가희와 방어율 무시
출제 의도
문제를 읽고, 수식을 계산해서 조건에 맞는 처리를 할 수 있는가?

의도 난이도
브론즈 4

풀이
몬스터의 방어율이 a이고, 방어 무시율이 b%라면, 내가 느끼는 몬스터의 체감 방어율은 a(100-b)/100 으로 구할 수 있습니다. 예를 들어, a가 300이고, b가 20이라면, 내가 느끼는 체감 방어율은 300(100-20)/100 = 240이 됩니다.

이 수치가 100보다 크거나 같으면 데미지를 입힐 수 없습니다.

필요한 스킬
수학, 조건문

가희와 카오스 파풀라투스
출제 의도
시각 처리를 할 수 있는가?

의도 난이도
실버 3

풀이
^을 먹으면, 가리키는 영역을 무효화 시키면 됩니다. 10MIN, 30MIN 등의 아이템을 먹었을 때 어떻게 처리해야 하는지가 문제입니다. hh:mm 꼴로 들어오는 경우 60hh + mm, 즉 분으로 변환합니다. 그러면 10MIN, 30MIN 등이 들어왔을 때, 10분만큼, 혹은 30분 만큼 증가시킬 수 있습니다.

주의할 점은, 0시하고 12시하고 상태가 같고, 1시하고 13시하고 상태가 같다는 것입니다. 그 말인 즉슨, 720분을 주기로 돈다는 이야기입니다. 이 점만 조심하시면 무난하게 푸실 수 있습니다.

필요한 스킬
시각 처리, 조건문

가희와 파일 탐색기 2
출제 의도
상황에 맞는 적절한 자료 구조를 이용할 수 있는가?
(os) 권한에 대해서 알고 있는가?

의도 난이도
골드 4

풀이
먼저 주어지는 연산을 분석해 봅시다. 어떤 유저가 특정한 파일에 대해, R/W/X 연산을 수행할 수 있는가? 그러면, 이 특정한 파일에 대해, 어떤 유저가 유저 권한을 가지는지, 그룹 권한을 가지는지, 아니면 둘 다 아닌지를 먼저 파악해야 합니다.

유저가 유저 권한을 가지는지 파악하는 것은 쉽습니다. 유저 이름이 “y”라고 하면, 파일의 소유자가 “y”인지 체크하면 되기 때문입니다. 파일의 소유 그룹이 “g”라고 했을 때, 유저 “y”가 소유 그룹 “g”에 속하는지는 어떻게 빠르게 판단해야 할까요?

group[“g”]에 “y”가 속한다. 라는 정보를 저장하면 어떨까요? 이 경우, “g”에 속하는 유저는 여러 명이 나올 수 있습니다. “g”에 특정 유저가 속하는지 빠르게 판단하기 위해서, group[“g”] 에 저장하는 자료구조 또한 set이나 hash 계열이 되어야 합니다.

조금 더 쉬운 방법이 없을까요? 각 유저가 속하는 그룹 수는 10개 이하입니다. 따라서, “y”가 그룹 “g”에 속하는지 검사하기 위해, user “u”가 속하는 그룹들을 선형탐색 해도 됩니다. 이 때, key는 유저 이름이 되고, value는 유저가 속한 그룹들이 됩니다. 문제는, 유저가 속한 그룹의 문자열 길이가 최대 10까지 들어오므로, 아슬아슬할 수 있습니다. 이럴 때 쓸 수 있는 방법은, 문자열을 수로 좌표 압축 하는 것입니다.

이제, 유저가 어느 종류의 권한을 가지는지 파악했다면, 비트 연산으로 R, W, X 연산을 수행할 수 있는지 판단하면 됩니다. 여담으로, 문제에 나온 권한 중 755, 600, 644 등은 많이 보일 것이므로, 이 기회에 익혀두시는 것도 도움이 될 듯 합니다.

필요한 스킬
자료구조, 해시나 set에 대한 정확한 이해, 좌표 압축, 권한에 대한 기본적인 이해

가희와 중부 내륙선
출제 의도
상황을 이해하고 모델링 할 수 있는가?
적절한 자료구조를 선택할 수 있는가?
(os) lock 메커니즘을 이해하고 구현할 수 있는가?
(cs) event driven 구조 (혹은 callback 구조)를 알고 있는가?

의도 난이도
골드 1 ~ 플레 5

풀이
수인 분당선 문제에 이어서, 또 다른 철도 문제가 등장하였습니다. 뭔가 어려워 보이지만, 문제에서 요구하는 것은 하나입니다. 역과 역 사이에 구간이라는 것이 있는데, 구간에는 단 1대의 열차만 접근이 가능하다. 여기서 한 가지 파악해야 할 점은 각 구간은 'lock'을 걸어야 하는 대상이라는 것입니다.

