Sogang Programming Contest 2021

Host

Sogang ICPC Team

서강대학교 Sogang University

Official Solutions

2021 제17회 서강대학교 프로그래밍 대회 해설

2 → 5 PM @ Dasan 104/105
November 27th, Saturday

Sponsors

NAVER D2 STARTLINK

Master

문제		의도한 난이도	출제자
A	Ресторан	**Beginner**	박수현 shiftpsh
B	벼락치기	**Beginner**	이상원 gumgood
C	야바위 게임	**Easy**	이준석 semteo04
D	버스 노선 개편하기	**Easy**	이상원 gumgood
E	Player-based Team Distribution	**Medium**	강효규 djs100201, 이상원 gumgood
F	방탈출	**Medium**	조원빈 wbcho0504
G	알고리즘 과외	**Hard**	조원빈 wbcho0504
H	내가 몇 등이었지??	**Challenging**	이상원 gumgood

A. Ресторан

`string`, `implementation`
출제진 의도 – **Beginner**

- ✓ 제출 66번, 정답 27명 (정답률 40.909%)
- ✓ 처음 푼 사람: **강원호**, 3분
- ✓ 출제자: 박수현`shiftpsh`

A. Ресторан

문자열을 입력받아, 각 글자마다 적절한 대체 글자를 출력해주는 문제입니다. 크게 두 가지 정도의 구현 방식이 있겠습니다.

- ✓ 글자를 하나 입력받을 때마다 대체 글자를 출력해주는 방식
- ✓ `std::replace` 등의 라이브러리 메서드를 활용하는 방식

A. Ресторан

대회에서 쓸 수 있는 언어들에 존재하는 `replace` 함수들은...

C++ `std::replace(str.begin(), str.end(), 'H', 'n')`

Python `str.replace("H", "n")`

Java `str.replace("H", "n")`

Kotlin `str.replace("H", "n")`

가 있습니다. 모든 글자에 대해 반복해 주면 됩니다.

B. 벼락치기

`arithmetic`, `simulation`
출제진 의도 – **Beginner**

- ✓ 제출 140번, 정답 26명 (정답률 18.571%)
- ✓ 처음 푼 사람: **박준서**, 9분
- ✓ 출제자: 이상원`gumgood`

B. 벼락치기

벼락치기 공부법으로 N 개의 챕터를 순서대로 공부할 때, 절반 이상 공부한 챕터의 개수를 구하는 문제입니다.

- ✓ 30 분마다 공부하던 챕터를 멈추고 넘어가는 규칙에 따라 시뮬레이션합시다.
- ✓ 각 챕터에 대해 공부한 시간이 $\lceil T_i/2 \rceil$ 이상인 경우를 세주면 됩니다.
- ✓ T_i 가 홀수인 경우에 주의합시다.

C. 야바위 게임

`graphs`, `graph_traversal`, `bfs`
출제진 의도 – **Easy**

- ✓ 제출 33번, 정답 5명 (정답률 5.556%)
- ✓ 처음 푼 사람: **박준서**, 52분
- ✓ 출제자: 이준석`semteo04`

C. 야바위 게임

처음에 큐 안에 시작 정점을 넣어둡니다. 여기에는 현재 단계에 공이 있을 수 있는 정점을 저장합니다.

- ✓ 큐를 돌면서 해당 정점에 인접한 정점을 체크해 줍니다.
- ✓ 큐를 비우고 체크된 정점을 큐에 넣어 줍니다.
- ✓ 체크된 정점들에서 체크를 전부 풀어 줍니다.

위 과정을 컵이 움직인 횟수만큼 반복하면 됩니다.

D. 버스 노선 개편하기

`sorting`, `sweeping`
출제진 의도 – **Easy**

- 제출 64번, 정답 14명 (정답률 21.875%)
- 처음 푼 사람: **박준서**, 25분
- 출제자: 이상원`gumgood`

D. 버스 노선 개편하기

비어있는 일직선 도로에 버스 노선을 하나식 추가해봅시다.

- ✓ 각 버스 노선을 추가할 때 겹치는 버스 노선이 있다면 해당 버스 노선과 합칩니다.
- ✓ 그렇지 않다면 새로운 버스 노선으로 만듭니다.

어떤 버스 노선의 구간과 겹치는 버스 노선을 찾는 다양한 방법이 있습니다. 그 중 가장 간단하다고 생각되는 풀이를 소개하겠습니다.

D. 버스 노선 개편하기

다음과 같은 관찰을 해봅시다.

- ✓ 구간을 어떠한 순서로 합치더라도 개편 후 버스 노선의 구간은 같습니다.
- ✓ 개편 후의 버스 노선의 가격 또한 순서에 영향을 받지 않습니다.
  개편 후의 버스 노선에 포함된 버스 노선들 중 가장 낮은 가격을 따르고 이는 순서와 무관합니다.

따라서 버스 노선을 임의의 순서로 합치더라도 답은 달라지지 않습니다.

D. 버스 노선 개편하기

S가 작은 순으로 버스 노선을 추가해봅시다.

- ✓ S가 i번째로 작은 버스 노선의 구간을 $[S_i, E_i]$라고 합시다.
- ✓ $1 \sim i-1$번째까지 추가된 새로운 버스 노선에 i번째 버스 노선을 추가해보겠습니다.
  - (새로운 버스 노선들의 시작 구간 S) $\leq S_i$를 항상 만족합니다.
  - 따라서 위치 S_i에서 겹치는지만 확인하면 됩니다.
  - 새로운 버스 노선들은 서로 겹치지 않기 때문에 (가장 오른쪽 버스 노선의 E)와 S_i만 비교하면 됩니다.
    - (가장 오른쪽 버스 노선의 E) $\geq S_i$이면, 가장 오른쪽 버스 노선에 합칩니다.
    - (가장 오른쪽 버스 노선의 E) < S_i이면, 새로운 버스 노선으로 만듭니다.

D. 버스 노선 개편하기

- ✓ Sweeping을 통해 $O(n)$에 구현할 수 있습니다.

원래 다음과 같이 출제하려 했으나 난이도를 낮추게 되었습니다. 해결방법을 한 번 생각해보세요!

- ✓ 두 버스 노선을 새 노선으로 합칠 때, 두 구간이 더 짧은 노선의 금액을 따른다.
- ✓ 두 구간의 길이가 같은 경우, 더 낮은 금액을 따른다.
- ✓ 이때, 개편 후 버스 노선의 금액의 합을 최소로 만들어 보자.

E. Player-based Team Distribution

`greedy`
출제진 의도 – **Medium**

- ✓ 제출 95번, 정답 10명 (정답률 10.526%)
- ✓ 처음 푼 사람: **손기령**, 24분
- ✓ 출제자: 강효규 `djs100201`, 이상원 `gumgood`

E. Player-based Team Distribution

다음과 같은 전략으로 문제를 해결할 수 있습니다.

