# 20.11.25 동아리 풀이

선린인터넷고등학교 소프트웨어과
30610 나정휘
https://JusticeHui.github.io

# 문제 목록

- A. 구슬 찾기 - 2003 KOI 중등부 1번
- B. 구간과 쿼리
- C. 플로이드에 오타가?
- D. 한 번 남았다
- E. League of ... Moloco
- F. Boggle - 2013 GCPC A번
- G. Stretch Rope - 2017 Google APAC Univ Test Round D D번
- H. Alternating Current - 2018 BOI Day1 A번

# 구슬 찾기 (1)

- A가 B보다, B가 C보, C가 A보다 무겁다 -> 불가능
  - 사이클이 없다.
  - DAG?

- A가 B보다 무겁다 -> A에서 B로 가는 간선을 만들자.
  - u가 v보다 무겁다 -> u에서 v로 가는 경로가 있다.
    - 방향 그래프를 만들고
    - Floyd-Warshall을 이용해 모든 정점 순서쌍에 대해 경로가 있는지 구하자.

# 구슬 찾기 (2)

- 어떤 구슬이 중간이 될 수 없는 경우
  - 그 구슬보다 무거운 구슬이 floor( (N+1)/2 )개 이상 또는
  - 그 구슬보다 가벼운 구슬이 floor( (N+1)/2 )개 이상

- http://boj.kr/0143257080e748789c3ed725e77b5cc0

# 구간과 쿼리 (1)

- BFS 연습 문제
- 부등호 주의해서 코딩하자.

- http://boj.kr/9db4ce4aec8b4a398282062474897c44

# 플로이드에 오타가? (1)

- 플로이드에서, `D(i, k) + D(k, j)`의 의미는?
  - `i`에서 출발, `k`를 경유, `j`에 도착
- `k = n`인 경우를 보지 않았다
  - `n`을 지나는 경로를 고려하지 않았다.
  - 최단경로가 `n`을 지나도록 만들면 된다.

- http://boj.kr/5f1f858c9e8a4e00a5095f701bc03ef1

# 한 번 남았다 (1)

- 벨만포드 알고리즘
  - 최단 경로를 구성하는 간선의 개수를 하나씩 늘려 나감
  - 간선 V-1개로 구성된 최단 경로를 만들면 된다.

- 모든 가중치가 -1인 직선 형태의 그래프를 만들면 된다.

- http://boj.kr/445517e1af7a476bb3bb610eaf02cfc3

# League of Overwatch at Moloco (1)

- 주어진 그래프가 이분 그래프인지 판별하는 문제

- http://boj.kr/862177c0da0a4355b92b0ad10da30a4e

# Boggle (1)

- Trie 만들고 DFS하면 된다.
- 구현 연습 문제

- http://boj.kr/c6b560cf1c6f42d6aca1b4d576d97f93

# Stretch Rope (1)

- 배낭의 무게가 [Ai, Bi]라는 구간으로 표현되는 0/1 냅색 문제
  - D(i, j) = i번째 물건까지, 무게 합이 j일 때 최소 가치

