Network Flow 2

나정휘

https://justiceHui.github.io/

목차

Review

- 지난 시간에 배운 것
  - Ford-Fulkerson: 증가 경로를 찾아서 유량을 보내는 것을 반복하면 최대 유량 구할 수 있음
  - Edmonds-Karp: 매번 증가 경로로 유량을 보내면 $O(VE^2)$
  - Max-Flow Min-Cost Theorem: 최대 유량 = 최소 절단

Definition

Definition

- Bipartite Graph
  - 그래프 $G = (L \cup R, E)$의 모든 $(u, v) \in E$가 $u \in L, v \in R$을 만족하면 "이분 그래프"라고 부름
    - 정점 집합을 교집합이 없는 두 부분 집합 L, R로 나눴을 때 각 집합 안에 간선이 없는 그래프
  - 성질
    - 이분 그래프 ⇔ 2-colorable
    - 이분 그래프 ⇔ 홀수 길이 사이클 없음
    - 오늘은 몰라도 되는 내용

Definition

- 무향 그래프 $G = (V, E)$에서...
  - Matching
    - $M \subseteq E$의 모든 간선이 서로 인접하지 않으면 M을 "매칭"이라고 부름
    - 최소 매칭의 크기는 0, 최대 매칭의 크기는?
    - 일반적인 그래프에서 $O(|V|^3)$어려움 또는 $O(|E|\sqrt{|V|})$매우 어려움
  - Vertex Cover
    - 그래프의 모든 간선의 끝점 중 최소 한 개가 $C \subseteq V$에 속하면 C를 "정점 덮개"라고 부름
    - 최대 정점 덮개의 크기는 $|V|$, 최소 정점 덮개의 크기는?
    - 일반적인 그래프에서 NP-Complete
  - Independent Set
    - $I \subseteq V$의 모든 정점이 서로 인접하지 않으면 I를 "독립 집합"이라고 부름
    - 최소 독립 집합의 크기는 0, 최대 독립 집합의 크기는?
    - 일반적인 그래프에서 NP-Complete

Definition

- 방향 그래프 $G = (V, E)$에서...
  - Path Cover
    - 단순 경로들의 집합 $C = \{P_1, P_2, \cdots, P_k\}$가 아래 조건을 만족하면 C를 "경로 덮개"라고 부름
      - 모든 $i \neq j$에 대해 $V(P_i) \cap V(P_j) = \emptyset$: 정점이 겹치지 않음
      - $\cup_{P \in C} V(P) = V$
    - 최대 경로 덮개의 크기는 $|V|$, 최소 경로 덮개의 크기는?
    - 일반적인 그래프에서 NP-Complete
- Partially ordered set에서 유도된 DAG $G = (V, E)$에서...
  - $E = \{(u, v); u \leq v\}$: 비교 가능한 원소들을 방향 간선으로 연결
  - Antichain
    - $A \subseteq V$의 모든 정점이 서로 이동하는 경로가 존재하지 않으면 A를 "반사슬"이라고 부름
      - 서로 위상 관계가 없는 정점들의 집합
    - 최소 반사슬의 크기는 0, 최대 반사슬의 크기는?

Definition

- 이번 시간에 배우는 것
  - Bipartite Matching
    - 이분 그래프의 최대 매칭을 $O(|V||E|)$에 구하는 방법
  - Konig's Theorem
    - 이분 그래프의 최소 정점 덮개를 $O(|V||E|)$에 구하는 방법
    - 이분 그래프의 최대 독립 집합을 $O(|V||E|)$에 구하는 방법
  - Dilworth's Theorem
    - DAG의 최소 경로 덮개를 $O(|V||E|)$에 구하는 방법
    - Poset의 최대 반사슬을 $O(|V||E|)$에 구하는 방법

질문?

