메모리 박람회

김 태 현

210710pro_놀이공원

문제 분석

- booth 수 : 3 ~ 100
- booth id : 0 ~ 1,000,000,000
- tick : 1 ~ 100,000,000 , 증가하는 방향

- **init(N, bid[], duration[], capacity[])**
  - N개의 booth 정보

- **add(tick, bid, cnt, priority)** : 10,000회
  - tick 시간에 priority 우선순위를 갖는 cnt명이 bid에 추가
  - 대기 고객 중 최우선순위 값 반환

- **search(tick, cnt, bid[], num[])** : add()+search() 20,000회
  - tick 시간에 우선순위 높은 cnt개의 체험관 반환 (cnt<=10)
  - 우선순위 : 1)대기인원수 많은 순 2) ID 큰 순

해법 연구

고려할 사항

- id 처리 : unordered_map renumbering (c++)
- booth별 대기인원 관리
- 우선순위 높은 booth 선택
- api간 tick 사이 booth 정보 업데이트

해법 연구

booth별 대기인원 관리

booth별 필요한 정보

- 마지막 체험 종료 시점 int
- duration int
- capacity int
- 총 대기 인원 int
- 대기인원 우선순위 정보 ??
- 오리지날 id int (C++)

booth의 체험이 종료되고 대기중인 인원이 있다면
우선순위 별로 입장

우선순위 순서로 대기인원 관리
: map<priority, cnt> m[103]

m[id][priority]+=cnt; // log(N)

Python: heapq

해법 연구

우선순위 높은 부스 선택 (최대 10개)

- 우선순위 1) 대기인원수 많은 순 2) ID 큰 순

booth 수 : 100, search 수 : 20,000

C++

1. 미리 set이나 pq로 관리 : 어차피 search 호출시에 전부 다시 업데이트 필요
2. **partial sort로 search시에 구하기** : O(20,000 * 100 * log10)

Python

1. 미리 heapq로 관리 : 어차피 search 호출시에 전부 다시 업데이트 필요
2. **nlargest로 search**시에 구하기 : O(20,000 * 100 * log10)

해법 연구

api간 tick 사이 booth 정보 업데이트

- booth 별로 완전히 독립적으로 진행
- update 함수 생성

update(bid, tick) : bid booth를 tick시간 까지 업데이트

while

1. finish <= tick
   - 대기인원 존재 : 바로 입장
   - 대기인원 존재X : update 종료
2. finish > tick
   - update 종료

해법 연구

api간 tick 사이 booth 정보 업데이트

1. **search**

   모든 booth tick까지 update

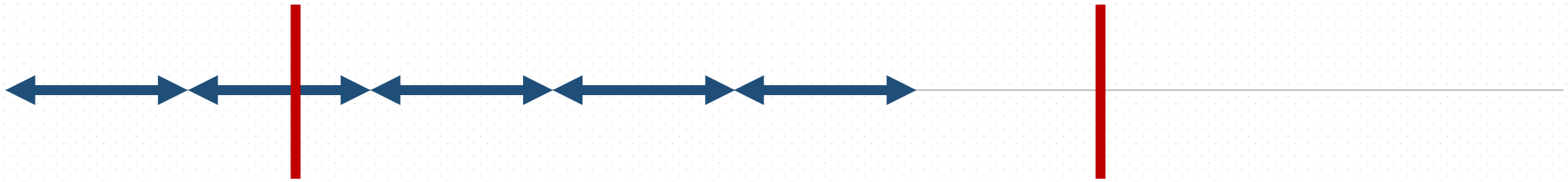
2. **add**
   1. update(bid, tick-1)
   2. 인원 추가
   3. 만약 booth가 멈춰 있으면 finish = tick
   4. update(bid, tick)

add 시,
tick 시점에 맞춰서 끝나는 경우, add 인원 추가 후 업데이트 필요
tick 시점 이전에 끝나는 경우, add 인원 제외 업데이트 필요

=> 우선 tick-1 까지 update 후, 인원 추가 후, tick update 로 해결

인원 추가 없는 경우

: update(bid, tick)



