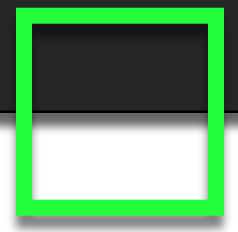
#고급반
#8주차

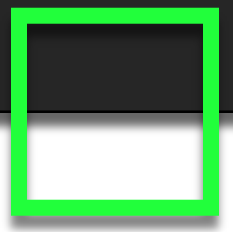
Network Flow, Dinic Algorithm

T. 노주찬



#1

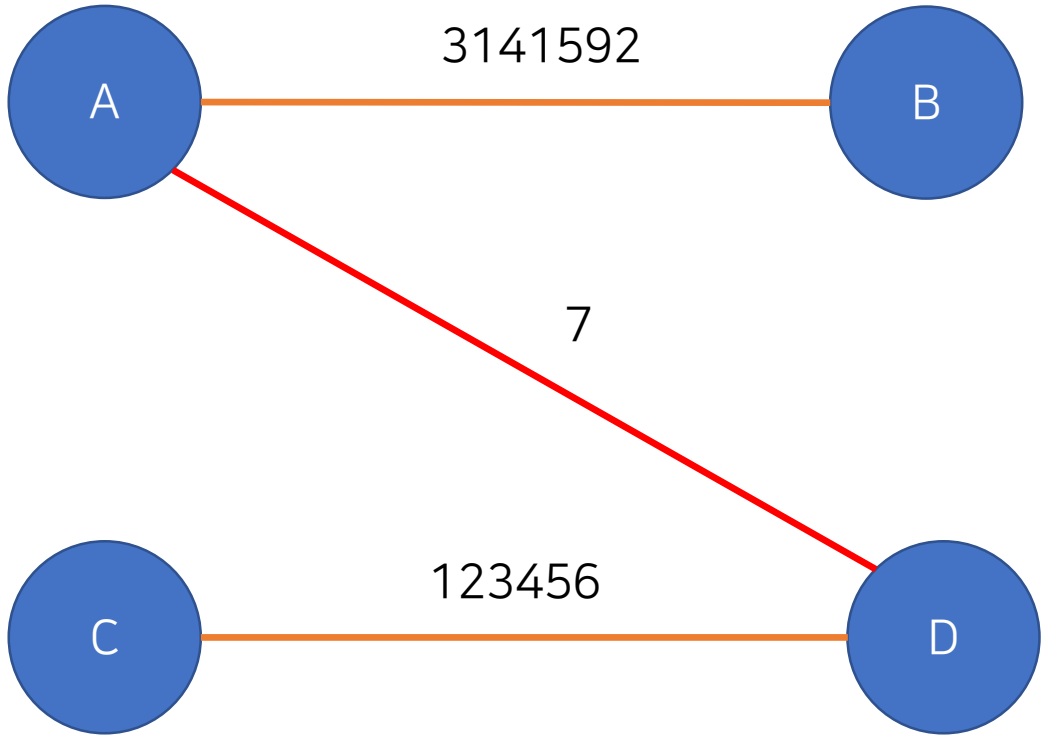
Minimum Cut



Minimum Cut

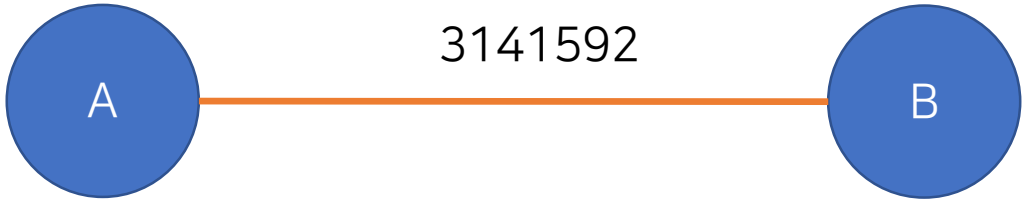
Cut :
하나의 그래프를 두 개의 다른 집합으로
분리시키는 간선들의 집합.

Minimum Cut:
간선의 가중치 (없는 경우 개수) 를 최소로 하는 Cut.



Minimum Cut

Cut :
하나의 그래프를 두 개의 다른 집합으로
분리시키는 간선들의 집합.



Minimum Cut:
간선의 가중치 (없는 경우 개수) 를 최소로 하는 Cut.

오른쪽 그래프의 경우 A와 D를 연결하는 간선이 Minimum Cut.