Bipartite Matching

Bipartite Matching

- 이분 매칭
  - 이분 그래프 $G = (L \cup R, E)$에서 매칭의 개수를 최대화해야 함

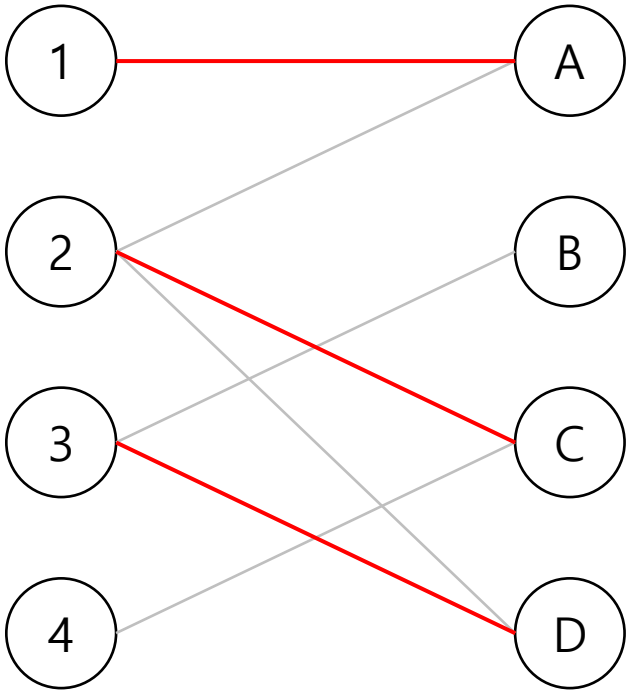
  - 최대 유량 모델링
    - source와 $u \in L$을 용량이 1인 간선으로
    - $u \in L$과 $v \in R$을 용량이 ∞인 간선으로
    - $v \in R$과 sink를 용량이 1인 간선으로
    - $(u, v) \in M$은 source - u - v – sink 경로에 대응됨
    - 최대 매칭 = 최대 유량

