shake!

2017 예선문제 풀이

A. 경인지역 6개대학 연합 프로그래밍 경시대회

출제자: 김현정
작성자: 김현정

- Small/Large:
- N개의 데이터에 대해서
- S가 가장 크면서
  - 그 중 C는 가장 작고
    - 그 중 L이 가장 작은
- 위 조건에 맞는 한 명을 비교 연산을 통해 구한다.

- 시간복잡도: O(N)

shake!

B. 동방 프로젝트

출제자: 서병찬
작성자: 서병찬

- Small:
- check 배열을 만들고 행동마다 [x, y)의 상태를 1로 만든다.
- 모든 행동이 끝나고
- [1, N]에서 1이 아닌 방의 개수가 답이 된다.

- 시간복잡도: O(M * N)

B. 동방 프로젝트

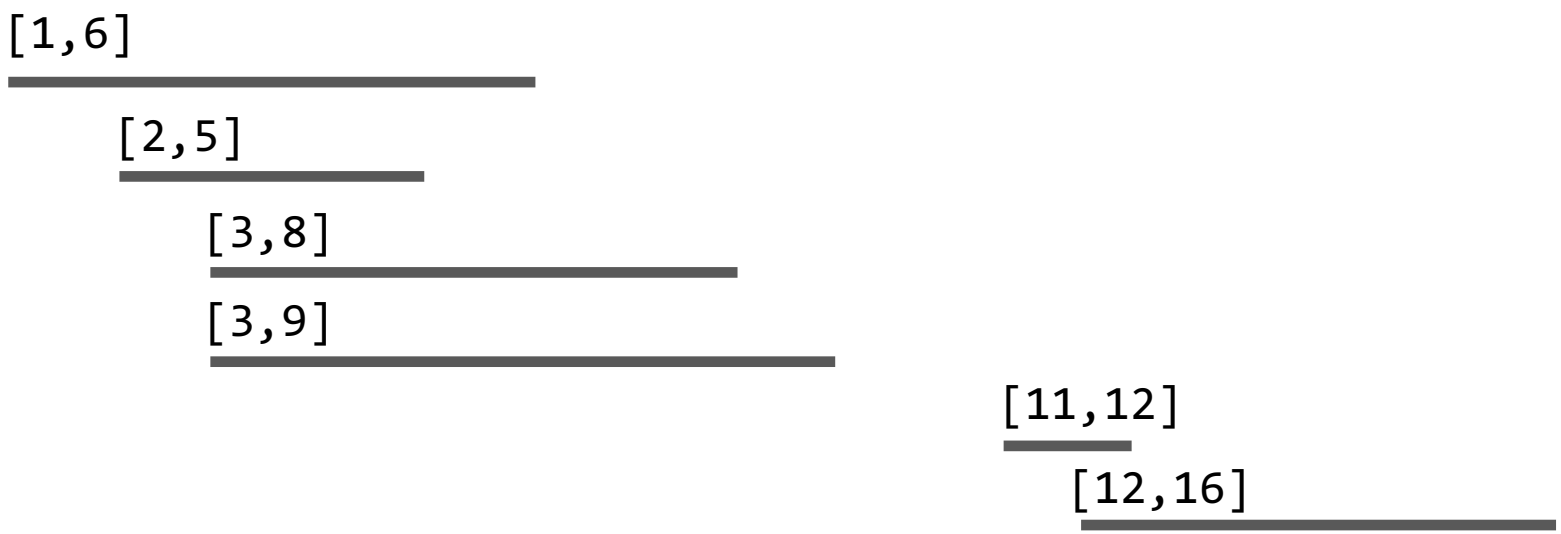
출제자: 서병찬
작성자: 서병찬

- Large:
- 각 행동을 x기준으로 정렬한다.
- 가장 마지막 위치를 가리키는 변수 r을 두고
- x기준으로 정렬된 행동의 y를 보면서
- r이 y보다 작으면 갱신하면서
- 하나로 합쳐진 방의 크기를 조사한다.

B. 동방 프로젝트

출제자: 서병찬
작성자: 서병찬

- 아래와 같이 행동할 때
- [1,6] ~ [3,9]에서 r은 6->8->9로 갱신되고,
- [11,12]에서 x보다 r값이 작아지면 그 이전까지가 한 방이 된다.

