단계별 모의고사 6회차 해설

2023년 03월

문제		의도한 난이도	출제자
A	가희야 파일 탐색기	**Medium**	chogahui05
B	가희의 키워드	**Medium**	chogahui05
C	가희와 은행	**Medium**	chogahui05
D	가희와 btd5	**Hard**	chogahui05
E	가희와 수인 분당선 1	**Challenging**	chogahui05
F	가희와 비행기	**Hard**	chogahui05
G	가희와 거북이 인형	**Challenging**	chogahui05
H	가희와 읽기 쓰기 놀이 2	**Challenging**	chogahui05

A. 가희와 파일 탐색기

`implementation`, `sort`

출제진 의도 – **Medium**

- 출제자: chogahui05

A. 가희와 파일 탐색기

- ✓ $sorting$ 문제인 것은 알겠습니다.
- ✓ os 에서 붙은 확장자가 붙은 것이 먼저 나오게 한다는 조건이 어렵습니다.

A. 가희와 파일 탐색기

- 매 정렬 비교 함수마다 OS에 붙은 확장자인지 자료구조에서 검색하는 너무 느립니다. 왜?
  - 비교가 $\mathcal{O}(nlogn)$ 번 일어나기 때문입니다.
  - 그 때 마다 상수가 큰 $\mathcal{O}(1)$ 또는 $\mathcal{O}(logm)$ 의 복잡도를 가진 비교를 합니다.
- 그러면 어떻게 하면 될까요? 미리 전처리 하면 됩니다.

A. 가희와 파일 탐색기

- 파일에 대한 $meta$ 정보를 아래와 같이 저장합니다.
  - 파일명
  - 확장자가 os 에서 인식할 수 있는지 $flag$
  - 확장자
- 이제 정렬 전에 파일의 확장자가 os 에서 인식할 수 있는지 $hash$ 나 $tree$ 에서 찾으면 됩니다.

B. 가희와 키워드

`data structure`, `implementation`, `string`, `parsing`
출제진 의도 – **Medium**

- 출제자: chogahui05

B. 가희와 키워드

- ✓ 가희가 메모장에 있는 k에 대한 글을 쓰면 메모장에 있는 k가 삭제됩니다.
- ✓ 즉, 아래와 같은 알고리즘을 수행하면 됨을 알 수 있습니다.
  - $keywords$를 파싱해서 키워드를 뽑아냅니다.
  - 각 $keyword$마다 메모장에 있는지를 검사합니다.
  - 있으면 제거하면 됩니다.

B. 가희와 키워드

- 그런데 이걸 $brute$ 하게 하면 너무 느립니다. 어떻게 해야 할까요?
- 결국 k 를 찾아야 하는 것이잖아요. 찾기 연산에 최적화된 자료구조는 $hash$ 입니다. 따라서
  - 메모장에 있는 $keywords$ 를 $hash$ 로 관리합니다.
  - 블로그에 k 에 대해 썼다면 k 가 $hash$ 에 있는지 검사합니다. 그렇게 해서
    - 있으면 $hash$ 에서 제거합니다.
    - 없으면 $continue$
- 시간 복잡도는 $hash$ 를 쓴 경우 $\mathcal{O}(10m + n)$ 입니다.

C. 가희와 은행

`data structure`, `implementation`
출제진 의도 – **Medium**

- 출제자: `chogahui05`

C. 가희와 은행

- 은행에 사람이 n 명이 온다고 해요.
- 중간에 사람이 들어오기도 한대요.
- 1초 동안 일을 보고 난 뒤에는 대기 큐의 맨 뒤로 가야 한대요.
- 상황을 정리하면 $event$ 가 특정 시간에 발생하네요?

C. 가희와 은행

- **특정 시각**에 어떤 사람이 들어왔는지 저장합시다. 그러면
  - $event[x]$를 **시간 x에 들어온 사람**으로 정의할 수 있겠네요.
  - 그러면 x초일 때 **누군가 있다면** 새로 들어왔다고 판단해고 되겠군요.
- 이렇게 유저가 새로 들어온 것을 $event$로 관리했습니다.
- 이 기법은 제가 출제한 다른 모의고사에도 많이 등장하니 익혀두고 가도 좋습니다.

