제1회 피갤컵 풀이

by

Official Solutions

문제		의도한 난이도	출제자
A	PlayStation이 아니에요	**Very Easy**	plast
B	가위바위보	**Easy**	amel
C	짚단 베기	**Medium**	wapas
D	합성방진	**Medium**	bubbler
E	소신발언	**Medium**	dabbler1
F	좋은 격자	**Hard**	pyb1031
G	미로 챌린지	**Hard**	pyb1031
H	치터찾기	**Hard**	pyb1031
I	낭만고양이	**Challenging**	hamuim

A. PlayStation이 아니에요

`string`
출제진 의도 – **Very Easy**

- 제출 699번, 정답자 389명 (정답률 57.65%)
- 처음 푼 사람: **ychangseok**, 0분
- 출제자: `plast`

A. PlayStation이 아니에요

- 문자열 "PS"의 바로 다음에 오는 "4", "5"를 제거해 주어야 합니다.
- 출력할 문자열 S를 새로 구성하는데, 입력받은 문자열을 하나씩 S에 붙여가다가 문자열 S에서 PS 다음에 4, 5가 나오면 붙이지 않고 건너뛰는 방식으로 구현할 수 있습니다.

B. 가위바위보

`string`, `brute_force`
출제진 의도 – **Easy**

- 제출 359번, 정답자 102명 (정답률 28.69%)
- 처음 푼 사람: **xiaowuc1**, 5분
- 출제자: `amel`

- P 라운드 만에 종료되었다고 가정합시다.
- 모든 사람의 문자열을 $1, \ldots, P$ 번째 문자까지만 남겼을 때, 살아남은 사람들의 문자열은 모두 같아야 하고 살아남지 못한 사람들의 문자열은 그와 달라야 합니다.
- 그런데 이의 역도 성립합니다.
- 즉 K 명 이하의 사람들끼리 문자열이 모두 같고, 나머지 사람들은 그 문자열과 다른 경우 문제 조건에 맞게 종료됨을 알 수 있습니다.

- 따라서 $P = 1, 2, \ldots, M$ 인 경우에 위 조건을 만족하는지 검사하여 가능한 최소의 P 를 찾아주면 됩니다.
- 서로 같은 문자열이 몇 개씩 존재하는지는 먼저 각 문자열을 P 번째 문자까지 자른 후 `std::map`, `dictionary` 등의 자료구조를 활용하여 세면 됩니다.
- 정답을 출력할 때 문자열을 그대로 출력하지 말고 이기는 문자열을 출력해야 함에 유의합시다.
- $\mathcal{O}(NM^2 \log M)$ 정도의 시간복잡도로 통과할 수 있습니다.

C. 짚단 베기

`greedy`, `geometry`
출제진 의도 – **Medium**

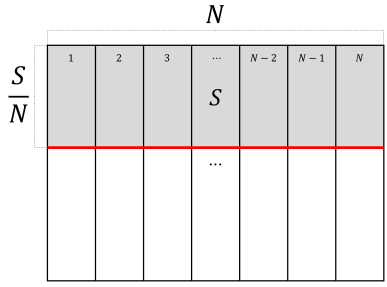
- ✓ 제출 252번, 정답자 73명 (정답률 28.968%)
- ✓ 처음 푼 사람: **mj1000j**, 10분
- ✓ 출제자: wapas

C. 짚단 베기

- N 단 베기를 하기 위해 드는 힘의 수식을 관찰합니다.
- 짚단을 베는데 드는 힘은 잘라낸 넓이에 따라 같이 증가합니다.
- 따라서 잘라낸 넓이를 가장 작게 자르는 것이 드는 힘이 가장 적습니다.
- 그러므로 멋진 N 단 베기를 할 때, 넓이를 정확히 S 만큼 잘라내는 것이 가장 힘이 적게 듭니다.

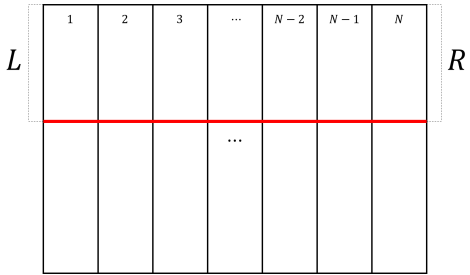
C. 짚단 베기

✓ 먼저 간단하게 가로가 N, 세로가 $\frac{S}{N}$ 이 되게 짚단을 베어봅니다.



