단계별 모의고사 4회차 해설

문제		의도한 난이도	출제자
A	가희야 거기서 자는거 아니야	**Easy**	chogahui05
B	가희의 고구마 먹방	**Medium**	chogahui05
C	가희와 프로세스 1	**Medium**	chogahui05
D	가희와 로그 파일	**Hard**	chogahui05
E	가희와 자원 놀이	**Hard**	chogahui05
F	가희와 읽기 쓰기 놀이	**Hard**	chogahui05
G	리버스 가희와 프로세스 1	**Challenging**	chogahui05
H	가희와 프로세스 2	**Challenging**	chogahui05

A. 가희야 거기서 자는거 아니야

`implementation`, `math`
출제진 의도 – **Easy**

- 출제자: chogahui05

A. 가희야 거기서 자는거 아니야

- 가희의 가로, 세로 크가와 베개의 가로, 세로 크기가 주어졌습니다.
- 사각형의 넓이는 **가로**와 **세로**의 곱입니다.
- 결국 **베개의 가로와 세로 크기의 곱**과 **P의 개수**가 다르면 가희는 베개 위에서 자고 있는 것이 됩니다.

B. 가희의 고구마 먹방

`implementation`, `brute force`
출제진 의도 – **Medium**

- 출제자: `chogahui05`

B. 가희의 고구마 먹방

- 가희는 1초마다 상하좌우로 이동하거나 머무를 수 있습니다.
- 그런데 사실 가희는 움직이는 것이 이득입니다. 왜?
- 움직이지 않은 횟수를 c라고 하면, 갈 수 있는 기회를 c번이나 놓친 것이기 때문입니다.
- 즉, 움직이지 않는다는 선택지는 함정입니다.

B. 가희의 고구마 먹방

- ✓ 각 시간들에 대해 상하좌우로 움직일 수 있는 경우를 시뮬레이션 해 봅시다.
- ✓ 그러면 총 가짓수는 $4^{10} = 2^{20}$ 이 됩니다. 돌릴 만 합니다.
- ✓ 이제 남은 것은 고구마가 있는 장소에 다시 방문했을 때 count 처리를 어떻게 할 것이냐입니다.
  - bfs에서 방문한 지점을 어떻게 방문하지 않게 했나요? `visit` 배열을 두었습니다.
  - 혹은 hash 계열의 구조로 key가 있는지 없는지 빠르게 파악할 수도 있습니다.

C. 가희와 프로세스 1

data structure, physics
출제진 의도 – **Medium**

- 출제자: chogahui05

C. 가희와 프로세스 1

- 프로세스가 n 개 있습니다.
- 매번 프로세스는 우선순위가 가장 높은 것을 선택합니다.
- 그리고 1초가 지나면 다른 프로세스들의 우선순위가 1 높아집니다.
- **가장 한**과 우선순위 큐는 항상 짝꿍입니다. 그런데 하라는 대로 하면 시간 복잡도는 매 초마다 $\mathcal{O}\left(nlogn\right)$에 돌 겁니다. 더 좋은 방법이 없을까요?

C. 가희와 프로세스 1

- **상대적**인 관점에서 접근해 봅시다.
  - 철수, 영희, 바둑이의 점수가 10, 8, 8점이였다고 해 봅시다.
  - 다음 시험에서 철수, 영희, 바둑이의 점수가 10, 9, 9점이 되었습니다.
  - 그러면 영희와 바둑이의 입장에서는 철수가 자기 점수보다 2 높았던 게 1 높은 걸로 변경됩니다.
  - 이는 철수가 아닌 **다른 사람들** 관점에서 철수의 점수를 관찰한 겁니다.

- 이 문제에 적용하면, 아래와 같이 생각할 수 있습니다.
  - 철수는 선택된 프로세스
  - 영희와 바둑이는 **선택받지 못한 프로세스들**
- 그러면 각 시간마다 선택받은 프로세스는 우선순위가 하나 내려가는 것과 같습니다. 따라서

C. 가희와 프로세스 1

- 우선순위 큐에서 우선순위, id 순으로 priority를 설정합니다.
- 선택된 프로세스는 우선순위를 하나 낮춥니다.
- 실행 시간이 남았다면 다시 priority queue에 넣습니다.
- 최종 복잡도는 매 시간마다 $\mathcal{O}\left(logn\right)$ 이 됩니다. 전체 복잡도는 $\mathcal{O}\left(Tlogn\right)$ 이 됩니다.

