코드포스 3주차 문제 풀이

문제 난이도

	문제 번호	문제 이름	난이도
A	9047	6174	B1
B	5576	콘테스트	B2
C	29757	트리 긋기	S1
D	20117	호반우 상인의 이상한 품질 계산법	S1
E	13884	삭삽 정렬	G5
F	17140	이차원 배열과 연산	G4
G	3008	직각 삼각형의 개수	P5
H	3662	보석 분배	P1

A. 6174

구현, 문자열, 정렬, 시뮬레이션

A. 6174

- ✓ 숫자가 주어지면 가장 큰 수로 정렬시키고 가장 작은 수로 정렬시켜서 두 숫자의 차이를 뺍니다. 이것을 반복시키면 6174에 도달하게 되는데, 얼마나 반복을 해야하는지 카운트하는 문제입니다.

- ✓ 시간복잡도 : $O(T)$

B. 콘테스트

구현, 정렬

B. 콘테스트

- ✓ W대학과 K대학이 각각 10줄씩 주어집니다. 이 10개의 데이터를 정렬하여 상위 데이터 3개씩 추출하고 더하여 출력하면 정답입니다.

- ✓ 시간복잡도 : $O(1)$

C. 트리 긋기

정렬, 애드 혹

C. 트리 긋기

- 트리를 그을 때 선분이 교차하면 안되기 때문에, x축 기준 또는 y축 기준으로 정렬시켜서 순서대로 이어주면 정답입니다.

- 시간복잡도 : $O(NlgN)$

D. 호반우 상인의 이상한 품질 계산법

그리디 알고리즘, 정렬

D. 호반우 상인의 이상한 품질 계산법

- 호반우를 묶고 묶어진 호반우의 중앙값 * 묶음의 크기로 가격을 재조정 하고 있는데, 두 개씩 묶는 경우 가장 큰 값*2가 묶음의 가격입니다.
- 즉, 호반우 가격을 정렬시킨 후 (1, N), (2, N - 1), (3, N - 2) ... 식으로 묶고 호반우 갯수가 홀수인 경우 중간에 있는 값을 더해주면 정답입니다.
- 시간복잡도 : $O(NlgN)$

E. 삭삽 정렬

그리디 알고리즘, 정렬

E. 삭삽 정렬

- ✓ 원소를 하나 빼서 맨 뒤에 삽입하기 때문에, 오름차순으로 정렬하는 것이 가장 좋습니다.
- ✓ 자신보다 작은 원소가 뒤에 있는 경우 정렬하기 위해서 무조건 빼야 합니다. 빼야 하는 원소들을 모두 카운트 합니다.
- ✓ 뺀 것 중에 가장 작은 것이 배열에 어디에 들어가야 할 지 찾습니다. 찾은 위치로부터 뒤에 있는 원소들은 다 빼야 하기 때문에, 뒤에 있는 원소도 카운트 해주면 정답입니다.
- ✓ 시간복잡도 : $O(NlgN)$

F. 이차원 배열과 연산

구현, 정렬, 시뮬레이션

F. 이차원 배열과 연산

- ✓ [3, 1, 1, 2] -> 1이 2개, 2가 1개, 3이 1개. -> [2, 1, 3, 1, 1, 2]
- ✓ 카운트 수 기준 오름차순, 카운트 수가 같으면 수의 크기 기준 오름차순 정렬입니다.

- ✓ 행의 개수 >= 열의 개수인 경우(배열이 세로로 긴 경우)엔 각 행을 기준으로 정렬시키고, 아닌 경우엔 각 열을 기준으로 정렬시킵니다.