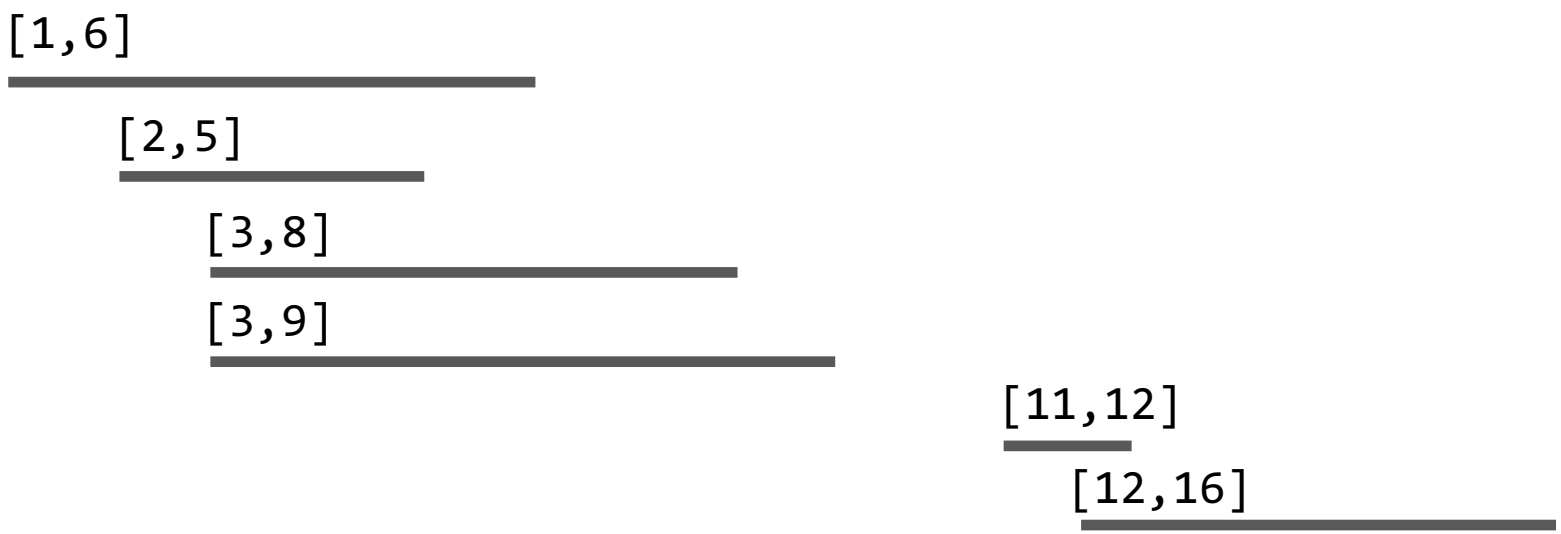


shake!

B. 동방 프로젝트

출제자: 서병찬
작성자: 서병찬

- 이후 [11,12] ~ [12,16]도 같은 과정을 거쳐 한 방이 되고
- [1,9], [11,16]이 한 방인것을 알았으므로
- 전체 방 개수에서 위 구간에 합쳐진 방의 개수를 빼면 답이 된다.



shake!

B. 동방 프로젝트

출제자: 서병찬
작성자: 서병찬

- 시간복잡도: O(M * logM)
  - x를 기준으로 정렬: O(M * logM)
  - r을 갱신하며 합쳐진 방의 개수를 구함: O(M)

- Disjoint-set 자료구조를 이용해도 풀 수 있다.

