Coder's High 2015 Side Contest

Solution Sketch

A. Revenge of the ants

1. Solved / Trial : 1 / 46 (2.174%)

   First Accept : hong_gihas_face (116min)

2. Solved / Trial : 0 / 10 (0%)

   First Accept : N/A

- 출제 : 김경근(kriii)

개미 문제는 전통적으로

- 먼저 개미에게 위치가 증가하는 순서대로 0에서 $N + M - 1$까지 번호를 붙입니다.

- 두 개미간의 충돌을 스쳐 지나가면서 서로의 번호를 교환하는 것으로 보면 이해하기 쉽습니다.

- 위와 같이 이해하면 L초의 시간이 지나면 번호는 몰라도 속도는 (외형 이라고 하겠습니다.)처음과 같은 상태가 됩니다.

이 문제는 개미들이 원형레일 위에 있다.

- 0으로 번호 붙인 개미를 쫓아갑니다. 개미들이 어떻게 충돌하더라도 0에서 시계방향으로 이동했을 때 나오는 가장 첫 개미는 1, 다음은 2, …, 마지막으로 $N+M-1$이니까 한 개미만 어디에 있는지 알면 다른 모든 개미의 위치도 알 수 있습니다.

- 이것은 시뮬레이션으로 알 수 있는데, 이를 $O\left((N+M)^2\right)$로 하면 1번 문제를 해결할 수 있고, $O\left((N+M)\lg(N+M)\right)$에 하면 2번 문제를 해결할 수 있습니다.

이 문제는 개미들이 원형레일 위에 있다.

- 그래서 처음에 0번의 위치에 있던 개미가 i번의 위치로 갔다고 하면, 이것을 $\frac{N+M}{\gcd(N+M,i)}$ 번 반복하면 모든 개미가 처음 상태가 되는 것을 알 수 있습니다.

- 실은 개미의 배치가 어떻든 $i \equiv N - M (mod\ (N+M))$입니다. 이제 답은 $\frac{N+M}{\gcd\left(N+M,M+N+(N-M)\right)} \times L$이 될 것 같지만...

L초보다 더 빨리 외형이 같아질 수 있다.

- 주어진 개미의 배치에서 패턴을 찾아야 합니다. 최소 몇 초가 지나야 외형이 같아지는지 찾습니다. (KMP 등의 방법을 사용하면 됩니다.) 이 시간을 T라고 합시다.

- 이제 답은 $\frac{N+M}{\gcd(N+M, N+M+(N-M)*\frac{T}{L})} \times T$입니다.

더 말하고 싶은 것

- 개미의 충돌을 시뮬레이션 해보지 않아도 되는 이유는 한 개미와 충돌하는 개미들이 시계 방향으로 움직이는 개미와 반시계 방향으로 움직이는 개미가 교대로 충돌해서 어차피 번호는 같아지기 때문입니다.

- 개미의 배치에서 패턴을 찾는 것은 ALGOSPOT의 JAEHASAFE문제를 참고하세요.

