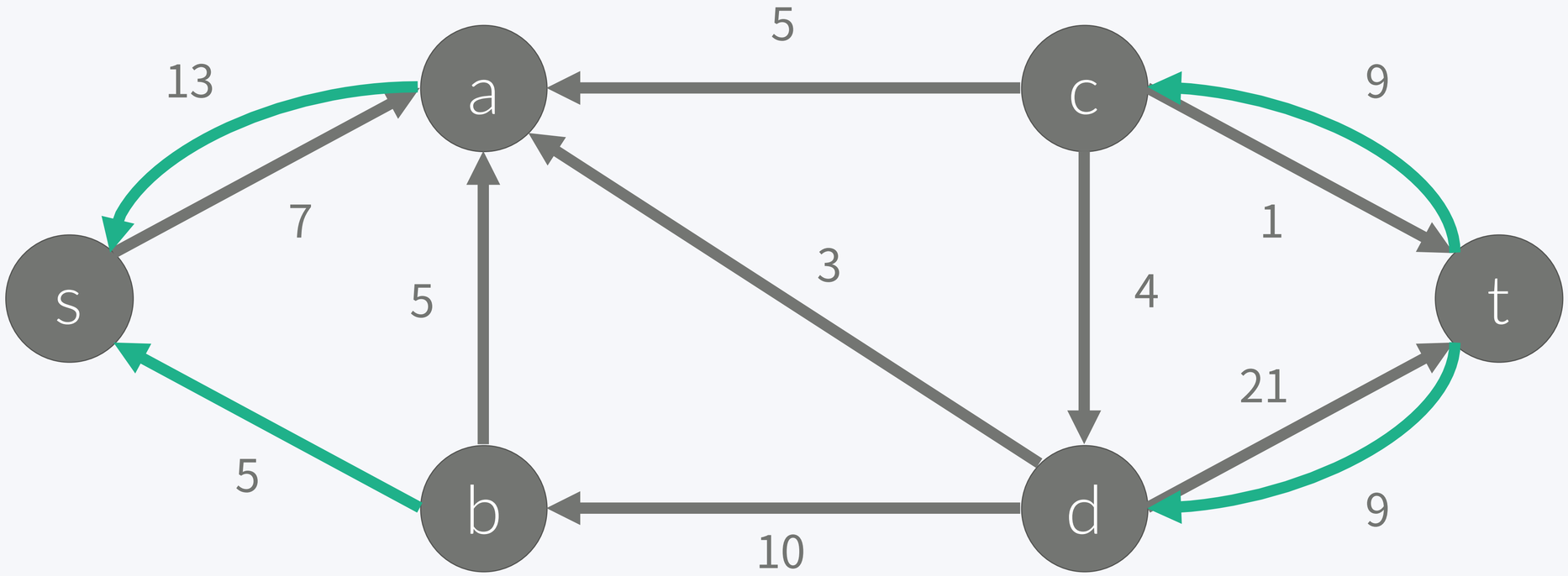
- 더 이상 Augmenting path가 없다



Ford-Fulkerson

Maximum Flow

- Maximum Flow: 18



Ford-Fulkerson

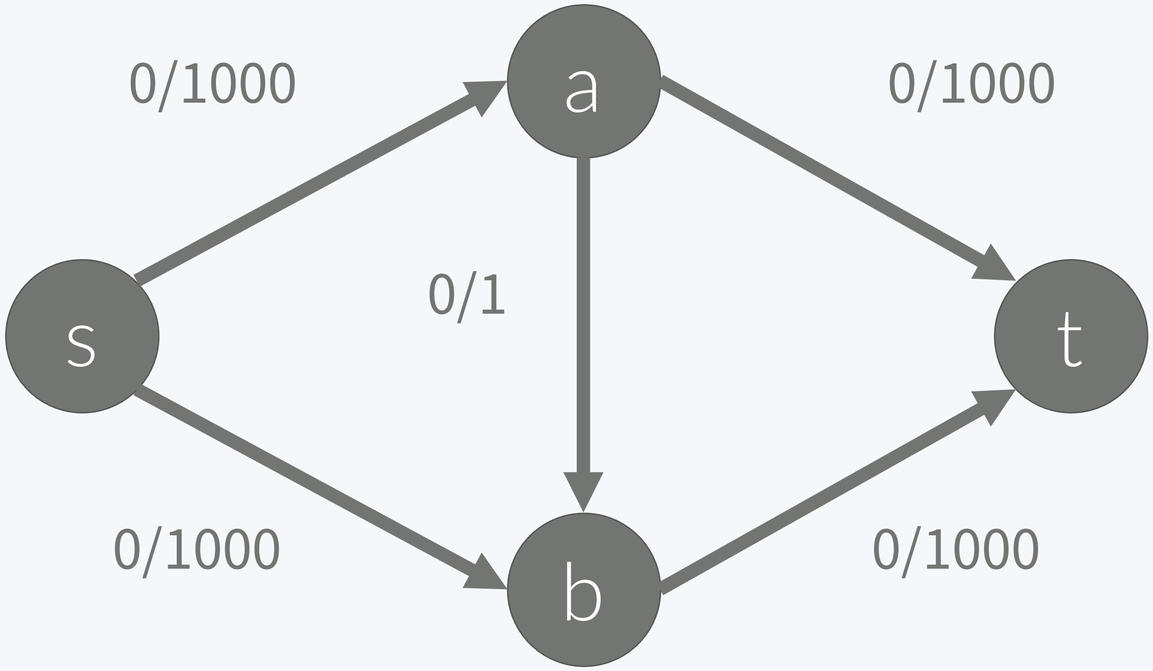
Maximum Flow

- 시간복잡도: O(Ef)
    - E : Edge 개수
    - f: Maximum Flow

Ford-Fulkerson

Maximum Flow

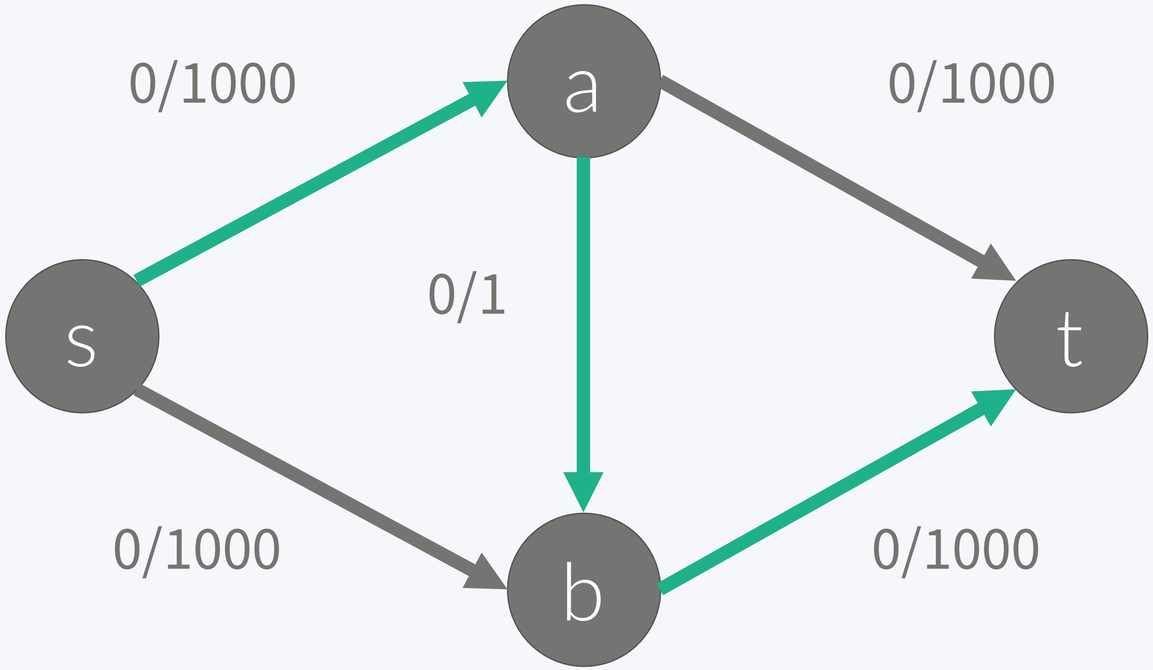
- 이런 경우



Ford-Fulkerson

Maximum Flow

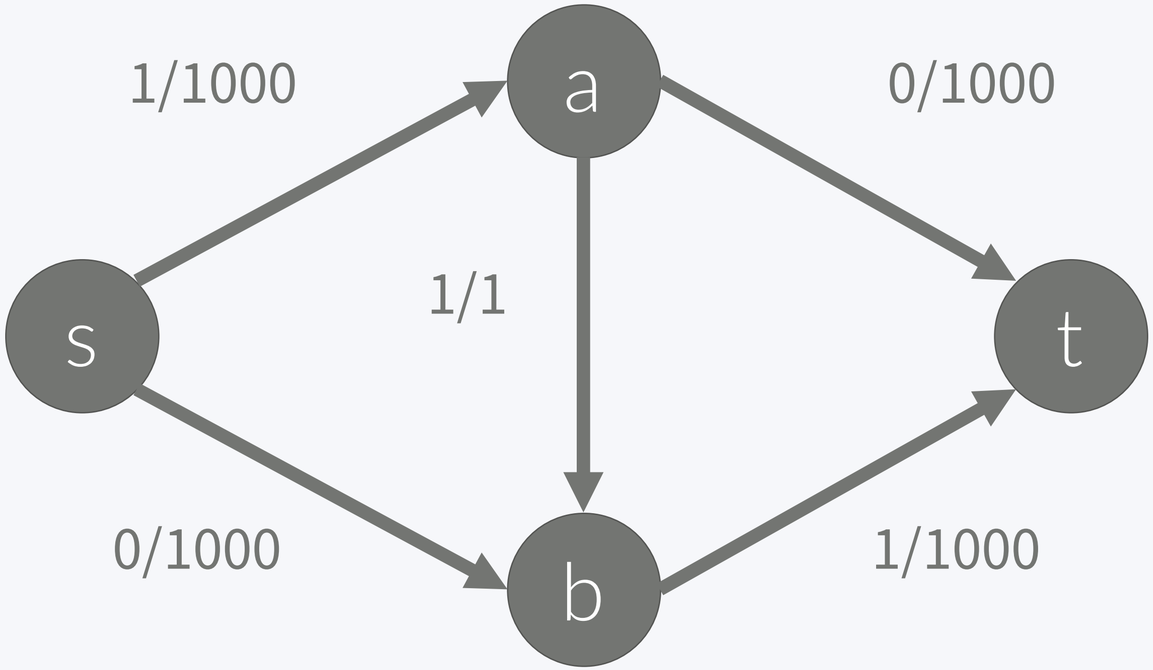
- 이런 경우



Ford-Fulkerson

Maximum Flow

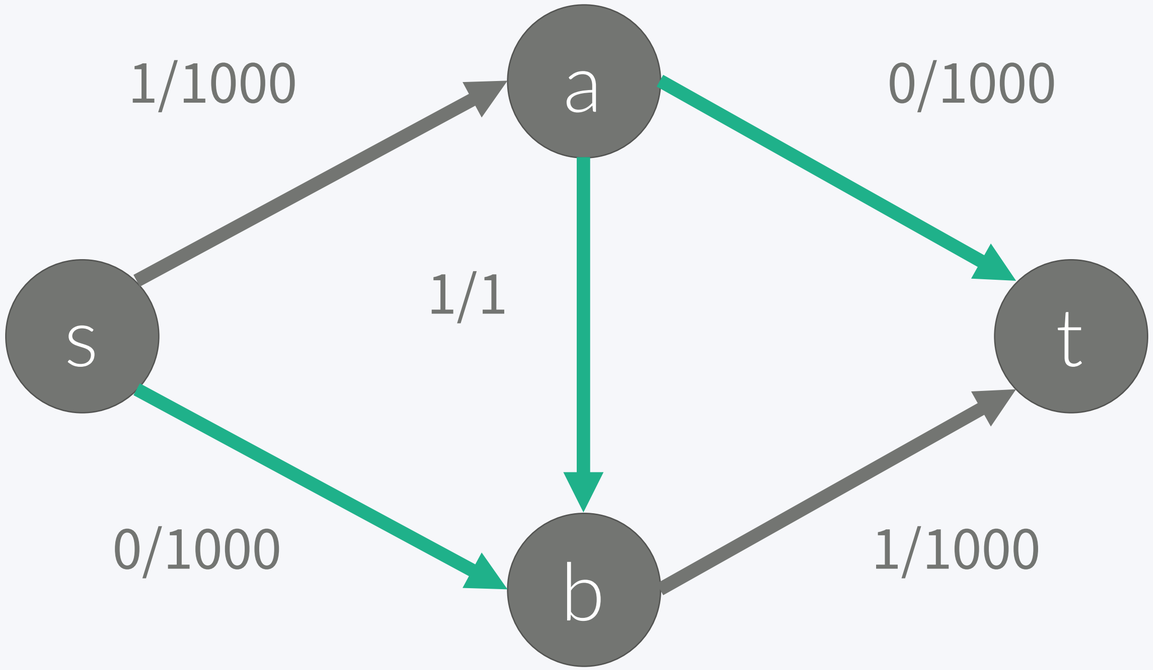
- 이런 경우



Ford-Fulkerson

Maximum Flow

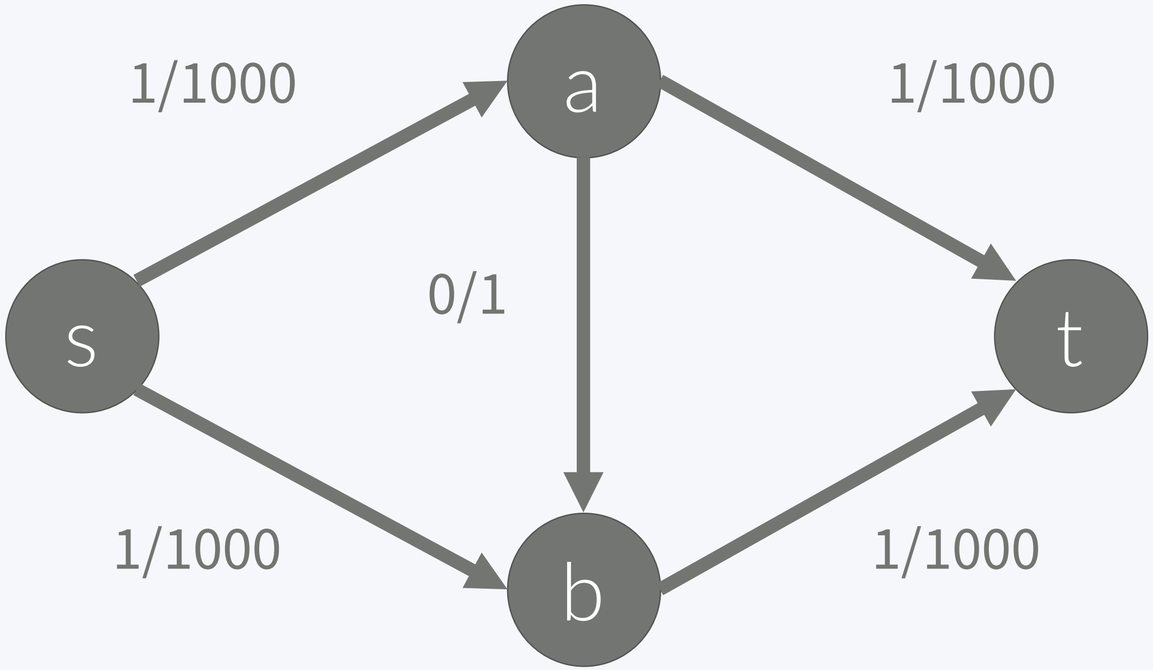
- 이런 경우



Ford-Fulkerson

Maximum Flow

- 이런 경우



Ford-Fulkerson

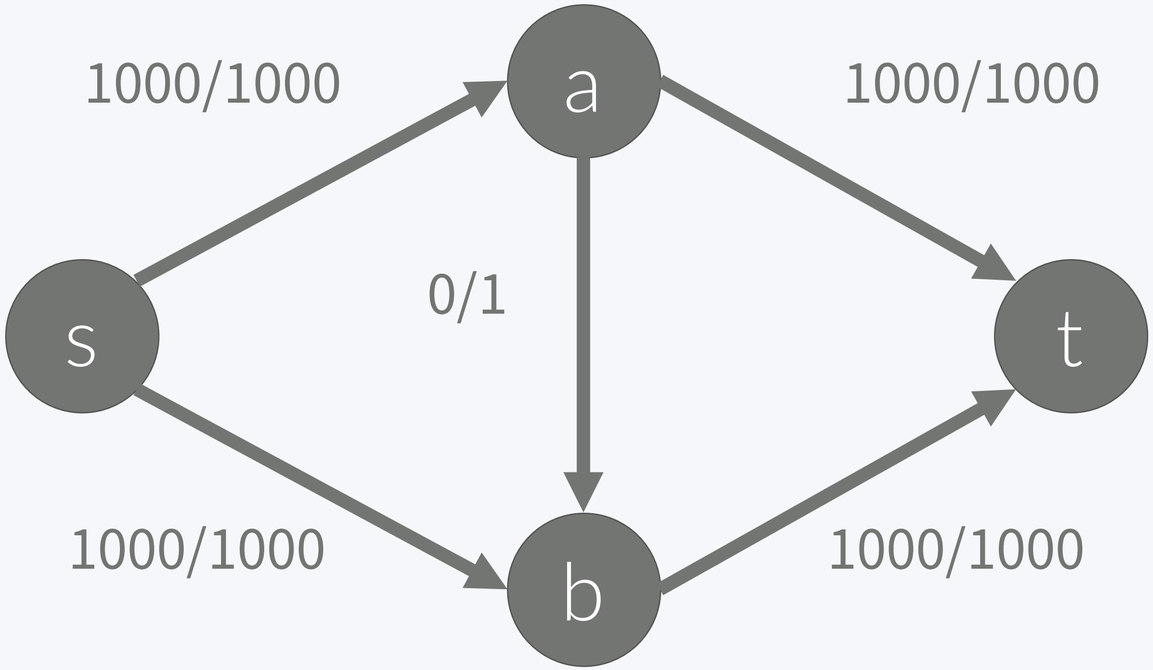
Maximum Flow

- 이 과정을 1998번 더 하면

Ford-Fulkerson

Maximum Flow

- Flow network



Edmond-Karp

Edmond-Karp

Maximum Flow

- 최대 유량을 구하는 알고리즘

1. Augmenting Path를 BFS를 이용해서 구한다.
2. m = Augmenting Paht 상에서의 최소값을 구한다
3. (u_i, u_{i+1}) 방향의 Residual Capacity에서 m을 뺀다
4. (u_{i+1}, u_i) 방향의 Residual Capacity에 m을 더한다.
5. 위의 과정을 Augmenting Path를 못 구할때 까지 계속 한다

Total Flow

https://www.acmicpc.net/problem/6086

- 최대 유량을 구하는 문제

Total Flow

https://www.acmicpc.net/problem/6086

- Matrix, DFS, Residual
  - https://gist.github.com/Baekjoon/63c6f5c07cbd5251d0d6
- Matrix, DFS, Capacity, Flow
  - https://gist.github.com/Baekjoon/d92e03bd97277ebef159
- Matrix, BFS, Residual
  - https://gist.github.com/Baekjoon/44811c5633c388a16831
- List, BFS, Residual
  - https://gist.github.com/Baekjoon/d67c59ead5a81d1e03d4
- List, DFS, Residual
  - https://gist.github.com/Baekjoon/a1acd6844d9a8ef068e6

Total Flow

https://www.acmicpc.net/problem/6086

- struct NetworkFlow (DFS)
    - https://gist.github.com/Baekjoon/4245b1376bdda0f5bc04
- struct NetworkFlow (BFS)
    - https://gist.github.com/Baekjoon/6aa876d32b85c709d0e3

이분 매칭

이분 매칭

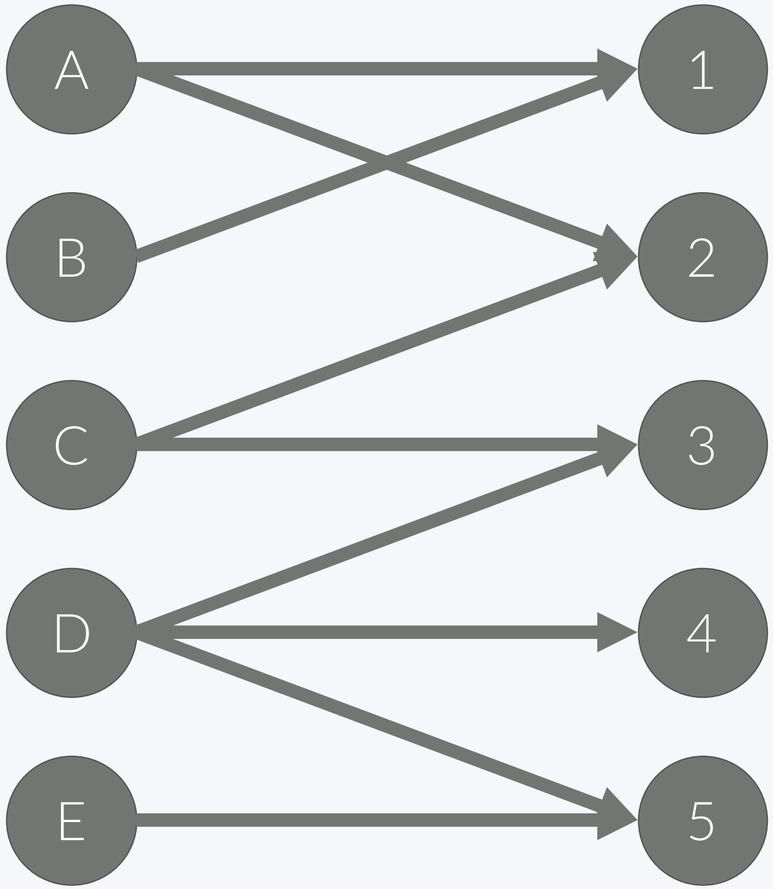
Bipartite Matching

- Matching: 그래프 G에서 적절히 간선을 선택했을 때, 각각의 간선이 공통된 vertex를 공유하지 않음
- Maximum Matching: Edge의 최대값

이분 매칭

Bipartite Matching

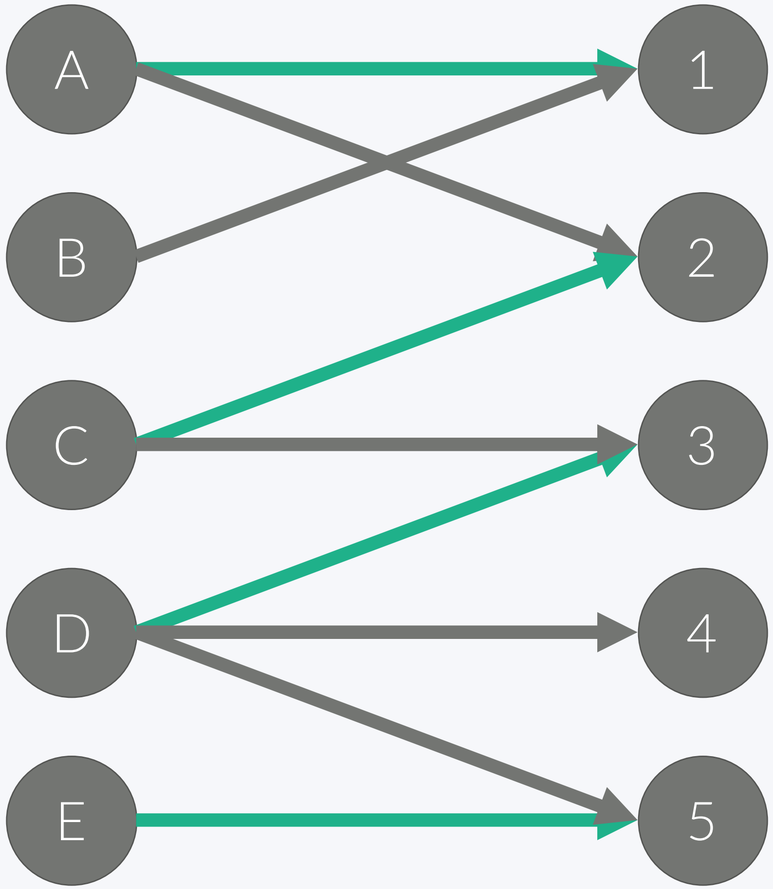
- 사람: A~E, 일: 1~5, 각 사람이 할 수 있는 일을 Edge로 연결



이분 매칭

Bipartite Matching

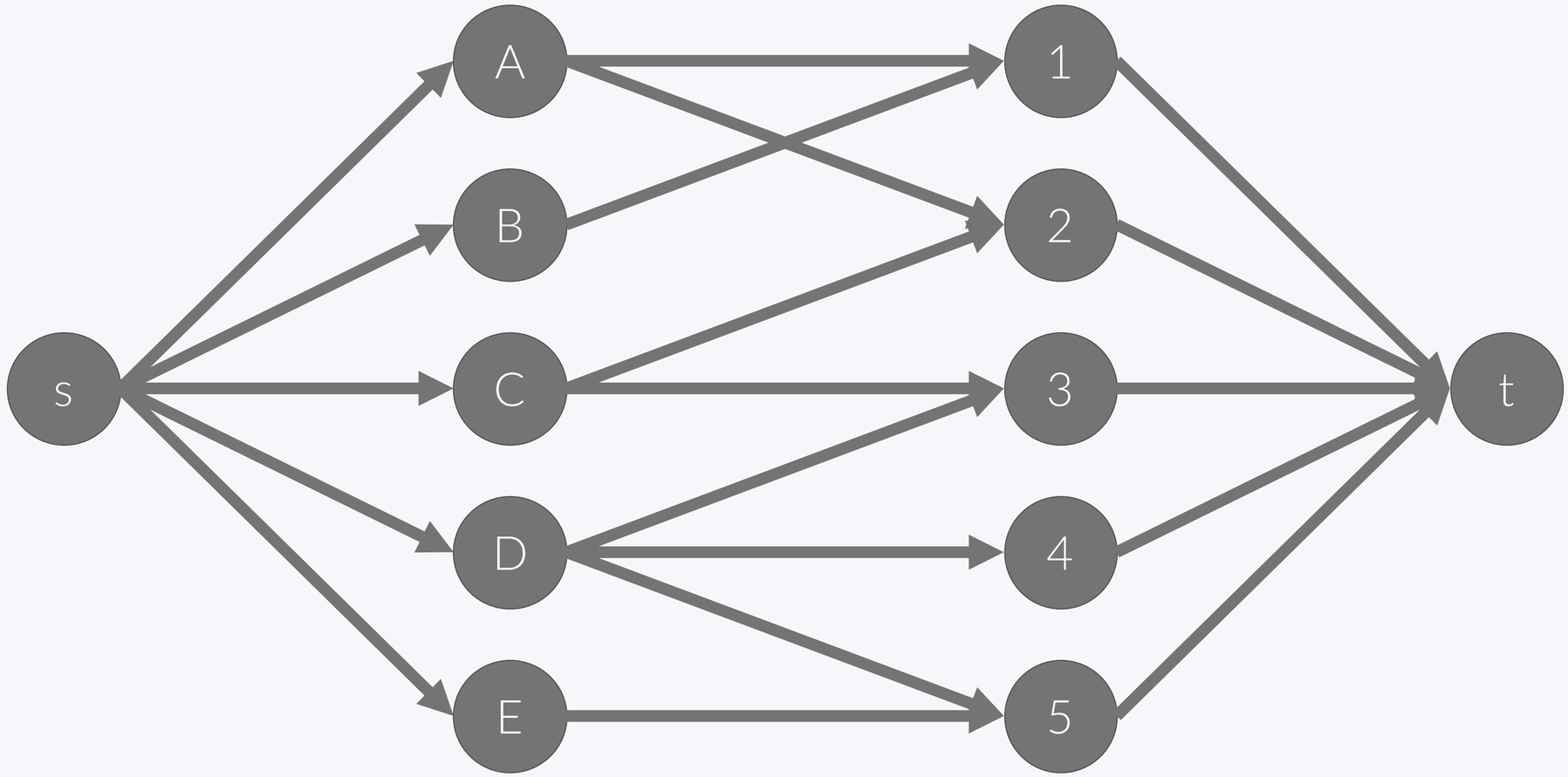
- 사람: A~E, 일: 1~5, 각 사람이 할 수 있는 일을 Edge로 연결



이분 매칭

Bipartite Matching

- 네트워크 플로우 문제로 바꿔서 풀 수 있다. (capacity = 1)



이분 매칭

Bipartite Matching

- 이분 그래프에서 (A -> B)
- 소스(s)와 싱크(t)를 추가하고
- s -> A로 edge를 연결
- B -> t로 edge를 연결
- 모든 capacity = 1
- Maximum Flow가 답이 된다

