가희야 거기서 자는 거 아니야

출제 의도
간단한 for loop를 활용할 수 있는지
간단한 문자열 처리를 할 수 있는지

의도한 난이도
브론즈 1

풀이
가희가 베게 위에서 자고 있다면 'P'가 있어야 할 자리에 'G'가 있을 겁니다. 베개의 넓이를 S라고 했을 때, 'P'의 개수가 S보다 작다면 가희가 베게 위에서 자고 있는 것입니다.

필요한 스킬
문자열 처리, 탐색, 반복문

가희의 고구마 먹방

출제 의도
백 트래킹을 사용할 수 있는지
상황에 맞는 적절한 자료구조를 생각할 수 있는지

의도한 난이도
실버 1 ~ 골드 5

풀이
가희의 시작 위치를 찾습니다. 시작 위치로부터 백 트래킹을 하면서 고구마를 몇 개 먹었는지에 대해서 체크해 주면 됩니다. 주의해야 할 사항이 있는데요.

고구마가 있는 자리를 중복 방문했을 때, 두 번 이상 세지 않는다.

이 부분을 어떻게 처리하면 좋을까요? 답은 중복을 제거하는 자료구조를 이용하는 것입니다. 키, 값 형식으로 저장하는 자료구조를 좋습니다. 그러면, 중복 방문한 것은 키에 1번 들어가게 될 겁니다.

재귀적으로 풀이하신다면 어떻게 하면 될까요? 키, 값 형식으로 저장하는 자료 구조가 있다고 생각해 봅시다. 방문 횟수를 추가하는 함수와, 제거하는 함수를 만들면 됩니다. 어떻게 하면 될까요? 만약에 이미 방문했는데, 또 방문한다면, value만 증가시키면 됩니다. 새로 방문하는 거라면, key를 추가 한 다음에, value 값을 1로 셋팅하면 됩니다. 방문 횟수를 하나 감소하는 것은 역으로 생각하시면 됩니다. 어떤 위치의 방문 횟수가 1이였던 경우, 방문 횟수를 1 감소 시키면 0이 됩니다. 그러면 방문 안 한 것과 같으므로, 어떤 위치를 visit에서 제거하면 됩니다. 그렇지 않으면 방문 횟수만 1 감소시켜서, map에 다시 넣으시면 됩니다.

필요한 스킬
백 트래킹, 브루트 포스, 수학

가희와 프로세스 **1**

출제 의도
상황에 맞는 적절한 자료구조를 생각할 수 있는지
상대 속도의 개념을 아는가

의도한 난이도
골드 5 ~ 골드 4

풀이
실행된 프로세스를 제외하고 우선 순위가 하나씩 증가했다는 말은 무엇을 의미할까요?
상대적 관점에서 보면, 실행된 프로세스는 우선 순위가 하나 감소했다는 말과 동치입니다.
우리는, 매 시간마다 우선 순위가 제일 높은 프로세스를 꺼내 써야 합니다.

이제, 우리는 스케쥴러에 프로세스가 들어가면, 스케줄러가 실행한 프로세스의 우선 순위와 실행이 완료되는 데 필요한 시간을 하나씩 감소시키면 됩니다. 우선 순위가 제일 높은 것을 구하고, 꺼내고, 추가하는 것을 빠르게 할 수 있는 자료구조는 pq 입니다.

필요한 스킬
물리, 우선순위 큐

가희와 로그 파일

출제 의도
시각을 파싱할 수 있는가
좌표 압축의 개념을 알고 있는가

의도한 난이도
골드 3

풀이
시각을 unix timestamp로 바꾸는 방법도 고려해 볼 만 합니다만, 조금 더 쉬운 방법을 고민해 봅시다. 이 문제에서 중요한 것은 시각 데이터의 선후 관계임을 알 수 있습니다. 따라서, 시각 자체를 문자열로 받는 방법을 고려해 볼 만 합니다. 예를 들어서 2020-02-03 11:22:33은 2020-02-03 12:22:34보다 앞에 있습니다. 이를 어떻게 알 수 있을까요? 2020-02-03 11:22:33이 2020-02-03 12:22:34보다 사전 순으로 앞에 있다는 것으로도 알 수 있습니다.