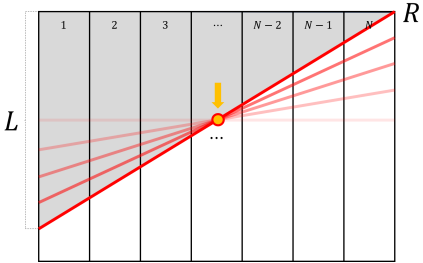
C. 짚단 베기

- 좌우 양끝 변의 길이를 각각 L, R이라 했을 때 $L + R = \frac{2S}{H}$를 만족합니다.



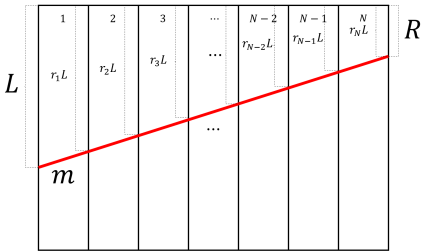
C. 짚단 베기

- $L + R = \dfrac{2S}{H}$ 를 유지하면서 베기 직선을 그어봅니다.
- 그러면 베기 직선은 기울기와 상관없이 특정한 한 점을 지나는 것을 알 수 있습니다.



C. 짚단 베기

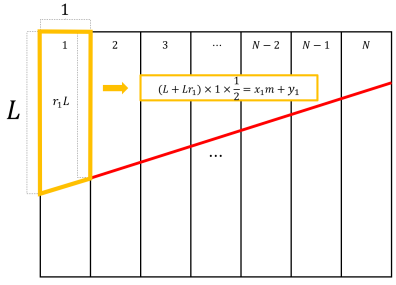
- 직선의 기울기를 m 이고, $\overline{B_iC_i}$ 를 r_iL 로 정의합니다. 그러면 $m = r_iL - r_{i+1}L$ 입니다.
- 또한, $m = \dfrac{L-R}{N}$ 입니다.



- $L + R = \frac{2S}{H}$ 에서 $R = \frac{2S}{H} - L$
- $m = \frac{L - R}{N}$ 에서 $L = Nm + R$ 이므로
- $L = k_0 m + c_0$ 로 표현할 수 있습니다. (k_0 과 c_0 은 상수)
- 그리고 $m = L - r_1 L$ 과 $m = r_i L - r_{i+1} L$ 를 통해 $r_i L = k_i m + c_i$ 로 표현할 수 있습니다. (k_i 와 c_i 는 상수)

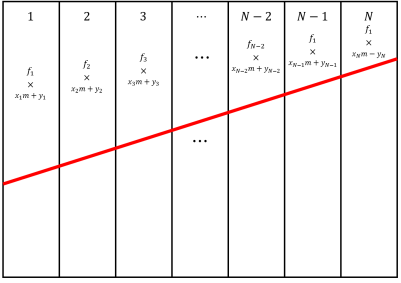
C. 짚단 베기

- 따라서 i 번 짚단의 베어진 넓이는 $x_i m + y_i$ 로 표현할 수 있습니다. (x_i 와 y_i 는 상수)



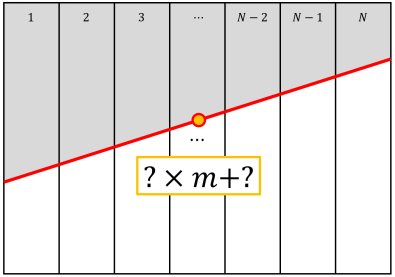
C. 짚단 베기

✓ $\sum_{i=1}^{N} f_i \times (x_i m + y_i)$ 또한 m에 관한 일차방정식으로 표현할 수 있습니다.



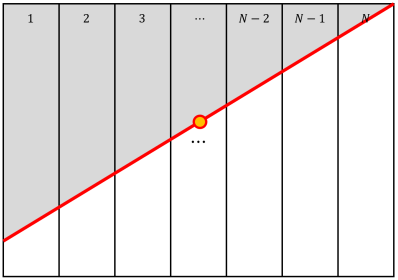
C. 짚단 베기

- ✓ 그러므로 직선의 기울기 m에 따라 짚단 N단 베기하는데 드는 힘은 선형적입니다.
- ✓ 따라서 드는 힘이 최소가 될 수 있는 후보는 m이 최대일 때와 m이 최소일 때입니다.



