네트워크 플로우

최백준 choi@startlink.io

최대 유량

최대 유량

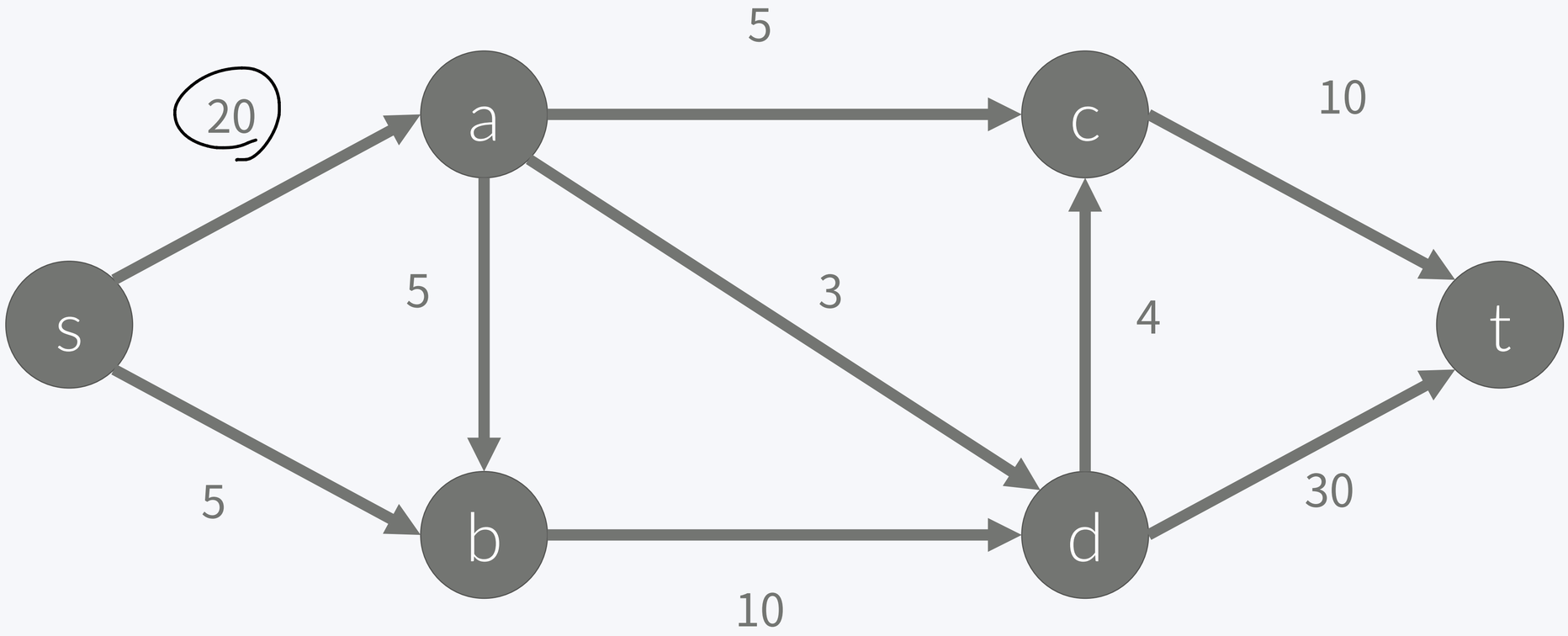
Maximum Flow

weight

Cost

Capacity

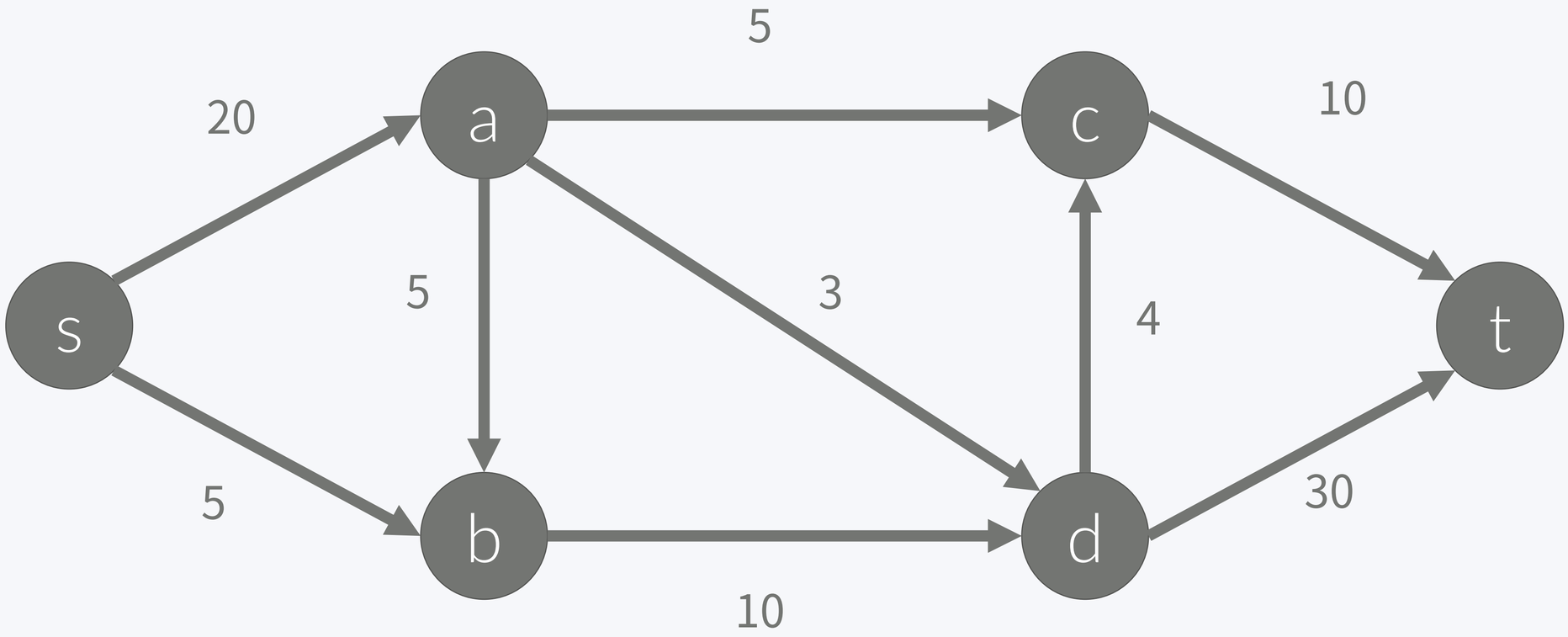
- 각 간선이 나타내는 것은 흐를 수 있는 양
- s에서 t로 최대 얼마나 흐를 수 있는가?



최대 유량

Maximum Flow

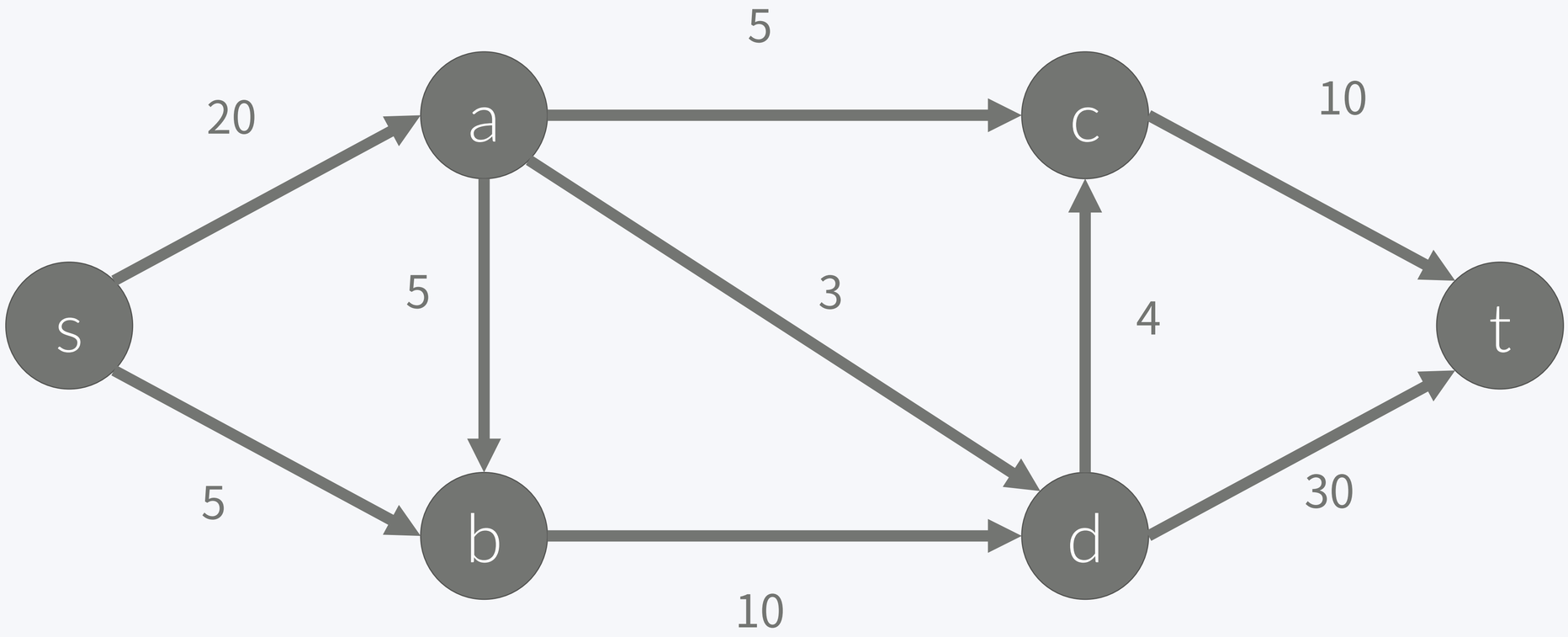
- s -> a로 최대 20만큼만 흐를 수 있다.
- a -> c로 최대 5만큼 흐를 수 있다.



최대 유량

Maximum Flow

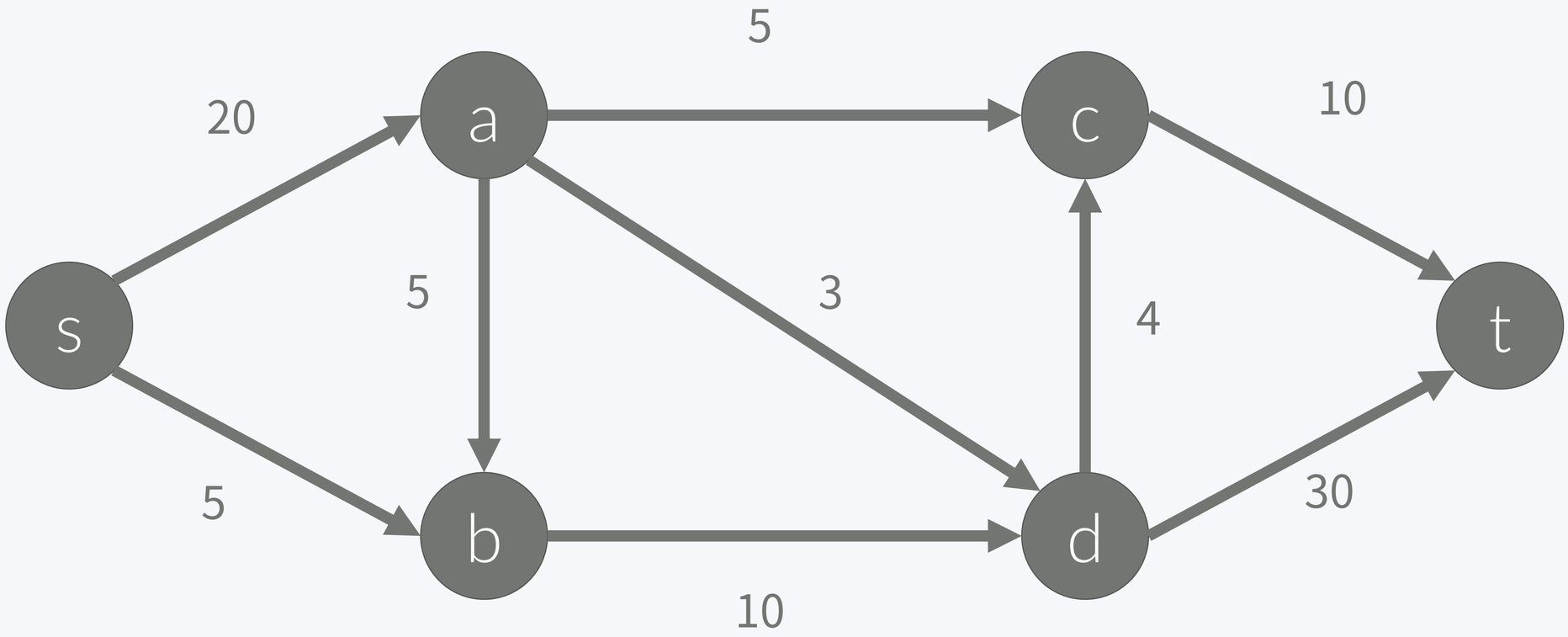
- s -> a로 20을 보냈다고 하더라도
- a -> c, a -> d, a -> b는 5+3+5 = 13이기 때문에, 13 이상을 보낼 수 없다



최대 유량

Maximum Flow

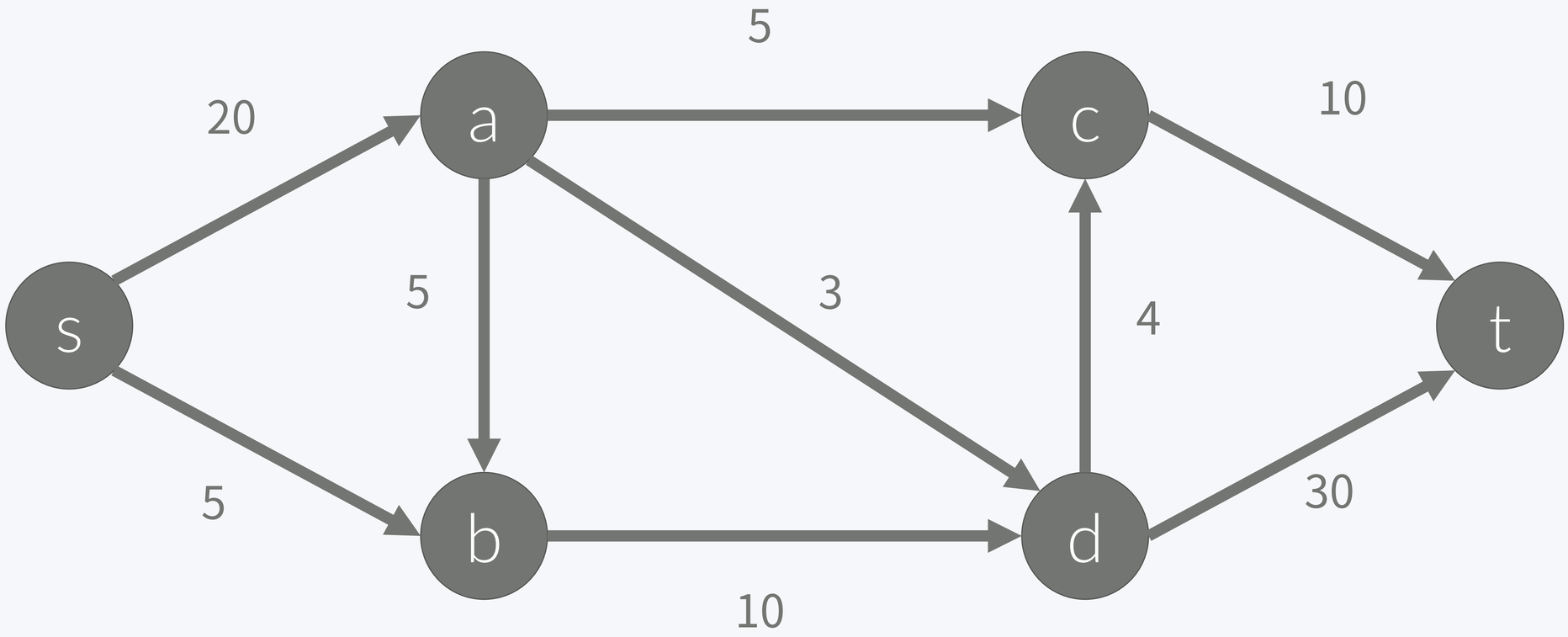
- s에서 t로 최대 얼마나 보낼 수 있는지를 구하는 문제



최대 유량

Maximum Flow

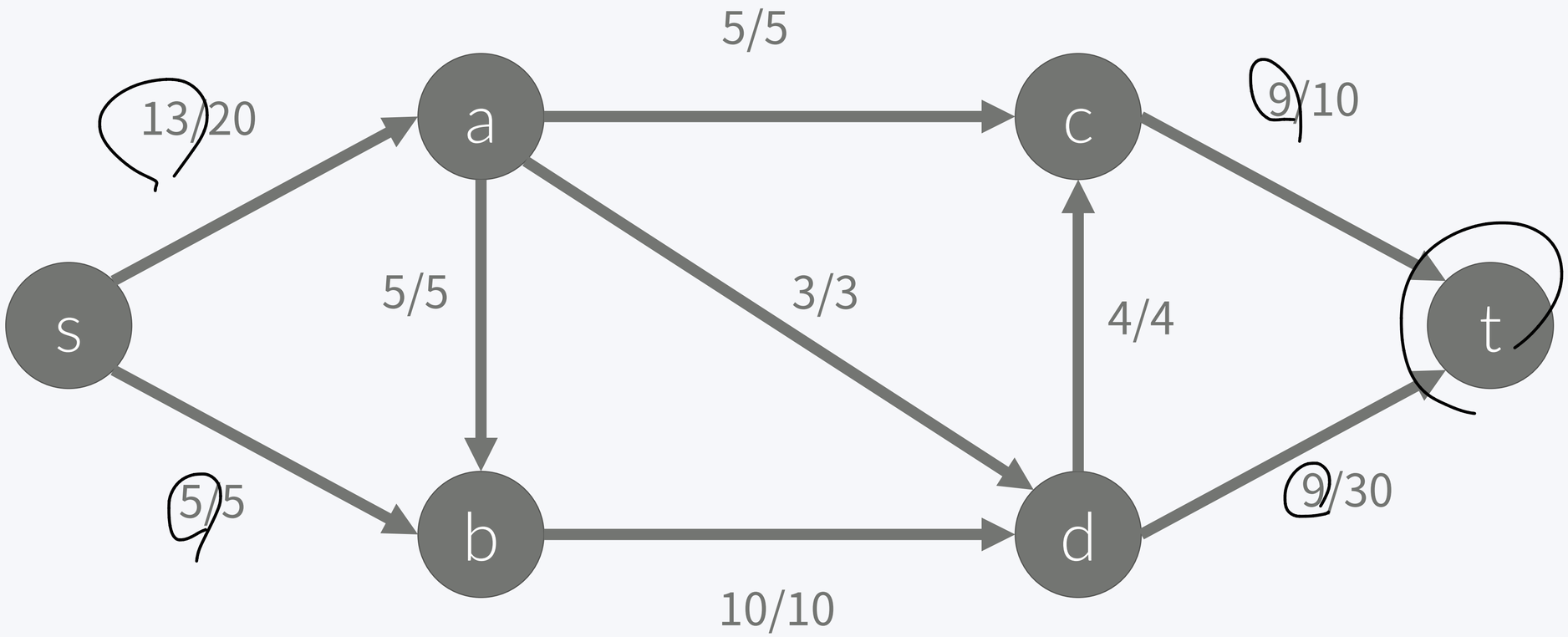
- 간선에 나타나있는 것: capacity
- 실제 그 간선을 따라서 흐른 양: flow