- ✓ $a_i \geq 0$인 플레이어를 전부 **하나의 큰 팀**으로 합칩니다.
- ✓ $a_i < 0$인 플레이어는 혼자 한 팀으로 만듭니다.
- ✓ 그 뒤로 $a_i < 0$인 플레이어 중 a_i가 큰 플레이어부터, 한 명씩 **하나의 큰 팀**에 합칩니다. 넣을 때마다 플레이어의 점수 합을 $\mathcal{O}(1)$에 구할 수 있습니다.
- ✓ 이를 이용해 최댓값을 계속 갱신해나가면 답을 구할 수 있습니다.

Greedy한 배치과정의 정당성을 증명해보겠습니다.

- ✓ 먼저, 팀원들의 점수 합이 0 이상인 팀이 1개 이하인 최적해가 존재함을 보입시다.
  - 팀원들의 점수 합이 x, 인원 수가 y인 팀을 (x, y)로 나타내겠습니다.
  - 점수 합이 0 이상인 팀이 2개 이상인 최적해가 있다고 가정합시다. 점수 합이 0 이상인 두 팀 $(a, b), (c, d)$를 합치면 $(a + c, b + d)$인 팀이 됩니다. 이때 점수 합은 $ab + cd \leq (a + c)(b + d)$ 이므로 팀의 수를 하나 줄이면서 최적해를 유지하게 됩니다. 이를 통해 점수 합이 0 이상인 팀을 1개로 만들 수 있습니다.

E. Player-based Team Distribution

✓ 이제 최적해에서는 팀의 점수 합이 0 미만인 팀에는 $a_i \geq 0$인 팀원이 없음을 보입시다.

- 팀의 접수 합이 0 미만인 팀이 있는 최적해가 있다고 가정합시다.
- 이 때, $a_i \geq 0$인 팀원들의 a_i 합을 a, 인원 수를 n이라하고 $a_i < 0$인 팀원들의 a_i 합을 b, 인원수을 m이라 합시다. $(a + b < 0)$
- 이 팀의 점수 합은 $(a + b)(n + m)$입니다. $ai \geq 0$인 팀원을 전부 모으고 나머지 팀원들은 혼자 한 팀으로 새롭게 분배해 봅시다.
  즉, $(a, n), (b_1, 1), (b_2, 1), (b_3, 1), \cdots, (b_m, 1)$으로 플레이어들을 분배합니다.
- 그러면 이들의 점수 합은 $an + b$가 되고 $b - am - bm - bn$ 만큼 변했습니다.
- 식을 했을 때 $(1 - m - n)b - ma > (1 - m - n)b + mb = (1 - n)b \geq 0$ 이므로 팀의 점수합이 증가합니다. 이는 최적해라는 전제에 모순입니다.

정리하면

- ✓ $a_i \geq 0$ 인 플에이어가 한 팀에 있는 최적해가 존재합니다.
- ✓ $a_i < 0$ 인 플레이어는 모두 혼자 한 팀으로 있는 것이 자명합니다.
- ✓ 현재 상태에서 점수를 늘릴 수 있는 방법은 $a_i \geq 0$ 인 플레이어가 모여있는 팀에 $a_i < 0$ 인 플레이어를 팀원으로 추가하는 방법 뿐입니다.
- ✓ $a_i < 0$ 인 플레이어 중 a_i 이 큰 순서대로 한 명씩 넣어보면서 전체 팀 점수 합을 확인하면 됩니다.

F. 방탈출

`mst`

출제진 의도 – **Medium**

- ✓ 제출 11번, 정답 3명 (정답률 27.273%)
- ✓ 처음 푼 사람: **박준서**, 88분
- ✓ 출제자: 조원빈`wbcho0504`

F. 방탈출

이 문제는 그래프 모델링을 통해 해결할 수 있습니다.

- ✓ 각 방을 $1 \sim N$ 번 노드로, 출구를 0 번 노드로 둡니다.
- ✓ 그러면 워프는 a_i 번 노드와 b_i 번노드를 잇는 간선으로, 비상탈출구는 0 번 노드와 i 번 노드를 잇는 간선으로, 설치하는 데 드는 시간은 간선의 가중치로 볼 수 있습니다.

F. 방탈출

- ✓ 다만, 모든 간선을 사용할 필요는 없습니다.
- ✓ 적당히 간선을 지워, 다음 두 조건을 만족하면서 가중치 합이 최소가 되도록 합니다.
  1. 모든 노드들이 연결되어 있다.
  2. 사이클이 존재하지 않는다

F. 방탈출

- ✓ 모든 간선을 사용할 필요는 없습니다.
- ✓ 적당히 간선을 지워, 다음 두 조건을 만족하면서 가중치 합이 최소가 되도록 합니다.
  1. 모든 노드들이 연결되어 있다.
     - ▶ 모든 노드는 출구를 의미하는 0번 노드와 연결되어있어야 합니다.
     - ▶ 이는 모든 노드가 연결되어 있어야 한다는 위의 조건과 동치입니다.
  2. 사이클이 존재하지 않는다

F. 방탈출

- ✓ 모든 간선을 사용할 필요는 없습니다.
- ✓ 적당히 간선을 지워, 다음 두 조건을 만족하면서 가중치 합이 최소가 되도록 합니다.
  1. 모든 노드들이 연결되어 있다.
     - ▶ 모든 노드는 출구를 의미하는 0번 노드와 연결되어있어야 합니다.
     - ▶ 이는 모든 노드가 연결되어 있어야 한다는 위의 조건과 동치입니다.
  2. 사이클이 존재하지 않는다
     - ▶ 사이클이 존재하는 경우 간선 하나를 지워도 모든 정점들은 똑같이 연결된 상태입니다.
     - ▶ 즉, 간선 하나를 지워 가중치 합을 더 작게 만들 수 있습니다.
     - ▶ 따라서 최적의 경우에서는 사이클이 존재하지 않습니다.

F. 방탈출

- 위의 두 조건을 모두 만족하는 그래프의 형태는 트리입니다.
- 따라서 최소 신장 트리(Minimum Spanning Tree)를 구함으로써 문제를 해결할 수 있습니다.
- 최소 신장 트리는 크루스칼 알고리즘, 프림 알고리즘 등을 이용해 구할 수 있습니다.

G. 알고리즘 과외

`segment tree`, `sweeping`

출제진 의도 – **Hard**

- ✓ 제출 13번, 정답 0명 (정답률 0.000%)
- ✓ 출제자: 조원빈 `wbcho0504`

G. 알고리즘 과외

- N 명의 학생은 각각 레이팅 a_i 를 가지고 있고, 과외를 하기 위해 두 학생을 뽑을 때 레이팅 차의 최댓값을 구하는 문제입니다.
- 단, i 번 학생은 자신과의 번호 차이가 l_i 이상 r_i 이하인 학생과만 과외를 할 수 있습니다.