C. 가희와 은행

- 이제 로직을 설계해 봅시다.
  - 먼저 매 초마다 가장 앞에 있는 고객 p를 제거해요.
  - 고객 p가 더 이상 처리할 일이 없으면 그냥 나가면 됩니다. 그렇지 않으면?
    - 해당 시간에 고객이 들어왔는지 확인해요.
    - 고객이 들어오면, 그 고객 먼저 $queue$에 넣어요.
    - 그 다음에 p를 넣으면 되겠죠?

D. 가희와 btd5

`binary search`, `math`
출제진 의도 – **Hard**

- 출제자: chogahui05

D. 가희와 btd5

- 사실 지금 **solved 난이도**를 보면 쉬워 보이지만
- 실제로 이 문제가 **대회 당시 통곡의 벽**이였음을 알 수 있습니다.
- 제가 의도하지 않았던 ccw 로 선회하신 분도 계셨는데 저도 그랬을 거 같습니다.

D. 가희와 btd5

- 아니. 코딩테스트에 왜 ccw 가 나오나요?
- ccw 를 쓴다면 어떻게 풀면 되나요? ccw 는
  - 점 3개를 **순서대로 이은 것**의 방향성을 판단하기 위해 씁니다.
  - 이 값이 0이면 **일직선 상**에 있으니 코너 처리를 하면 됩니다.
- 그런데 사실.. 이거 안 쓰고도 푸는 방법이 있습니다.

D. 가희와 btd5

- 우리에게 필요한 정보는 2가지란 말이지요.
  - 원점과 적을 잇는 직선의 기울기
  - 몇 사분면에 위치해 있는지

D. 가희와 btd5

- ✓ $y = kx$ 라는 그래프가 있다고 해 볼게요.
  - (a, b) 가 $y = kx$ 위에 있어요.
  - 그러면 t 가 0이 아니라면 $(\frac{a}{t}, \frac{b}{t})$ 도 $y = kx$ 위에 있어요. 왜?
  - $b = ka$ 를 만족하면 $\frac{b}{t} = \frac{ka}{t}$ 이기 때문이에요.

- 이 사실을 이용해 봅시다.
  - a와 b의 최대 공약수를 g라고 해 볼게요. 그러면 아래 두 사실을 만족합니다.
    - $a = ga'$
    - $b = gb'$
  - 따라서 (a, b)와 (a', b')은 $y = \frac{b}{a}x$ 위에 있어요.
  - 우리에게 필요한 정보는
    - 원점을 지나면서
    - 기울기가 얼마인 직선에 적이 있는지 입니다.

D. 가희와 btd5

- 먼저 적이 있는 위치와 원점을 지나는 직선의 기울기에 대한 정보를 **정규화** 해 봅시다.
  - 원점을 지나면서 $(\frac{a}{g}, \frac{b}{g})$ 를 지나는 직선과
  - 원점을 지나면서 (a, b) 를 지나는 직선은 같은 직선 이라고 했는데요.
- 그러면 a 와 b 를 a 와 b 의 최대공약수 g 로 나눠서 정규화 하면 어떤가요? 즉
  - 원점을 지나면서 (a, b) 를 지난다는 정보를
  - 원점을 지나면서 $(\frac{a}{g}, \frac{b}{g})$ 를 지난다는 정보로 **정규화** 하겠다는 겁니다.

D. 가희와 btd5

- ✓ 점 $(44, 55)$ 과 점 $(88, 110)$ 이 있습니다. 이 둘과 $(0, 0)$ 은 일직선상에 있을까요?
  - 44와 55의 gcd 는 11이므로, 점 $(44, 55)$ 는 $(4, 5)$ 로 정규화 됩니다.
  - 88과 110의 gcd 는 22이므로, 점 $(88, 110)$ 은 $(4, 5)$ 로 정규화 됩니다. 일직선 상에 있네요.
- ✓ 점 $(2, 3)$ 과 $(6, 12)$ 는 어떨까요?
  - 2와 3의 gcd 는 1이므로, 점 $(2, 3)$ 는 $(2, 3)$ 로 정규화 됩니다.
  - 6과 12의 gcd 는 6이므로, 점 $(6, 12)$ 은 $(1, 2)$ 로 정규화 됩니다.
  - $(2, 3)$ 과 $(1, 2)$ 는 다르네요? 따라서 일직선 상에 있지 않아요.

