Dinic

CONTENTS

Ⅰ. Dinic

Ⅰ. Dinic

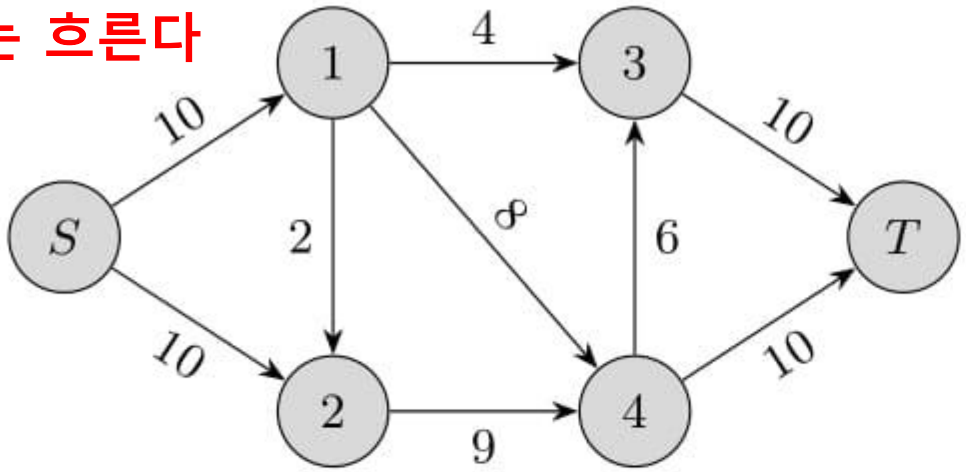
1. Network Flow

- **상당히 생소한 개념**이고, 겉으로 보기에 **전혀 상관이 없어 보이는 문제가 Network Flow를 사용하여 풀리는 경우**도 있다.
- 대회에 출제되면 대개 고난도 문제로 분류되며, 대부분 **그래프를 모델링 하는 과정이 문제를 해결하는 과정의 전부**이다. (요즘은 잘 안 나오는 것 같긴 하다.)
- 알아야할 사전지식이 상당히 많고 알고리즘 자체도 어렵지만, ICPC를 준비하는 사람 입장에서는 **다양한 문제를 많이 풀고, 주어진 문제를 Flow Network로 바꾸는 능력을 키우는 것이 효율적**이다.

Ⅰ. Dinic

1. Network Flow

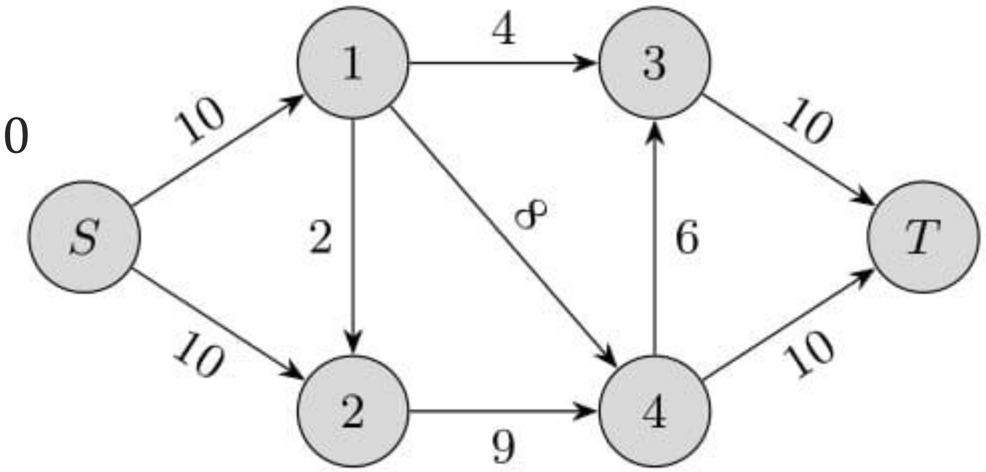
- 일단 **Flow Network**를 알아보자.
- Flow Network는 지금까지 본 일반적인 그래프들과 달리 **그래프의 가중치**가 갖는 의미가 **해당 간선에 흐를 수 있는 flow의 최댓값을 의미하는 capacity**이다.
- 또한 **들어오는 간선이 없는 Source(=S)**와 **나가는 간선이 없는 Sink(=T)**가 존재한다.
- **S에서 T로 흘릴 수 있는 최대 유량을 찾는 것**이 Network Flow 문제를 푸는 것이다.
- 단, 한 정점에서 다른 정점으로 **단방향으로만 flow는 흐른다**고 가정한다.
- 따라서 **양방향 간선은 2개의 단방향 간선으로 쪼갠 뒤 Flow Network**를 만들면 된다.



Ⅰ. Dinic

1. Network Flow

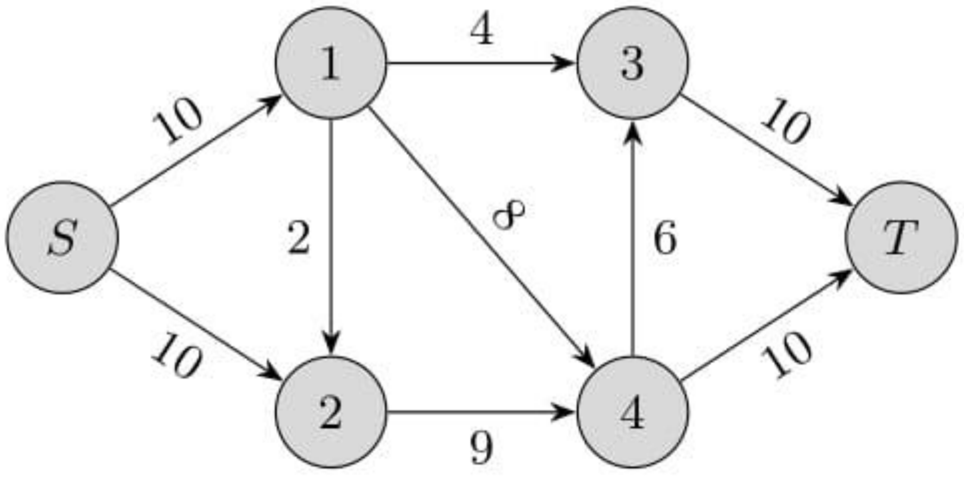
- **정점 u에서 정점 v로의 간선 (u, v)**에 대하여 다음을 정의하자.
- **$c(u, v)$ = 간선 (u, v)의 capacity**, 즉 flow의 상한선
- **$f(u, v)$ = 간선 (u, v)에 현재 흐르고 있는 flow**
- 이때 다음과 같은 **3가지 성질**이 성립한다.
- Capacity constraint: $\forall u, v \in V,\ f(u, v) \leq c(u, v)$
- Skew symmetry: $\forall u, v \in V,\ f(u, v) = -f(v, u)$
- Flow conservation: $\forall u \in V - \{S, T\},\ \sum_{v \in V} f(u, v) = 0$



Ⅰ. Dinic

1. Network Flow

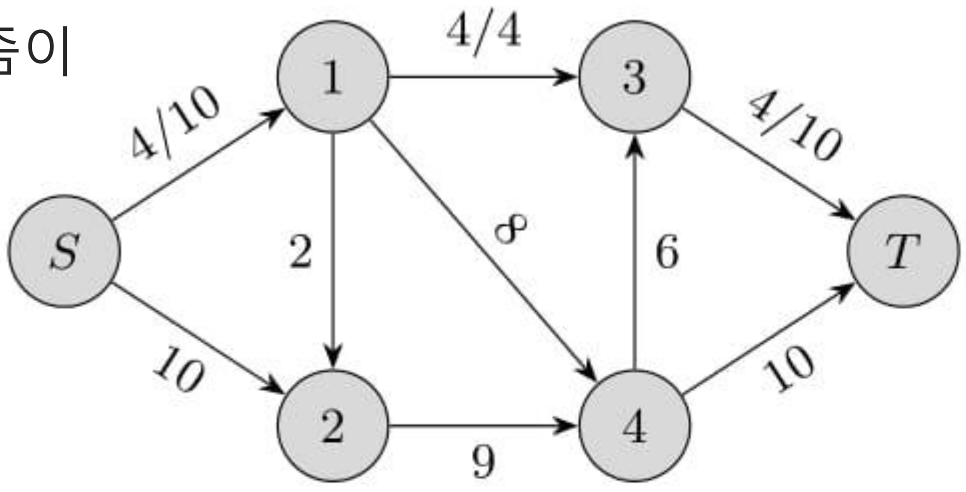
- 하나씩 알아보도록 하자.
- **Capacity constraint:** $\forall u, v \in V,\ f(u,v) \leq c(u,v)$
- 앞서 말한 capacity의 의미를 떠올리면 자명하다.
- **Capacity는 해당 간선에서 흐를 수 있는 flow의 상한선이기 때문**이다.



Ⅰ. Dinic

1. Network Flow

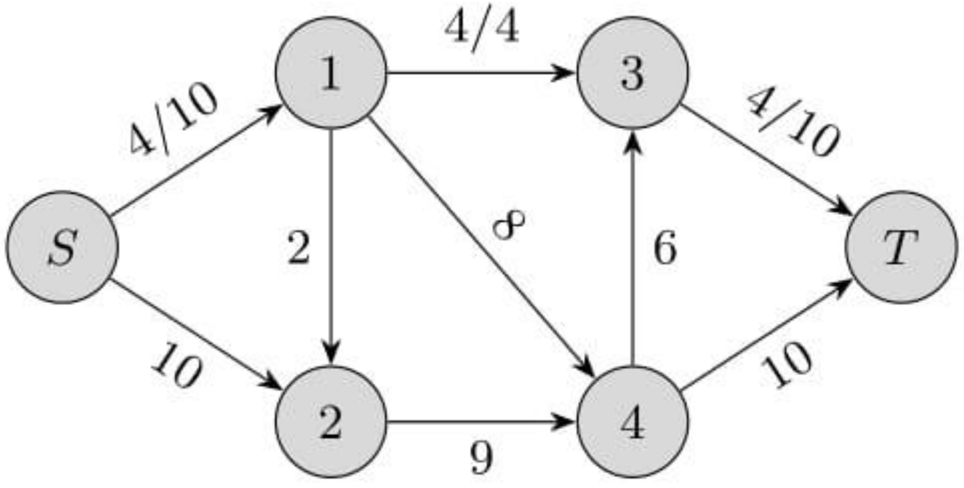
- 하나씩 알아보도록 하자.
- **Skew symmetry:** $\forall u, v \in V,\ f(u,v) = -f(v,u)$
- 앞서 언급한 Capacity constraint 성질보다는 자명하지 않다.
- 의미를 해석하자면, **간선 (u,v)로 $f(u,v)$만큼 flow가 흐르면, 역으로 간선 (v,u)로 가상의 $-f(u,v)$만큼의 flow가 흐른다**는 뜻이다.
- 잘 와 닿지 않지만 대부분의 Network Flow 알고리즘이 이 사실을 유용하게 사용한다.
- 오른쪽 경우 $f(3,1) = -f(1,3) = -4$이다.