B. Binary Mobile Tree

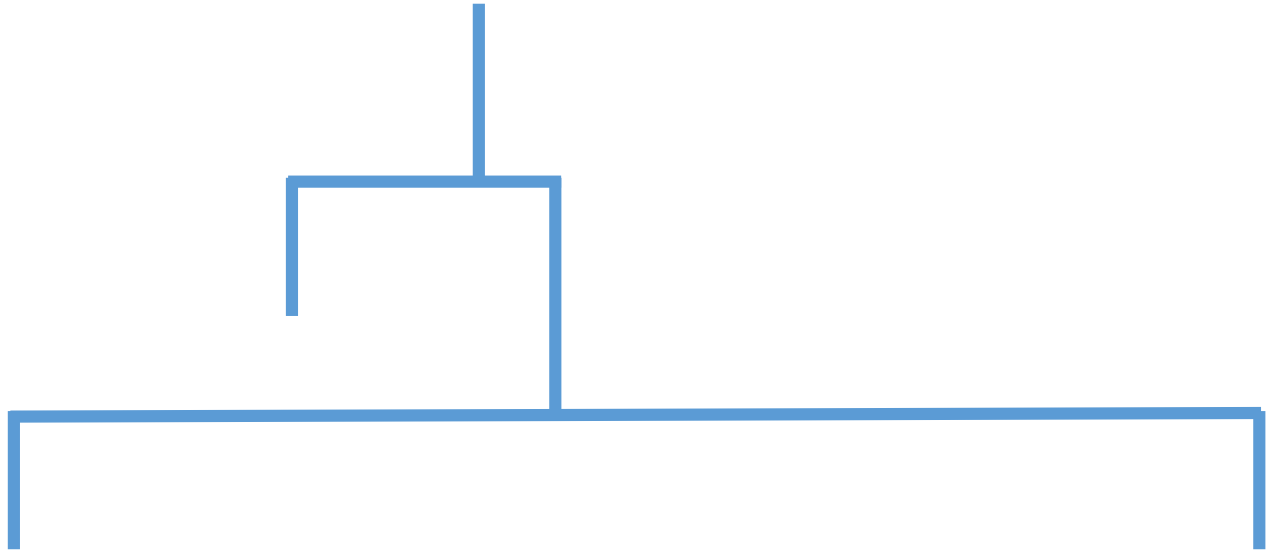
1. Solved / Trial : 10 / 70 (14.286%)

   First Accept : koosaga (24min)

- 출제 : 류현종(xhae)

시키는 대로 짜면 됩니다.

- 아래와 같은 경우를 조심해서!



C. 판게아

1. Solved / Trial : 8 / 24 (33.333%)

   First Accept : koosaga (57min)

2. Solved / Trial : 0 / 3 (0%)

   First Accept : N/A

- 출제 : 류현종(xhae)

주어진 그래프에 간선을 계속 추가하면서 MST를 찾는 문제

- Easy
  - 매 번 새로운 그래프에 대해서 MST를 찾으면 $O(mVlgV)$
- Hard
  - 새로 놓인 간선과 새로운 MST, 기존 MST간의 관계
    - Use Link-cut tree!
      - $O(lgN)$ for link, cut, find value on path
      - $O(mlgV)$ for model solution

D. 빛의 왕과 거울의 미로

1. Solved / Trial : 2 / 5 (40%)

   First Accept : hong_gihas_face (103min)

2. Solved / Trial : N/A

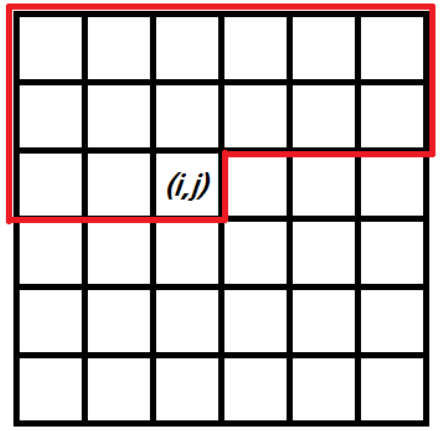
   First Accept : N/A

- 출제 : 김경근(kriii)

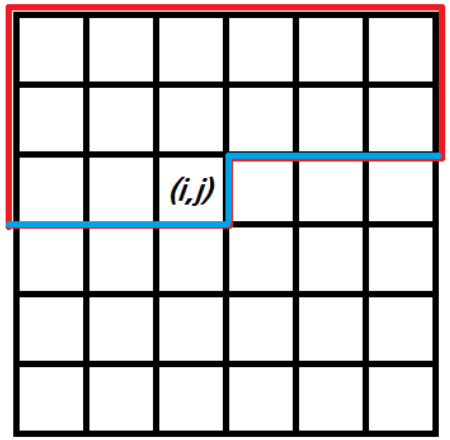
DP를 하자

- DP상태가 복잡합니다.