D. 가희와 로그 파일

`binary search`, `prefix sum`, `string`, `parsing`
출제진 의도 – **Hard**

- 출제자: chogahui05

D. 가희와 로그 파일

- ✓ 카카오 기출의 **하위 호환**문제로 출제하였습니다.
- ✓ 문제에 주어진 조건을 보겠습니다.
  - 로그 레벨이 많아봐야 6개입니다.
  - 시각은 60만개 등장합니다.
- ✓ 고로, 로그 레벨마다 순회해도 된다는 것을 알 수 있습니다.

D. 가희와 로그 파일

- 시각은 일정한 형식으로 주어집니다.
- 사전순으로 앞선 것이 더 빠른 날짜임을 알 수 있습니다. 그러니 $sorting$ 을 하면 됩니다.
- 이제 정렬된 배열에서 s 이상 e 이하인 것의 개수를 구해야 합니다. 이는
  - s 보다 크거나 같은 원소의 개수에서
  - e 보다 큰 원소의 개수
- 를 빼면 됩니다. 각각 $lower\_bound(s)$ 와 $upper\_bound(e)$ 의 위치를 찾으면 되겠네요.

D. 가희와 로그 파일

- 조건을 만족하는 레벨마다 반복하면 됩니다.
- 매 Q 마다 $\mathcal{O}\left(6logn\right)$ 이 됩니다.

E. 가희와 자원 놀이

`data structure`, `implementation`, `math`
출제진 의도 – **Hard**

- 출제자: `chogahui05`

E. 가희와 자원 놀이

- 복잡해 보입니다.
- 그런데 하라는 대로만 하면 됩니다.
- $next$ 와 $acquire$ 하고 $release$ 가 있습니다.

E. 가희와 자원 놀이

- 먼저 $next$
- 가져왔네요? 그냥 버리면 됩니다.

E. 가희와 자원 놀이

- 다음 $release$
- 공용공간에 자원 n을 반납합니다.
- 다음에 카드를 버립니다.

E. 가희와 자원 놀이

- ✓ 어려운 건 $acquire$ 입니다. 로직을 하나씩 파 보죠.
  - 공용공간에 자원 n 이 있는지 확인합니다.
    - ▶ 있으면 가져온 다음에 카드를 버립니다. 그리고 자기 공간에 자원을 넣습니다.
    - ▶ 없으면 카드 들고 있습니다.
- ✓ 그러면 **공용 공간**에 자원이 있는지 보면 되는데, 이 공간에는 20억개의 자원이 있습니다.
- ✓ 20억개의 자원이 있는지 없는지 메모리에 저장하면 **메모리 초과** 맛을 볼 수 있겠죠?

E. 가희와 자원 놀이

- 자원은 공용 공간에 있거나 유저 공간에 있습니다.
  - 공용 공간에 없으면 유저 공간에 있고
  - 유저 공간에 없으면 공용 공간에 있습니다.
- 그런데, T 턴이 지나면, 최대 T 개의 자원이 유저 공간에 들어가겠죠?
- 즉, 유저 공간에 있는 자원을 기준으로 탐색하면 됩니다.

E. 가희와 자원 놀이

- 이제 이런 구조를 생각해 봅시다.
  - 유저가 자원 r 을 선택하면 유저 공간을 관리하는 자료구조에 r 을 넣습니다.
  - 유저가 자원 r 을 반납하면 유저 공간을 관리하는 자료구조에 r 을 제거합니다.
  - 자원 r 이 공용 공간에 있다면 반납하면 유저 공간에 r 이 없는 것입니다.
- 즉, 어떠한 키를 빠르게 찾고, 넣고, 제거하는 구조가 필요합니다.
- $hash$ 나 $tree$ 계열이 있는데, 키가 $integer$ 계열이기 때문에 $tree$ 계열이 안전합니다.