최대 유량

Maximum Flow

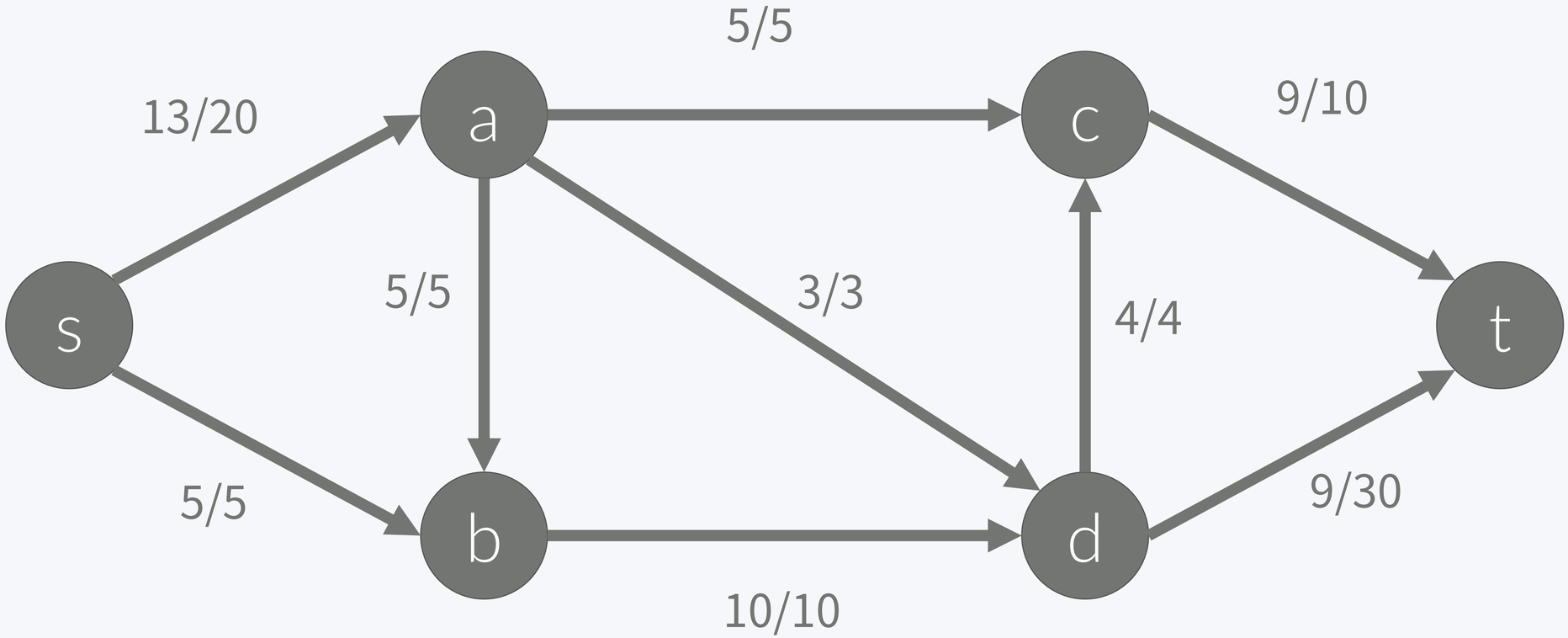
- flow/capacity



최대 유량

Maximum Flow

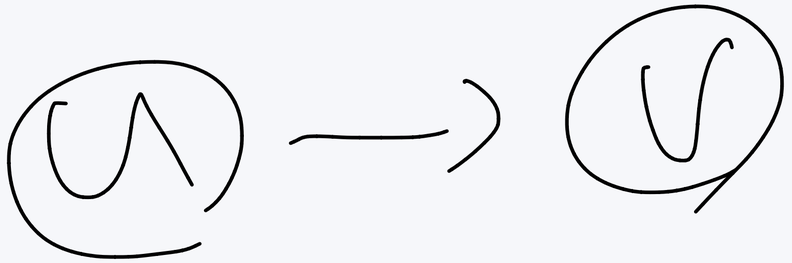
- 다른 예시 vertex: 교차로, edge: 도로
- capacity: 1분동안 그 도로를 지나갈 수 있는 차의 개수
- flow: 1분동안 그 도로를 지나가는 차의 개수



최대 유량

Maximum Flow

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

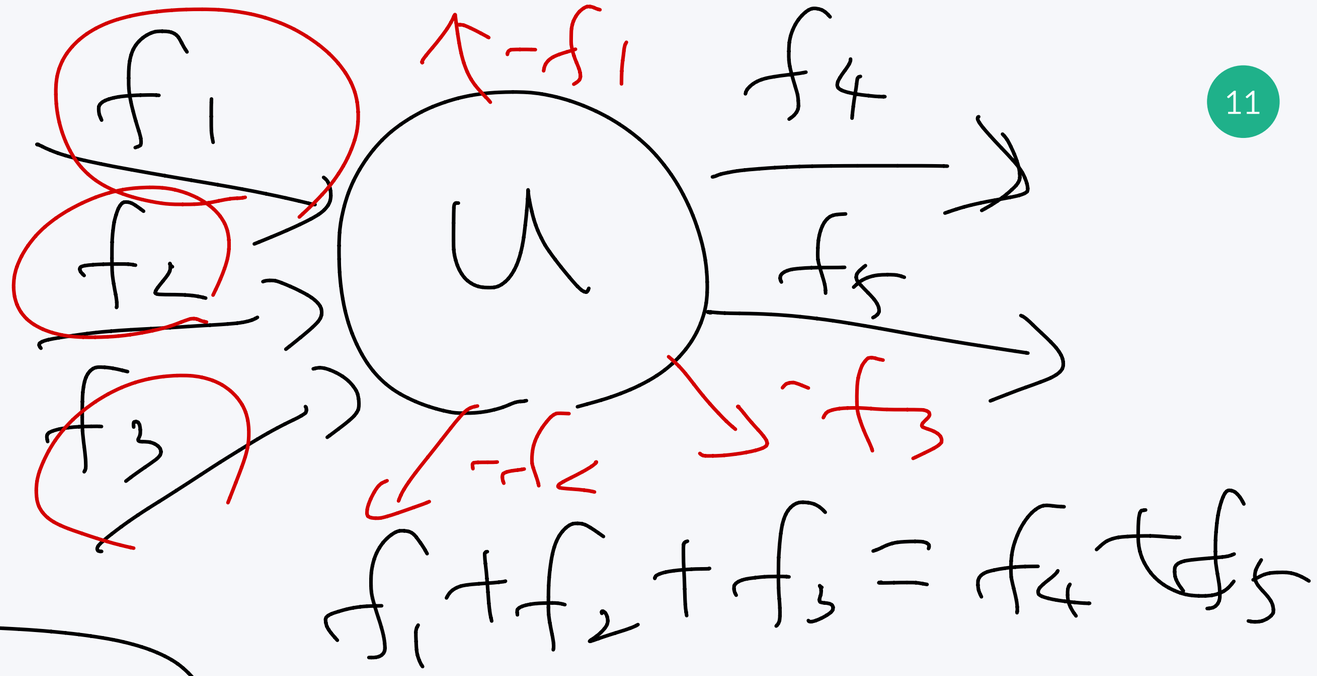


$$f(u,v) \le c(u,v)$$

$$f(u,v) = -f(v,u)$$

최대 유량

Maximum Flow



- 세가지 속성을 만족해야 한다
- Capacity constraint
  - f(u, v) ≤ c(u, v)
- Skew symmetry
  - f(u, v) = -f(v, u)
  - u->v로 x를 보냈으면, v->u로는 –x를 보낸다고 한다
- Flow conservation
  - Σv ∈Vf(u,v) = 0
  - u와 연결된 모든 간선의 흐른양의 합은 0이다
  - Skew symmetry에 의해서 음수를 저장했기 때문

최대 유량

Maximum Flow

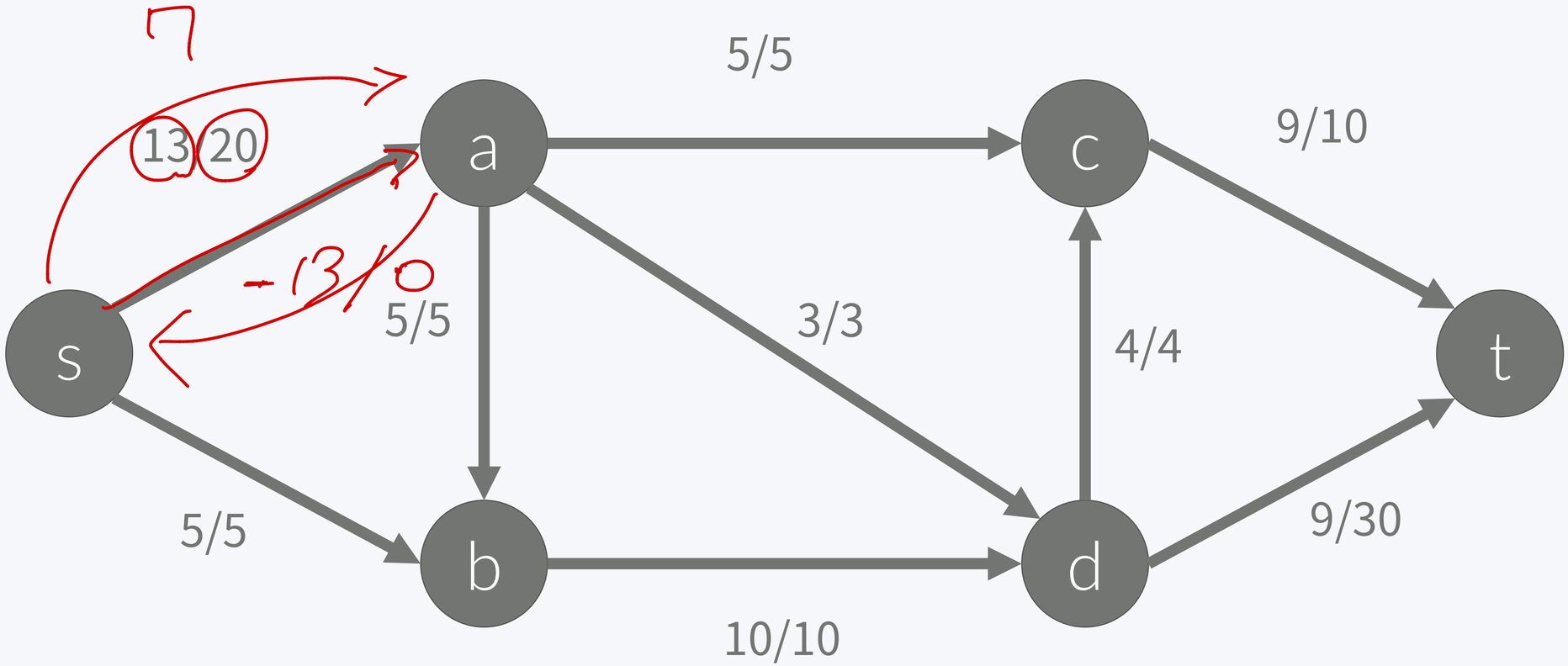
- Source (s): 시작
- Sink (t): 끝
- 총 흐른 양
  - Source에서 나간 양
  - 또는
  - Sink로 들어온 양

Residual Capacity

잔여 용량

Maximum Flow

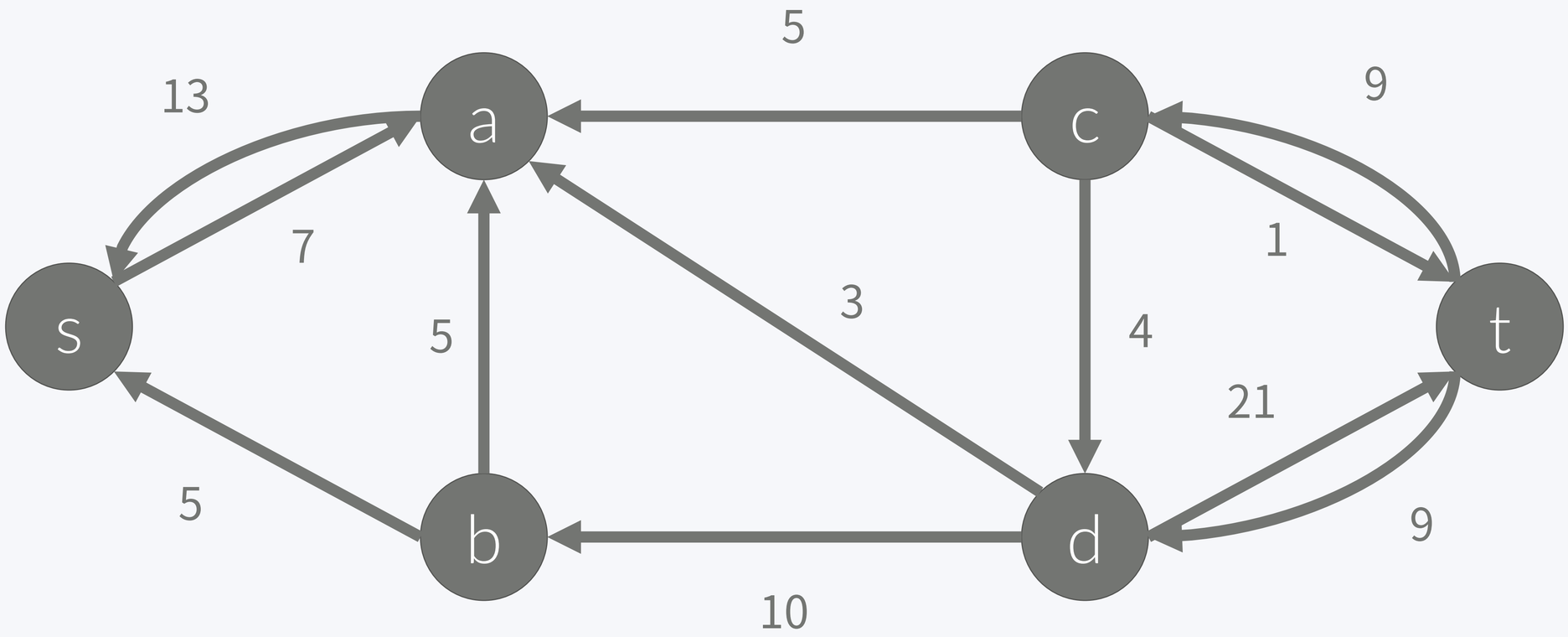
- cf(u, v) = c(u, v) – f(u, v)
- Available Capacity