- $D_{i,j,s} = (i,j)$까지의 칸을 채웠고 현재 상태가 s인 경우의 수



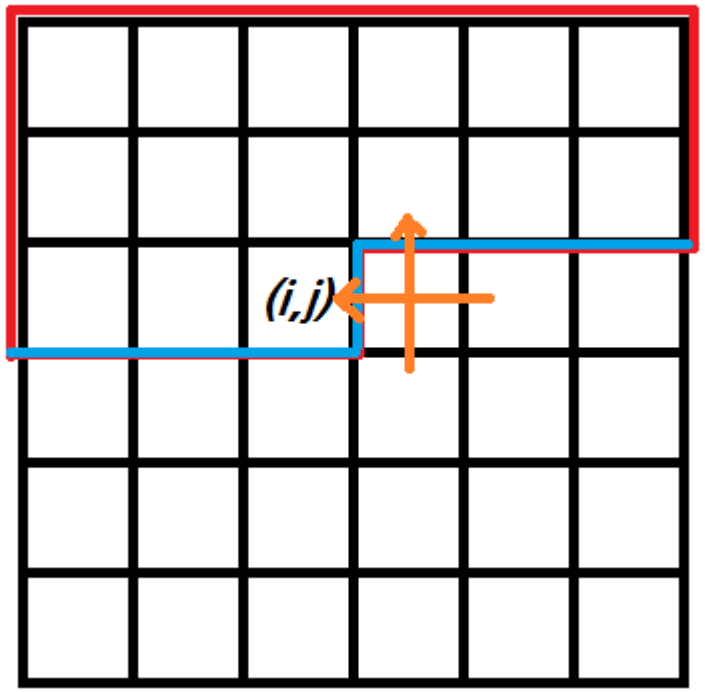
빨간색 안에 있는 칸을 모두 채운 것입니다.



상태는 파란색 선에 속하는 변으로 레이저를 쏘면 어느 변으로 나오는지를 모두 저장합니다.

다음 칸 채우기(빈칸)

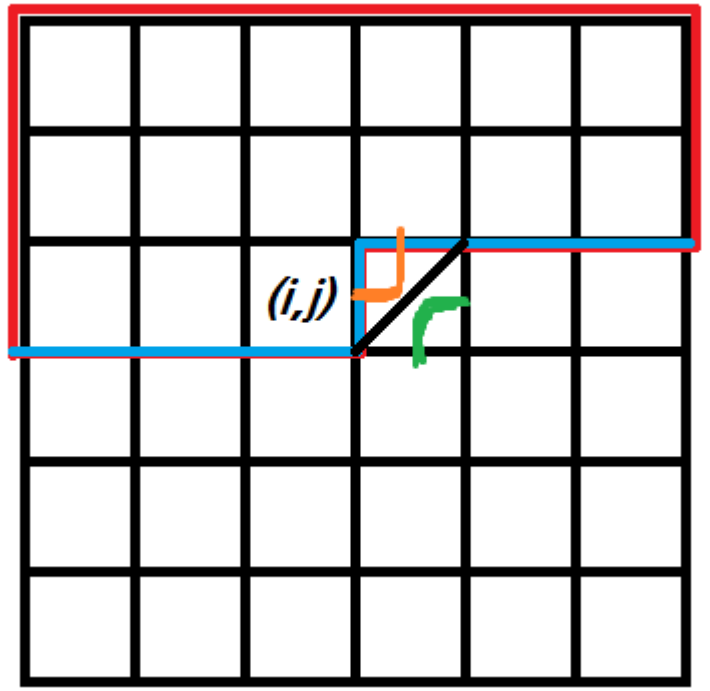
- 이전 상태와 상태가 달라지지 않습니다.



다음 칸 채우기(/)

- 주황색으로 표시된 두 변으로 레이저를 쐈을 때 어느 변으로 갔느냐에 따라 달라집니다.