E. 가희와 자원 놀이

- ✓ 유저는 카드를 **최대 1개** 가질 수 있습니다. 왜?
- ✓ 카드를 가지고 있으면 다른 카드를 가지고 올 수 없기 때문입니다.
- ✓ 따라서
  - 카드를 들고 있으면 카드 번호를
  - 그렇지 않으면 -1을 들고 있다는 식으로 관리하면 됩니다.

F. 가희와 읽기 쓰기 놀이

`brute force`, `implementation`
출제진 의도 – **Hard**

- 출제자: chogahui05

F. 가희와 읽기 쓰기 놀이

- 9!은 40320이니, $brute force$ 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

F. 가희와 읽기 쓰기 놀이

- 예제 2를 보면
  - 1번 사람이 1, 2 순서로 내고
  - 2번 사람이 3을 낸단 말이죠.
- 카드 1, 2, 3은 사람 1, 1, 2번이 내야 하는 것입니다.
- 그런데 1번은 1번 다음에 2번을 내야 하는 **순서**가 잡혀 있어요.
- 아직 모든 카드를 내지 않은 상태라면 1번은 1, 2번은 3을 내야 해요.

F. 가희와 읽기 쓰기 놀이

- 321 순서대로 카드가 내졌다고 해 볼게요.
  - 2번 사람이 3을 낸 거니 $valid$ 하죠? 여기까지는 $no\_problem$
  - 다음에 2가 나왔는데요.
    - 2번 카드는 1번 사람이 내야 하네요.
    - 그런데 1번 사람이 내야 할 카드는 2가 아니라 1이네요?
  - 따라서 $valid$ **하지 않아**요.

F. 가희와 읽기 쓰기 놀이

- 이제 좀 더 복잡한 예제를 볼게요.
  - 1번 사람은 3, 1 순서로 내야 하고
  - 2번 사람은 2, 4 순서로 내야 합니다.
- 3241은 $valid$ 하게 낸 것일까요?

F. 가희와 읽기 쓰기 놀이

- 카드 1, 2, 3, 4는 1, 2, 1, 2번 사람이 내야 합니다.
  - 먼저 3을 냈는데요.
    - 3번 카드는 1번 사람이 내야 하네요.
    - 1번 사람이 먼저 내야 할 카드는 3이니까 맞네요? 따라서 ok
    - 그 다음에 **1번이 낼 카드**를 가져와야 겠죠? 이미 낸 3번 카드는 제거합니다.
  - 다음에 2가 나왔네요.
    - 2번 카드는 2번 사람이 내야 하네요.
    - 2번 사람이 먼저 내야 할 카드는 2이니까 맞네요? 따라서 ok
    - 그 다음에 **2번이 낼 카드**를 가져와야 겠죠? 이미 낸 2번 카드는 제거합니다.

F. 가희와 읽기 쓰기 놀이

- ✓ 카드 1, 2, 3, 4는 1, 2, 1, 2번 사람이 내야 합니다.
  - 그 다음에 4을 냈는데요.
    - 4번 카드는 2번 사람이 내야 하네요.
    - 4번 사람이 이 시점에 내야 할 카드는 4니까 맞네요? 따라서 ok
    - 그 다음에 **4번이 낼 카드**를 가져와야 겠죠? 이미 낸 4번 카드는 제거합니다.
  - 다음에 1가 나왔네요.
    - 1번 카드는 1번 사람이 내야 하네요.
    - 1번 사람이 이 시점에 내야 할 카드는 1이니까 맞네요? 따라서 ok
    - 그 다음에 **1번이 낼 카드**를 가져와야 겠죠? 이미 낸 1번 카드는 제거합니다.

- ✓ 따라서 3241은 $valid$ 하다는 것을 알 수 있어요. 이걸 어떻게 검사했는가?
  - 각 카드를 누가 내야하는지를 저장해요.
  - 각 사람마다 내야 하는 카드의 순서도 저장합니다.
  - 순열에서 i 번째로 뽑은 카드가 c 라고 할게요.
    - c 를 누가 뽑아야 하는지 봐요. p 번 사람이 뽑아야 한다고 할게요.
    - p 가 현재 뽑아야 하는 것이 c 인지를 봐서 아니라면 not_ok
    - 맞다면 p 가 뽑아야 할 카드에서 c 를 제거하면 되겠네요.