Residual Capacity

Maximum Flow

- $cf(u, v) = c(u, v) - f(u, v)$



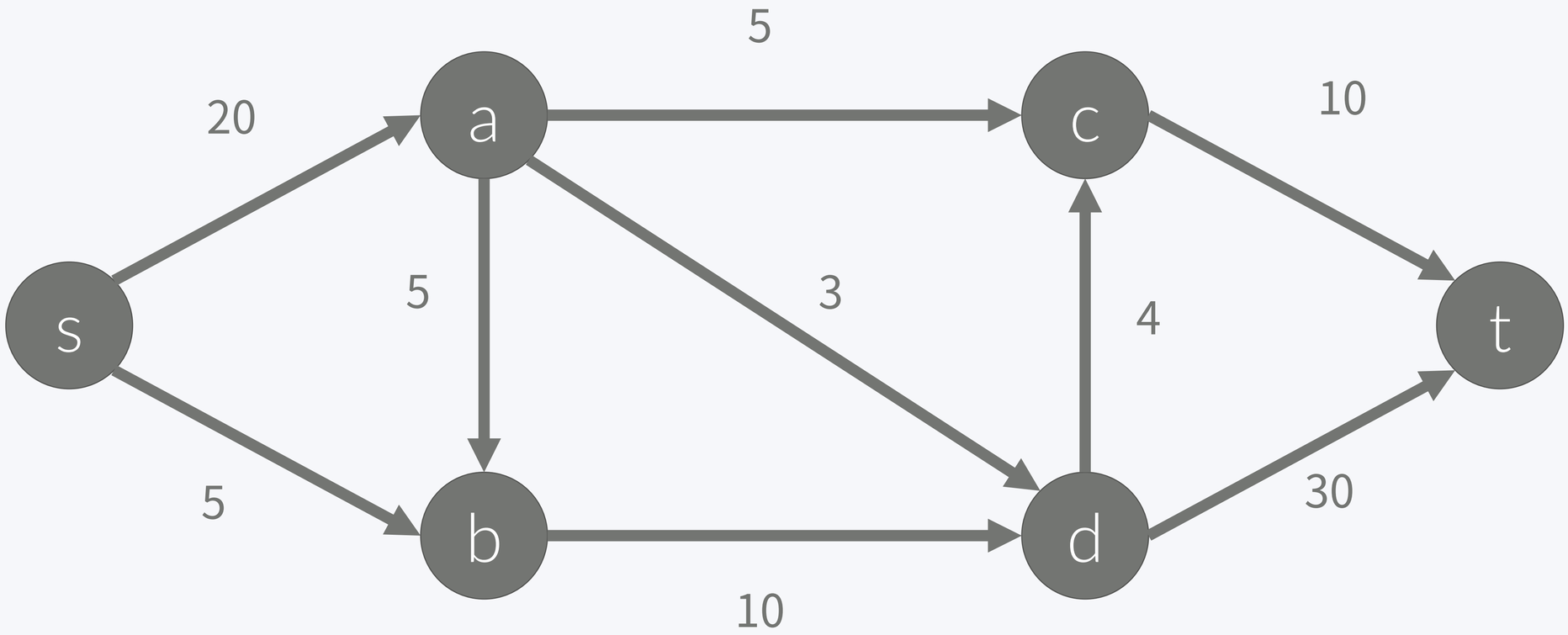
Augmenting Path

Maximum Flow

S → t 경로

아무거나

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



DFS

Ford-Fulkerson

Ford-Fulkerson

Maximum Flow

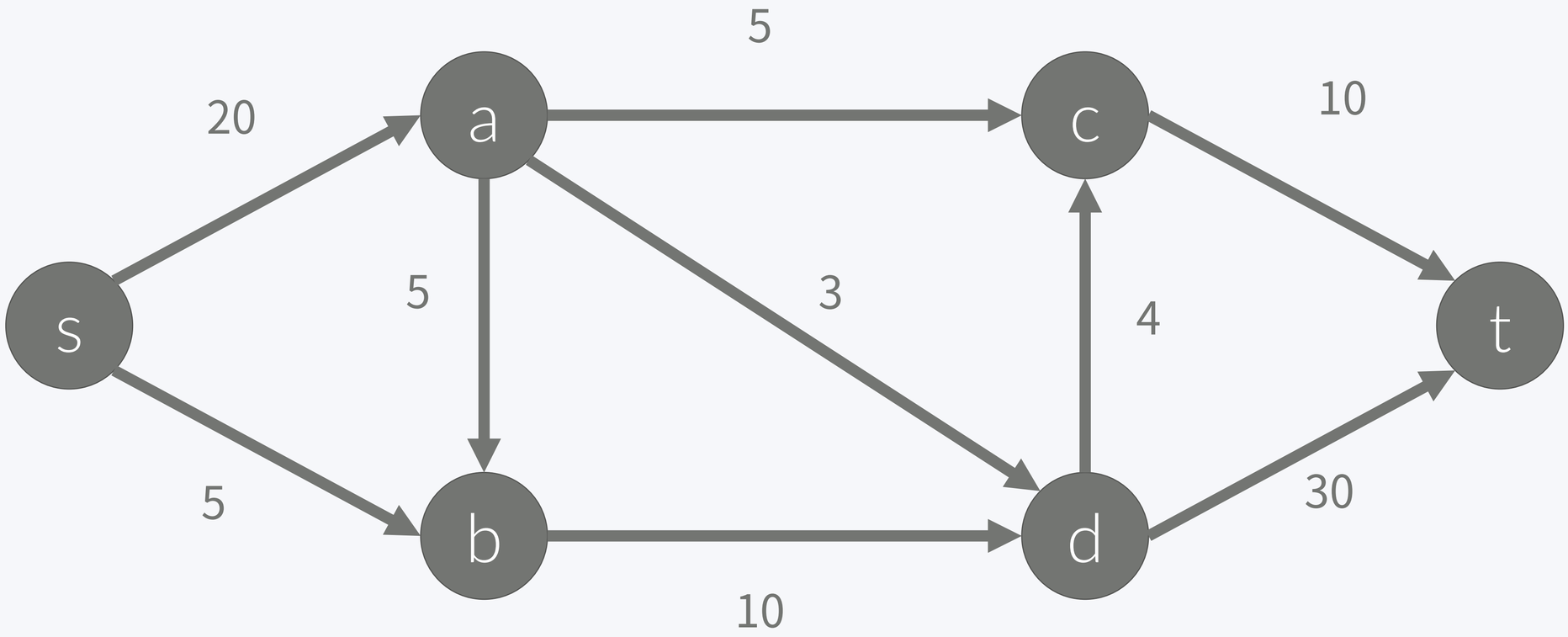
- 최대 유량을 구하는 알고리즘

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

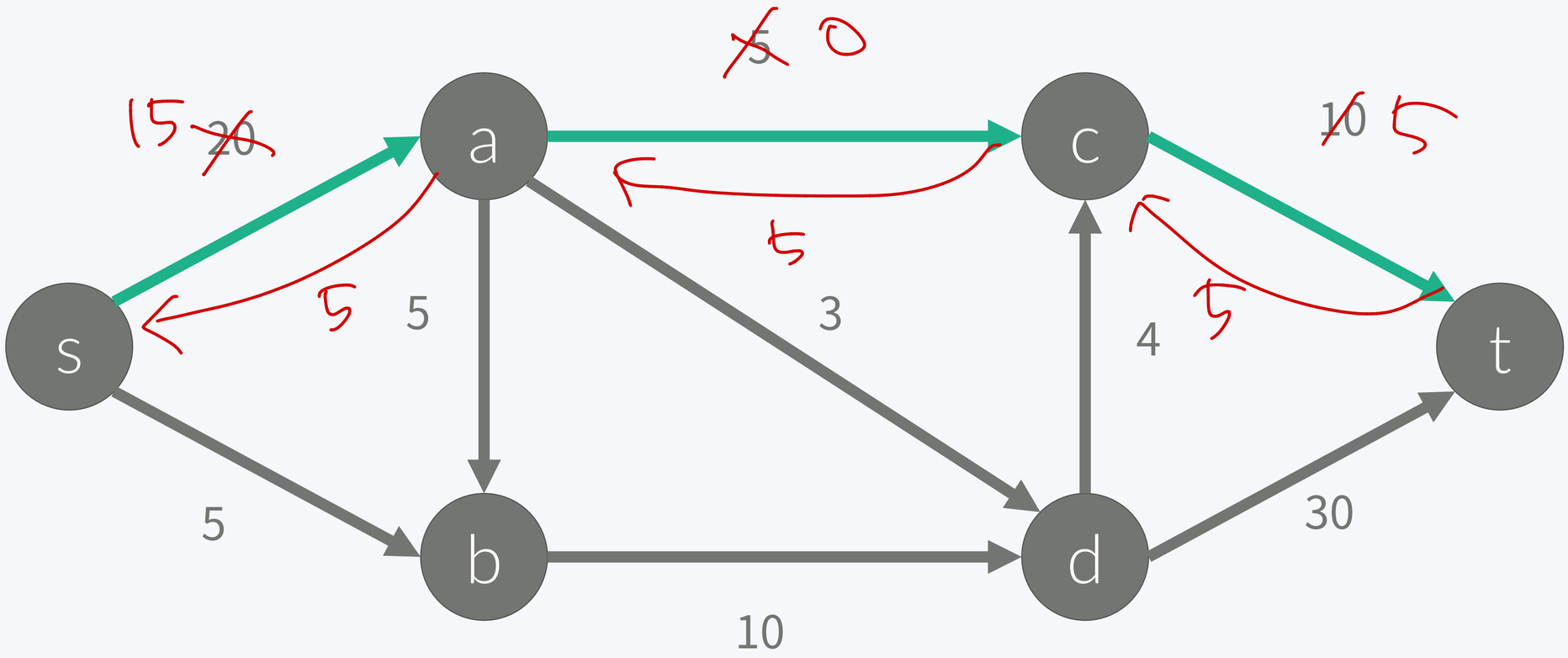
- 그래프



Ford-Fulkerson

Maximum Flow

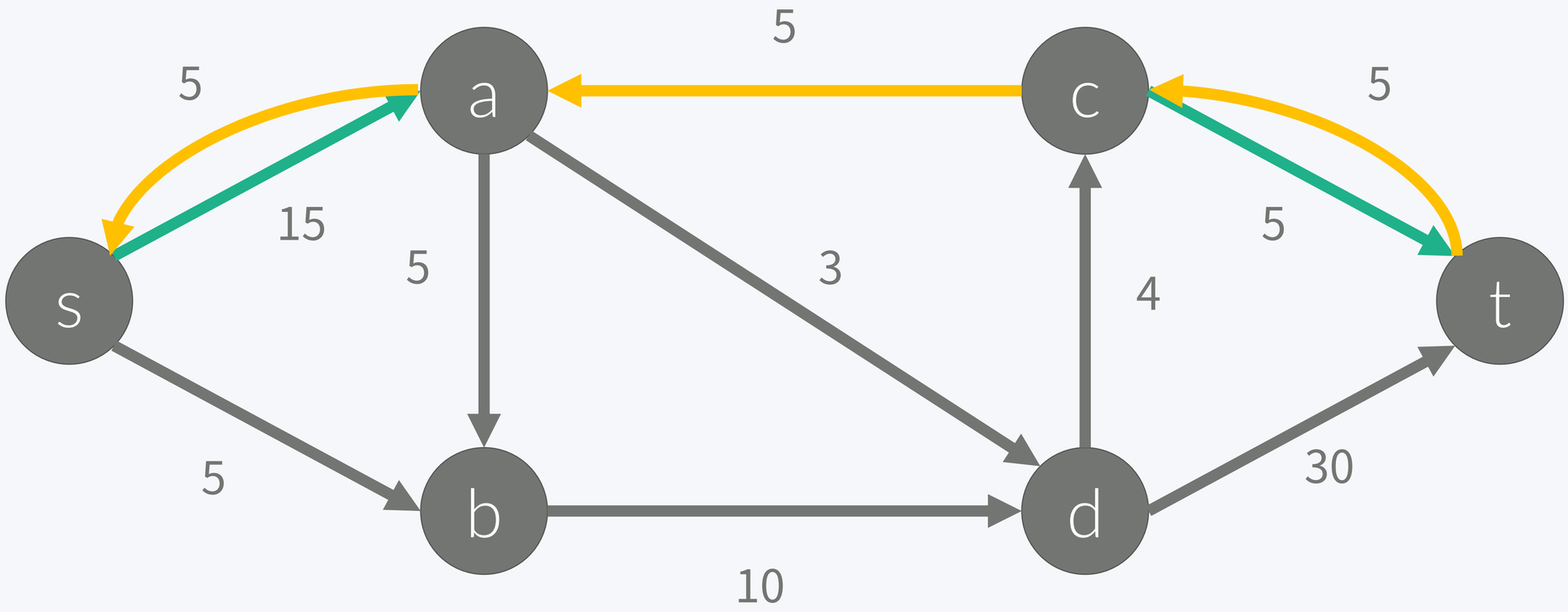
- Augmenting Path: s -> a -> c -> t
- m = 5



Ford-Fulkerson

Maximum Flow

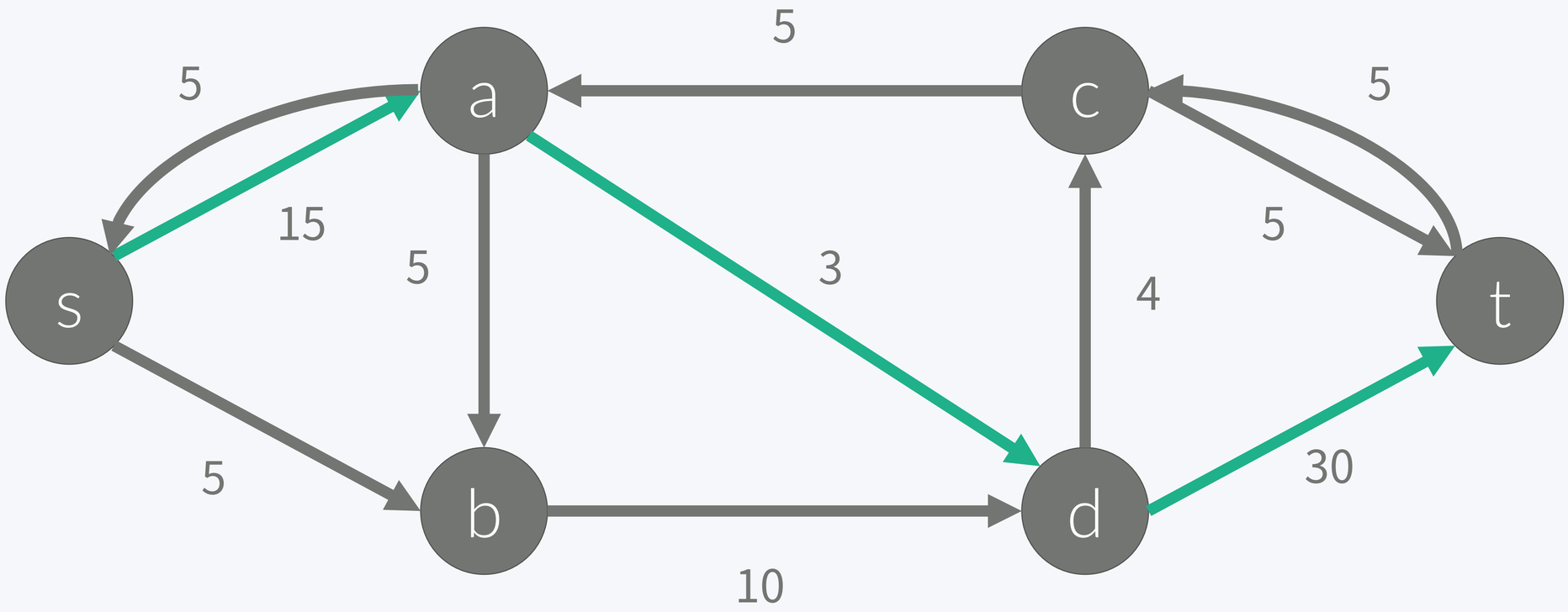
- Augmenting Path: s -> a -> c -> t
- m = 5



Ford-Fulkerson

Maximum Flow

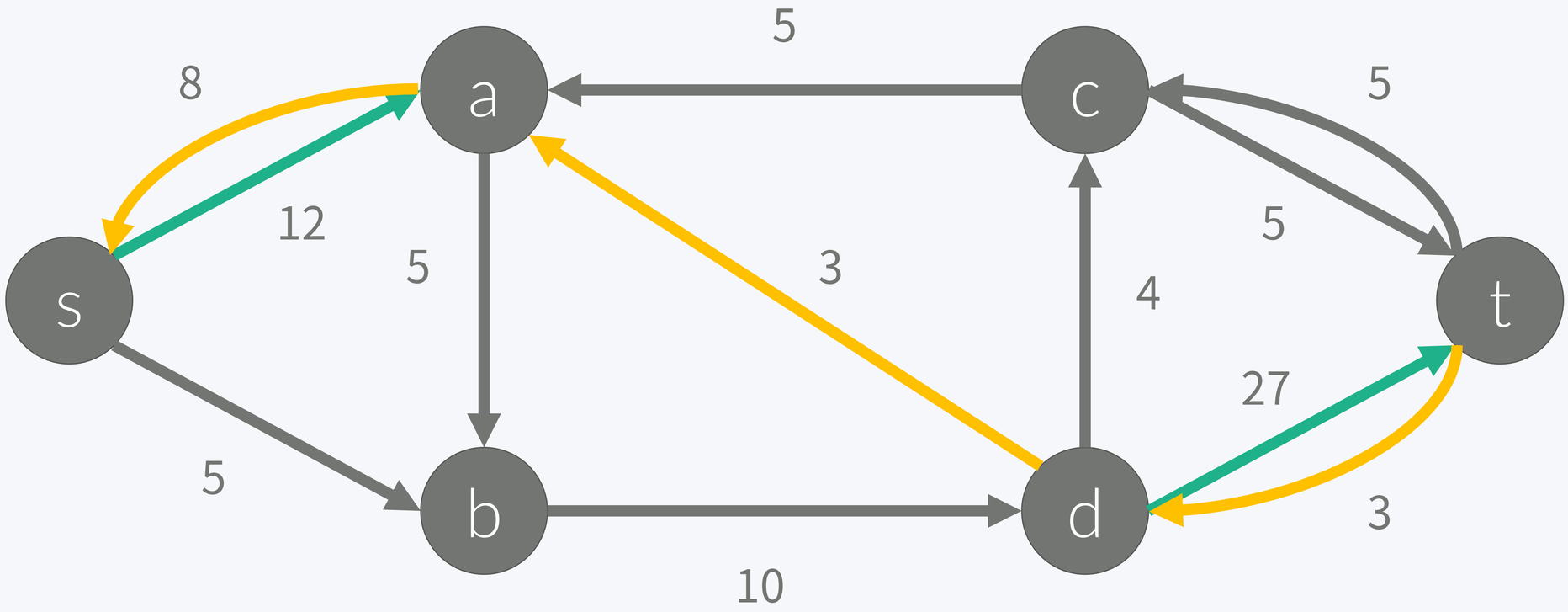
- Augmenting Path: s -> a -> d -> t
- m = 3



Ford-Fulkerson

Maximum Flow

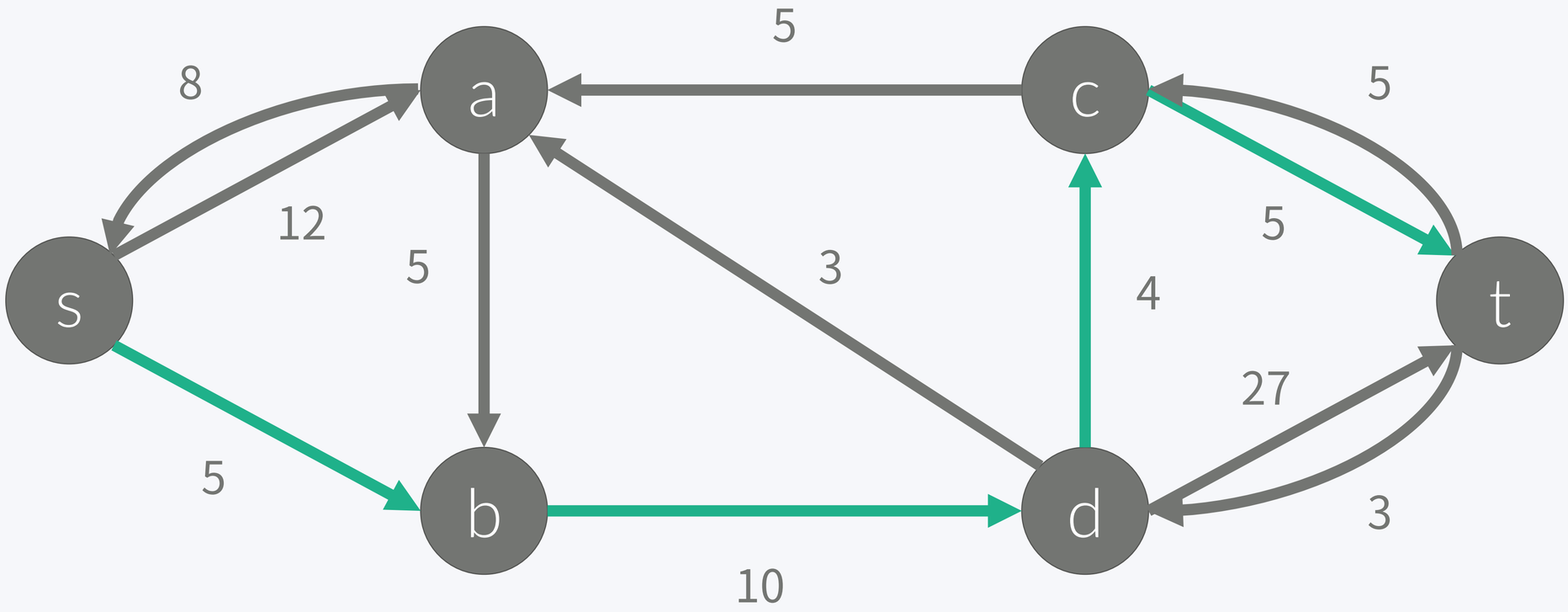
- Augmenting Path: s -> a -> d -> t
- m = 3



Ford-Fulkerson

Maximum Flow

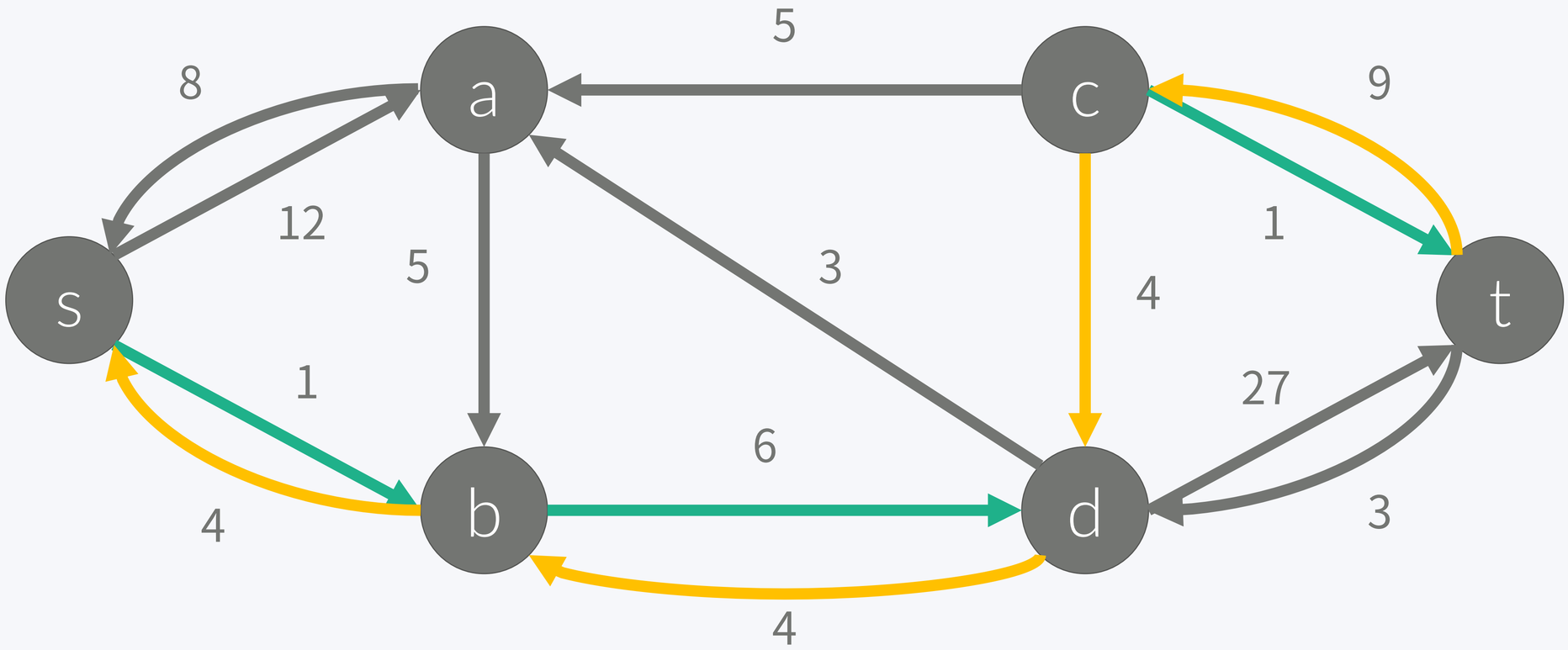
- Augmenting Path: s -> b -> d -> c -> t
- m = 4



Ford-Fulkerson

Maximum Flow

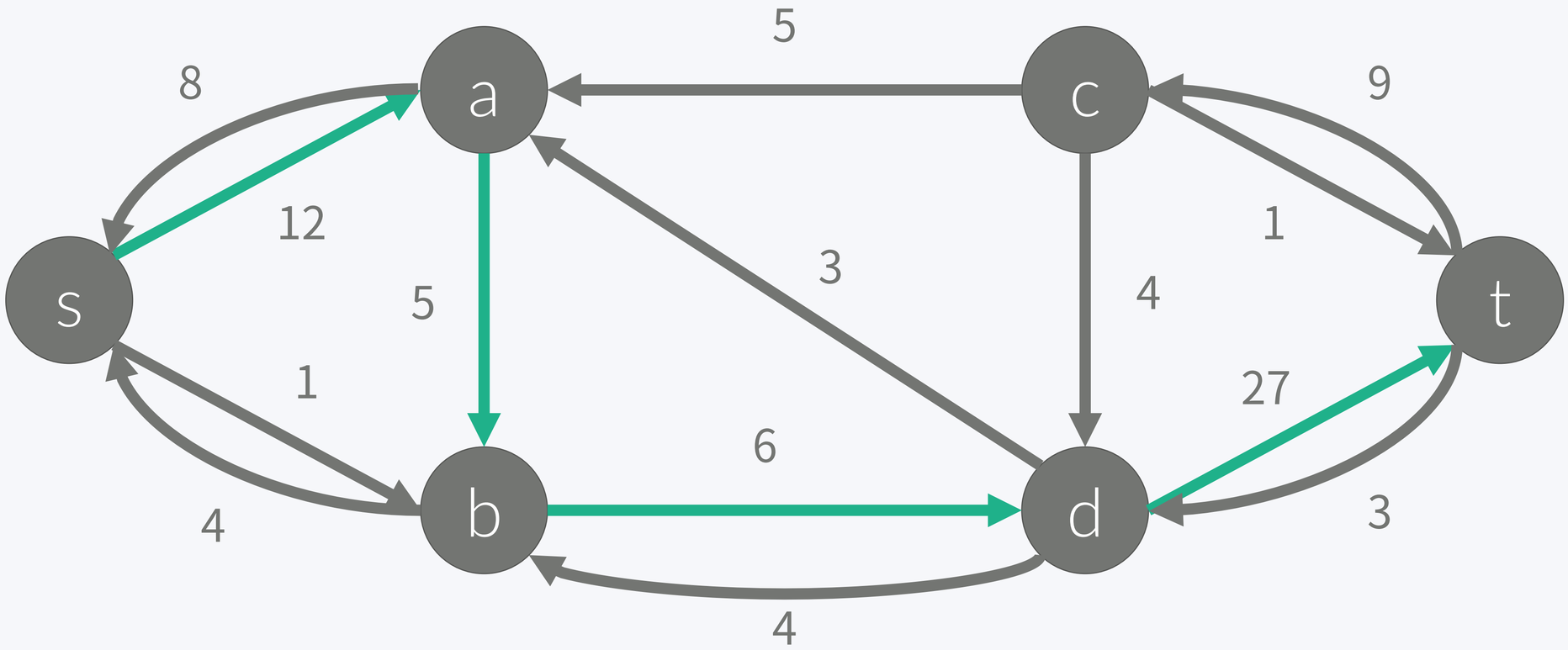
- Augmenting Path: s -> b -> d -> c -> t
- m = 4