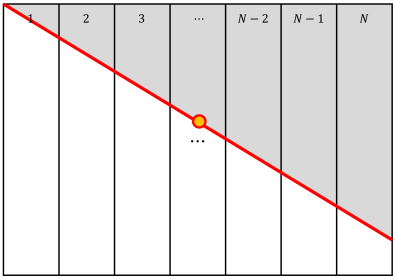
C. 짚단 베기

- ✓ m이 최대일 때, 우측 짚단을 최대한 적게 베는 것과 같습니다.



C. 짚단 베기

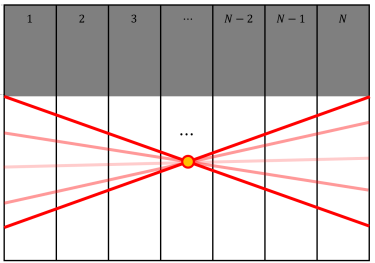
- ✓ m이 최소일 때, 좌측 짚단을 최대한 적게 베는 것과 같습니다.



C. 짚단 베기

- ✓ 따라서 베기 직선 기울기를 최대, 최소로 한 뒤 얻은 값 중 더 작은 값이 정답입니다.
- ✓ 단, 아래와 같이 S가 $\frac{NH}{2}$를 초과하는 경우 예외 처리가 필요합니다. 이 경우에도 풀이는 같습니다.

$$\frac{2S}{N} - H$$



D. 합성방진

`math, constructive`
출제진 의도 – **Medium**

- 제출 138번, 정답자 72명 (정답률 52.174%)
- 처음 푼 사람: **kes0716**, 7분
- 출제자: `bubbler`

D. 합성방진

- 먼저, 서로 다른 이웃한 정수 쌍의 가지수를 최소화하는 방법을 생각해 봅시다.
- 첫째 줄을 정한 다음에 각각의 수를 다음과 같이 대각선 방향으로 배열하면 이를 달성할 수 있습니다.

a b c d e f g h
b c d e f g h a
c d e f g h a b
⋮
h a b c d e f g

- 이렇게 하면 이웃한 수의 쌍의 집합은 a-b, b-c, ⋯ , g-h, h-a가 됩니다.

D. 합성방진

- 이제 이웃 합이 모두 합성수인 원순열을 찾는 문제가 되었습니다.
- 짝수 + 짝수 = 짝수, 홀수 + 홀수 = 짝수이므로, 짝수끼리, 홀수끼리 먼저 뭉쳐주면 거의 모든 쌍을 쉽게 합성수로 만들 수 있습니다.
- 짝수 + 홀수 부분을 합성수로 만들기 위해서는 더해서 9가 나오는 두 개의 쌍을 두 경계 자리에 고정시켜 놓으면 됩니다.
- 예를 들어 $n = 8$이면 1 3 5 7 2 4 6 8을 돌려가면서 출력하면 됩니다.
- 이 구성을 사용하면 $n \geq 6$인 모든 n에 대한 해를 구할 수 있습니다.

E. 소신발언

`parametric_search`
출제진 의도 – **Medium**

- ✓ 제출 101번, 정답자 59명 (정답률 58.42%)
- ✓ 처음 푼 사람: **moonrabbit2**, 3분
- ✓ 출제자: `dabbler1`

- 먼저 T가 주어졌을 때 T의 시간 안에 얼음을 모두 녹일 수 있을지를 생각해봅시다.
- j번 소의 얼음을 T의 시간 안에 녹이려면 히터를 설치할 수 있는 위치는 $|i-j| \le T/a_j$를 만족하는 i가 될 것입니다.
- 따라서 모든 j에 대해 위 조건을 만족하는 i가 존재한다면 T의 시간 안에 얼음을 녹일 수 있고, 존재하지 않으면 녹일 수 없습니다.
- 이 방식으로 고정된 T에 대해 $O(N)$ 안에 가능 여부를 확인할 수 있고, T의 최댓값이 $O(N \max a_j)$이므로, 매개변수 탐색을 통해 $O(N \log(N \max a_j))$ 시간 안에 문제를 해결할 수 있습니다.