이분 매칭

Bipartite Matching

- A에 속해 있는 노드는 최대 1개의 나가는 edge를 선택
- B에 속해 있는 노드는 최대 1개의 들어오는 edge를 선택

열혈강호

https://www.acmicpc.net/problem/11375

- 사람 N명 일 M개
- 각 직원은 한 개의 일만 할 수 있고, 각각의 일을 담당하는 사람은 1명이어야 한다.
- 할 수 있는 일의 최대 개수

열혈강호

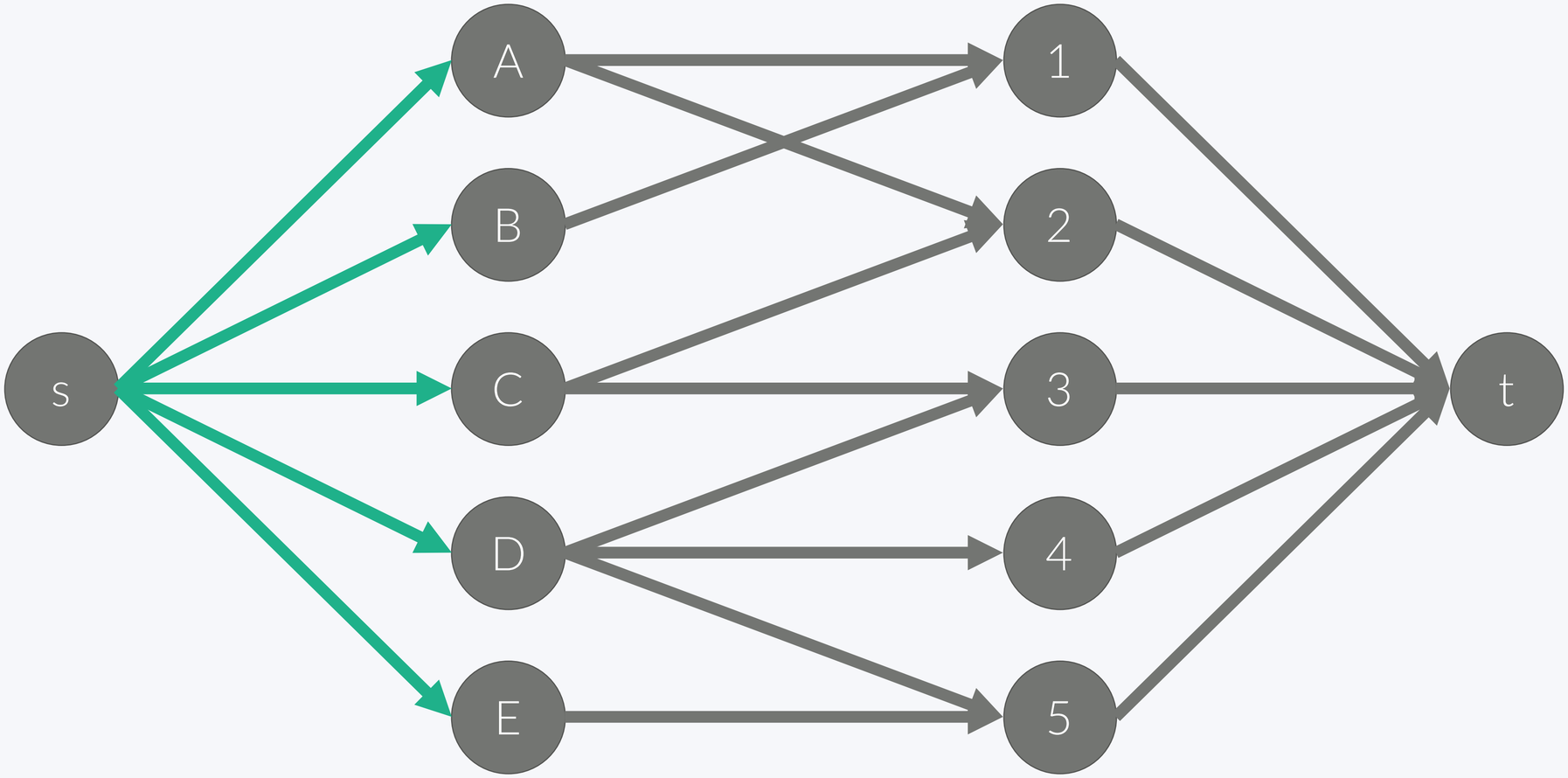
https://www.acmicpc.net/problem/11375

- Ford-Fulkerson: https://gist.github.com/Baekjoon/4b6f576da76e9a035405
- Ford-Fulkerson 다른 방식 구현: https://gist.github.com/Baekjoon/7dc06797bddc4664f599

이분 매칭

Bipartite Matching

- 항상 1 또는 0이기 때문에, edge를 만들지 않아도 된다



이분 매칭

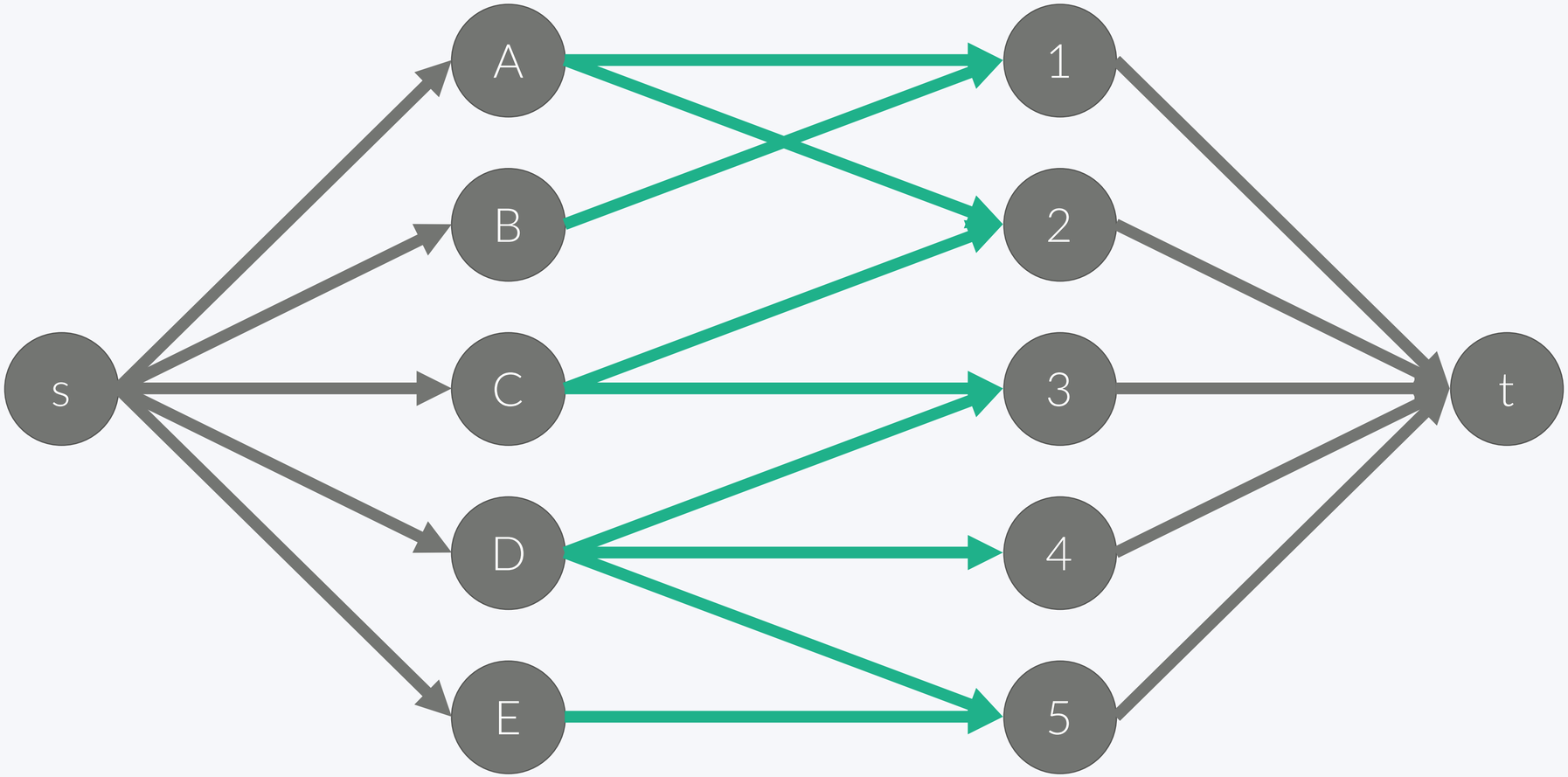
Bipartite Matching


```
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;
}
```

이분 매칭

Bipartite Matching

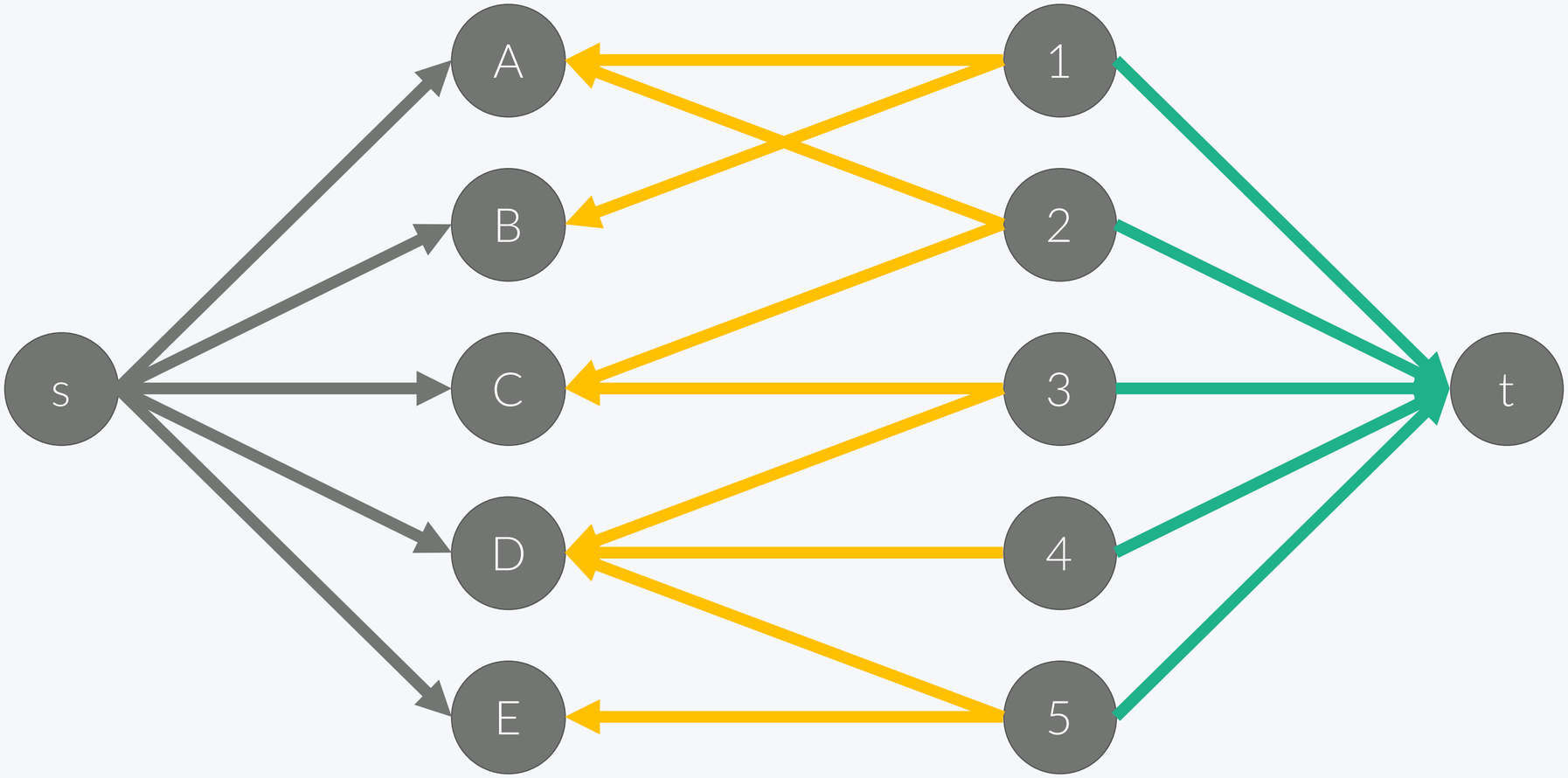
- 그대로 유지해야 한다



이분 매칭

Bipartite Matching

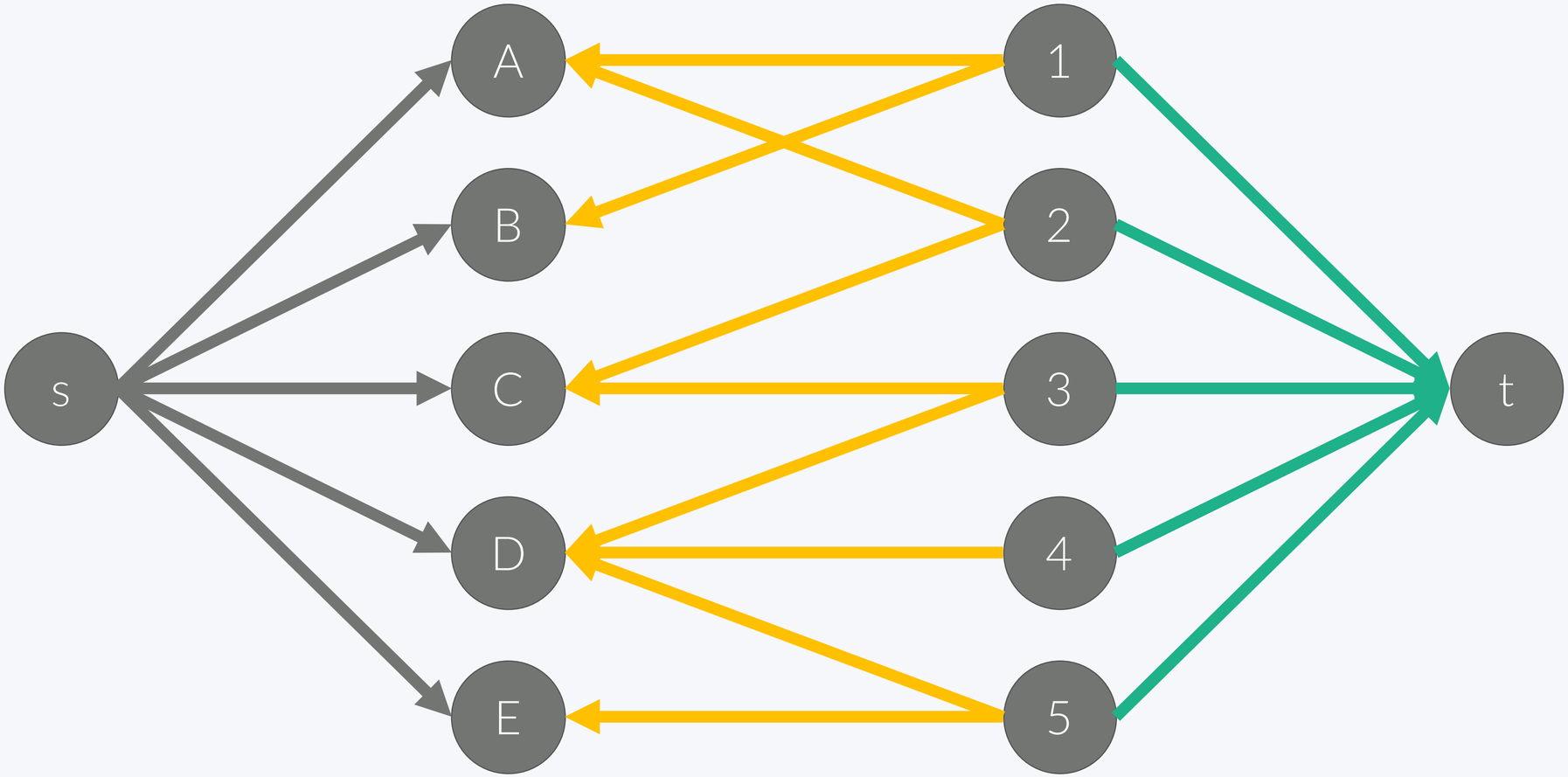
- 오른쪽은 항상 Sink로 가거나 아니면 다시 왼쪽으로 가게된다



이분 매칭

Bipartite Matching

- Sink로 가는 경우에는 -1, 왼쪽으로 가는 경우에는 그 정점 번호



이분 매칭

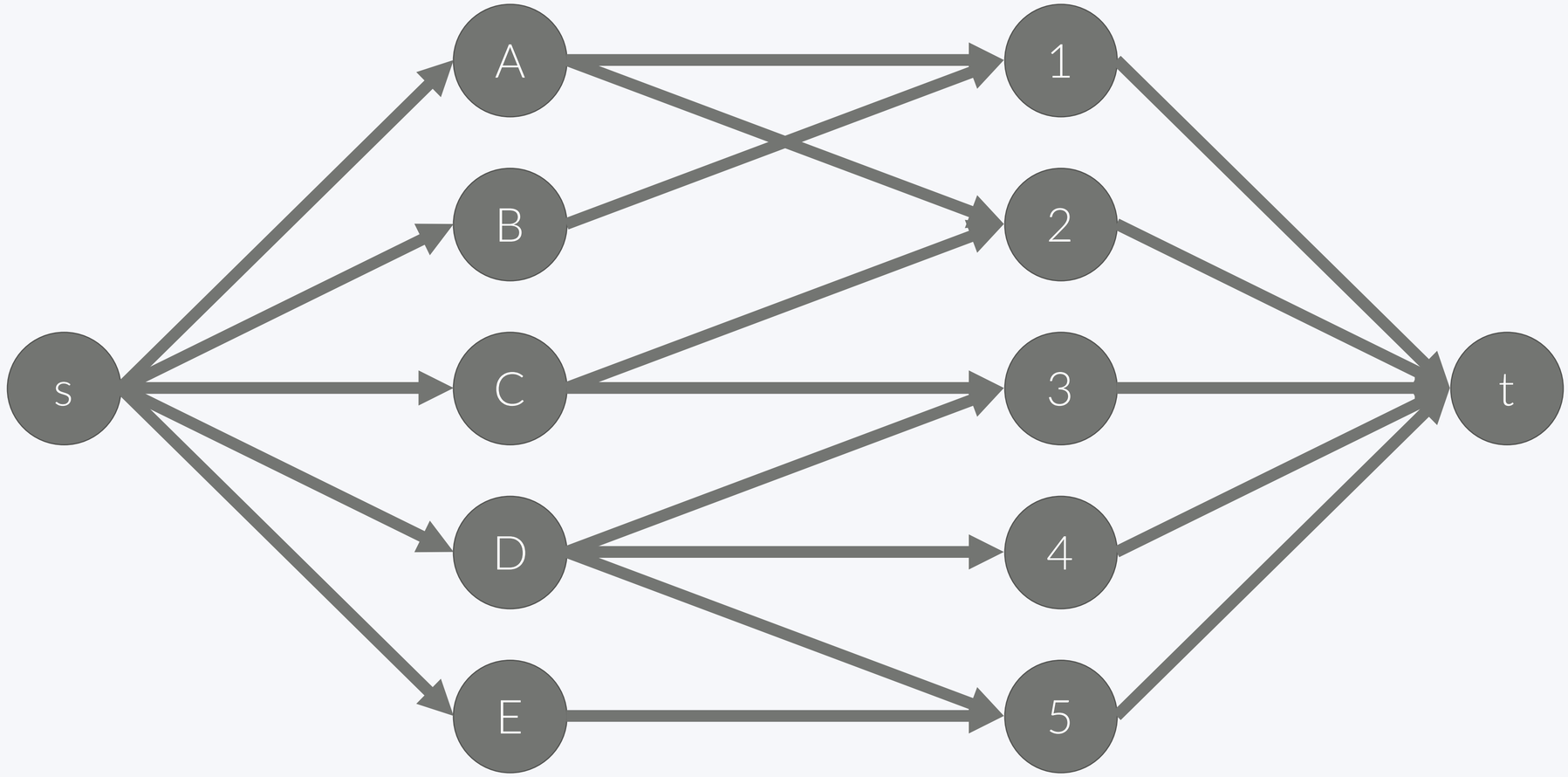
Bipartite Matching


```
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;
}
```

열혈강호

https://www.acmicpc.net/problem/11375

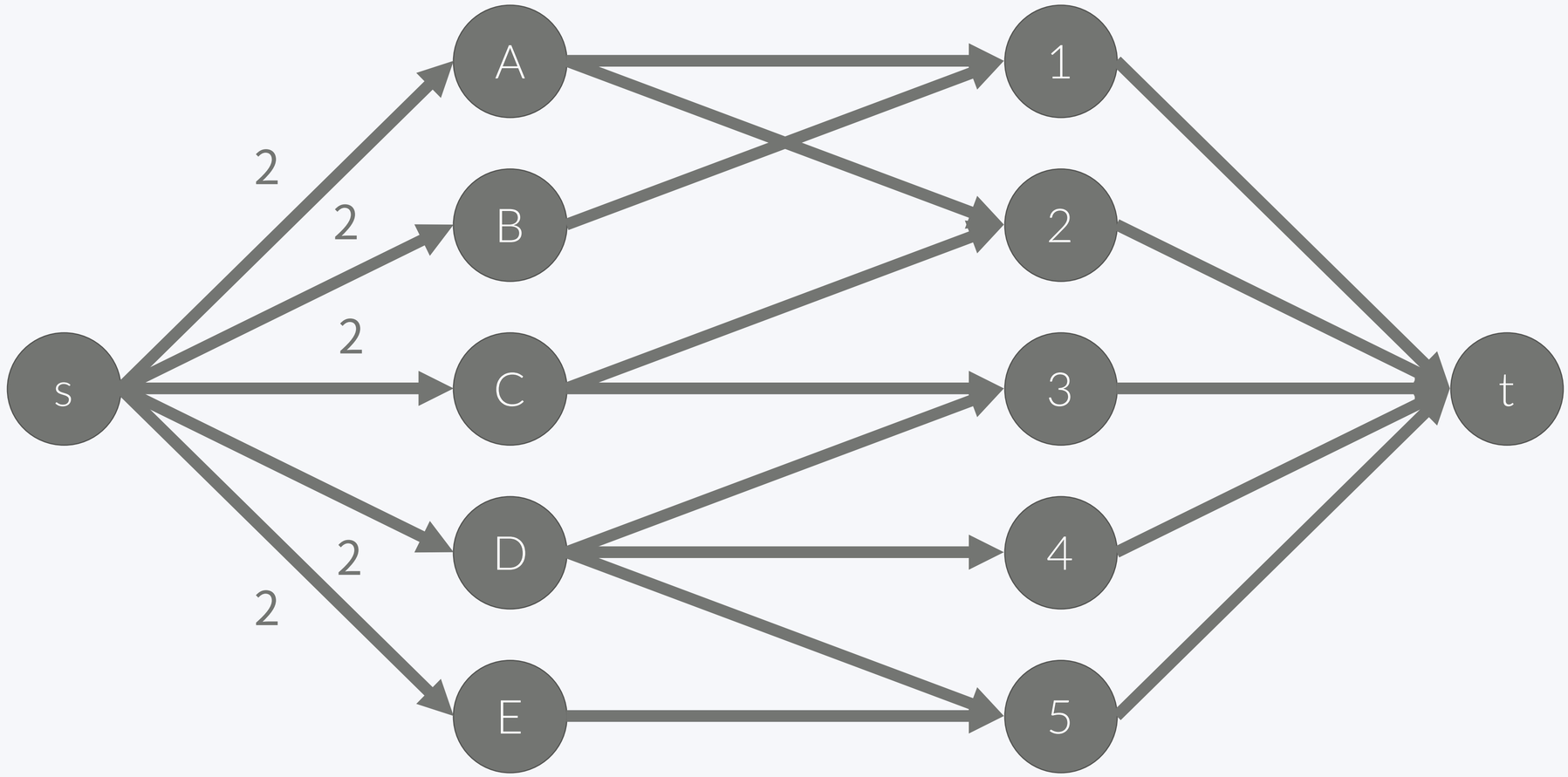
- https://gist.github.com/Baekjoon/0b8072965d77305b9beb



열혈강호 2

https://www.acmicpc.net/problem/11376

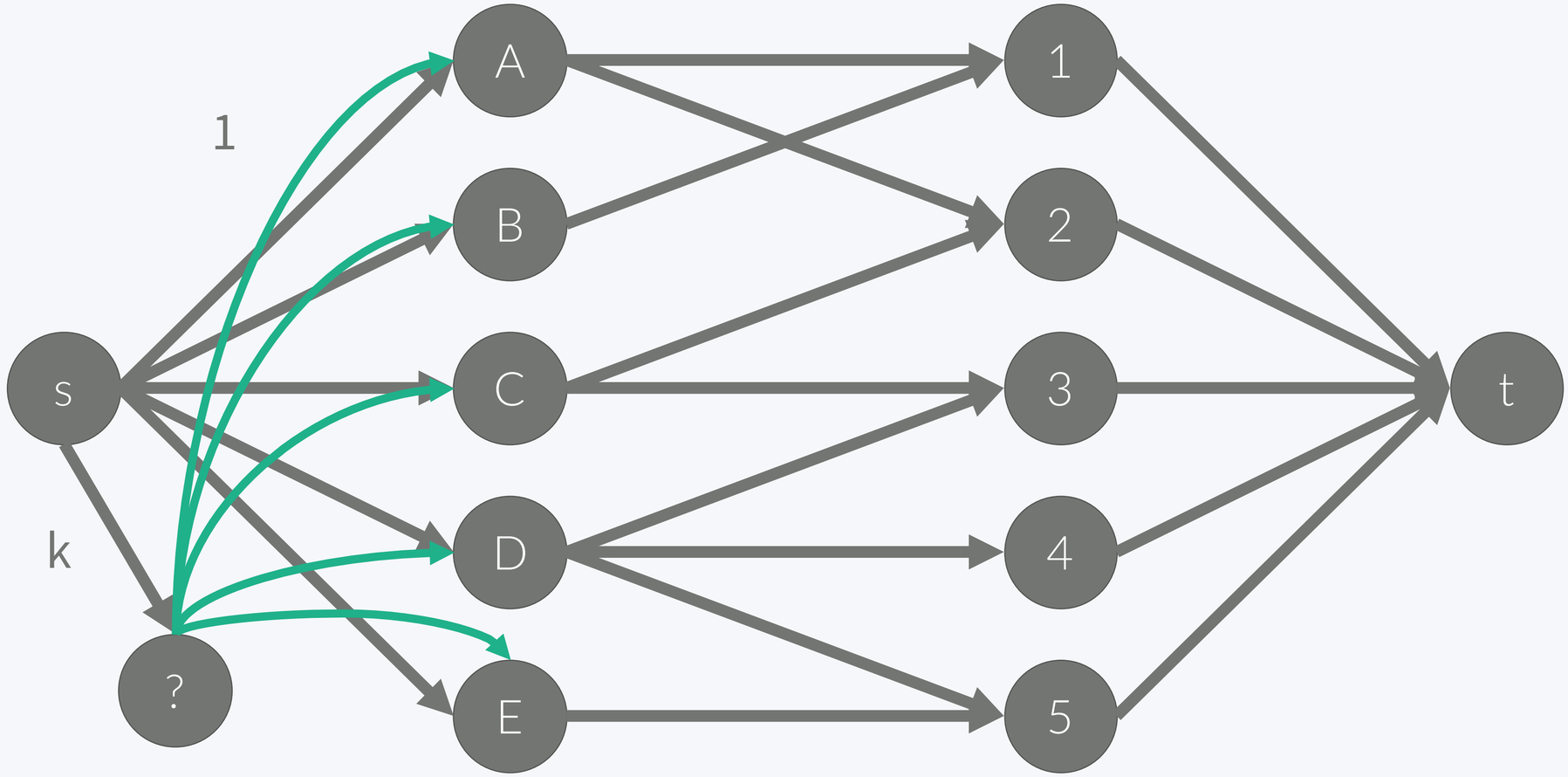
- https://gist.github.com/Baekjoon/01163c4a0b051fb476cd



