[22.06.24] 생존열차

TS열차

김 태 현

문 제

- 0 ~ N-1 번호를 갖는 N명의 탑승객 , M명 정원의 객차, (N/M)개의 객차 수 존재
  탑승객마다 포인트와 한 개의 직업을 보유, 0 ~ J-1 번호의 J개의 직업 존재
  초기에 맨 앞칸에 0 ~ M-1 번호의 탑승객 탑승
  둘째칸에 M ~ 2M-1 번호의 탑승객 탑승
  순서대로 마지막 칸까지 탑승

- 포인트는 개별 또는 직업별로 업데이트

- 객차별 우선순위 순으로 상위 인원은 앞 칸으로, 하위인원은 뒤 칸으로 이동
  우선순위 기준 1) 포인트 많은 순 2) ID 작은 순

* N=8, M=4 인 경우 예시

ID	0	1	2	3	4	5	6	7
Job	2	0	2	1	0	1	2	2
Point	8	19	18	9	18	1	4	1

void init(int N, int M, int J, int Point[], int JobID[])

N: 탑승객 수, M: 객차 정원, J: 직업 수, N/M: 객차 개수, N은 M의 배수
Point[] : 탑승객 초기 포인트 , JobID[] : 탑승객 직업 번호

int update(int uID, int Point)

uID 번호의 탑승객의 포인트가 Point(-1,000 ~ 1,000) 만큼 증가
탑승객 보유 포인트 반환

int updateByJob(int JobID, int Point)

JobID 직업의 모든 인원의 포인트가 Point(-1,000 ~ 1,000)만큼 증가
JobID 직업 탑승객의 보유 포인트 총합 반환

int move(int num)

객차별로 상위 num명은 앞 객차로, 하위 num(1 ~ 5)명은 뒤 객차로 이동
이동한 모든 탑승객의 보유 포인트 총합 반환

※ 제약사항

N <= 100,000
M <= 10,000
J <= 1,000
N/M <= 10
직업별 최대 인원 <= 200명
update() <= 10,000회
updateByJob() <= 300회
move() <= 1,000

Naive

- **update(uID, point)** : Point[uID] += point
  O(1) * 10,000회

- **updateByJob(jobID, point)** : 모든 jobID 탑승객에 대해 Point[uID] += point
  O(200) * 300회

- **move(num)** : 매번 모든 객차 정렬 후 이동 (sort)
  O((N/M) * M log M) * 1,000회
  =O(N * log M) * 1,000회
  =O(100,000 * 14) * 1,000회

  or 상위 num명, 하위 num명 부분 정렬 후 이동 (partial_sort)
  O((N/M) * 2 * M log num) * 1,000회

직업별 탑승객[jobID] , 2차원 배열 or 벡터 배열

JobID	uIDs
0	1, 4, 6, 10, 11, 16, 18, 20
1	3, 7, 12, 13, 14, 15, 17, 19
2	0, 2, 5, 8, 9,

객차별 탑승객[jobID] , 2차원 배열

객차	uIDs
0 (앞)	0, 1, 2, 3, 4, 5, 6
1	7, 8, 9, 10, 11, 12, 13
2 (뒤)	14, 15, 16, 17, 18, 19, 20

Point[uID, 0 ~ N-1] , 배열

uID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Point	8	19	18	9	18	19	2	1	4	1	16	18	2	7	0	6	9	2	11	18	1

Priority Queue

- **update(uID, point)** : Point[uID] += point
  상위, 하위 PQ에 업데이트된 상태 push
  O(2 * log M) * 10,000회

- **updateByJob(jobID, point)** : 모든 jobID 탑승객에 대해 Point[uID] += point,
  상위, 하위 PQ에 업데이트된 상태 push
  O(200 * 2 * log M) * 300회

- **move(num)** : 모든 객차의 상위PQ and 하위PQ에서 pop을 해나가며
  유효한 num개 기록 후, 이동시키는 객차에 push
  O((N/M) * 2 * 2 * num log M) * 1,000회
  =O(10 * 2 * 2 * 5 * 14) * 1,000회

직업별 탑승객[jobID] , 2차원 배열 or 벡터 배열

JobID	uIDs
0	1, 4, 6, 10, 11, 16, 18, 20
1	3, 7, 12, 13, 14, 15, 17, 19
2	0, 2, 5, 8, 9

객차별 상위 PQ[객차ID]

(point, uid)

1. point 높은 순
2. uid 낮은 순

객차별 하위 PQ[객차ID]

(point, uid)