- 다른 풀이로는 볼록성을 이용하는 것이 있습니다.
- 각 j에 대해 $f_j(x) = |x - j| \times a_j$라고 하면, f_j는 볼록함수입니다.
- 그러면 문제의 T_i는 $T_i = \max_{1 \le j \le n} f_j(i)$로 표현되는데, 볼록함수들의 최댓값은 역시 볼록함수이므로 결국 특정 볼록함수의 정수 좌표에서의 최솟값을 구하는 문제로 귀결됩니다.
- 따라서 삼분탐색 등을 활용해 답을 구할 수 있습니다.
- 이외에도 컨벡스 헐 트릭, 리차오 트리 등을 활용할 수 있습니다.

F. 좋은 격자

`permutation_cycle_decomposition`, `graph_traversal`
출제진 의도 – **Hard**

- 제출 79번, 정답자 26명 (정답률 32.91%)
- 처음 푼 사람: **xiaowuc1**, 43분
- 출제자: pyb1031

F. 좋은 격자

- 먼저 좋은 격자의 조건에 대해 생각해봅시다.
- 어떤 격자가 좋은 격자라면 1 부터 $N \times N$ 까지의 수를 순서대로 지나는 경로가 존재합니다. 즉, $1 \leq i < N \times N$ 인 모든 i 에 대해 i 와 $i+1$ 이 인접해있어야 합니다.
- 잘 생각해보면 이 역도 성립함을 알 수 있습니다. 따라서 $1 \leq i < N \times N$ 인 모든 i 에 대해 i 와 $i+1$ 을 인접하게 하는 문제가 됩니다.

F. 좋은 격자

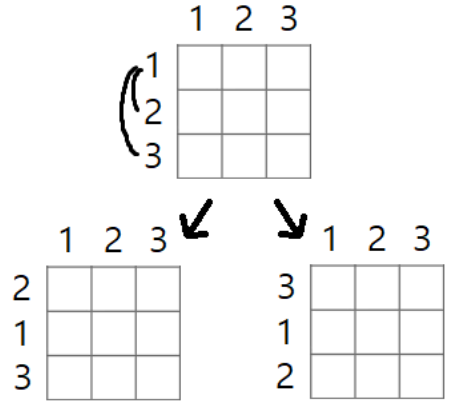
- 다음으로는 문제에서 주어진 시행으로 어떤 두 수 a, b를 인접하게 하고싶다고 해봅시다.
- 이 때 a의 행과 b의 행이 다르고 a의 열과 b의 열이 다르다면 불가능함을 알 수 있습니다.
- 즉, a와 b를 인접하게 하려면 열 또는 행 중 하나는 같아야 합니다.
- a의 행과 b의 행이 같다면 초기위치 기준으로 a의 열과 b의 열이 인접해야 한다는 사실을 알 수 있습니다.
- 비슷하게 a의 열과 b의 열이 같다면 초기위치 기준으로 a의 행과 b의 행이 인접해야 한다는 사실을 알 수 있습니다.
- 행과 열이 서로 영향을 미치지 않으므로 둘을 독립적으로 처리해줄 수 있습니다.

F. 좋은 격자

- ✓ $1 \leq i < N \times N$ 인 모든 i에 대해 i와 $i+1$가 인접해야 하므로, 인접해야 하는 행들과 인접해야 하는 열들을 알 수 있습니다.
- ✓ 인접해야 하는 행들의 정보를 가지고 있다면 행의 번호를 정점으로 하는 그래프를 만들 수 있습니다. 인접해야 하는 두 행의 번호 사이에 정점을 이어줍시다.
- ✓ 격자에서 행의 번호는 위에서 아래로 일렬로 나열되기 때문에 좋은 격자를 만드는것이 가능하다면 이 그래프에서 multiple edge를 제거했을 때 일자모양(리프가 2개인 트리)이 나와야 합니다.
- ✓ 이 그래프의 한쪽 리프에서 반대쪽 리프로 순회하며 나오는 번호들을 순서대로 저장한 배열을 $order$라고 합시다. 좋은 격자가 만들어졌을 때 행의 번호는 $order$와 같거나 $order$를 뒤집은것과 같습니다.

F. 좋은 격자

- 밑의 그림은 1행과 2행, 1행과 3행이 인접해야 하는 상황을 나타낸 예시입니다.



F. 좋은 격자

- 한 번의 시행으로 두 행의 번호를 바꿀 수 있습니다.
- 따라서 행의 번호를 어떻게 만들것인지 고정한다면 최소 시행횟수는 순열 사이클 분할을 이용해 구할 수 있습니다.
- 가능한 두가지 경우 중 작은것을 고르면 됩니다.
- 열에 대해서도 비슷하게 구해준 후 두 값을 더하면 답이 나옵니다.