D. 가희와 btd5

- gcd를 이용해서 적의 위치를 **정규화** 합니다.
- x 축, y 축 위에 있는 경우는 따로 처리하면 되겠네요. 문제는 부호인데
- 적의 x 좌표, y 좌표에 절댓값을 붙인 gcd 값인 g 를 구합니다.
- 다음에 x 와 y 에 g 를 나눠 정규화 하면 됩니다.

D. 가희와 btd5

- 이제 체력이 깍이는 처리를 해 봅시다.
- 각 적마다 매번 d씩 깍는다면 최악의 경우에 $\mathcal{O}(NM)$ 만큼 수행합니다.
- **변하지 않는 것**은 총 체력이 있는데 너무 비효율적이지요.

D. 가희와 btd5

- 아래와 같은 처리를 하면 어떤가요?
  - 깍인 체력보다 총 체력인 크다면 아직 남아 있습니다.
  - 총 체력보다 깍인 체력이 더 크거나 같다면 사라집니다.
- 따라서 각 방향마다 받은 총 데미지보다 **총 체력이 더 큰 적의 개수**를 세어주면 됩니다.
- 이는 $binary\_search$ 로 할 수 있습니다.

E. 가희와 수인 분당선 1

`implementation`
출제진 의도 – **Challenging**

- 출제자: `chogahui05`

E. 가희와 수인 분당선 1

- 가희와 비행기와 동일한 난이도로 봤습니다. 그런데 1명만 풀었습니다.
- 아마 철도 용어들 때문에 그랬으리라 보입니다.
- 그 덕분에 가희와 중부내륙선을 **저평가** 하지 못했습니다. 한 가지 위안이네요.

E. 가희와 수인 분당선 1

- ✓ 열차들이 운행을 종료하거나 시작하는 역은 5역입니다.
  - 왕십리역
  - 죽전역
  - 고색역
  - 오이도역
  - 인천역

E. 가희와 수인 분당선 1

- 열차들의 출발역과 종착역을 **구간**처럼 처리할 수 있습니다.
  - 예를 들어 K210을 출발해서 K233에 도착하는 열차가 있다고 해 봅시다.
  - 그러면 이 열차는 구간 210 233을 운행한다고 할 수 있습니다.
- 이제 저는 특정 역에서 열차를 탈 겁니다. 어떻게 타야 할까요?

E. 가희와 수인 분당선 1

- ✓ 번호가 k인 역에 무조건 먼저 도착하는 열차에 탑승합니다.
- ✓ 여기서 질문. 중간에 끊기면 어떻게 할까요?
  - 그 역에서 내립니다.
  - 다음 구간으로 가는 열차로 갈아탑니다.
- ✓ 왜 이 전략이 유효할까요?
  - 앞에 오는 열차를 타면 뒤에 오는 열차를 탈 수 있지만
  - 뒤에 오는 열차를 타면 **앞에 오는 열차**를 탈 수 없기 때문입니다.

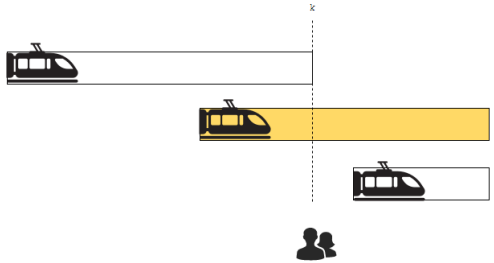
E. 가희와 수인 분당선 1

- 그런데 이대로 구현하면 구현이 다소 복잡해 집니다. 왜?
  - 쭉 타고 가는 경우와
  - 중간 종착역에서 내려야 하는 경우를 나눠야 하기 때문입니다.
- 이를 단순화 하기 위해, 중간 종착역에서는 쭉 타고 갈 수 있어도 **내렸다가 탑니다.**

E. 가희와 수인 분당선 1

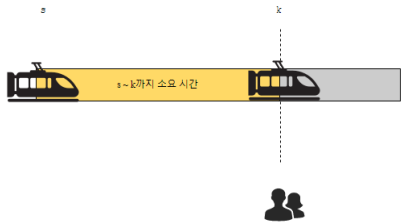
- 이제 남은 것은 한 가지입니다. 열차를 탈 수 있는가? 이를 판단하기 위해서
  - 열차가 역 k을 운행하는가?
  - 역 k에 열차가 언제 도착하는가?
- 첫 번째 질문부터 해결해 봅시다.