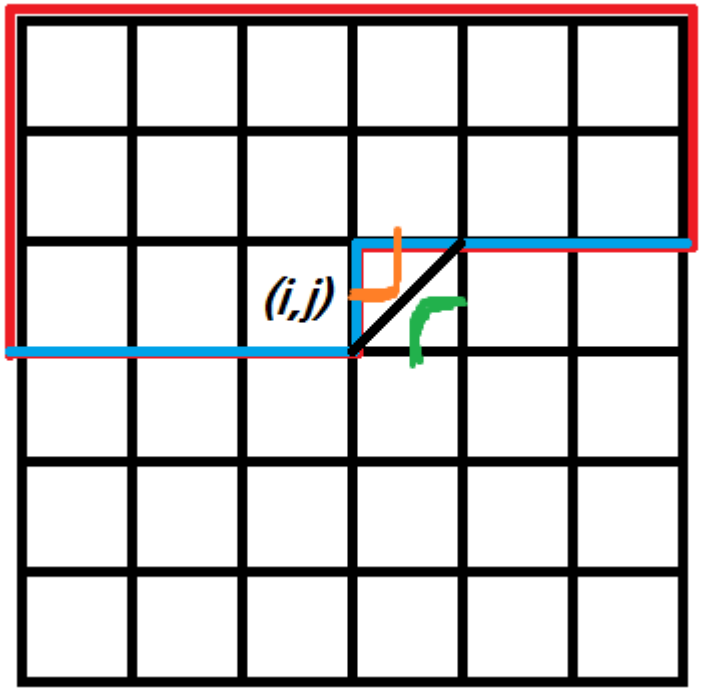
- 한쪽이라도 파란색 변으로 가면, 두 변이 적절히 이어지도록 상태를 조정하고 초록색으로 이어진 두 변을 이어줍니다.



다음 칸 채우기(/)

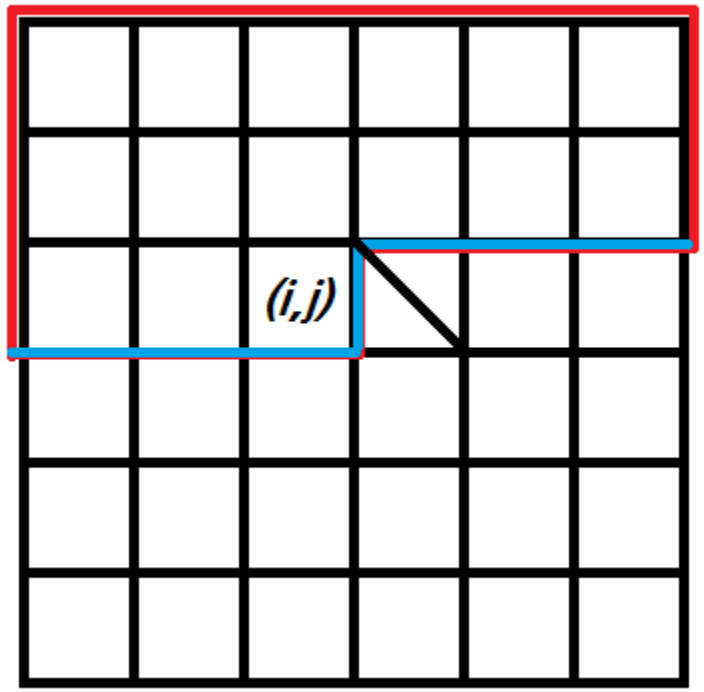
- 둘 다 빨간색 변으로 갈 때,

1) 둘이 우리가 원하는 x, y라면 나머지 ?는 모두 마음대로 채워서 답을 갱신한 후 버리고
2) 하나만 원하는 x, y라면 이 상태는 다음으로 가도 x, y를 이을 수 없으므로 버리고
3) 나머지 경우에는 상태를 다음으로 넘겨줍니다.



다음 칸 채우기(\)

- 이 경우에는 이 칸 왼쪽과 위로 가는 변을 교환해 주는 것으로 상태를 만들 수 있습니다.



예외처리

- i, j가 변의 끝에 닿는 경우 적절한 예외처리를 통해 답을 구해야 합니다. 이전에 상태를 만드는 방식과 매우 유사합니다.