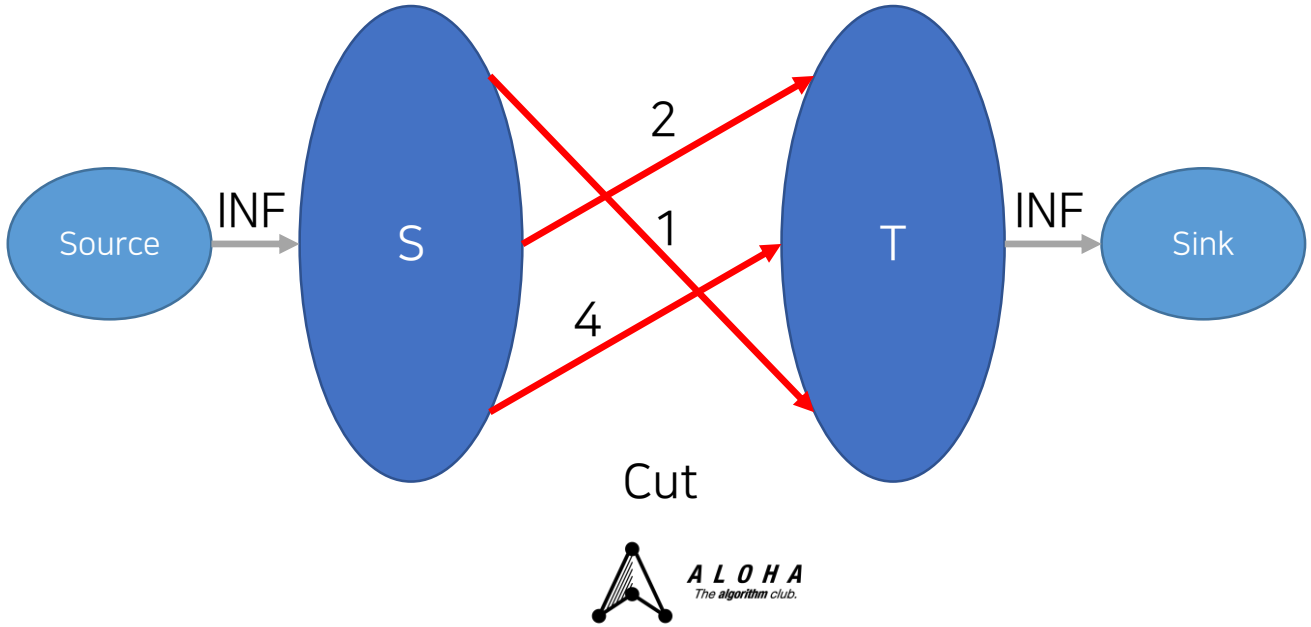
Network Flow 문제 형식의 그래프에선 Source와 Sink를 분리해야하는 경우로 자주 주어진다.

Min-Cut Max-Flow Theorem

요약 : Network Flow 문제 형식의 그래프에서 Minimum Cut은 Maximum Flow 와 같다.

증명 1. |Maximum Flow| <= |Minimum Cut|

임의의 Minimum Cut 을 통해 그래프가 2개의 집합(S, T)으로 나눠진다고 가정하자.
S에서 T로 흐르는 Flow는 그 Cut을 반드시 지나야하므로, 최대 그 Cut의 용량만큼 Flow를 흘려줄 수 있다.



ALOHA
The algorithm club.

Min-Cut Max-Flow Theorem

증명 2. |Maximum Flow| 만큼의 Minimum Cut을 구할 수 있다.

1. 증가경로가 없을 때까지 유량을 구해서, 최대유량을 구했다고 가정하자.
2. Source 에서 도달 가능한 정점들의 집합을 S, 그 여집합을 T라고 하자.
3. S에 속하는 정점에서 T에 속하는 정점을 잇는 간선의 집합 C 에 속하는 간선들은 모두 포화되어있다.
(포화되어있지 않으면 그 간선을 타고 Source에서 도달할 수 있기 때문에, 2번에 위배된다.)
4. Source->Sink 로 흐르기 위해선 C에 속하는 간선을 반드시 지나가야 하기 때문에
C를 통해 흐르는 유량이 Maximum Flow 임을 알 수 있다.
5. 이 때, C가 그래프를 S와 T로 나눌 수 있으니 Cut임이 자명하다.

증명 1에 의해 |Maximum Flow| <= |Minimum Cut| 이고,
증명 2에 의해 |Maximum Flow| == |Minimum Cut| 인 Cut이 항상 존재하므로
|Maximum Flow| == |Minimum Cut|.