F. 가희와 읽기 쓰기 놀이

- 유저가 내야 하는 카드의 순서는 $deque$ 나 $queue$ 등으로 관리하면 쉽습니다. 왜?
- 먼저 내야 하는 카드가 앞에 있고, 직관적으로 이해하긴 편하기 때문입니다.
- 이제 시키는 대로 하면 됩니다.

G. 리버스 가희와 프로세스

`ad_hoc`, `greedy`, `math`
출제진 의도 – **Challenging**

- 출제자: chogahui05

G. 리버스 가희와 프로세스

- 실행 기준부터 봅시다.
  - 우선 순위가 높은 것이 먼저
  - 우선 순위가 같다면 id가 낮은 순으로
- 가희와 프로세스 1에서 프로세스는 실행될 때 마다 우선순위가 1씩 낮아진다고 했죠.

G. 리버스 가희와 프로세스

- ✓ $a\,b$ 순으로 등장했는데 $a > b$였다고 해 봅시다.
  - a의 우선순위가 b보다 같거나 더 낮다면 b가 a보다 먼저 등장했을 겁니다.
  - 따라서 이 경우 a가 b보다 우선순위가 높습니다.
- ✓ 따라서 이전 id가 현재 id보다 크다면 **반드시 다른 우선순위**로 묶입니다.
- ✓ 같을 때도 마찬가지로, **반드시 다른 우선순위**로 묶입니다. 왜?

- ✓ $a\,b$ 순으로 등장했는데 $a < b$였다고 해 봅시다.
  - 이 경우, a가 b보다 우선순위가 더 높습니다.
  - 혹은 같을 수도 있습니다. 왜냐하면, id는 a가 더 작기 때문입니다.
- ✓ 따라서 이전 id가 현재 id보다 작다면 **같은 우선순위**로 묶일 수 있습니다.

G. 리버스 가희와 프로세스

- ✓ 순증가 수열별로 묶습니다. 묶어진 덩어리들을 앞에서부터 1, 2, ... 로 붙여봅시다.
- ✓ 어떤 id가 덩어리 s에서 먼저 나오고 덩어리 e에서 마지막으로 나왔다면
  - 실행 시간은 $e - s + 1$ 입니다.
  - 우선순위는 $10^6 - s$로 설정하면 됩니다.

H. 가희와 프로세스 2

`parametric_search`
출제진 의도 – **Challenging**

- 출제자: chogahui05

H. 가희와 프로세스 2

- ✓ T 가 매우 크기 때문에, $parametric\_search$ 인 것은 쉽게 간파할 수 있습니다.
- ✓ 그런데 어떻게 잡아야 할까요? 가희와 프로세스 1로 다시 돌아가 봅시다.

H. 가희와 프로세스 2

- ✓ 가희와 프로세스 1에서 프로세스는 실행될 때 마다 우선순위가 1씩 낮아진다고 했죠.
- ✓ 그러니 우선순위가 p이고 실행 시간이 k인 id가 i인 프로세스를 아래 정보로 바꿔 봅시다.
  - $p-k+1$ 이상 p 이하 수가 보따리 i가 있다.
  - 이는 시간 k 동안 실행되면, 우선 순위가 $p-k+1$까지 낮아지기 때문입니다.
- ✓ 이제 $f(x)$를 x 이상인 수가 몇 개 있는지로 정의하면 되겠네요.

H. 가희와 프로세스 2

- ✓ $f(x) < T$인 제일 작은 x는 이진 탐색으로 구할 수 있어요.
- ✓ 이제 각 보따리별로 어느 범위 내에 있는 수들을 가지고 있는지 탐색해 주면 되어요.
- ✓ 시간 복잡도는 $\mathcal{O}\left(Q(n + nlog(10^{12}))\right)$가 됩니다.