- 유사한 문제 : COCI 2010/2011 Contest #6 6. VODA

E. 룩과 구슬

1. Solved / Trial : 0 / 7 (0%)

   First Accept : N/A

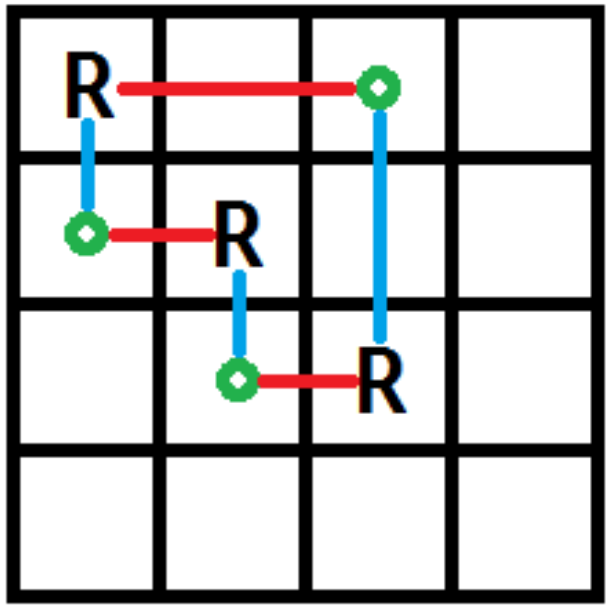
- 출제 : 김경근(kriii)

플로우 같은데..??

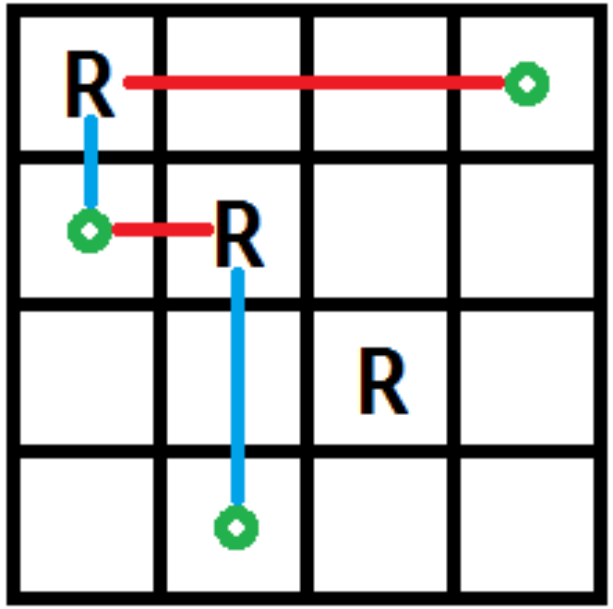
- 하지만 이분 매칭 같은 것으로는 모델링이 되지 않습니다.

구슬 하나씩 놓아보기

두 가지 경우가 있습니다.



사이클을 따라 구슬을 놓음
(두 룩에 동시에 공격받는 칸에서 시작)



한 경로를 따라 구슬을 놓음
(한 룩에만 공격받는 칸에서 시작)

Cycle Canceling

- 룩을 하나의 정점으로, 앞에서 파란색 선 – 구슬 – 빨간색 선을 파란색 쪽에서 빨간색 쪽으로 가는 하나의 간선으로 봅니다. 칸에 놓을 수 있는 구슬의 수가 간선의 용량이 됩니다. 비용은 구슬이 모두 동등하므로 1입니다.

- 룩에 한번만 공격받는 부분은 이 부분만을 모두 공격할 수 있다는 설정을 가진 슈퍼 룩을 가정하는 것으로 처리할 수 있습니다.

Cycle Canceling

- 이제 Bellman-Ford Algorithm을 이용해 최장경로를 구한 후, 양수사이클이 있는지 검사하여 양수 사이클을 확장경로로 하여 MCMF를 합니다. 양수 사이클이 검사되지 않을 때까지 하면 됩니다.

- 2011 World Finals D번을 대회 중에 풀 수 있을 정도로 간단화 시킨 문제입니다.