이해가 어려우니 일단 Minimum Cut 문제는 Maximum Flow 문제로 변환된다는 것만 기억하자.

연습 문제

BOJ 9169

나는 9999번 문제를 풀 수 있다

문제와 별개로 9999번 문제의 정답이 궁금하면 갠톡주세요

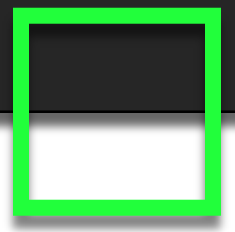
연습 문제

BOJ 1420
학교 가지마!

그래프 모델링이 어렵습니다. 과제로 하는 걸 권장합니다.

#2

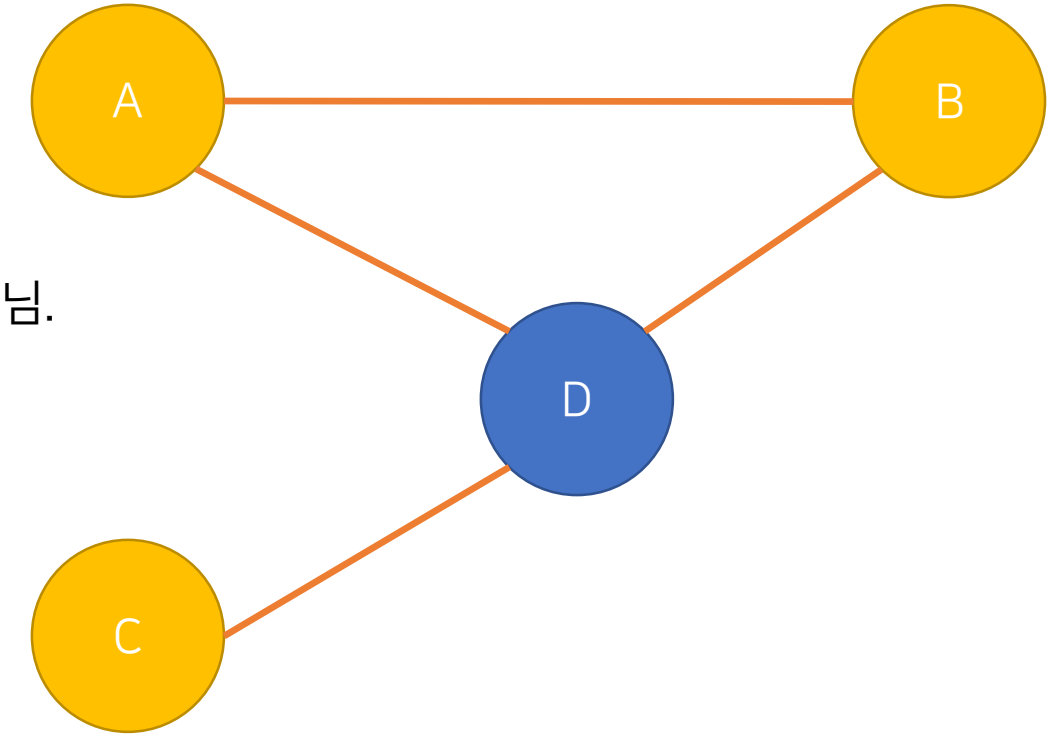
Minimum Vertex Cover



Minimum Vertex Cover

Vertex Cover:
정점들의 집합 중에서, 집합의 요소와 연결된 간선들의 집합이 그래프의 모든 간선인 집합.
<-> 그래프의 간선 중에 Vertex Cover에 속하는 정점과 연결되지 않는 간선은 존재하지 않는다.

Minimum Vertex Cover:
Vertex Cover 중에서 정점 개수를 최소로 하는 Vertex Cover.