C. Quilting

출제자: 김동이
작성자: 김현정

- Small:
- 각 줄에 한 칸씩 고르는 모든 경우의 수를
- 재귀적으로 다 볼 수 있다.
- 모든 경우 중 최소의 부자연스러운 정도

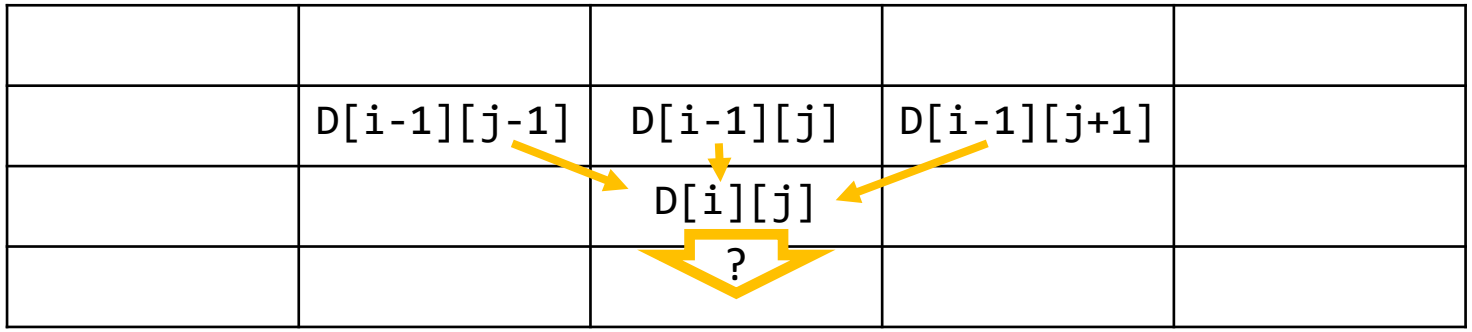
- 시간복잡도: $O(W * 3^H)$
  - 첫 줄의 시작 칸을 고르는 경우의 수 W
  - 다음 줄의 칸을 고르는 방법: 3가지

shake!

C. Quilting

출제자: 김동이
작성자: 김현정

- Large:
- D[i][j]: (i,j)가 경계일때 i번째줄까지 부자연스러운 정도의 최소값
- 이라고 정의하면 경계선 이동 원칙에 따라
- D[i][j]는 D[i-1][j-1], D[i-1][j], D[i-1][j+1]
- 이 세가지 상태 이후에 올 수 있다.



shake!

C. Quilting

출제자: 김동이
작성자: 김현정

- $D[i][j] = \min_{dw = -1 \text{ to } 1}(D[i-1][j+dw]) + (A[i][j] - B[i][j])^2$
- 답: $\min_{j = 1 \text{ to } W}(D[N][j])$
  - 마지막줄 경계 중 어디에서 끝나도 상관 없으므로,
  - 마지막줄 경계 중 최소의 부자연스러운 정도

- 시간복잡도: $O(W * H)$

shake!

D. 인공지능 테트리스

출제자: 김동이
작성자: 김동이

- 7가지 도형과 4방향, 총 28가지 경우의 수를 미리 4x4격자 배열 혹은 비트로 정의해두자.
  - 격자가 남을 경우 도형을 왼쪽 상단으로 몰아둔다.
  - 회전은 배열로도 처리할 수 있다.
- 정의한 도형들도 리스트로 관리하면 이후에 수 많은 if문을 줄일 수 있다.

1	0	0	0
1	0	0	0
1	1	0	0
0	0	0	0

1	1	0	0
0	1	0	0
0	1	0	0
0	0	0	0

1	1	0	0
1	1	0	0
0	0	0	0
0	0	0	0

1	0	0	0
1	0	0	0
1	0	0	0
1	0	0	0

shake!

D. 인공지능 테트리스

출제자: 김동이
작성자: 김동이

- 테트리스 게임판도 0, 1로 이루어진 2차원 배열로 정의할 수 있다.
  - 게임판의 가장자리를 만들어 1로 채워두면 이후에 처리가 편함
  - 0은 아무 블럭도 없는 비어있는 칸
  - 1은 가장자리 혹은 블럭이 이미 놓여있는 칸

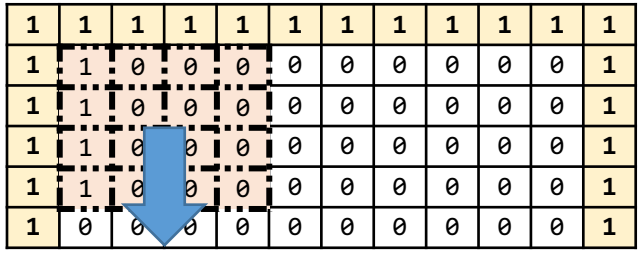
1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
1	1	1	1	0	0	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1