이 문제에서는 무엇을 해야 할까요? 일단, 열차가 구간에 접근한 순간 다른 열차가 접근하지 못하게 'lock'을 걸어야 합니다. 언제 'lock'을 풀어야 할까요? 구간에 접근하고 나서 '구간의 소요시간'만큼 지난 후에. 이 때 뭘 해야 할까요? 관제에게 다른 열차가 지나가도 된다는 알림, 즉 release를 해야 합니다. 이것은 하나의 '이벤트' 로 관리할 수 있습니다. 어떻게 관리하면 될까요? 예를 들어 731 열차가 부발역을 07시 30분에 관제의 허가를 받아서 부발역을 출발한다고 해 봅시다. 그러면, 07시 30분에 부발 ~ 가남 구간을 잠그고, 07시 37분에 부발 ~ 가남 구간에 대한 잠금을 푸는 이벤트를 넣어버리면 됩니다.

그런데 이벤트는 이것만 있는 것이 아닙니다. 각 역에 대해 출발 준비를 마치는 이벤트도 있어야 합니다. 출발 준비를 마치면 어떻게 해야 할까요? 역별 대기큐에 넣어야 합니다. 대기 큐는 어떤 자료구조가 좋을까요? 먼저 온 열차가 먼저 떠나야 하므로, FIFO. 즉 큐나 deque가 적합합니다.

이벤트는 이 두 종류가 있다는 것 까지는 알았습니다. 출발 준비를 마쳐서 대기 큐에 들어가는 이벤트, 그리고 각 구간을 잠궈버리고 푸는 이벤트. 그런데, 이것을 어떻게 시뮬레이션 해야 할까요? 차근 차근 생각해 봅시다.

각 구간은 역 사이에 있습니다. 부발 ~ 가남 구간은 하행 방향의 부발, 상행 방향의 가남. 이렇게 2개가 쓸 수 있습니다. 구간 A ~ B를 운행하고 있는 열차가 있다면, 관제가 구간에 진입하는 것을 허락하지 않는다고 되어 있습니다. 이 말은 A역에서 출발한 열차가 B역에 도착하자마자 구간 A ~ B에 대한 잠금을 풀어야 한다는 의미입니다. 따라서, 각 구간에 대해서 먼저 해야 할 일은 '잠금을 푸는 이벤트' 가 있는지를 검사하는 것입니다.

다음에, 구간 A ~ B를 운행하려고 하는 열차가 있다면 관제가 어떠한 기준에 의해 판단하는 로직이 2부터 서술되어 있습니다. 열차가 출발 준비를 마치면 특정 구간을 운행하려고 하는 열차가 됩니다. 예를 들어, 하행 열차가 부발역에서 출발 준비를 마치면, 해당 열차는 부발 ~ 가남 구간을 운행하려고 하는 열차가 됩니다. 따라서, 열차가 역에 도착한 즉시, 1분 후에 해당 역에서 어느 방향으로 출발 준비를 하는 트리거를 걸어야 합니다. 당연히, 출발 준비 트리거가 해당 시간에 걸려있는지도 검사해야 합니다.

이제, 각 구간에 들어갈 열차들을 검사해서 조건에 맞는 열차를 구간에 넣으면 됩니다. 만약, 열차 a가 구간 A에 들어갔다면, 구간 A에 락을 걸어버리고, 구간 A의 소요시간 후에 구간 A의 잠금이 풀린다는 트리거를 걸면 됩니다.

이 과정을 구현하면 됩니다. 이를 쉽게 구현하는 틀을 소개하겠습니다.

> [1분마다 아래의 로직을 수행합니다.]
> 각 구간에 대해 잠금이 풀리는 이벤트가 있는지 검사하고 있으면, 잠금을 풀어버린다.
> - 이 열차들에 대해, 1분 뒤에 각 역의 ~ 방향에 출발 대기를 하는 큐에 넣는다는 트리거를 추가한다.
>
> 출발 준비를 마친 열차에 대해, 대기 큐에 넣는다.
>
> 각 구간에 들어가야 하는 열차를 뽑는다. 뽑힌 열차는 출발 대기를 하는 큐에서 빠지게 된다.

이를 잘 구현하시면 됩니다.

필요한 스킬
락 매커니즘의 이해, 이벤트 처리 (콜백 처리)에 대한 이해, 자료구조, 상황 모델링

가희와 지하철
출제 의도
최단 거리 알고리즘에 대해 정확하게 알고 있는가?
환승역의 제한이 왜 작은지 분석할 수 있는가?