1. point 낮은 순
2. uid 높은 순

탑승객 정보[uID, 0 ~ N-1] , 배열

uID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Point	8	19	18	9	18	19	2	1	4	1	16	18	2	7	0	6	9	2	11	18	1
Car	0	0	0	0	0	0	0	1	1	1	1	1	1	1	2	2	2	2	2	2	2

Priority Queue

PQ 에서 노드의 우선순위가 변경/삭제 될 때 처리 방법

1. lazy update
2. real-time update (indexed heap)

* STL PQ 활용시, 1번만 사용 가능

중요 Point1

lazy update 방식은 top 값을 구할 때 유효한 값인지 판별 필요

* 유효하지 않은 경우
  1) 최신 point와 다른 경우 ex) update(19, -100)
  2) 객차에서 이동한 경우 ex) update(19, -1), update(19, 1), move(1), move(1)
  3) 직전에 뽑힌 탑승객과 동일한 경우 ex) update(19, -1), update(19, 1), move(2)

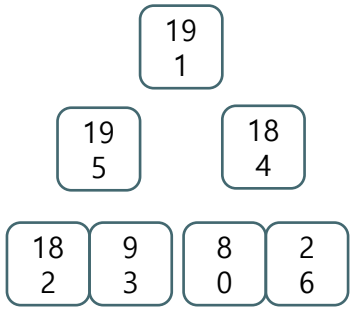
중요 Point2

부분 부분 이동시키는 경우 동일 탑승객이 여러 번 이동하는 문제 발생 ex) 뒤 객차부터 바로바로 이동

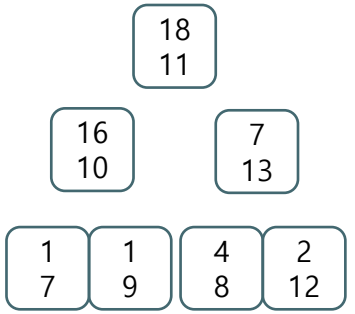
ex) 한칸씩 처리하는데 바로바로 이동하는 객차 pq에 push 하는 경우

따라서, 모든 이동객을 뽑아서 따로 기록한 뒤 한번에 push 필요

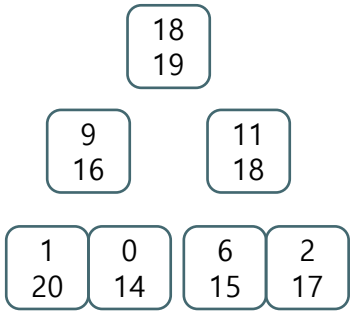
상위 PQ[0]



상위 PQ[1]



상위 PQ[2]



탑승객 정보[uID, 0 ~ N-1] , 배열

uID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Point	8	19	18	9	18	19	2	1	4	1	16	18	2	7	0	6	9	2	11	18	1
Car	0	0	0	0	0	0	0	1	1	1	1	1	1	1	2	2	2	2	2	2	2

Set

- **update(uID, point)** : Point[uID] += point
  set에서 기존 노드 삭제 후 새로운 정보 삽입
  삭제시에는 uid별로 iterator 기록하여 활용 권장
  O(2 * log M) * 10,000회

- **updateByJob(jobID, point)** : 모든 jobID 탑승객에 대해 Point[uID] += point,
  set에서 기존 노드 삭제 후 새로운 정보 삽입
  O(200 * 2 * log M) * 300회

- **move(num)** : set에서 begin()부터 상위 num명, rbegin()부터 하위 num명 선별,
  삭제 후 이동하는 객차로 삽입
  O((N/M) * 2 * 2 * num log M) * 1,000회
  =O(10 * 2 * 2 * 5 * 14) * 1,000회

직업별 탑승객[jobID] , 2차원 배열 or 벡터 배열

JobID	uIDs
0	1, 4, 6, 10, 11, 16, 18, 20
1	3, 7, 12, 13, 14, 15, 17, 19
2	0, 2, 5, 8, 9

객차별 우선순위 높은 순 Set[객차ID]

(point, uid)

1. point 높은 순
2. uid 낮은 순

begin: 상위 <-> rbegin: 하위

탑승객 정보[uID, 0 ~ N-1] , 배열

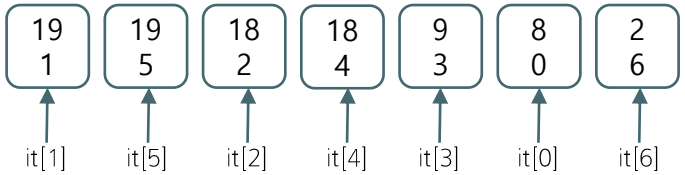
uID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Point	8	19	18	9	18	19	2	1	4	1	16	18	2	7	0	6	9	2	11	18	1
Car	0	0	0	0	0	0	0	1	1	1	1	1	1	1	2	2	2	2	2	2	2