G. 알고리즘 과외

- ✓ 과외를 하는 두 학생의 번호를 x, y 라고 합시다. $(x < y)$
- ✓ y 를 고정하고 생각해보면, x 는 $y - r_y \leq x \leq y - l_y$ 를 만족해야 합니다.
- ✓ 또한 레이팅의 차가 최대가 되기 위해서는 x 번 학생의 레이팅이 최소 또는 최대가 되어야 합니다.
- ✓ 따라서 y 를 고정했을 때 레이팅 차의 최댓값은 다음의 식을 계산하여 구할 수 있습니다.

$$\max\left\{\left[\max_{y-r_y\leq i\leq y-l_y} a_i\right] - a_y, a_y - \left[\min_{y-r_y\leq i\leq y-l_y} a_i\right]\right\}$$

- ✓ 특정 구간의 최댓값과 최솟값은 segment tree를 통해 $\mathcal{O}(\log n)$ 에 구할 수 있습니다.

- 그런데 위의 풀이는 y가 $x + l_x \leq y \leq x + r_x$를 만족하는지 확인하지 못합니다.
- 따라서 이러한 문제점을 해결하기 위해 스위핑 기법을 사용합니다.

G. 알고리즘 과외

- ✓ y를 1 부터 n까지 증가시키면서 y를 고정했을 때 레이팅 차의 최댓값을 계산합니다.
- ✓ 처음에 모든 segment tree의 노드는 ∞ 또는 $-\infty$로 초기화합니다.
- ✓ y보다 작은 임의의 x에 대하여 $y = x + l$을 만족하면 segment tree의 x번 노드를 a_x로 업데이트합니다.
- ✓ y보다 작은 임의의 x에 대하여 $y = x + r + 1$을 만족하면 segment tree의 x번 노드를 ∞ 또는 $-\infty$로 업데이트합니다.
- ✓ 이제는 y가 $x + l_x \leq y \leq x + r_x$를 만족하지 못할 경우 ∞ 또는 $-\infty$로 처리되어 고려하지 않게 됩니다.
- ✓ 업데이트는 최대 $2n$번 일어나고, 업데이트 하는 데 $\mathcal{O}(\log n)$이 걸리므로, 총 시간복잡도는 $\mathcal{O}(n \log n)$이 됩니다.

H. 내가 몇 등이었지??

`geometry`, `half_plane_intersection`
출제진 의도 – **Challenging**

- ✓ 제출 4번, 정답 0명 (정답률 0.000%)
- ✓ 출제자: 이상원`gumgood`

H. 내가 몇 등이었지??

- i 번째 학생이 j 번째 학생보다 최종 성적이 높은 경우, 다음 부등식이 성립합니다.

$$Ap_i + Bq_i + Cr_i > Ap_j + Bq_j + Cr_j$$

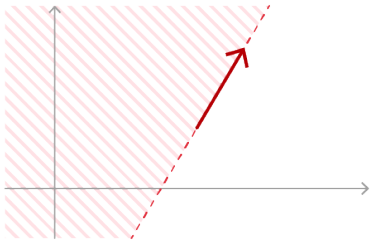
$$\frac{A}{C}(p_i - p_j) + \frac{B}{C}(q_i - q_j) + r_i - r_j > 0 \quad (\because C > 0)$$

- $X = \frac{A}{C}, Y = \frac{B}{C}$ 라고 하면 다음과 같은 일차 부등식이 됩니다.

$$(p_i - p_j)X + (q_i - q_j)Y + (r_i - r_j) > 0$$

H. 내가 몇 등이었지??

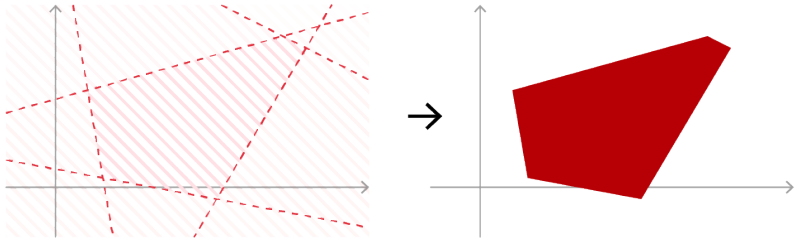
- 이를 $X - Y$ 평면에 나타내면 다음과 같습니다.



- $X - Y$ 평면을 직선 $(p_i - p_j)X + (q_i - q_j)Y + (r_i - r_j) = 0$ 을 기준으로 반으로 나누었을 때 한 쪽 부분에 해당하게 됩니다. 이를 **반평면(half-plane)**이라 합니다.
- 즉, 부등식의 영역인 반평면이 X, Y 가 존재할 수 있는 영역입니다.

H. 내가 몇 등이었지??

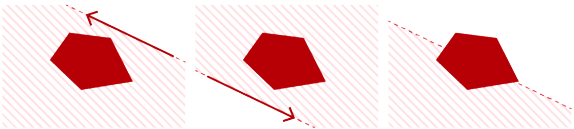
- ✓ 이를 확장해 봅시다. 두 학생의 대소 관계가 M 개 주어지고, M 개의 반평면이 생기게 됩니다.



- ✓ M 개의 부등식을 모두 만족하는 X, Y 는 모든 M 개의 반평면 위에 있어야 합니다.
- ✓ 즉, 모든 반평면이 겹치는 영역인 **반평면 교집합 (half-plane intersection)** 이 X, Y 가 존재할 수 있는 영역입니다.

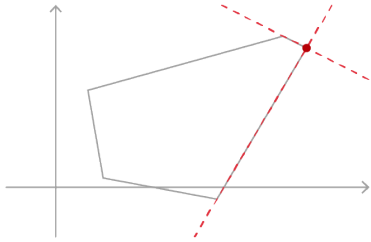
H. 내가 몇 등이었지??

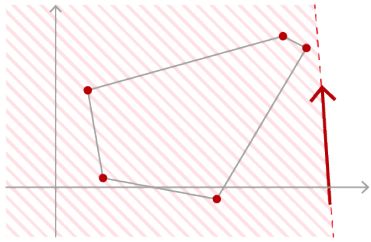
- ✓ M 개의 부등식으로 이뤄진 반평면 교집합은 고정되어 있습니다.
- ✓ 각 질문에 해당하는 부등식과 M 개의 부등식을 모두 만족하는 X, Y 가 있는지 확인하면 됩니다.
- ✓ 이는 부등식에 맞는 반평면이 고정된 반평면 교집합을 완전히 포함하는 것과 같습니다.
  - i 번째 학생보다 j 번째 학생의 최종성적이 높다고 가정하고 반평면을 만들어 확인합니다.
  - i 번째 학생보다 j 번째 학생의 최종성적이 낮다고 가정하고 반평면을 만들어 확인합니다.
  - 둘 다 성립하지 않는 경우, 결정할 수 없습니다.