Ⅰ. Dinic

1. Network Flow

- 하나씩 알아보도록 하자.
- **Flow conservation:** $\forall \boldsymbol{u} \in \boldsymbol{V} - \{\boldsymbol{S}, \boldsymbol{T}\},\ \sum_{\boldsymbol{v} \in \boldsymbol{V}} \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{v}) = \mathbf{0}$
- 이는 **Source와 Sink를 제외한 모든 정점**에 대하여 **자신에게 들어온 유량과 나가는 유량은 같다**는 의미이다. 다음 예시를 살펴보자.
- $\sum_{v \in V} f(3, v) = f(3, S) + f(3,1) + f(3,2) + f(3,4) + f(3, T)$
  $= 0 + (-4) + 0 + 0 + 4 = 0$



Ⅰ. Dinic

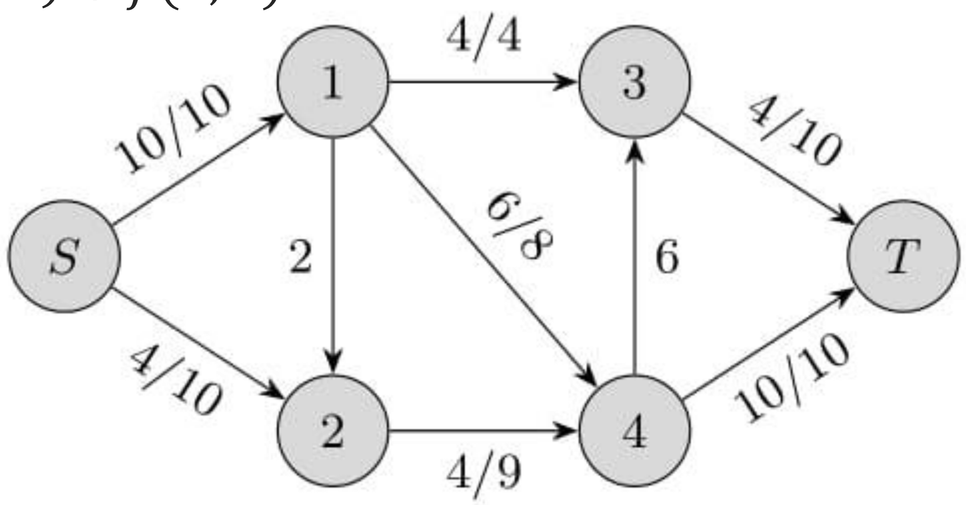
1. Network Flow

- 추가로 현재 **Flow Network에서 흐르는 Flow값**을 다음과 같이 정의하도록 하자.
- $|\boldsymbol{f}| = \sum_{\boldsymbol{v} \in \boldsymbol{V}} \boldsymbol{f}(\boldsymbol{S}, \boldsymbol{v}) = \sum_{\boldsymbol{v} \in \boldsymbol{V}} \boldsymbol{f}(\boldsymbol{v}, \boldsymbol{T})$
- 이는 **Source에서 나가는 Flow값과, Sink로 들어오는 Flow값이 같음**을 의미한다.
- 아래 예시의 경우 위 사실이 성립함을 알 수 있다.
- $|f| = \sum_{v \in V} f(S, v) = f(S,1) + f(S,2) + f(S,3) + f(S,4) + f(S,T)$

  $= 10 + 4 + 0 + 0 + 0 = 14$
- $|f| = \sum_{v \in V} f(v, T)$

  $= f(S,T) + f(1,T) + f(2,T) + f(3,T) + f(4,T)$

  $= 0 + 0 + 0 + 4 + 10 = 14$



Ⅰ. Dinic

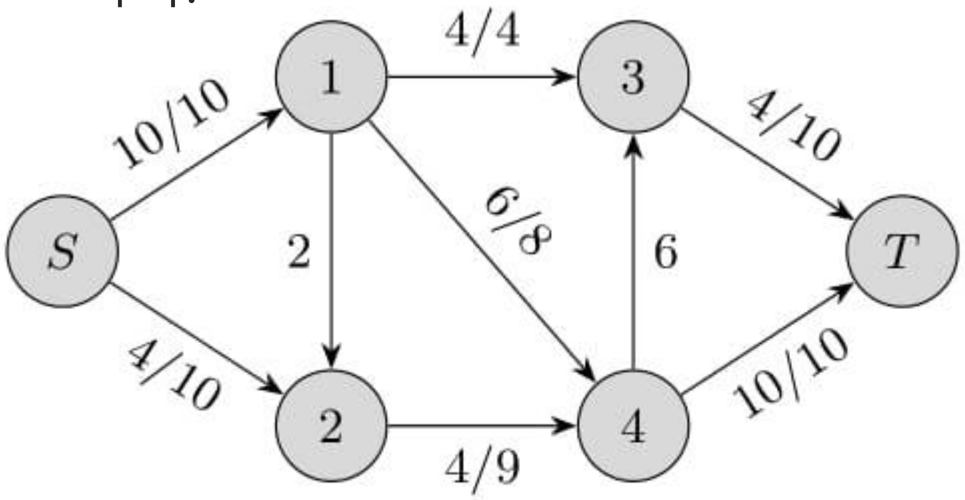
1. Network Flow

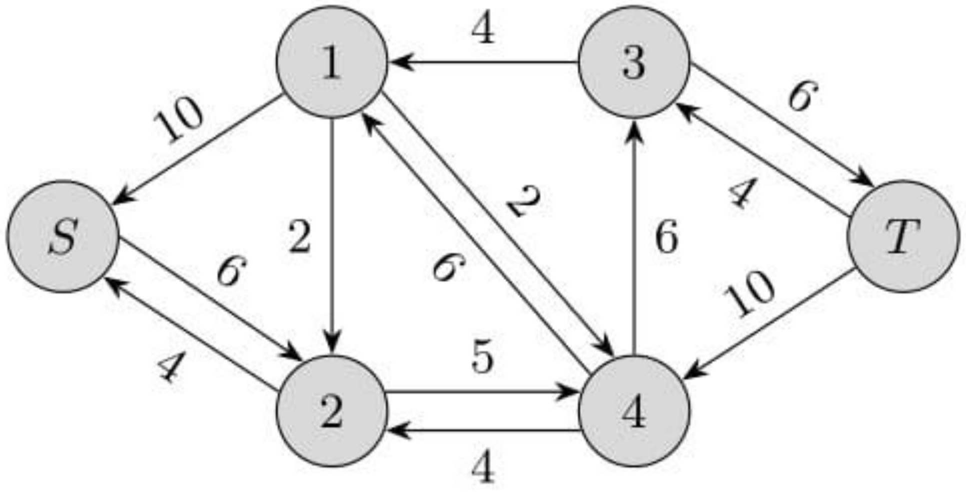
- Flow Network로부터 **Residual Network**라는 새로운 그래프가 만들어 지는데 이는 다음과 같다.
- 우선, **정점 u에서 정점 v로의 간선 (u, v)**에 대하여 다음을 정의하자.
- $c_f(u, v) = c(u, v) - f(u, v)$ **간선 (u, v)의 residual capacity**
- 즉, 추가로 흐를 수 있는 flow의 양을 의미한다.
- Residual Network는 **residual capacity가 0보다 큰 간선만을 나타낸 그래프**이다.

Ⅰ. Dinic

1. Network Flow

- $c_f(u,v) = c(u,v) - f(u,v)$ **간선 (u,v)의 residual capacity**
- Residual Network는 **residual capacity가 0보다 큰 간선만을 나타낸 그래프**이다.
- 아래의 경우 다음과 같은 residual network가 만들어진다.
- 앞서 말한 **skew symmetry에 의해서 생긴 간선(=역방향 간선)**들을 유심히 보도록 하자.

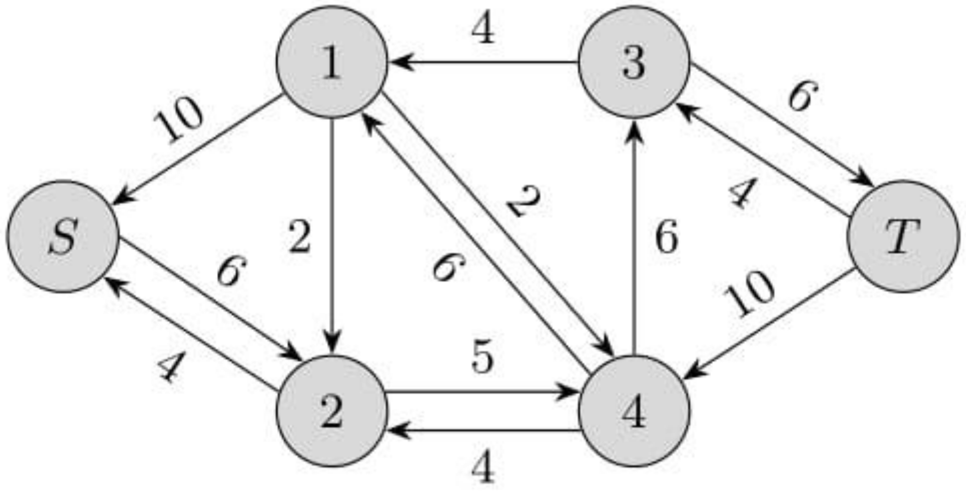


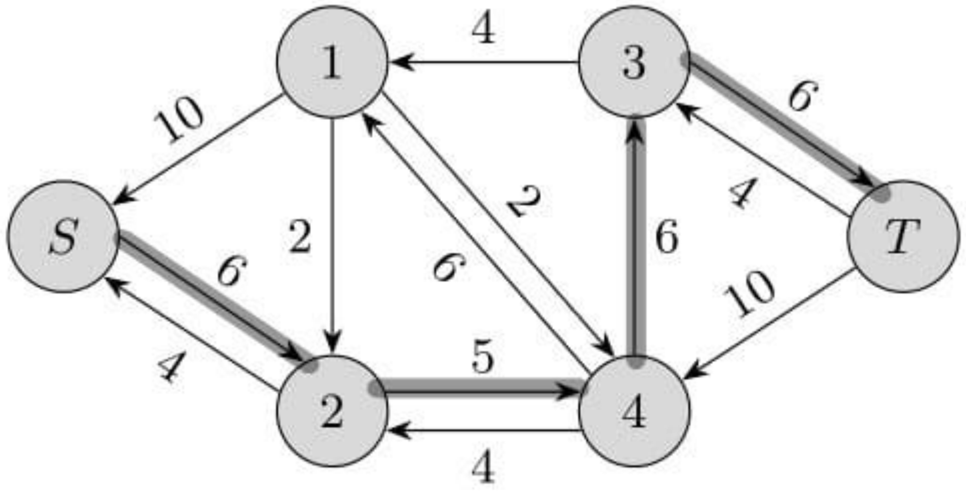


Ⅰ. Dinic

1. Network Flow

- 이러한 Residual Network에서 **Augmenting path**를 정의하도록 하자.
- Augmenting path는 **residual network에서 Source(=S)로부터 Sink(=T)까지의 simple path**를 의미한다.
- 아래와 같은 경우 회색으로 칠한 경로가 Augmenting path이다.