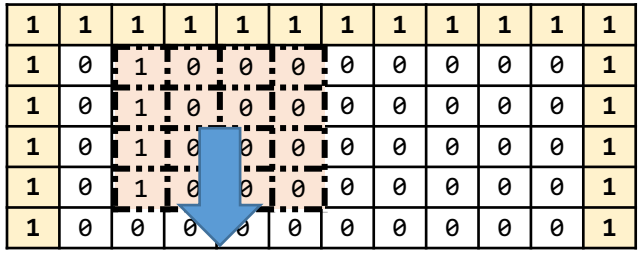
D. 인공지능 테트리스

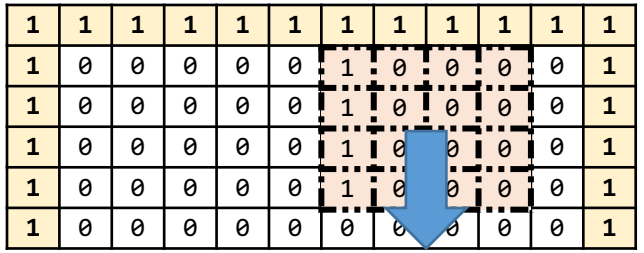
출제자: 김동이
작성자: 김동이

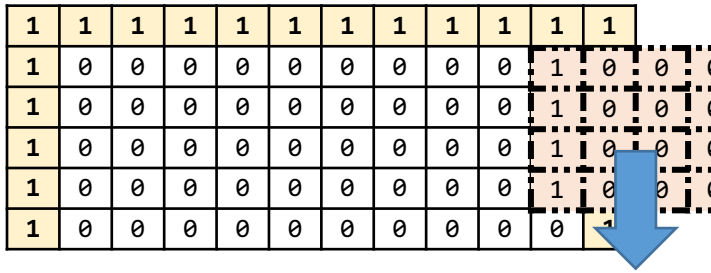
• Small : 각 도형마다 아래 과정을 반복한다.

1. 도형이 담긴 격자의 가장 왼쪽 열을 게임판의 모든 열에 놓아본다.
    1. 게임 판과 가장자리에서 충돌한다면 건너뛴다.
    2. 게임판의 가장 윗쪽에 두고 시작한다.
2. 이후 도형을 한 칸씩 계속 내려본다.
    1. 더 이상 내리게 되면 게임판과 격자의 1이 겹치게 된다면 그만 내린다.
    2. 내리는 걸 멈췄을 때 게임판에서 1로 가득찬 행의 수를 계산한다.
    3. 이 행의 수가 얻을 수 있는 점수가 되므로 최대값을 갱신한다.
3. 이 과정을 모든 열과 도형에 대해 수행한다.
    1. 그 과정에서 얻을 수 있던 가장 큰 점수를 출력한다.









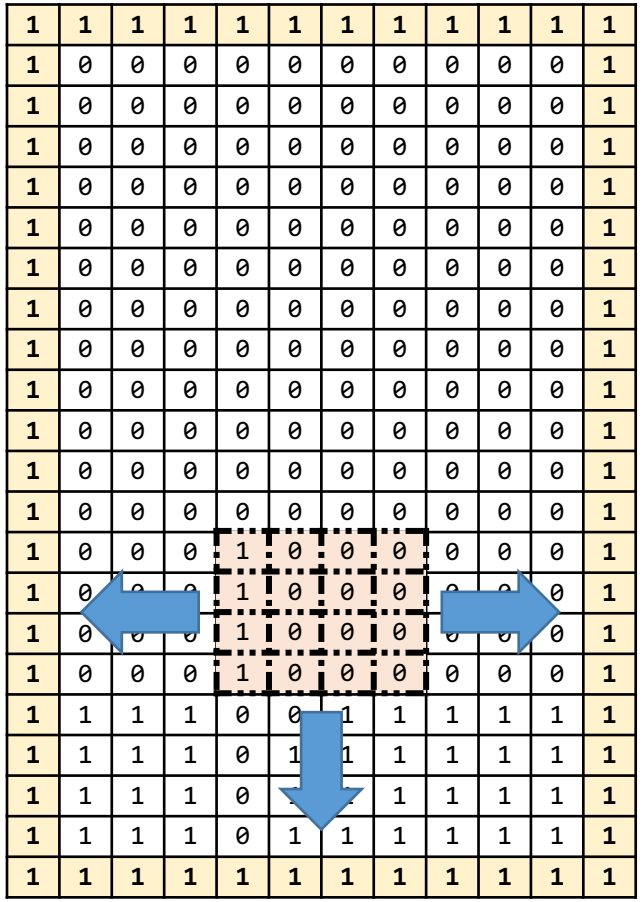
shake!

D. 인공지능 테트리스

출제자: 김동이
작성자: 김동이

- Large : 각 도형에 대해 아래 과정을 수행한다.