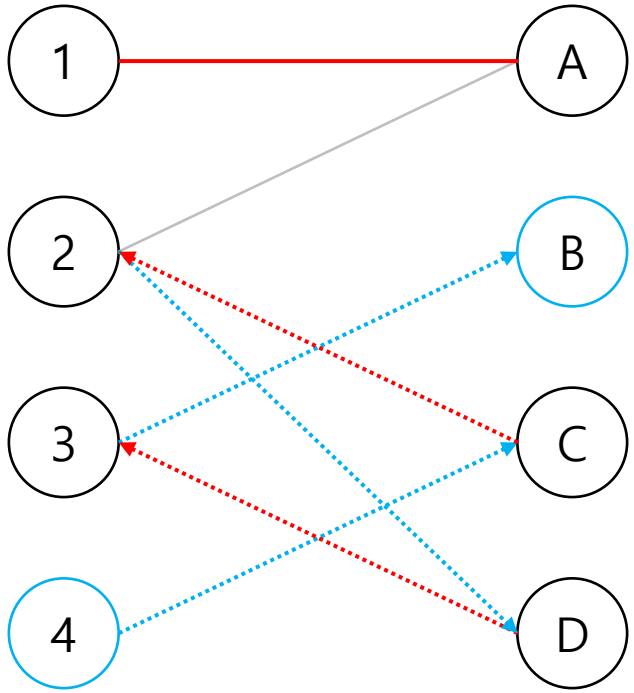
    - 하지만 더 쉽게 구현할 수 있음

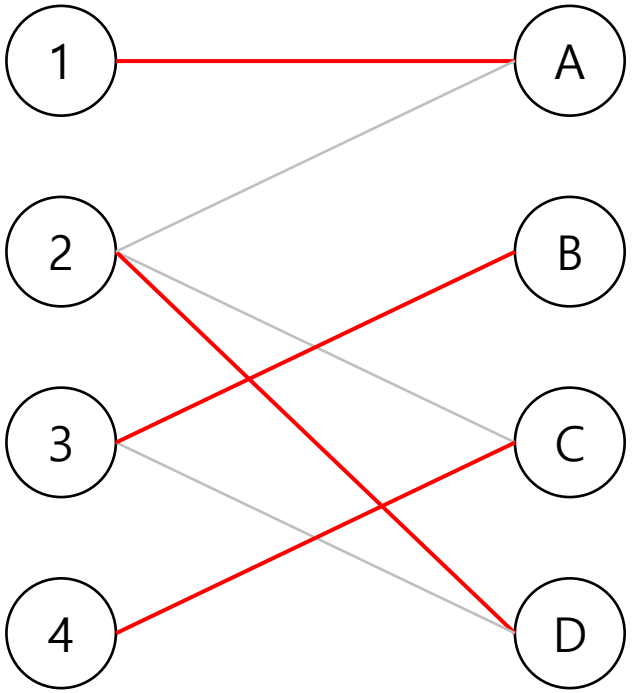
Bipartite Matching

- 이분 매칭
  - 증가 경로
    - 증가 경로에서 S, T에 붙어 있는 간선을 제외한 다른 간선을 보면...
      - 매칭 아닌 간선 – 매칭 간선 – 매칭 아닌 간선 – 매칭 간선 – ... 반복
    - 증가 경로를 따라 유량을 흘리는 것 = 간선들의 매칭 여부를 반전시키는 것
      - $|f| = |M|$이 1 만큼 증가
  - 유량 네트워크를 명시적으로 구축하지 않아도 됨

Bipartite Matching







Bipartite Matching

- 구현
  - G[u]: $u \in L$과 연결된 정점 $v \in R$의 리스트
  - R[v]: $v \in R$과 매칭된 정점 $u \in L$, 없으면 -1
  - C[v]: 증가 경로를 찾을 때 $v \in R$을 방문했으면 1

  - Augment(u): $u \in L$에서 시작하는 증가 경로가 있으면 true
    - 만약 R[i] = -1 이거나 R[i]부터 시작하는 증가 경로가 존재한다면
    - R[i] = u로 변경 (매칭 여부 반전)

  - Augment(1..|V|) 호출하면 최대 매칭을 구할 수 있음
  - $O(|E|)$ DFS를 $|V|$번 수행하므로 $O(|V||E|)$
    - Ford-Fulkerson 써도 $|f| \leq |V|$이므로 $O(|V||E|)$

```
bool Augment(int v){
    for(auto i : G[v]){
        if(C[i]) continue; C[i] = 1;
        if(R[i] == -1 || Augment(R[i])){
            R[i] = v; return true;
        }
    }
    return false;
}

int Match(){
    int res = 0;
    memset(R, -1, sizeof R);
    for(int i=1; i<=N; i++){
        memset(C, 0, sizeof C);
        res += Augment(i);
    }
    return res;
}
```

Bipartite Matching

- BOJ 11375 열혈강호
  - 이분 매칭 구현 문제

- BOJ 11376 열혈강호 2
  - 왼쪽 정점을 최대 2번씩 사용할 수 있는 문제

  - 앞에서 본 알고리즘은 $v \in R$의 사용 횟수만 1로 제한하고 $u \in L$의 사용 횟수는 신경 안 씀
    - 최대 매칭 구할 때는 Augment(1..N)을 1번씩만 호출해서 $u \in L$의 사용 횟수를 1로 제한
    - 따라서 Augment(1..N)을 2번씩 호출하면 됨

질문?