Ford-Fulkerson

Maximum Flow

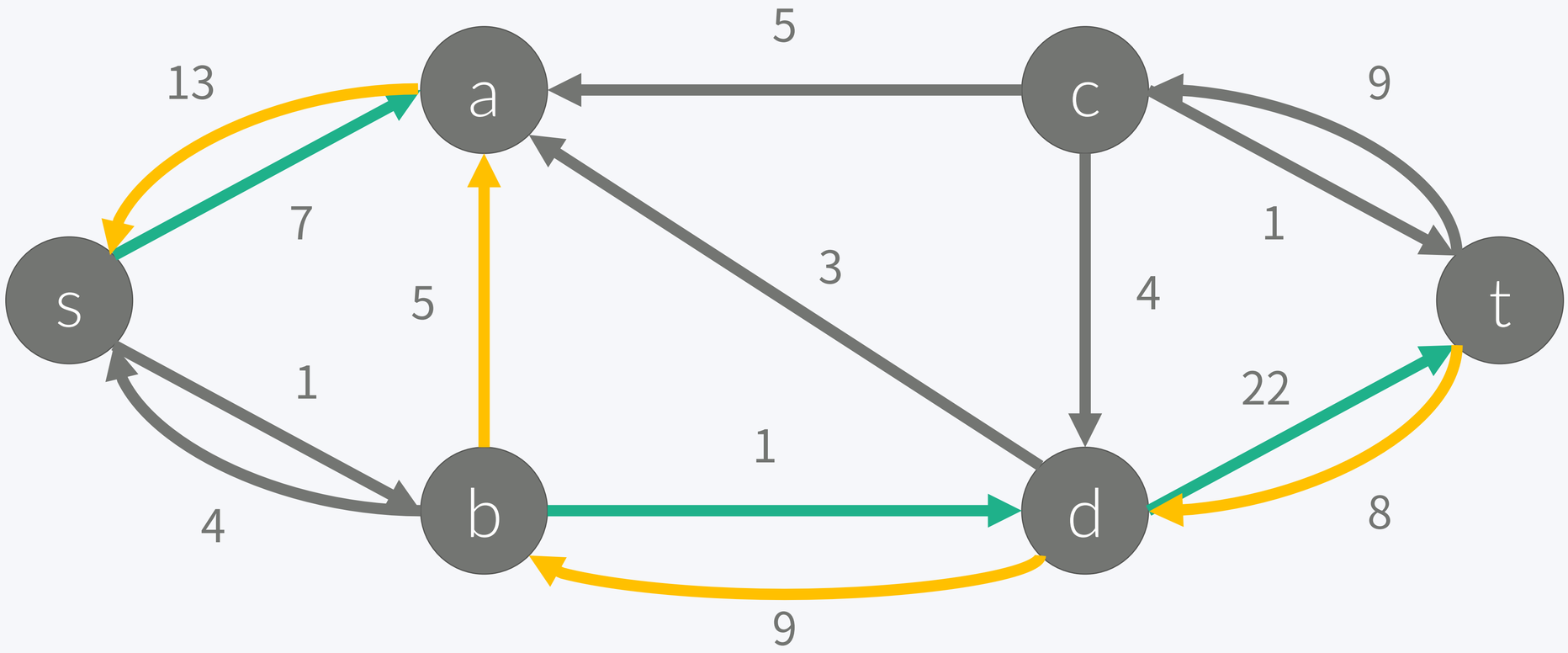
- Augmenting Path: s-> a-> b-> d-> t
- m = 5



Ford-Fulkerson

Maximum Flow

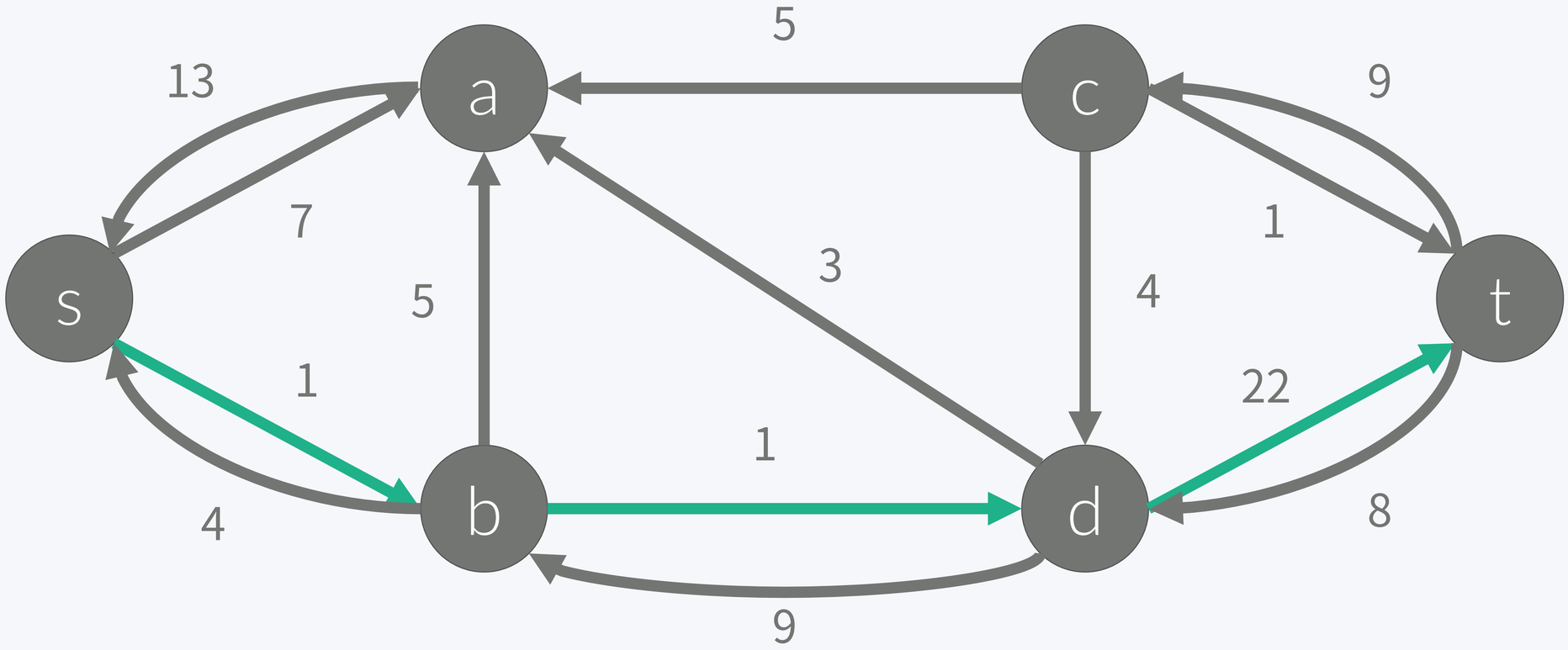
- Augmenting Path: s-> a-> b-> d-> t
- m = 5



Ford-Fulkerson

Maximum Flow

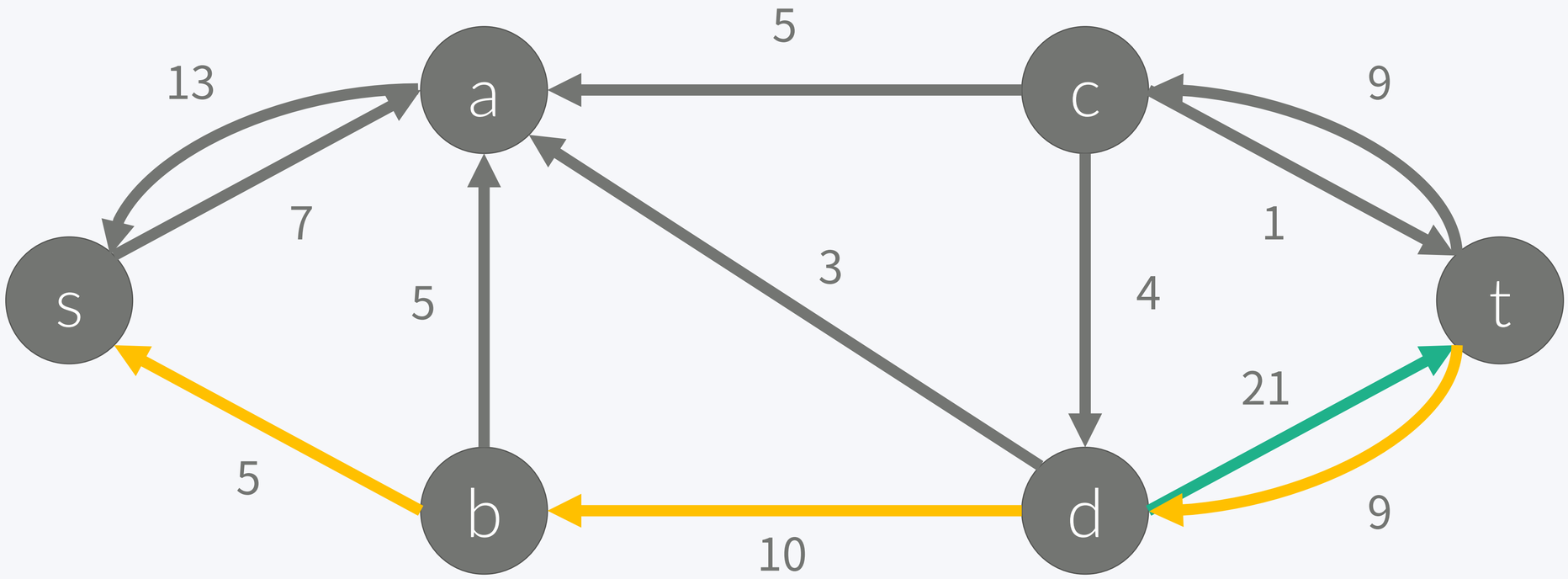
- Augmenting Path: s ->b -> d -> t
- m = 1



Ford-Fulkerson

Maximum Flow

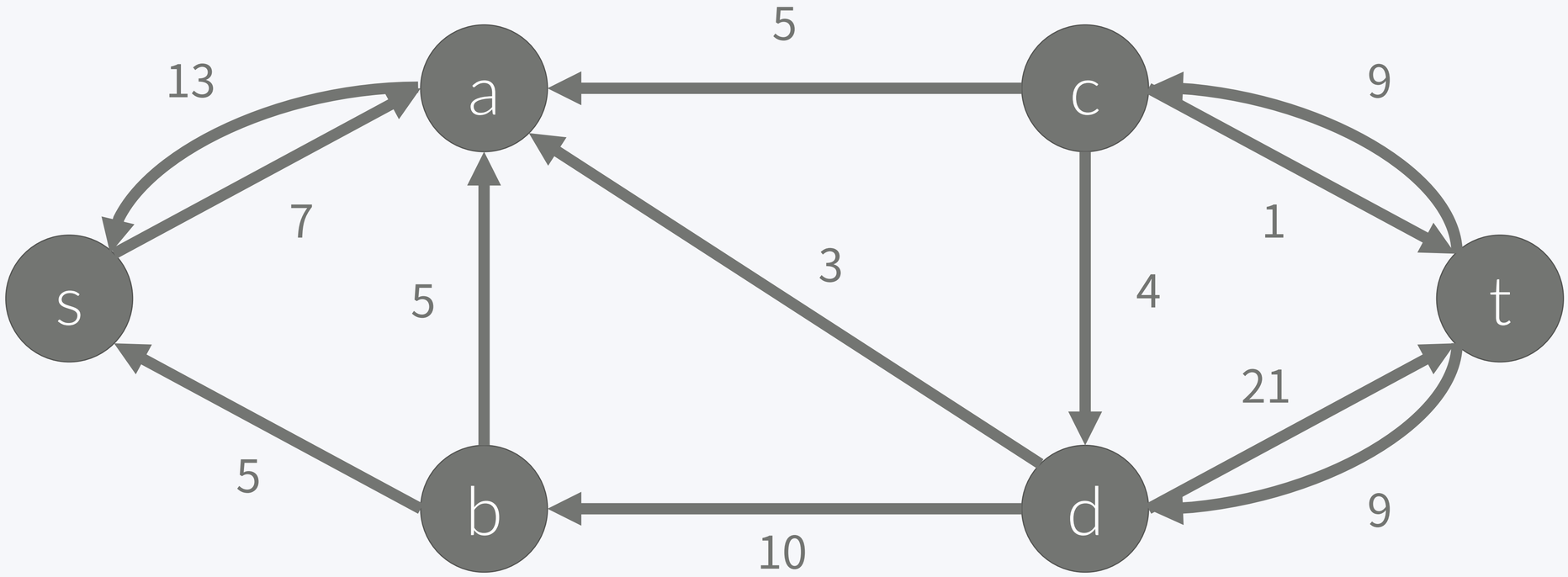
- Augmenting Path: s ->b -> d -> t
- m = 1



Ford-Fulkerson

Maximum Flow

- 더 이상 Augmenting path가 없다



Ford-Fulkerson

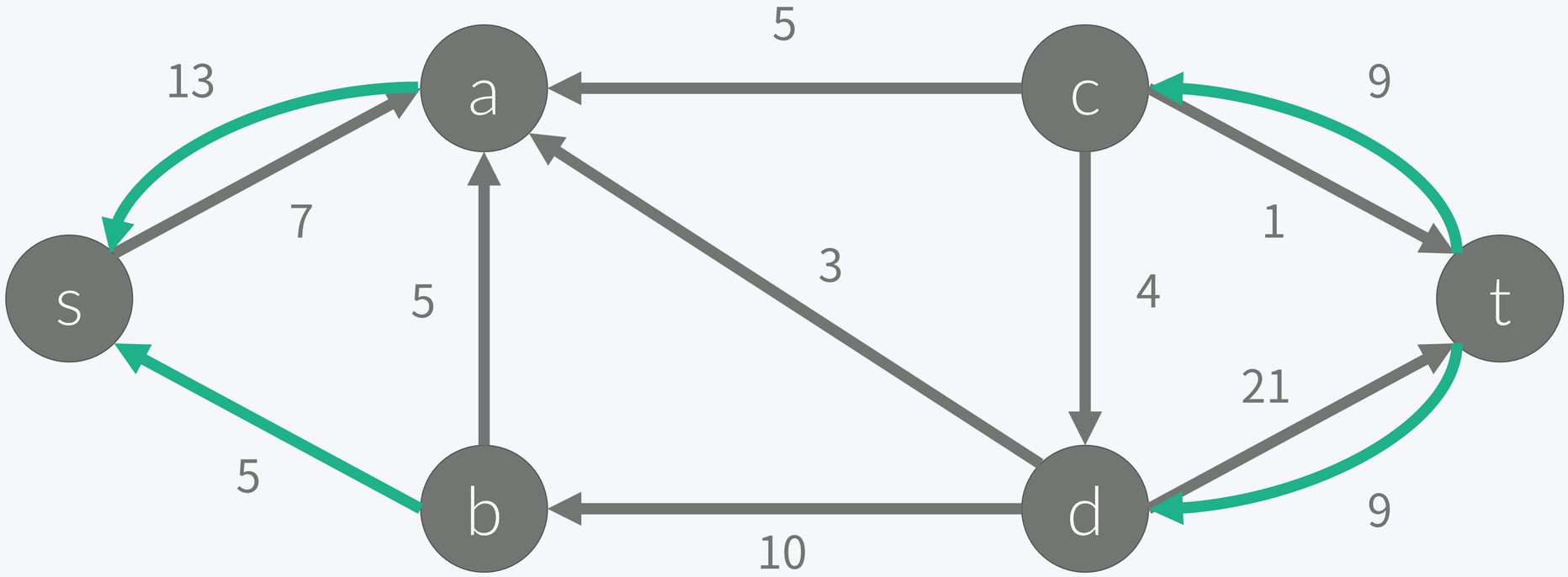
Maximum Flow

- Maximum Flow: 18

DFS : O(E)

O(E×f)