1. 선택한 도형 격자가 게임판과 1이 겹치지 않게 가장 윗줄에 배치한다.
    1. 이 문제는 항상 가장 위 네 줄이 비어있기 때문에 어떤 열도 가능하다.
2. 격자를 BFS를 사용해 여러 방향으로 옮겨보며 아래 과정을 반복한다.
    1. 격자와 게임판의 1이 겹치지 않게 격자를 아래로 이동시킬 수 없다면
        1. 현재 위치에서 격자를 고정했을 때 1로 가득 차는 행의 수를 계산한다.
        2. 이때 그런 행의 수가 제거할 수 있는 줄의 수, 즉 얻을 수 있는 점수이다.
        3. 얻을 수 있는 최대 점수를 갱신한다.
    2. 격자와 게임판의 1이 겹치지 않게 아래, 왼쪽, 오른쪽으로 이동 가능하면
        1. 이동가능한 각 방향으로 이동한 상태를 탐색할 수 있게 BFS로 탐색한다.
        2. 한번 본 위치를 또 보지 않도록 탐색위치를 체크해둔다.
3. 다음 도형에 대해 위의 과정을 또 수행한다.

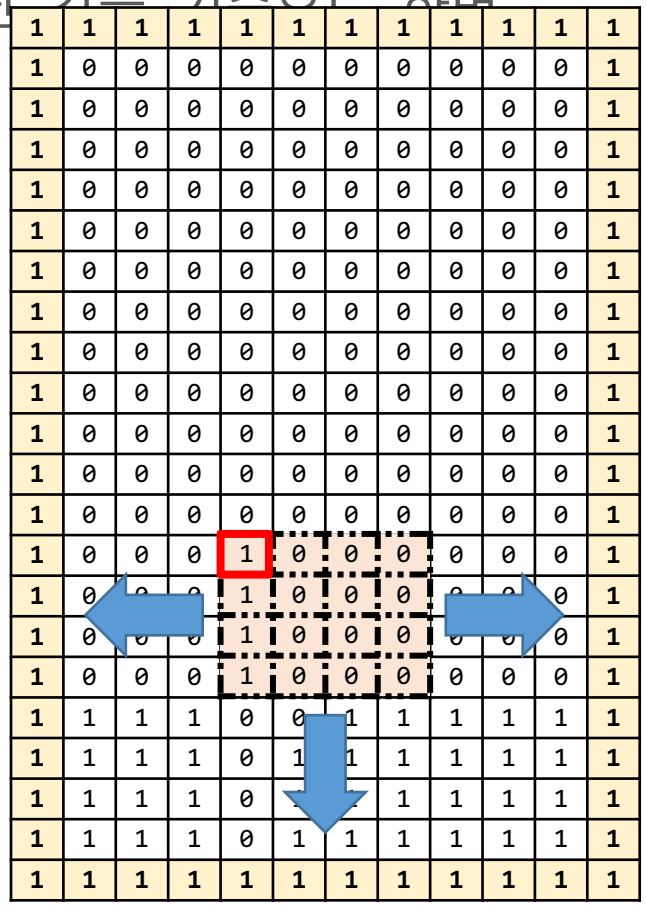


shake!

D. 인공지능 테트리스

출제자: 김동이
작성자: 김동이

- [Tip]
- 각 격자의 위치를 기억하거나 게임 판과의 겹침을 판단할 때 왼쪽 상단 칸을 기준으로 하면 편리하다.
  - 앞서 격자 속 도형을 왼쪽 상단으로 정렬했기 때문에 놓치는 경우가 없다.
  - 왼쪽 상단 칸 기준으로 아래와 오른쪽으로 4칸씩 보면 된다.
- 제거 될 행의 검사도 격자가 포함된 네 줄만 수행하면 된다
  - 격자가 영향을 주지 않는 행이 제거 될 리 없다.



shake!

E. 소금과 후추

출제자: 이승철
작성자: 김현정

- Small:
- (1,1) ~ (M,N)을 차례로 보면서
- (i,j)를 왼쪽 위로 하는 각 변의 길이가 W인 정사각형 구역에서
- 구역에 있는 모든 수를 모은 뒤 정렬한다.
- 이때 중앙에 있는 값을 출력한다.