G. 미로 챌린지

`graph_theory`, `constructive`
출제진 의도 – **Hard**

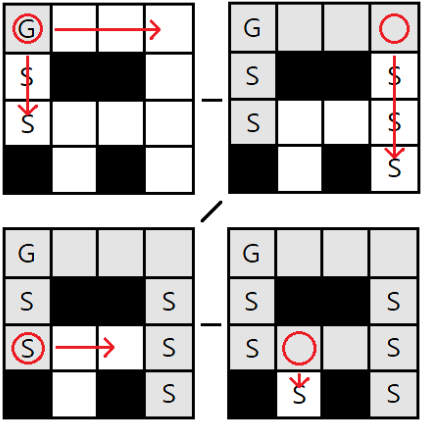
- 제출 83번, 정답자 9명 (정답률 10.84%)
- 처음 푼 사람: **flappybird**, 36분
- 출제자: pyb1031

G. 미로 챌린지

- ✓ 의도한 풀이와 의도하지 않은 풀이가 있습니다. 일단 의도한 풀이를 설명하겠습니다.
- ✓ 두번째 실행에서 알 수 있는것은 돌멩이가 있는가 없는가로 단 2가지 뿐입니다.
- ✓ 돌멩이가 있다면 아래로 가고 없다면 왼쪽으로 가는 방법을 생각할 수 있습니다. 만약 벽을 만나면 다시 반대쪽으로 되돌아갑니다.
- ✓ 이 알고리즘만으로 목적지를 찾게하도록 돌멩이를 배치하는것이 가능합니다.
- ✓ 다음과 같은 알고리즘을 생각해봅시다:
  1. 큐에 목적지의 위치를 넣은 다음 큐가 빌 때 까지 다음을 반복합니다.
  2. 큐에서 위치 하나를 뺀 다음 그 위치에서 상하좌우 방향으로 이미 방문한 칸 또는 벽이 나올 때 까지 직진하며 지나는 칸들의 위치를 큐에 넣습니다. 이 때 방향이 상/하 라면 돌멩이를 배치합니다.

G. 미로 챌린지

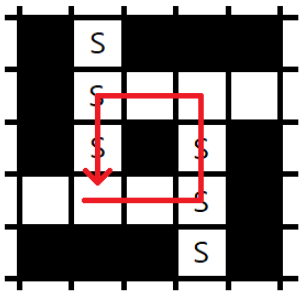
✓ 밑의 그림은 알고리즘을 적용한 예시를 나타냅니다.



- 이미 방문한 칸 중 어떤 칸에서 시작해도 목적지까지 도달하는것이 가능하다고 가정합시다.
- 이 상황에서 어떤 방문한 칸을 골라 한 방향으로 쭉 직진하며 방문한다면 그 양끝은 이미 방문한 칸 또는 벽이 됩니다.
- 양끝이 모두 벽일리는 없고 벽을 만나면 반대방향으로 되돌아가기 때문에 새롭게 방문한 칸에서도 기존에 방문했던 칸을 도달할 수 있습니다.
- 따라서 귀납적으로 어떤 칸에서 시작해도 목적지에 도달할 수 있게 됩니다.

G. 미로 챌린지

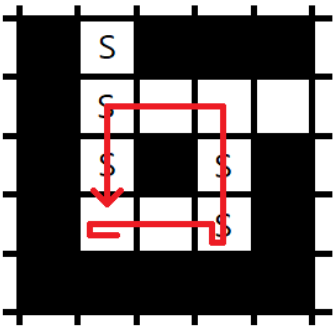
- ✓ 의도하지 않은 풀이는 목적지에서 시작해 BFS를 하며 위아래로 이동할 때만 돌멩이를 배치하는것입니다. 두번째 실행은 거의 같습니다.
- ✓ 단, 두번째 실행에서 방향 설정에 유의해야 합니다. 단순히 한쪽 방향으로 갔다가 되돌아 온다면 그림과 같이 사이클에 걸릴 수 있기 때문입니다.



- ✓ 일반성을 잃지 않고 먼저 가는 방향을 왼쪽, 아래쪽으로 고정했다고 가정합시다.
- ✓ 사이클이 생기지 않는다면 언젠가는 목적지에 도달할 수 있습니다. 사이클에 주목해봅시다.
- ✓ 사이클이 있다고 가정했을 때 어떤 모양일지 생각해봅시다.