최댓값



Ford-Fulkerson

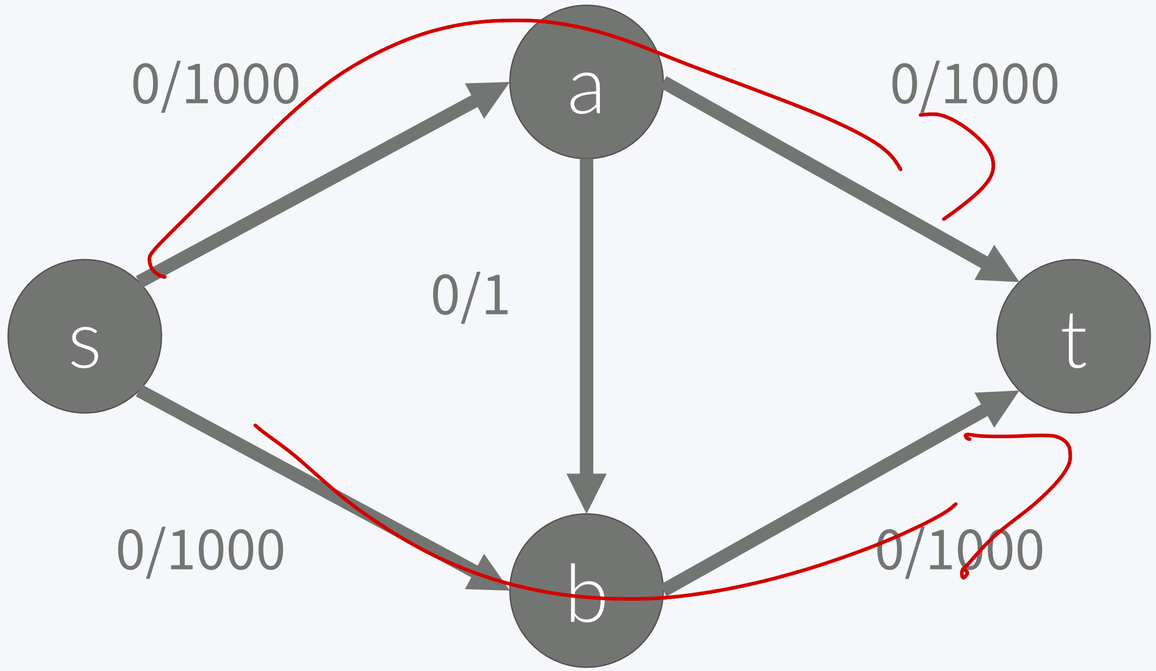
Maximum Flow

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

Ford-Fulkerson

Maximum Flow

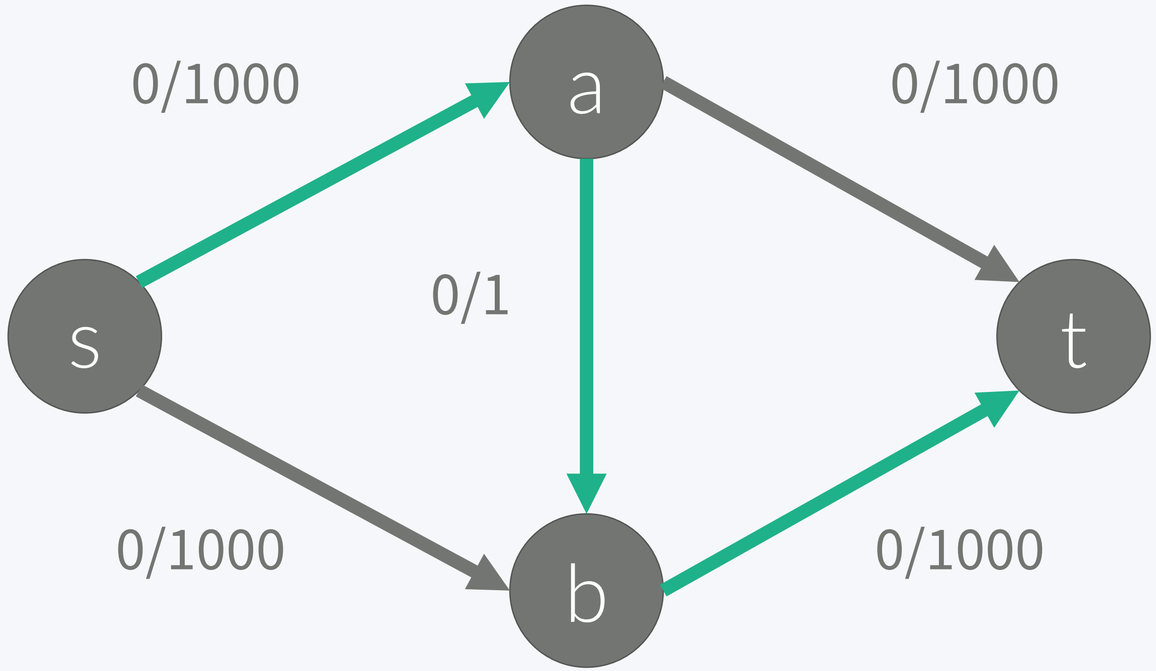
- 이런 경우



Ford-Fulkerson

Maximum Flow

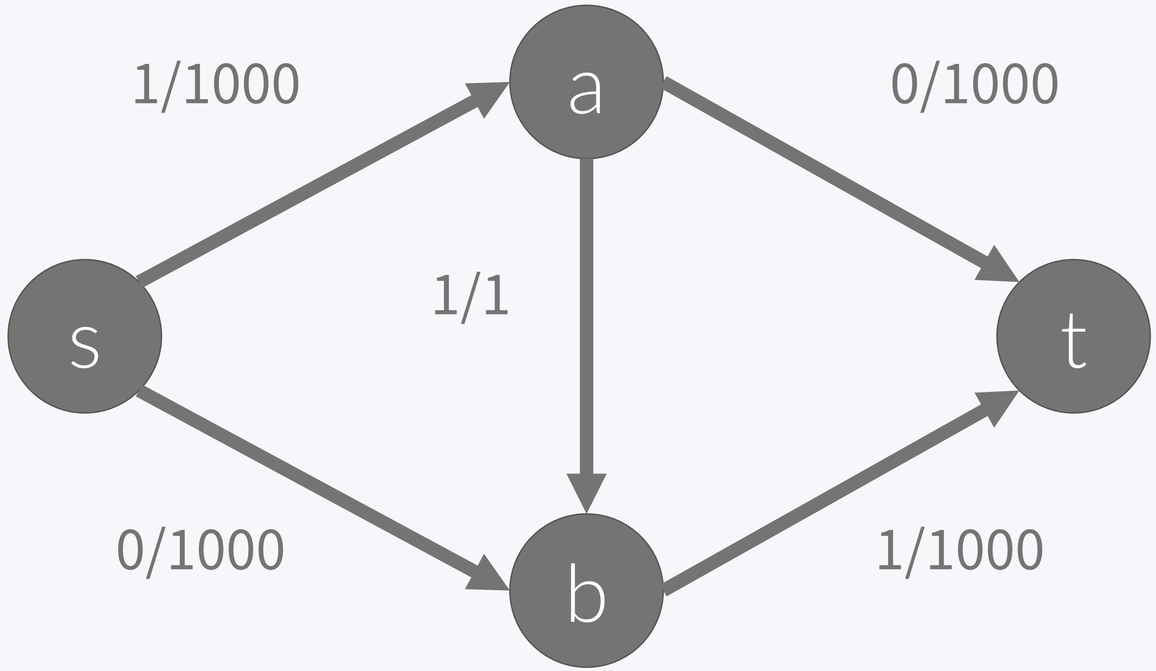
- 이런 경우



Ford-Fulkerson

Maximum Flow

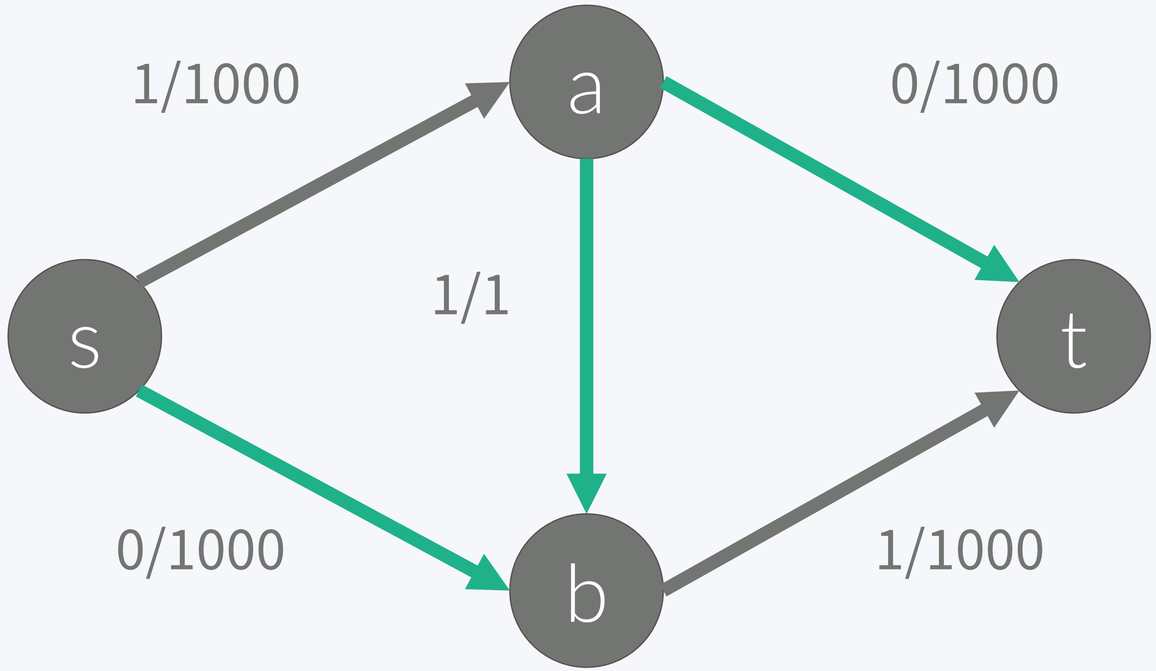
- 이런 경우



Ford-Fulkerson

Maximum Flow

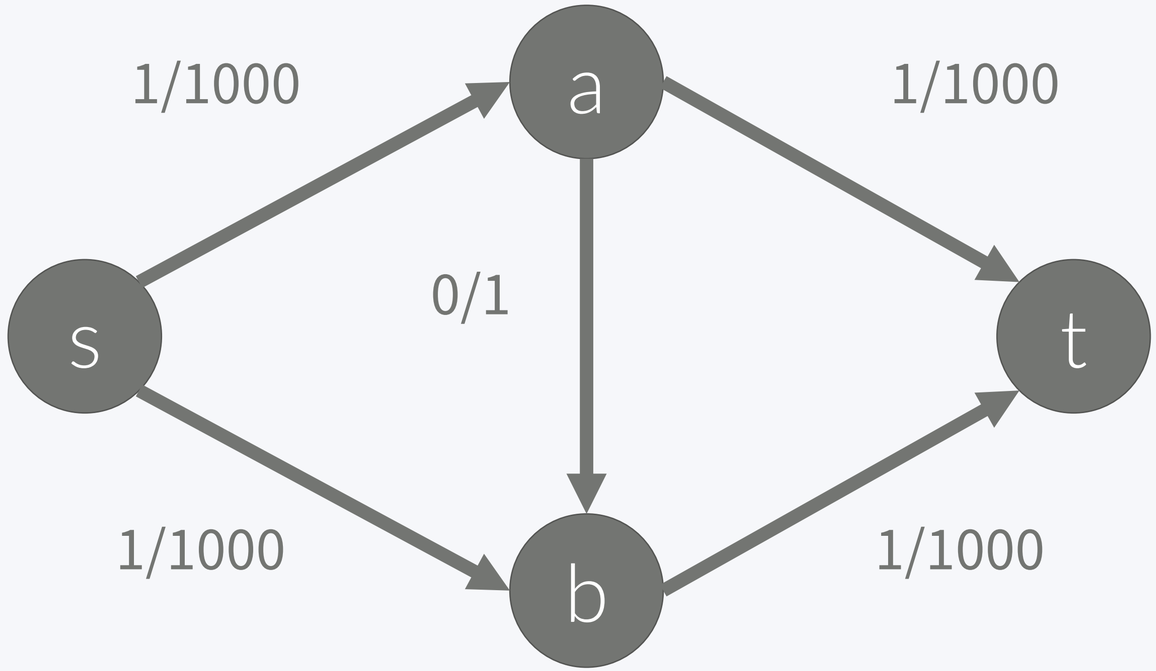
- 이런 경우



Ford-Fulkerson

Maximum Flow

- 이런 경우



Ford-Fulkerson

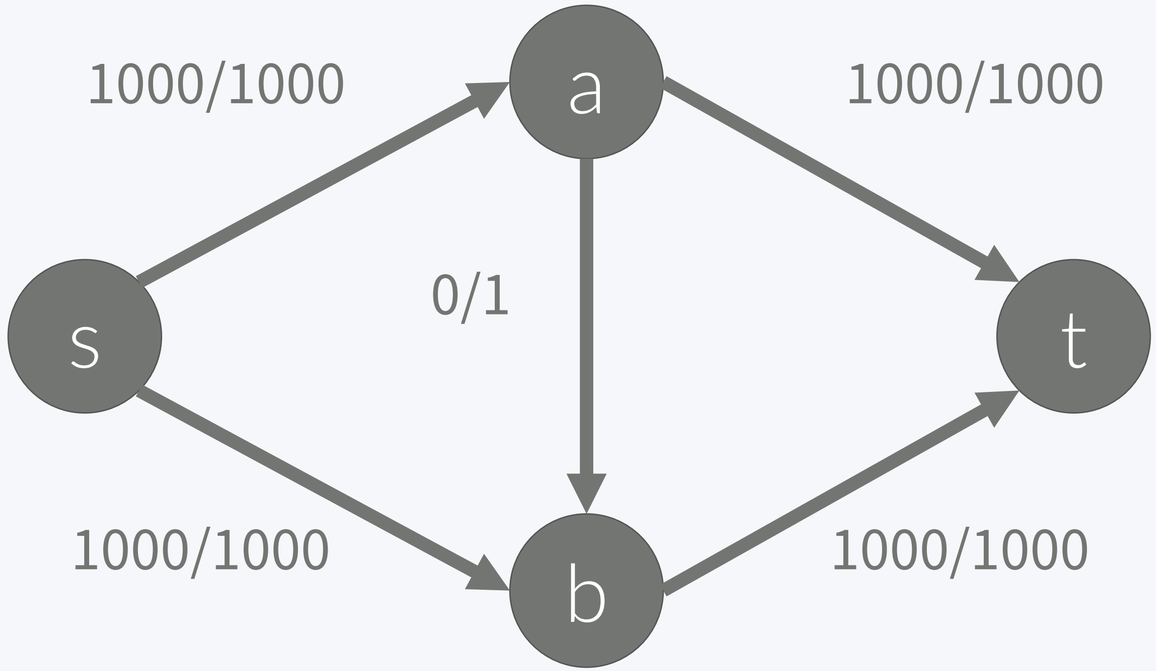
Maximum Flow

- 이 과정을 1998번 더 하면

Ford-Fulkerson

Maximum Flow

- Flow network



Edmond-Karp

Edmond-Karp

Maximum Flow

- 최대 유량을 구하는 알고리즘

$O(VE^2)$

$O(E)$ (BFS) $\times$ $O(VE)$

$E \times \frac{V}{2} - 1$

$V - 1$

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

최대 유량

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

- 최대 유량을 구하는 문제

최대 유량

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

- Ford Fulkerson 소스: http://boj.kr/8288724169874211 9ef3ba95b8690108
- Edmond Karp 소스: http://boj.kr/1b21cc1dce10447c9c341c5915c218db

DFS : $O(E \cdot f)$

BFS : $O(VE^2)$

Dinic $O(V^2E)$

이분 매칭

이분 매칭

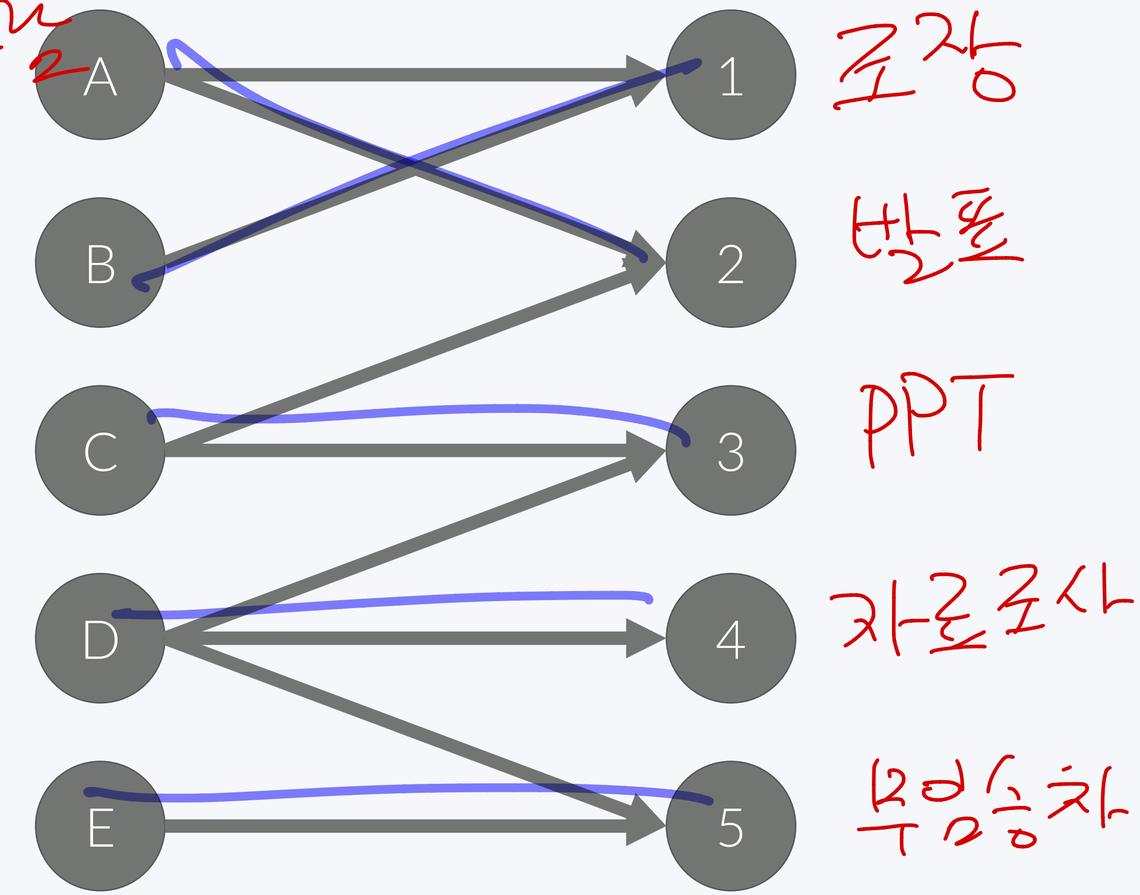
Bipartite Matching

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

이분 매칭

Bipartite Matching

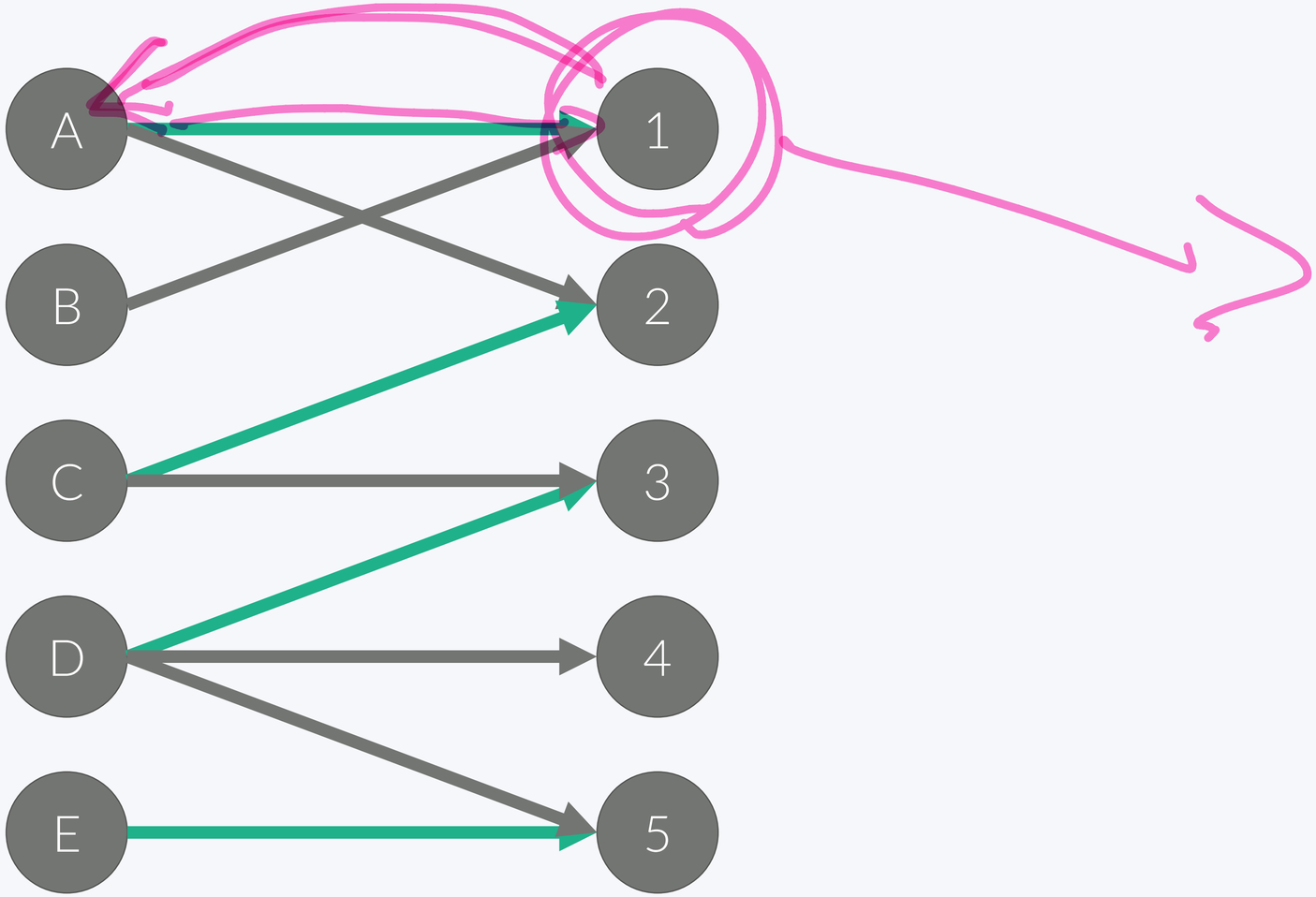
- 사람: A~E, 일: 1~5, 각 사람이 할 수 있는 일을 Edge로 연결



이분 매칭

Bipartite Matching

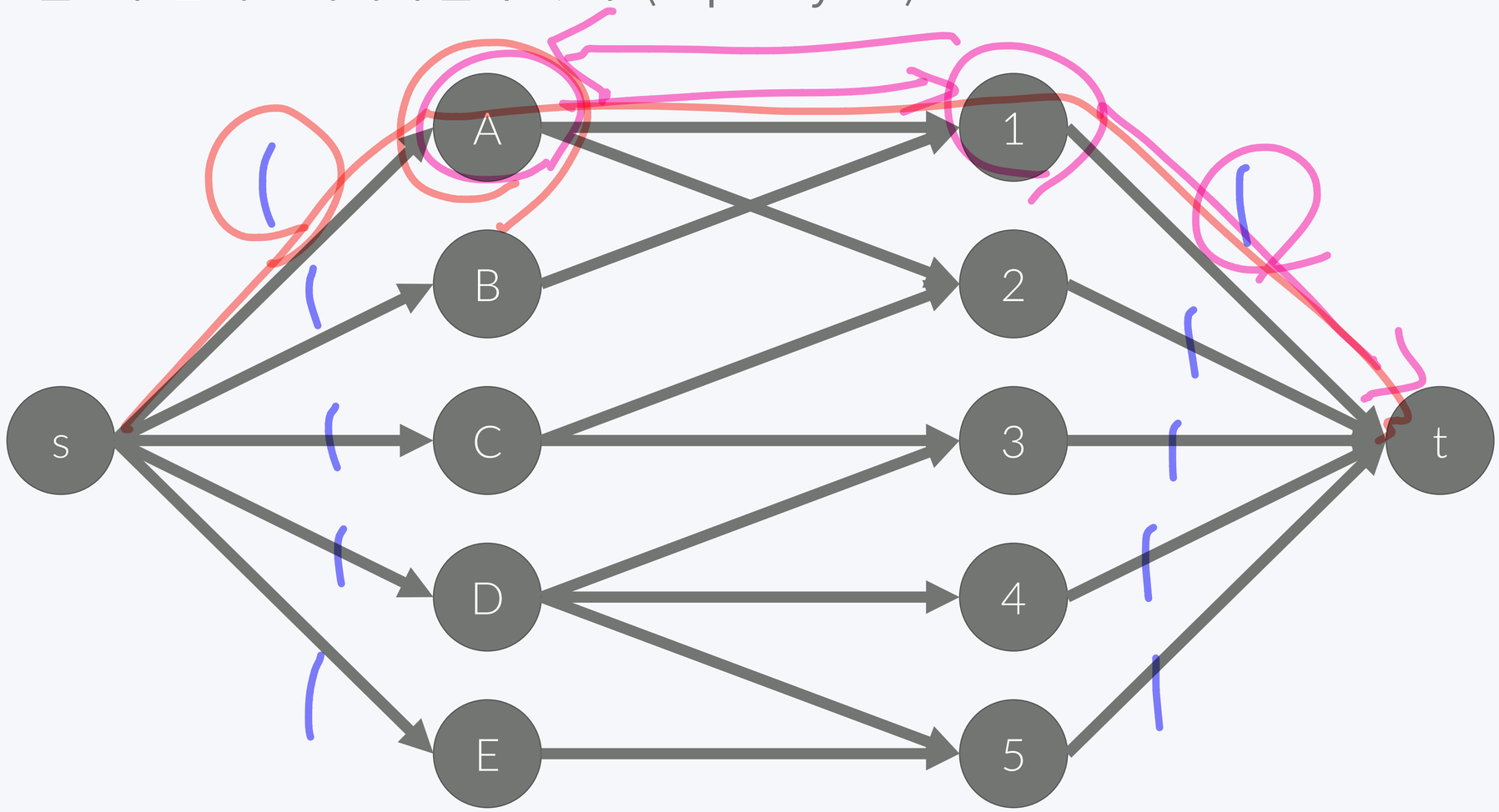
- 사람: A~E, 일: 1~5, 각 사람이 할 수 있는 일을 Edge로 연결