열혈강호 3

https://www.acmicpc.net/problem/11377

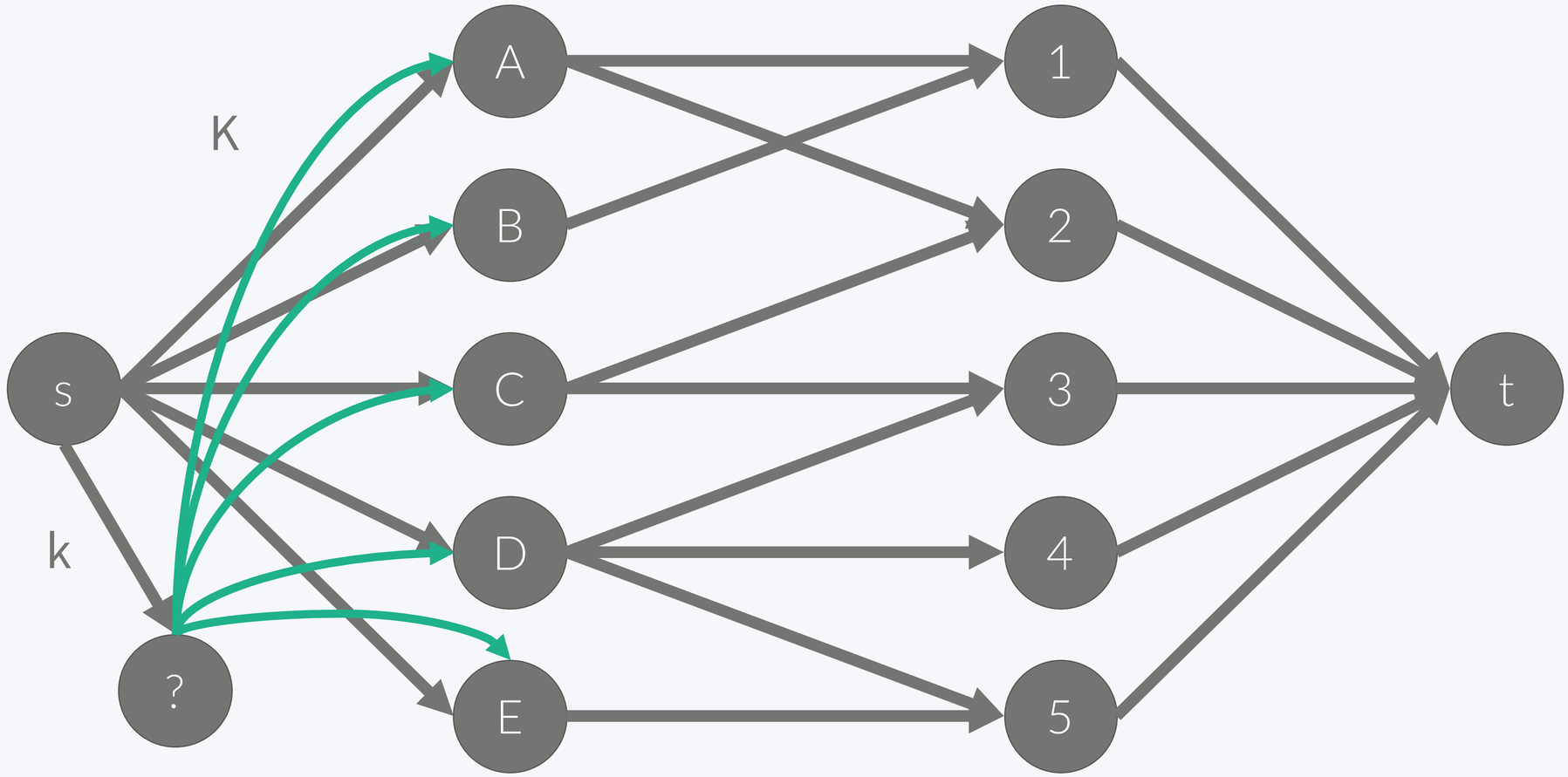
- https://gist.github.com/Baekjoon/8961653081d03e51fd99



열혈강호 4

https://www.acmicpc.net/problem/11378

- https://gist.github.com/Baekjoon/df740831fd0a657a01ac



축사 배정

https://www.acmicpc.net/problem/2188

- 열혈강호와 똑같은 문제
- 왼쪽: 소, 오른쪽: 축사

노트북의 주인을 찾아서

https://www.acmicpc.net/problem/1298

- 열혈강호와 똑같은 문제
- 왼쪽: 사람, 오른쪽: 노트북
- https://gist.github.com/Baekjoon/2742f4217e8efb72d1c1

소수 쌍

https://www.acmicpc.net/problem/1017

- 첫 번째 수와 짝지을 수를 미리 구한 다음에
- 나머지를 매칭한다
- 왼쪽: 홀수 오른쪽: 짝수
- https://gist.github.com/Baekjoon/4f6fefbe83302fa6807b

상어의 저녁식사

https://www.acmicpc.net/problem/1671

- 상어는 서로를 먹는다
- A의 크기, 속도, 지능이 B의 크기, 속도, 지능보다 크거나 같으면
- A는 B를 먹을 수 있다
- 한 상어가 최대 2마리 상어만 먹을 수 있다

- 살아남을 수 있는 상어의 수

상어의 저녁식사

https://www.acmicpc.net/problem/1671

- 상어는 서로를 먹는다
- A의 크기, 속도, 지능이 B의 크기, 속도, 지능보다 크거나 같으면
- A는 B를 먹을 수 있다
- 한 상어가 최대 2마리 상어만 먹을 수 있다

- 살아남을 수 있는 상어의 수 = N – 최대 매칭

상어의 저녁식사

https://www.acmicpc.net/problem/1671

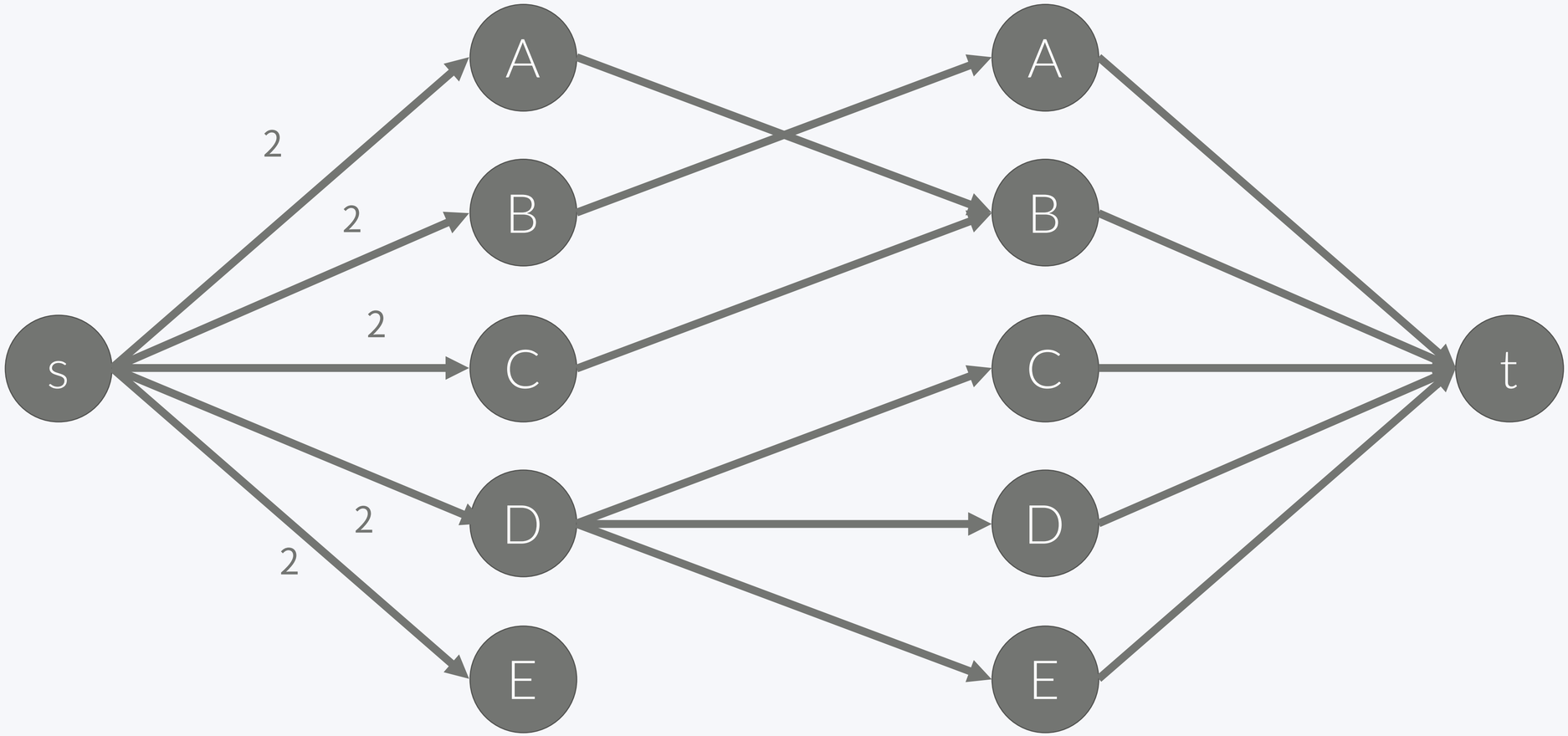
- 상어는 서로를 먹는다
- A의 크기, 속도, 지능이 B의 크기, 속도, 지능보다 크거나 같으면
- A는 B를 먹을 수 있다
- 한 상어가 최대 2마리 상어만 먹을 수 있다

- 살아남을 수 있는 상어의 수 = N – 최대 매칭
- 같은 경우에 서로를 잡아먹는 경우를 방지하기 위해서
- 같은 경우에는 i < j이면 잡아먹을 수 있다고 가정

상어의 저녁식사

https://www.acmicpc.net/problem/1671

- d



+이분 탐색

주차장

https://www.acmicpc.net/problem/1348

- R*C 격자
- C는 인접한 칸으로 1초에 1칸씩 이동
- 주차하는데 필요한 최소 시간
- 모든 차는 동시에 이동한다.

- .C.....P.X...
- XX.......X..P
- XX.....C.....

주차장

https://www.acmicpc.net/problem/1348

- 먼저, BFS로 모든 차와 주차장 사이의 쌍에 대해서 이동하는데 필요한 시간을 구한다.

- 그 다음, Binary Search를 이용해서 허용하는 최대 이동 시간을 결정하고, edge를 적절히 연결한 다음에 mathing의 개수가 차의 개수와 같은지 확인

주차장

https://www.acmicpc.net/problem/1348

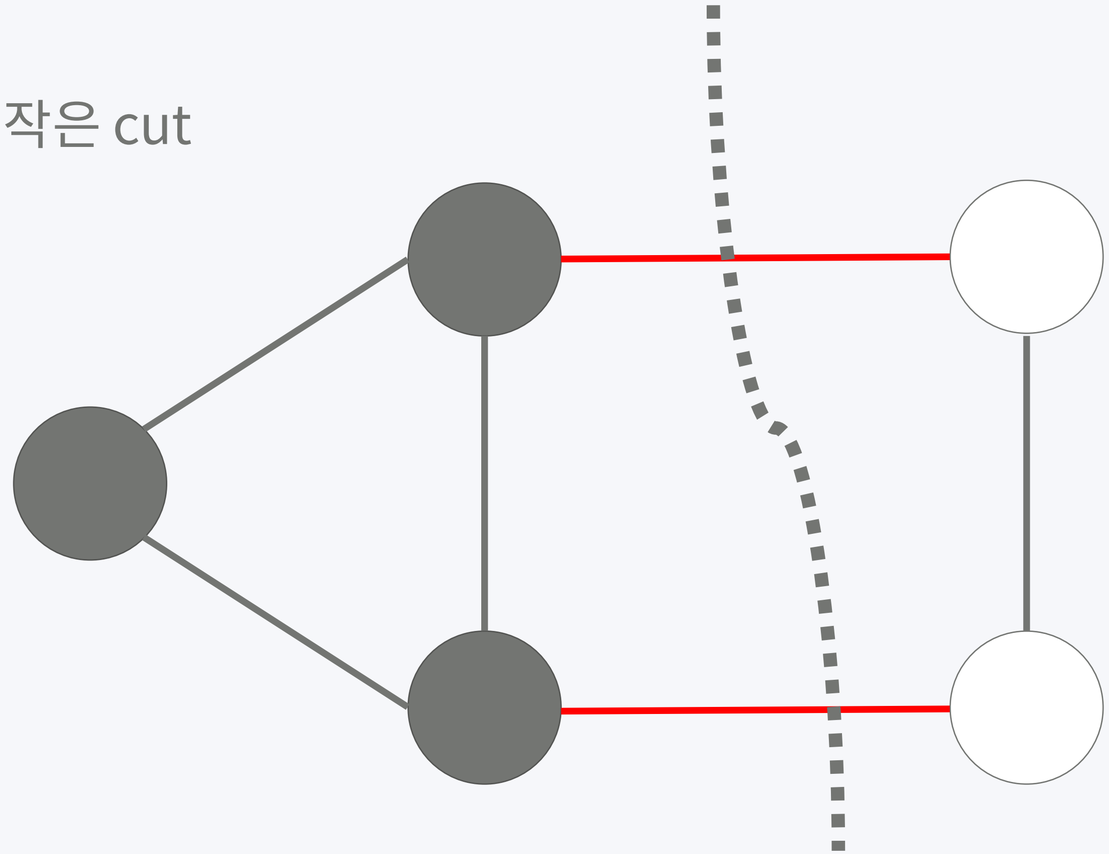
- https://gist.github.com/Baekjoon/4d3d07aa3c255b5baa6f

Min-cut

Minimum cut

Minimum Cut

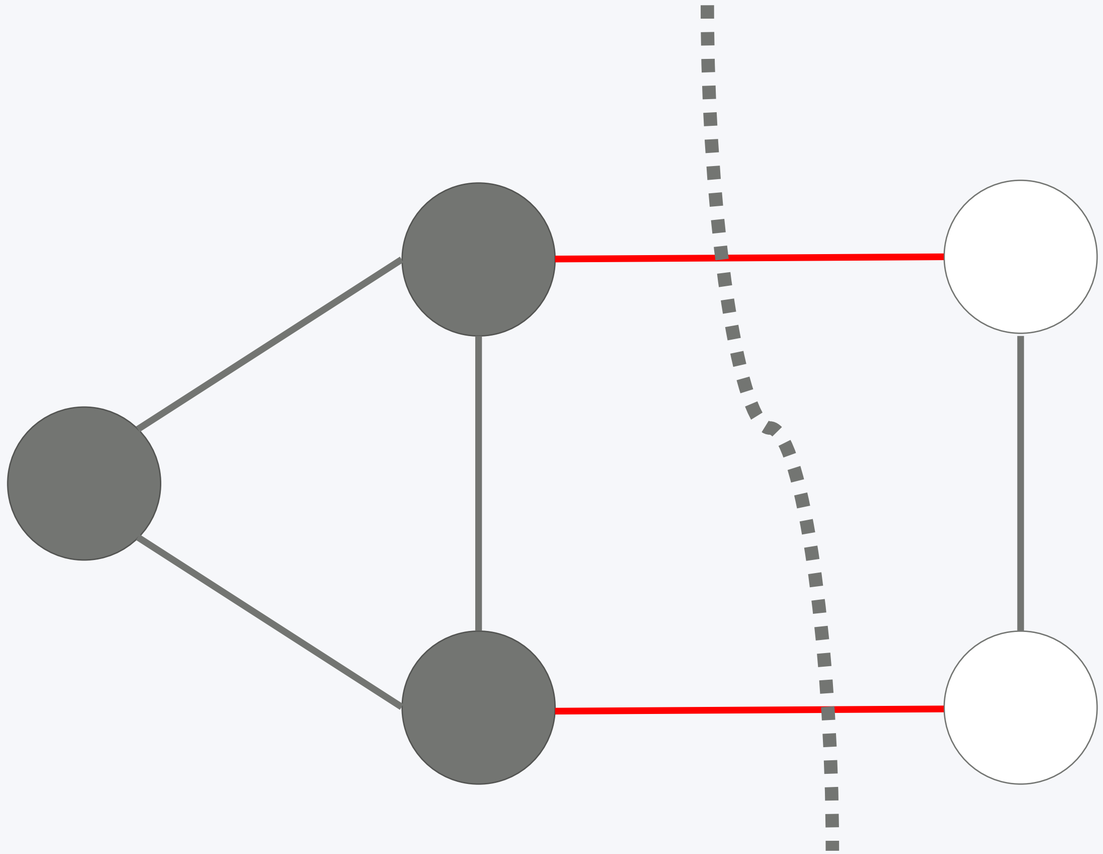
- Cut: 그래프를 2개의 서로다른 집합으로 나누는 것
- 두 집합을 A와 B라고 했을 때, 한 쪽 끝은 A에 다른 한 쪽 끝은 B에 있는 간선을 cut-set이라고 한다
- Minimum cut: 가장 작은 cut



Max-flow Min-cut Theorem

Max-flow Min-cut Theorem

- Flow network에서 min-cut은 max-flow와 같다.
- 여기서 min-cut은 source->sink로 흐르지 못하게 하기 위해 제거해야 하는 edge capacity 합의 최소값



학교 가지마!

https://www.acmicpc.net/problem/1420

- N*M 크기의 도시
- 빈 칸: .
- 벽: #
- 도현: K
- 학교 H

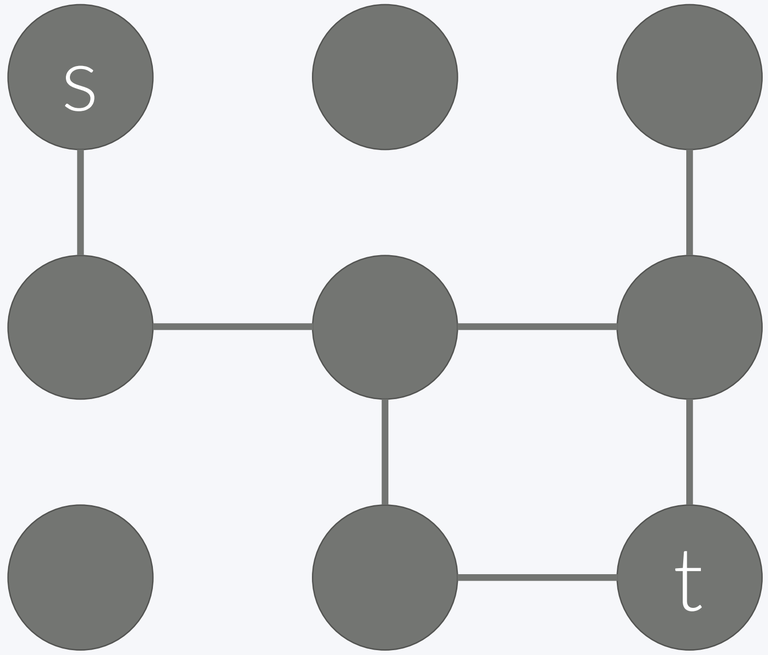
- 빈 칸을 적절히 벽으로 바꿔서 학교로 가지 못하게 하는 문제
- 최소 개수를 구해야 한다

학교 가지마!

https://www.acmicpc.net/problem/1420

- 도시를 플로우 네트워크로 바꾸고, min-cut을 구하는 문제이다.

K	#	
#		H

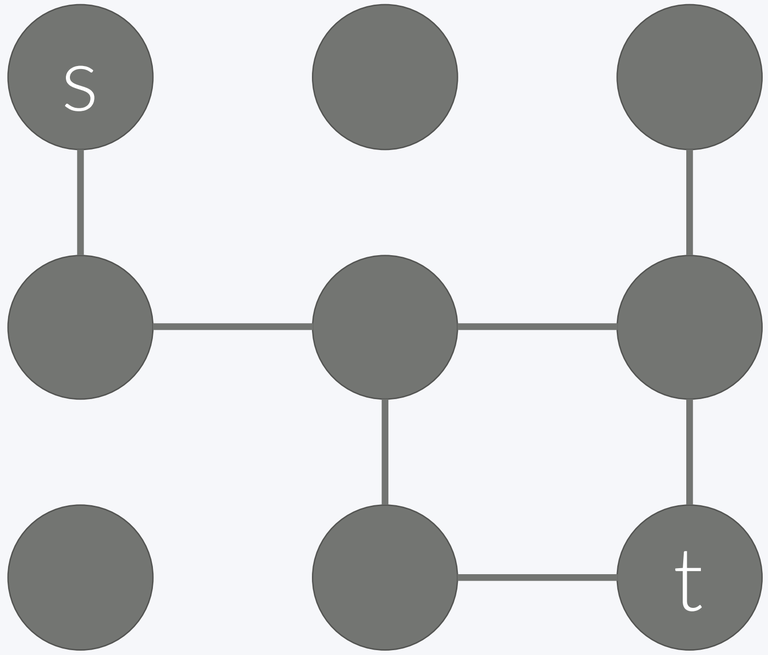


학교 가지마!

https://www.acmicpc.net/problem/1420

- 그런데, 도시와 도시를 연결하는 edge를 cut하면 안된다
- 도시에 벽을 놓는 것이지, 도시와 도시 사이를 막는 것이 아니기 때문

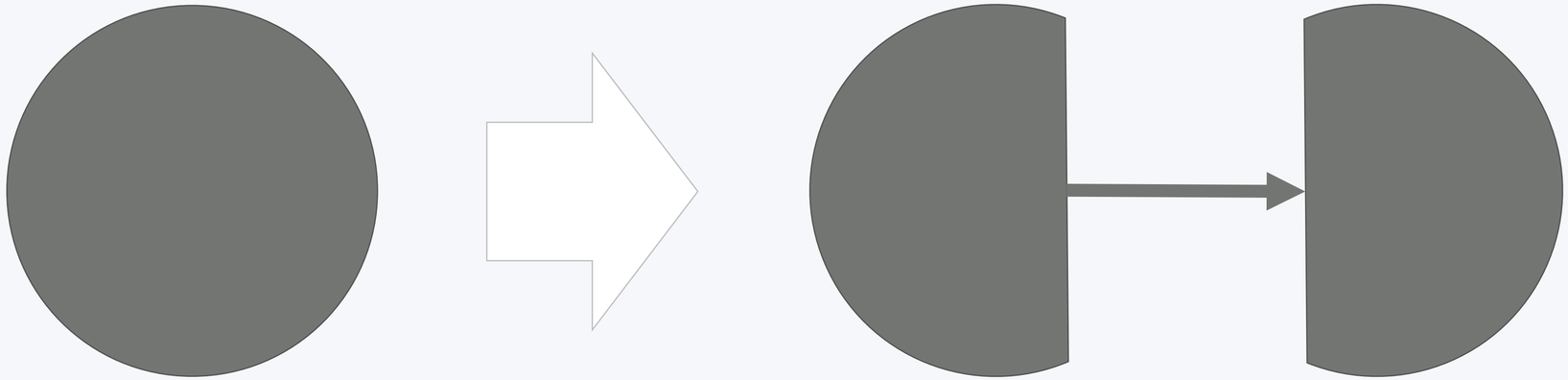
K	#	
#		H



학교 가지마!

https://www.acmicpc.net/problem/1420

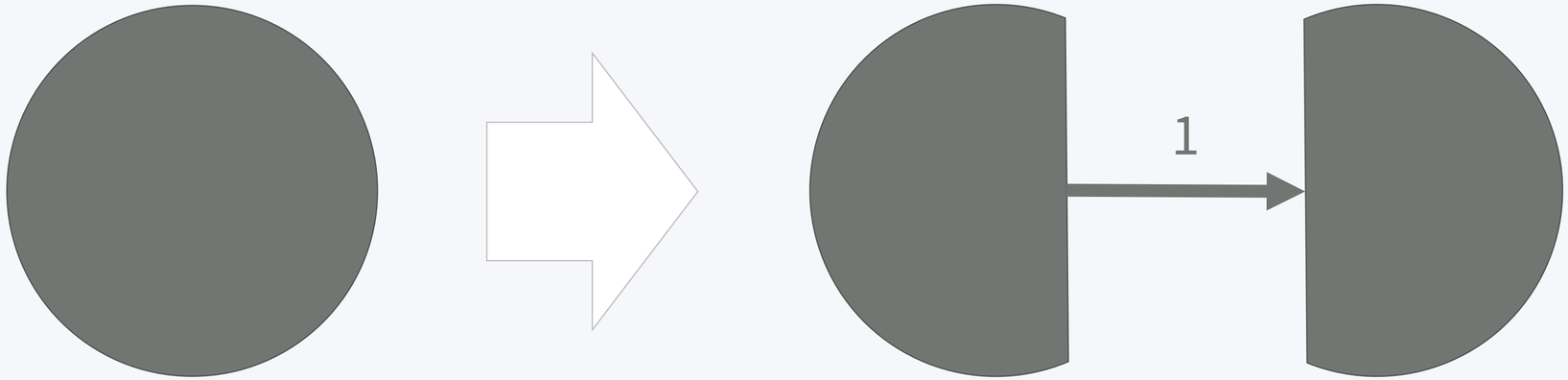
- 각 칸을 둘로 나눈다
- capacity는 몇?



학교 가지마!

https://www.acmicpc.net/problem/1420

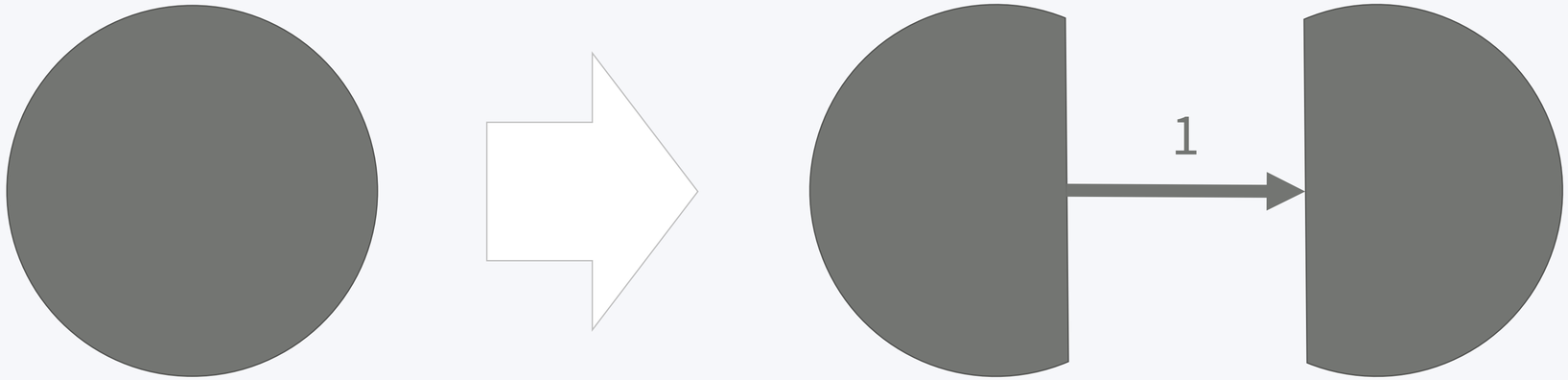
- 각 칸을 둘로 나눈다
- capacity는 몇? 1



학교 가지마!

https://www.acmicpc.net/problem/1420

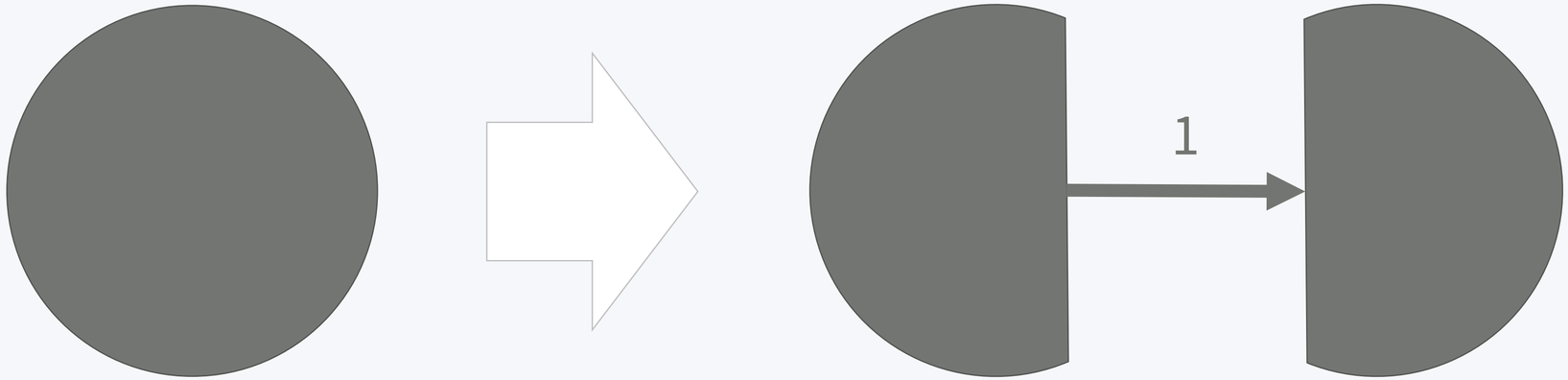
- 정점 X를 Xin과 Xout 으로 나누고, Xin -> Xout은 1로 연결
- X와 Y를 이동할 수 있으면, Xout -> Yin, Yout -> Xin 을 연결
- 이 때 capacity?