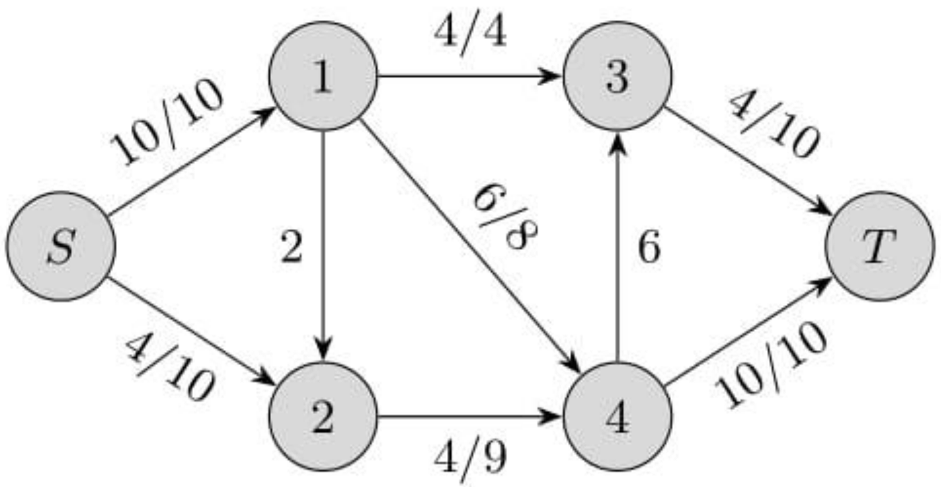


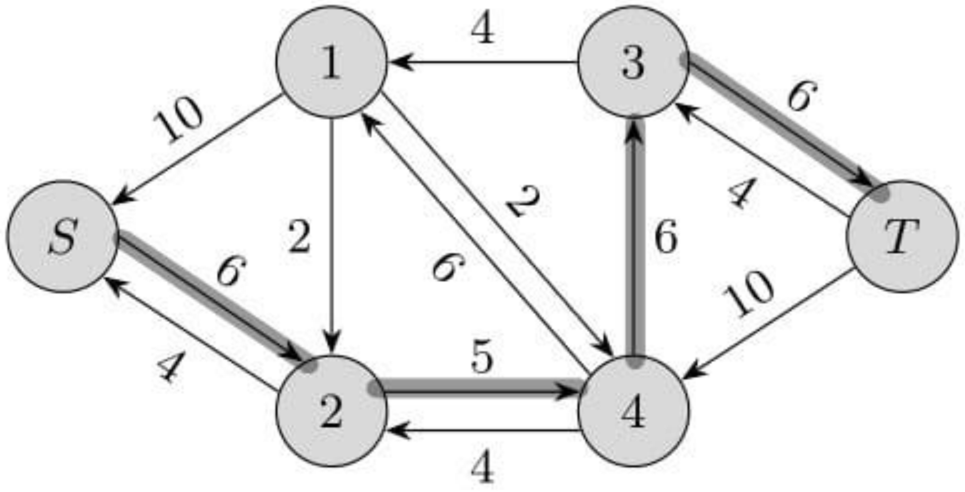


Ⅰ. Dinic

1. Network Flow

- **Augmenting path가 갖는 의미**는 해당 **경로상의 가중치 값 중 최솟값만큼** 현재상태의 Flow Network에 **Flow를 더 흘려 보낼 수 있다**는 것이다.
- 즉, 아래의 예시에서는 5만큼의 flow를 추가로 흘릴 수 있다.

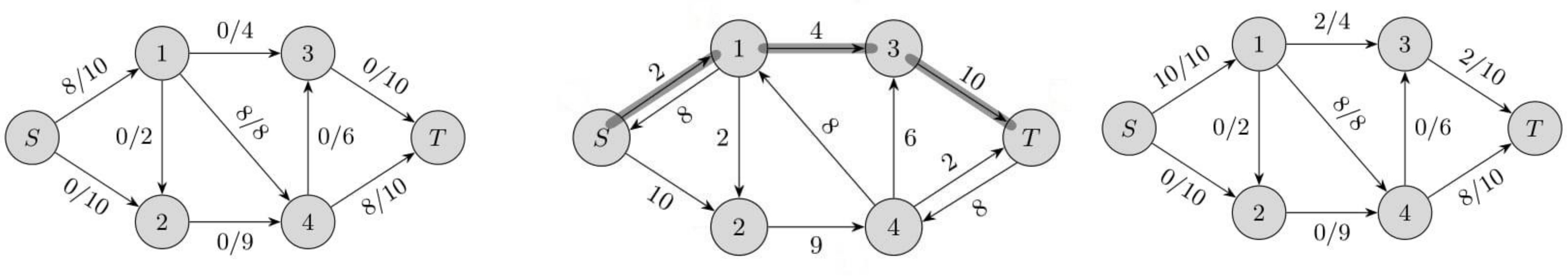




I . Dinic

1. Network Flow

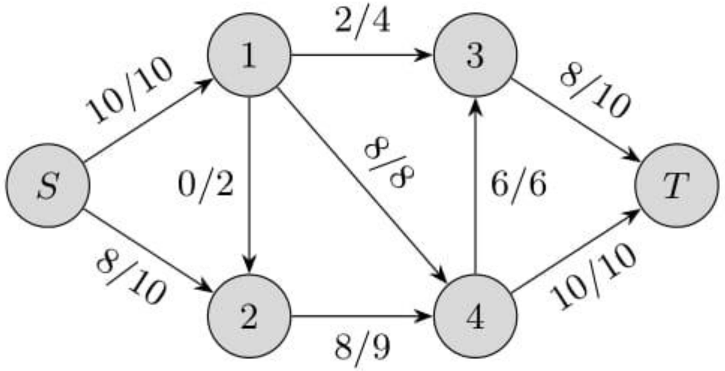
- 그렇다면 Augmenting path가 **skew symmetry에 의해 생긴 간선을 포함할 때와 안 할 때의 차이**를 살펴보자.
- 먼저 **skew symmetry에 의해 생긴 간선을 포함하지 않는 경우**이다.
- 보다시피 **S -> 1 -> 3 -> T로 2만큼 flow가 늘어난 것**을 볼 수 있다.

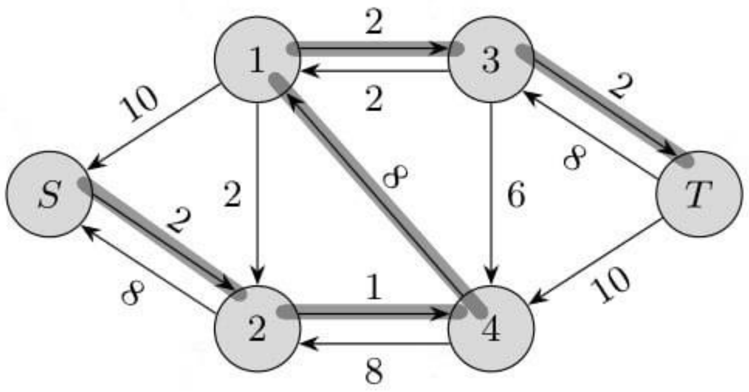


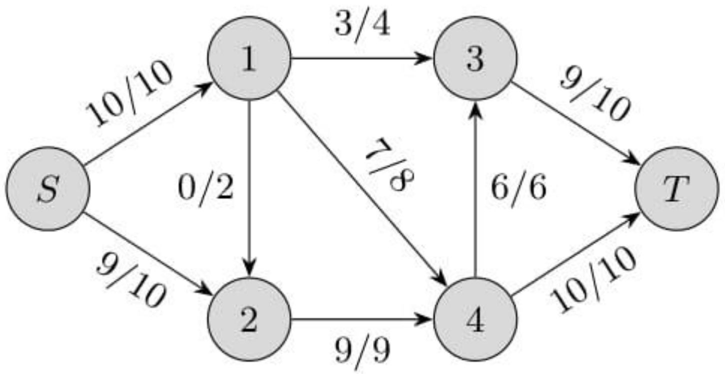
Ⅰ. Dinic

1. Network Flow

- 다음은 **skew symmetry에 의해 생긴 간선을 포함하는 경우**이다.
- 보다시피 **S -> 2 -> 4 와 1 -> 3 -> T로 1만큼 flow가 늘어난 것**을 볼 수 있다.
- 반면, **skew symmetry에 의해 생긴 간선은 1만큼 flow가 줄어든 것**을 볼 수 있다.
- 즉, 최종 결론을 보면 **skew symmetry에 의해 생긴 간선을 통과하는 경우, flow가 흐르는 방향을 조절하여 더 많은 양의 flow를 흘린 것으로 해석**할 수 있다.







Ⅰ. Dinic

1. Network Flow

- 지금까지 정말 많고 생소한 개념들이 나왔다. Dinic을 배우기 전에 아래 내용들을 다시 한번 정리하고 넘어가기를 바란다.
- Flow Network, Capacity, Capacity constraint, Skew symmetry, Flow conservation, Residual capacity, Residual Network, Augmenting path

Ⅰ. Dinic

2. 개념

- **Dinic**은 앞서 나온 Flow Network에 대하여 흘릴 수 있는 **maximum flow를 찾아 주는 알고리즘**이다.
- 일반적으로 maximum flow 알고리즘은 다음과 같은 **공통된 성질**이 있다.

1. 현재의 상태로부터 **Residual Network를 만든다.**
2. **Augmenting path가 존재하면 해당 경로를 통해 flow를 더 흘린다.**
3. **Augmenting path가 존재하지 않을 때까지, 1, 2를 반복**한다.

- 즉, **flow를 흘릴 수 있는 경우의 수가 존재하면 계속 흘려보는 방식**을 취하는 것이다.

Ⅰ. Dinic

2. 개념

- 즉, **flow를 흘릴 수 있는 경우의 수가 존재하면 계속 흘려보는 방식**을 취하는 것이다.
- 하지만, 이 과정을 **어떻게 수행하는가에 따라서 알고리즘의 시간 복잡도는 천차만별**이 된다.
- 오늘 다룰 **Dinic**은 현 시점에서 Problem Solving 대회에 출제 가능한 가장 빠른 maximum flow 알고리즘이고, $O(|V|^2|E|)$의 시간 복잡도를 갖는다.

- 다른 더 빠른 알고리즘들이 존재하지만 해당 알고리즘의 구현이 너무 복잡하여 아직 PS 대회에 출제되기에는 무리라고 한다.

Ⅰ. Dinic

2. 개념

- Dinic 알고리즘의 수행 과정을 알기 전에 앞서 말한 Network Flow의 기초 내용 외에 몇가지를 더 정의해야 한다.
- **Level Graph**는 **Residual Network에서 Source로부터 각 정점까지의 최단거리 (=Level)를 의미**하며, **Source에서 BFS를 수행**하여 만든다.
- **Blocking Flow**는 **Augmenting path**의 일종으로, **한 정점 u에서 다른 정점 v로 이동할 때 반드시 $level(v) = level(u) + 1$이 성립하는 경로**를 말한다.

Ⅰ. Dinic

2. 개념

- Dinic 알고리즘은 아래와 같이 수행된다.

1. **Level Graph를 만든다.** 이때, Source에서 **Sink에 도달할 수 없으면 종료**된다.
2. Level Graph에서 **모든 Blocking Flow를 찾아 그 유량만큼 총 유량에 더하고 1.로 돌아간다. (residual capacity는 바뀌지만, level은 그대로이다.)**

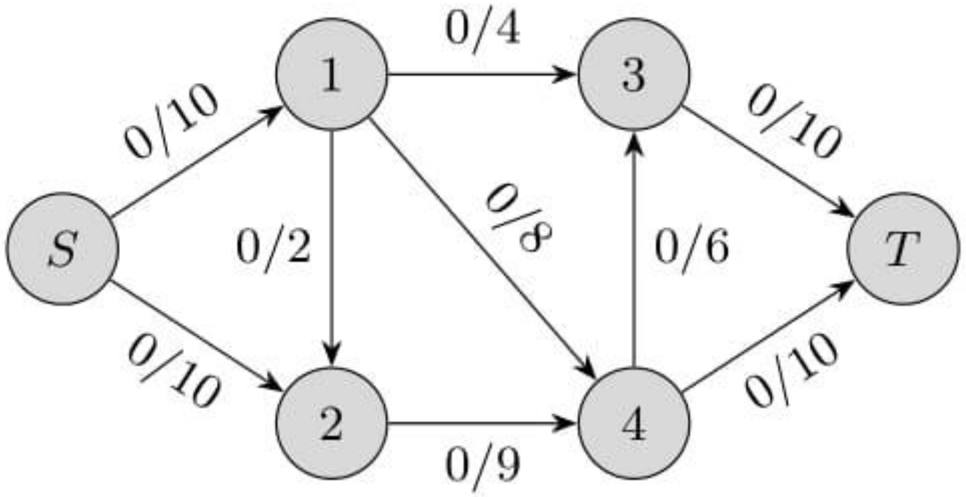
- 말은 정말 간단하다. 예시를 살펴보도록 하자.