E. 소금과 후추

출제자: 이승철
작성자: 김현정

- Small:
- 시간복잡도: $O(N * M * W^2 * \log W)$
  - (1, 1) ~ (M, N): $O(N * M)$
  - W X W에 있는 점 모으기: $O(W^2)$
  - 모은 점 정렬하기: $O(W^2 * \log W)$
  - 시간복잡도: $O(N * M) * (O(W^2) + O(W^2 * \log W))$

E. 소금과 후추

출제자: 이승철
작성자: 김현정

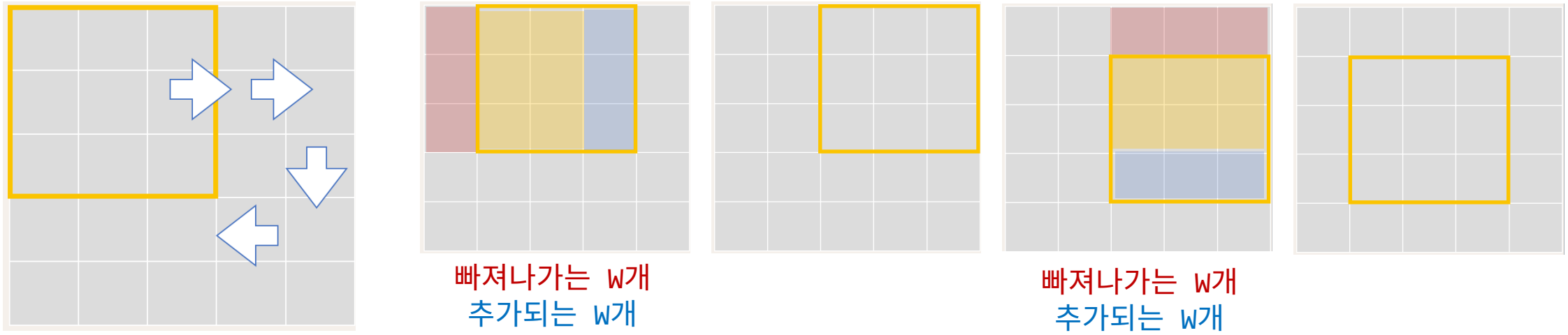
- Large:
- 어떤값 X가 중앙값이 될 수 있는지를 판단한다.
- (i,j)를 왼쪽 위로 하는 W X W 정사각형 안의
- 원소 개수의 누적합을 알 수 있는 자료구조를 만든다
  - 원소 개수의 누적합: k보다 작거나 같은 원소의 개수
  - Segment Tree, Fenwick Tree
- Binary Search를 통해 누적합이 $(W^2+1)/2$가 되는 X를 찾는다.

shake!

E. 소금과 후추

출제자: 이승철
작성자: 김현정

- Large:
- 이때 (i,j) -> (i,j+1)로 갈때 모든 원소를 다시 세는 것이 아니라
- Sliding Window를 통해 빠져나가는 W개, 추가되는 W개만 변화시킨다.



shake!

E. 소금과 후추

출제자: 이승철
작성자: 김현정

- 시간복잡도: O(N * M * W * logK)
  - (1, 1) ~ (M, N): O(N * M)
  - 추가되거나 삭제되는 원소의 개수: 2 * W개
  - 이러한 2 * W개의 추가/삭제, 누적합 갱신: O(W * logK)
  - BinarySearch의 횟수: logK번
  - BinarySearch마다 누적합 구해오기: O(logK)
  - 시간복잡도: O(N * M) * (O(W * logK) + O((logK)^2))

shake!

E. 소금과 후추

출제자: 이승철
작성자: 김현정

- Large:
- 두개의 Heap으로도 풀 수 있다.
- 앞선 풀이와 마찬가지로 한번에 2W개씩 추가/삭제할 때
- 작은 원소를 담아둘 MaxHeap과 큰 원소를 담아둘 MinHeap을 둔다.
- 이때 MaxHeap의 가장 큰 원소가
- MinHeap의 가장 작은 원소보다 작거나 같도록 하면서
- MinHeap의 크기가 항상 MaxHeap보다 1만큼 크거나 같도록 하면
- MinHeap의 가장 작은 원소가 중앙값이 된다.