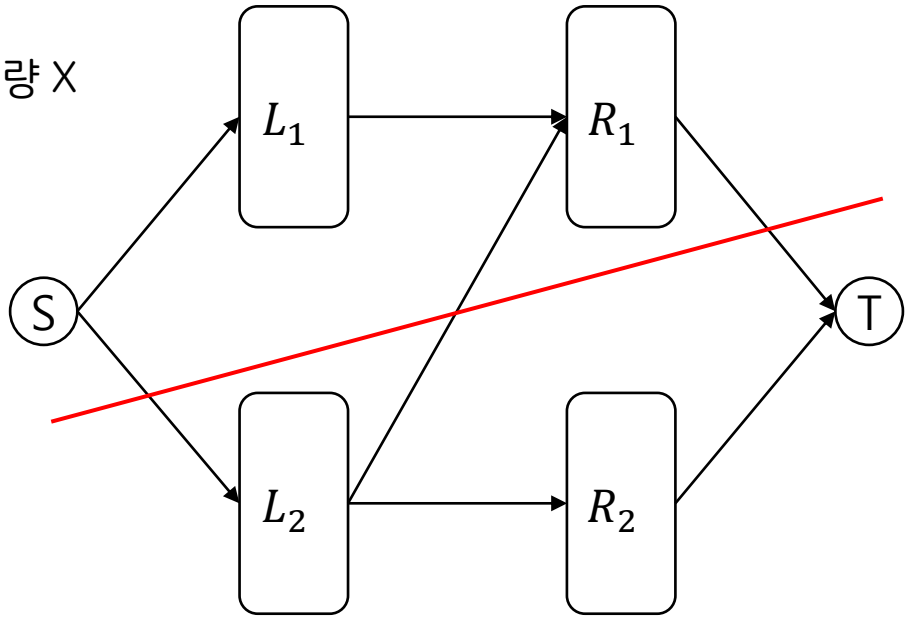
Konig's Theorem

Konig's Theorem

- 정점 덮개의 성질
  - 이분 그래프 $G = (L \cup R, E)$에서 임의의 매칭 M과 정점 덮개 C에 대해 $|M| \leq |C|$가 성립함
    - 매칭을 이루는 간선의 최소한 한쪽 끝점은 정점 덮개에 포함되어야 함
    - 따라서 최대 매칭 ≤ 임의의 정점 덮개
    - 사실 이건 일반적인 그래프에서도 성립
  - 최소 정점 덮개는 최대 매칭과 어떤 연관이 있을까?

Konig's Theorem

- Konig's Theorem
  - 이분 그래프 $G = (L \cup R, E)$의 최대 매칭 M'과 최소 정점 덮개 C'의 크기는 동일함
    - 최소 절단 $(S, V \backslash S)$에 대해 $L_1 \cup R_1 = S, L_2 \cup R_2 = V \backslash S$를 정의하자. (단, $L_* \in L, R_* \in R$)
      - 최소 절단의 용량 $c(S) = |L_2| + |R_1|$
        - $(s, u \in L_2), (v \in R_1, t)$ 간선을 끊기 때문
        - L_2에서 R_1으로 가는 간선은 $S \leftarrow V \backslash S$방향이므로 절단 용량 X
      - L_1에서 R_2로 가는 간선은 존재하지 않음
        - 만약 존재한다면 절단 용량이 ∞가 되기 때문
        - 따라서 L_1의 이웃 $= N(L_1) \subseteq R_1$
        - 마찬가지로 $N(R_2) \subseteq L_2$
    - 따라서 $L_2 \cup R_1$은 정점 덮개
      - $|L_2 \cup R_1| = f(S) = |M'|$
      - $|M'| \leq$ 임의의 정점 덮개의 크기
      - $|L_2 \cup R_1| = |M'|$이므로 $C' = L_2 \cup R_1$은 최소 정점 덮개



Konig's Theorem

- Konig's Theorem
  - 최소 정점 덮개를 찾는 방법
    - 먼저 이분 매칭을 구한 다음
    - L_2 = source에서 도달할 수 없는 왼쪽 정점
    - R_1 = source에서 도달할 수 있는 오른쪽 정점
  - 최대 독립 집합을 찾는 방법
    - 임의의 그래프에서 $I = V \backslash C$
    - 따라서 최소 정점 덮개의 여집합을 구하면 됨

```cpp
bool Augment(int v){
    for(auto i : G[v]){
        if(C[i]) continue; C[i] = 1;
        if(R[i] == -1 || Augment(R[i])){
            L[v] = i; R[i] = v; return true;
        }
    }
    return false;
}

void DFS(int v){
    if(v == -1 || C[v]) return; C[v] = 1;
    for(auto i : G[v]) C[N+i] = 1, DFS(R[i]);
}

vector<int> VertexCover(){
    memset(L, -1, sizeof L);
    memset(R, -1, sizeof R);
    for(int i=1; i<=N; i++) memset(C, 0, sizeof C), Augment(i);
    vector<int> V;
    memset(C, 0, sizeof C);
    for(int i=1; i<=N; i++) if(L[i] == -1) DFS(i);
    for(int i=1; i<=N; i++) if(!C[i]) V.push_back(i);
    for(int i=1; i<=M; i++) if(C[N+i]) V.push_back(N+i);
    return V;
}
```

질문?