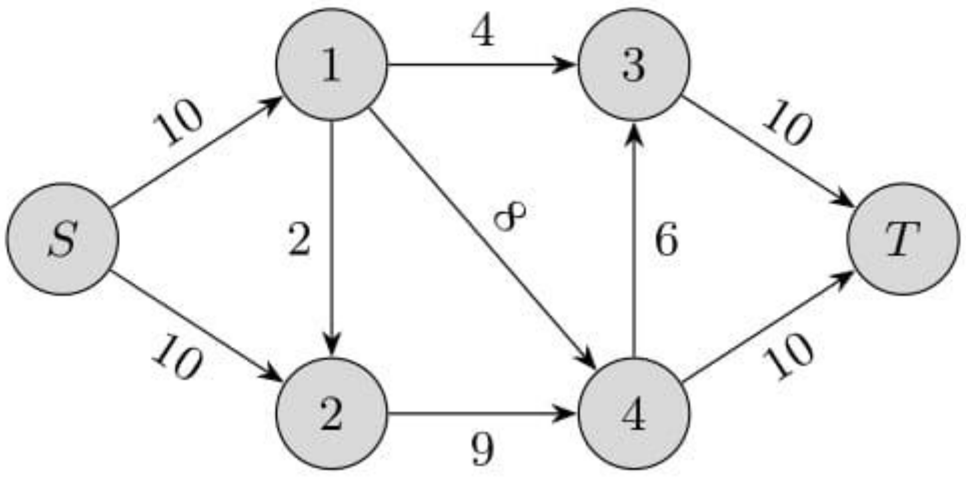
Ⅰ. Dinic

2. 개념

- 아래와 같은 Flow Network가 주어졌다면, Residual Network는 오른쪽과 같다.

1. Level Graph를 만든다. 이때, Source에서 Sink에 도달할 수 없으면 종료된다.
2. Level Graph에서 모든 Blocking Flow를 찾아 그 유량만큼 총 유량에 더하고 1.로 돌아간다. (residual capacity는 바뀌지만, level은 그대로이다.)



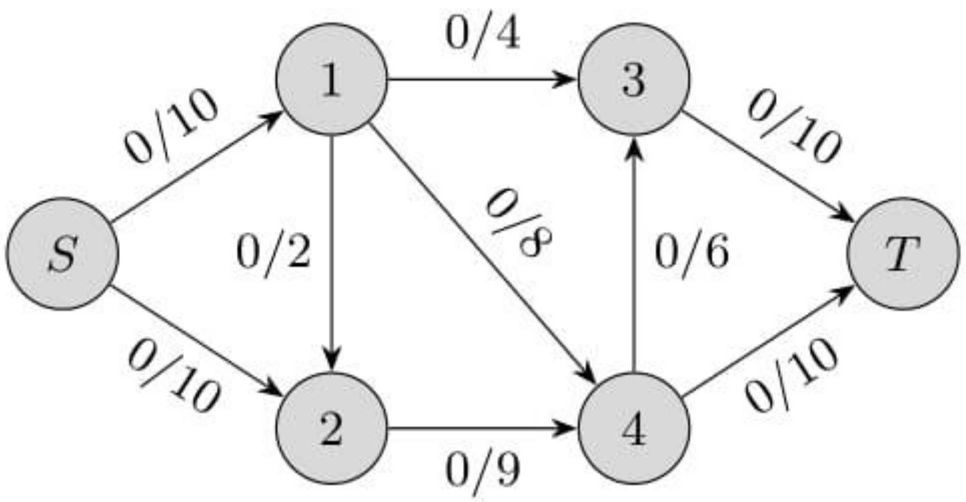


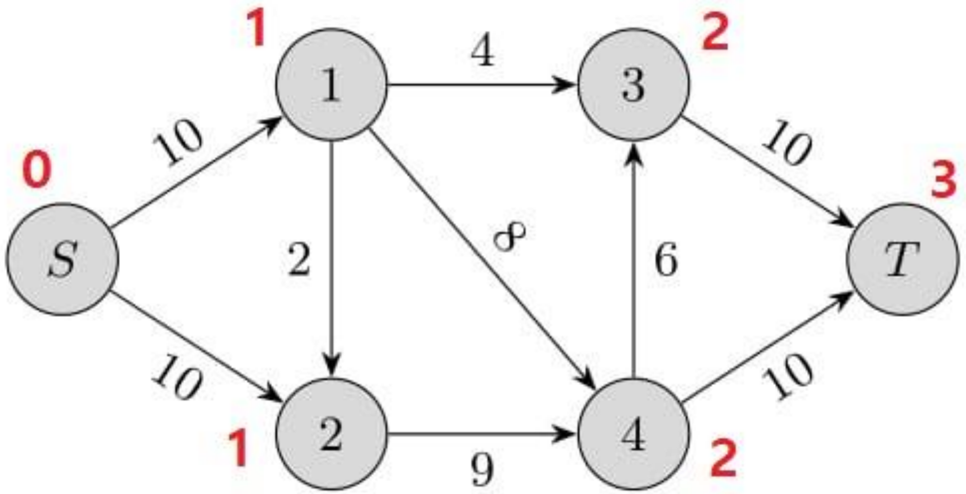
Ⅰ. Dinic

2. 개념 – 1번째 iteration

- 아래와 같은 Flow Network가 주어졌다면, Residual Network는 오른쪽과 같다.

1. **Level Graph를 만든다. 이때, Source에서 Sink에 도달할 수 없으면 종료된다.**
2. Level Graph에서 모든 Blocking Flow를 찾아 그 유량만큼 총 유량에 더하고 1.로 돌아간다. (residual capacity는 바뀌지만, level은 그대로이다.)



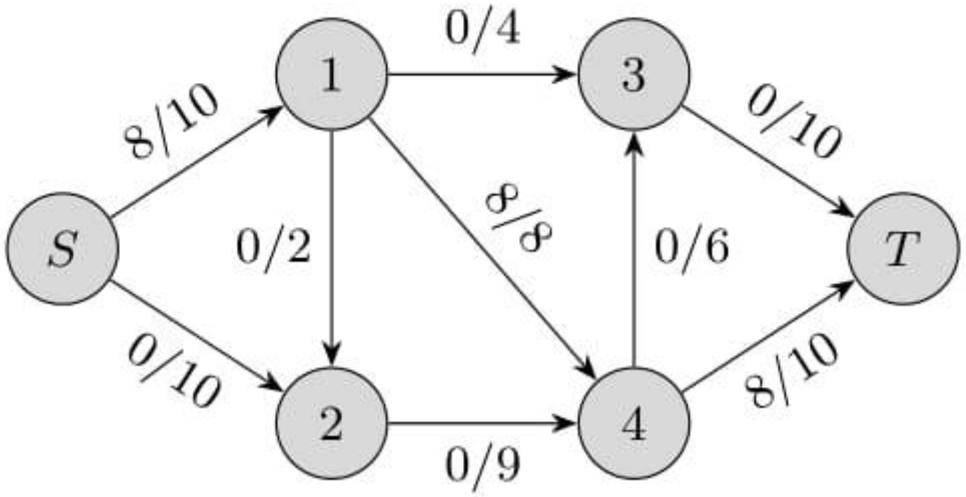


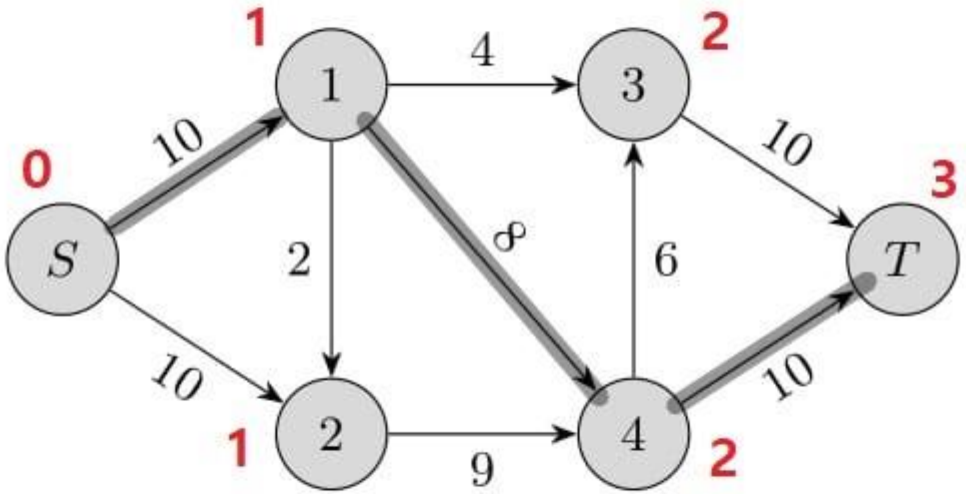
Ⅰ. Dinic

2. 개념 – 1번째 iteration

- 아래와 같은 Flow Network가 주어졌다면, Residual Network는 오른쪽과 같다.

1. Level Graph를 만든다. 이때, Source에서 Sink에 도달할 수 없으면 종료된다.
2. **Level Graph에서 모든 Blocking Flow를 찾아 그 유량만큼 총 유량에 더하고 1.로 돌아간다. (residual capacity는 바뀌지만, level은 그대로이다.)**



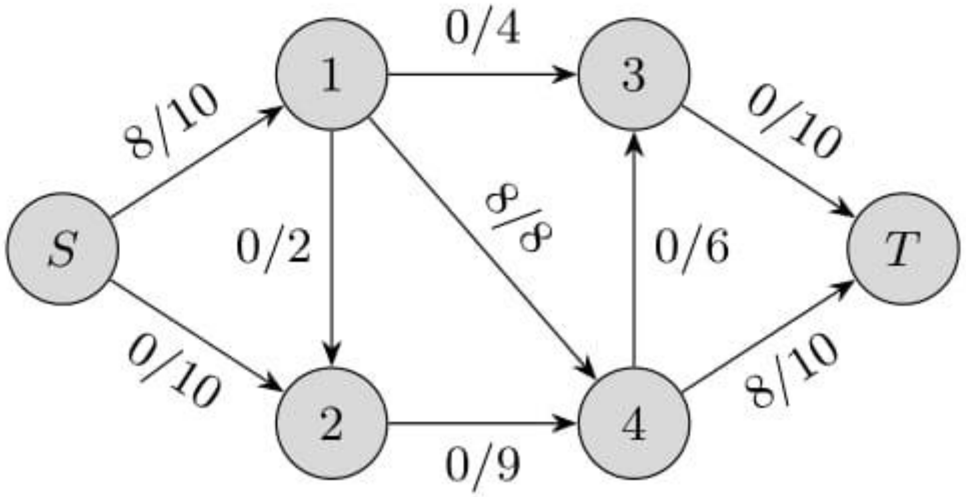


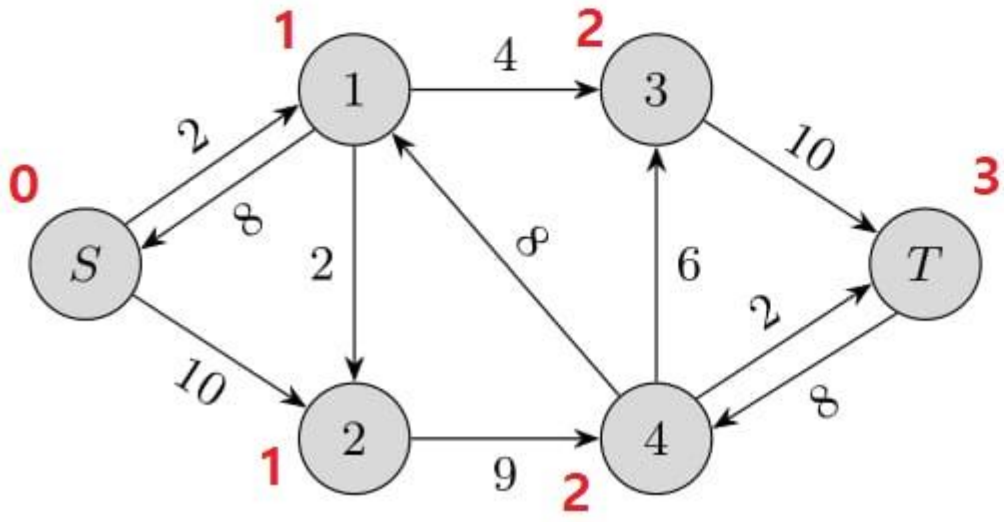
Ⅰ. Dinic

2. 개념 – 1번째 iteration

- 아래와 같은 Flow Network가 주어졌다면, Residual Network는 오른쪽과 같다.