F. XOR삼형제

1. Solved / Trial : 13 / 46 (28.261%)

   First Accept : Nada (13min)

2. Solved / Trial : 11 / 21 (52.381%)

   First Accept : ainu7 (23min)

- 출제 : 류현종(xhae)

F

- ans(n) =
  - ans(n - 1) + n 이 가능할경우 n을 추가
  - 불가능할 경우 ans(n - 1) 에서 제일 작은 숫자의 leading one 상위비트에 1 추가, 제일 작은 숫자는 삭제
    - n을 더하고 제일 작은 숫자를 가져오는 결과

1. 가능할 경우:
   - ans(n)은 ans(n - 1)보다 사이즈가 최대 1만큼 클 수 있기에 Valid
2. 불가능할 경우:
   - 위와 같은 형태로 답을 Build해 나갈 때, case by case로 n을 더하는 경우가 최선임이 증명 가능

G. 업적의 노예

1. Solved / Trial : 1 / 11 (9.091%)
   First Accept : ainu7 (94min)
2. Solved / Trial : 1 / 3 (33.333%)
   First Accept : ainu7 (192min)
3. Solved / Trial : 7 / 19 (36.842%)
   First Accept : ainu7 (68min)

- 출제 : 김경근(kriii)

점화식은 간단

$$P_x \equiv (P_{x-N} + P_{x-N+1} + \cdots + P_{x-N+K}) \times (K+1)^{-1} \ (mod \ \ 10^9 + 7)$$

우리가 구해야 하는 i번째 값은 초기값이

$$P_x = \begin{cases} 1 \ \ if \ i = x \\ 0 \ \ if \ i \neq x \end{cases} \quad (0 \leq x < N)$$

일 때의 P_M입니다.

키타마사법

- 이런 점화식을 빠르게 푸는 법으로는 행렬을 이용하는 법이 가장 널리 알려져 있는데, 이 방법의 시간복잡도는 $O(N^3 lgM)$의 시간이 걸려 이 문제를 풀기에는 느립니다.

- 그러므로 이 문제를 풀기 위해서는 좀 더 빠른 방법이 필요한데, 그 중 하나가 일본의 비술 키타마사법 입니다.

키타마사법

- 점화식의 형태에 의해 $P_x \equiv b_0P_0 + b_1P_1 + \cdots + b_{N-1}P_{N-1}$ 입니다.

- 초기값이 다른 경우라고 생각하면 아래 식도 성립합니다.

$$P_{i+x} \equiv b_0P_i + b_1P_{i+1} + \cdots + b_{N-1}P_{i+N-1}$$

- $P_{2x} \equiv b_0P_x + b_1P_{x+1} + \cdots + b_{N-1}P_{x+N-1}$

$$\equiv c_0P_0 + c_1P_1 + \cdots + c_{2N-2}P_{2N-2}$$

키타마사법

- c_0에서 c_{2N-2}까지는 $\sum_{i=0}^{N-1} b_i x^i$ 를 제곱하는 것으로 $O(N^2)$에 구할 수 있습니다.

- P_N 에서 P_{2N-2}를 P_0에서 P_{N-1}까지의 합으로 나타내는 것도 $O(N^2)$에 가능합니다.

- 즉 P_{2x}를 표시하는 법을 P_x에서 알 수 있다.

키타마사법

- $P_M \equiv b_0 P_0 + b_1 P_1 + \cdots + b_{N-1} P_{N-1}$로 나타난다면 답은 순서대로 b_0에서 b_{N-1}까지입니다.

G3

$L_x = P_x$만 1인 경우의 극한값 이라고 합시다.

$P_x = c_x (0 \leq x < N)$인 경우

$\lim_{x \to \infty} P_x = c_0 L_0 + \cdots + c_{N-1} L_{N-1}$입니다.

P_x는 어떤 초기값을 가져도 수렴하기 때문입니다.

G3

편의를 위해

$$P_x \equiv \frac{(q_0 P_{x-N} + q_1 P_{x-N+1} + \cdots + q_N P_{x-1})}{q_0 + q_1 + \cdots + q_N}$$

이라고 합시다.

$$q_0 + q_1 + \cdots + q_N = S$$

라고 하고

$N = 5$인 예를 들겠습니다.