학교 가지마!

https://www.acmicpc.net/problem/1420

- https://gist.github.com/Baekjoon/236b37c8d011e3d79bd8



최소 버텍스 커버

최소 버텍스 커버

Minium Vertex Cover

- Vertex Cover: 정점 집합 S가 있을 때, 모든 간선은 양 끝점중 하나가 S에 포함되어야 함
- Minimum Vertex Cover: 최소값

König's theorem

Minium Vertex Cover

- Bipartite Graph에서
- Maximum Matching은
- Minimum Vertex Cover와 같다

돌멩이 제거

https://www.acmicpc.net/problem/1867

- N행 N열에 K개의 돌멩이가 있다
- 격자 한 칸에 들어가 있고, 두 개가 한 칸에 들어간 경우는 없다
- 한 행 또는 한 열을 따라서 직선으로 움직이면서 돌멩이를 모두 줍는다
- 최소 몇 번이나 달려야 하는가?

돌멩이 제거

https://www.acmicpc.net/problem/1867

- 이분 그래프를 만든다

- 왼쪽: 행
- 오른쪽: 열
- i행 j열에 돌멩이가 있으면
- 왼쪽 i -> 오른쪽 j를 연결

돌멩이 제거

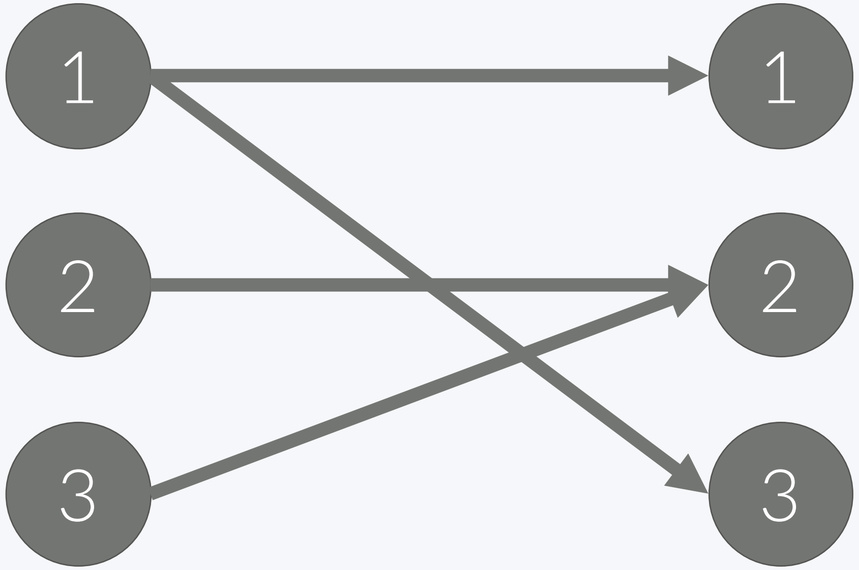
https://www.acmicpc.net/problem/1867

- 3 4
- 1 1
- 1 3
- 2 2
- 3 2

돌		돌
	돌	
	돌	

돌멩이 제거

https://www.acmicpc.net/problem/1867



돌		돌
	돌	
	돌	

돌멩이 제거

https://www.acmicpc.net/problem/1867

- https://gist.github.com/Baekjoon/9481c68db12c6c9f8786

게시판 구멍 막기

https://www.acmicpc.net/problem/2414

- N*M 모양의 게시판에 구멍이 뚫려있다
- 폭이 1인 테이프로 막으려고 한다
- 길이는 무한하지만
- 끊어내는 횟수를 최소로
- 아래 예제 정답: 4

```
*.*.
.***
***.
..*.
```

게시판 구멍 막기

https://www.acmicpc.net/problem/2414

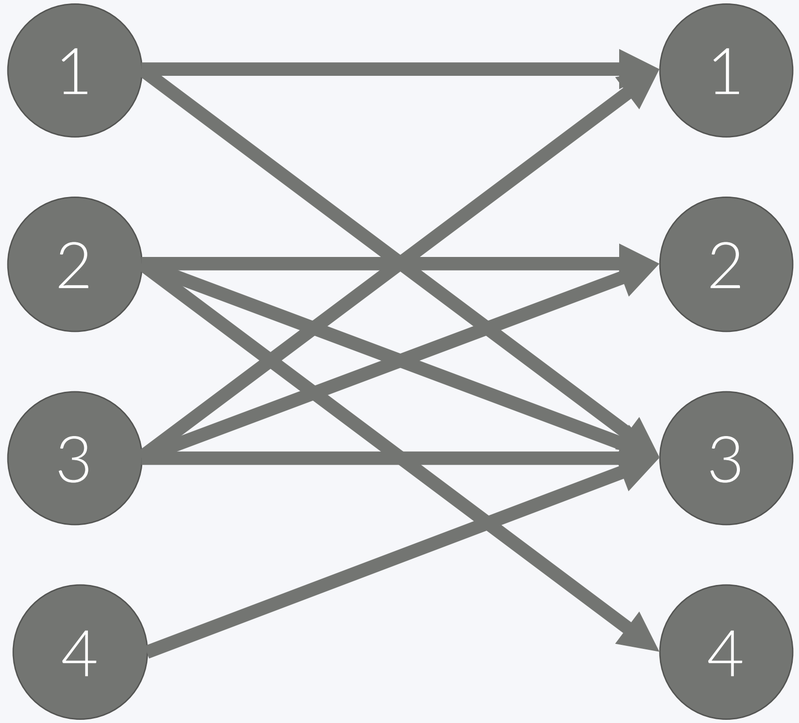
- 돌멩이 줍기와 똑같은 문제다
- 하지만 구멍만 막아야 한다.
- 따라서 그래프를 조금 수정해야 한다

게시판 구멍 막기

https://www.acmicpc.net/problem/2414

- 돌멩이 줍기와 똑같은 문제다. 이 그래프가 아니다.

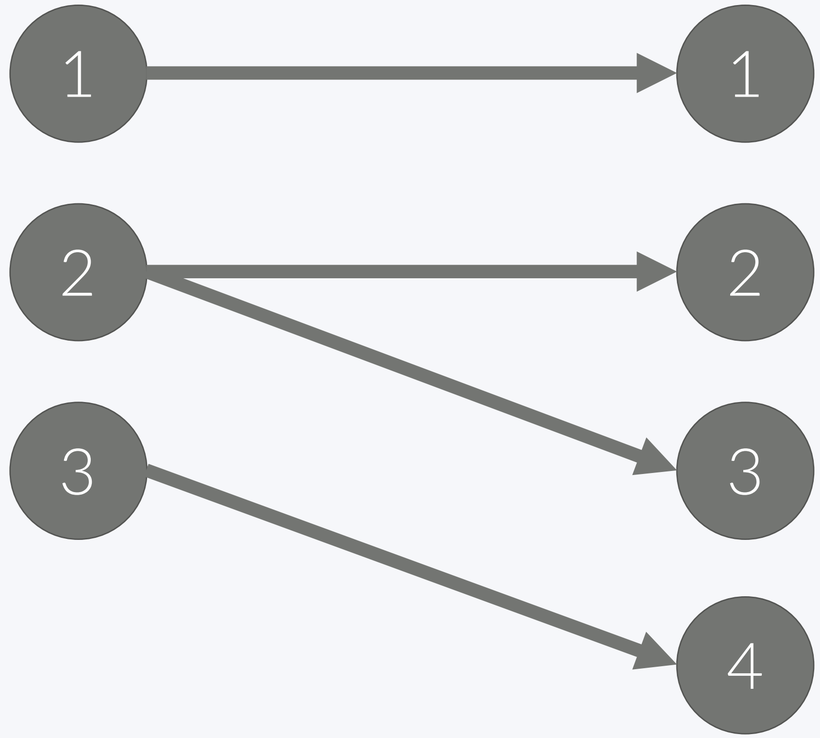
#		#	
	#	#	#
#	#	#	
		#	



게시판 구멍 막기

https://www.acmicpc.net/problem/2414

- 간단하게 1x8 크기의 경우를 생각
- .#.##.#.
- 1행은 한 번에 모두 막을 수가 없음
- 인접한 # 끼리 그룹으로 나눠야 함
- .1.22.3.
- 이렇게 한 행을 여러 그룹으로 나눠야 함
- 열을 기준으로도 나눌 수 있음
- .1.23.4.



게시판 구멍 막기

https://www.acmicpc.net/problem/2414

#		#	
	#	#	#
#	#	#	
		#	

1		2	
	3	3	3
4	4	4	
		5	

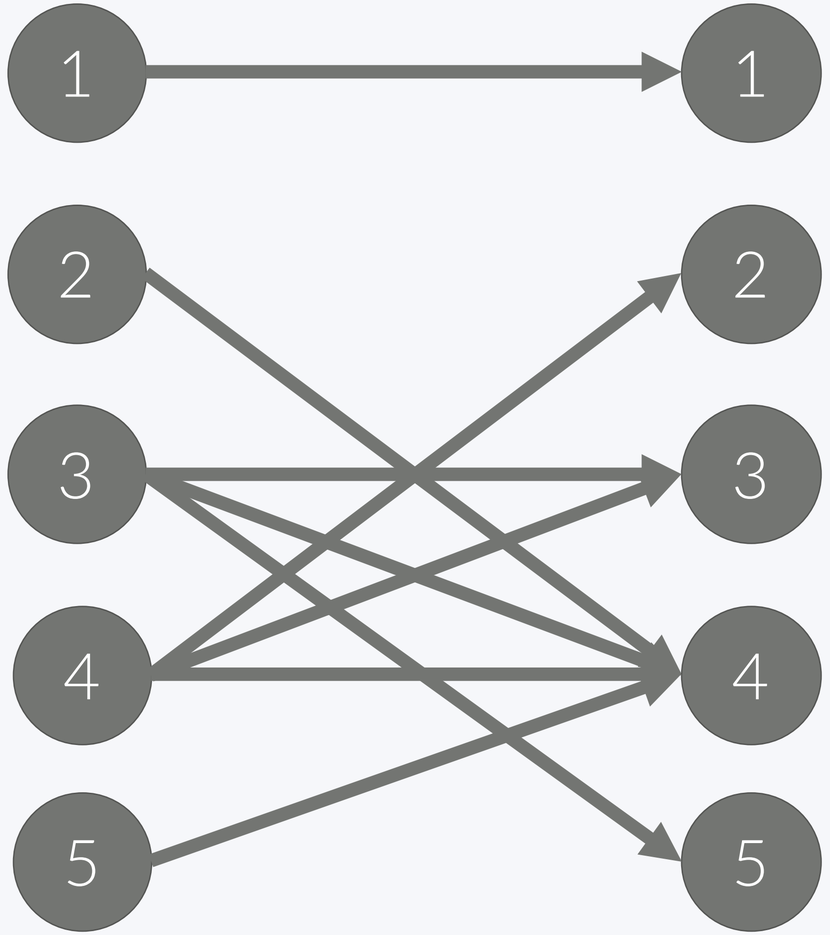
1		4	
	3	4	5
2	3	4	
		4	

게시판 구멍 막기

https://www.acmicpc.net/problem/2414

1		2	
	3	3	3
4	4	4	
		5	

1		4	
	3	4	5
2	3	4	
		4	



게시판 구멍 막기

https://www.acmicpc.net/problem/2414

- https://gist.github.com/Baekjoon/d49ba57cf43ea887a79d

N-Rook

https://www.acmicpc.net/problem/1760

- 돌멩이, 게시판 구멍 막기와 같은 최소 버텍스 커버 문제다

- 게시판 구멍 막기와 같이 벽이 아닌 칸에 대해서 번호를 매겨준 다음에
- 구덩이가 아니 칸 (i, j)를 연결해주면 된다.

N-Rook

https://www.acmicpc.net/problem/1760

- https://gist.github.com/Baekjoon/9adfec8ca58b8360f904

비숍2

https://www.acmicpc.net/problem/2570

- N-Rook과 같지만, 행/열 대신 /와 \로 문제를 풀 수 있다
- https://gist.github.com/Baekjoon/58436830ac3c7febbef0

최대 독립 집합

최대 독립 집합

Maximum Independent Set

- 그래프 G의 정점 집합
- 집합에 포함된 모든 정점끼리를 연결하는 Edge가 없어야 함
- 이 때 최대
- 이 문제는 NP-Hard

최대 독립 집합

Maximum Independent Set

- Independent Set의 Complement는 Vertex Cover다
- Maximum Independent Set의 Complement는 Minimum Vertex Cover이다
- 그래프가 이분그래프인 경우 Minimum Vertex Cover는 Maximum Flow다
- flow로 풀 수 있는 문제이다

컨닝 2

https://www.acmicpc.net/problem/11014

- 각 칸을 vertex 로 생각하고, 앉을 수 없는 칸을 edge로 연결하자
- 문제는 이 그래프에서 Maximum Independent Set을 찾는 문제

컨닝 2

https://www.acmicpc.net/problem/11014

- column 의 홀 짝을 기준으로 왼쪽과 오른쪽을 나눌 수 있다.
- https://gist.github.com/Baekjoon/e27c28b4aac30a4ba527

문제 풀이

등번호

https://www.acmicpc.net/problem/1733

- 티셔츠는 뒤집어 입을 수 있다
- 안쪽 면과 바깥쪽 면에 적힌 수는 다르다
- 한쪽 면을 선택해서 입어야 한다
- 같은 번호 티셔츠를 입지 않게 알려주자

등번호

https://www.acmicpc.net/problem/1733

- 왼쪽 사람
- 오른쪽 등번호

- x의 티셔츠에 y와 z가 써있으면

- x – y, x – z 를 이어준다

- 이 문제는 무엇이 매칭되는지를 찾아야 한다

등번호

https://www.acmicpc.net/problem/1733

- 코드에서 pred는 오른쪽 vertex가 무엇과 매치되는지를 나타낸다
- 응용해서 matchL과 matchR을 작성할 수 있다.
- https://gist.github.com/Baekjoon/689e5e3340bc7abe6b7e

흔한 수열 문제

https://www.acmicpc.net/problem/2787

- 길이가 N인 수열 A
- 1~N까지 수가 한 번씩 등장한다

- 설명 형식
- 1 x y v: x번째 부터 y번째 수 중에서 제일 큰 값은 v
- 2 x y v: x번째 부터 y번째 수 중에서 제일 작은 값은 v

흔한 수열 문제

https://www.acmicpc.net/problem/2787

- 왼쪽: ai
- 오른쪽: 수 j
- edge (ai, j)는 ai자리에 수 j가 들어갈 수 있다는 (ai = j) 라는 의미이다.
- 불가능한 조합을 빼주고 매칭을 돌리면 된다.
- 1 x y v인 경우
- x ≤k ≤ y 이고 v+1 ≤ l ≤ n 인 모든 (ak, l) edge를 제거한다
- 2 x y v인 경우
- x ≤k ≤ y 이고 1 ≤ l ≤ v-1 인 모든 (ak, l) edge를 제거한다

흔한 수열 문제

https://www.acmicpc.net/problem/2787

- 이렇게 풀면 틀린다.
- edge(x,y) 가 있을 때
- x번째 숫자가 y라면
- x는 y가 포함된 모든 구간의 교집합이어야 하고, y는 x에서 가능한 모든 숫자에 포함되어 있어야 하기 때문

흔한 수열 문제

https://www.acmicpc.net/problem/2787

- https://gist.github.com/Baekjoon/7a04a0b212ff56f7ec76

Crucial Links

https://www.acmicpc.net/problem/5651

- 어떤 flow network에서
- crucial link의 개수를 세는 문제

- 어떤 edge의 capacity를 1 줄였을 때
- maximum flow가 1 감소한다면
- 그 edge는 crucial link다

Crucial Links

https://www.acmicpc.net/problem/5651

- 일단 flow network를 구한 다음에
- 각각의 edge (u, v)에 대해서
- u -> v로 가는 augmenting path를 찾는다
- 존재하지 않으면 그 edge는 crucial link다

- 존재하는 경우에는 (u, v)의 flow를 1 감소시키고 찾은 augmenting path에 flow를 1 증가하면 되기 때문

Crucial Links

https://www.acmicpc.net/problem/5651

- https://gist.github.com/Baekjoon/ba9ad7a176dd71cf6b1f

돼지 잡기

https://www.acmicpc.net/problem/1658

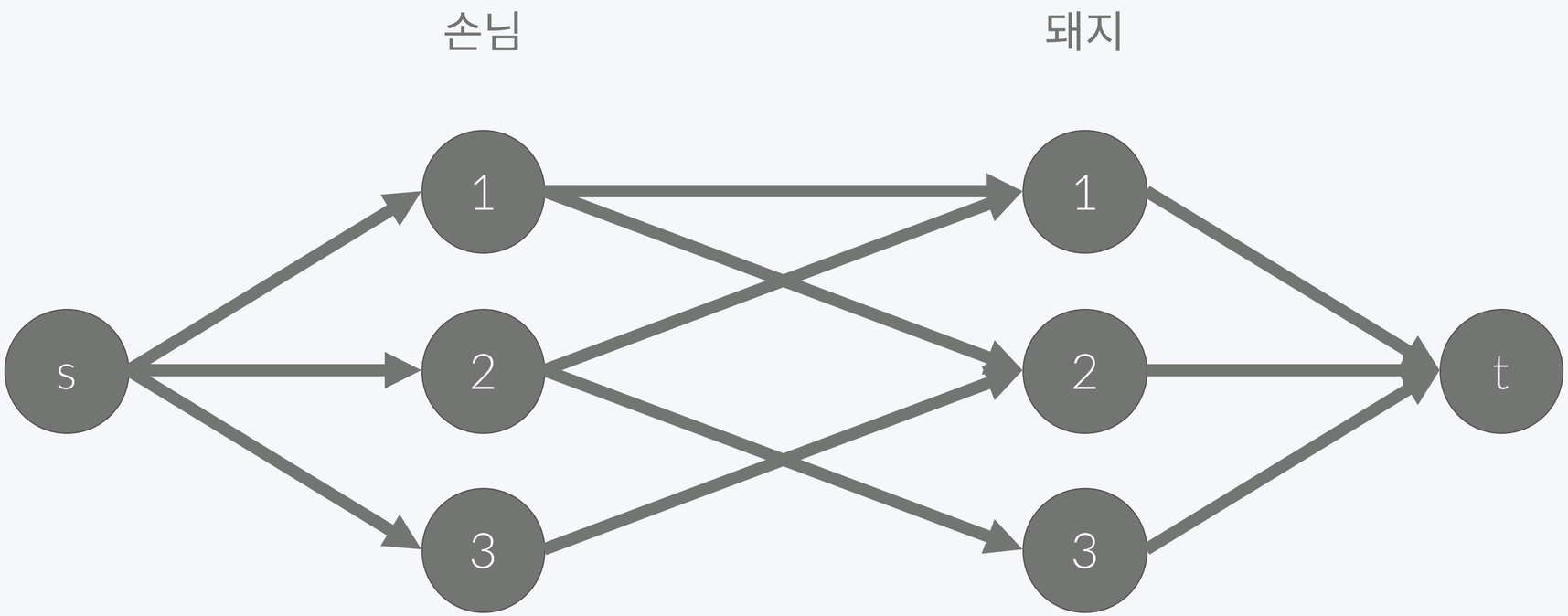
- 돼지 우리 M개
- 손님 N명
- 손님은 하루에 한 명씩 온다.
- 우리를 열고 자신이 원하는 만큼 돼지를 사간다

1. 손님이 도착해서 가지고 있는 열쇠로 열 수 있는 모든 우리들을 연다.
2. 손님에게 몇몇 돼지들을 판다. (손님이 원하는 이상의 돼지를 팔 순 없지만 그 이하로는 팔 수 있다.)
3. 종혁이는 팔고 남은 돼지들을 현재 열려져 있는 우리들을 상대로 재분배 할 수 있다.

돼지 잡기

https://www.acmicpc.net/problem/1658

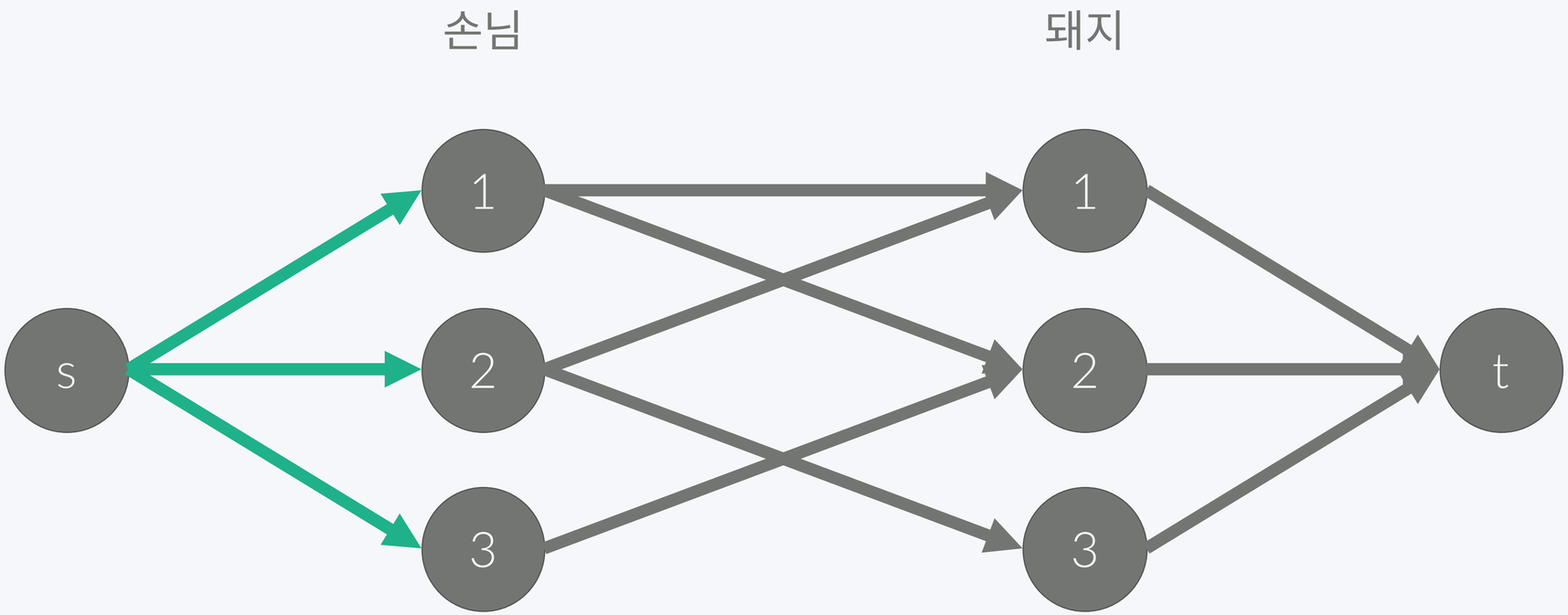
- 왼쪽: 손님, 오른쪽: 돼지 우리로 이분그래프를 만들고
- Maximum Flow가 답이 된다.



돼지 잡기

https://www.acmicpc.net/problem/1658

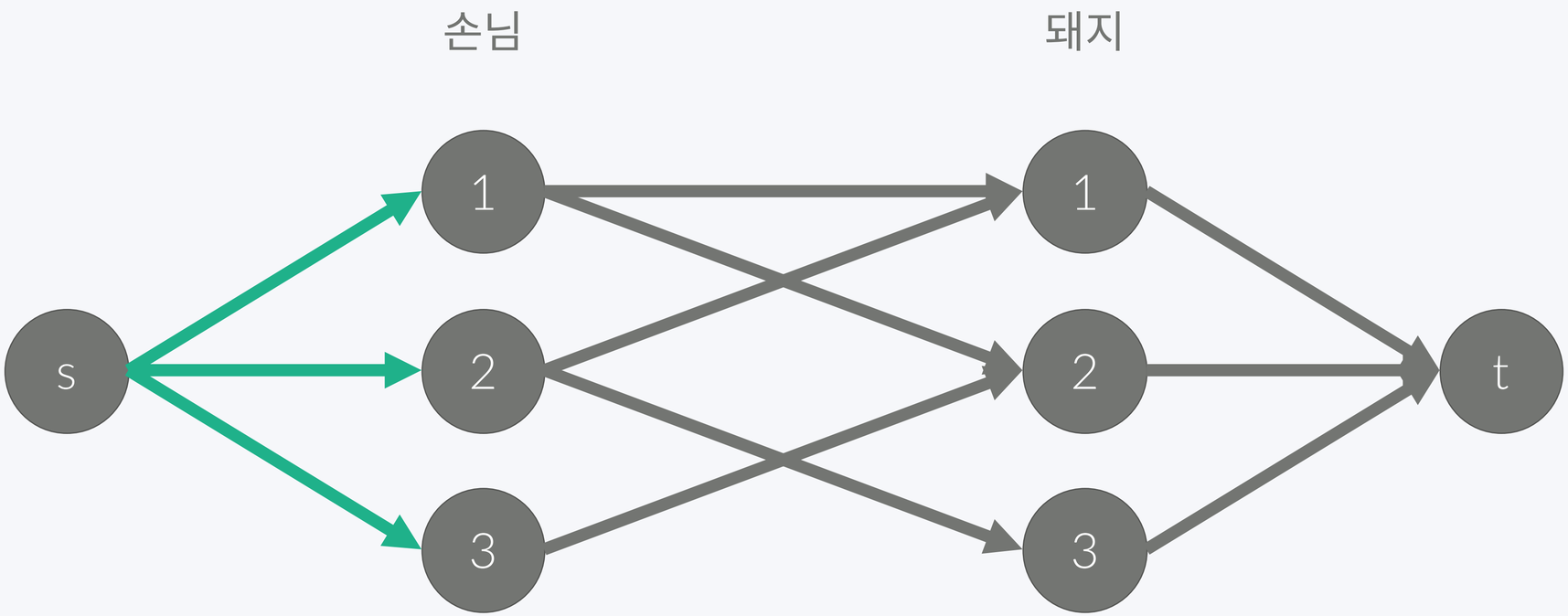
- capacity?