1. Level Graph를 만든다. 이때, Source에서 Sink에 도달할 수 없으면 종료된다.
2. **Level Graph에서 모든 Blocking Flow를 찾아 그 유량만큼 총 유량에 더하고 1.로 돌아간다. (residual capacity는 바뀌지만, level은 그대로이다.)**

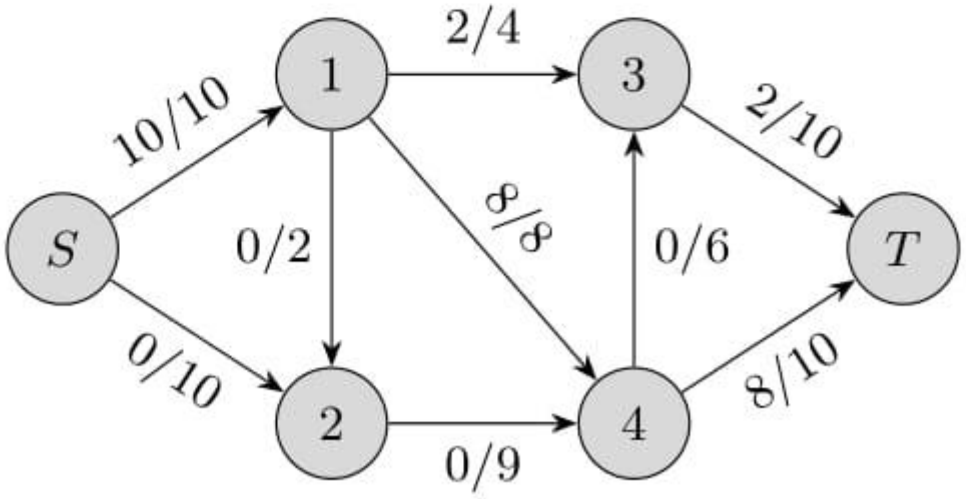


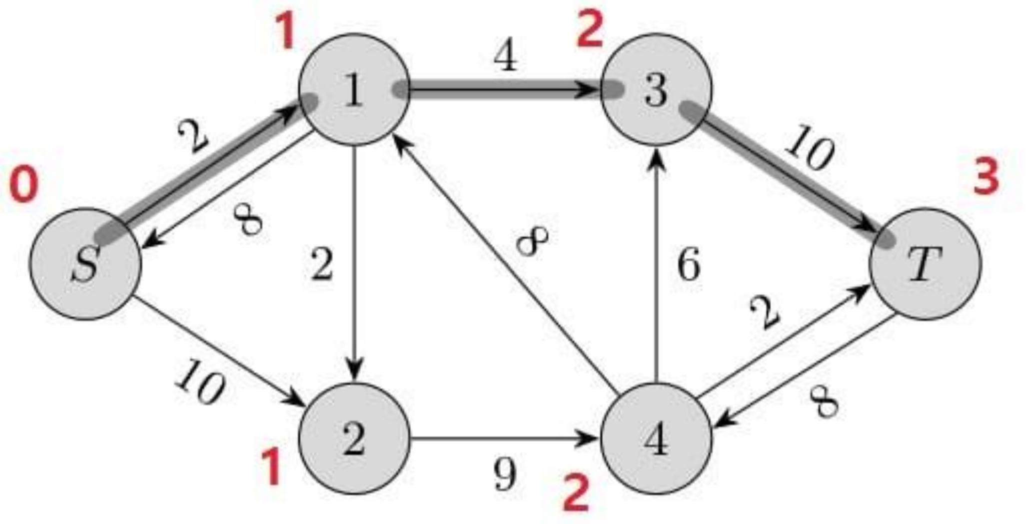


2. 개념 – 1번째 iteration

- 아래와 같은 Flow Network가 주어졌다면, Residual Network는 오른쪽과 같다.

1. Level Graph를 만든다. 이때, Source에서 Sink에 도달할 수 없으면 종료된다.
2. **Level Graph에서 모든 Blocking Flow를 찾아 그 유량만큼 총 유량에 더하고 1.로 돌아간다. (residual capacity는 바뀌지만, level은 그대로이다.)**



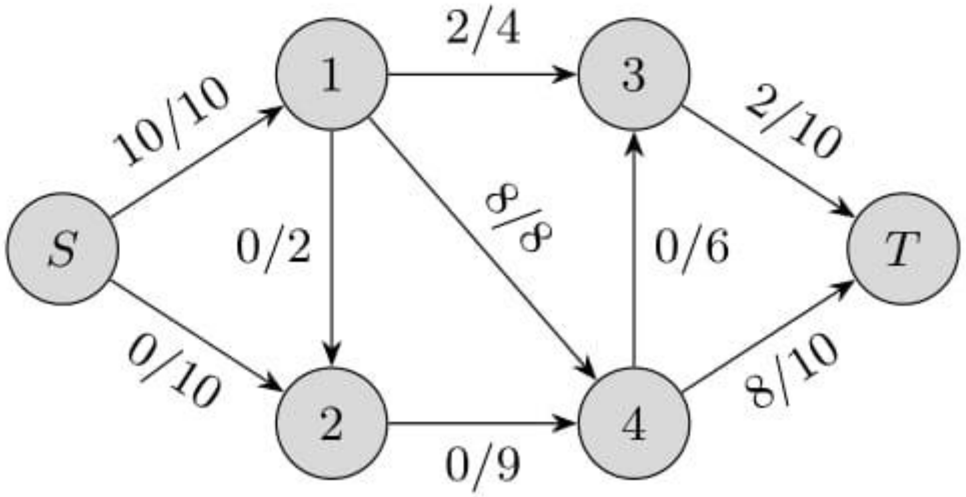


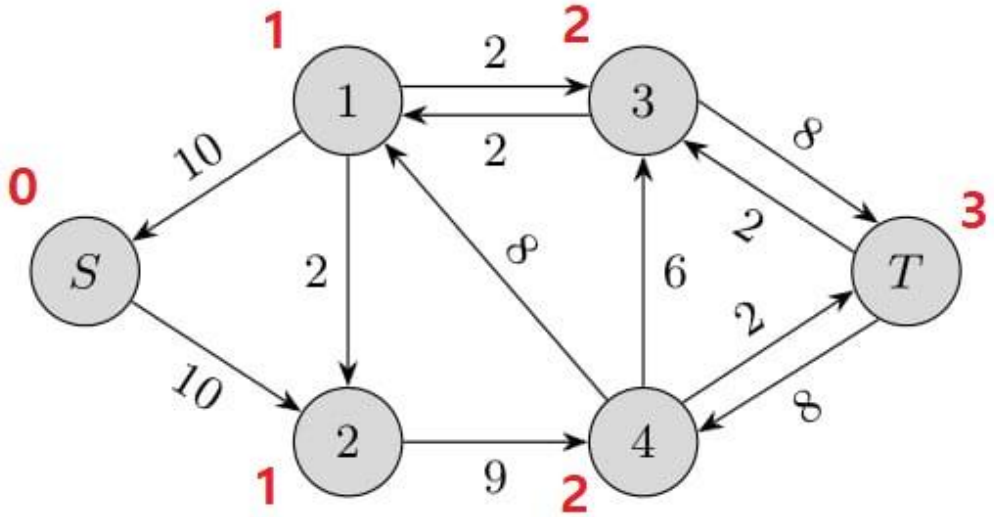
Ⅰ. Dinic

2. 개념 – 1번째 iteration

- 아래와 같은 Flow Network가 주어졌다면, Residual Network는 오른쪽과 같다.

1. Level Graph를 만든다. 이때, Source에서 Sink에 도달할 수 없으면 종료된다.
2. **Level Graph에서 모든 Blocking Flow를 찾아 그 유량만큼 총 유량에 더하고 1.로 돌아간다. (residual capacity는 바뀌지만, level은 그대로이다.)**



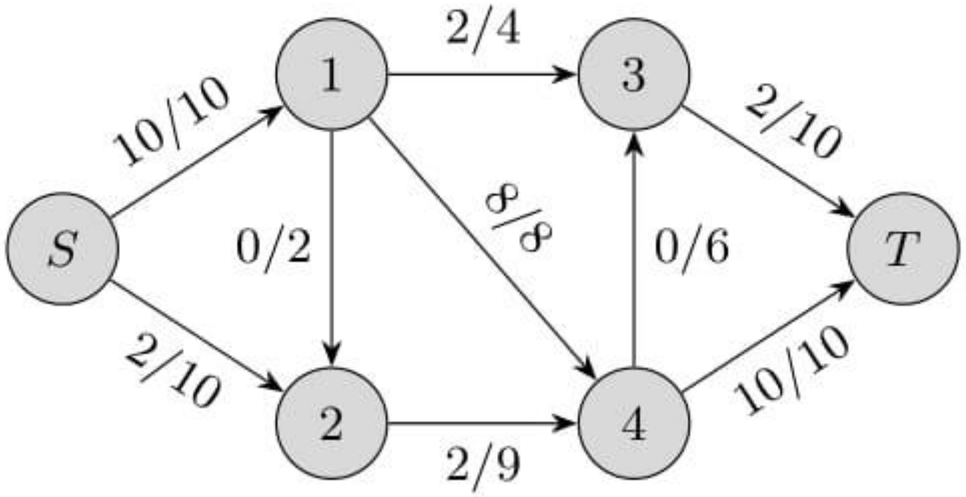


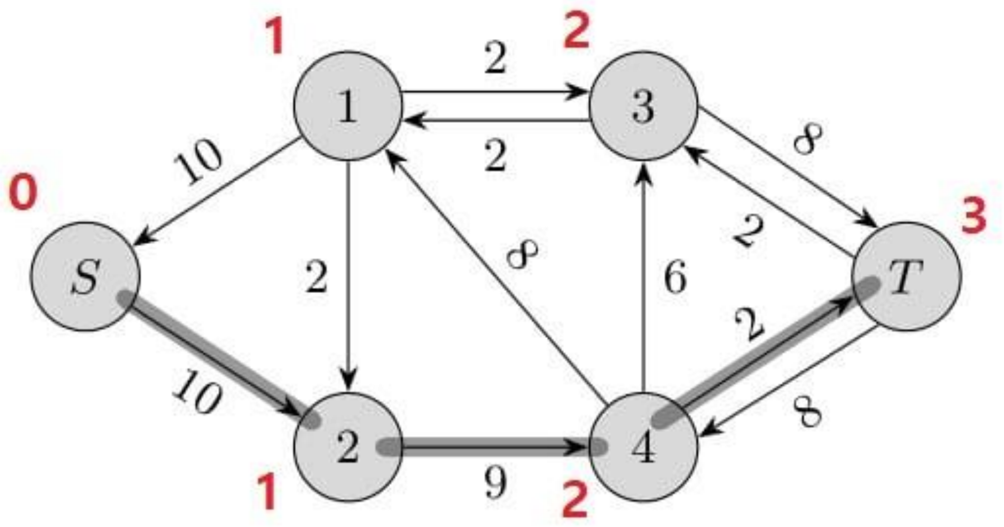
Ⅰ. Dinic

2. 개념 – 1번째 iteration

- 아래와 같은 Flow Network가 주어졌다면, Residual Network는 오른쪽과 같다.