H. 내가 몇 등이었지??

- ✓ 반평면이 어떤 반평면 교집합을 완전히 포함하는지 확인하는 것은 어렵지 않습니다.





- ✓ 반평면 교집합의 모든 꼭지점이 반평면 위에 있으면 완전히 포함됩니다.
- ✓ 이때 꼭지점은 반평면 교집합의 변에 해당하는 두 직선의 교점입니다.

H. 내가 몇 등이었지??

- ✓ 반평면 교집합은 $O(M)$ 개의 꼭짓점을 가질 수 있으므로 쿼리당 $O(M)$ 에 처리할 수 있습니다.
- ✓ 반평면 교집합을 구하는 알고리즘은 $O(N^3), O(N^2), O(N \log N)$ 과 같이 다양합니다.
- ✓ 이 문제는 반평면 교집합을 $O(M^2)$ 이하에 구하면 전체 $O(M^2 + QM)$ 에 해결할 수 있습니다.
- ✓ $X > 0, Y > 0$ 도 빠지지 않도록 유의합시다.

각도 정렬과 Binary Search 를 통해 $O((M + Q) \log M)$ 에 해결할 수도 있습니다.

Champion

문제		의도한 난이도	출제자
A	문자열 압축 해제	**Beginner**	이상원 gumgood
B	와드	**Medium**	박수현 shiftpsh
C	카드컨트롤	**Medium**	이준석 semteo04
D	방문 판매	**Medium**	이선호 glanceyes
E	Leader-based Team Distribution	**Hard**	강효규 djs100201
F	던전 릴레이	**Hard**	이선호 glanceyes
G	카드 잘 섞기	**Challenging**	이준석 semteo04
H	연결 요소와 쿼리	**Challenging**	박수현 shiftpsh, 이혜아 ho94949

A. 문자열 압축 해제

`string`, `implementation`
출제진 의도 – **Beginner**

- ✓ 제출 66번, 정답 22명 (정답률 27.273%)
- ✓ 처음 푼 사람: **전해성**, 2분
- ✓ 출제자: 이상원`gumgood`

A. 문자열 압축 해제

대문자로 된 압축된 문자열이 주어지면 소문자 문자열 패턴으로 압축을 해제한 뒤, S 번째에서 E 번째 문자까지 출력하는 문제입니다.

- ✓ 압축되기 전 문자열의 최대 길이는
  (각 소문자 문자열 패턴의 최대 길이) $\times$ (압축된 문자열 최대 길이) $= 10^6$ 입니다.
- ✓ 따라서 단순하게 대문자를 하나식 대체해도 충분히 시간 내에 해결 할 수 있습니다.

원래 압축되기 전 문자열 최대 길이는 10^{12} 이면서 $E - S < 10^6$ 인 조건으로 출제하려했으나 난이도를 낮추게 되었습니다. 해결 방법을 한 번 생각해보세요!

B. 와드

`simulation`, `graphs`, `graph_traversal`
출제진 의도 – **Medium**

- ✓ 제출 36번, 정답 13명 (정답률 36.111%)
- ✓ 처음 푼 사람: **김정모**, 16분
- ✓ 출제자: 박수현`shiftpsh`

B. 와드

‘코딩 테스트에 나온다면 약간 어려울 것 같다’ 정도를 의도하고 출제했습니다.

- ✓ 와드를 놓으면 현재 속한 연결 요소의 모든 칸이 시야에 들어옵니다.
- ✓ 연결 요소를 찾으려면 BFS 또는 DFS를 한 번 돌려 주면 됩니다. $\mathcal{O}(RC)$가 걸립니다.
- ✓ 그런데 와드가 최대 $200\,000$개까지 놓일 수 있는 것을 감안하면, $200\,000 \times 1\,000^2 = 2 \times 10^{11}$으로, 연결 요소를 매번 찾아준다면 1초 안에 돌지 않습니다.

B. 와드

- ✓ BFS 또는 DFS를 돌릴 때 방문한 정점들을 기록하는 배열 — 예를 들어, `vis` 등 — 을 활용해 봅시다.
- ✓ 와드를 놓을 곳이 이전에 BFS 또는 DFS로 이미 방문한 곳이라면, 그 연결 요소 안의 칸들은 이미 시야로 확보했다는 뜻이 됩니다. 따라서 무시해줄 수 있습니다.

B. 와드

- ✓ 혹은 와드들을 맨 마지막에 처리해줄 수도 있습니다.
- ✓ 여행 기록을 처리하는 중엔 큐에 와드들의 좌표들만을 넣어 두고, 여행 기록 처리가 끝난 후에 BFS를 수행합니다.
- ✓ 시작 정점이 여러 개인 그래프 탐색이 됩니다.

마지막에 현재 위치와 현재 위치 상하좌우의 칸도 시야에 확보해 줘야 함을 잊지 말고 구현해 주세요.

C. 카드컨트롤

`graphs`, `graph_traversal`, `bfs`

출제진 의도 – **Medium**

- ✓ 제출 53번, 정답 8명 (정답률 15.094%)
- ✓ 처음 푼 사람: **김정모**, 36분
- ✓ 출제자: 이준석 `semteo04`

카드는 총 10장으로

$$\binom{10}{5} = 252$$

개의 상태가 있습니다. 이 문제를 그래프 문제로 바꿔봅시다.

- ✓ 각각의 상태를 정점으로 생각합니다.
- ✓ 어떤 상태에서 한 번의 조작으로 다른 상태로 바뀔 수 있다면 두 상태를 나타내는 정점을 연결합니다.

어떤 정점이 이기는 정점인지 모두 체크합니다.
처음 상태에 해당하는 정점에서 가장 가까운 이기는 정점까지의 거리를 출력하면 됩니다.
이것은 BFS를 이용하면 해결할 수 있습니다.

D. 방문 판매

`dp`, `backtracking`

출제진 의도 – **Medium**

- ✓ 제출 21번, 정답 4명 (정답률 19.048%)
- ✓ 처음 푼 사람: **전해성**, 28분
- ✓ 출제자: 이선호`glanceyes`

D. 방문 판매

- ✓ Knapsack DP를 이용하여 최소 몇 명의 고객에게 제품을 판매해서 주어진 할당량을 채울 수 있는지 구할 수 있습니다.
  - $dp[i][j][k]$: i번 고객까지 방문하여 제품 ***X***, ***Y***를 각각 j, k만큼 팔았을 때 판매에 성공할 수 있는 최소 고객 수

- ✓ i번 고객에게 판매에 성공했을 때 팔 수 있는 제품 ***X***, ***Y***의 양을 각각 x_i, y_i라고 하면

$$dp[i][j][k] = min(dp[i-1][j][k], dp[i-1][j-x_i][k-y_i]+1)$$

D. 방문 판매

- ✓ for-loop를 이용해 1번 고객부터 N번 고객까지 차례대로 탐색하면서 $j \geq X$ 이고 $k \geq Y$ 를 만족하는 dp배열의 최소 원소 값과 그때의 고객 번호를 구합니다.
- ✓ 최종 시간복잡도는 $\mathcal{O}(NXY)$ 입니다.