E. 가희와 수인 분당선 1



- 열차가 $s1$ 역에서 출발하여 $e1$ 역에 종착한다고 해 보겠습니다. k 에서 탑승하려고 할 때
  - 열차가 k 역, 혹은 k 역에서 종착하면 타지 못합니다.
  - 열차가 k 역 이후에 나오는 역이 출발역일 때 타지 못합니다.

E. 가희와 수인 분당선 1



- 열차가 s 역에서 출발하여 k 역에 도착했다고 해 봅시다.
  - 그러면 s 에서 k 역까지 오는 데 걸리는 시간을 $prefix\_sum$ 으로 구할 수 있습니다.
  - 출발 시각 + 소요 시간 + 20초가 도착한 시간과 같거나 앞서 있다면 타지 못하겠네요.

E. 가희와 수인 분당선 1

- ✓ 이 작업을 모란역과 종착역이 될 수 있는 역마다 하면 됩니다.
- ✓ 시간 복잡도는 $\mathcal{O}(4N)$ 이 됩니다.

F. 가희와 비행기

dynamic programming, combination
출제진 의도 – **Hard**

- 출제자: chogahui05

F. 가희와 비행기

- 먼저 아래의 행동은 고정입니다.
  - 맨 처음에 고도 1이 높아진다.
  - 착륙 직전에 고도 1이 낮아진다.
- 이 외에는 **고도 1 이상으로 유지**를 해야 합니다.

- ✓ 그러면 길이 $d-2$에서 올바른 괄호쌍을 구하라는 문제와 같아집니다. 왜?
  - 고도 1 밑으로 내려가면 괄호쌍이 깨지고
  - 거리가 1일 때와 $d-1$일 때의 고도가 같아야 하기 때문입니다.
- ✓ 고로, **카탈란 수**를 적용할 수 있습니다.

F. 가희와 비행기

- 혹은 $dp[x][h]$ 를 수평 좌표가 x 이고 고도가 h 인 좌표로 가는 방법으로 정의할 수 있습니다.
  - $dp[x][h] = dp[x-1][h-1] + dp[x-1][h+1]$
  - $dp[0][0] = 1$
- h 가 절대로 될 수 없는 코너 케이스를 잘 걸러주면 됩니다.

G. 가희와 거북이 인형

`bfs`, `physics`, `math`
출제진 의도 – **Challenging**

- 출제자: `chogahui05`

G. 가희와 거북이 인형

- 모든 운영진이 플레로 평가한 **유일한 문제**입니다.
- bfs 문제임은 쉽게 간파할 수 있습니다. 거북이의 크기가 매우 큽니다.
- 그런데 집, 장애물의 개수는 **매우 적어요.**

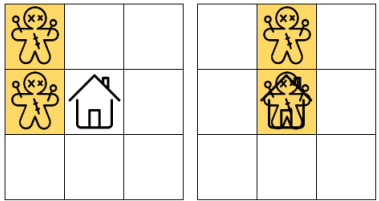
G. 가희와 거북이 인형

- 그러면 거북이가 이동할 때
  - 거북이를 이동시키기 보다는
  - 거북이를 그대로 두고 집, 장애물을 이동시키는 게 낫겠네요.
- 그러면 거북이 입장에서 **집, 장애물**이 이동하면 됩니다.
- 즉 거북이가 본 집, 장애물의 **상대 속도**를 구하는 게 1차 목표입니다.

G. 가희와 거북이 인형

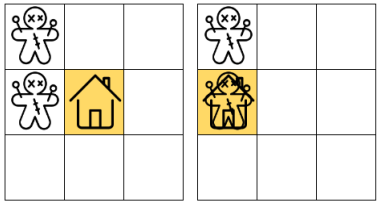
- ✓ 그런데 집, 장애물은 고정되어 있네요?
- ✓ 따라서 거북이가 본 집, 장애물의 상대 속도는 거북이가 움직인 방향의 반대 방향이 되겠네요.
- ✓ 즉, 아래 두 상황이 같은 상황입니다.
  - 거북이가 U 방향으로 움직였다.
  - 거북이가 그대로 있고 집, 장애물이 D 방향으로 움직였다.