G3

우리가 원하는 $(P_0, P_1, P_2, P_3, P_4, P_5)$쌍을 적어보면

$$(1,0,0,0,0,\frac{q_0}{S})$$
$$(0,1,0,0,0,\frac{q_1}{S})$$
$$(0,0,1,0,0,\frac{q_2}{S})$$
$$(0,0,0,1,0,\frac{q_3}{S})$$
$$(0,0,0,0,1,\frac{q_4}{S})$$

즉

G3

$$L_0 = \frac{q_0}{S} L_{N-1}$$

$$L_1 = L_0 + \frac{q_1}{S} L_{N-1}$$

$$L_2 = L_1 + \frac{q_2}{S} L_{N-1}$$

$$L_3 = L_2 + \frac{q_3}{S} L_{N-1}$$

$$L_4 = L_3 + \frac{q_4}{S} L_{N-1}$$

입니다. 이 식들을 풀면

G3

$$L_0 : L_1 : L_2 : L_3 : L_4$$
$$= \sum_{i=0}^{0} q_i : \sum_{i=0}^{1} q_i : \sum_{i=0}^{2} q_i : \sum_{i=0}^{3} q_i : \sum_{i=0}^{4} q_i$$

큰 N에 대해서도 마찬가지이고, $\sum L_i = 1$ 이므로 적절히 q를 넣어서 계산해 주시면 되겠습니다.

H. 소방차 게임

1. Solved / Trial : N/A

   First Accept : N/A

- 출제 : 류현종(xhae)

소방차는 멈추지 않는다.

- 소방서와 교점을 정점으로, 선분을 간선으로 하는 그래프
  - Issue 1: 선분과 선분의 교점
  - Issue 2: 선분과 원의 교점
- 가상의 Start Node -> 모든 소방서로의 거리 0
- 한 번의 Dijkstra로 모든 쿼리 한번에 해결!
- Further question: 두 도로의 교점이 둘 이상이 생기는 경우

I. *I*교신자

1. Solved / Trial : 2 / 7 (28.571%)
   First Accept : hong_gihas_face (185min)
2. Solved / Trial : N/A
   First Accept : N/A
3. Solved / Trial : 0 / 2 (0%)
   First Accept : N/A

- 출제 : 김경근(kriii)

DP를 하자!

쓰여진 모든 카드가 스택의 가장 위에 있는 원소를 만드는 데 필요한 경우만 따로 모읍니다.

+카드 i장, $*$ 카드 j장 사용했을 때

$D_{i,j}$ = 가능한 모든 원소의 합

$C_{i,j}$ = 가능한 카드의 배치 수

i, j가 정해지면 I카드의 개수는 정해짐을 알 수 있습니다.

DP를 하자!

+ 카드를 사용하면

$$D_{i,j} = \sum D_{x,y} * C_{i-x-1,j-y} + D_{i-x-1,j-y} * C_{x,y}$$

* 카드를 사용하면

$$D_{i,j} = \sum D_{x,y} * D_{i-x,j-y-1}$$

두 경우를 모두 더해서 C도 계산

$$C_{i,j} = \sum C_{x,y} * C_{i-x-1,j-y} + \sum C_{x,y} * C_{i-x,j-y-1}$$

그러나 이것으로는 부족하다.

k를 추가합니다 : 카드 배치의 가장 앞에 k개의 I카드를 연속해서 쓴다.

스택의 처음에 무한한 개수의 I가 있기 때문입니다.

위에서 K번째는..?

앞의 DP값을 모두 구하고 마지막에 새로운 DP를 하나 더 돌립니다.

$G_{k,i,j}$ = 이전에 k개의 값을 보았고, +카드를 i장, $*$ 카드를 j장 사용했을 때의 경우의 수

적절한 예외 처리를 하여 원래 있던 스택 영역을 침범하는 경우를 처리해줘야 답이 나옵니다.

J. It has the same Suffix Array

1. Solved / Trial : 1 / 2 (50%)

   First Accept : ainta (127min)

- 출제 : 김경근(kriii)

J

- 1. suffix array를 구한다.
- 2. 순서대로 맨 앞글자를 보면서 각 글자가 변할 수 있는 경우의 수를 더한다. (같아질 경우 다음글자 suffix를 비교해 판단)

- 시간복잡도 : O(N)