돼지 잡기

https://www.acmicpc.net/problem/1658

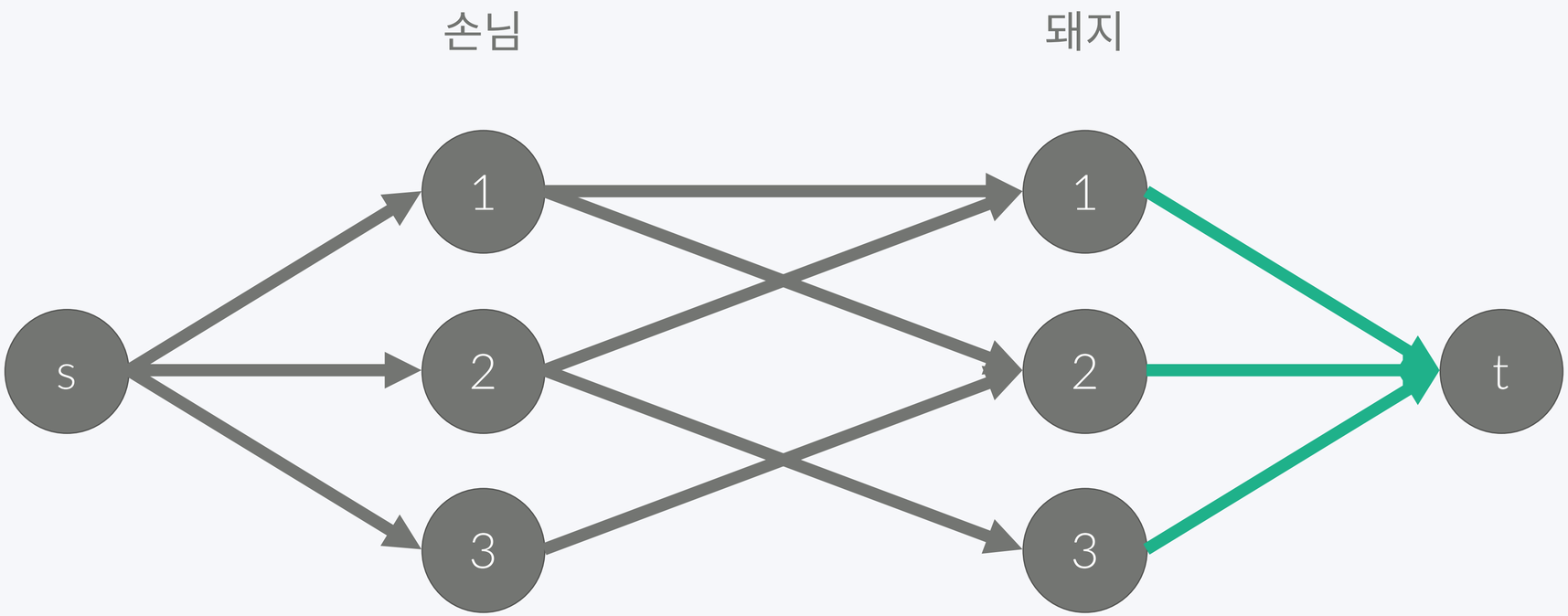
- capacity = 각 손님이 사려고 하는 돼지의 수



돼지 잡기

https://www.acmicpc.net/problem/1658

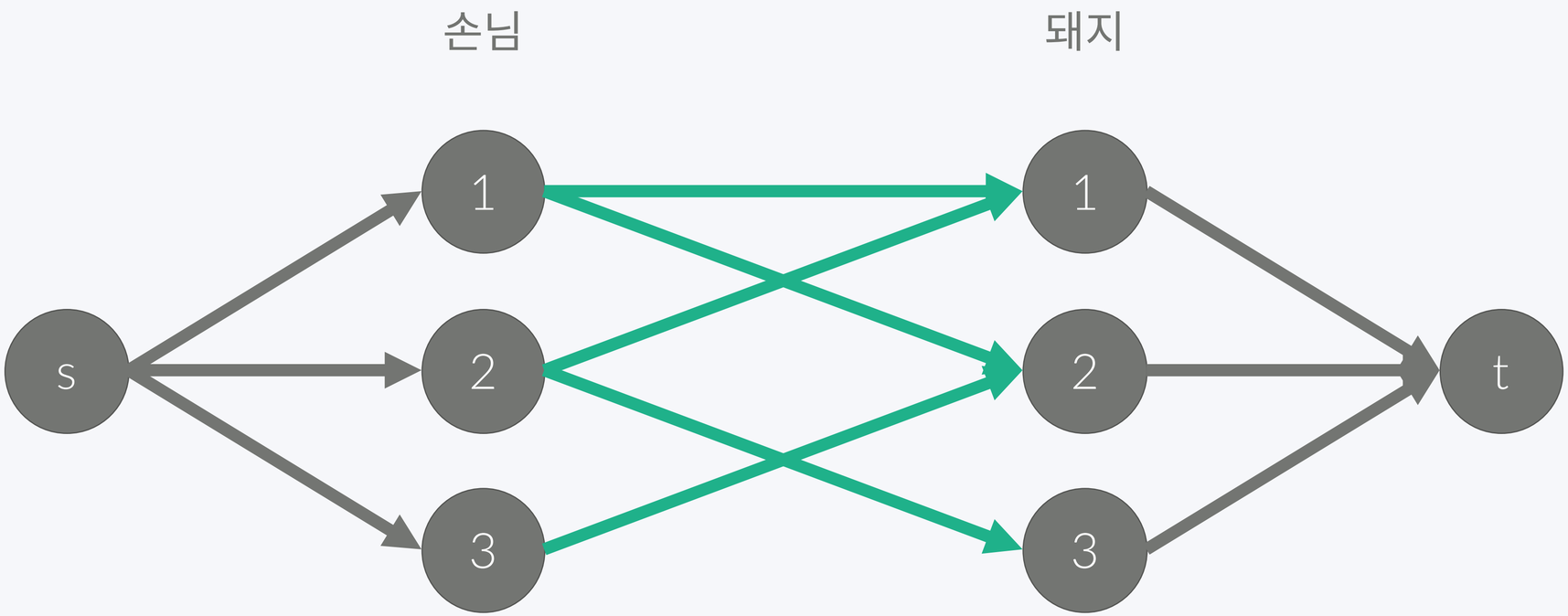
- 각 우리에 들어있는 돼지의 수



돼지 잡기

https://www.acmicpc.net/problem/1658

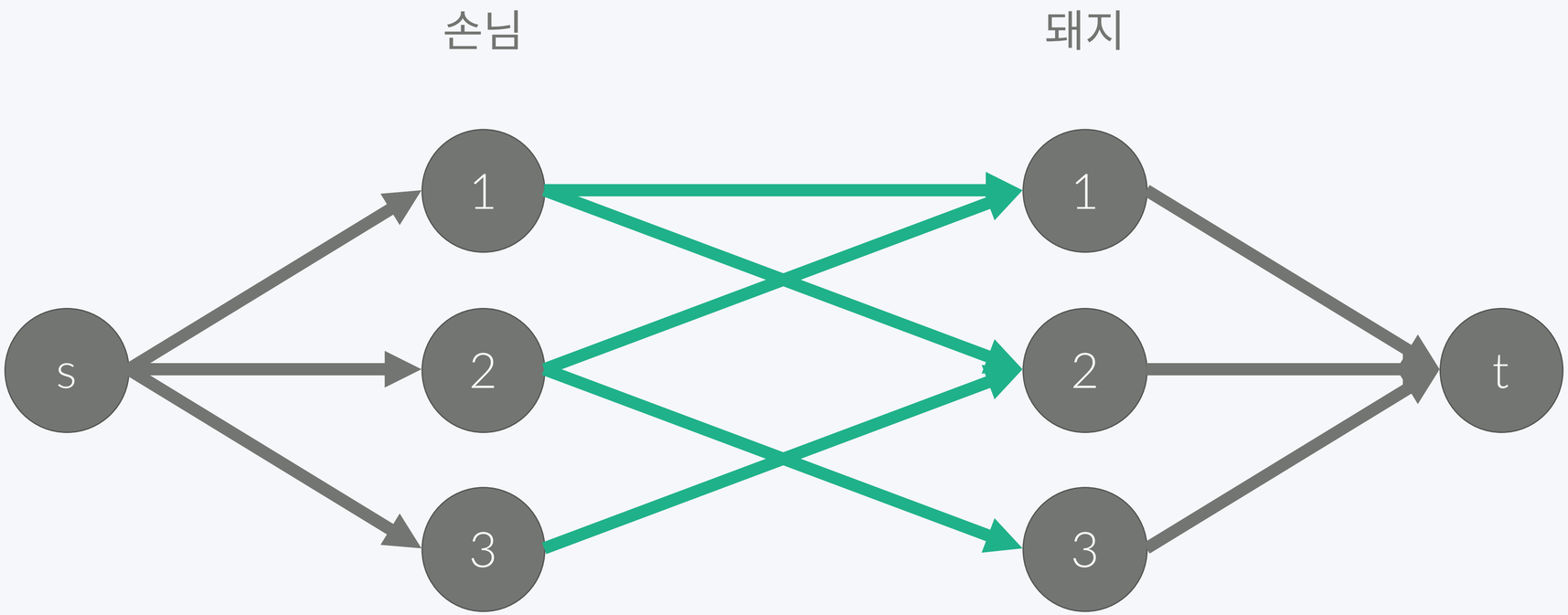
- 손님과 돼지는 언제 연결해야 할까?



돼지 잡기

https://www.acmicpc.net/problem/1658

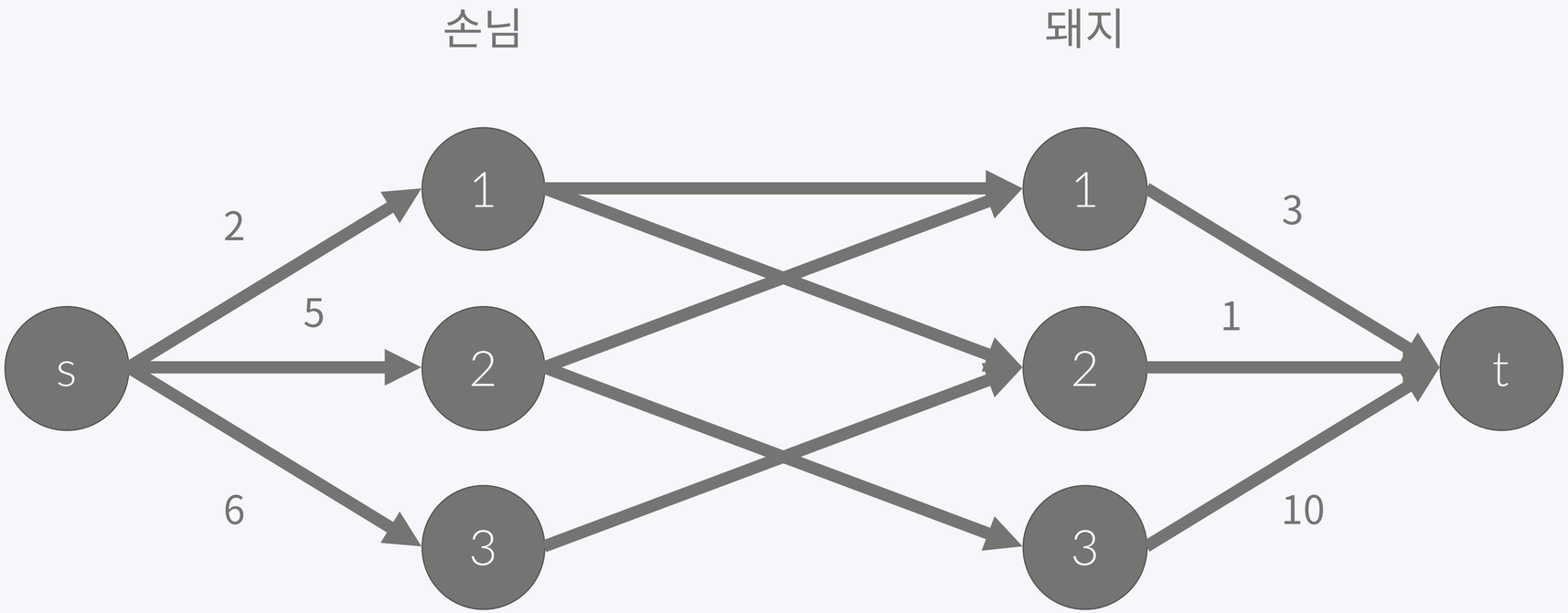
- 그 손님이 열쇠를 가지고 있을 때 (capacity: infinity)



돼지 잡기

https://www.acmicpc.net/problem/1658

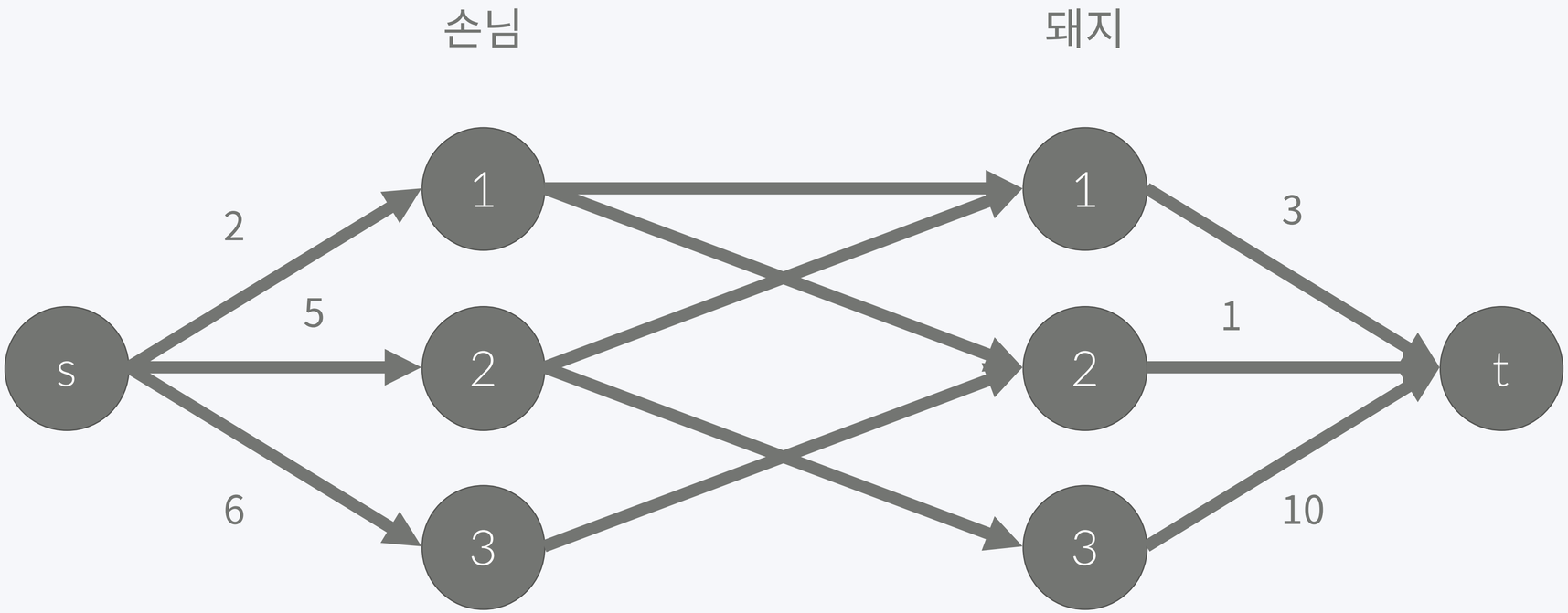
- 예제 그림



돼지 잡기

https://www.acmicpc.net/problem/1658

- 이렇게 풀면 3번 조건을 고려하지 않은 그래프이다
- 종혁이는 팔고 남은 돼지들을 현재 열려져 있는 우리들을 상대로 재분배 할 수 있다.

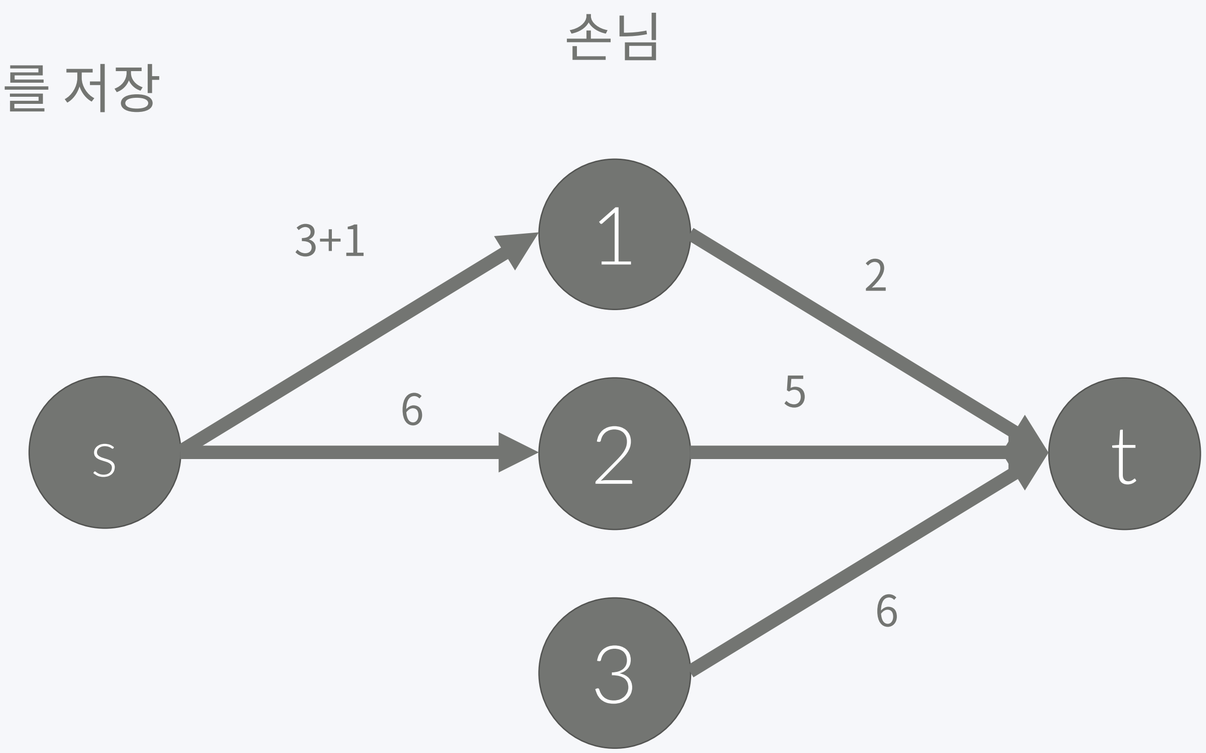


돼지 잡기

https://www.acmicpc.net/problem/1658

- 그래프를 다시 그린다.

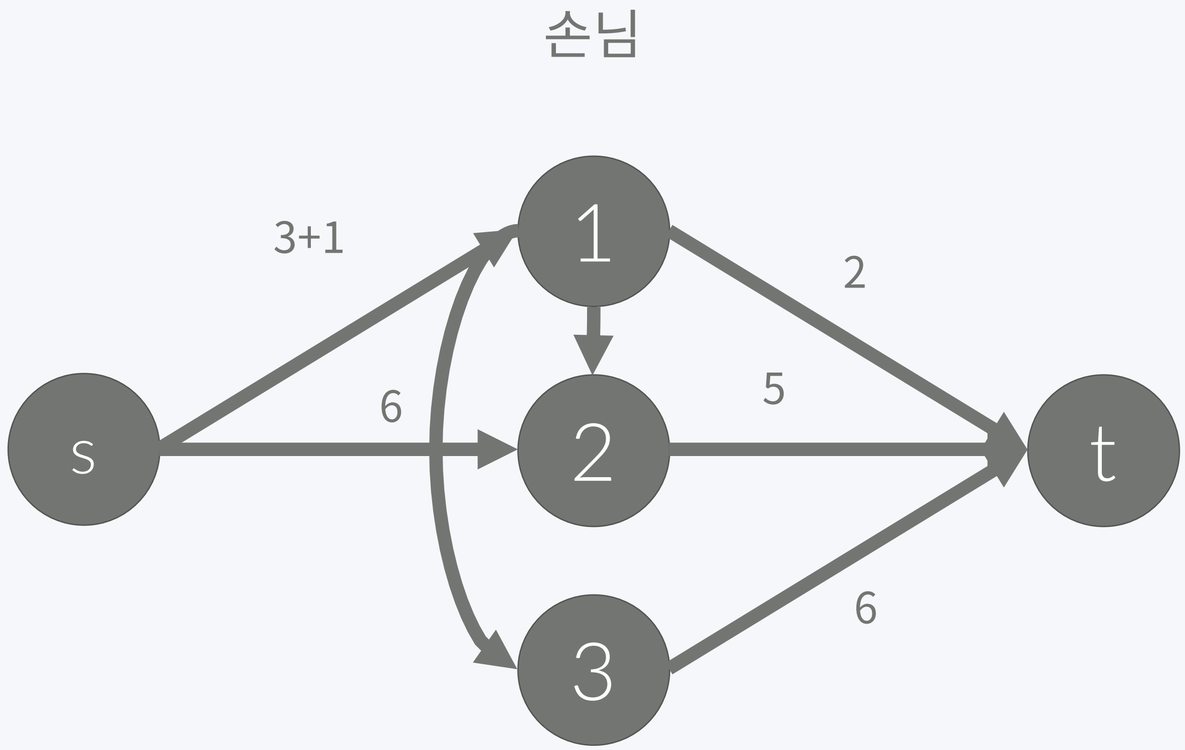
- 각 우리별로 열쇠를 가지고 있는 손님의 번호를 저장
- 첫 번째로 우리를 여는 사람에게
- 해당하는 우리에 들어있는 돼지의 수를
- capacity로 간선을 만들어 준다
- 1번 우리: 1 2
- 2번 우리: 1 3
- 3번 우리 2



돼지 잡기

https://www.acmicpc.net/problem/1658

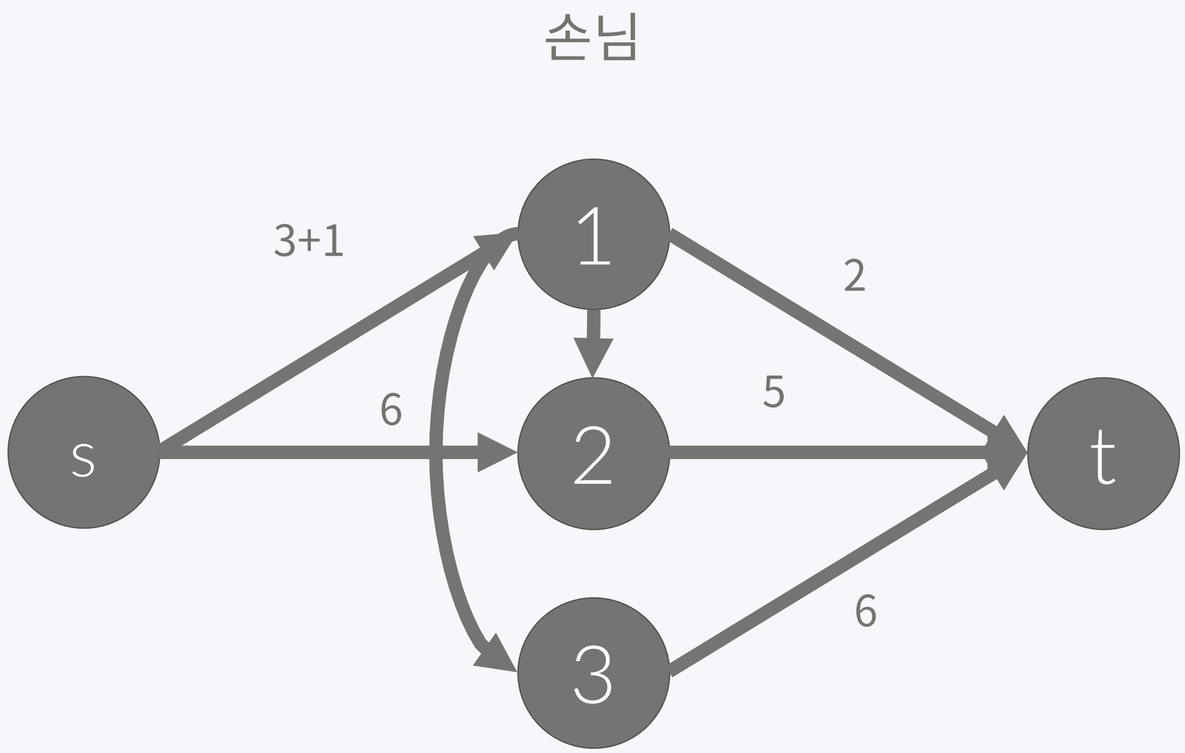
- 그 다음, 같은 우리 열쇠를 가지고 있는 사람끼리는
- 서로 돼지를 공유할 수 있기 때문에 edge를 연결해 준다
- 1번 우리: 1 2
- 2번 우리: 1 3
- 3번 우리 2



돼지 잡기

https://www.acmicpc.net/problem/1658

- https://gist.github.com/Baekjoon/90081103386ac1638d5b



Avoiding the Apocalypse

https://www.acmicpc.net/problem/10319

- 좀비로 변하지 않기 위해
- s분 이내에 병원에 도착해야 한다
- 최대 몇 명이 병원에 도착할 수 있는지 구하는 문제
- 사람 1명이 아니고 여러 명이다

Avoiding the Apocalypse

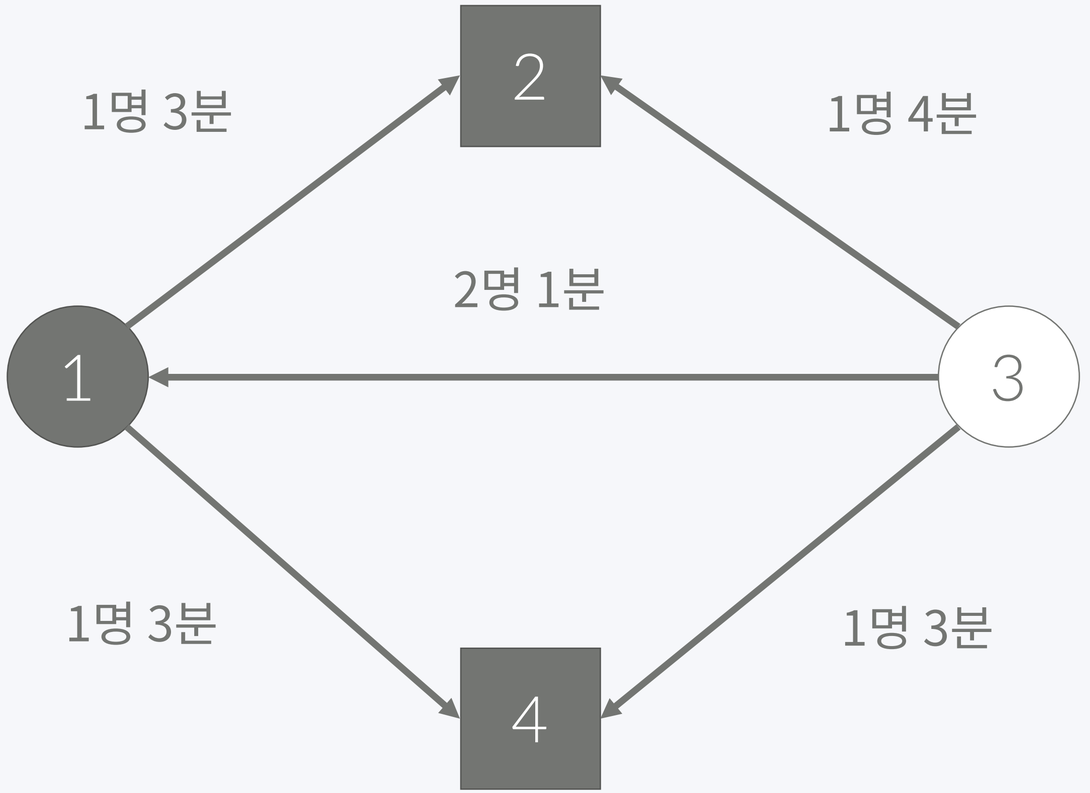
https://www.acmicpc.net/problem/10319

- Vertex: n개
- Edge: m개
- 시작 위치: i
- 사람의 수: g명
- 시간 제한: s분
- 병원의 개수: m
- 도로의 개수 r
- 도로 정보: a b p t
- a -> b로 가는 도로이고, 1분에 p명이 새로 들어갈 수 있으며, 지나가는데 필요한 시간은 t초
- 정점 위에 서 있어도 됨. 정점은 제한 없음

Avoiding the Apocalypse

https://www.acmicpc.net/problem/10319

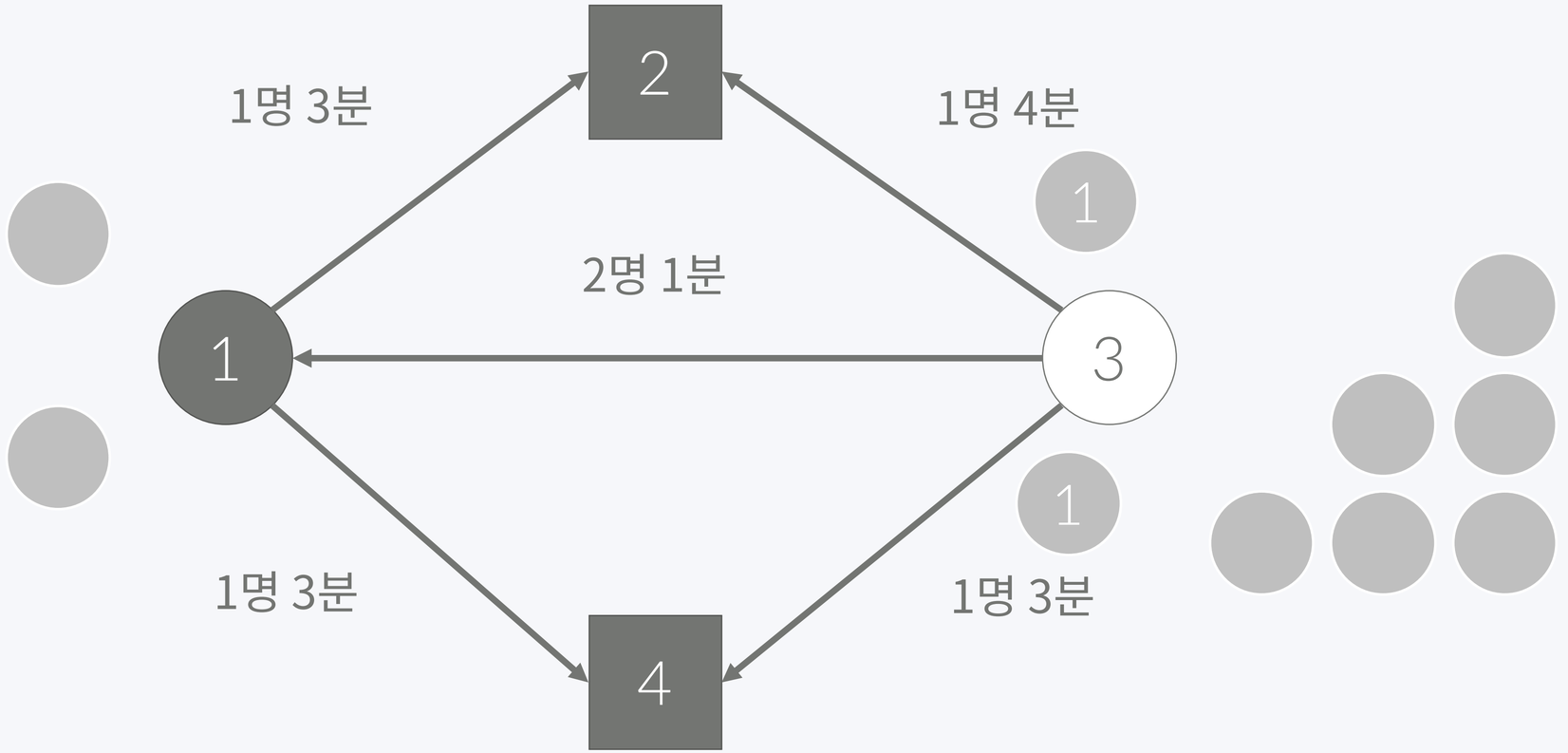
```
4
3 10 5
2
2
4
5
1 2 1 3
3 2 1 4
3 1 2 1
1 4 1 3
3 4 1 3
```