문자열을 정렬하고 lower_bound 등으로 처리하면 좌표 압축을 하실 수 있습니다. 이제, 로그 레벨이 lv 이상이면서 시각 s와 e 사이에 있는 로그 갯수를 구하는 것이 문제입니다. 레벨은 6보다 작으므로, 7 x 60만 1만큼의 누적 배열을 생성하고, 계산하면 됩니다. 문자열을 탐색하는 데 오래 걸린다면 YYYY-MM-DD hh:mm:ss 포맷을 -과 공백, :을 기준으로 분리한 후에 정수 처리하셔도 됩니다.

필요한 스킬
누적합, 정렬, 문자열 파싱, 시각 처리, 이분 탐색

가희와 자원 놀이

출제 의도
프로그램의 복잡한 기능을 모듈별로 쪼갤 수 있는지
oop의 개념을 응용할 수 있는지
상황에 맞는 자료구조를 선택할 수 있는지

의도한 난이도
골드 3 ~ 골드 2

풀이
문제가 상당히 복잡합니다. 그러므로, 기능을 잘 쪼개 봅시다.

- player가 손에 카드를 들고 있을 수도 있고 없을 수도 있다.
- 게임 판에 있는 것은 자원 카드, 연산 카드 덱이 있다.

기능을 잘 보면, 이 2개가 다임을 알 수 있습니다. 연산을 처리하는 것은 추후에 생각해 봅시다. 그러면, 아래와 같이 클래스를 설계할 수 있습니다.

```
class Player{
  /*
  holdCard가 NULL이라면 hold를 하고 있지 않은 거임.
 */
  Card holdCard;
}
```

먼저, Player는 Card를 hold 하고 있는지, 아닌지가 중요합니다. 카드를 소모했다면, 단지 Card 참조 변수인 holdCard를 NULL로 바꿔주면 됩니다. 그렇지 않으면, 현재 들고 있는 카드로, 연산을 수행 (Execute) 하면 됩니다.

그러면, 카드로 연산을 수행했을 때, 어떤 것의 상태가 바뀔까요? 자원, 연산 카드 더미가 바뀔 수 있습니다. p가 resource r을 가지고 갔을 때, p가 자원 r을 내가 소유했다고 noti 하는 규칙 하나를 추가합시다. 그러면, GameState는 아래와 같이 바뀔 수 있습니다.

```
class GameState{
  /*
  각각 자원과, 연산 카드 dummy를 의미함.
  */
  map <Integer, Integer> resource;
  List <Card> dummy;
}
```

resource는 map 형태로 해 보았는데요. K, V라고 하면 K는 자원의 번호, V는 자원을 갖고 있는 사람을 의미합니다. 문제에서 주어진 상황을 보면, 해당 자원을 누가 소유하고 있는지를 빠르게 판단하기 위해서 썼습니다. GameState는 누가 자원을 가지고 갔는지도 중요하고,

더미에 뭐가 남아 있는 지도 중요하기 때문에, 이 둘을 집어 넣었습니다. 이제, 카드의 명령을 실행시키는 것도 추가해 봅시다.

```
class gameExecutor{
   public void execute(Card c){
      if(c.op == 1) //next
      {
      }
      else if(c.op == 2) //acquire r
      {
      }
      else //release r
      {
      }
   }
}
```

이런 식으로 추가하면 됩니다. Player 클래스에는 holdCard와 putDownCard, getHodlingCard 이렇게 3개를 추가하면 됩니다. acquire r 명령을 수행하는 경우, 자원 r을 누가 갖고 있는지 판단하는 것을 gameState 클래스에서 얻어오면 됩니다.