G. 미로 챌린지

- 만약 사이클이 존재한다면 그림과 같이 오른쪽으로 가는 길의 왼쪽과 위로 가는 길의 아래쪽은 벽으로 막혀있어야 합니다. 하지만 그런 상황에서는 BFS를 통해 사이클이 만들어지는것이 불가능함을 귀류법을 통해 보일 수 있습니다. 자세한 증명은 생략합니다.



H. 치터찾기

`seg_tree, two_pointer`
출제진 의도 – **Hard**

- 제출 42번, 정답자 11명 (정답률 26.19%)
- 처음 푼 사람: **greedev**, 17분
- 출제자: pyb1031

- 다음 두 조건을 만족하는 순서쌍 (l, r) 을 찾는 문제입니다.
  - 조건 1: 구간 $[l, r]$ 에 있는 피돌이들의 주장이 구간 $[l, r]$ 과 겹치지 않아야 한다.
  - 조건 2: 구간 $[l, r]$ 에 있지 않은 피돌이들의 주장이 구간 $[l, r]$ 와 겹쳐야 한다.
- 잘 생각해보면 치터 피돌이들이 많아질수록 조건 1은 만족하기 어려워지고 조건 2는 만족하기 쉬워짐을 알 수 있습니다.

- ✓ l을 고정하고 두 조건을 만족하는 r이 있는지 찾아보는 방법을 쓸 수 있습니다.
- ✓ l을 고정했을 때 조건 1을 만족하게 하는 r의 값 중 가장 큰 것을 $f(l)$, 조건 2를 만족하게 하는 r의 값 중 가장 작은것을 $g(l)$이라고 정의합시다. 그런 r이 없는 경우는 일단 생각하지 않습니다.
- ✓ $l \le i \le f(l)$인 모든 i에 대해 (l, i)이 조건 1을 만족하고 $g(l) \le i \le N$인 모든 i에 대해 $f(l, i)$이 조건 2를 만족합니다.
- ✓ 즉 $f(l)$을 구한 후 $(l, f(l))$가 조건 2를 만족하는지만 검사해주면 됩니다.

- ✓ (l, r) 이 조건 1을 만족할 때 $(l+1, r)$ 도 조건 1을 만족합니다.
- ✓ $(l, f(l))$ 가 조건 1을 만족하므로 $(l+1, f(l))$ 도 조건 1을 만족하고 $f(l) \leq f(l+1)$ 이 성립합니다.
- ✓ 따라서 l 의 값을 1 부터 N 까지 늘려가며 r 의 값을 $f(l)$ 로 맞추는 투포인터가 가능합니다.
- ✓ 이제 구간들을 세그먼트 트리 또는 멀티셋과 같은 자료구조를 이용해 관리하면 문제를 풀 수 있습니다.

I. 낭만고양이

`sqrt_decomposition`, `coordinate_compression`, `two_pointer`, `queue`
출제진 의도 – **Challenging**

- 제출 40번, 정답자 0명 (정답률 -%)
- 처음 푼 사람: -
- 출제자: `hamuim`

- 모든 점을 x 좌표가 같은것끼리 모아 $X_1, X_2, X_3 \cdots$ 의 집합이 만들어졌다고 생각하겠습니다.
- 정사각형의 왼쪽과 오른쪽변은 y 축과 평행하므로 이는 X_i 의 서로 다른 두 점을 고르는거로 생각할 수 있습니다.
- X 들 중 집합의 크기가 $sqrt(N)$ 보다 큰 집합을 big, 작은 집합을 $small$, 모든 집합을 합쳐 all 이라 하겠습니다.
- 문제를 둘로 나누어 (big, all) 끼리 $(small, small)$ 끼리 보아 모든 정사각형을 찾아 봅시다.

- 먼저 해야할게 하나 있는데 모든 점들을 좌표 압축 해주고 x, y 축과 평행한 방향으로 같은 색깔의 점이 연속적으로 존재하는 구간의 시작점과 길이를 저장한 배열인 $range$ 를 계산해 줍니다.
- (big, all) 을 먼저 살펴봅시다.
- big 에 속하는 X_i 를 하나 선택하고 크기 N 의 배열 $index$ 에 X_i 내의 좌표 압축된 y 좌표마다 X_i 에서 어떤 점이 있는지를 저장합니다.
- 이제 all 에서 X_i 로 선택한적이 없는 X_j 를 하나 고르게되면 X_i 와 X_j 의 x 좌표가 x_i, x_j 일때 한 변의 길이가 $|x_i - x_j|$ 이고 각 집합에서 2 개의 꼭짓점을 가지는 정사각형을 찾을 수 있습니다.