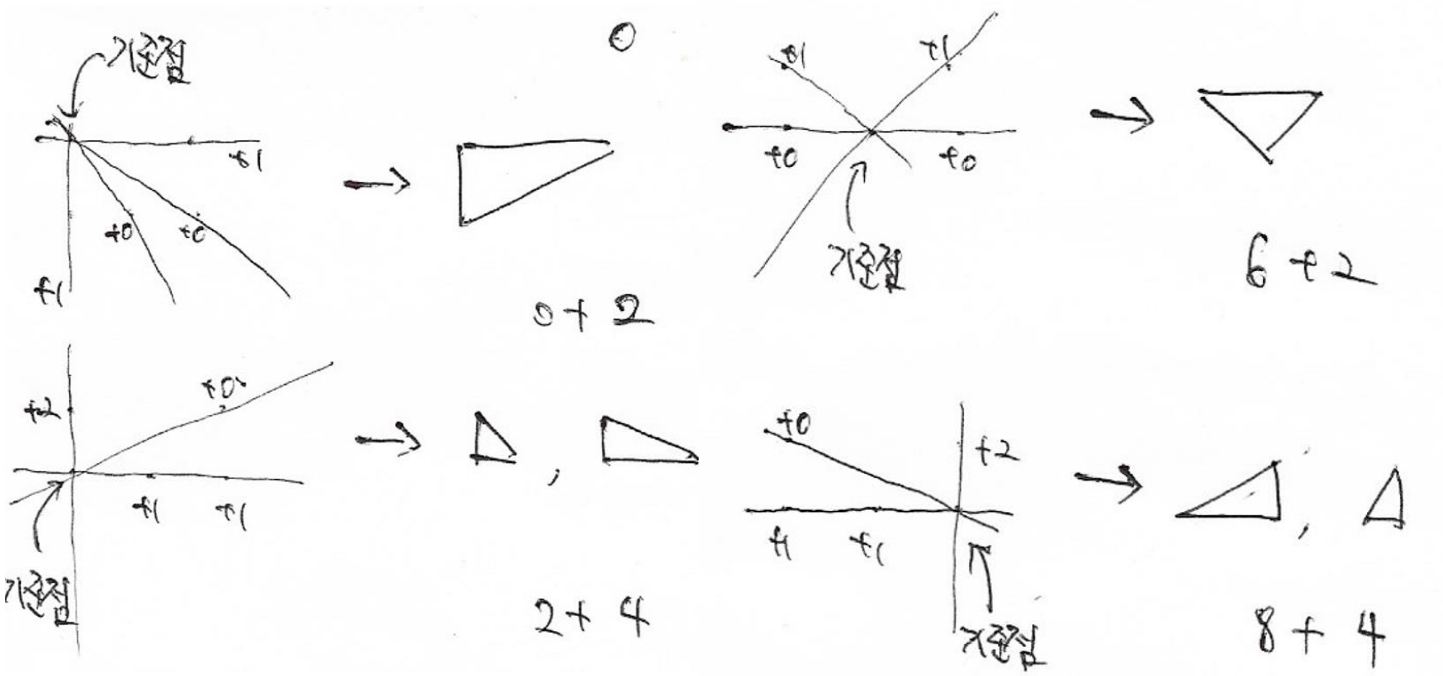
- ✓ 정렬을 시키고 크기가 줄어들 수 있기 때문에 주의해야 합니다.
- ✓ 정렬시키고 r행 c열에 k가 있으면 그 때의 시간을 출력하면 정답입니다.
- ✓ 시간복잡도 : $O(RC)$

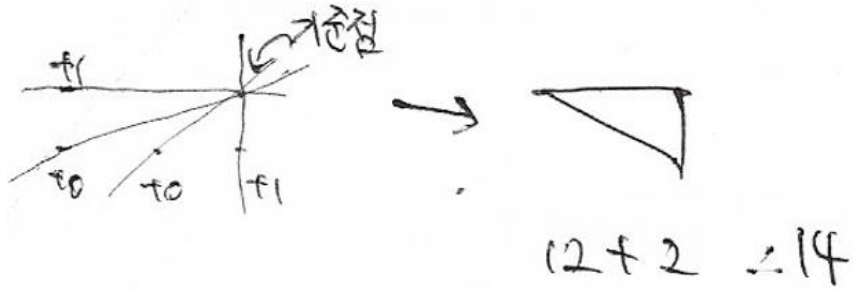
G. 직각 삼각형의 개수

자료구조, 정렬, 기하학, 이분 탐색, 해시를 사용한 집합과 맵

G. 직각 삼각형의 개수

- ✓ N이 최대 1500 이므로 O(N^2) 풀이가 가능합니다. 한 점을 기준으로 모든 점을 차례로 탐색하여 나올 수 있는 직각삼각형을 찾아야 합니다.
- ✓ 다음은 예제 3번을 가져온 것입니다.

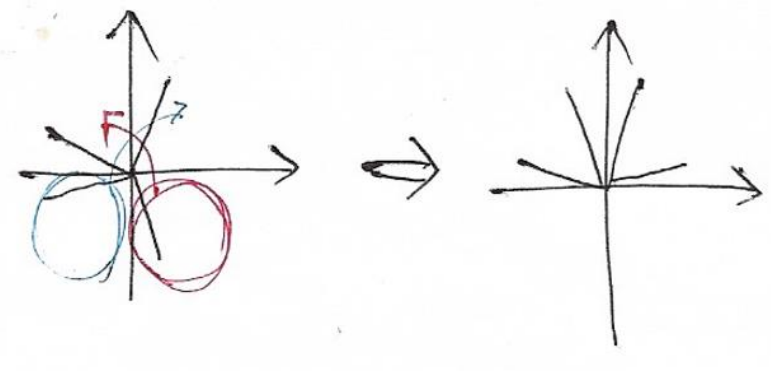




- ✓ 삼각형이 각각 두 번 세어졌기 때문에 14/2 = 7이 정답입니다.