G. 가희와 거북이 인형



- 인형이 R 방향으로 움직였습니다.
- 그러면 인형의 오른쪽 아래에 집이 있습니다.

G. 가희와 거북이 인형

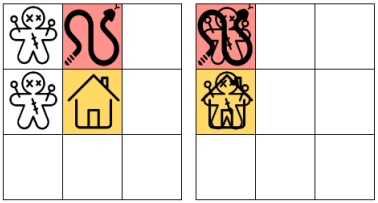


- 그런데 집 하나가 L 방향으로 움직여도 결과는 같습니다.
- 인형의 오른쪽 아래에 집이 있는 상황이 같기 때문입니다.

G. 가희와 거북이 인형

- 문제는 장애물하고 집이 같이 있다는 것입니다. 이 경우
  - 장애물이 있는 위치는 **절대로** 가지 못합니다.
  - 따라서 어떤 방향으로 이동했을 때, 장애물과 부딪치면 못 갑니다.
- 예를 들어봅시다.

G. 가희와 거북이 인형

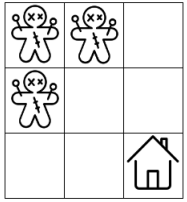


- 이 경우 R 방향으로 인형이 이동할 수 없습니다. 왜?
  - R 방향으로 이동시키는 건 장애물과 집이 L 방향으로 이동하는 것과 같은데
  - 인형들의 위치와 장애물이 충돌하기 때문입니다.

G. 가희와 거북이 인형

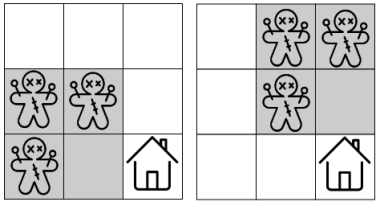
- ✓ 여기까지 정리해 봅시다.
  - 집을 이동시켜 봐서 인형이 이동할 수 있는 위치를 찾는다.
  - 장애물을 이동시켜 봐서 인형이 이동할 수 없는 위치를 찾는다.
- ✓ 이러면 끝일 거 같은데, 사실 아닙니다. 왜?

G. 가희와 거북이 인형



- 이런 맵이 있다고 생각해 봅시다.
- 인형은 어디까지 갈 수 있을까요?

G. 가희와 거북이 인형



- ✓ $bounding\_box$ 를 그려보면 매우 쉽게 알 수 있습니다.
  - 즉, 원 위치로부터 아래로 2칸 이동하거나
  - 원 위치로부터 오른쪽으로 2칸 이동하지 못합니다.

G. 가희와 거북이 인형

- 따라서 알고리즘은 아래와 같이 됩니다.
  - 집을 이동시켜 봐서 인형이 이동할 수 있는 위치를 찾는다.
  - 장애물을 이동시켜 봐서 인형이 이동할 수 없는 위치를 찾는다.
  - **맵 바깥**으로 가는 경우를 찾아서 그 경우를 모두 제외시킨다.
- 맵 구축이 다 되었다면, 남은 것은 이제 역추적은 어떻게 하냐이군요.

G. 가희와 거북이 인형

- $wif[x][y]$ 에 어느 좌표로부터 왔는지를 저장합시다.
- 우리는 bfs 를 돌리는 과정에서, 어느 좌표로부터 어느 좌표로 이동하는지 알 수 있습니다.
  - (cx, cy) 를 방문한 다음에 (nx, ny) 를 방문했다면
  - $wif[nx][ny] = (cx, cy)$ 가 됩니다.
- 이 정보들을 모두 저장한 뒤에 **역추적** 하면 됩니다.

G. 가희와 거북이 인형

- ✓ 혹은 $dist[x][y]$에 탐색 시작한 지점으로부터 얼만큼 떨어져 있는지를 저장합시다.
- ✓ 이제 도착점으로부터 $dist$ 정보를 가지고 역추적 하면 됩니다. 어떻게?
  - 한 번 탐색할 때 마다 거리가 1만큼 줄어들면 됩니다.
  - 4방향을 모두 탐색하는데, 거리가 1 줄어드는 아무 곳이나 가면 됩니다.
  - 시간 복잡도는 $\mathcal{O}(21RC)$가 됩니다.