Avoiding the Apocalypse

https://www.acmicpc.net/problem/10319

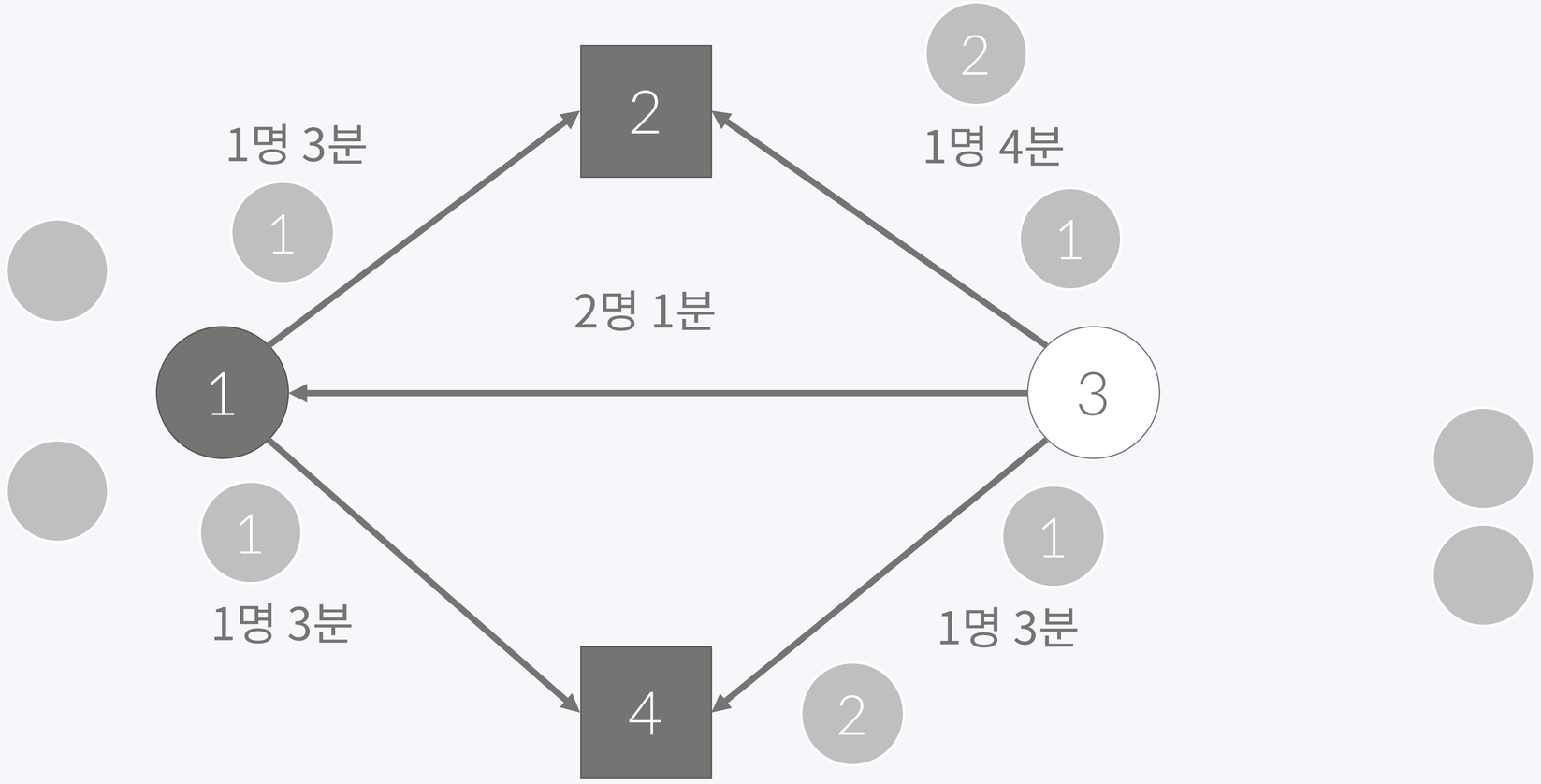
- 1분



Avoiding the Apocalypse

https://www.acmicpc.net/problem/10319

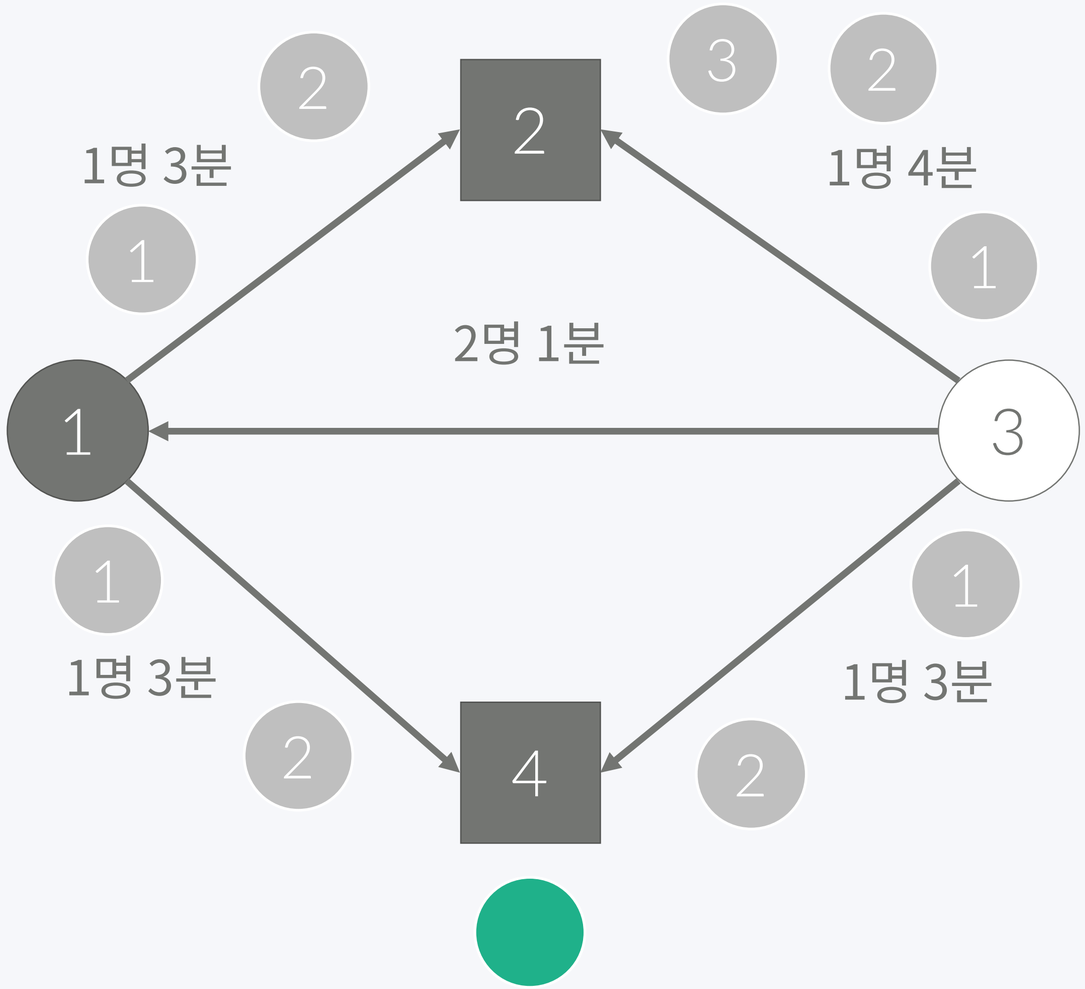
- 2분



Avoiding the Apocalypse

https://www.acmicpc.net/problem/10319

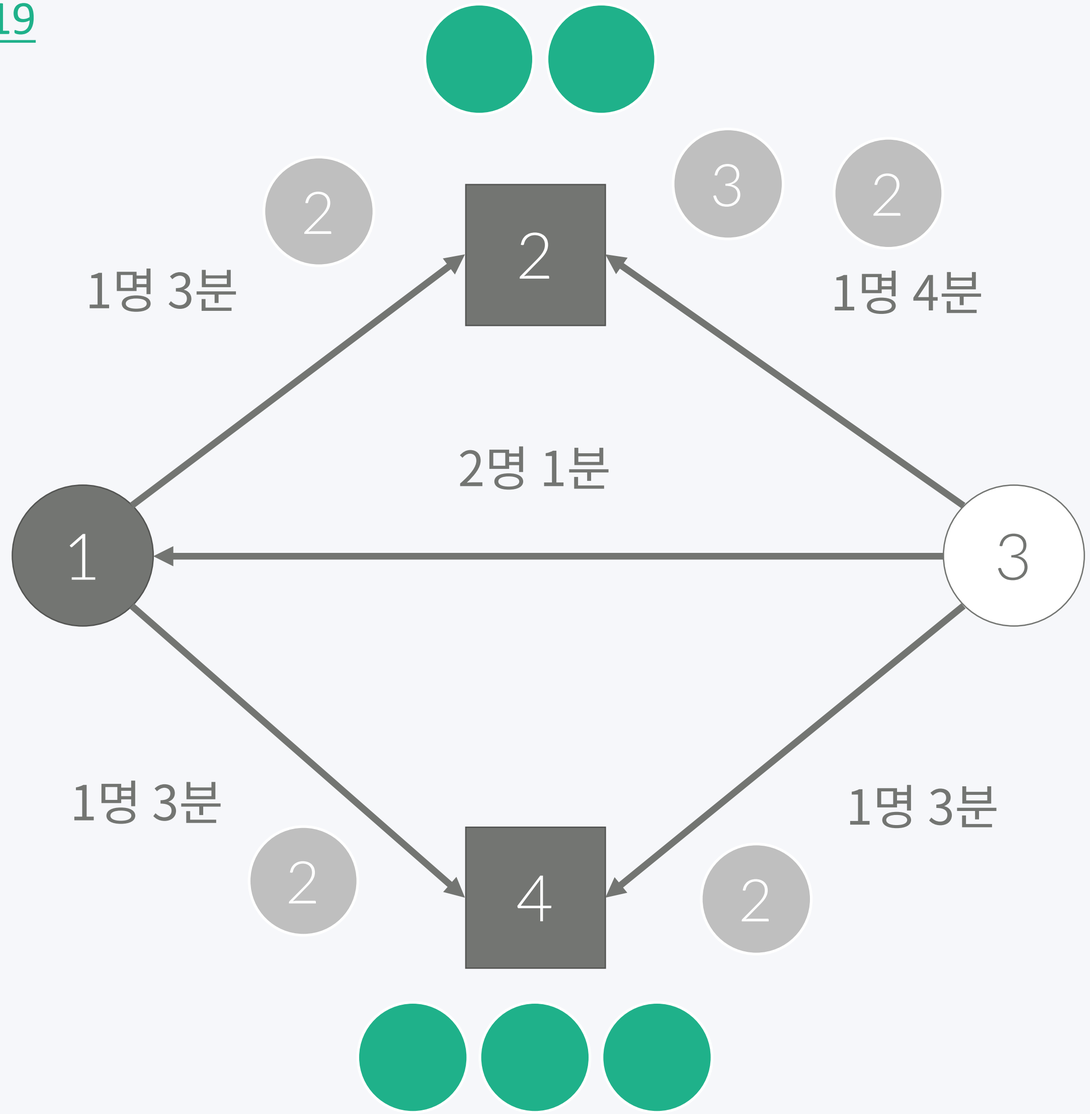
- 3분



Avoiding the Apocalypse

https://www.acmicpc.net/problem/10319

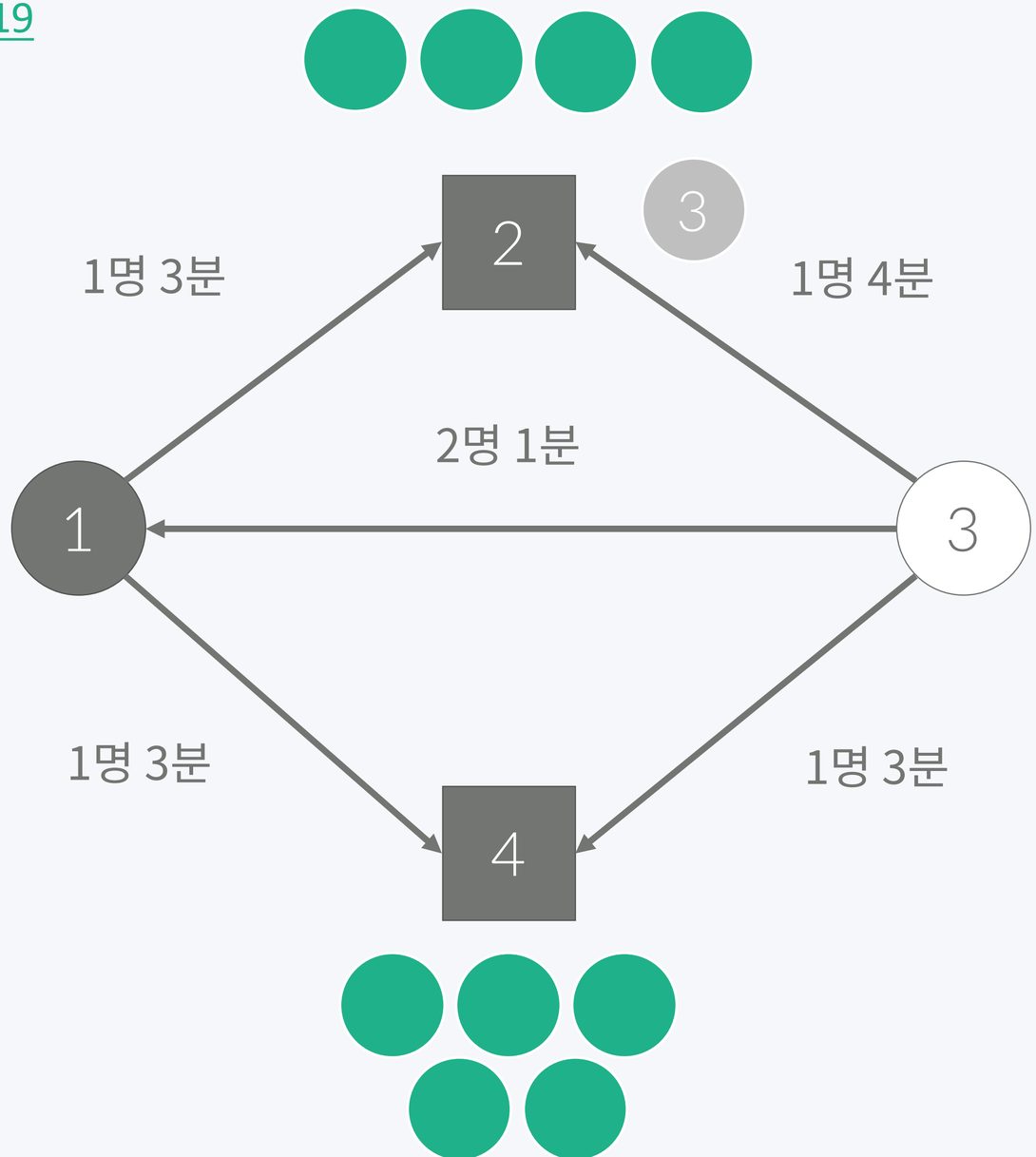
- 4분



Avoiding the Apocalypse

https://www.acmicpc.net/problem/10319

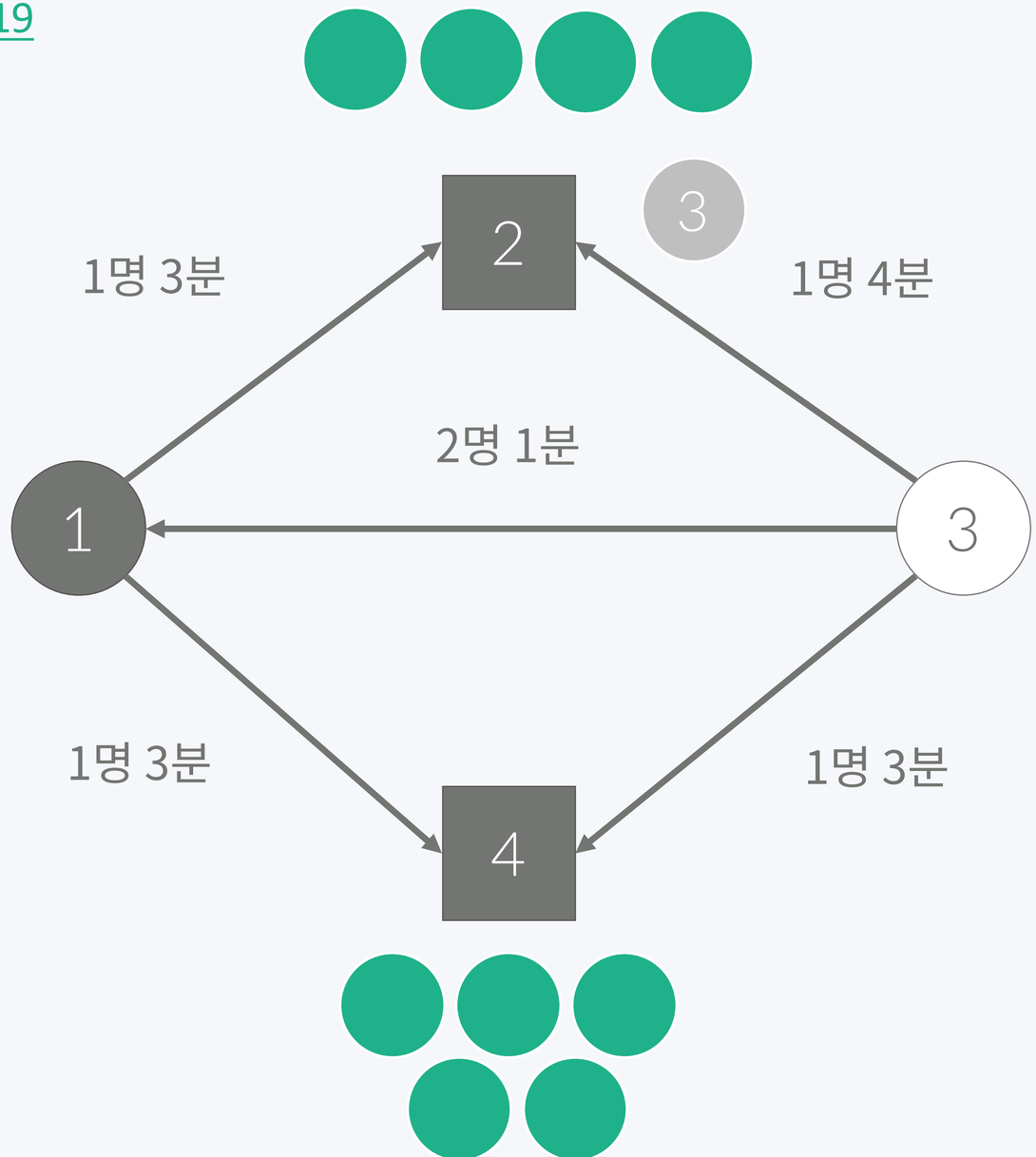
- 5분



Avoiding the Apocalypse

https://www.acmicpc.net/problem/10319

- 1명은 살아남지 못하고
- 좀비로 변하게 된다

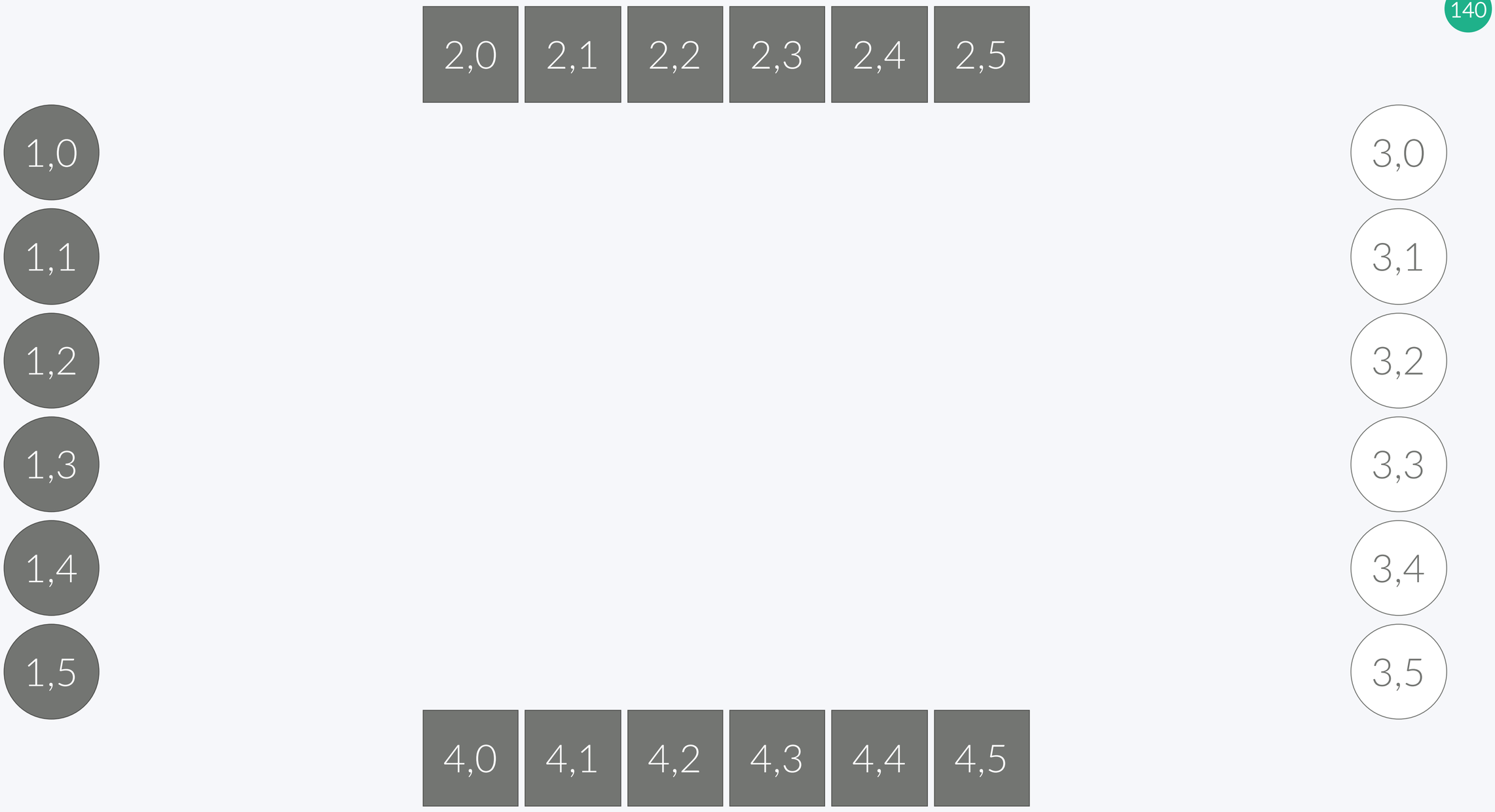


Avoiding the Apocalypse

https://www.acmicpc.net/problem/10319

- vertex를 s+1조각 내는 것

- (vertex, 0) -> 0초후 vertex
- (vertex, 1) -> 1초후 vertex

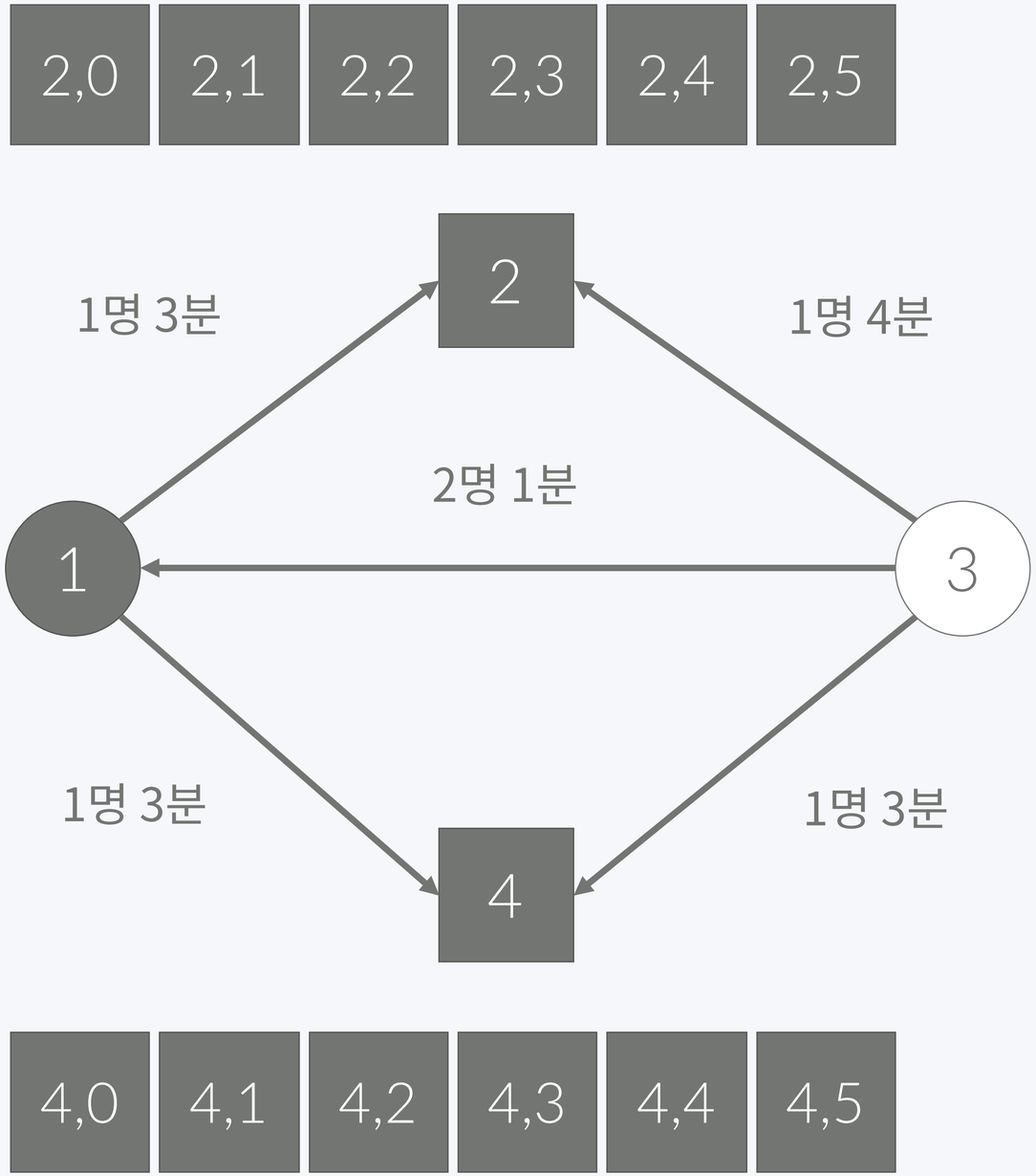

2,0
2,1
2,2
2,3
2,4
2,5
1,0
1,1
1,2
1,3
1,4
1,5
3,0
3,1
3,2
3,3
3,4
3,5
4,0
4,1
4,2
4,3
4,4
4,5