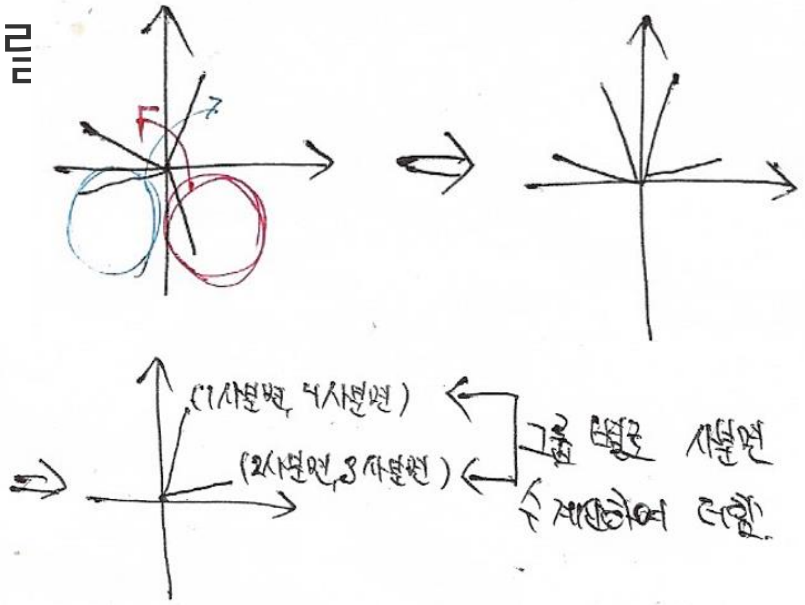
G. 직각 삼각형의 개수

- ✓ 기준점으로 부터의 기울기를 모아주어 시간을 단축시키는 과정이 필요합니다. 이 때 두 가지 방법이 있습니다.
- ✓ 1. 기울기를 약분하여 해시맵에 저장하는 방법
  - ✓ 제 3, 4분면에 있는 것을 제1, 2분면에 모아서
  - ✓ 직각 삼각형이 있는지 탐색하는 방법입니다.
- ✓ 시간복잡도 : $O(N^2)$



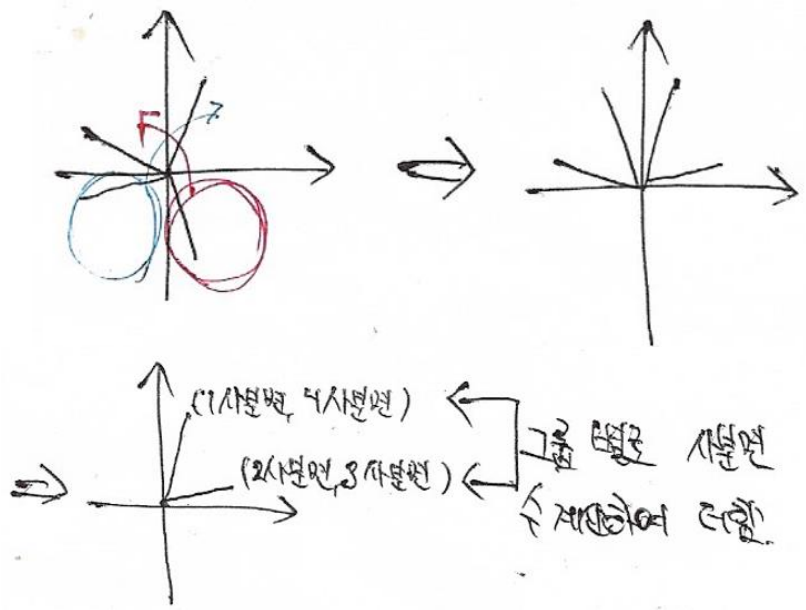
G. 직각 삼각형의 개수

- ✓ 2. 기울기를 한 사분면에 모아놓고 정렬시키는 방법
  - ✓ 1사분면에 모두 모아놓고 기울기 기준으로 정렬
  - ✓ 그룹별로 계산합니다.
  - ✓ 1사분면의 선분 수 * 2사분면의 선분 수 +
  - ✓ 2사분면의 선분 수 * 3사분면의 선분 수 +
  - ✓ 3사분면의 선분 수 * 4사분면의 선분 수 +
  - ✓ 4사분면의 선분 수 * 1사분면의 선분 수