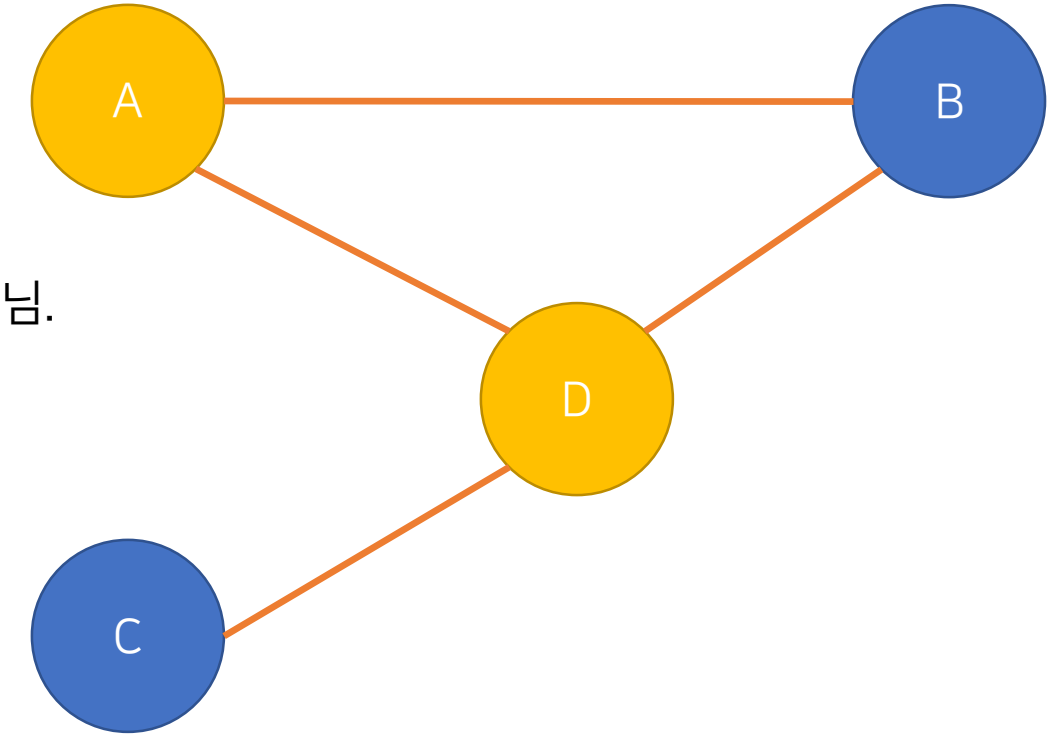


{ A, C, B } 는 Vertex Cover 이지만 { D } 는 Vertex Cover가 아님.
(A-B 를 연결하는 정점을 Cover하지 못함.)

Minimum Vertex Cover

Vertex Cover:
정점들의 집합 중에서, 집합의 요소와 연결된 간선들의 집합이 그래프의 모든 간선인 집합.
<-> 그래프의 간선 중에 Vertex Cover에 속하는 정점과 연결되지 않는 간선은 존재하지 않는다.

Minimum Vertex Cover:
Vertex Cover 중에서 정점 개수를 최소로 하는 Vertex Cover.



{ A, C, B } 는 Vertex Cover 이지만 { D } 는 Vertex Cover가 아님.
(A-B 를 연결하는 정점을 Cover하지 못함.)

{A, D} 또는 {B, D} 는 최소 정점 2개인 Vertex Cover 이므로
Minimum Vertex Cover.

MVC == MM

이분 그래프에서 |Maximum Machting| == |Minimum Vertex Cover|

증명 1. |Maximum Matching| <= |Miniimum Vertex Cover|

Maximum Matching 보다 작은 Vertex Cover가 존재한다고 하자.
그러면 적어도 최소 1개 이상의 Matching은 그 Vertex Cover에 의해 Cover 되지 않을 것이다.
Vertex Cover라는 가정에 모순이므로 |Maximum Matching| <= |Miniimum Vertex Cover|

MVC == MM

증명 2. |Maximum Matching| 만큼의 Vertex Cover가 존재한다.

Min-Cut Max-Flow 증명 2와 같이 S와 T 두 집합으로 나눈다.
왼쪽 정점들의 집합을 L, 오른쪽 정점들의 집합을 R이라하고
L과 R 사이를 잇는 간선들은 모두 무한한 용량을 가진다고 가정하자.

무한한 용량을 가지므로 L과 R 사이를 잇는 간선은 Cut이 될 수 없고, Source – L, R – Sink만 Cut이 될 수 있다.
이 때, Min-Cut을 이루는 간선들의 양 끝 점에 속하는 간선들을 집합으로 표현하면 아래와 같다.

(T & L) | (S & R)

이제 이 집합의 크기는 Minimum Cut과 같고, Minimum Cut은 Maximum Flow의 크기와 같다.

MVC == MM

이제 (T & L) | (S & R) 이 모든 간선을 커버하는지 증명하자.

모든 L-R 간선을 S-T / S-S / T-S / T-T 를 잇는 간선으로 분류해보자.