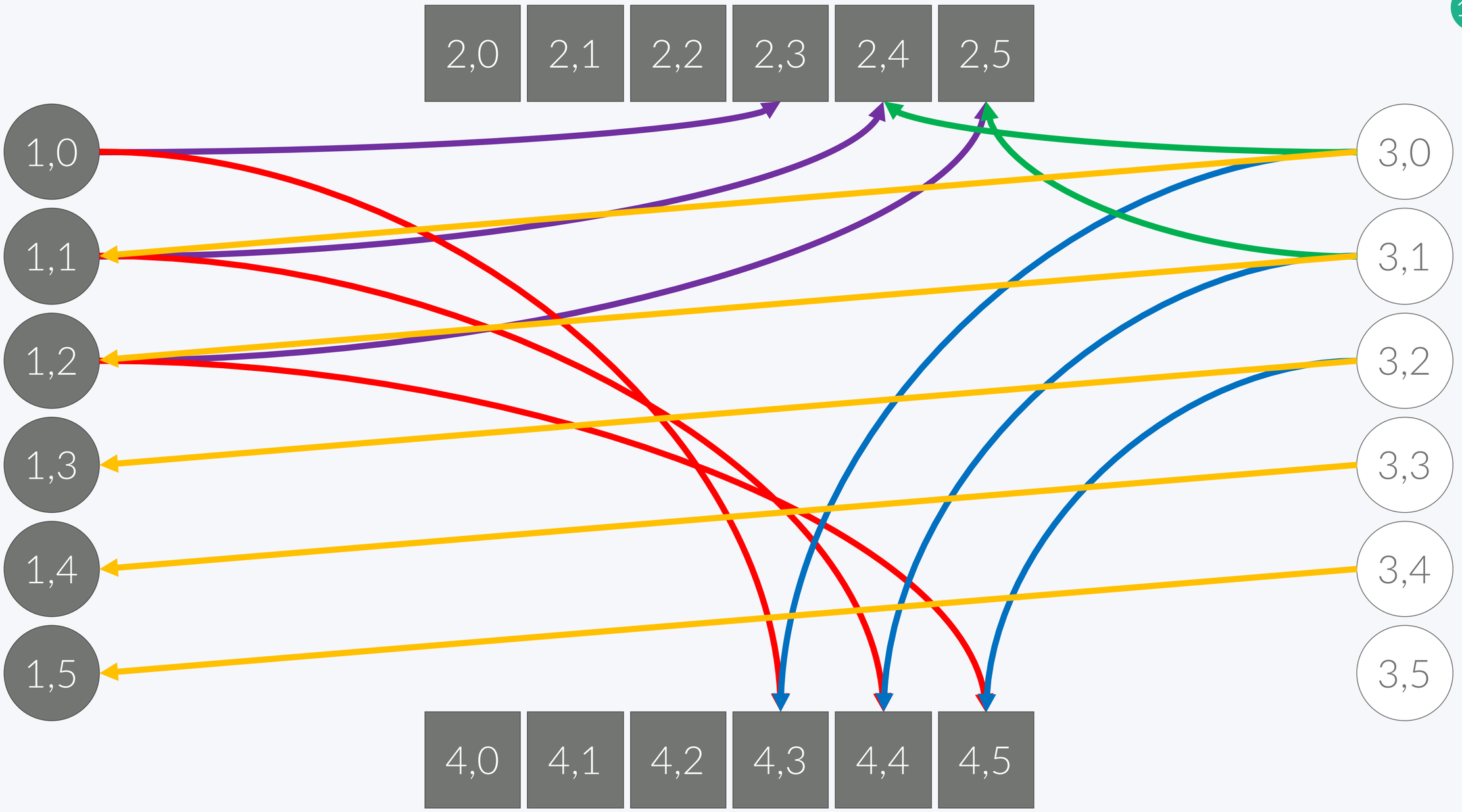
Avoiding the Apocalypse

https://www.acmicpc.net/problem/10319

- edge를 연결한다

- a -> b로 t초 걸리고 p명이 이동할 수 있으면
- (a, i) -> (b, i+t)
- capacity: p

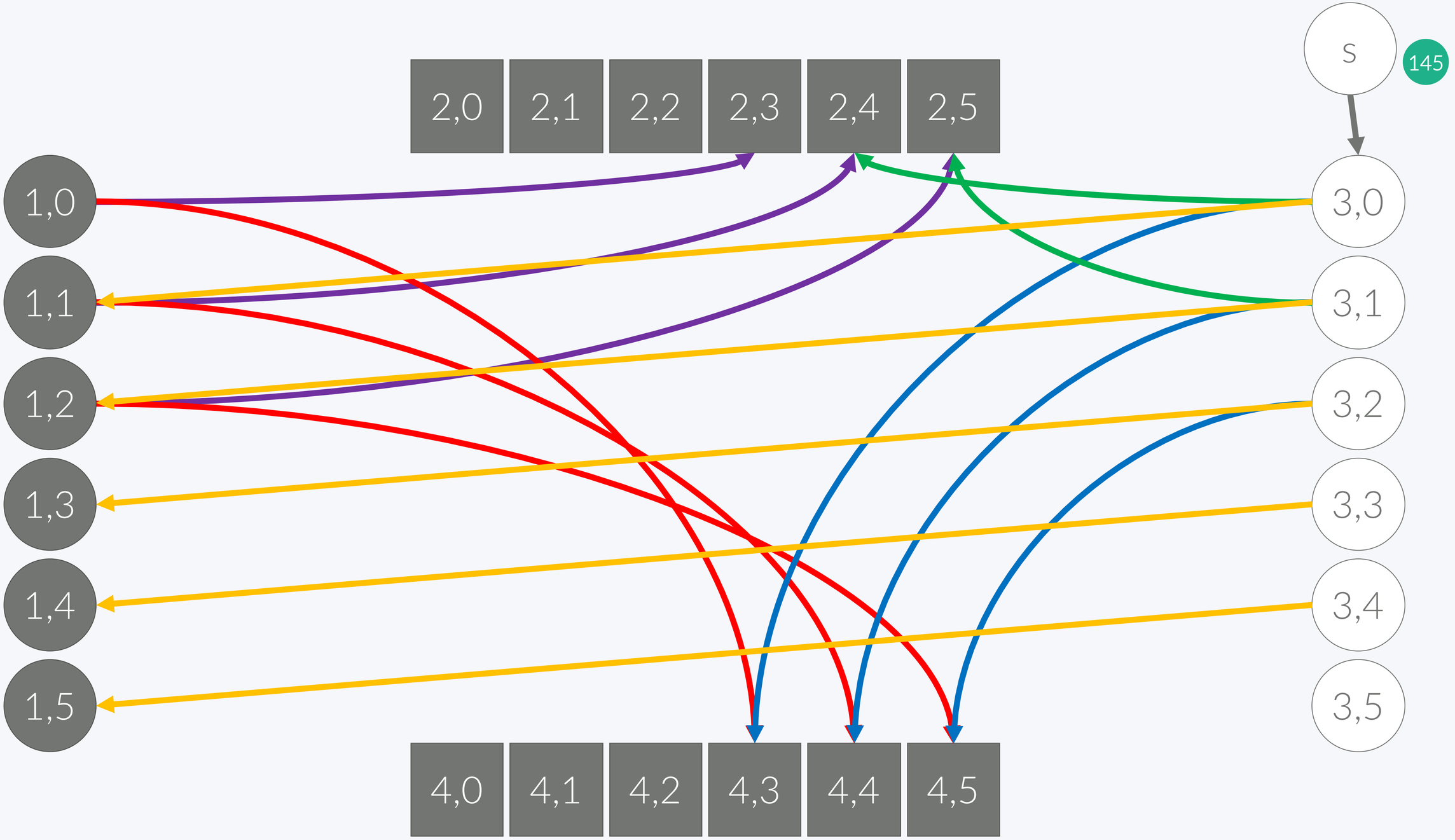

2,0
2,1
2,2
2,3
2,4
2,5
1,0
1,1
1,2
1,3
1,4
1,5
2
1명 3분
1명 4분
2명 1분
1
3
1명 3분
1명 3분
4
3,0
3,1
3,2
3,3
3,4
3,5
4,0
4,1
4,2
4,3
4,4
4,5


2,0
2,1
2,2
2,3
2,4
2,5
1,0
1,1
1,2
1,3
1,4
1,5
3,0
3,1
3,2
3,3
3,4
3,5
4,0
4,1
4,2
4,3
4,4
4,5

Avoiding the Apocalypse

https://www.acmicpc.net/problem/10319

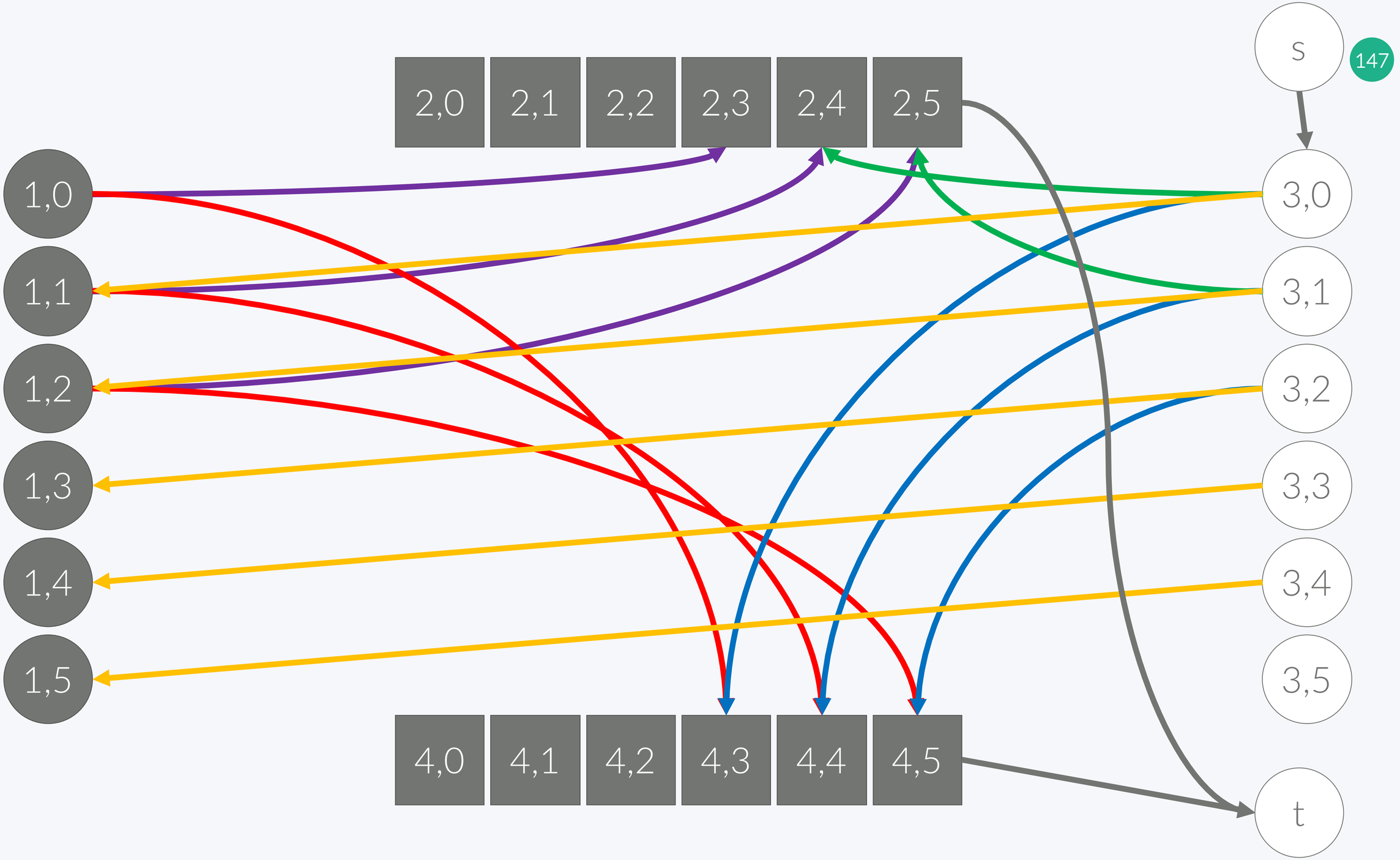
- Source에서 0초의 시작 Vertex로 출발하는 사람 capacity 만큼 edge 추가


s
145
2,0
2,1
2,2
2,3
2,4
2,5
1,0
1,1
1,2
1,3
1,4
1,5
3,0
3,1
3,2
3,3
3,4
3,5
4,0
4,1
4,2
4,3
4,4
4,5

Avoiding the Apocalypse

https://www.acmicpc.net/problem/10319

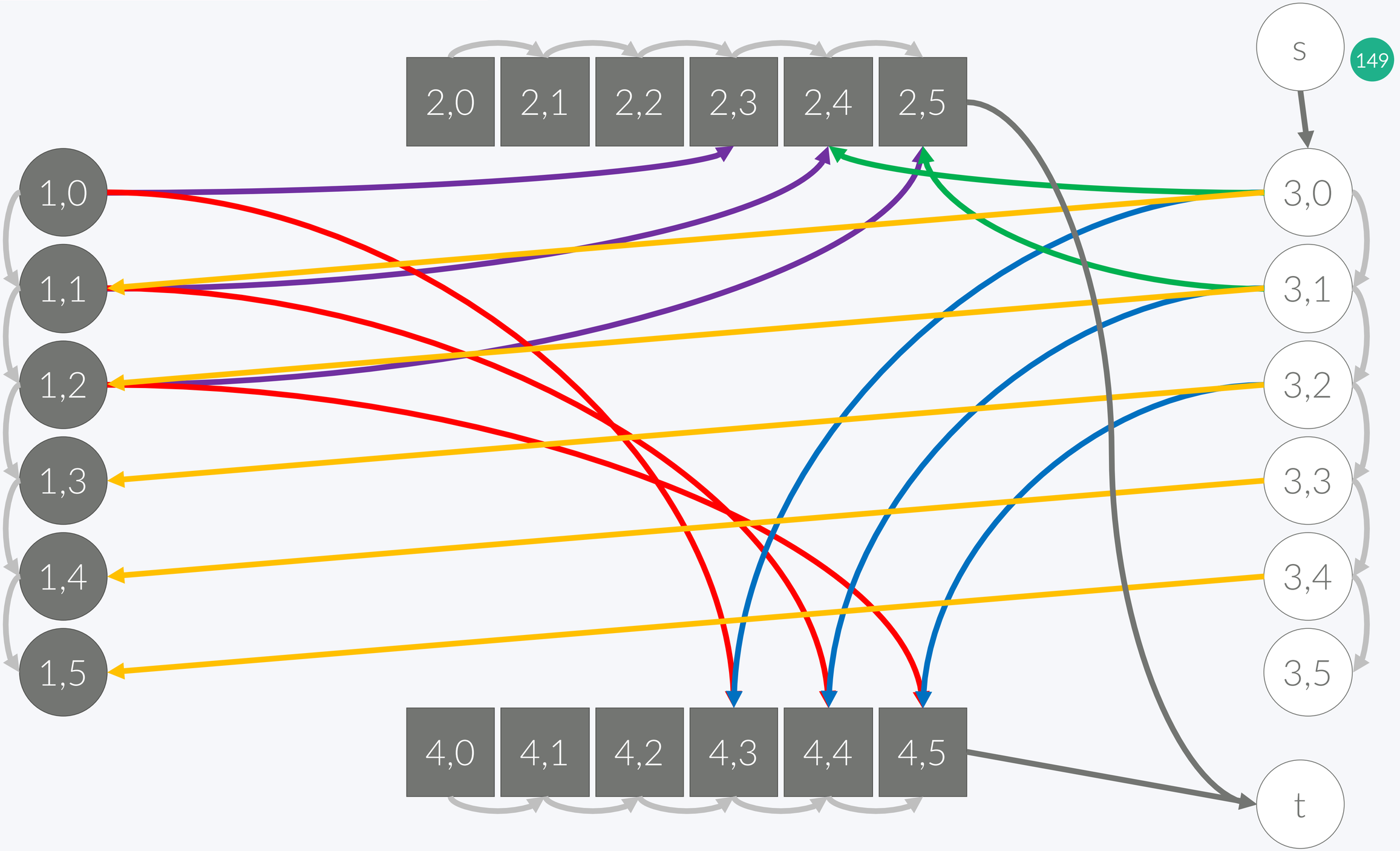
- 모든 병원 vertex의 s초에서 Sink로 Capacity 무한대만큼 edge 추가


s
147
2,0
2,1
2,2
2,3
2,4
2,5
1,0
1,1
1,2
1,3
1,4
1,5
3,0
3,1
3,2
3,3
3,4
3,5
4,0
4,1
4,2
4,3
4,4
4,5
t

Avoiding the Apocalypse

https://www.acmicpc.net/problem/10319

- vertex에 서서 기다릴 수 있으므로
- 모든 vertex의 i초와 i+1초를 연결하는 edge를 만듦
- capacity: 무한대


s
149
2,0
2,1
2,2
2,3
2,4
2,5
1,0
1,1
1,2
1,3
1,4
1,5
3,0
3,1
3,2
3,3
3,4
3,5
4,0
4,1
4,2
4,3
4,4
4,5
t

Avoiding the Apocalypse

https://www.acmicpc.net/problem/10319

- https://gist.github.com/Baekjoon/de05c27250da7fffa40f

스타 대결

https://www.acmicpc.net/problem/1031

- 지민이의 팀: N명
- 한수의 팀: M명
- 각 사람당 해야하는 경기의 수가 정해져 있다.
- 경기: 1 vs 1
- 지민이의 팀 1명, 한수의 팀 중 1명
- 같은 대결은 1번 밖에 못함
- 모든 팀원은 해야하는 경기수 만큼 경기를 해야함
- 경기 대진표를 찾는데, 가능한 대진표가 여러가지면 사전순으로 앞서는 것

스타 대결

https://www.acmicpc.net/problem/1031

- edge를 사전순으로
- (1, 1), (1, 2), … (1, M), (2,1), (2,2), …, (2, M), ..., (N, 1), ..., (N, M)
- 순서대로
- 하나씩 빼면서 Maximum Flow를 돌린다
- 정답을 구할 수 있으면, 필요없는 edge이니 지워버린다.

스타 대결

https://www.acmicpc.net/problem/1031

- 시간복잡도: O(NM) * O(N^2M^2) = O(N^3M^3)

스타 대결

https://www.acmicpc.net/problem/1031

- edge를 사전순으로
- (1, 1), (1, 2), … (1, M), (2,1), (2,2), …, (2, M), ..., (N, 1), ..., (N, M)
- 순서대로
- 하나씩 빼면서 Augmenting Path를 찾는다.
- 찾으면, 그 edge를 통하지 않고 flow를 돌릴 수 있기 때문에, 필요없는 edge

스타 대결

https://www.acmicpc.net/problem/1031

- https://gist.github.com/Baekjoon/5572680b0d1f421f08eb

No Smoking, Please

https://www.acmicpc.net/problem/5406

- 레스토랑은 N*M 크기 직사각형이다
- 금연구역과 흡연구역으로 나눠야 함
- 흡연구역은 입구와 directed하게 연결되어 있어야 함
- 금연구역은 주방과 연결되어 있어야 함
- 레스토랑은 1*1크기의 방으로 이루어져 있음
- 금연구역과 흡연구역 사이 복도에는 Air Lock과 Hatch를 만들어야 함
- Air Lock: 복도의 면적 * 1000유로
- Hatch: 1000 유로

No Smoking, Please

https://www.acmicpc.net/problem/5406

- Source:
- Sink:
- 그래프 구성:

No Smoking, Please

https://www.acmicpc.net/problem/5406

- Source: 입구
- Sink: 주방
- 그래프 구성: 인접한 칸 사이에 edge를 연결 cost: 비용

No Smoking, Please

https://www.acmicpc.net/problem/5406

- min-cut 문제가 된다.
- 주의해야 할 점
- 복도의 면적이 0이면, air lock과 hatch를 설치할 수 없어서 cost가 0이다.
- https://gist.github.com/Baekjoon/5fab3c76514eeb6a17c5

Chess Competition

https://www.acmicpc.net/problem/5424

- 체스 시합이다
- N명이 참가한다
- 1:1 대결을 해서 이기면 1점, 지면 0점, 비기면 0.5점씩 나눠갖는다
- 가장 많은 점수를 가진 사람이 우승
- 우승자가 여러명이면 tie-break game을 가진다.

- 중간 결과가 주어졌을 때, 우승할 수 있는 사람을 구하는 문제

Chess Competition

https://www.acmicpc.net/problem/5424

- 1: 승리
- 0: 패배
- d: 무승부
- . : 아직 경기 하지 않음
- x: i == i인 경우 (경기를 할 수 없음)

- 항상 consistent한 결과만 입력으로 주어짐

Chess Competition

https://www.acmicpc.net/problem/5424

- 각 사람이 우승할 수 있는지 없는지를 검사해야 함
- 사람 A가 남은 경기를 모두 이겼다고 가정하고
- 남은 경기의 결과가 적절히 결정되었을 때
- A보다 높은 점수를 가진 사람이 없어야 함

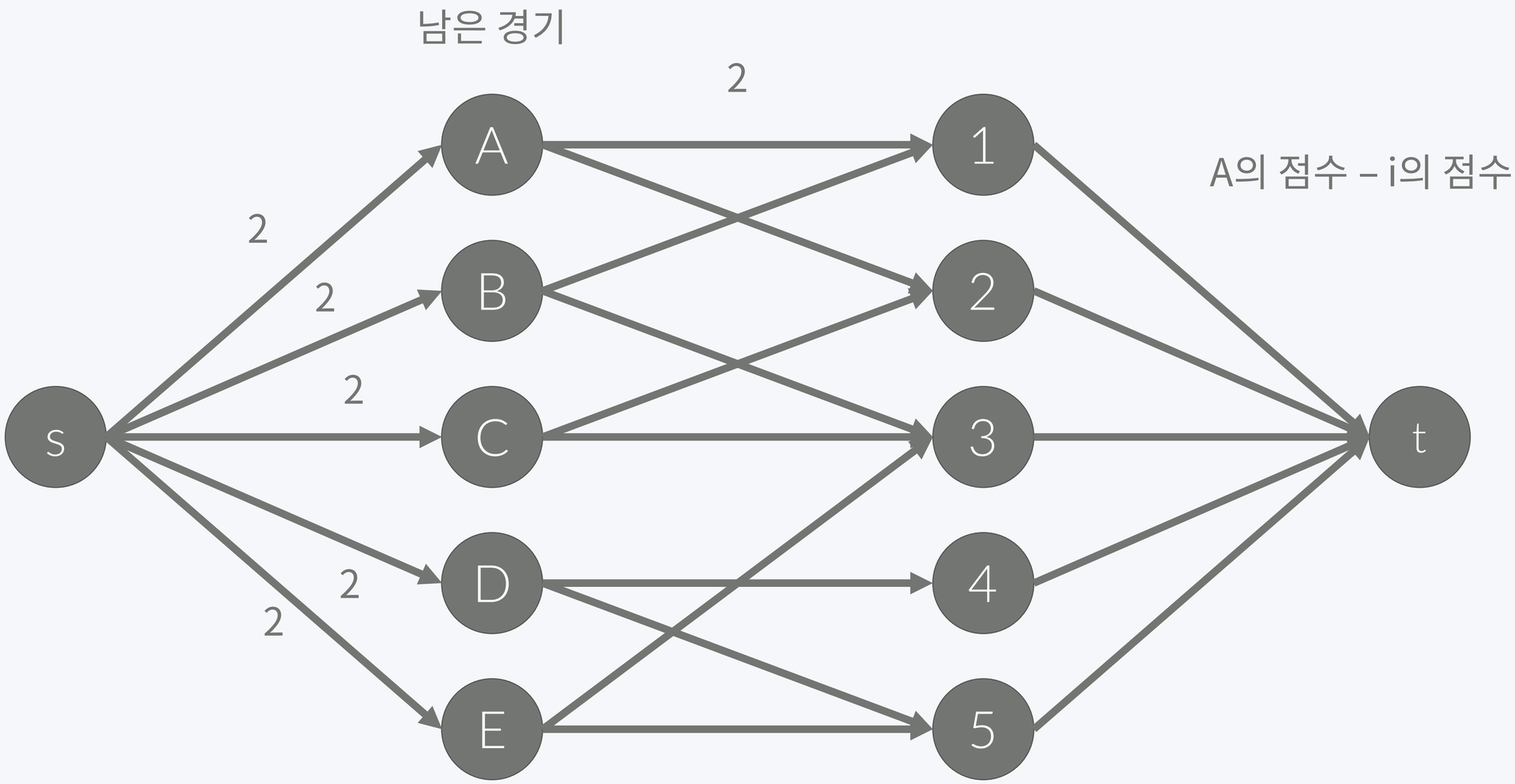
Chess Competition

https://www.acmicpc.net/problem/5424

- 소수점을 피하기 위해
- 이기면 2점
- 비기면 1점씩

Chess Competition

https://www.acmicpc.net/problem/5424



Chess Competition

https://www.acmicpc.net/problem/5424

- flow를 돌렸을 때, maximum flow가 point 보다 작으면, 이길 수 없는 것
- https://gist.github.com/Baekjoon/7d347457462d06a336e0

도시 왕복하기

https://www.acmicpc.net/problem/2316

- 어떤 그래프 G가 주어진다.
- 1번 도시와 2번 도시를 오가는 최대 횟수 구하기
- 이 때, 한 번 방문했던 도시는 다시 방문하지 않게 한다.

도시 왕복하기

https://www.acmicpc.net/problem/2316

- Vertex Capacity 를 이용해 각 정점을 Vin과 Vout으로 쪼갠다
- Vin -> Vout의 capacity를 1로 두면, 같은 정점은 1번만 방문하게 된다.

- 이제 이 문제는 1을 source로, 2 를 sink로 할 때
- Maximum Flow를 구하는 것이 답이다.