E. 소금과 후추

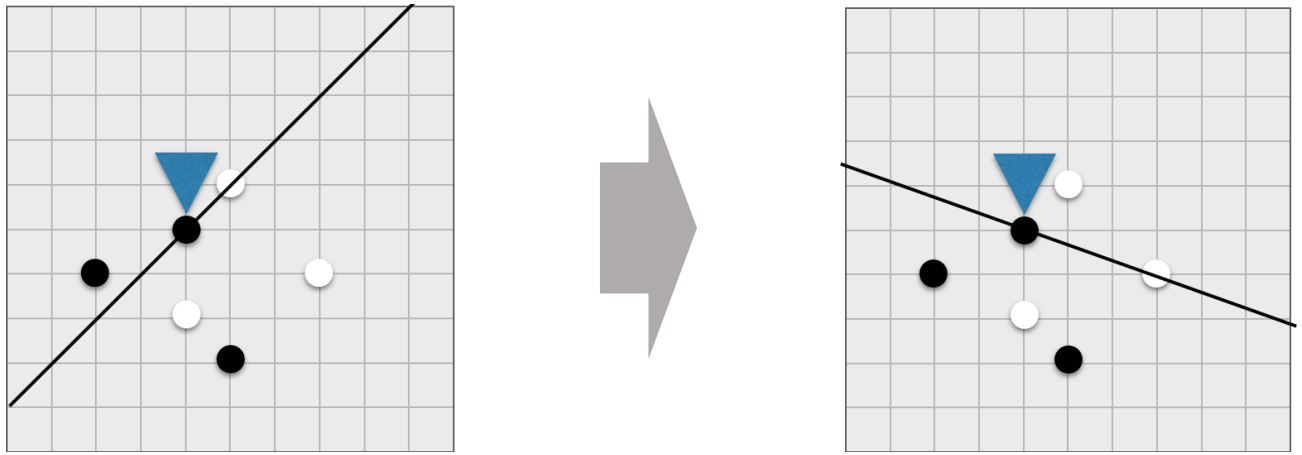
출제자: 이승철
작성자: 김현정

- 시간복잡도: O(N * M * W * logW)
  - (1, 1) ~ (M, N): O(N * M)
  - 추가되거나 삭제되는 원소의 개수: 2 * W개
  - 이러한 2 * W개의 추가/삭제: O(W * logW)

F. Over Fitting

출제자: 김동이
작성자: 서병찬

- Small:
- 임의의 점 A를 지나는 직선을 회전시킨다.
- 회전이란 A를 고른 상태에서,
- 다른 점 B를 찾아 연결하는 것으로 생각해볼 수 있다.



shake!

F. Over Fitting

출제자: 김동이
작성자: 서병찬

- Small:
- 이러한 직선에 대해
- 다른 N-2개의 점을 보면서
- Positive 정점만 속하는 방향(시계/반시계)이 있는지,
- 이때 몇 개의 Positive 정점이 해당 방향에 있는지를 구한다.
- A, B중 Positive 정점이 있다면 각도를 조금 틀거나 평행이동을 통해 같은 방향에 속하게 할 수 있다.

F. Over Fitting

출제자: 김동이
작성자: 서병찬

- Small:
- 모든 A, B쌍으로 얻어지는 직선에 대해
- 얻게 된 Positive 정점 수의 최대값이 답이 된다.

- 시간복잡도: $O(N^3)$
  - 두 개의 점: $O(N^2)$
  - 두 개의 점으로 이루어진 직선에 대해 다른 모든 점을 조사: $O(N)$

F. Over Fitting

출제자: 김동이
작성자: 서병찬

- 직선을 고를때 A, B의 조합은 아래의 세가지다.
- Positive-Positive: 직선을 평행이동하여 두 점을 Positive 방향에 속하게 한다.
- Positive-Negative: 직선을 회전하여 각 점을 해당하는 방향에 속하게 한다.
- Negative-Negative: 직선을 평행이동하여 두 점을 negative 방향에 속하게 한다.