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

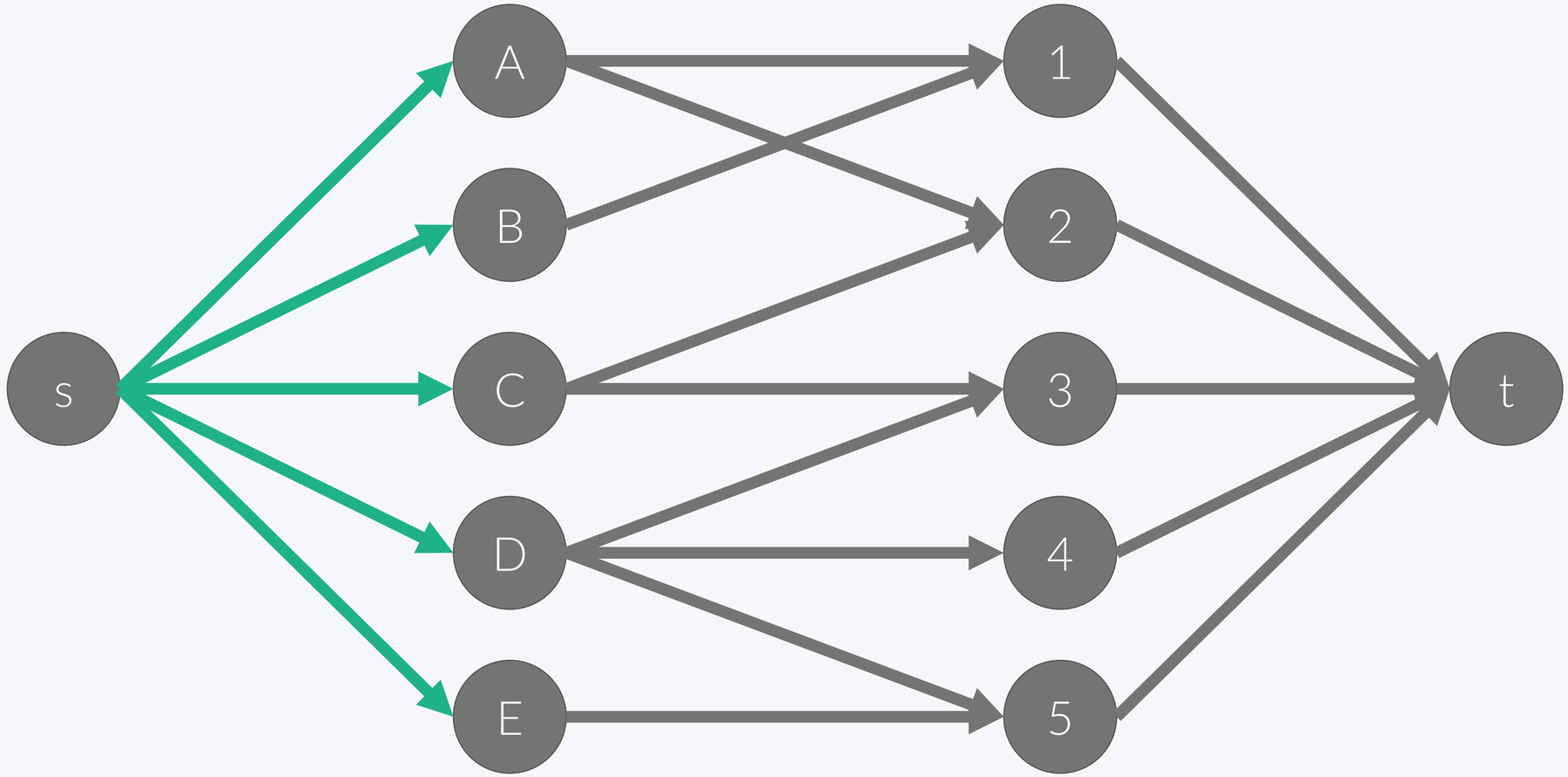
https://www.acmicpc.net/problem/11375

- 소스: http://boj.kr/d4dcb76918524f09b95fe7f516103dbe
- 소스: http://boj.kr/e03ae7c267ea406d8fa80c6045de859b

이분 매칭

Bipartite Matching

- 항상 1 또는 0이기 때문에, edge를 만들지 않아도 된다



이분 매칭

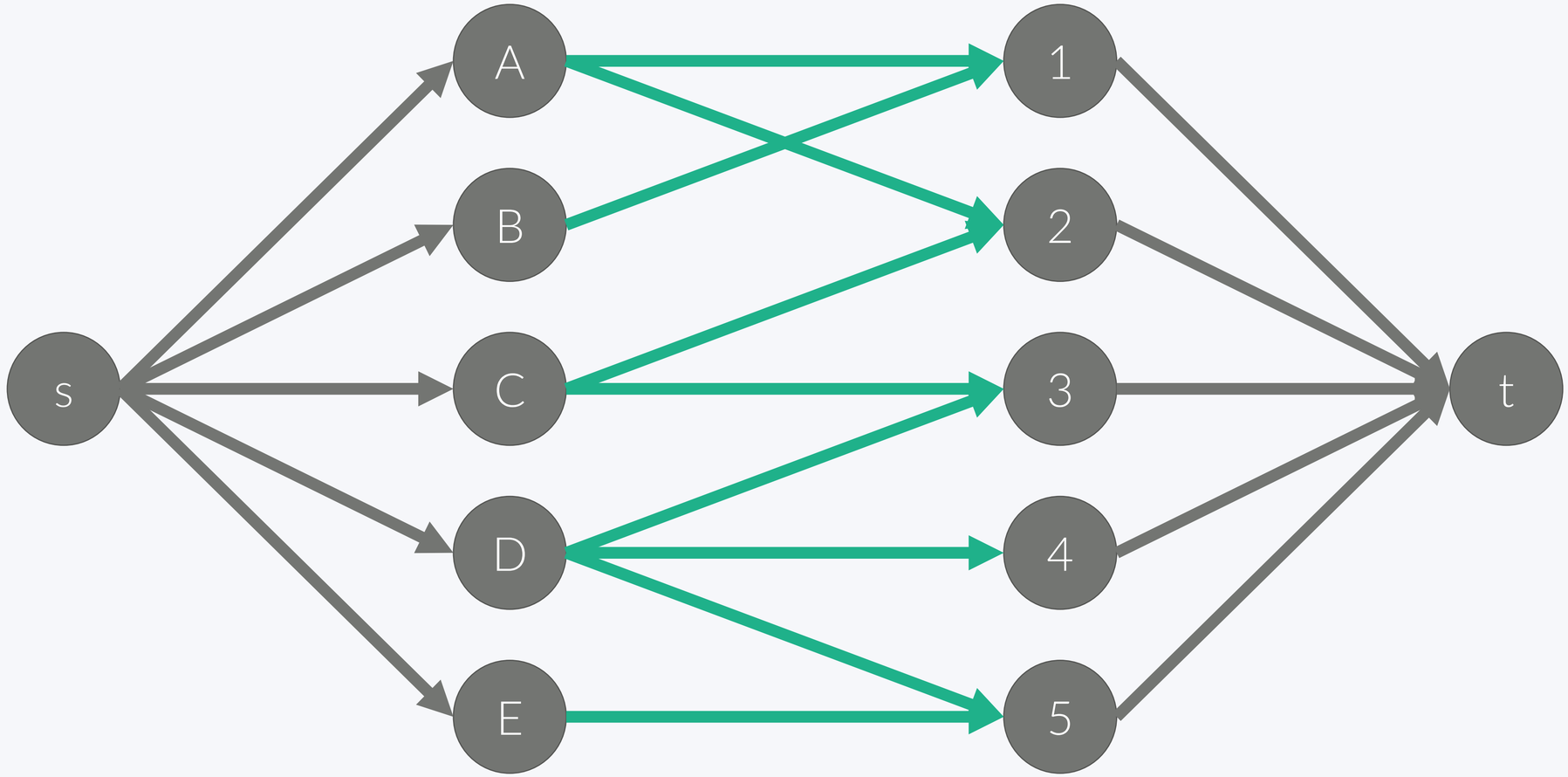
Bipartite Matching


```
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

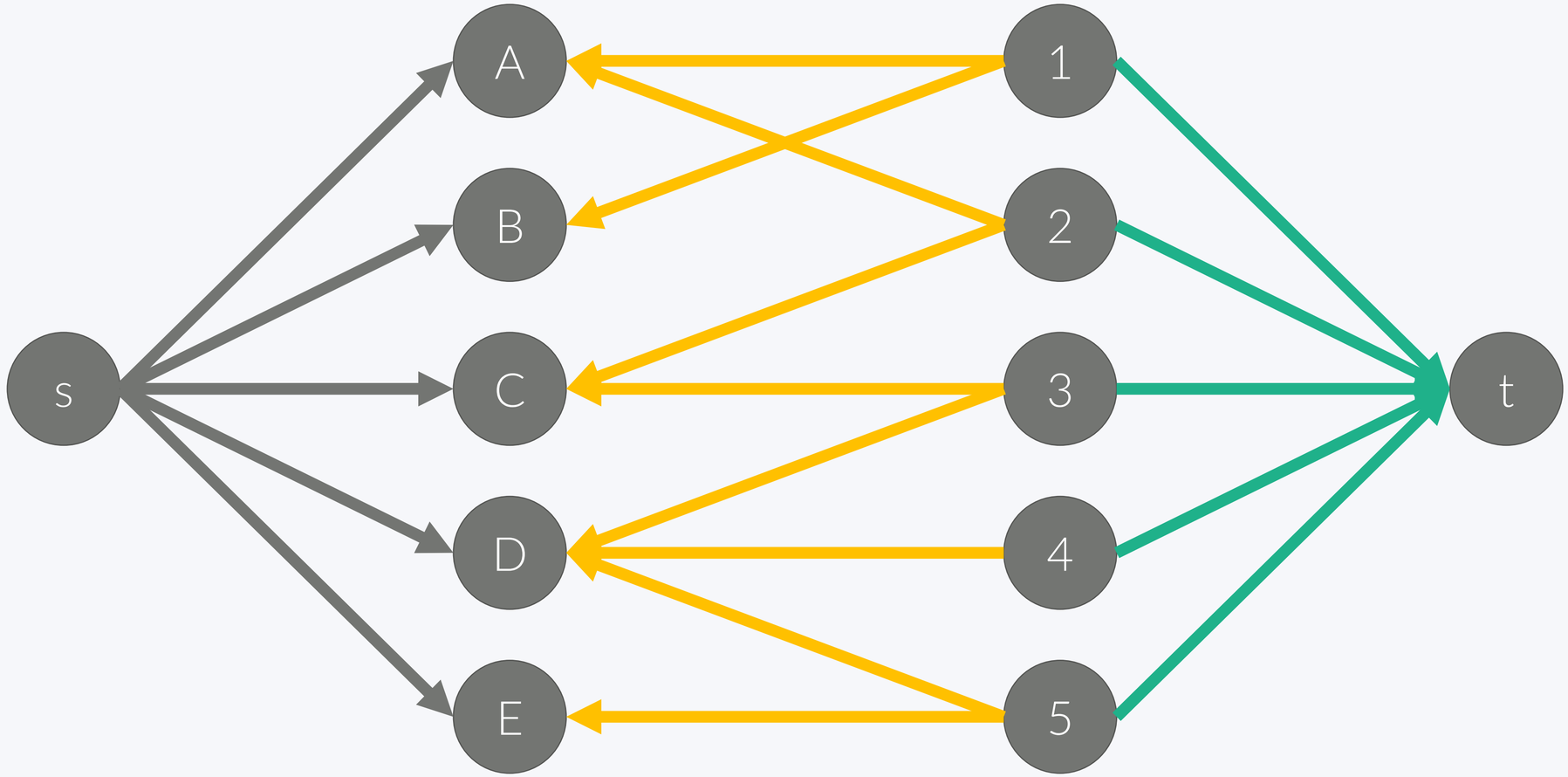
- 그대로 유지해야 한다



이분 매칭

Bipartite Matching

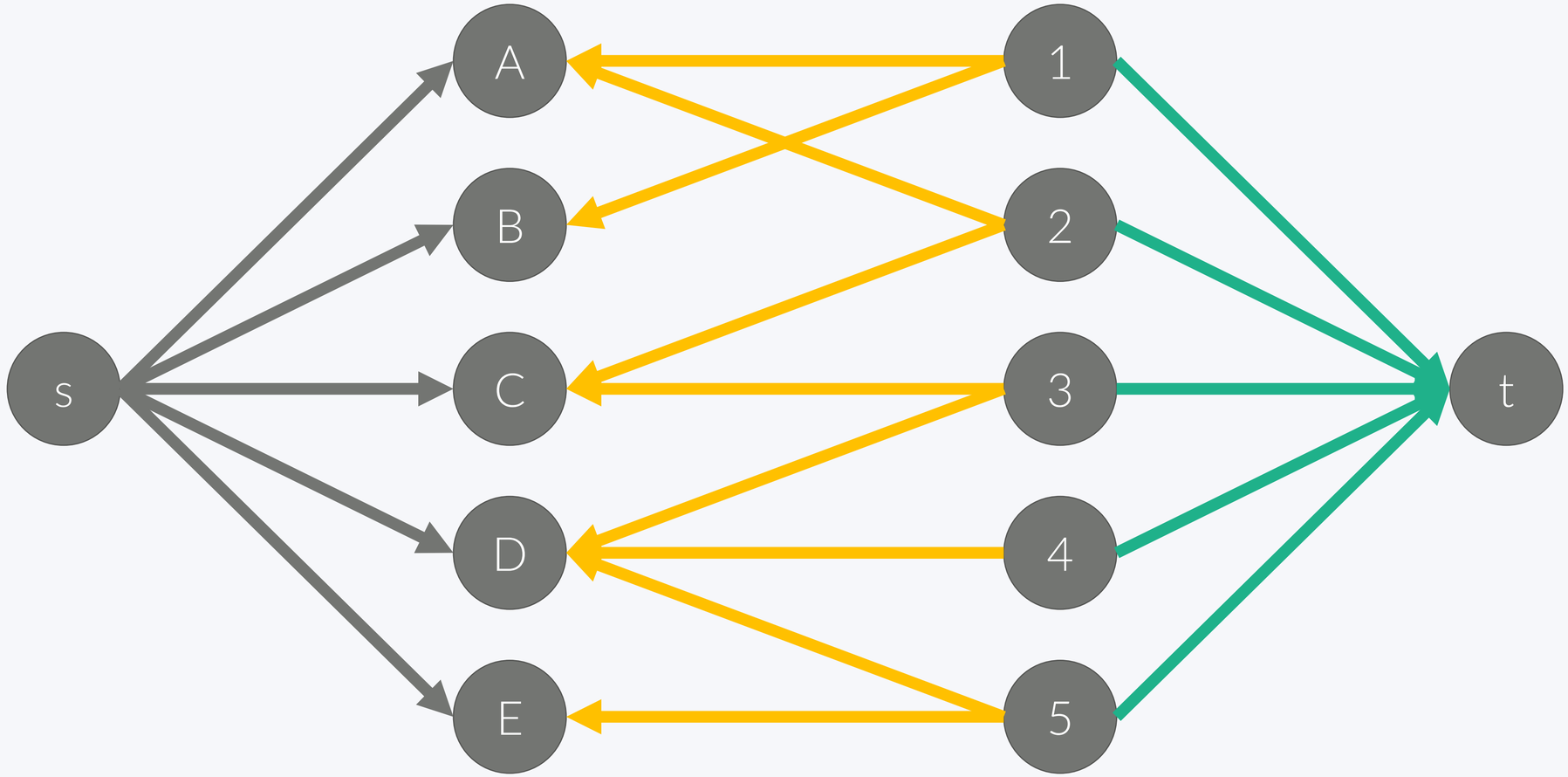
- 오른쪽은 항상 Sink로 가거나 아니면 다시 왼쪽으로 가게된다



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;
}
```