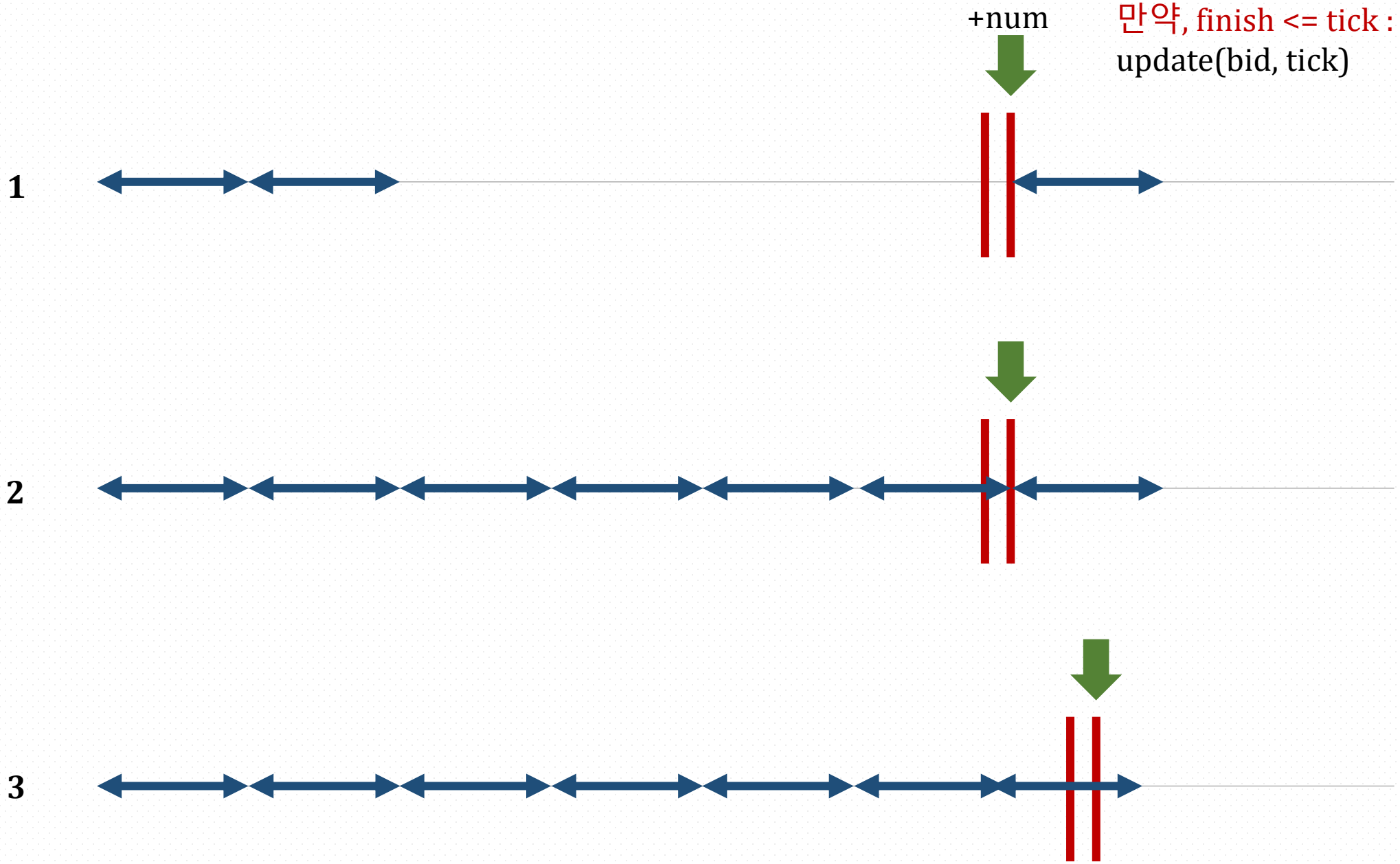
인원 추가 있는 경우

+num

: update(bid, tick-1)
인원추가
update(bid, tick)

인원 추가 있는 경우

: update(bid, tick-1)
인원추가
만약, finish <= tick : finish = tick
update(bid, tick)



감사합니다