Dilworth's Theorem

Dilworth's Theorem

- DAG의 경로 덮개와 이분 그래프
  - DAG $G = (V, E)$에 대해, 다음과 같은 이분 그래프 $G = (L \cup R, F)$를 생각해 보자.
    - $L = \{v_l; v \in V\}, \mathrm{R} = \{v_r; v \in V\}$
    - $F = \{(u_l, v_r); (u, v) \in E\}$

  - Claim: G에 크기가 k인 경로 커버 존재 $\Leftrightarrow$ H에 크기가 $|V| - k$인 매칭 존재
    - H의 최대 매칭을 찾으면 G의 최소 경로 덮개를 찾을 수 있음

Dilworth's Theorem

- DAG의 경로 덮개와 이분 그래프
  - 모든 $i \neq j$에 대해 $V(P_i) \cap V(P_j) = \emptyset$이고 $\bigcup_{P \in C} V(P) = V$를 만족하는 C는 "경로 덮개"
    - 경로의 시작 정점 집합 V_s와 그렇지 않은 정점 집합 $V_t = V \backslash V_s$를 생각해 보자.
    - V_s는 서로 다른 경로들의 시작점을 갖고 있으므로 $|V_s| = |C|$
    - V_t는 C에 속한 모든 간선의 끝점을 포함하므로 $|V_t| = \sum_{P \in C} |P|$
    - 따라서 $|V| = |V_s| + |V_t| = |C| + \sum_{P \in C} |P|$
  - G의 임의의 경로 덮개 C에 대해, H에 $|M| = |V| - |C|$를 만족하는 매칭 M이 존재함
    - $M = \{(u_l, v_r); (u, v) \in P_i\}$을 정의하자. 경로 덮개와 H에 정의에 따라 M은 매칭
      - M에 정점을 공유하는 두 간선 e_1, e_2가 있다면 $P_1 \ni e_1$과 $P_2 \ni e_2$가 정점을 공유하므로 경로 덮개가 아님
    - M은 C의 간선을 포함하므로 $|M| = \sum_{P \in C} |P| = |V| - |C|$

Dilworth's Theorem

- DAG의 경로 덮개와 이분 그래프
  - H의 임의의 매칭 M에 대해, M에서 매칭되지 않은 정점 $u_l \in L$과 $v_r \in R$이 존재함
    - 조건을 만족하는 u_l의 개수와 v_r의 개수는 동일하므로 완전 매칭이 존재하지 않음을 보이면 됨
    - 완전 매칭 $M' = \{(v_{1,l}, v_{2,r}), (v_{2,l}, v_{3,r}), \cdots, (v_{n-1,l}, v_{n,r}), (v_{n,l}, v_{1,r})\}$이 존재한다고 가정하자.
    - 이는 G의 간선 $(v_1, v_2), (v_2, v_3), \cdots, (v_{n-1}, v_n), (v_n, v_1)$에 대응되고, G가 DAG라는 것에 모순
    - 완전 매칭이 존재하지 않으므로 조건을 만족하는 정점 $u_l \in L, v_r \in R$이 존재함

Dilworth's Theorem

- DAG의 경로 덮개와 이분 그래프
  - H의 임의의 매칭 M에 대해, G에 $|C| = |V| - |M|$를 만족하는 경로 덮개 C가 존재함
    - $C = \{(v_1, v_2, \cdots, v_k); (v_{i,l}, v_{i+1,r}) \in M, v_{1,l}\ and\ V_{k,r}\ aren't\ matched\}$
      - M에서 매칭되지 않은 정점 $u_r \in R$을 선택하자.
      - 만약 $u_l \in L$도 매칭되지 않았다면 C에 $P = (u)$ 삽입
      - 그렇지 않은 경우, u_l과 매칭된 $v_r \in R$를 찾고, 만약 $v_l \in L$이 매칭되지 않았다면 $P = (u, v)$
      - 그렇지 않은 경우, 매칭이 되지 않은 정점을 만날 때까지 따라간 정점들을 모두 경로로 잡음
      - 모든 정점을 커버할 때까지 반복하면 됨
        - 한 번 본 정점들을 모두 제거해도 여전히 매칭이므로 매칭되지 않은 정점 $u_r \in R$ 존재
    - C는 올바른 경로 덮개
      - 매칭의 각 간선은 G의 간선에 대응되므로 C의 원소들은 올바른 경로를 구성함
      - 알고리즘을 모든 정점이 커버될 때까지 반복하고, 각 정점을 정확히 한 번씩 커버함
    - 경로를 구성하는 간선은 모두 매칭에 대응되므로 $|M| = \sum |P| = |V| - |C|$
      - 따라서 $|C| = |V| - |M|$

Dilworth's Theorem

- 정리
  - 최소 경로 덮개를 찾는 방법
    - 이분 그래프 $H = (L \cup R, F)$를 만들고
    - H의 최대 매칭 M'을 찾고
    - M'에 대응되는 크기 $|V| - |M'|$짜리 경로 덮개를 찾으면 됨
  - 주의: 아직 Dilworth's Theorem 시작 안 함

질문?