Set example

탑승객 정보[uID, 0 ~ N-1] , 배열

uID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Point	8	19	18	9	18	19	2	1	4	1	16	18	2	7	0	6	9	2	11	18	1
Car	0	0	0	0	0	0	0	1	1	1	1	1	1	1	2	2	2	2	2	2	2

0번 객차 set



1번 객차 set

2번 객차 set

직업별 탑승객[jobID] , 2차월 배열 or 벡터 배열

JobID	uIDs
0	1, 4, 6, 10, 11, 16, 18, 20
1	3, 7, 12, 13, 14, 15, 17, 19
2	0, 2, 5, 8, 9

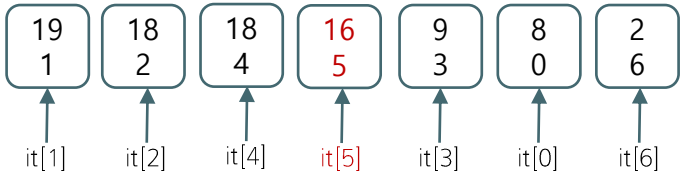
update(5, -3)

Set example

탑승객 정보[uID, 0 ~ N-1] , 배열

uID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Point	8	19	18	9	18	16	2	1	4	1	16	18	2	7	0	6	9	2	11	18	1
Car	0	0	0	0	0	0	0	1	1	1	1	1	1	1	2	2	2	2	2	2	2

0번 객차 set



1번 객차 set

2번 객차 set

직업별 탑승객[jobID] , 2차원 배열 or 벡터 배열

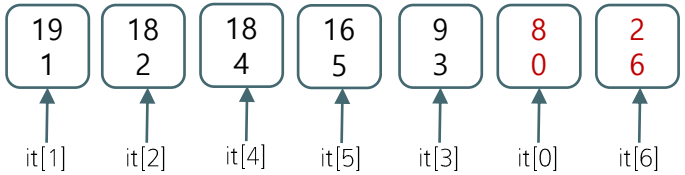
JobID	uIDs
0	1, 4, 6, 10, 11, 16, 18, 20
1	3, 7, 12, 13, 14, 15, 17, 19
2	0, 2, 5, 8, 9

Set example

탑승객 정보[uID, 0 ~ N-1] , 배열

uID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Point	8	19	18	9	18	16	2	1	4	1	16	18	2	7	0	6	9	2	11	18	1
Car	0	0	0	0	0	0	0	1	1	1	1	1	1	1	2	2	2	2	2	2	2

0번 객차 set



1번 객차 set

2번 객차 set

직업별 탑승객[jobID] , 2차원 배열 or 벡터 배열

JobID	uIDs
0	1, 4, 6, 10, 11, 16, 18, 20
1	3, 7, 12, 13, 14, 15, 17, 19
2	0, 2, 5, 8, 9

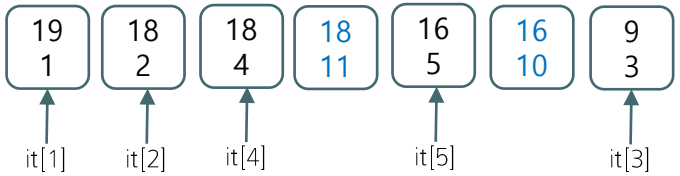
move(2)

Set example

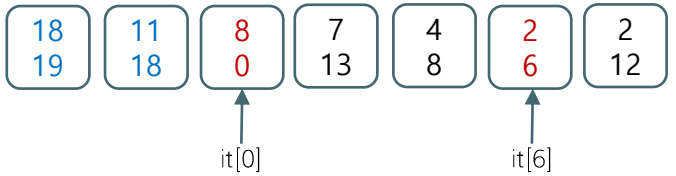
탑승객 정보[uID, 0 ~ N-1] , 배열

uID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Point	8	19	18	9	18	16	2	1	4	1	16	18	2	7	0	6	9	2	11	18	1
Car	1	0	0	0	0	0	1	2	1	2	0	0	1	1	2	2	2	2	2	2	2

0번 객차 set



1번 객차 set



2번 객차 set

직업별 탑승객[jobID] , 2차원 배열 or 벡터 배열

JobID	uIDs
0	1, 4, 6, 10, 11, 16, 18, 20
1	3, 7, 12, 13, 14, 15, 17, 19
2	0, 2, 5, 8, 9

* set 정보 삭제/삽입 시마다
  iterator 재설정 필요


감사합니다