1. Level Graph를 만든다. 이때, Source에서 Sink에 도달할 수 없으면 종료된다.
2. **Level Graph에서 모든 Blocking Flow를 찾아 그 유량만큼 총 유량에 더하고 1.로 돌아간다. (residual capacity는 바뀌지만, level은 그대로이다.)**



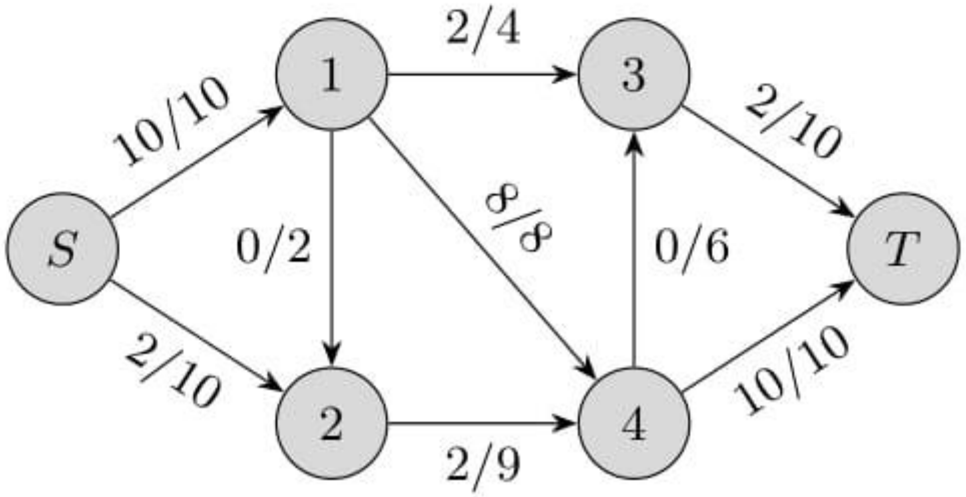


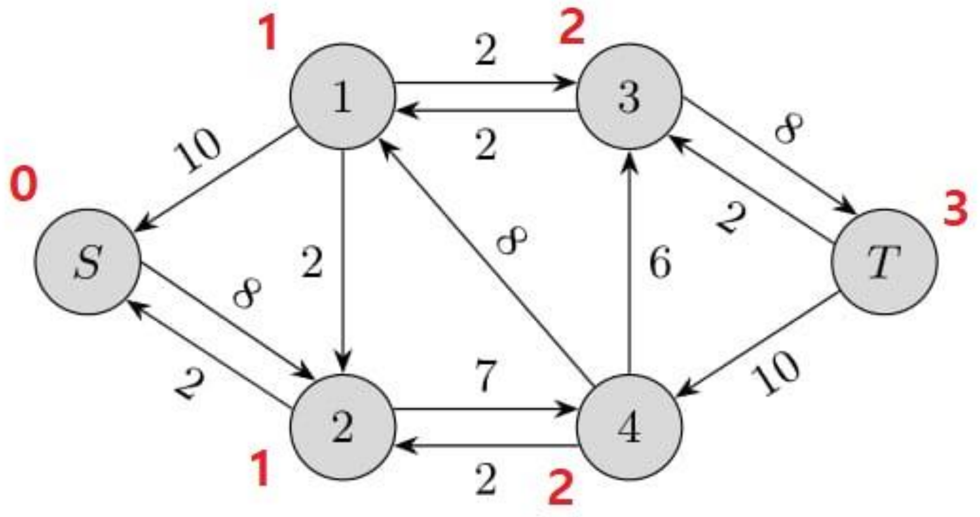
Ⅰ. Dinic

2. 개념 – 1번째 iteration

- 아래와 같은 Flow Network가 주어졌다면, Residual Network는 오른쪽과 같다.

1. Level Graph를 만든다. 이때, Source에서 Sink에 도달할 수 없으면 종료된다.
2. **Level Graph에서 모든 Blocking Flow를 찾아 그 유량만큼 총 유량에 더하고 1.로 돌아간다. (residual capacity는 바뀌지만, level은 그대로이다.)**



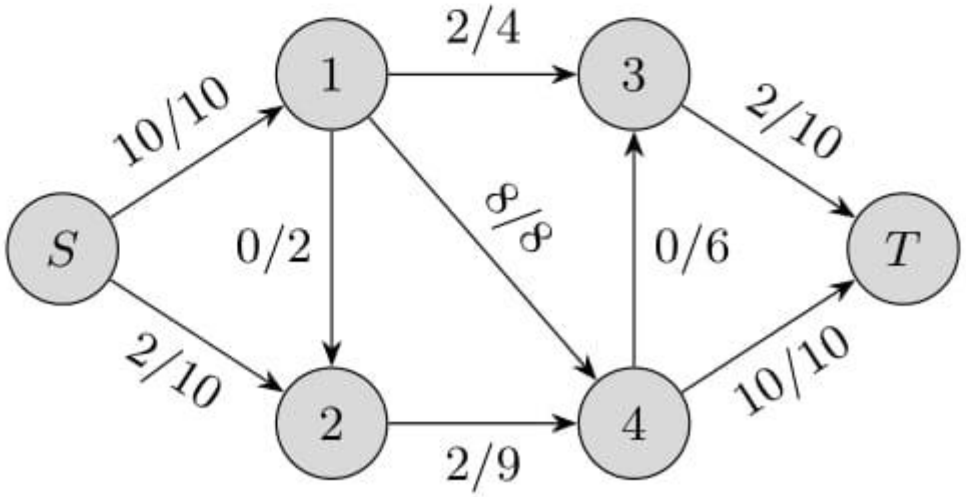


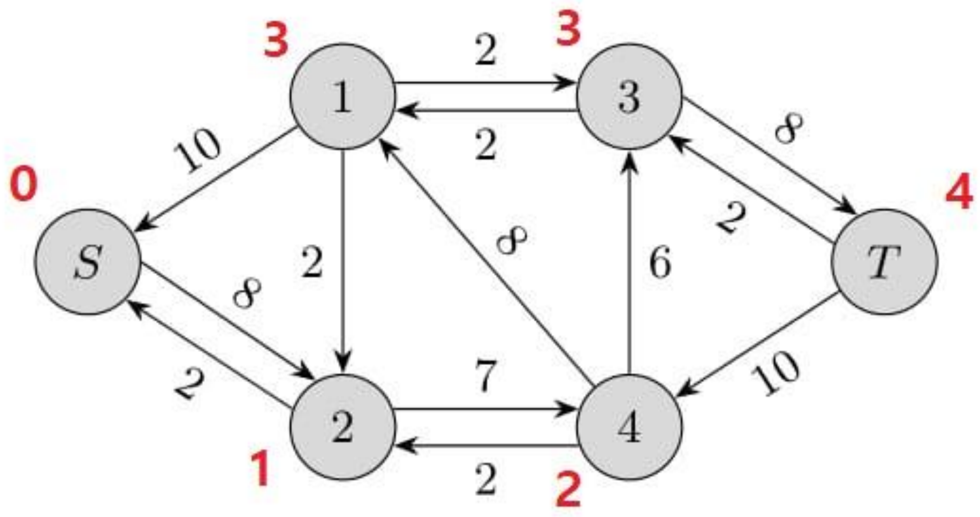
Ⅰ. Dinic

2. 개념 – 2번째 iteration

- 아래와 같은 Flow Network가 주어졌다면, Residual Network는 오른쪽과 같다.

1. **Level Graph를 만든다. 이때, Source에서 Sink에 도달할 수 없으면 종료된다.**
2. Level Graph에서 모든 Blocking Flow를 찾아 그 유량만큼 총 유량에 더하고 1.로 돌아간다. (residual capacity는 바뀌지만, level은 그대로이다.)



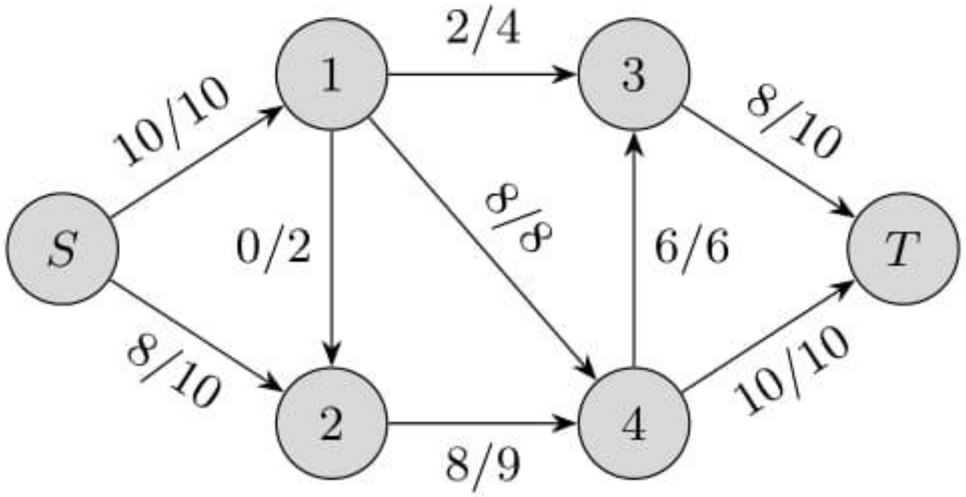


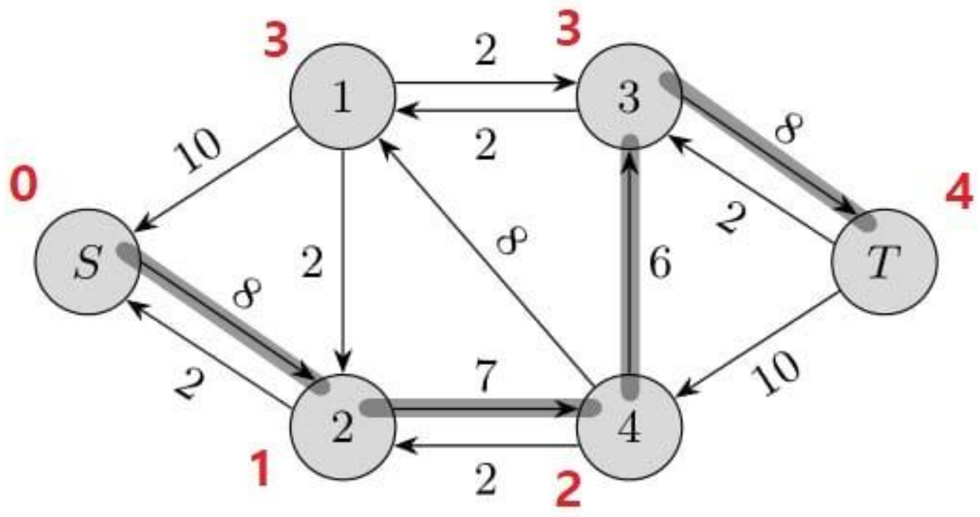
Ⅰ. Dinic

2. 개념 – 2번째 iteration

- 아래와 같은 Flow Network가 주어졌다면, Residual Network는 오른쪽과 같다.