Dilworth's Theorem

- Dilworth's Theorem
  - Poset/Poset에서 유도된 DAG $G = (V, E)$에서 최대 반사슬의 크기 = 최소 경로 커버의 크기
    - 경로 덮개에서 만들었던 이분 그래프 $H = (L \cup R, F)$를 생각해 보자.
    - Konig's Theorem에 의해 최대 매칭 M'과 크기가 같은 최소 정점 덮개 C'이 존재함
    - 이때 $A = \{v; v_l \notin C \ and \ v_r \notin C\}$은 반사슬
      - A에 속한 모든 정점이 없어도 모든 간선을 커버할 수 있으므로 서로를 향한 간선이 없음
      - 따라서 $|A| \geq |V| - |M'|$
    - 최대 매칭 M'에 포함된 간선으로만 구성된 그래프 G'을 생각해 보자.
      - G'에서 각 정점의 in-degree와 out-degree는 최대 1
      - 따라서 G'은 몇 개의 경로(체인)들로 구성되어 있음
      - 체인의 개수는 in-degree가 0인 정점의 개수 = $|V| - |M'|$
      - 반사슬을 만들기 위해서는 각 체인에서 최대 1개만 선택해야 하므로 $|A| \leq |V| - |M'|$
    - 따라서 최대 반사슬 A', 최소 경로 커버 C'에 대해 $|A'| = |V| - |M'| = |C'|$

Dilworth's Theorem

- Dilworth's Theorem
  - 최대 반사슬을 찾는 방법
    - 이분 그래프 $H = (L \cup R, F)$를 만들고
    - H의 최소 정점 커버 C'을 찾고
    - v_l, v_r 모두 C'에 속하지 않는 정점 v를 구하면 됨

질문?

Applications

Applications

- BOJ 1867 돌멩이 제거
  - 격자에 돌멩이가 있음
  - 행 또는 열을 하나 골라서 돌멩이를 전부 제거하는 연산을 수행할 수 있음
  - 돌멩이를 모두 제거하는데 필요한 최소 연산 횟수

  - 웰노운 테크닉: 격자 문제는 보통 이분 그래프를 이용해서 풀 수 있음
    - row + col % 2에 따라 분할하고 상하좌우로 인접한 칸 연결하면 이분 그래프
    - (row, col)에 원소가 있으면 왼쪽의 row번째 정점과 오른쪽의 col번째 정점을 연결해도 이분 그래프

    - 어떤 그래프를 사용해야 할까?

Applications

- BOJ 1867 돌멩이 제거
  - 왼쪽과 오른쪽에 정점을 각각 N개씩 두고, (r, c)에 돌멩이가 있으면 r과 c를 연결하자.
    - 돌멩이는 간선에 대응됨
    - 한 줄에 있는 돌멩이를 모두 없애는 것은 정점 하나를 정점 덮개에 추가하는 것
  - 최소 정점 덮개를 구하는 문제!

Applications

- BOJ 11014 컨닝 2
  - (i, j)에 있는 사람은 (i-1, j-1), (i-1, j+1), (i, j-1), (i, j+1)에 있는 사람의 답지를 배낄 수 있음
  - 아무도 컨닝을 할 수 없도록 학생을 배치할 때, 배치할 수 있는 학생의 최대 수

  - 컨닝할 수 있는 학생들을 간선으로 연결하면 최대 독립 집합을 찾는 문제
  - 그래프가 이분 그래프일까?
    - 각 학생이 배낄 수 있는 답지는 모두 j-1 또는 j+1에 있음
    - j의 기우성에 따라 분할하면 이분 그래프

질문?