- ✓ 시간복잡도가 $\mathcal{O}(N^2XY)$ 인 풀이로도 해결할 수 있습니다.
  - $dp[i][j][k][l]$: i번 고객까지 방문하여 j명의 고객에게 판매를 성공했을 때 제품 X, Y를 각각 k, l만큼 파는 것이 가능한지의 여부

$$dp[i][j][k][l] = dp[i-1][j][k][l] \cup dp[i-1][j-1][k-x_i][l-y_i]$$

E. Leader-based Team Distribution

`greedy`, `priority_queue`
출제진 의도 – **Hard**

- ✓ 제출 3번, 정답 0명 (정답률 0.000%)
- ✓ 출제자: 강효규 `djs100201`

E. Leader-based Team Distribution

어려운 그리디 문제입니다.

- ✓ greedy 하게 접근하지 못하면 헤맬 수 있습니다.
- ✓ L_i 가 큰 순서대로, 같다면 P_i 가 큰 순서대로 플레이어를 정렬합니다.
- ✓ T_i 는 내림차순으로 정렬해줍니다.

E. Leader-based Team Distribution

팀을 분배한다기 보다는, 리더를 고른다고 생각해봅시다.

- ✓ L_i 가 가장 큰 사람은 반드시 리더가 되어야 합니다.
- ✓ 정렬되어 있는 순서대로 그 뒤로 T_1 명의 플레이어를 리더 후보군에 넣어줍니다.
- ✓ 지금 후보군에 있는 플레이어중 반드시 한명은 리더가 되어야 합니다. P_i 가 가장 큰 플레이어를 리더로 뽑아주면 됩니다.

설명한 과정을 반복해주면 됩니다.

- ✓ 현재 다음 순서부터 T_i 명의 플레이어를 후보군에 넣어주고 후보군에서 P_i 가 가장 큰 플레이러를 리더로 선택하는 과정을 $M-1$ 번 반복해주면 됩니다.
- ✓ T_i 를 내림차순으로 정렬해주는 이유는 뽑는 것을 미룰수록 이득이기 때문입니다.
- ✓ 플레이어를 추가하면서, 그 중 P_i 가 가장 큰 사람을 리더로 골라주는 작업은 priority queue 로 구현할 수 있습니다.

F. 던전 릴레이

`segtree`, `lazyprop`, `prefix_sum`
출제진 의도 – **Hard**

- 제출 7번, 정답 2명 (정답률 28.571%)
- 처음 푼 사람: **전해성**gumgood, 88분
- 출제자: 이선호glanceyes

F. 던전 릴레이

- ✓ 우선 던전 번호의 크기와 난이도가 서로 비례함이 보장되지 않으므로 난이도 순으로 정렬했을 때의 던전 번호를 새로 구해야 합니다. 이는 좌표 압축 또는 정렬 후 이진 탐색으로 가능합니다.

- ✓ 최대 던전 수 10만과 최대 쿼리 수 10만 개를 고려했을 때, segment tree 등의 자료구조를 사용하여 해결할 수 있습니다.
- ✓ 해설의 편의를 위하여 "i번째 던전"은 "i번째로 난이도가 낮은 던전"으로 정의하겠습니다. (i번째 던전 $\neq i$번 던전)

F. 던전 릴레이

- 누적합을 사용하여 다음과 같이 배열을 정의해봅시다.
  - $sum[i]$: 1 번째 던전부터 $i-1$ 번째 던전까지 차례대로 깨고, i 번째 던전에 입장료를 지불한 후 가지고 있을 캐시 금액
- 구간 $[i, j]$ 에 관하여 segment tree 배열을 다음과 같이 정의해봅시다. 여기서는 릴레이 도중 입장료가 부족하여 실패하는 경우는 무시하고, 설정한 마지막 던전까지 무조건 깬다고 가정합니다.
  - $segtree_{i,j}$: 1 번째 던전부터 시작하여 x 번째 던전의 직전 던전까지 차례대로 깨고, x 번째 던전에서 입장료를 지불한 후 가지고 있을 캐시 금액의 최솟값 $(i \leq x \leq j)$

F. 던전 릴레이

- ✓ 1로 시작하는 쿼리는 다음과 같이 처리합니다.
  - 입력 받은 난이도를 가지고 릴레이를 시작하고 끝내야 하는 던전의 서수 i, j를 각각 구합니다.
  - 다음 조건을 만족하면 i번째 던전부터 j번째 던전까지 깨는 릴레이를 성공적으로 마칠 수 있습니다.

$$segtree_{i,j} - (sum[i-1] + f[i-1]) + c \geq 0$$

  - 릴레이를 마친 후 가지고 있을 캐시 금액은 다음과 같습니다.

$$sum[j] + f[j] - (sum[i-1] + f[i-1]) + c$$

F. 던전 릴레이

- ✓ 2 또는 3으로 시작하는 쿼리는 다음과 같이 처리합니다.
  - 입력 받은 던전 번호 x를 통해 구한 i번째 던전의 입장료(보상 캐시)를 수정하면 i번째($i+1$번째) 던전부터 N번째 던전 범위에 속하는 segment tree 원소 값이 변하므로 lazy propagation을 사용합니다.
  - i번째 던전의 입장료를 $fee[i]$에서 v로 바꾸고, i번째 던전부터 N번째 던전 범위에 속한 segment tree 원소 값에 $fee[i]-v$를 더합니다.
  - i번째 던전의 보상 캐시를 $reward[i]$에서 v로 바꾸고, $i+1$번째 던전부터 N번째 던전 범위에 속한 segment tree 원소 값에 $v-reward[i]$를 더합니다.

- ✓ 계산의 용이함을 위해 segment tree의 인덱스를 두 배로 늘려서 i번째 던전에 입장료를 지불할 때와 보상 캐시를 받을 때인 두 가지 경우로 분리할 수도 있습니다.

G. 카드 잘 섞기

`ad_hoc`

출제진 의도 – **Challenging**

- ✓ 제출 12번, 정답 0명 (정답률 0.000%)
- ✓ 출제자: 이준석 `semteo04`

G. 카드 잘 섞기

홀수인 경우를 먼저 해결해 봅시다.
카드가 5 장이라고 한다면

1 2 3 4 5

와 같이 배치됩니다. 인덱스는 0 부터 시작합니다.
각각의 인덱스를 2 배 하면

1 0 2 0 3 0 4 0 5 0

이 됩니다. 보기 쉽게 하기 위해서 카드 사이와 마지막에 0을 넣었습니다.