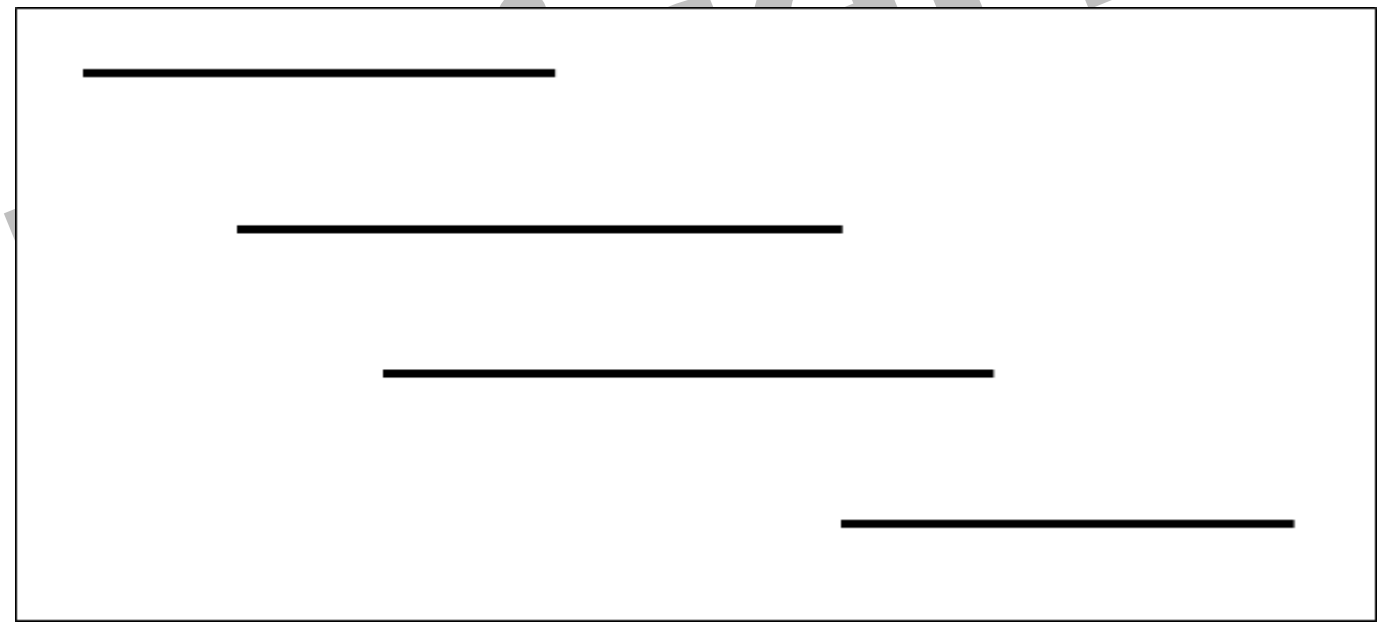
- 상태 전이
  - 무게 합이 j가 되기 위해서는
  - 기존 무게가 [j-Bi, j-Ai] 범위에 있어야 한다.
  - Deque 혹은 Segment Tree로 구간 최솟값을 구하면 된다.
  - 아래 코드는 Segment Tree로 구현함
- http://boj.kr/4934b0abd6f84a53a78097507d54b7e5

# Alternating Current (1)

- 어떤 선분 a가 b에 완전히 포함된다면
  - a와 b는 서로 다른 색을 배정해주면 된다.
  - a를 없애도 된다.
    - 다른 선분에 완전히 포함되는 선분을 제거하고
    - 어떤 선분에 의해 제거되었는지 저장하자.
    - KOI 2014 버스 노선처럼 Sliding Window로 구할 수 있다.
  - 코드에서 `calc_elimination` 함수 참고

# Alternating Current (2)

- 각 구간은 최소 한 개 이상의 선분으로 덮여 있다.
- 제거되지 않은 선분을 정렬해서 보면 아래와 같다.



# Alternating Current (3)

- 만약 정답이 **존재한다면**
- 아래 그림처럼 색을 교차해서 배정했을 때 정답이 된다.
  - 제거된 선분은 자신을 포함하는 선분과 다른 색을 배정하면 된다.

# Alternating Current (4)

- 제거되지 않은 선분의 개수를 K라고 하자.
  - K가 짝수라면 단순이 교차해서 배치하고 정답인지 확인하면 된다.
  - 홀수라면 K가지의 경우를 모두 확인해야 한다.
    - 01010 (1번 선분부터 시작)
    - 11010 (2번 선분부터 시작)
    - 10010 (3번 선분부터 시작)
    - 10110 (4번 선분부터 시작)
    - 10100 (5번 선분부터 시작)

    - k가지 경우를 Naïve하게 모두 확인하면 서브태스크 123을 해결해 55점을 받는다.

# Alternating Current (5)

- 5가지를 잘 보자.
  - 01010 (1번 선분부터 시작)
  - 11010 (2번 선분부터 시작)
  - 10010 (3번 선분부터 시작)
  - 10110 (4번 선분부터 시작)
  - 10100 (5번 선분부터 시작)
- i번째 비트열과 i+1번째 비트열은 i번째 비트만 다르다.
- K가지 경우를 모두 확인할 때 각 선분의 색을 한 번만 바꿔도 된다.

# Alternating Current (7)

- Seg Tree + Lazy Prop으로 구간의 최솟값을 관리하면
- 선분이 잘 덮여 있는지 빠르게 확인할 수 있다.

- O(M log M + M log N)에 문제를 풀 수 있다.
  - 필요 없는 선분 제거 O(M)
  - 선분 정렬 O(M log M)
  - 정답이 존재하는지 확인 O(M log N)

- http://boj.kr/46aeae8030df4a38b8b9af87cd8b0fc9