Applications

- BOJ 1671 상어의 저녁식사
  - 상어들이 서로 잡아먹을 수 있는 관계가 DAG로 주어짐
    - 능력치가 같은 상어는 인덱스로 비교해서 간선을 만들면 됨
  - 각 상어는 최대 2마리의 상어를 잡아 먹을 수 있을 때 살아남는 상어 수의 최솟값을 출력

  - 각 상어가 최대 1마리만 먹을 수 있다고 하자.
    - 살아 남은 상어의 수 = 경로 커버의 개수
    - 따라서 최소 경로 커버를 구하는 문제
    - 이분 매칭에서 매칭된 정점의 의미
      - 왼쪽 정점은 잡아 먹히는 상어, 오른쪽 정점은 잡아 먹는 상어를 의미함

  - 오른쪽 정점을 2개씩 만들면 2마리 먹는 문제를 해결할 수 있음
    - 매칭 안 된 왼쪽 정점의 개수를 구하면 됨

Applications

- BOJ 8898 스포츠 전문 채널 GSK
  - N개의 스포츠 경기가 열리고, i번째 경기는 S[i]에 시작해서 D[i] > 0 만큼 지속됨
  - i번째 경기에서 j번째 경기로 이동하는데 T[i][j] ≥ 0 만큼의 시간이 걸림
  - 한 리포터가 취재할 수 없는 가장 큰 경기의 부분 집합을 구하는 문제

  - 최대 반사슬을 구하는 문제

질문?

Applications

- BOJ 9522 직선 게임
  - 이차원 평면에 N개의 점이 있고, 두 명의 플레이어가 번갈아 가며 게임을 함
    - 첫 번째 플레이어는 N개의 점 중 하나를 통과하면서 좌표축에 평행한 직선을 그림
    - 그 다음부터 각 플레이어는 좌표축에 평행하면서 N개의 점 중
      상대방이 이전에 그린 직선 위의 점을 직선을 하나 그림
  - 더 이상 그릴 수 있는 직선이 없는 사람이 짐

  - 이거 설명은 안 할 건데 고민해 보고 모르면 슬라이드 다운 받아서 풀이 공부해 보세요.

Applications

- BOJ 9522 직선 게임
  - 점을 지나는 직선을 모두 그린 뒤, 직선을 하나씩 지우는 문제로 바꿔서 생각하자.

  - 가로 직선을 왼쪽 정점, 세로 직선을 오른쪽 정점, 교차하는 두 직선을 연결한 이분 그래프
    - 선공: 임의의 직선 삭제 — 왼쪽 정점 L_1에 토큰을 놓음
    - 후공: 선공이 제거한 직선과 교차하는 직선 중 하나 삭제 — L_1과 인접한 R_1으로 토큰 이동
    - 선공: 후공이 제거한 직선과 교차하는 직선 중 하나 삭제 — R_1과 인접한 L_2로 토큰 이동
    - … — …
    - 지울 수 있는 직선이 없으면 패배 — 토큰을 이동할 수 없으면 패배

    - 단, 토큰은 같은 정점을 여러 번 방문하지 않음

Applications

- BOJ 9522 직선 게임
  - 선공이 처음에 정점 v를 골랐다고 하자.

  - 만약 v를 포함하는 최대 매칭이 존재한다면 후공이 항상 이김
    - 후공은 선공이 어떻게 이동하더라도 항상 매칭을 따라 토큰을 이동할 수 있음
    - 따라서 후공 승

  - v를 포함하는 최대 매칭이 존재하지 않는다면 선공이 항상 이김
    - 후공은 첫 번째 턴에서 항상 최대 매칭에 포함된 정점으로 이동함
      - 최대 매칭에 포함되지 않은 정점이 존재한다면 매칭을 하나 더 만들 수 있으므로 최대 매칭이 아님
    - 이제부터 선공은 후공이 어떻게 이동하더라도 항상 매칭을 따라 토큰을 이동할 수 있음
    - 따라서 선공 승

Applications

- BOJ 9522 직선 게임
  - 완전 매칭(모든 정점이 참여하는 매칭)이 존재하면 모든 정점을 포함하는 최대 매칭이 존재
    - 따라서 후공 승
  - 완전 매칭이 존재하지 않으면 최소한 한 정점은 최대 매칭에 포함되지 않음
    - 따라서 선공 승

질문?