- ✓ y 좌표 순서로 정렬된 X_j 에서 두 포인터로 $|x_i - x_j|$ 만큼의 구간을 계속 유지해 줍니다.
- ✓ 구간 내의 첫번째 점과 마지막 점이 정사각형의 꼭짓점이 되어야하므로 y 좌표 차이가 $|x_i - x_j|$ 이어야 하며 $Index$ 배열을 통해 X_i 에서 해당 y 좌표를 가지는 꼭짓점으로 가지는 점을 알아낼 수 있습니다.
- ✓ 이제 알아낸 정사각형의 4 개의 꼭짓점에서 $Range$ 배열을 통해 점들이 서로 구간 내에 존재하는지를 확인해 정사각형의 테두리에 위치한 모든 점들의 색깔이 같은지를 확인할 수 있습니다.
- ✓ big 에 해당하는 집합의 개수가 많아봐야 $sqrt(N)$ 개이고 all 의 모든 점들을 최대 한번까지 확인하므로 (big, all) 에서의 시간복잡도는 $\mathcal{O}\left(Nsqrt(N)\right)$ 입니다.

- ✓ $(small, small)$ 을 볼건데 $small$ 에 해당하는 점들을 y 좌표가 같은것끼리 모아 Y_1, Y_2, Y_3 의 집합이 만들어졌다고 생각하겠습니다.
- ✓ Y_i 를 하나 선택해 Y_i 내의 점을 x 좌표가 작은 순서대로 $P_1, P_2, P_3 \cdots$ 라 하겠습니다.
- ✓ P_1 부터 차례대로 보면서 P_j 와 x 좌표가 같은 X 에서 y 좌표가 P_i 보다 더 큰 점중에 P_j 의 $Range$ 범위 안에 있는 점 T_j 를 하나 고르겠습니다.
- ✓ T_j 의 좌표압축된 y 에 위치한 큐 Q_{T_j} 에 저장해가면서 P_j 와 T_j 를 오른쪽 변으로 가지는 정사각형을 모두 찾아 봅시다.

- 현재 정사각형 오른쪽 변의 두 점 P_j, T_j와 두 점의 y좌표 차이이자 한변의 길이인 L이 결정되었으므로 왼쪽 변의 두 점만 알게된다면 정사각형을 찾을 수 있습니다.
- Q_{T_j}에는 이전 $k < j$인 P_k들에서 고른 T_k에 대한 정보가 x좌표의 오름차순으로 들어가 있습니다.
- Q_{T_j}에서 점을 하나씩 빼면서 Q_{T_j}의 제일 앞에 들어있는 점인 F와 T_j의 x좌표 차이가 L이 되게 해봅시다.
- 만약 F와 T_j의 x좌표 차이가 L이라면 정사각형의 세 점이 결정되며 나머지 한 점은 F와 x좌표가 같으며 Y_i에 속한 점이어야하는데 애초에 F를 큐에 넣기 위해서는 방금 조건을 만족하는 점이 존재할수밖에 없습니다.

- ✓ 앞서 T_j 는 P_j 의 $Range$ 번위 안에서 고른다고했기때문에 정사각형의 왼쪽 변과 오른쪽 변 위의 모든 점의 색깔은 같습니다.
- ✓ 그러므로 T_j 와 P_j 의 x 축과 관련된 $Range$ 구간이 왼쪽으로 길이 L 이상인지만 확인하면 정사각형의 테두리에 위치한 모든 점들의 색깔이 같은지를 확인할 수 있습니다.
- ✓ Y_i 의 모든 점은 $small$ 에 속하므로 P_j 마다 확인해야하는 T_j 의 개수가 $sqrt(N)$ 보다 작기때문에 $(small, small)$ 에서의 시간복잡도는 $\mathcal{O}\left(Nsqrt(N)\right)$ 입니다.
- ✓ 총 시간복잡도는 $\mathcal{O}\left(Nsqrt(N)\right)$ 입니다.