1. S-T 간선 : 존재할 수 없다. 존재한다면 Cut이 아니므로 Min-Cut Max-Flow의 그래프라는 가정에 어긋난다.

2. S-S 간선 : S-S 간선은 이는 S & R에서 커버된다. (Source, Sink는 Vertex Cover에 포함X)

3. T-T 간선 : T-T 간선은 이는 T & L에서 커버된다.

4. T-S 간선 : T-S 간선은 S & R에서 커버된다.

따라서 |Maximum Matching| == |Vertex Cover| 가 항상 존재한다.
증명 1에 의해 |Maximum Matching| <= |Miniimum Vertex Cover| 이므로
|Maximum Matching| == |Miniimum Vertex Cover|

연습 문제

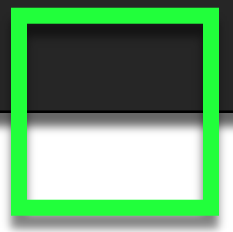
BOJ 1867
돌멩이 제거

연습 문제

BOJ 2414
게시판 구멍 막기

#3

Dinic Algorithm



저번 주 내용 Review

Edmond-Karp Algorithm : 증가경로 (Augmenting Path) 를 BFS 로 찾아서 역방향으로 다시 흘려주는 알고리즘.

Edmon-Karp의 시간복잡도 : O(VE^2)

Why?
BFS 한번의 시간복잡도 : O(E)
BFS의 특성 : 가까운 순서로 정점을 찾는다. -> 증가경로를 최단거리로 찾게된다.
증가경로를 한번 찾을 때마다 적어도 1개의 간선을 포화 간선(capacity == flow 인 간선)으로 만든다.
증가경로의 길이가 D라고 하고 유량을 흘려서 생성되는 역방향 간선을 타게 되면 증가경로의 길이가 D+2가 된다.
즉, 일정한 최단경로에 대해서 새로 생성되는 역방향 간선을 타는 일은 존재하지 않으니
모든 간선이 막힐 때까지 최대 O(E) 번까지 증가경로를 찾게 된다.

최단경로는 최대 V까지 늘어날 수 있으니 증가경로를 찾는 횟수는 O(VE).
따라서 O(VE^2).

Dinic Algorithm

O(VE^2).

간선의 개수에 대한 제약이 없으면, 간선의 개수는 최대 V^2 개 생길 수 있다. (Connected Graph).
즉, V의 관점으로 시간복잡도를 나타내면 O(V^5).

최악의 경우, V가 100만 되어도 시간 초과가 난다.
~~물론 실제 Network Flow 문제에서 최악의 시간복잡도가 나오는 경우는 매우 적습니다.~~

결론 : 더 나은 시간복잡도의 Network Flow 알고리즘이 필요하다.

Dinic Algorithm

1. BFS를 통해서 비포화 간선을 타고 각 정점들에 Source로부터의 최단거리로 Level 을 지정한다.

2. DFS를 통해서 Level이 1씩 증가하는 순서대로만 증가경로를 찾아서 Flow를 흘린다. (역방향 Flow도.)

3. 더 이상 증가경로를 찾을 수 없을 때까지, 2번 과정을 반복한다.
   단, 반복할 때 이미 찾을 수 없는 간선을 다시 탐색할 필요는 없으니
   증가경로를 찾아서 Flow를 흘릴 때, 각 정점마다 몇번째 간선까지 탐색했는지 기록하고
   2번 과정을 다시 반복할 때 기록한 부분부터 탐색을 시작한다.

4. Sink에 Level 을 지정할 수 없을 때까지 1~3 과정을 반복한다.

5. 그 때까지 구한 Flow가 Maximum Flow.

Dinic Algorithm

Dinic 의 시간복잡도: O(V^2 E)

Why?
BFS 한번의 시간복잡도: O(E)
특정 최단경로에 대해 DFS가 **총** 돌아가는 시간복잡도: O(E)
특정 최단경로에 대해 증가경로를 찾는 횟수: O(E)
(최단경로를 찾을 때마다 최소 1개의 간선이 포화되므로.)
증가경로의 길이는 최대 V까지 늘어날 수 있으니 증가경로 1번에 Flow를 흘리는 시간복잡도: O(V)

BFS 한번에 일어나는 작업의 시간복잡도는 O(E) + O(E) * O(V) = O(VE)

최단경로는 최대 V까지 늘어날 수 있으니 Level을 매기는 횟수는 O(V).
따라서 O(V^2 E).

연습 문제

BOJ 13161
분단의 슬픔

수고했어요!