1. Level Graph를 만든다. 이때, Source에서 Sink에 도달할 수 없으면 종료된다.
2. **Level Graph에서 모든 Blocking Flow를 찾아 그 유량만큼 총 유량에 더하고 1.로 돌아간다. (residual capacity는 바뀌지만, level은 그대로이다.)**



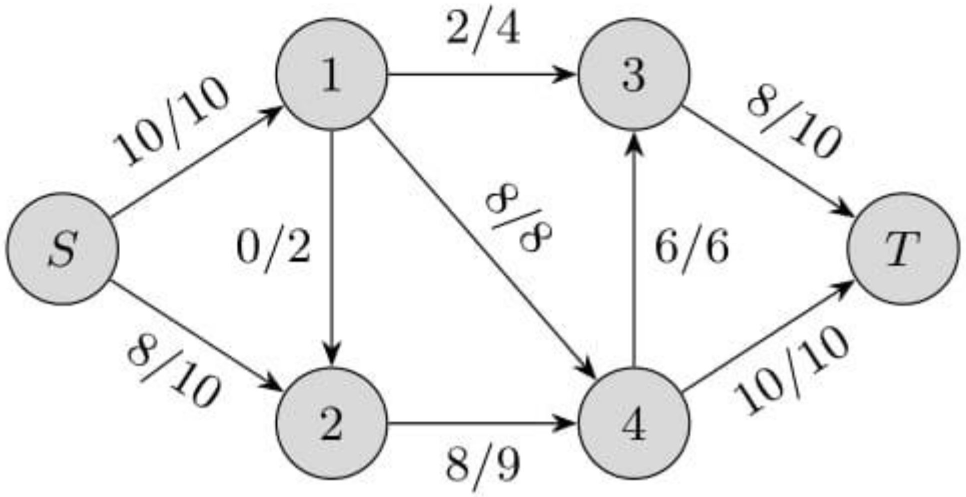


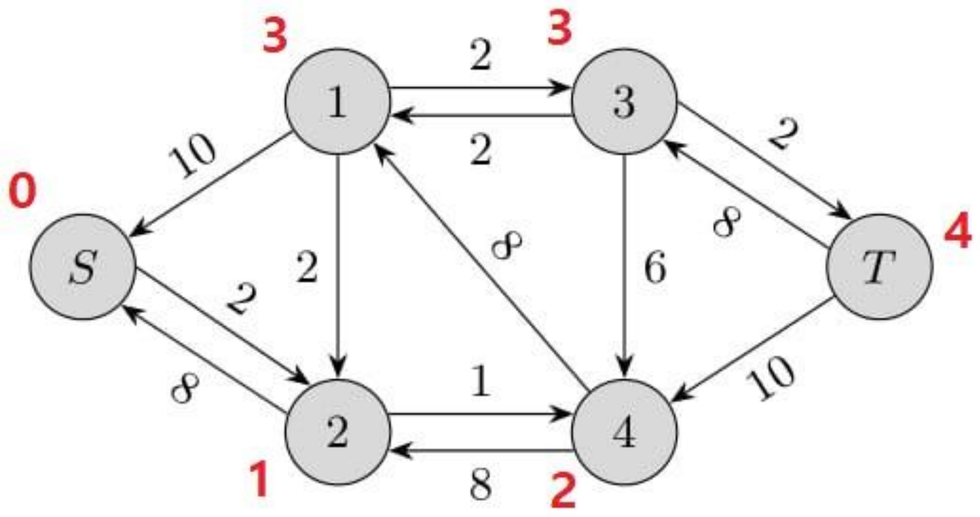
Ⅰ. Dinic

2. 개념 – 2번째 iteration

- 아래와 같은 Flow Network가 주어졌다면, Residual Network는 오른쪽과 같다.

1. Level Graph를 만든다. 이때, Source에서 Sink에 도달할 수 없으면 종료된다.
2. **Level Graph에서 모든 Blocking Flow를 찾아 그 유량만큼 총 유량에 더하고 1.로 돌아간다. (residual capacity는 바뀌지만, level은 그대로이다.)**



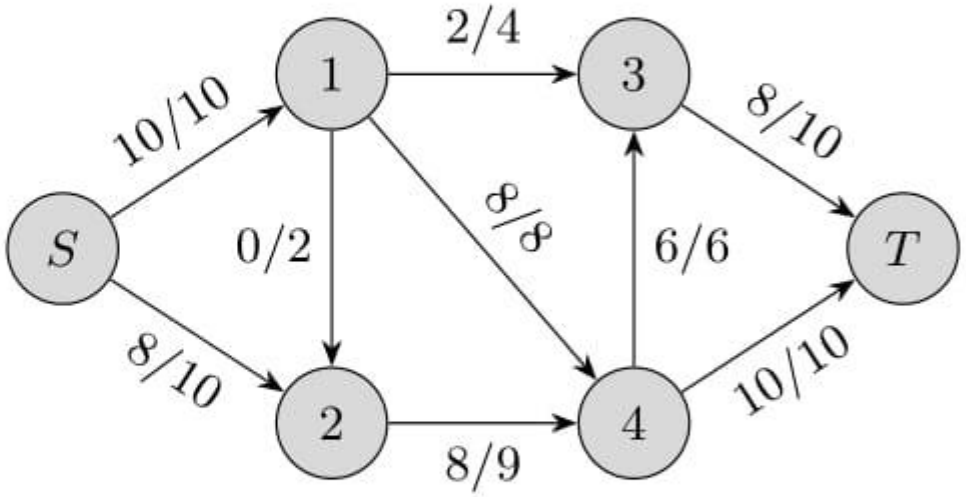


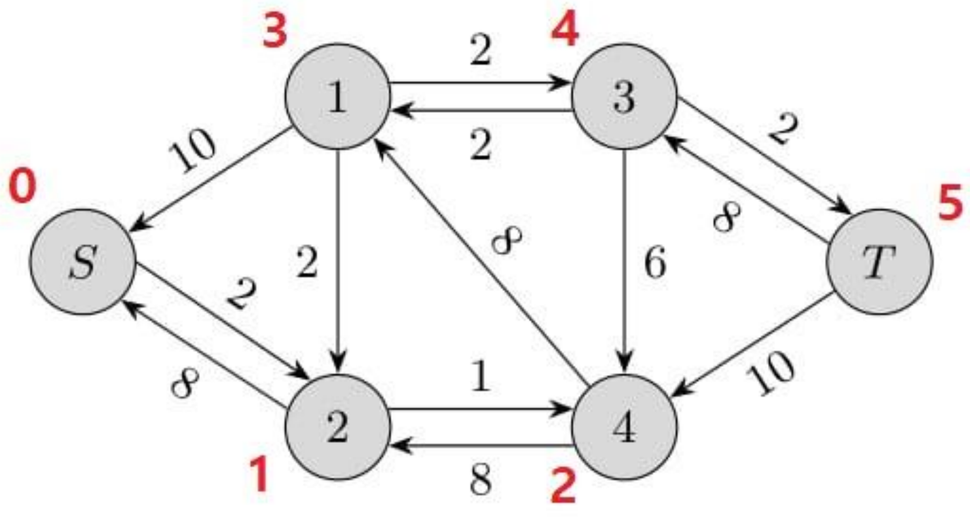
Ⅰ. Dinic

2. 개념 – 3번째 iteration

- 아래와 같은 Flow Network가 주어졌다면, Residual Network는 오른쪽과 같다.

1. **Level Graph를 만든다. 이때, Source에서 Sink에 도달할 수 없으면 종료된다.**
2. Level Graph에서 모든 Blocking Flow를 찾아 그 유량만큼 총 유량에 더하고 1.로 돌아간다. (residual capacity는 바뀌지만, level은 그대로이다.)



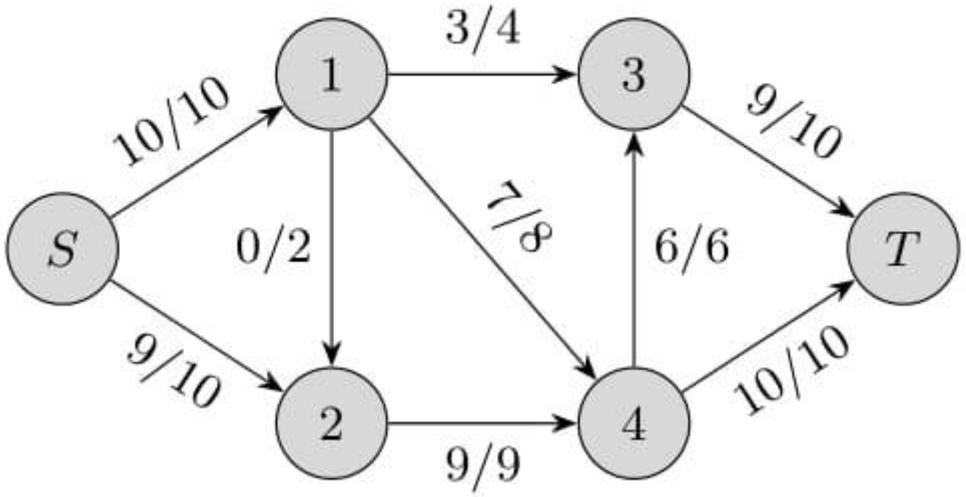


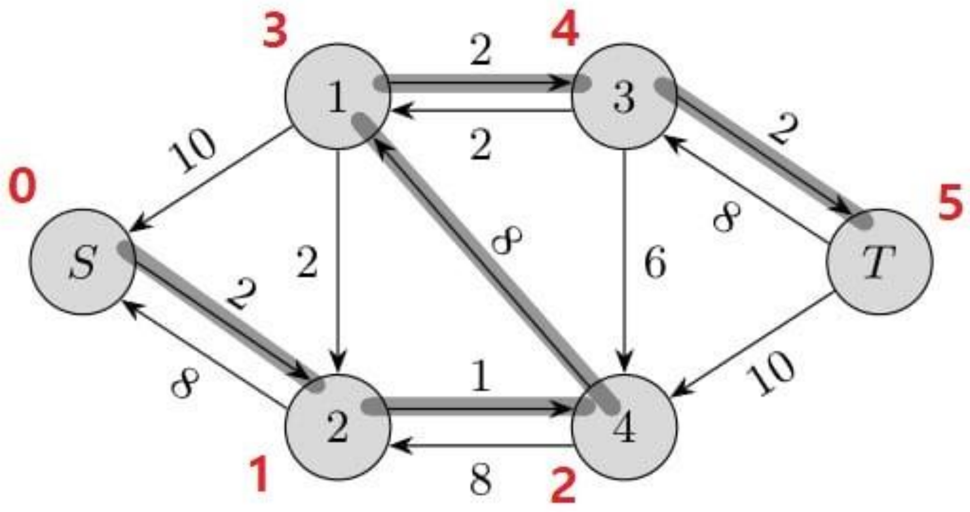
Ⅰ. Dinic

2. 개념 – 3번째 iteration

- 아래와 같은 Flow Network가 주어졌다면, Residual Network는 오른쪽과 같다.