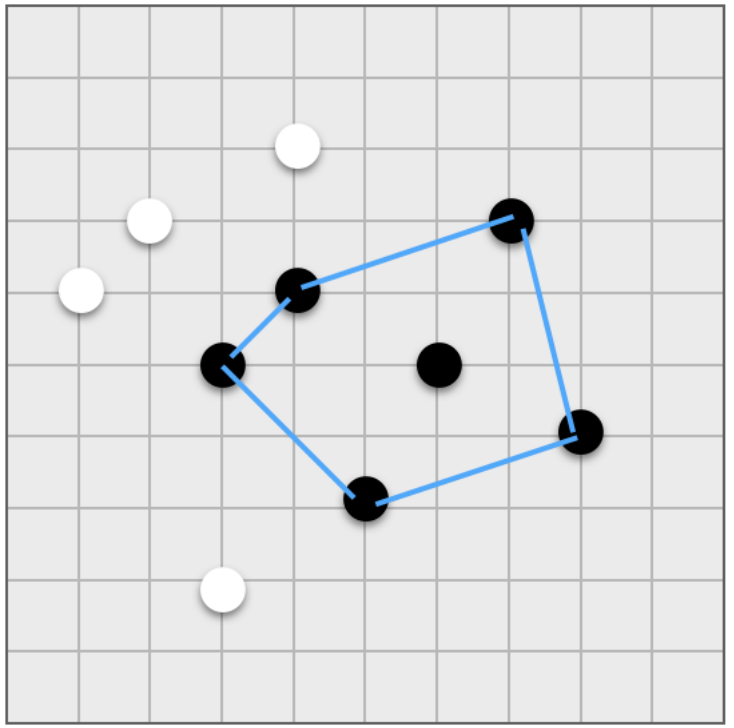
- 이때 Positive-Positive, Negative-Negative 직선을 회전하면
- 항상 Positive-Negative의 경계로 근사할 수 있으므로
- Positive-Negative 직선만 고려해도 된다.

shake!

F. Over Fitting

출제자: 김동이
작성자: 서병찬

- 직선을 만들 때 고려해야할 Negative 점은
- Negative점만을 고려했을 때 최외곽에 있는 점들이다.
- 내부에 있는 점과 Positive점을 이은 직선은
- 외곽의 Negative-Positive 직선을
- 회전한 것과 같기 때문이다.



- => Negative 정점의 컨벡스헐을 구한다.

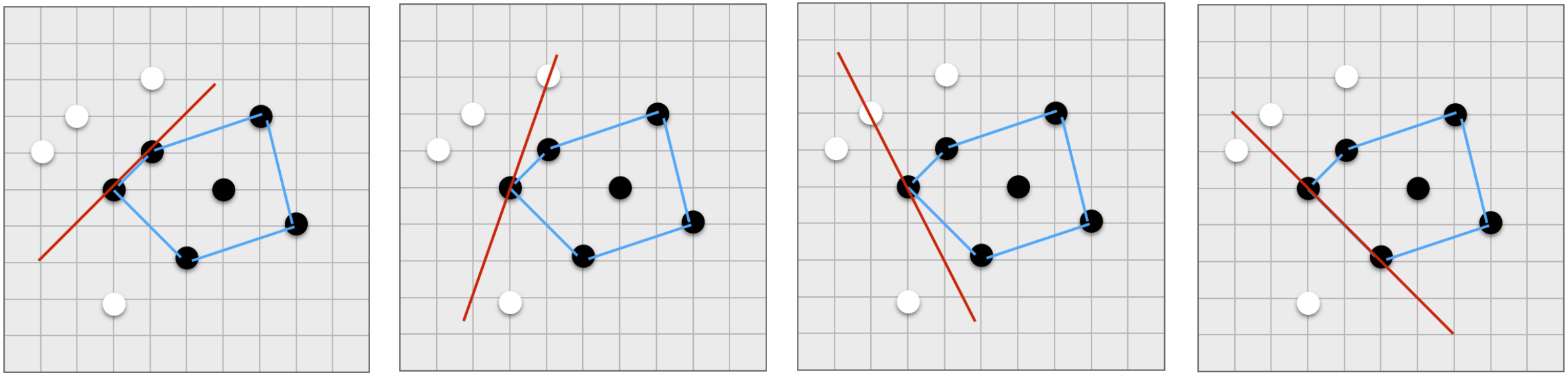
shake!

F. Over Fitting

출제자: 김동이
작성자: 서병찬

- 컨벡스 헐 각 변에 대해 Positive 정점들을
- 해당 변과 이루는 각도를 기준으로 정렬한다.
- 이후 각도 정렬된 순서대로 직선을 회전한다.

반시계방향 회전의 예시



shake!

F. Over Fitting

출제자: 김동이
작성자: 서병찬

- 직선이 시계/반시계방향으로 돌아간다고 할 때,
- 시계/반시계방향으로 움직일 때 가장 앞에서 Positive 정점이 삭제되고
- 가장 뒤에서 Positive 정점이 새로 추가된다.
- Two Pointer를 이용하면 O(N)만에 모든 직선에 대한 Positive 정점의 개수를 구할 수 있다.

- 시간복잡도: $O(N^2 * logN)$
  - 컨벡스헐 변의 개수 N개에 대해
  - 각 변에 대해 정렬: O(N * logN)
  - 이후 추가/삭제: O(N)

shake!