G. 직각 삼각형의 개수

- ✓ 그룹별로 연산한 것을 다 더하면 정답입니다.
- ✓ 이 경우, 두 번 카운트 되지 않았기 때문에 2로
- ✓ 나눌 필요가 없습니다.
- ✓ 시간복잡도 : $O(N^2 lgN)$



H. 보석 분배

다이나믹 프로그래밍, 자료 구조, 그리디 알고리즘, 정렬, 우선순위 큐

H. 보석 분배

- ✓ 민혁이(Petra)는 자신한테 가장 좋은 보석을 고릅니다. 그런 보석이 여러 개면 민규(Jan)에게 가장 안 좋은 보석을 고릅니다.
- ✓ 민규(Jan)은 최종적으로 자신한테 유리한 선택을 합니다. 그러한 선택이 여러개일 경우, 민혁이(Petra)에게 가장 이득이 되게끔 선택합니다.
- ✓ 두 번째 예제에서 처음에 Petra가 (10, 1)을 가져갑니다. 그 다음 Jan은 (6, 6)을 가져가고 Petra가 (4, 4), Jan이(1, 10)을 가져갑니다.
- ✓ 만약 Jan이 처음에 (6, 6)이 아닌 (1, 10)을 가져갔다면, Petra가 (6, 6)을 가져가게 되어 최종적으로 Petra = 16, Jan = 14를 가져가게 되어 Jan에겐 최선의 선택이 아니게 됩니다.

H. 보석 분배

- ✓ 이 문제의 접근 방식으로 두 가지를 고려해 볼 수 있습니다.
- ✓ 1. 다이나믹 프로그래밍 $O(TN^2)$
  - ✓ Petra에게 좋은 기준으로 정렬합니다.
  - ✓ dp[n][m][k] -> n : 현재 인덱스, m : 현재 고른 보석 갯수, k : 현재 인덱스에서 보석을 선택했는지 여부(0, 1)
  - ✓ dp[n][m][0] = max(dp[n - 1][m][0], dp[n - 1][m][1])
  - ✓ dp[n][m][1] = max(dp[n - 1][m - 1][0], dp[n - 1][m - 1][1])
  - ✓ 이렇게 정의할 수 있습니다.
  - ✓ Petra가 선인경우, n은 1부터 시작, Jan이 선인경우, n은 0부터 시작입니다.

H. 보석 분배

- ✓ 여기서 dp는 pair로 해볼 수 있는데, max()는 pair의 second값(Jan의 보석 가치) 기준 더 큰 것, 같다면 pair의 first값(Petra의 보석 가치) 기준 더 작은 것을 리턴합니다.
- ✓ Jan이 최대한 고를 수 있는 수가 정해져있습니다. N이 짝수면 N/2, N이 홀수고 Jan이 선이면 N/2 + 1, Petra가 선이면 N/2 입니다.
- ✓ Jan이 최대한 고른 것들 중 max()를 찾습니다. 그렇다면 pair형태로 나올 것인데, first값은 최대한 작은 것, second값은 최대한 큰 것을 고르게 됩니다.
- ✓ 처음에 데이터를 입력 받을 때 미리 Petra가 고를 값을 모두 더해주고, 이 값에 first값을 뺀 값을 출력하고 second값을 그대로 출력하면 정답입니다.

H. 보석 분배

- ✓ 2. 그리디 알고리즘 $O(TNlgN)$
  - ✓ 입력을 받고 마찬가지로 민혁이(Petra)에게 가장 유리하게 정렬합니다.
  - ✓ 배열을 처음부터 선형탐색하면서 우선순위 큐에 원소를 넣습니다.
  - ✓ 이 때 민규의 가치가 낮은 것이, 같다면 민혁이 가치의 높은 것으로 뽑히게끔 우선순위 큐를 설정해야 합니다.
  - ✓ 민혁이(Petra)의 턴일 때 우선순위 큐에서 하나를 뽑아 더해줍니다.

H. 보석 분배

- ✓ 민규(Jan)의 턴일 경우 우선순위 큐에 원소를 push만 하고 턴을 넘깁니다.
- ✓ 선형 탐색이 끝나면 우선순위 큐에 남아있는 것이 민규(Jan)의 몫입니다. 남아있는 값들을 민규(Jan)에게 더해주고 출력해주면 정답입니다.(!!)