1. Level Graph를 만든다. 이때, Source에서 Sink에 도달할 수 없으면 종료된다.
2. **Level Graph에서 모든 Blocking Flow를 찾아 그 유량만큼 총 유량에 더하고 1.로 돌아간다. (residual capacity는 바뀌지만, level은 그대로이다.)**



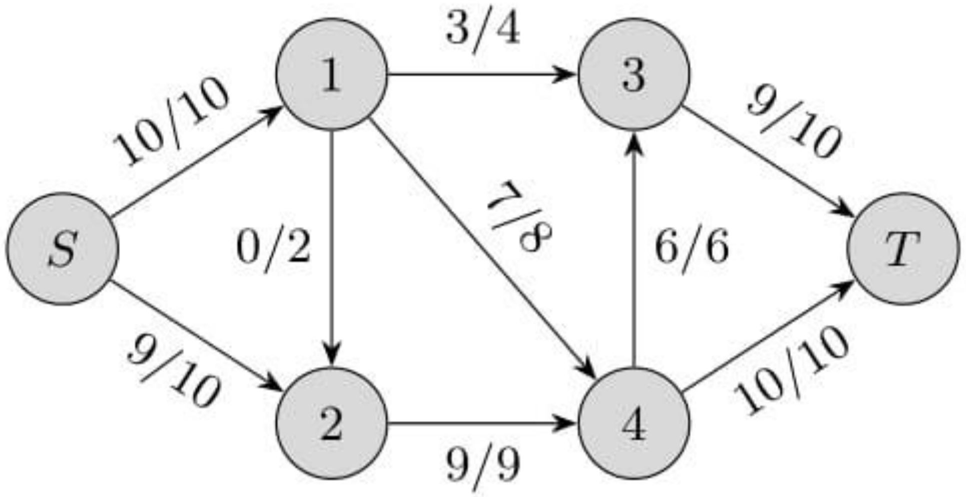


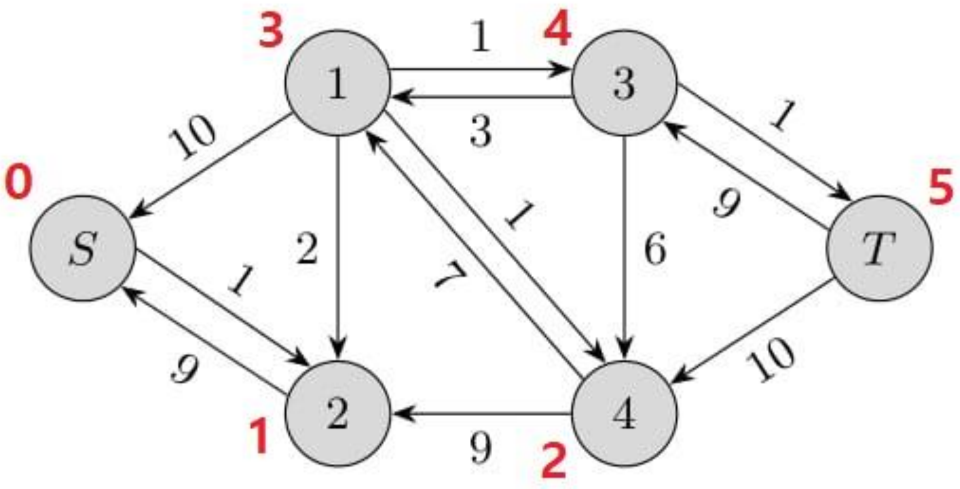
Ⅰ. Dinic

2. 개념 – 3번째 iteration

- 아래와 같은 Flow Network가 주어졌다면, Residual Network는 오른쪽과 같다.

1. Level Graph를 만든다. 이때, Source에서 Sink에 도달할 수 없으면 종료된다.
2. **Level Graph에서 모든 Blocking Flow를 찾아 그 유량만큼 총 유량에 더하고 1.로 돌아간다. (residual capacity는 바뀌지만, level은 그대로이다.)**



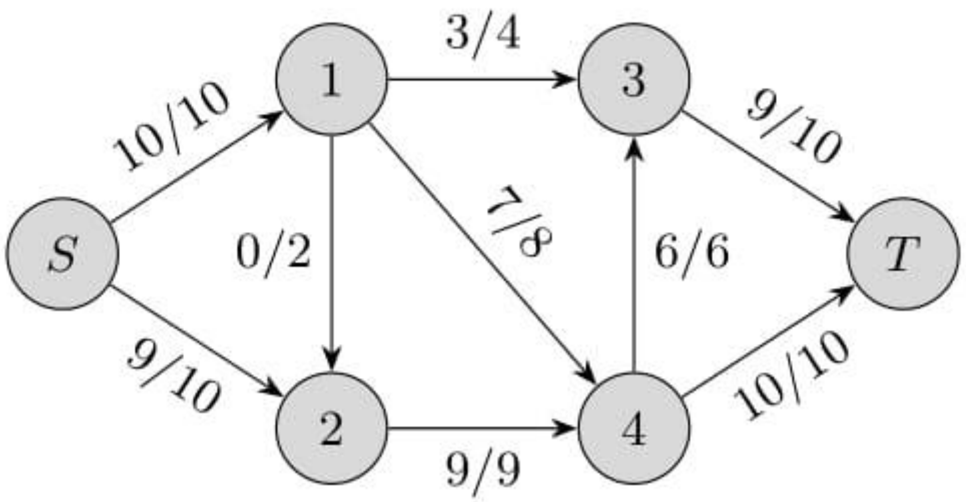


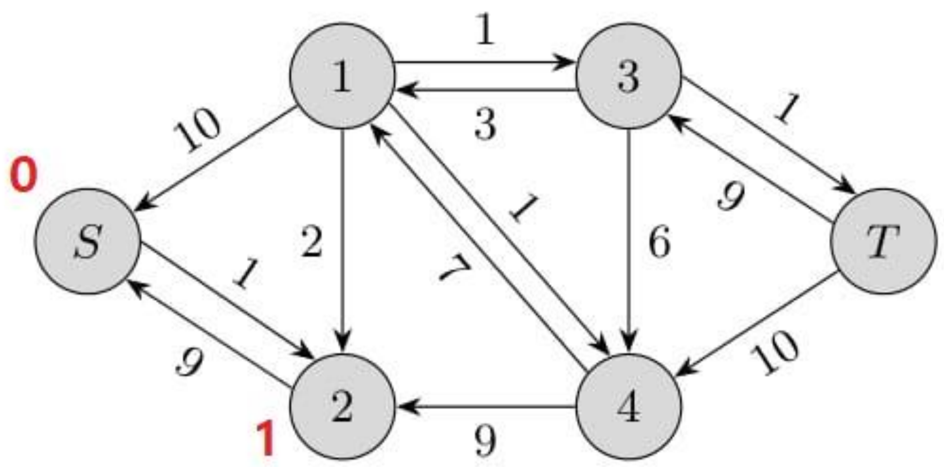
Ⅰ. Dinic

2. 개념 – 4번째 iteration

- 아래와 같은 Flow Network가 주어졌다면, Residual Network는 오른쪽과 같다.

1. **Level Graph를 만든다. 이때, Source에서 Sink에 도달할 수 없으면 종료된다.**
2. Level Graph에서 모든 Blocking Flow를 찾아 그 유량만큼 총 유량에 더하고 1.로 돌아간다. (residual capacity는 바뀌지만, level은 그대로이다.)



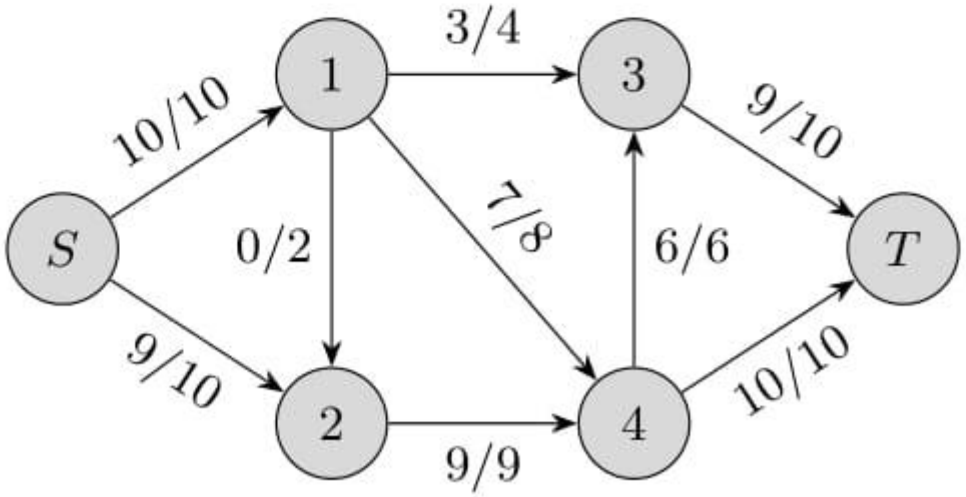


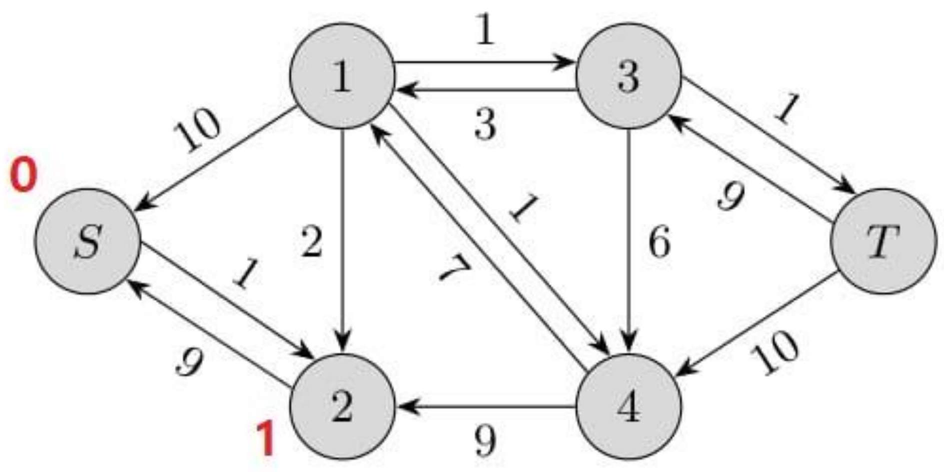
Ⅰ. Dinic

2. 개념 – 4번째 iteration -> 종료

- 아래와 같은 Flow Network가 주어졌다면, Residual Network는 오른쪽과 같다.

1. **Level Graph를 만든다. 이때, Source에서 Sink에 도달할 수 없으면 종료된다.**
2. Level Graph에서 모든 Blocking Flow를 찾아 그 유량만큼 총 유량에 더하고 1.로 돌아간다. (residual capacity는 바뀌지만, level은 그대로이다.)





Ⅰ. Dinic

2. 개념

- 따라서 구하고자 하는 Maximum Flow는 19가 된다.
- 앞서 언급한 바와 같이, 해당 알고리즘의 시간 복잡도는 $O(|V|^2|E|)$이다.
- 그 이유는 **각 1, 2가 최대 $O(|V|)$번 실행**되고
- **2가 최악의 경우 $O(|V||E|)$의 시간 복잡도**를 갖기 때문이다.
- 따라서 **해당 알고리즘의 시간 복잡도는 $O(|V|^2|E|)$**가 된다.
- 하지만 모든 Maximum Flow 알고리즘이 그러하듯 **실제 수행시간은 $O(|V|^2|E|)$보다 훨씬 빠르게 작동**한다.
- 대략 **정점이 500~1,000개까지는 무리없이 작동** 한다고 생각해도 무방하다.