G. 카드 잘 섞기

이제 오른쪽 절반에 해당하는 카드를 왼쪽 절반 사이사이에 넣으면 됩니다.
이것은 오른쪽 절반에 해당하는 카드의 인덱스를 N 만큼 빼서 해결할 수 있습니다.

1 0 2 0 3
0 4 0 5 0

그럼 한 번 카드를 섞을 때 카드의 인덱스는

$$2x \bmod N$$

가 됩니다. (x 는 처음 카드의 인덱스)

이 동작을 셔플 횟수만큼 반복하면 됩니다. N 으로 나눈 나머지는, 마지막에 1 번만 해도 되므로,

$$2^k x \bmod N$$

가 됩니다. (k 는 셔플 횟수)
하지만 짝수일 때는 문제가 생깁니다. 앞의 동작을 그대로 따라하면

1 0 2 0 3 0
4 0 5 0 6 0

카드끼리 겹쳐서 합할 수 없습니다.

G. 카드 잘 섞기

카드가 10장이 있을 때 1 3 7 1 쿼리와 1 3 8 1 쿼리를 각각 수행해 봅시다.

$$1\,3\,7\,1\colon 1\,2\,3\,4\,5\,6\,7\,8\,9\,10 \to 1\,2\,3\,6\,4\,7\,5\,8\,9\,10$$
$$1\,3\,8\,1\colon 1\,2\,3\,4\,5\,6\,7\,8\,9\,10 \to 1\,2\,3\,6\,4\,7\,5\,8\,9\,10$$

두 동작이 완벽히 같음을 알 수 있습니다. 따라서 N이 짝수인 경우 마지막 카드를 빼서 홀수일 때와 같이 계산하면 됩니다.
이제 카드의 위치를 완벽히 계산할 수 있으니 각 쿼리마다 O카드의 위치를 따라가면 해결할 수 있습니다.

H. 연결 요소와 쿼리

`segtree`, `case_work`

출제진 의도 – **Challenging**

- ✓ 제출 0번, 정답 0명 (정답률 0.000%)
- ✓ 출제자: 박수현 `shiftpsh`, 이혜아 `ho94949`

$K = 1$일 때 부터 문제를 해결해 봅시다.

- ✓ 어떤 구간 A에 대해서 다음 정보를 알고 있다고 합시다.
  - A_L: 가장 왼쪽 원소를 포함하는 연결 요소의 노드의 값의 합의 최댓값.
  - A_R: 가장 오른쪽 원소를 포함하는 연결 요소의 노드의 값의 합의 최댓값.
  - A_S: 모든 연결 요소의 노드의 값의 합.
  - A_V: 가능한 연결 요소의 노드의 값의 합의 최댓값.
- ✓ A가 하나의 원소 v를 포함하면, $A_L = A_R = A_S = A_V = v$ 입니다.

H. 연결 요소와 쿼리

- ✓ 두 구간 A, B를 차례로 합한 구간 $A \oplus B$에 대한 정보는 다음과 같이 계산됩니다:
  - $(A \oplus B)_L = \max(A_L, A_S + B_L)$
  - $(A \oplus B)_R = \max(A_R + B_S, B_R)$
  - $(A \oplus B)_S = A_S + B_S$
  - $(A \oplus B)_V = \max(A_V, B_V, A_L + B_R)$
- ✓ 가장 왼쪽 원소를 포함하는 연결 요소는 B의 원소를 아예 포함하지 않거나, A의 원소를 전부 포함하고 B의 가장 왼쪽 원소를 포함합니다. (반대도 비슷하게 계산됩니다.)
- ✓ 노드의 값의 합이 최대인 연결 요소는, A의 원소만 포함하거나, B의 원소만 포함하거나, A의 가장 오른쪽 원소와 B의 가장 왼쪽 원소를 모두 포함합니다.
- ✓ 세그먼트 트리를 이용해 갱신과 부분합을 처리합니다.

이제 $K \geq 2$ 일 때로 일반화해 봅시다.

- ✓ $A \oplus B$ 에서 노드의 값의 합이 최대인 연결 요소는, A 와 B 를 차례로 이었을 때 모종의 방식으로 연결 요소가 하나로 이어지는 경우만 생각해야 합니다.
- ✓ 어떤 구간의 내부 연결이 어떤 식으로 되어 있든, 외부로 나와있는 왼쪽 끝 K 개의 노드와 오른쪽 끝 K 개의 노드, 총 $2K$ 개의 연결 방식이 같으면, 왼쪽과 오른쪽에 특정한 노드들이 추가 되었을 때 전체가 연결되어 있는지 아닌지의 여부는 동일합니다.
- ✓ $K = 1$ 일 때는 다음과 같습니다:
  - $L(R)$: 왼쪽(혹은 오른쪽) 끝이 연결 성분에 포함되어 있고, 반대쪽은 포함되어 있지 않음.
  - S: 왼쪽과 오른쪽이 같은 연결 성분에 속함
  - V: 왼쪽과 오른쪽 끝이 외부로 나와있는지 여부를 고려하지 않음.

H. 연결 요소와 쿼리

- 구간에서 왼쪽 K 개의 노드, 오른쪽 K 개의 노드를 연결 방식에 대해 노드의 합의 최댓값을 관리합니다. 구간 A 의 연결 방식이 c 인 경우의 최댓값을 A_c 라 표시합시다.
- 또한, 왼쪽의 연결 방식이 l, 오른쪽의 연결 방식이 r 인 경우 구간을 합친 이후 전체 구간의 연결 방식을 $l \times r$ 이라고 합시다. 두 구간의 합의 정보는 다음과 같은 방식으로 구할 수 있습니다:
  - $(A \oplus B)_c = \max_{c=l\times r} (A_l + B_r)$
- 연결 방식은, 왼쪽과 오른쪽에 적당한 구간이 추가되었을 때 하나의 연결 요소를 만들 수 있는 경우만 생각합니다.
  - 위 식에서는, 아무 노드를 고르지 않은 경우도 연결 방식으로 같이 고려되었습니다.
- 구간 A 에 대한 답은, 전체가 하나로 연결된 구간 c 에 대해 A_c 의 최댓값입니다.

H. 연결 요소와 쿼리

$K = 2$인 경우에는 다음과 같은 방식으로 해결할 수 있습니다.