의도 난이도
골드 2

풀이
역 a에서 역 b로 갈 때, 최단 거리는 두 type 중 하나가 나올 수 있습니다. 환승역 없이 가거나, 환승역을 경유해서 가는 경우. 각 환승역을 시작점으로 삼아서 bfs나 다익스트라를 돌립니다. 그러면 환승역 k를 경유해서 a에서 b로 가는 최단 경로를 구할 수 있습니다.

그런데 이 경우만 고려하면 끝일까요? 당연하게도 아닙니다. 같은 노선에 있는 경우를 생각해야 합니다. 부천 ~ 역곡을 들어봅시다. 부천역에서 역곡역까지는 단 2정거장입니다. 그런데, 부천 ~ 부평(1/인천1) ~ 부평구청(인천1/7) ~ 온수(1/7) ~ 역곡으로 갈 수도 있습니다. 이 중 어느 경로가 더 빠를까요? 당연하게도 전자입니다. 같은 노선만 타고 가는 경우가 더 빠르기도 하기 때문에, 이 경우도 같이 고려해 주어야 합니다.

이제, 각 역마다 좌표 압축을 해 준 다음에 환승역을 거쳤을 때 최단 거리의 최솟값, 환승역을 거치지 않았을 때 최단 거리의 최솟값을 구하면 됩니다.

필요한 스킬
최단 거리 알고리즘, case work

가희와 사직 구장
출제 의도
모든 경우의 수를 탐색할 수 있는가?
누적합, 혹은 그에 준하는 자료구조를 생각해 낼 수 있는가?

의도 난이도
골드 2

풀이
nC3가지 경우에 대해서, 3명을 배치한 다음에, 나머지 n-3명에 대해 매력이 큰 순으로 뽑으면 됩니다. 매력이 큰 순으로 뽑는 작업을 O(1)이나, O(log(n))으로 할 수 있는데, 이 중 O(1)로 하는 방법은 누적합을 이용하는 것입니다.

문제는 rank를 잘못 처리하기가 매우 쉽다는 것입니다. 그러한 실수 중 하나입니다.

```
[i, j, k에 대해]
ve = 0
n3_t = n-3위까지 point 총 합
i가 n-3위 안에 속하면 n3_t = n3_t - i번째 위치의 point, ve++
j가 n-3위 안에 속하면 n3_t = n3_t - j번째 위치의 point, ve++
k가 n-3위 안에 속하면 n3_t = n3_t - k번째 위치의 point, ve++

ve가 1이라면 n3_t = n3_t + v2[n-3].v
ve가 2라면 n3_t = n3_t + v2[n-3].v + v2[n-2].v
ve가 3이라면 n3_t = n3_t + v2[n-3].v + v2[n-2].v + v2[n-1].v
```

이 알고리즘에서 무엇이 잘못되었을까요? i, j, k가 n-3위 안에 속하면 ve값을 증가시키고, ve값에 따라 추가적인 작업을 수행합니다. ve가 0인 경우 아무런 문제가 되지 않습니다. 문제는, i가 n-3위, j가 n-2위, k가 n-1위일 때입니다.

이 때, ve가 1이므로, n3_t는 1위부터 n-3위까지의 합에서, n-3위까지가 빠질 겁니다. 문제는 여기에, n-3위의 데이터가 더해진다는 것입니다. 사실은 n위의 데이터가 더해져야 합니다. 이러한 문제를 어떻게 개선해야 할까요?

사실, 우리는, 1위부터 n-3위까지 데이터를 뽑기 위해서, 많아봐야 1위 ~ n위까지의 데이터만 보면 됩니다. 왜 그럴까요? 최악의 경우를 생각해 보면 매우 쉽게 알 수 있습니다.

i	j	k	[3 = 1위]	…	…	…	[n-1위 = n-3위]

즉, n-3위까지의 누적합, n-2위까지의 누적합, n-1위까지의 누적합, n위까지의 누적합을 검사해서, 실제로 몇 개의 영역이 (문제에서 설명하는 삼총사가 서 있는 지역이 아닌 곳의 개수를) 검사하면 됩니다. 여기까지만 들어보면 이게 무슨 소리인가 하실 것이니 예를 들어봅시다.

n을 7이라 합시다. 그러면 n - 3은 4가 됩니다.

	i		j	k			

i, j, k가 각각 2, 4, 5위라고 해 봅시다. n - 3의 값은 4이니, 앞에서부터 4위까지, 5위까지, 6위까지, 7위까지 삼총사가 서 있는 지역을 제외한 영역이 있는지 검사해 봅시다.