도시 왕복하기

https://www.acmicpc.net/problem/2316

- https://gist.github.com/Baekjoon/2637141271719edf72a5

숫자판 만들기

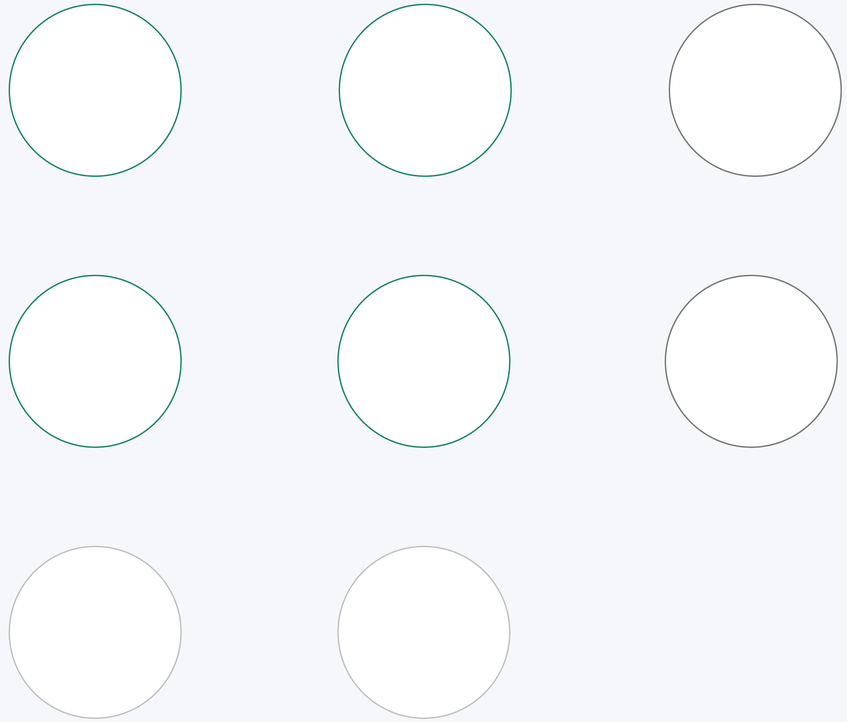
https://www.acmicpc.net/problem/2365

		12
		4
6	10	

숫자판 만들기

https://www.acmicpc.net/problem/2365

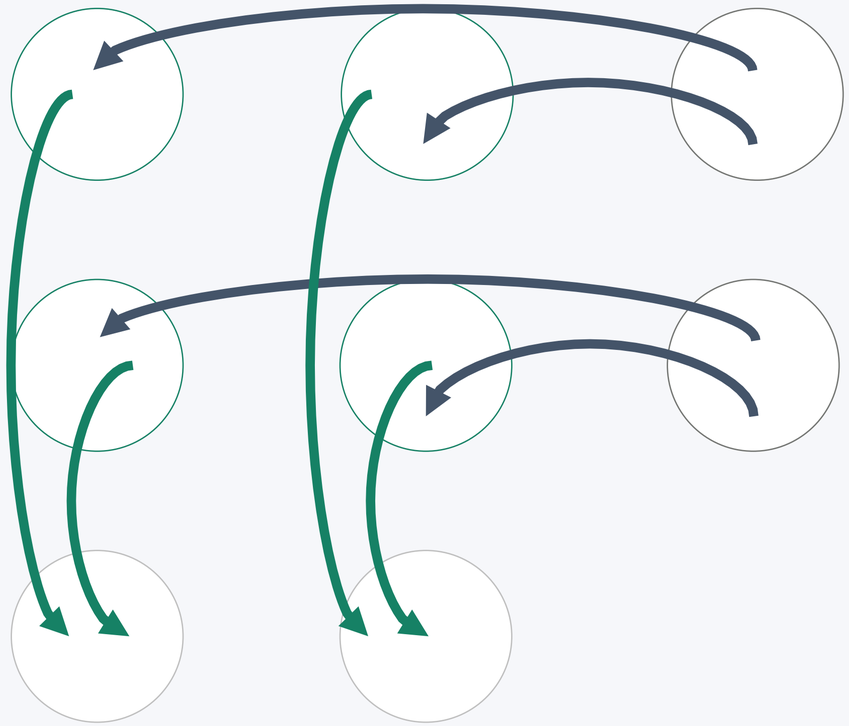
		12
		4
6	10	



숫자판 만들기

https://www.acmicpc.net/problem/2365

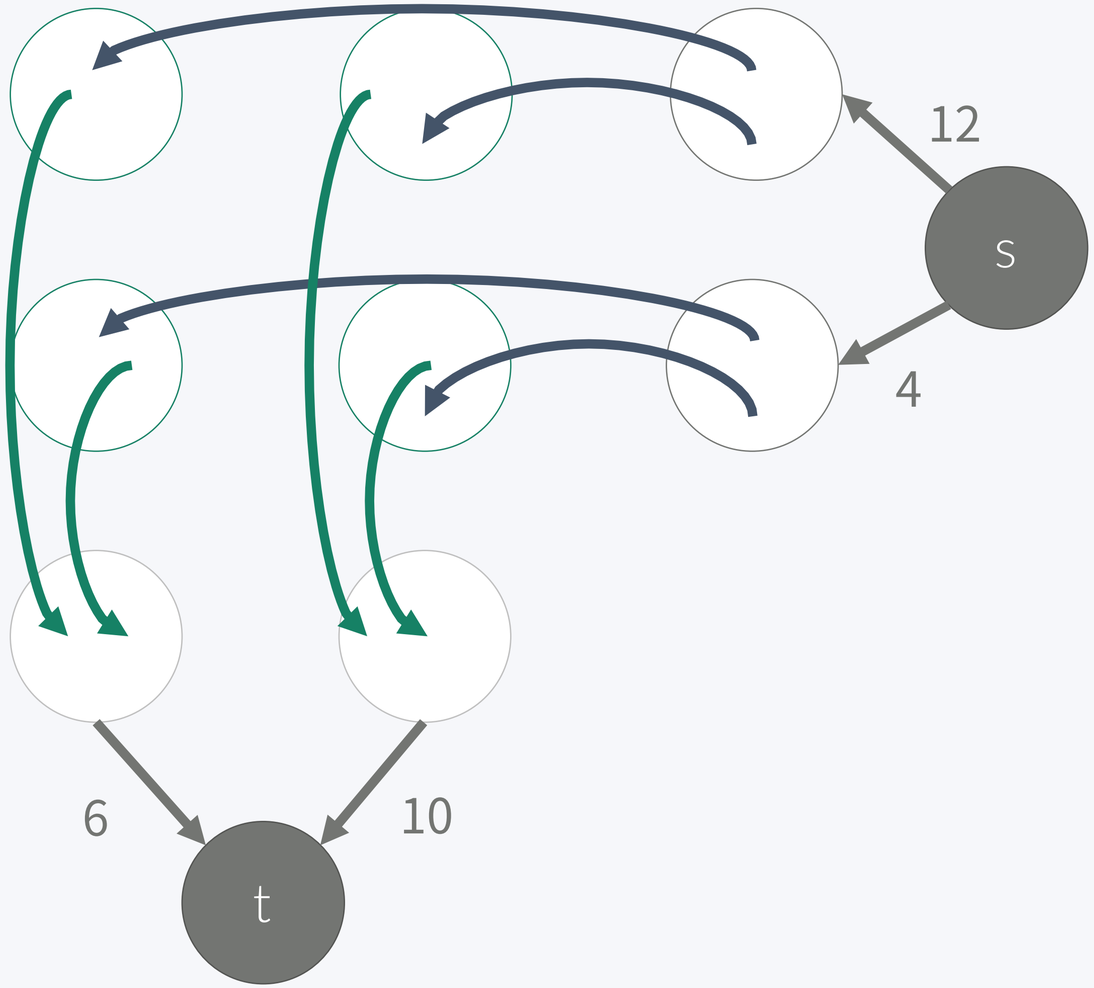
		12
		4
6	10	



숫자판 만들기

https://www.acmicpc.net/problem/2365

		12
		4
6	10	

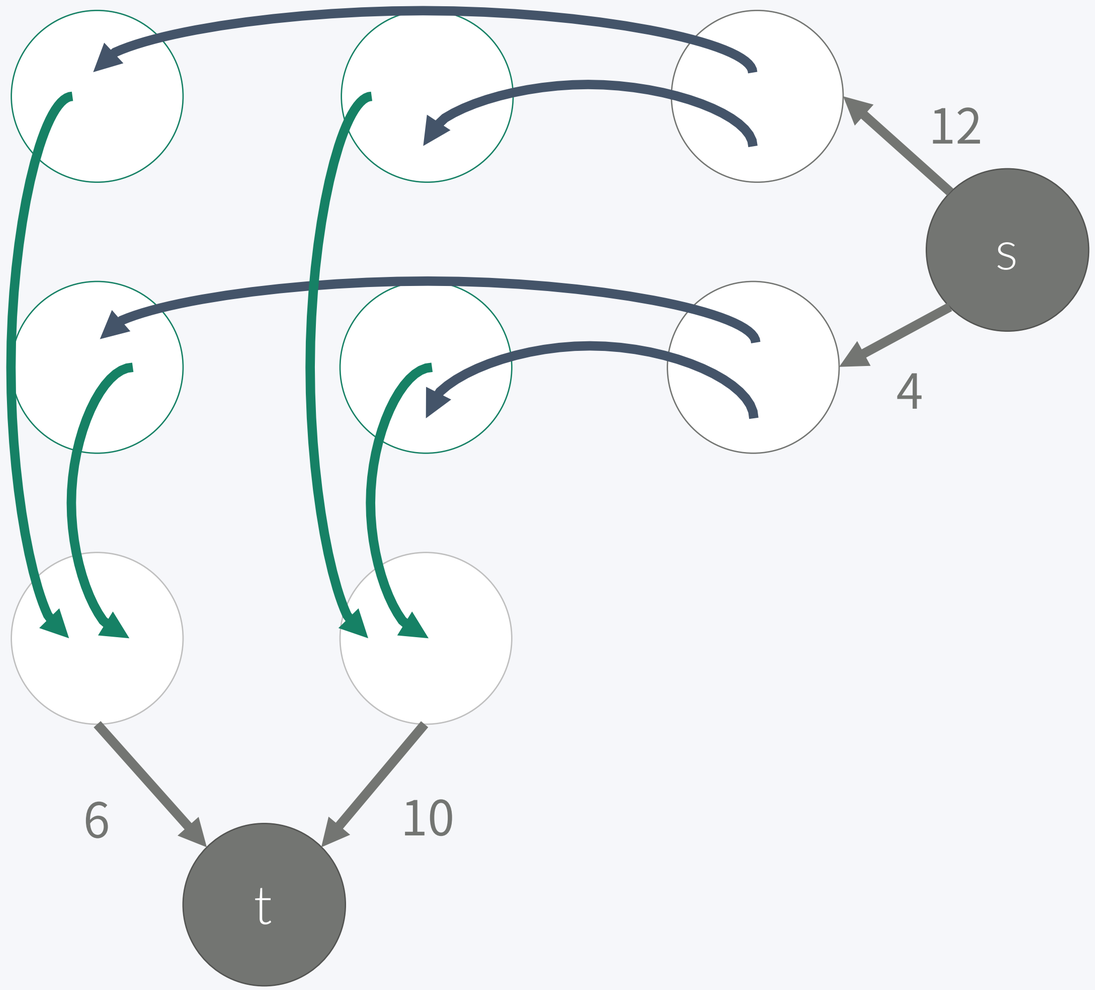


숫자판 만들기

https://www.acmicpc.net/problem/2365

- 최대 숫자를 K로 결정

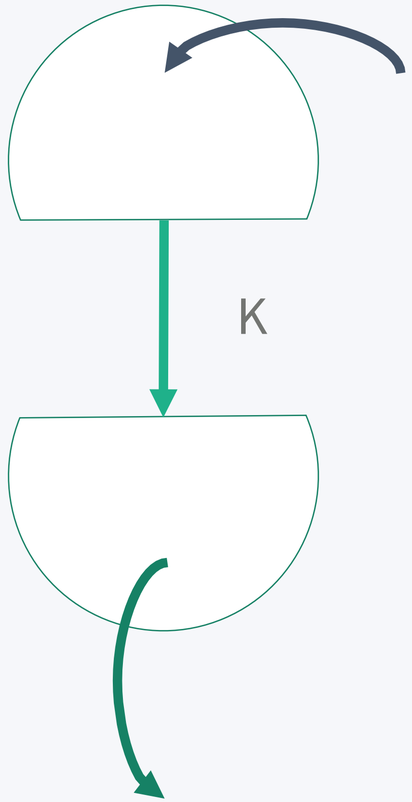
		12
		4
6	10	

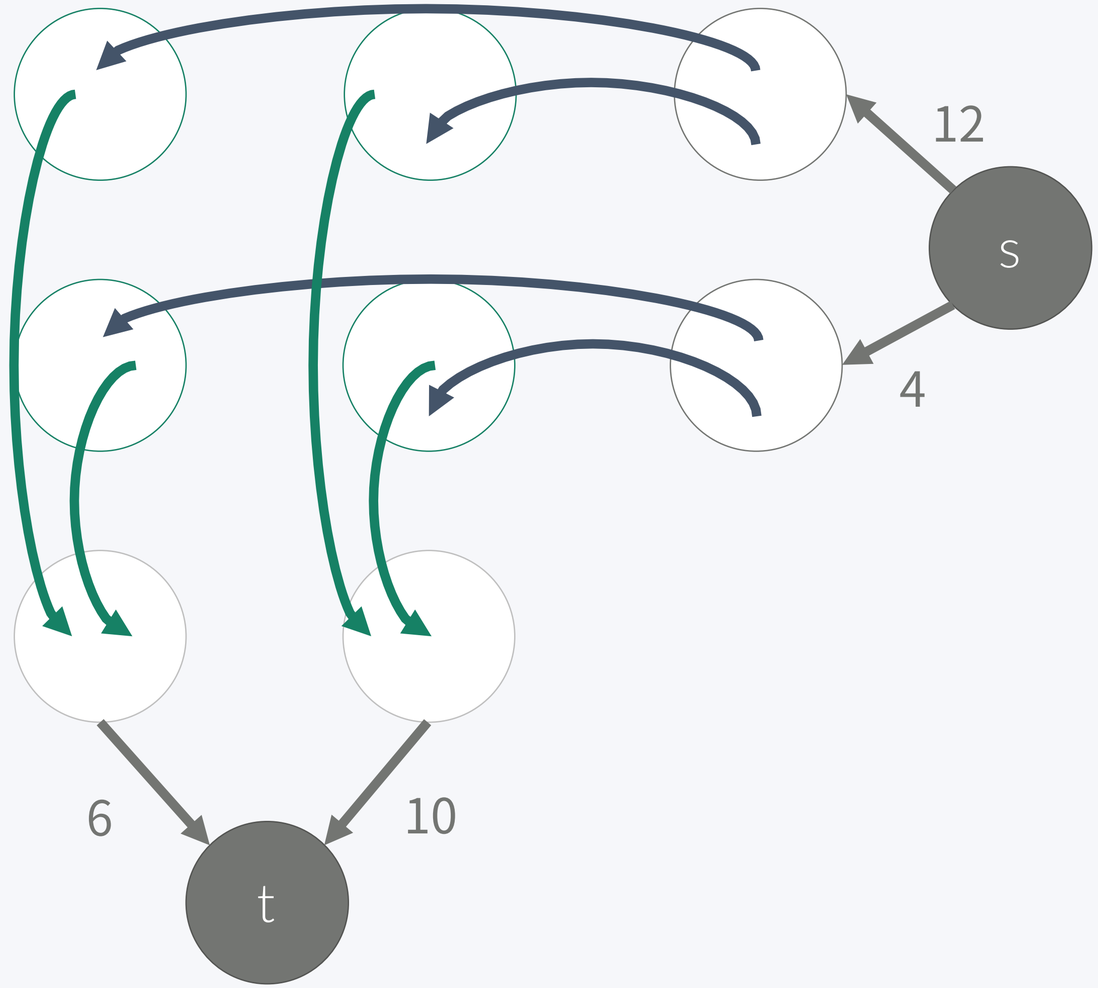


숫자판 만들기

https://www.acmicpc.net/problem/2365

- 최대 숫자를 K로 결정

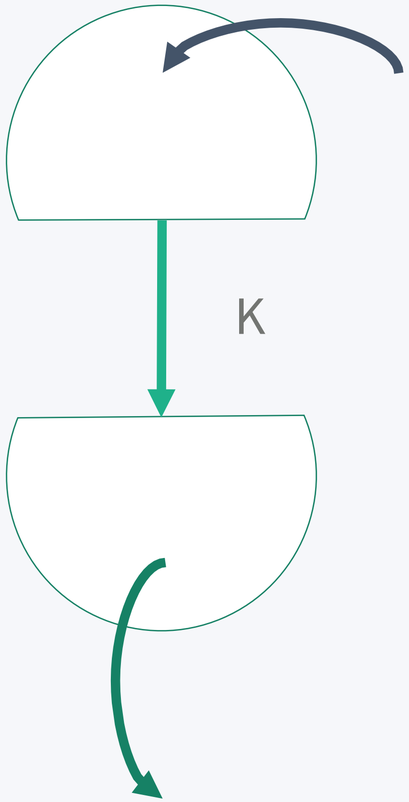


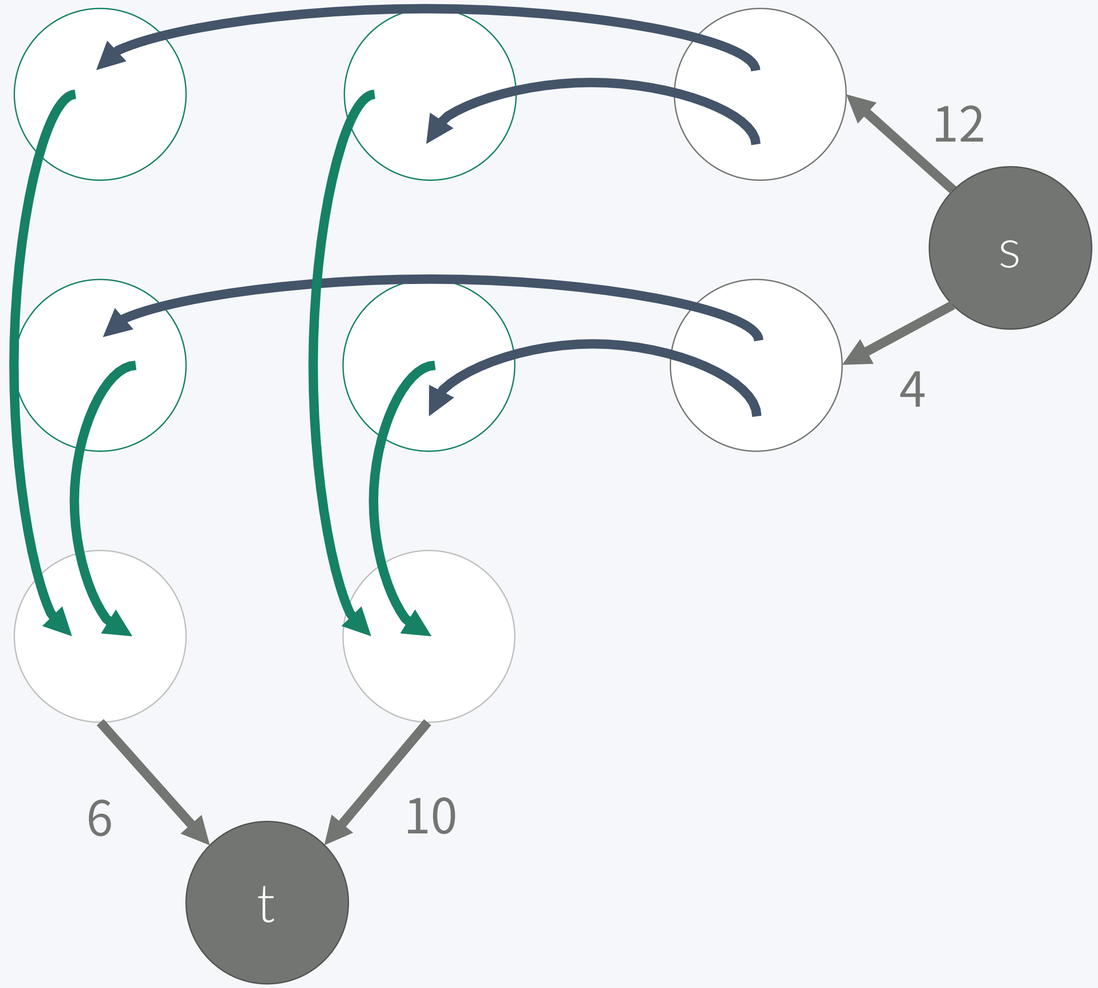


숫자판 만들기

https://www.acmicpc.net/problem/2365

- https://gist.github.com/Baekjoon/0a0992a460547d9fdc6b





Wrong Answer

https://www.acmicpc.net/problem/5398

- 크로스워드 퍼즐을 푼 결과가 주어진다.
- 올바르지 않게 채운 결과도 있다.
- 정답이 될 수 있는 단어의 최대 개수를 구하는 문제

Wrong Answer

https://www.acmicpc.net/problem/5398

- 2 2
- 0 1 BAPC
- 0 2 LEIDEN
- 0 0 SOLUTION
- 2 1 WINNER

S							
O		W					
L	E	I	D	E	N		
U		N					
T		N					
I		E					
O		R					
N							

Wrong Answer

https://www.acmicpc.net/problem/5398

- 두 단어가 겹칠 수 있다.
- 항상 가로 단어와 세로 단어가 겹치기 때문에
- 겹치는 것으로 그래프를 만들면 이분 그래프가 된다
- 최소 정답의 개수는 최대 매칭의 개수와 같다 (Max-flow Min-cut theorem)
- 정답은 H + V – 매칭 의 개수가 된다.

Wrong Answer

https://www.acmicpc.net/problem/5398

- https://gist.github.com/Baekjoon/48381e6a6df43bcc15d2

Fake Scoreboard

https://www.acmicpc.net/problem/3848

- 각 팀이 푼 문제 수와 각 문제가 풀린 횟수가 주어졌을 때
- 각 팀이 어떤 문제를 풀었는지 구하는 문제

Fake Scoreboard

https://www.acmicpc.net/problem/3848

- 세 팀이 2, 1, 2 문제를 풀었고, 세 문제를 푼 팀이 1, 2, 3인 경우에
- 가능한 경우
- NYY
- NNY
- YYN
- 또는
- NYY
- NYN
- YNY

Fake Scoreboard

https://www.acmicpc.net/problem/3848

- 가능한 경우가 여러가지면 사전 순으로 앞서는 것을 찾는 문제
- 최대 유량을 구한 다음, 사용 하는 것으로 체크되어 있는 간선 마다
- 그 간선을 이용하지 않고 다른 매칭을 구할 수 있으면 N, 이용해야 하면 Y

Fake Scoreboard

https://www.acmicpc.net/problem/3848

- https://gist.github.com/Baekjoon/ec14970d7544bbe40fea

들쥐의 탈출

https://www.acmicpc.net/problem/2191

- N(1≤N≤100)마리의 들쥐들과 M(1≤M≤100)개의 땅굴이 있다. 각각의 들쥐는 2차원 평면상의 한 위치에 있고, 각각의 땅굴들도 2차원 평면상의 한 점에 위치한다.
- 들쥐들을 잡아먹는 매가 들쥐들을 습격했을 때, 쥐들은 매를 피하기 위해서 땅굴 속으로 숨을 수 있다.
- 매는 현재를 기준으로 S(1≤S≤100)초가 지난 후에 지상에 도착한다. 각각의 들쥐들은 매 초당 V(1≤V≤100)만큼의 거리를 움직인다(즉 V가 쥐들의 초속이다). 만약 S초가 되기 전에 들쥐가 땅굴에 도착하게 되면 그 들쥐는 땅굴로 숨을 수 있다. 단, 들쥐가 도착하는 시간이 정확히 S인 경우에도 그 들쥐는 도망칠 수 있는 것으로 간주한다.
- 들쥐들은 종족 전체의 번영을 위해, 매에 잡아먹히게 되는 들쥐의 수가 최소가 되도록 도망치기로 하였다. 들쥐와 땅굴의 위치, 그리고 들쥐의 속도와 매가 도착하는 시간이 주어졌을 때, 잡아먹히게 되는 들쥐의 최소수를 구하는 프로그램을 작성하시오.

들쥐의 탈출

https://www.acmicpc.net/problem/2191

군사 배치

https://www.acmicpc.net/problem/1841

- 두 개의 도시와 N(0≤N≤150)개의 마을로 이루어진 나라가 있다. 몇 개의 마을과 도시 사이에는 양방향으로 이동 가능한 길이 있기도 하다. 각각의 길을 이용하여 이동하는데 걸리는 시간은 서로 다를 수 있지만, 하나의 길을 이동할 때에는 어느 방향으로 이동해도 같은 시간이 걸린다.
- 군대를 배치할 때에는 반드시 길 위에 배치해야 한다. 길 위에 배치할 때에는 그 길이 연결하는 두 개의 마을(혹은 도시) 중 어느 한쪽에 원하는 만큼 가까이 배치할 수 있다.
- 이 나라의 군대는 G(0≤G≤353535)명의 군인으로 구성되어 있는데, 한 곳에 모든 군인을 배치할 수도 있고, 서로 다른 길에 군인들을 배치할 수도 있으며, 한 길 위에도 여러 명의 군인을 (서로 다른 위치에) 배치할 수도 있다. 단, 이와 같이 군대를 배치했을 때, 두 도시 사이를 이동하기 위해서는 적어도 한 명의 군인과 반드시 마주치도록 배치해야 한다.

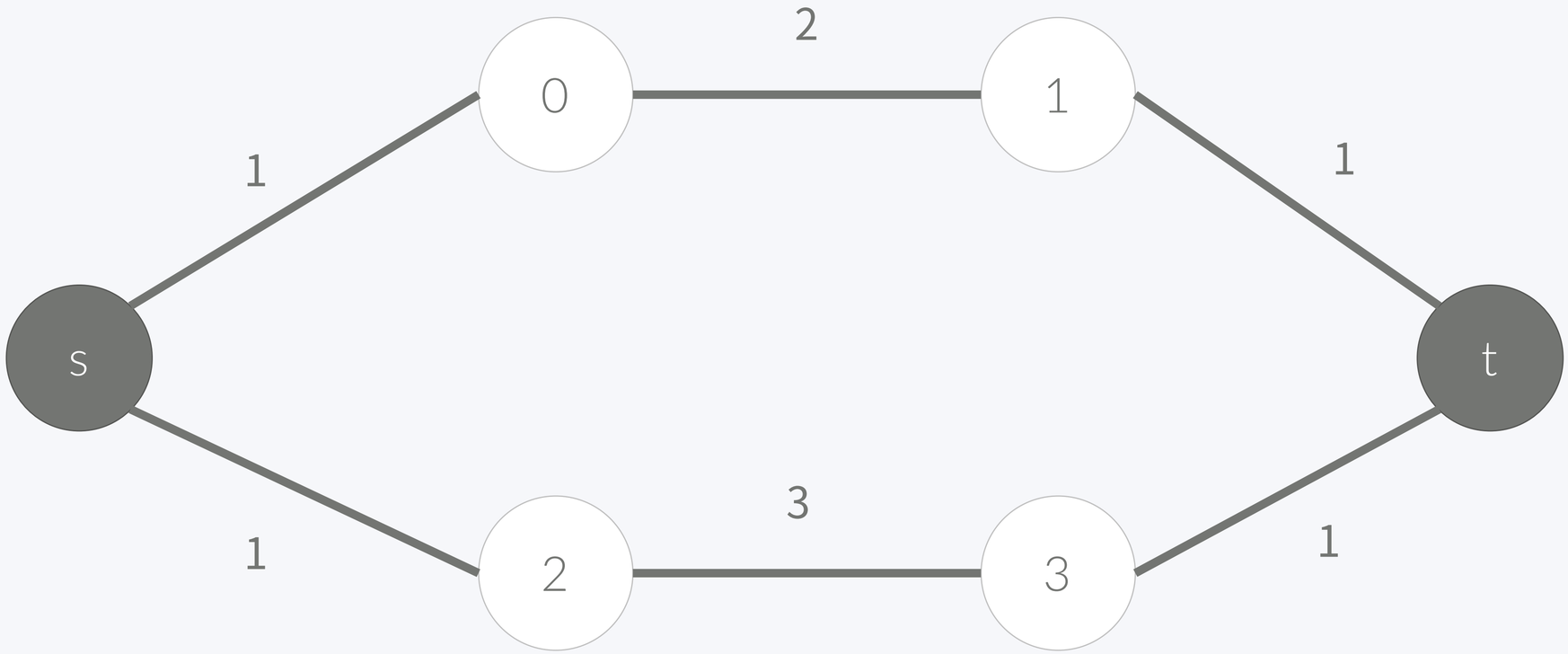
군사 배치

https://www.acmicpc.net/problem/1841

- 또한 때로는 각 도시로 군대를 소집해야 할 필요가 있을 수도 있다. 군대를 소집하는 것은 두 도시 중 어느 한 쪽에서도 할 수 있으며, 군대를 소집하는데 걸리는 시간은 제일 마지막 군인이 마을에 도착하는 시간이 된다. 나라에서는 군대를 효율적으로 배치하기 위해서, 두 도시에서 군대를 소집하는데 걸리는 시간 중 더 큰 값이 작아지도록 하려 한다.
- 도시와 마을에 대한 정보가 주어졌을 때, 군대를 소집하는데 걸리는 시간의 최소값을 구하는 프로그램을 작성하시오.

군사 배치

https://www.acmicpc.net/problem/1841



군사 배치

https://www.acmicpc.net/problem/1841

- 다익스트라, 이분 탐색, 최대 유량을 이용해서 문제를 풀어야 한다
- 먼저, 각 도시에서 마을로 가는 최단 경로를 구해야 한다. (다익스트라)
- 그 다음, 이분 탐색을 이용해 군대를 소집하는데 걸리는 시간을 결정한다 T
- 95050 도시에서 T초만에 이동할 수 없는 모든 마을을 소스로
- 104729 도시에서 T초만에 이동할 수 없는 모든 마을을 싱크로
- 이제 소스에서 싱크로 M명의 군인을 배치해서 이동할 수 없게 하는 문제가 되어 버리기 때문에
- 이는 최소 컷과 같다. 최소 컷은 최대 유량과 같다.

졸업

https://www.acmicpc.net/problem/1575

- 졸업 조건은 다음과 같은 형태를 가지고 있다. 1, 2, 3, 4중 2개의 수업을 들어라. 학생들은 이러한 졸업 조건을 N개 가지고 있다. N개를 모든 충족시켜야 졸업할 수 있다.
- 어떤 졸업 조건을 충족시킬 때 사용한 수업은, 다른 졸업 조건을 충족시킬 수 없다
- 예를 들어서, “1, 2, 3중 2개의 수업을 들어라“, ”3, 4, 5중 2개의 수업을 들어라“가 있을 때, 첫 번째 조건을 충족시키는 수업이 1, 3이면, 두 번째 조건을 충족시킬 때, 3은 이용되어서 안되는 것이다.
- 정문이가 현재 들은 수업과, 졸업 조건이 주어질 때, 그 학생이 졸업하기 위해서 들어야하는 최소 수업의 개수와, 어떤 수업을 들어야 하는지 구하는 프로그램을 작성하시오.

졸업

https://www.acmicpc.net/problem/1575

- 예를 들어 이미 들은 수업이 3, 4, 8이고
- 1, 16중 2개
- 3, 4, 5, 6중 3개
- 3, 4, 5, 6, 8 중에 1개를 들어야 하는 경우
- 들어야하는 수업: 1, 5, 16

졸업

https://www.acmicpc.net/problem/1575

프리즌 브레이크

https://www.acmicpc.net/problem/1886

- N*M 크기의 감옥이 있다. 감옥은 스코필드가 미리 뚫어 놓은 탈출구, 벽, 그리고 빈 칸으로 이루어져 있는데 각각의 모든 빈 칸에는 사람이 한 명씩 있다. 감옥 안에 있는 모든 사람들은 탈출구를 통해 탈옥을 해 내야 한다. 물론, 최대한 빨리 모든 사람이 탈출하도록 하고 싶다.
- 각 사람이 한 칸을 이동하는 데에는 1초가 걸리며, 하나의 탈출구를 통해서는 1초에 한 사람만 탈출을 할 수 있다. 그리고 사람들이 탈출하는 동안에는 하나의 빈 칸에 여러 명의 사람들이 있어도 된다. 감옥의 정보가 주어져 있을 때, 사람들이 모두 탈출하려면 모두 몇 초의 시간이 걸리는지 구하는 프로그램을 작성하시오.

프리즌 브레이크

https://www.acmicpc.net/problem/1886