클래스를 어떻게 설계해야 할 지 팁을 드렸습니다. 마음에 안 드실 수도 있고, 더 깔끔하게 하는 방법을 아시는 분들도 있을 겁니다. 제 설계를 따를 필요는 없습니다. 제가 클린 코딩이랑 객체 지향 프로그래밍을 연습하려고 낸 것이기 때문입니다. 여기서 제약 사항을 추가하거나, 규칙을 바꾸면 연습하기 좋을 겁니다.

하나 팁을 드리자면, unordered_map이나 unordered_set은 권장하지 않습니다. int를 key로 가지는 경우에 매우 쉽게 저격할 수 있기 때문입니다. 5번 폴더에 어떤 식으로 저격하였는지 코드 2개가 있으니, 보시는 것도 좋겠습니다. equal 연산을 빠르게 하기 위해 hash를 쓰기도 하지만, Tree 계열도 준수한 성능을 보이니, Tree 계열을 쓰는 것도 나쁘지 않습니다. 단, java는 hash 계열을 쓰셔도 됩니다. 충돌이 매우 많이 일어날 경우, 버킷을 리스트로 연결한 것을 트리로 연결하게끔 하기 때문입니다.

필요한 스킬
oop, 모듈을 잘게 쪼개는 능력, 자료구조

가희와 읽기 쓰기 놀이

출제 의도
조합과 순열을 응용할 수 있는지
oop를 활용할 수 있는지

의도한 난이도
골드 3

풀이
어떤 사람이 카드를 1, 3, 7 순서로 낸다고 해 봅시다. 그러면 1, 3, 7 입장에서는 1, 3, 7 순서로 내는 경우밖에 없습니다. 3, 1, 7이나, 7, 1, 3 순서는 유효하지 않습니다. 이 관찰을 끝냈다면, permutation을 쓰는 문제로 단순화를 시킬 수 있습니다.

예를 들어, 1번이 3, 2 순서로 카드를 내고 2번이 1, 4 순서로 카드를 낸다고 해 보겠습니다. 그 경우, 카드 1, 2, 3, 4는 각각 2, 1, 1, 2가 내게 될 겁니다. [2, 1, 1, 2]를 정렬하면 [1, 1, 2, 2]가 됩니다. 이를 permutation에 넣고 돌리면 됩니다. 이제 해결해야 할 질문은 1, 1, 2, 2가 있는 경우에는 어떻게 카드 순서를 채우면 되는지 입니다. 1번이 3, 2를 내고, 2번이 1, 4 순서대로 내므로, permutation에 1이 나타나면, 1번이 낼 카드를 채워 넣어주고 pos1을 증가시킵니다. 다음에, 2가 나타나면, 2번이 낼 카드를 채워주고 pos2를 증가시킵니다. 이런 식으로 낼 카드 순서를 채운 다음에, 문자열을 가지고 연산을 하시면 됩니다.