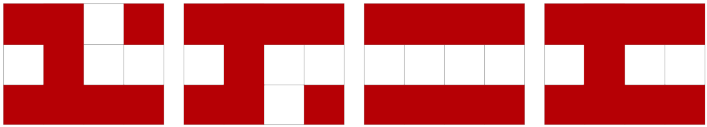
- ✓ 전체가 하나의 연결 성분인 경우만 고려합니다.
  - 바깥에서 이어서 하나의 연결 성분으로 만들어 줄 수 없기 때문입니다.
- ✓ 한쪽 끝에서, 위 칸만 연결 성분에 속한 경우를 U, 아래 칸만 연결 성분에 속한 경우를 D, 모두 연결 성분에 속한 경우를 B, 모두 연결 성분에 속하지 않은 경우를 X 라고 합시다.
- ✓ 왼쪽과 오른쪽이 하나의 연결 성분인 경우만 고려하기 때문에, 왼쪽 끝의 연결 관계와 오른쪽 끝의 연결 관계의 순서쌍으로 원소에 대한 정보를 저장할 수 있습니다.
- ✓ A의 폭이 1 이고 위, 아래 원소가 u, d이면 $A_{(U,U)} = u, A_{(D,D)} = d, A_{(B,B)} = u + d$이고, 나머지 경우는 조건에 맞도록 연결 방법을 만들 수 없기 때문에, $-\infty$ 입니다.
- ✓ 아무 노드도 고르지 않는 경우는 암시적으로 관리해줍니다.

H. 연결 요소와 쿼리

- ✓ $adj(k, s)$는 오른쪽이 k인 구간과 왼쪽이 s인 구간을 합친 후에 하나의 연결 성분이 되는지의 여부입니다.
  - k나 s가 X이거나, $\{k, s\} = \{U, D\}$인 경우 거짓이며, 나머지 경우는 참입니다.
- ✓ 두 구간을 합치는 식은 다음과 같습니다. $(i, j, k, s \in \{U, D, B\})$
  - $(A \oplus B)_{(X,X)} = \max\left(A_{(X,X)}, B_{(X,X)}, \max_{adj(k,s)}\left(A_{(X,k)} + B_{(s,X)}\right)\right)$
  - $(A \oplus B)_{(i,X)} = \max\left(A_{(i,X)}, \max_{adj(k,s)}\left(A_{(j,k)} + B_{(s,x)}\right)\right)$
  - $(A \oplus B)_{(X,j)} = \max\left(\max_{adj(k,s)}\left(A_{(X,k)} + B_{(s,j)}\right), B_{(X,j)}\right)$
  - $(A \oplus B)_{(i,j)} = \max_{adj(k,s)}\left(A_{(i,k)} + B_{(j,s)}\right)$
- ✓ 고려하는 연결 방법은 모두 하나의 연결 성분을 이루므로, 답은 모든 연결 방법의 최댓값입니다.

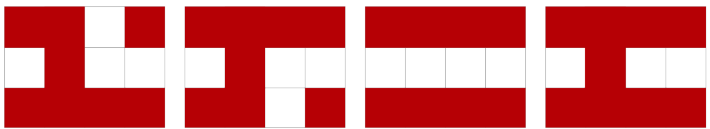
H. 연결 요소와 쿼리

- ✓ $K = 3$인 경우에도 $K = 2$와 같은 방법으로 해결하고 싶지만, 문제가 생깁니다.
  - 전체가 하나의 연결 성분인 경우만 고려할 수 없습니다.
  - 연결 성분이 여럿일 경우, 아래의 경우를 왼쪽과 오른쪽의 연결만으로 구분하기 어렵습니다.
- ✓ 식 $(A \oplus B)_c = \max_{c=l\times r}(A_l + B_r)$를 그대로 사용할 수는 있습니다.
- ✓ 가능한 연결 성분으로 $8 \times 8 + 1 + 9 = 74$가지가 나오기 때문에, 시간복잡도 $\mathcal{O}(N + Q \log N)$에 추가로 74^2 이라는 매우 큰 상수가 붙게 됩니다.



H. 연결 요소와 쿼리

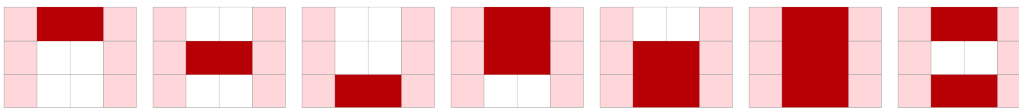
- ✓ 여러가지 상수를 줄일 수 있는 법을 찾아봅시다.
  - 가능한 방법중의 하나는, 구분하는 의미가 없는 연결 방식을 하나로 합치는 것입니다.
    - ▶ 아래 그림의 #1과 #2는 분명 다른 연결 방식이지만, 하나의 연결 성분으로 만들기 위해서는 오른쪽 위의 노드와 오른쪽 아래의 노드를 하나로 연결해야 하고, 같은 연결 성분으로 고려할 수 있습니다.
  - 이 방법으로 연결 방식을 줄여봐야 총 $8 \times 8 + 1 + 4 = 69$ 가지로 밖에 줄일 수 없습니다.



- ✓ 가능한 연결 방식 여러 개를 합치는 방식을 생각해 봅시다.
  - $K = 1$일 때, 가능한 연결 요소의 노드의 값의 합의 최댓값을 의미하는 A_V는 사실 하나의 연결 방식이 아니라, 네 가지의 연결 방식들의 최댓값입니다.
  - 결국 연결 방식에서 L과 R, 그리고 S에서 사용하는 것은 왼쪽 칸을 이용해서 연장할 수 있는지, 오른쪽 칸을 이용해서 연장할 수 있는지의 여부 뿐입니다.
- ✓ 이 아이디어를 사용해서 왼쪽 구간과 오른쪽 구간을 연결하는 방식을 자세히 살펴봅시다.

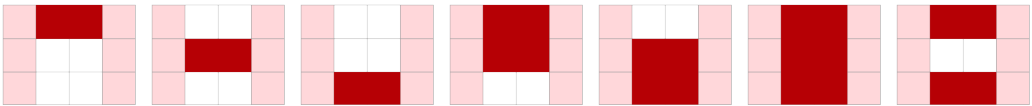
H. 연결 요소와 쿼리

- 왼쪽 구간과 오른쪽 구간을 하나로 연결할 때, 다음과 같은 7가지 방법이 있습니다.
- #1 ~ #6의 경우에는 왼쪽과 오른쪽의 큰 연결 성분을 그저 합칩니다.
- #7 의 경우에는 왼쪽 혹은 오른쪽의 분리되어 있는 구성성분이 하나로 합쳐질 수도 있습니다.