H. 가희와 읽기 쓰기 놀이 2

`parametric_search`
출제진 의도 – **Challenging**

- 출제자: `chogahui05`

H. 가희와 읽기 쓰기 놀이 2

- N 이 큽니다.
- 그런데 x 를 추가하라는 카드가 2개 이상 나온 경우가 많지 않습니다.
- 고로 $\mathcal{O}(6!C)$ 정도에 해결을 해도 된다는 의미가 됩니다.
- 1회 가희와 읽기 쓰기 놀이로 돌아가 봅시다.

H. 가희와 읽기 쓰기 놀이 2

- ✓ $brute force$ 임은 쉽게 간파할 수 있습니다.
- ✓ 중요한 것은 해당 순열이 $valid$ 한 지 판단하는 것이였습니다. 이것이 상당히 어려운데..
- ✓ 플레이어가 **사용하지 않은 카드**는 없다고 했잖아요? 이것이 결정적인 힌트에요. 그리고 아래 두 정보를 저장해 두면 생각보다 쉽게 풀 수 있어요.
  - 각 카드에 대해 어떤 사람이 내는지
  - 내야 하는 순서

H. 가희와 읽기 쓰기 놀이 2

- ✓ 여기서는 x를 추가한다만 있고 결과값에 추가된 결과만 주어졌어요.
- ✓ x라는 것을 추가하라는 카드를 누가 들고있는지를 저장하면 됨을 알 수 있어요.
- ✓ x가 나왔을 때 현재 내야 할 것이 x를 추가하라는 카드인 유저를 빠르게 찾아야 겠네요?
- ✓ 따라서 아래와 같은 자료구조들을 생각할 수 있어요.
  - 각 카드에 대해 어떤 사람이 내는지
  - 내야 하는 순서
  - x**를 추가하는 카드**를 내는 사람들은 누구인가?

H. 가희와 읽기 쓰기 놀이 2

- ✓ 이제 x**를 추가하는 카드**를 내는 사람들은 누구인가?를 가지고 dfs를 돌려요.
- ✓ 이게 무슨소리인가요? 조합을 구한다고 생각해 봅시다.
  - 각 자리에 들어갈 수 있는 수들을 가지고 dfs를 돌렸잖아요?
  - 한 $depth$ 들어가면 다음 자리에 들어갈 수 있는 수들을 가지고 dfs를 돌리고

H. 가희와 읽기 쓰기 놀이 2

- ✓ dfs를 수행할 때, x를 추가할 수 있는 사람이 k명 있을 거란 말이지요.
  - 그러면 그 k명 중 한 명이 카드를 내면 x가 추가될 겁니다.
  - 그것이 **상태를 만족시킬 수 있는 후보 해**인 셈입니다.
  - 그러면 각 dfs의 뎁스마다 x를 추가할 수 있는 사람을 찾아서 모두 돌려주면 되겠죠?
- ✓ 상태에 들어가고 빠져나올 때가 문제입니다.

H. 가희와 읽기 쓰기 놀이 2

- t 번 사람이 x 를 추가하는 카드를 낸 다음에, x' 을 추가하는 카드를 낼 수 있다 해 봅시다.
- 덱스가 하나 증가할 때
  - $able[x]$ 에 t 를 제거하면 됩니다.
  - 다음에 $able[x']$ 에 t 를 추가하면 됩니다.
- 상태를 원복시킬 때는 어떻게 하면 될까요?
  - $able[x']$ 에 t 를 제거하면 됩니다.
  - 다음에 $able[x]$ 에 t 를 추가하면 됩니다.

H. 가희와 읽기 쓰기 놀이 2

- 이 과정은 $hash$로 처리할 수 있습니다만
- 미리 결과값에 나오는 수들을 좌표압축한 뒤에 다시 번호를 붙여도 됩니다.
- 이 경우 시간 복잡도는 $\mathcal{O}\left(6!C + ClogC\right)$가 됩니다.