필요한 스킬
순열과 조합, 브루트 포스, 정렬

리버스 가희와 프로세스 **1**

출제 의도
상황을 도식화 시킬 수 있는지

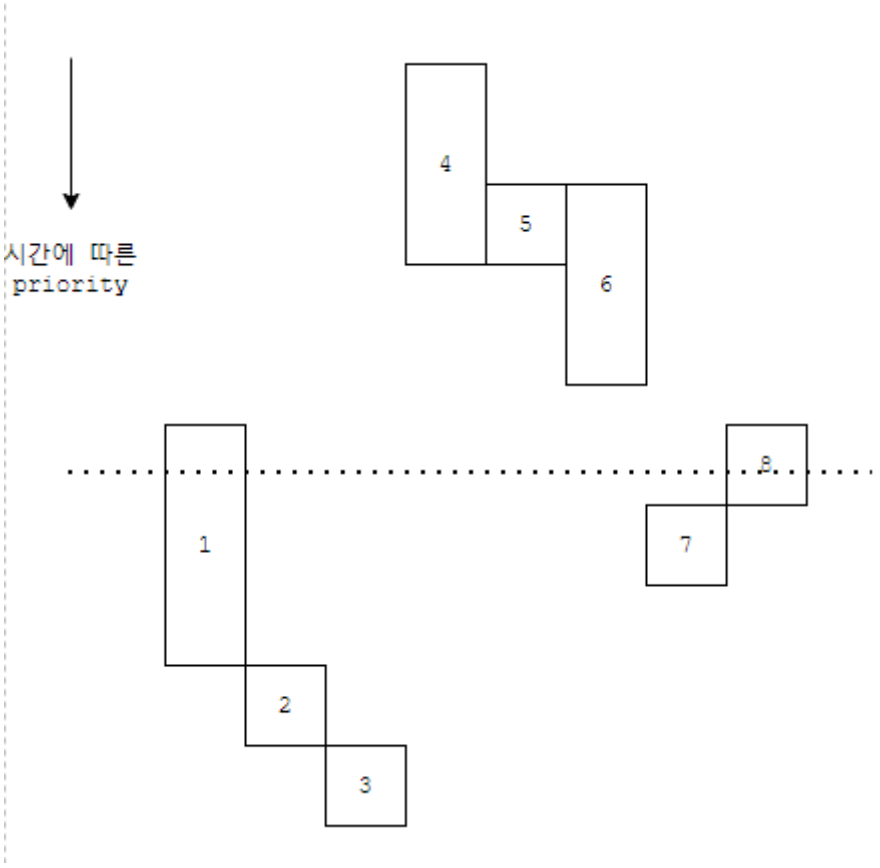
의도한 난이도
골드 3 ~ 골드 1

풀이
아래와 같은 문제로 바꿔 봅시다.

순증가/순증가/순증가/순증가/.../순증가

순증가 수열들로 배열을 끊어 봅시다. 프로세스 p가 s번째 순증가 수열에서 처음 나타났고, e번째 순증가 수열에서 마지막으로 나타났다면, p의 실행시간은 e-s+1입니다. 우선 순위는 10^6 - s로 쓸 수 있습니다.

그런데 왜 순증가 수열로 바꿀까요? 아래와 같이 상황을 도식화 시켜 봅시다.

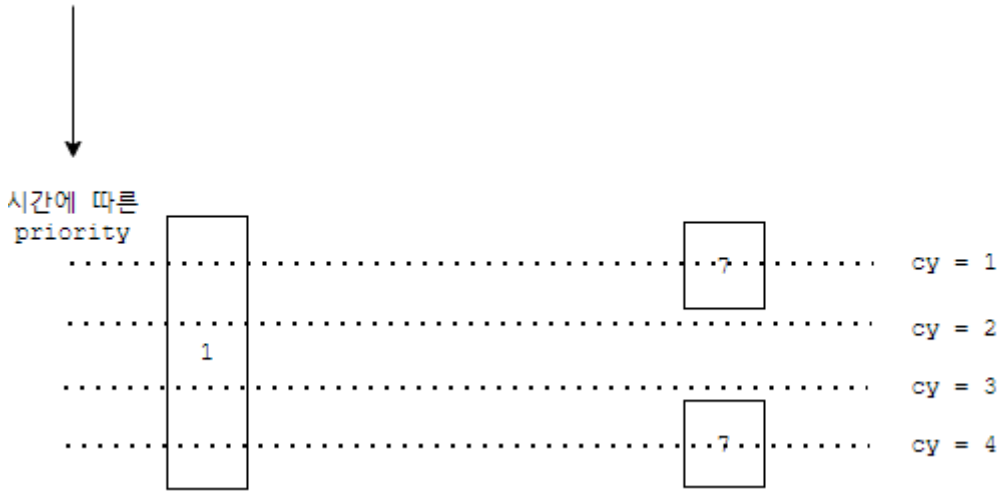


해당 시점에 우선 순위가 같다면, pid가 증가하는 순서대로 실행되기 때문입니다.