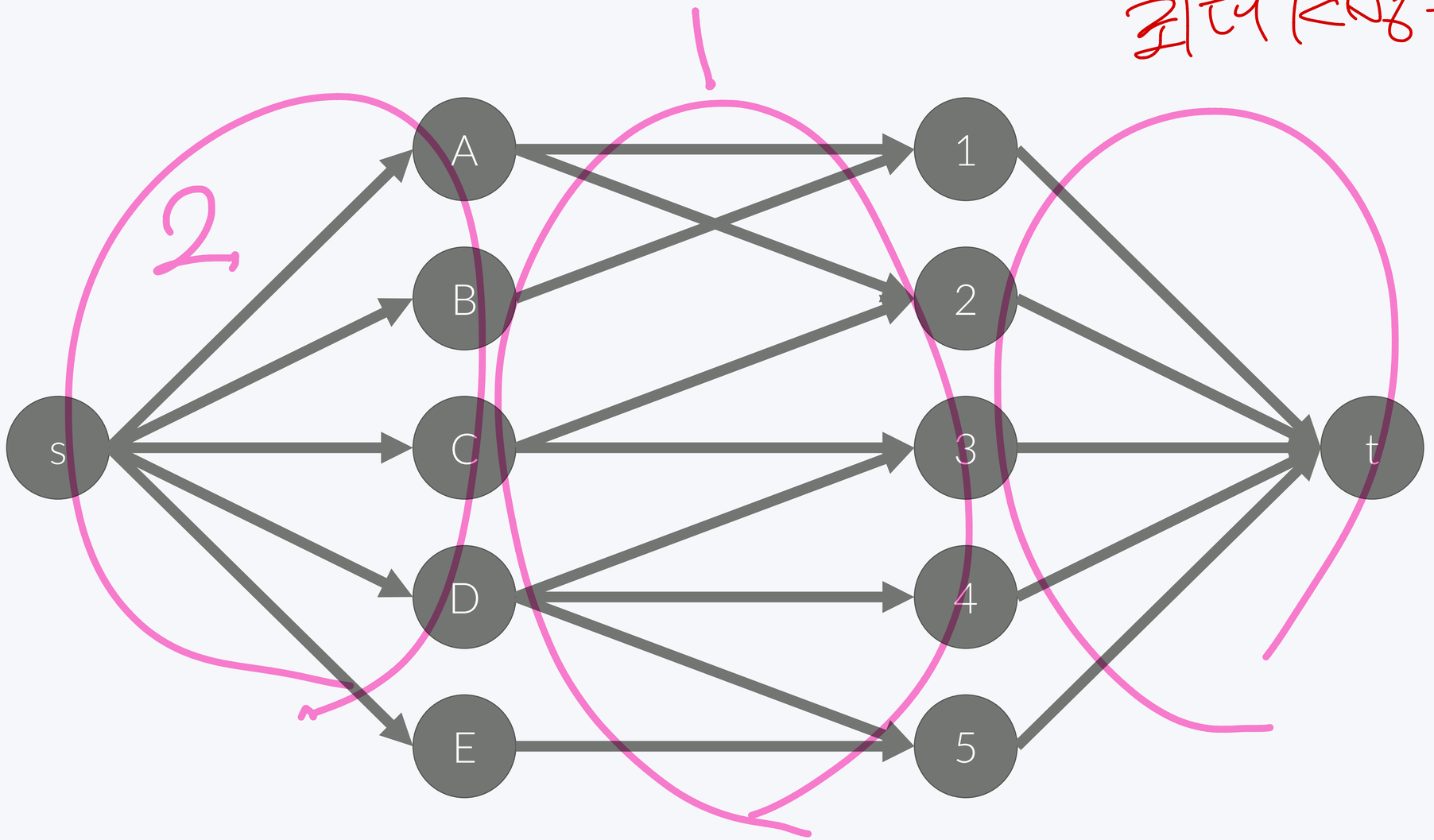
열혈강호

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

1명 최대 1개 역할

1역할 최대 1명

최대 k명은 2개



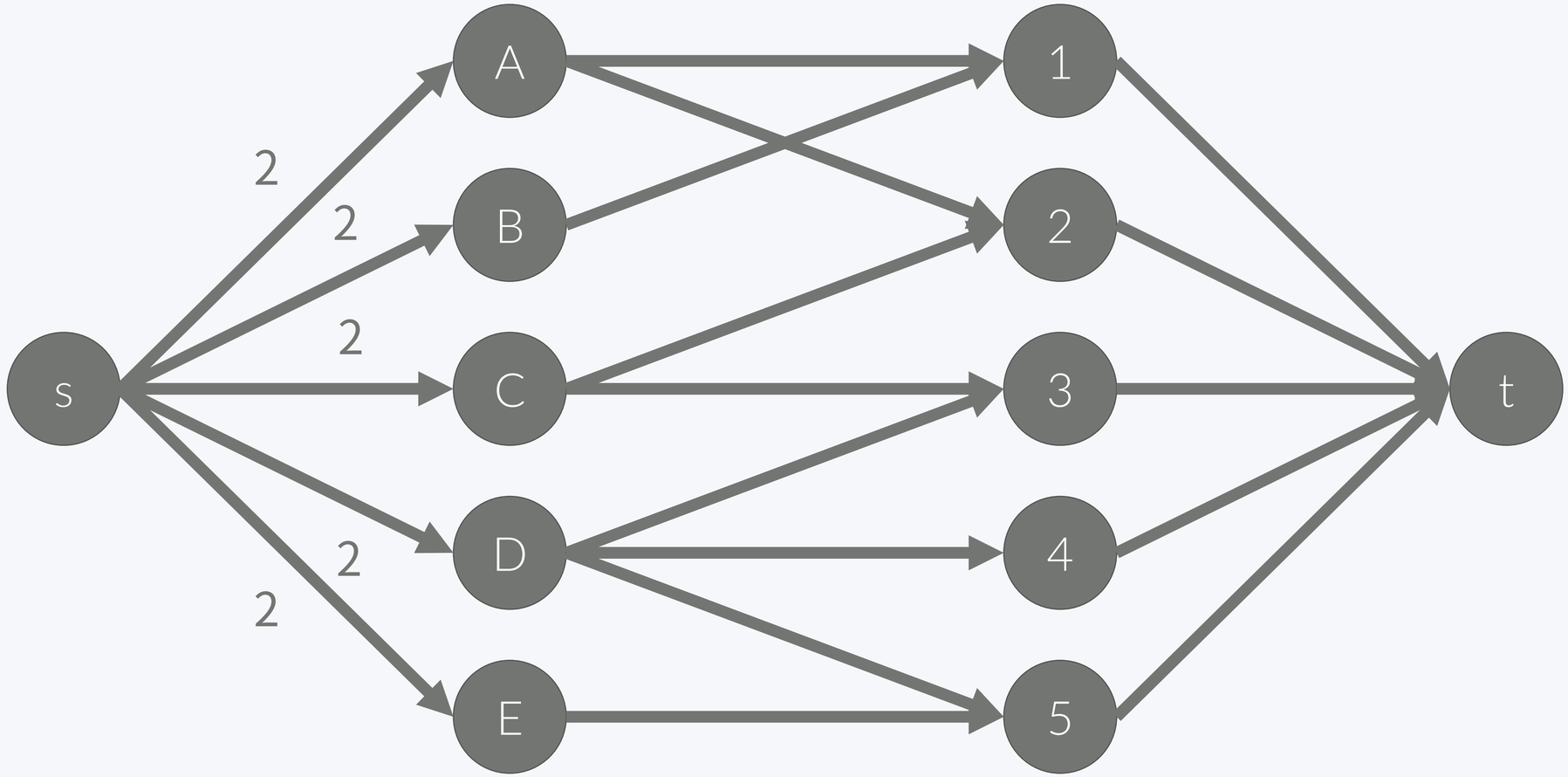
열혈강호

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

- 소스: http://boj.kr/d32c5e06808e4dc9b945c9dd73d8f63c

열혈강호 2

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



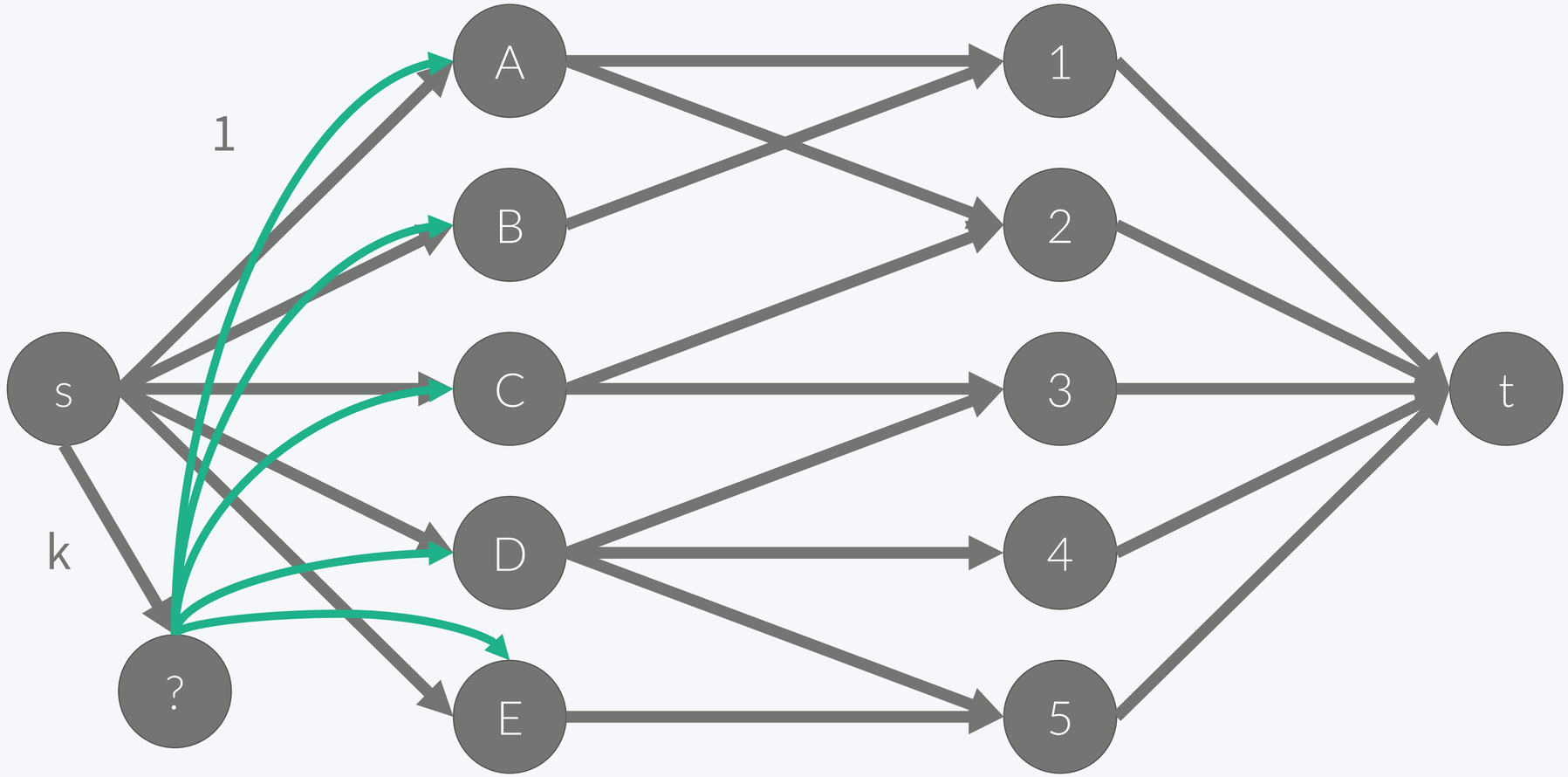
열혈강호 2

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

- 소스: http://boj.kr/ad3de93426eb4171b6b91bcf457c5564

열혈강호 3

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



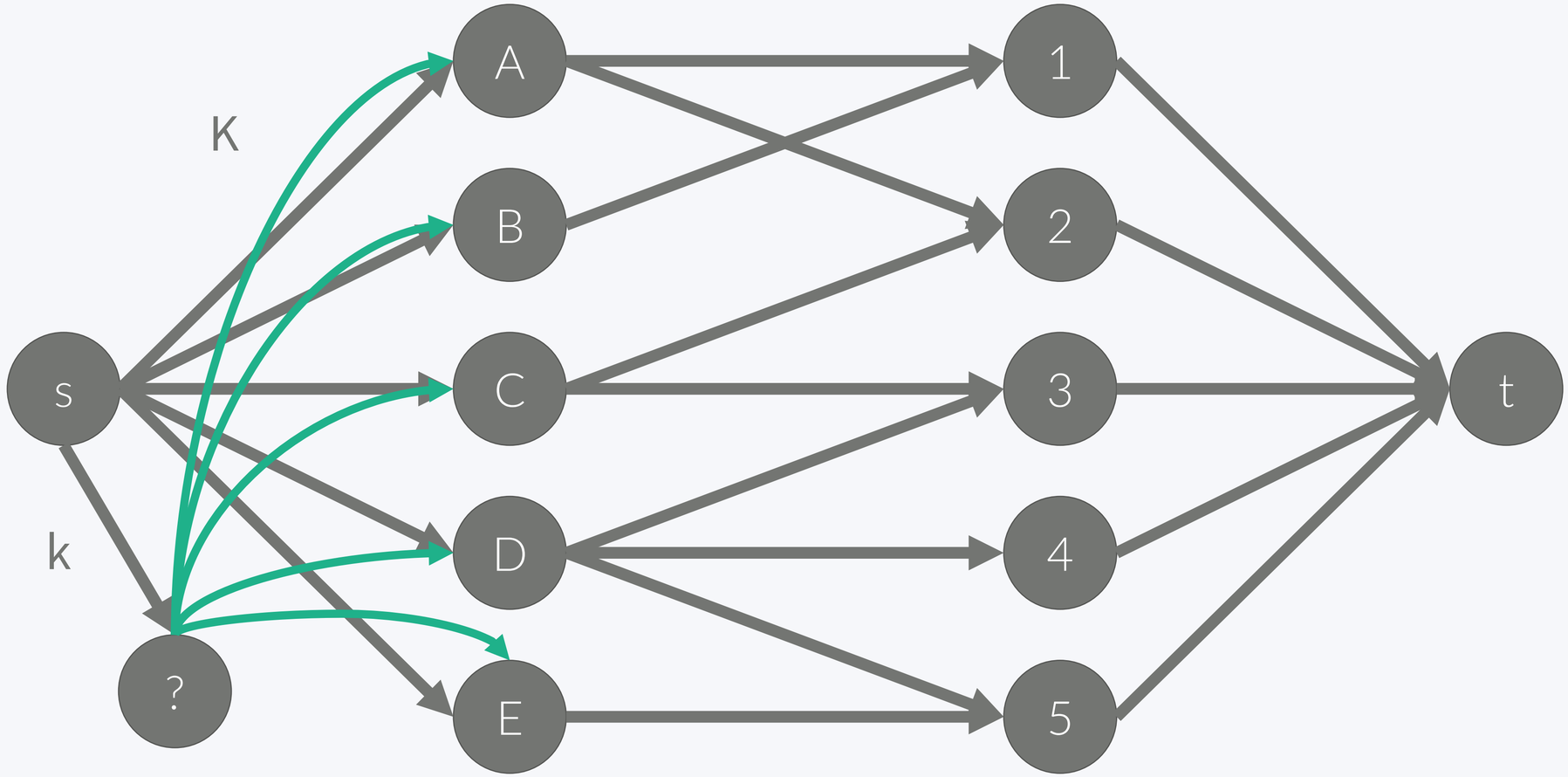
열혈강호 3

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

- 소스: http://boj.kr/e2207dd8b7ee47088394a3bbb466de94

열혈강호 4

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



열혈강호 4

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

- 소스: http://boj.kr/e68b0fd34c2149f18c6de2eb82947f6a

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

노트북의 주인을 찾아서

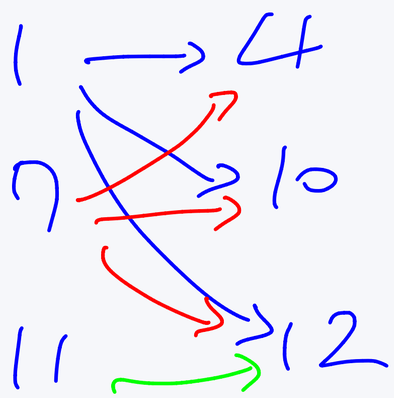
https://www.acmicpc.net/problem/1298

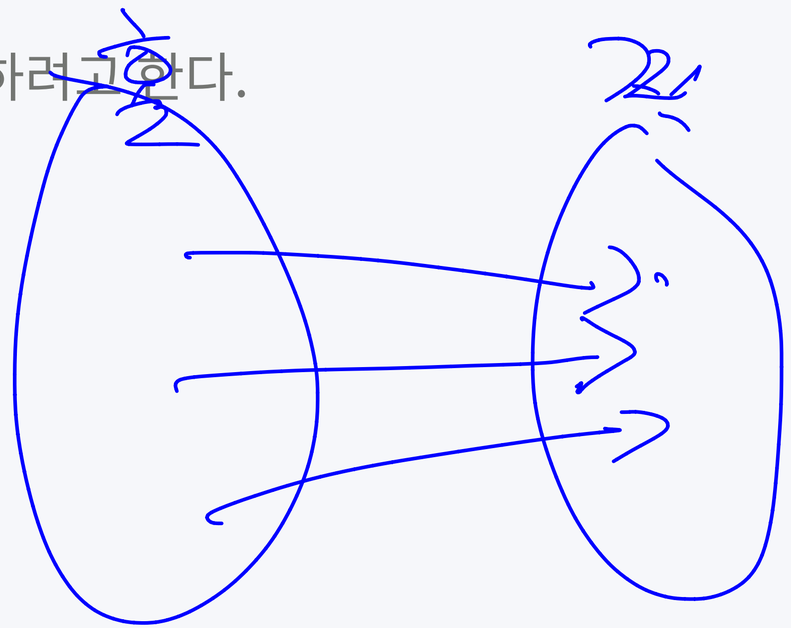
- 소스: http://boj.kr/6171641bbc794fdbbc6b9ce3b6052b1d

소수 쌍

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

- 수의 리스트가 있을 때, 이를 짝지어 각 쌍의 합이 소수가 되게 하려고한다.
- {1, 4, 7, 10, 11, 12}가 있으면
- 1 + 4 = 5, 7 + 10 = 17, 11 + 12 = 23
- 또는
- 1 + 10 = 11, 4 + 7 = 11, 11 + 12 = 23
- 가 가능하다
- 모든 수를 다 짝지었을 때, 첫번째 수와 어떤 수를 짝지었는지 오름차순으로 출력하는 문제





소수 쌍

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

- 첫 번째 수와 짝지을 수를 미리 구한 다음에
- 나머지를 매칭한다
- 왼쪽: 홀수 오른쪽: 짝수

소수 쌍

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

- 소스: http://boj.kr/a10d9141239b45228d379607f6f2f0db

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

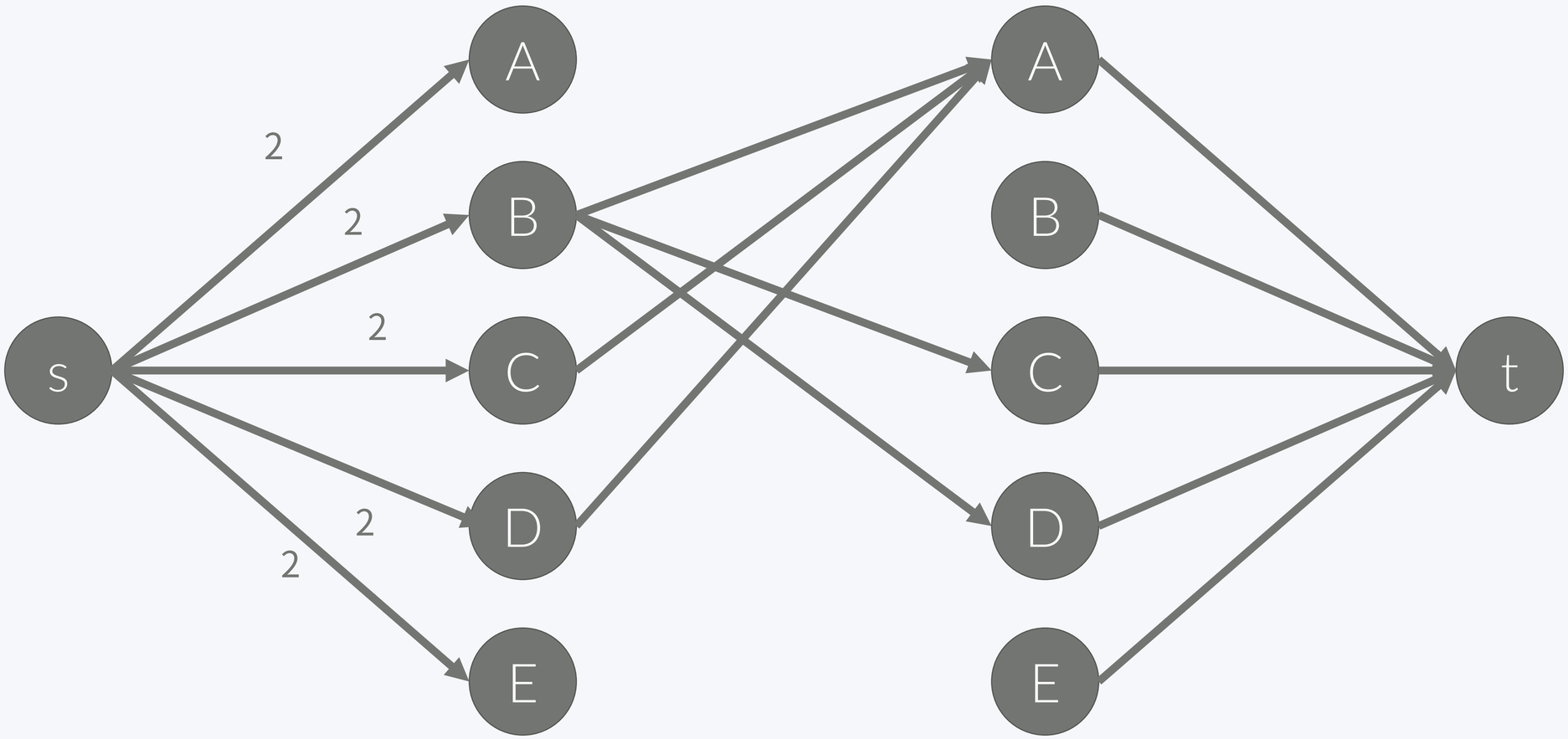
https://www.acmicpc.net/problem/1671

- 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



상어의 저녁식사

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

- 소스: http://boj.kr/662483d136944e7a8da60fdfc155422f

+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 matching의 개수가 차의 개수와 같은지 확인

주차장

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

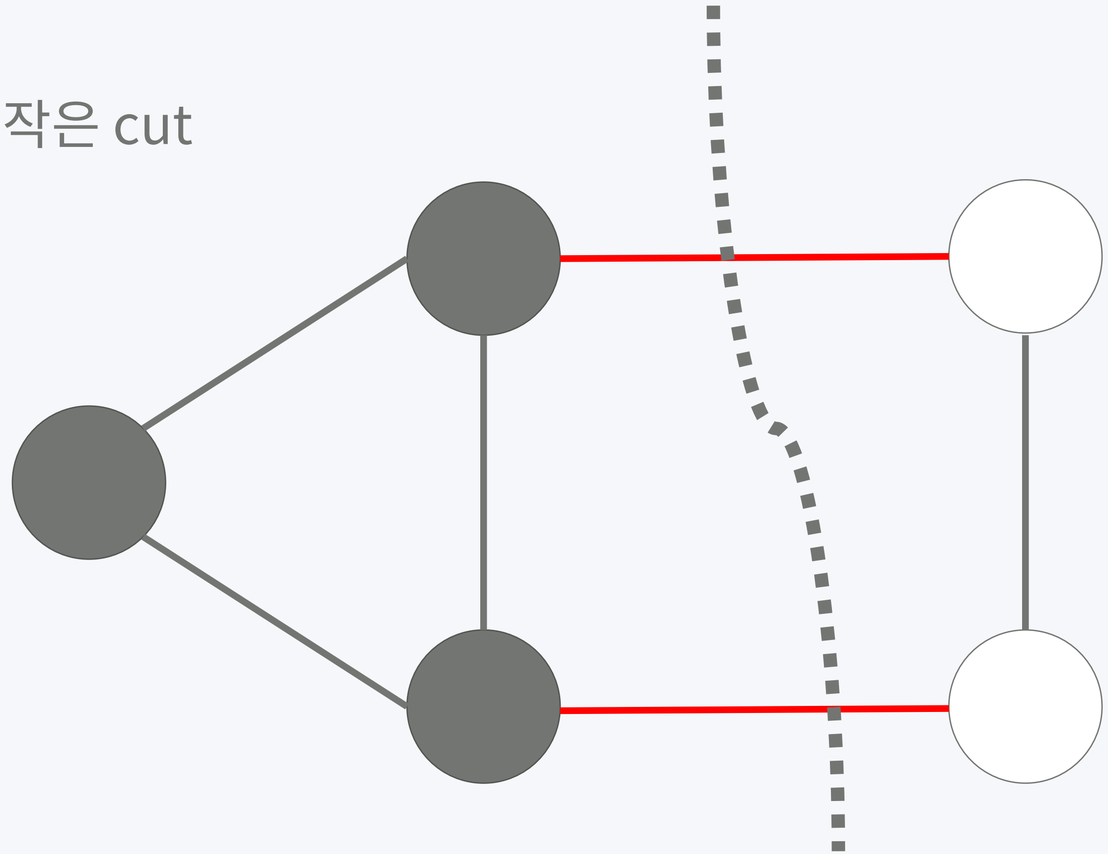
- 소스: http://boj.kr/ac45691da8b04eee97701ddb62895ce4

Min-cut

Minimum cut

Minimum Cut

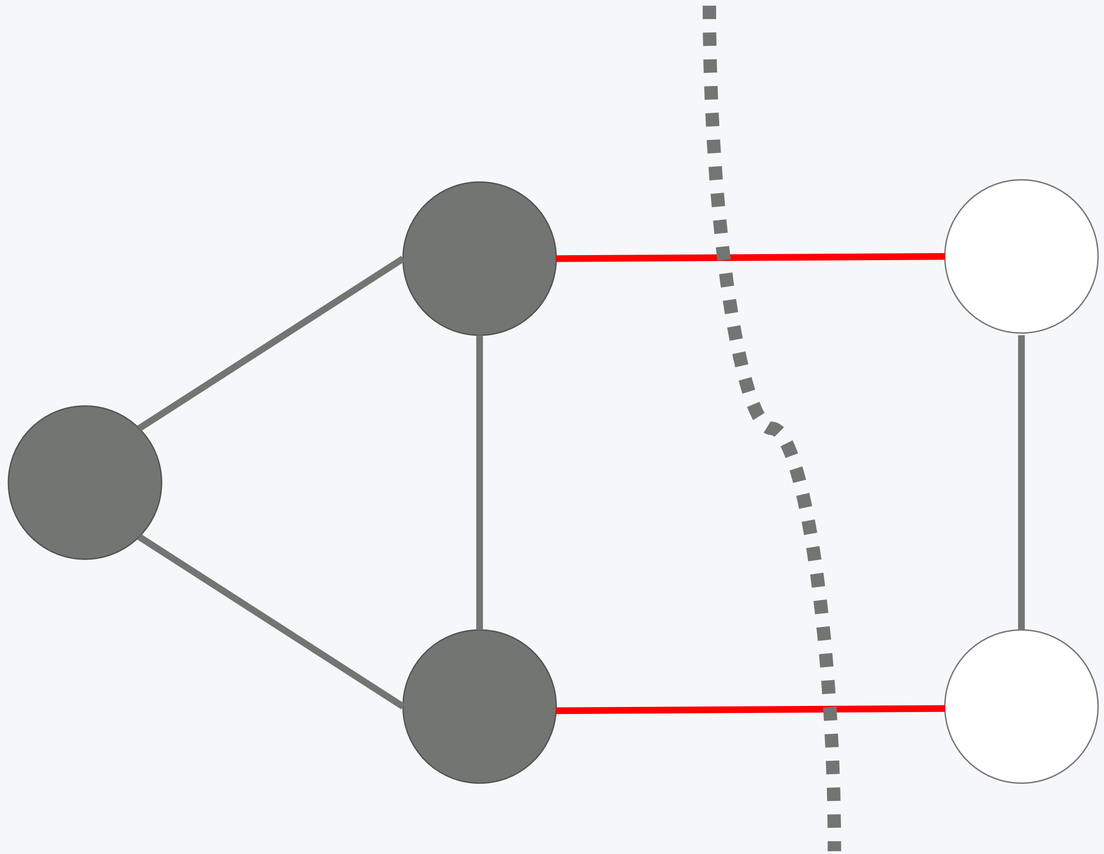
- Cut: 그래프를 2개의 서로다른 집합으로 나누는 것
- 두 집합을 A와 B라고 했을 때, 한 쪽 끝은 A에 다른 한 쪽 끝은 B에 있는 간선을 cut-set이라고 한다
- Minimum cut: 가장 작은 cut



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

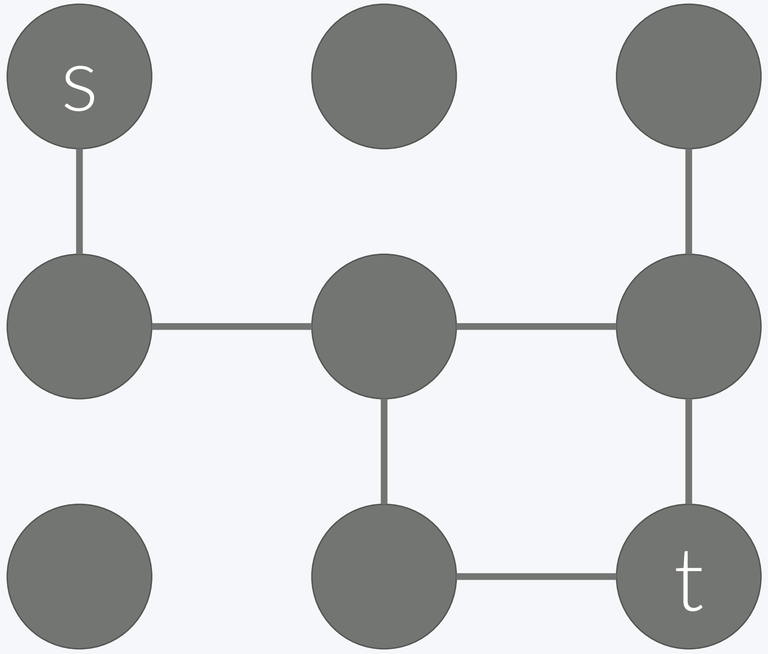
- 빈 칸을 적절히 벽으로 바꿔서 학교로 가지 못하게 하는 문제
- 최소 개수를 구해야 한다

학교 가지마!