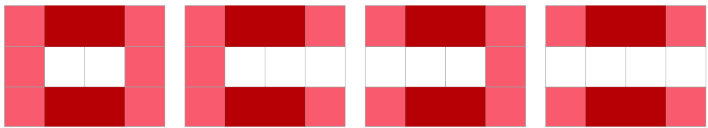
H. 연결 요소와 쿼리

- ✓ 이제 몇 가지 최적화를 합시다.
- ✓ #4 ~ #6의 경우는 사실 따로 고려될 필요가 없이 #1 ~ #3과 함께 고려되어도 됩니다.
  - #4의 경우에, #1의 연결 혹은 #2의 연결만 남겨도 왼쪽과 오른쪽이 잘 연결됩니다.
  - 그래서 #4를, #1과 #2 둘 모두로 활용될 수 있는 경우로 생각할 수 있습니다.
  - #5, #6의 경우에도 마찬가지입니다.



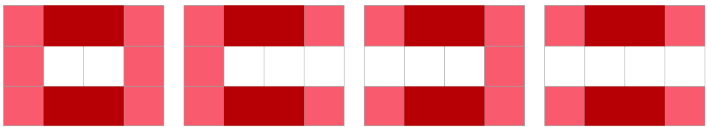
H. 연결 요소와 쿼리

- ✓ #7은 다음과 같은 경우가 있습니다.
  - #A 왼쪽과 오른쪽 모두 하나의 연결 성분으로 되어있는 경우
  - #B 왼쪽이 하나의 연결 성분인 경우
  - #C 오른쪽이 하나의 연결 성분인 경우
  - #D 왼쪽과 오른쪽이 모두 하나의 연결 성분이 아닌 경우
- ✓ #A 는 위의 #1 혹은 #3으로 모두 활용될 수 있는 경우입니다.



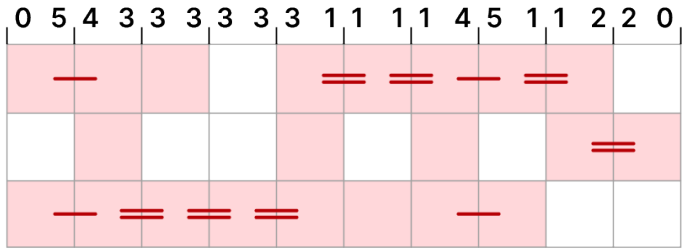
H. 연결 요소와 쿼리

- ✓ #B 와 #C 는 모두, 한 쪽의 끝은 다른 연결 성분이고, 다른 쪽의 끝은 같은 연결 성분인 두 연결 방법을 합치는 경우입니다.
  - #B 는 오른쪽 구간의 연결은 왼쪽 구간이 도와준 형태이며, #C 는 반대입니다.
- ✓ #D 의 경우에는, 매우 특이하게 왼쪽 **혹은** 오른쪽 구간이 연결되어야 하는 형태입니다.
  - 이 형태의 구간은, 같은 형태의 구간이 연장되다가, #B 혹은 #C 의 형태로 왼쪽 혹은 오른쪽 끝에서 연결됩니다.
  - 즉, #D 는 #B 와 #C 모두로 활용될 수 있는 형태입니다.



- ✓ 이 방식으로 정리하면, 구간의 한쪽 끝을 다음과 같은 6가지 형태로 정리할 수 있습니다.
  - #4는 끝쪽 위 아래의 두 노드가 연결된 형태이며, #5는 그렇지 않(을 수도 있)습니다.
  - #1은 #1과, #2는 #2와, #3은 #3과, #4는 #5와, #5는 #4와 연결됩니다.
  - #0은 왼쪽 혹은 오른쪽으로 추가로 연결하지 않겠다는 의미입니다.
- ✓ 이제 $K = 2$와 같은 형태로 왼쪽과 오른쪽을 나눠서 생각해 주면 됩니다.

✓ 이 방식으로 연결된 예는 다음과 같습니다.



- ✓ 식 계산은 다음과 같은 방법으로 가능합니다.
- ✓ $adj(1,1), adj(2,2), adj(3,3), adj(4,5), adj(5,4)$ 만 참입니다.
  - $(A \oplus B)_{(i,j)} \leftarrow \max_{adj(k,s)} \left(A_{(i,k)} + B_{(j,s)}\right)$
  - $(A \oplus B)_{(i,0)} \leftarrow \max\left((A \oplus B)_{(i,0)}, \max_{j \neq 5}\left(A_{(i,j)}\right)\right)$
  - $(A \oplus B)_{(0,j)} \leftarrow \max\left((A \oplus B)_{(0,j)}, \max_{i \neq 5}\left(B_{(i,j)}\right)\right)$
- ✓ 5는 하나의 연결 성분을 이루지 않기 때문에, $(i,0)$ 혹은 $(0,j)$ 를 구성할 때, 고려하면 안됩니다.

H. 연결 요소와 쿼리

- ✓ 이제 초기화를 잘 합시다. 폭이 1인 구간의 수를 위에서부터 v_1, v_2, v_3 이라고 합시다.
- ✓ 각 연결 성분의 왼쪽과 오른쪽이 어떤 방식으로 활용될 수 있는지 생각합시다.

$$
\begin{aligned}
\text{if}\quad & i=j=0 \to A_{(i,j)} = \max(v_1, v_2, v_3, v_1+v_2, v_2+v_3, v_1+v_2+v_3)\\
\text{else if}\quad & i,j \in \{0,1\} \to A_{(i,j)} = \max(v_1, v_1+v_2, v_1+v_2+v_3)\\
\text{else if}\quad & i,j \in \{0,2\} \to A_{(i,j)} = \max(v_1,0)+v_2+\max(v_3,0)\\
\text{else if}\quad & i,j \in \{0,3\} \to A_{(i,j)} = \max(v_1+v_2+v_3, v_2+v_3, v_3)\\
\text{else if}\quad & \{i,j\}=\{1,2\} \to A_{(i,j)} = \max(v_1+v_2, v_1+v_2+v_3)\\
\text{else if}\quad & \{i,j\}=\{2,3\} \to A_{(i,j)} = \max(v_2+v_3, v_1+v_2+v_3)\\
\text{else if}\quad & 2 \notin \{i,j\}, 5 \in \{i,j\} \to A_{(i,j)} = v_1+\max(v_2,0)+v_3\\
\text{else}\quad & \to A_{i,j} = v_1+v_2+v_3
\end{aligned}
$$

H. 연결 요소와 쿼리

- ✓ 세로 길이가 1, 2, 3인 경우를 각각 구현합시다.
- ✓ 쿼리의 주어진 x_1, x_2 값에 따라 세그먼트 트리를 만들고, 해당 세그먼트 트리에서 값을 반환하는 코드를 작성합시다. 구현해 보면 생각보다 빨리 돈다는 사실을 알 수 있습니다.

Open Contest

문제		의도한 난이도	출제자
Q	방문 판매 (Hard)	**Medium**	이선호 glanceyes
R	카드컨트롤 (Hard)	**Hard**	이준석 semteo04