프로세스 1의 관점에서 봅시다. cy = 1, 2, 3, 4는 각각 증가 수열이 1, 2, 3, 4인 시점에 실행되고 있다는 의미입니다. cy = 1과 만나는 프로세스가 1, 3, 4, 7이라고 한다면, 프로세스는 1, 3, 4, 7 순서대로 실행될 겁니다.

이제, 7이 cy = 1에는 있지만 cy = 2, cy = 3에는 없고 cy = 4에는 있다면 어떻게 그려질까요? 아래 그림과 같이 그려질 겁니다.



불가능함을 알 수 있습니다. 원래는, 불가능한 경우까지 출력하라는 것이였습니다. 그러나, 이 경우 출제 의도에 비해 지나치게 복잡하기 때문에 문제에서는 제외하였습니다.

필요한 스킬
상황 도식화 능력, 정렬, constructive

가희와 프로세스 **2**

출제 의도
상황에 맞는 자료 구조를 이용할 수 있는가?
이분 탐색을 응용할 수 있는가?

의도한 난이도
골드 1 ~ 플레 5

풀이
이 문제는 2가지 풀이가 있습니다. 간략하게 설명해 보도록 하겠습니다. 일단 두 풀이 다 아래 사실을 이용해야 합니다.

프로세스 p의 초기 우선 순위가 prio이고, 실행 시간이 t일 때 프로세스 p는 우선 순위가 **prio - t + 1**일 때 부터 **prio**일 때 까지 실행된다**.**

(1) 이분 탐색
프로세스들의 우선 순위가 prio가 되었을 때 프로세스들의 실행 시간의 총 합을 f(prio) 라 했을 때, f(prio) >= T인 최대 prio를 찾는 방법

여기까지 간파하셨다면 백준 1561번 놀이공원 문제와 비슷하게 푸시면 됩니다.

(2) 우선 순위 큐와 좌표 압축

13334번 철로 문제를 기억하시나요? 이것과 비슷하게 접근하시면 됩니다. 사실 하나를 이용하면 프로세스는 우선 순위가 s일 때 부터 e일 때 까지 실행된다는 정보를 얻을 수 있습니다. 즉, 우선 순위가 e일 때 프로세스 하나가 실행을 시작하고, s-1일 때, 실행을 끝내므로 이벤트 배열에 아래와 같이 넣습니다.

ev[e] = ev[e] + 1, ev[s-1] = ev[s-1] + 1

당연하게도 e, s-1은 절대값이 매우 커질 수 있으므로, 좌표 압축을 해야 합니다. 여기서 ev[i]는 우선 순위가 i일 때, 구간 내에 있는 프로세스가 얼마만큼 변하는지를 나타내는 배열입니다. 이 처리까지 수행하고 나면, 다음과 같은 정보를 얻을 수 있습니다.

구간의 길이, 구간에 속한 프로세스 개수

구간의 길이와, 우선 순위 구간에 속해있는 프로세스 개수를 알고 있다면, 그 다음에는 누적 합으로 처리해 주시면 됩니다. Q가 커지면, 블록 자료구조를 이용하거나 세그먼트 트리를 이용하실 수 있습니다.

파이썬으로 푸시는 경우, (2)로 풀었을 때, 통과가 됨을 확인하였습니다. 방법 (1)로 풀면 시간 복잡도는 Qnlog(10^12), (2)로 풀면 Qnlogn에 풀 수 있으며, (2)로 푸는 경우, 자료 구조를 잘 이용하면 nlogn + Qlogn에 푸시거나 nlogn + Qsqrt(n)에 푸실 수 있습니다. 이 문제는 시간 복잡도 Qnlogn도 통과하게끔 만들었습니다만, 도전해 보고 싶은 분들은 nlogn + Qlogn에 푸시는 것도 괜찮아 보입니다. 오프라인 쿼리와 자료 구조에 대해서 깊은 이해를 하실 수 있게 될 겁니다.

필요한 스킬

이분 탐색, 자료 구조, event driven 시뮬레이션 처리