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

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

K	#	
#		H

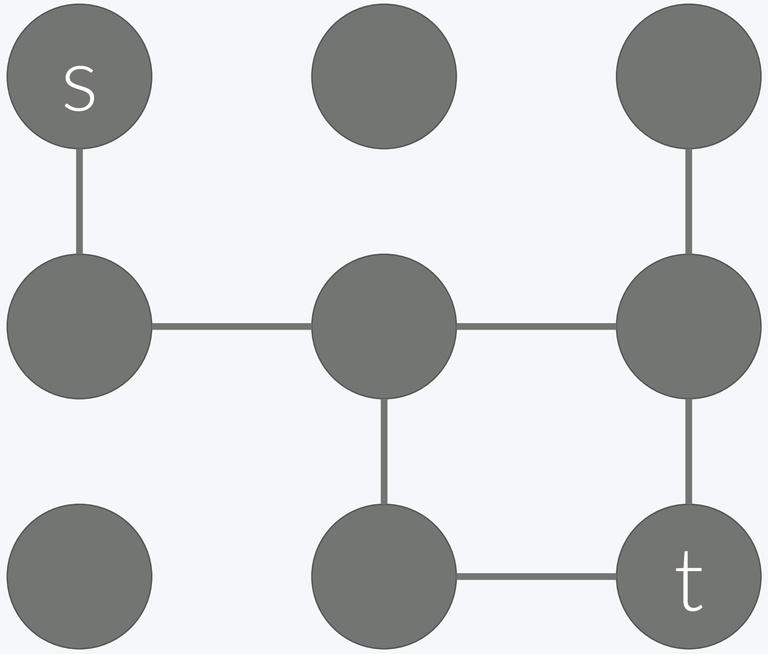


학교 가지마!

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

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

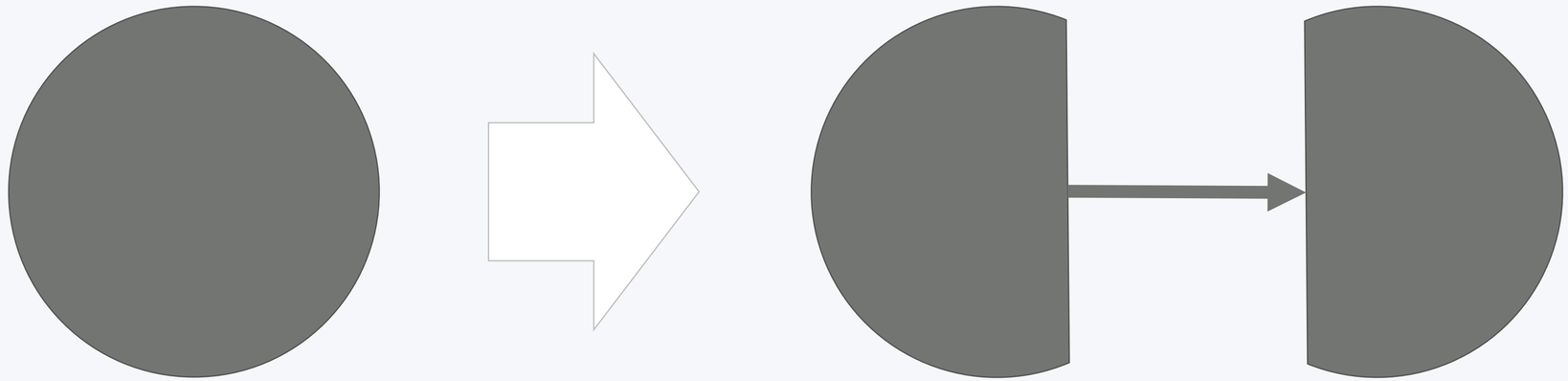
K	#	
#		H



학교 가지마!

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

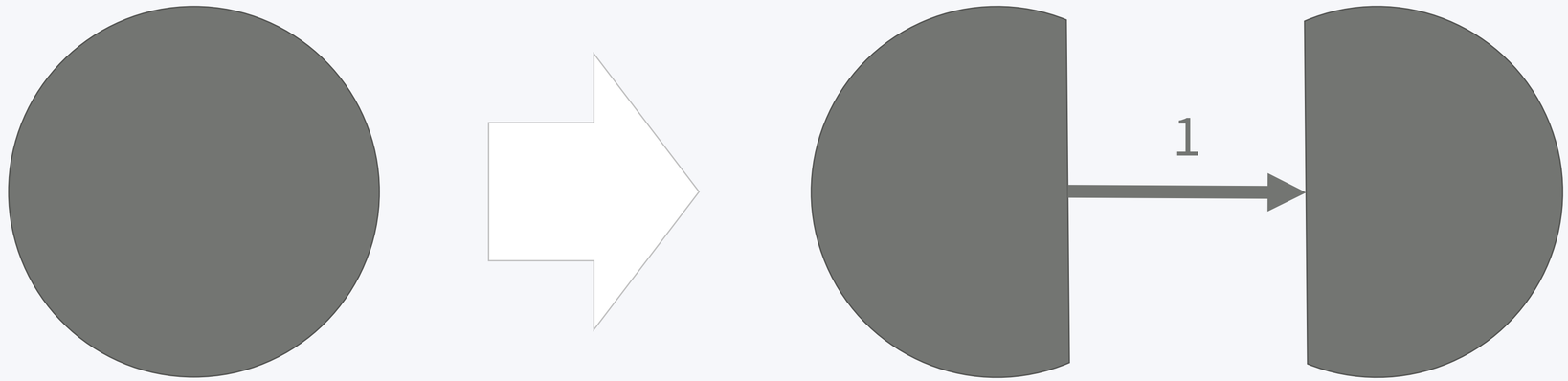
- 각 칸을 둘로 나눈다
- capacity는 몇?



학교 가지마!

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

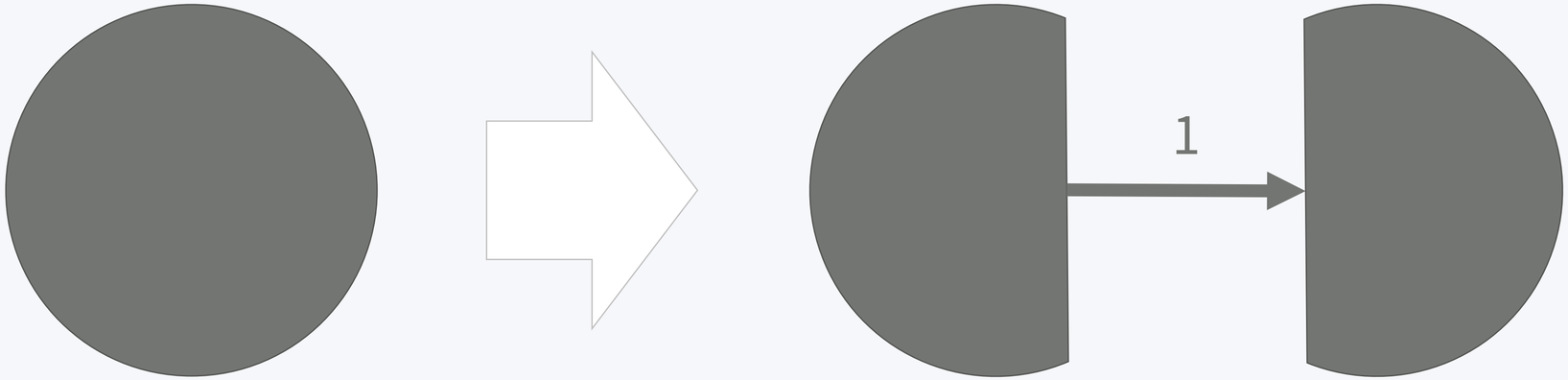
- 각 칸을 둘로 나눈다
- capacity는 몇? 1



학교 가지마!

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

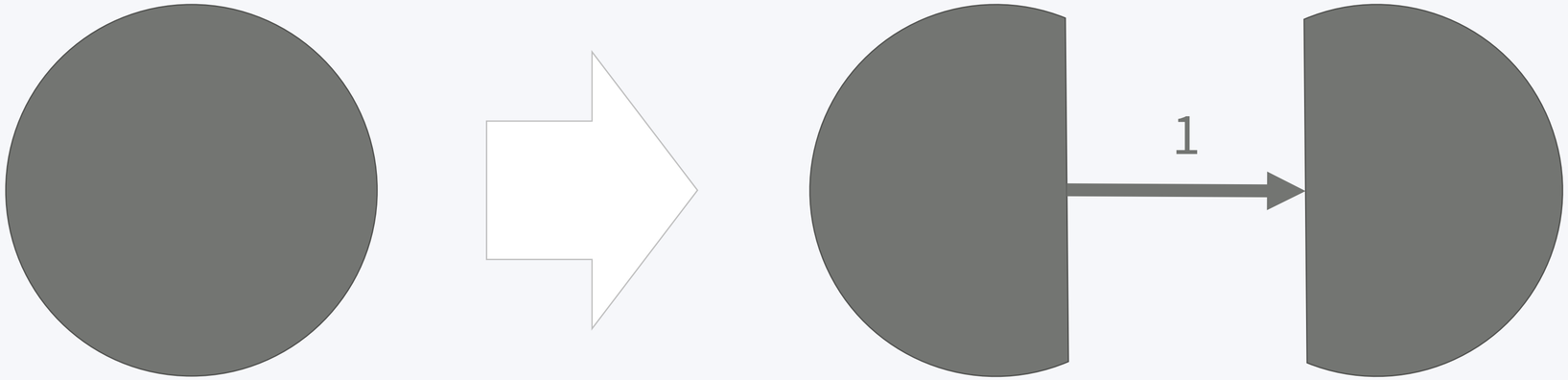
- 정점 X를 Xin과 Xout 으로 나누고, Xin -> Xout은 1로 연결
- X와 Y를 이동할 수 있으면, Xout -> Yin, Yout -> Xin 을 연결
- 이 때 capacity?



학교 가지마!

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

- 소스: http://boj.kr/a7784244ab2842c48eea50962da84b77



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

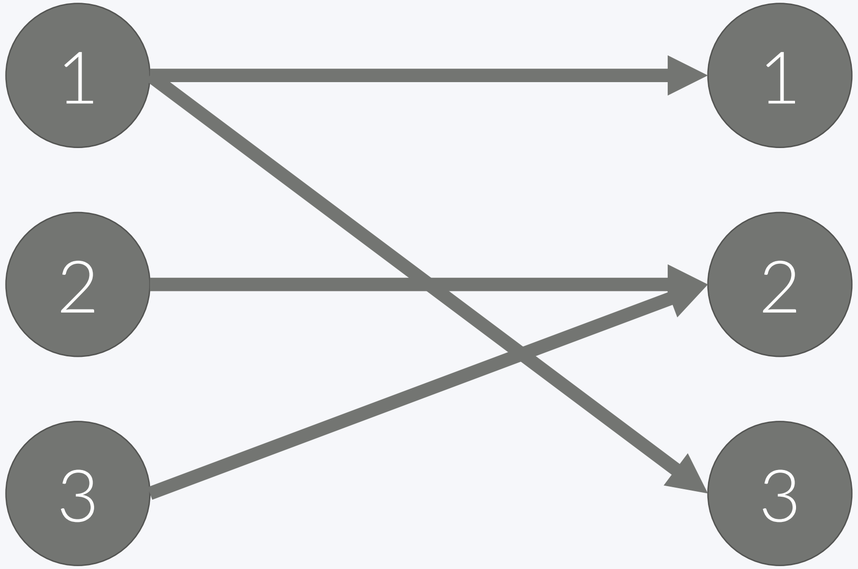
돌멩이 제거

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

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

돌멩이 제거

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



돌		돌
	돌	
	돌	

돌멩이 제거

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

- 소스: http://boj.kr/43cc7281a194462c86d22475a1197d8b

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

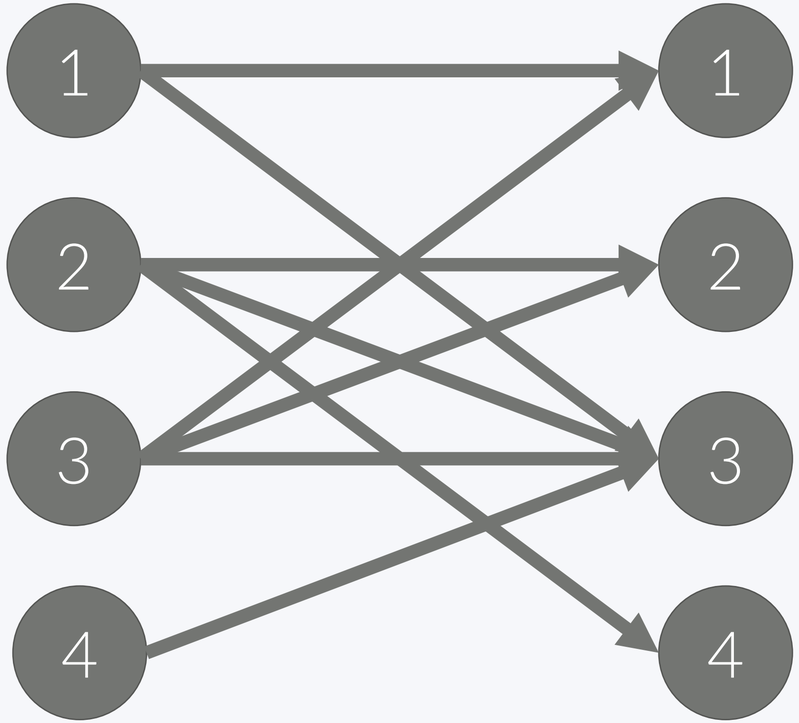
- 돌멩이 줍기와 똑같은 문제다
- 하지만 구멍만 막아야 한다.
- 따라서 그래프를 조금 수정해야 한다

게시판 구멍 막기

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

- 돌멩이 줍기와 똑같은 문제다. 이 그래프가 아니다.

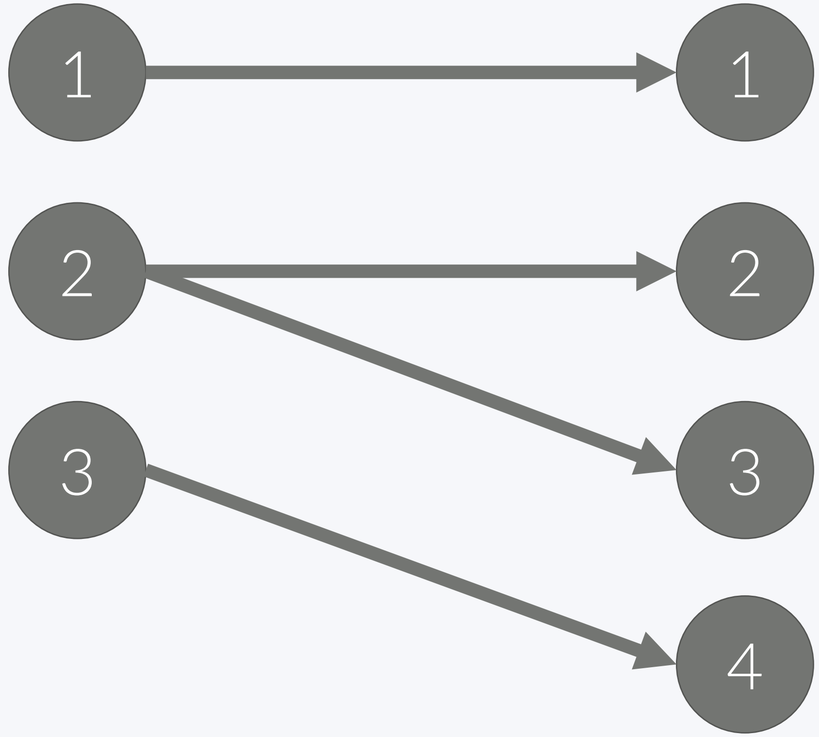
#		#	
	#	#	#
#	#	#	
		#	



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	

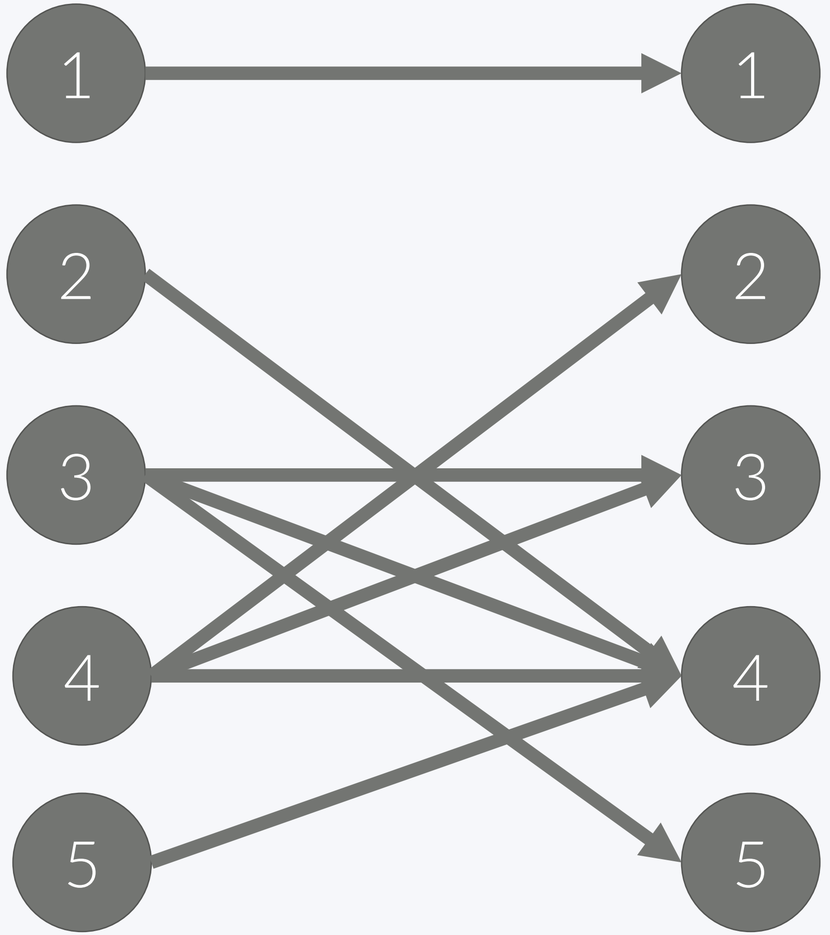
1		4	
	3	4	5
2	3	4	
		4	

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

- 소스: http://boj.kr/a0bf239cbb054bfeb39c71e864f9ee5f

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.

컨닝 2

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

- 소스: http://boj.kr/3f0b0dc279d84b1986ad69d52b764275