	i		j	k			

7위까지 뽑았을 때 최초로 삼총사가 선 지역 아닌 4개의 영역이 뽑히게 됩니다. 따라서, 아래 슈도코드를 수행하면 됩니다.

```
[각 경우에 대해]
3명을 뽑는다.
나머지 n-3명에 대해서 아래의 일을 수행한다.
상위 n-3명 중, 삼총사가 아닌 사람이 n-3명이 있는가? 있으면 break
상위 n-2명 중, 삼총사가 아닌 사람이 n-3명이 있는가? 있으면 break
상위 n-1명 중, 삼총사가 아닌 사람이 n-3명이 있는가? 있으면 break
상위 n명 중, 삼총사가 아닌 사람이 n-3명이 있는가? 있으면 break
```

이제, 아래의 질문이 남았습니다. 상위 u개의 영역 중 삼총사가 아닌 사람이 n-3명이 있다는 정보를 알고 있을 때, 우리는 상위 u개의 영역 score 합에 아래 3개를 빼면 됩니다.

```
i가 n-3위 안에 속한다면, 상위 u개의 영역 score 합에 영역 i의 point를 뺀다.
j가 n-3위 안에 속한다면, 상위 u개의 영역 score 합에 영역 j의 point를 뺀다.
k가 n-3위 안에 속한다면, 상위 u개의 영역 score 합에 영역 k의 point를 뺀다.
```

이를 구현하면 됩니다.

가희와 영상 추천 시스템
출제 의도
문제의 상황에 적합한 자료구조를 생각할 수 있는가?
시각 처리를 할 수 있는가?

의도 난이도
플레 4 ~ 플레 3

풀이
우리는 2가지를 구현해야 합니다.
영상을 볼 때 마다 영상의 조회수가 증가한다.
영상을 볼 때 마다 해당 영상의 카테고리 조회수가 증가한다.

이 둘을 통해 카테고리별로 priority_queue나 set, map 등을 써야 함을 알 수 있습니다. 이를 category pq (혹은 set/map)라 하겠습니다. 이제, 각 카테고리 별로 가장 연관도가 높은 영상들 중에, 연관도가 높은 친구들을 뽑아야 합니다. 이것 역시 priority_queue나 set, map 등으로 관리할 수 있습니다. 이러한 역할을 하는 것을 common pq (혹은 set/map)라고 하겠습니다.)

문제는 72시간 내의 영상을 어떻게 판단하는지 입니다. 그리고 이러한 정보를 어떻게 관리할 것인지입니다. 이벤트는 시간 순으로 들어오므로 queue나 deque 등을 이용해서 관리합니다. R 이벤트가 들어올 때 마다 72시간보다 더 오래 전에 본 영상들의 정보를 삭제하면 됩니다. 삭제가 일어나면, 해당 영상의 카테고리 조회수와, 해당 영상의 조회수가 감소하므로, 이에 맞게 common_pq와 category_pq를 업데이트 해 주어야 합니다.

72시간 내의 영상을 어떻게 판정할까요? 쉬운 방법은 YYYY-mm-dd까지만 보고 판단하는 것입니다. 이벤트가 앞선 것은 먼저 들어올 것이므로, 큐에 관리되고 있는 것 보다는 현재 검사하고 있는 이벤트가 시간순으로 뒤에 있을 겁니다. 만약, 이전 날짜를 yyyy-mm-dd라 하고 현재 검사하고 있는 날짜를 YYYY-MM-DD라고 하면, 이 차이가 4 이상인 경우 볼 필요도 없습니다. 2 이하인 경우도 볼 필요가 없습니다. 3인 경우가 문제가 됩니다.

YYYY-MM-DD와 yyyy-mm-dd의 날짜 차이가 3인 경우 어떻게 해야 할까요? YYYY-MM-DD HH:MM:SS, yyyy-mm-dd hh:mm:ss가 있을 때, HH:MM:SS가 hh:mm:ss보다 사전순으로 앞에 있거나 사전 순으로 같다면, (예를 들어 2020-03-05 01:00:00, 2020-03-02 02:00:00) 3일이 안 지난 것입니다. 그렇지 않으면 (2020-03-05 02:00:01, 2020-03-02 02:00:00) 기간이 3일이 지난 것입니다.

비슷한 문제 중에 가희의 수열놀이 시리즈와 가희와 btd5 2가 있으니, 먼저 풀어보고 오시는 것도 좋은 방법일 듯 합니다.

필요한 스킬
자료구조, 시각 처리