

**HONOR CODE:** I have neither given nor received any unauthorized assistance on this assignment.

SWE2026- 41  
2021 Spring

Student ID: 2020314315

Name: 박지원

HONOR  
CODE

박지원 (signature)

### PA 3 – Scholarship Officer

#### 1. Problem analysis (<= 7 lines)

Summarize the problem very briefly. Point out any special points. Explain how you could use these points to implement your solution.

매 쿼리마다 주어지는 Major, Semester, Credits, Number of satisfied graduation requirements를 만족하는 사람들 중에서 특정 tsa를 갖는 사람이 있는지를 알아보거나, 특정 tsa 이상/이하의 tsa를 갖는 사람의 수를 구하는 문제입니다.

#### 2. Lesson from reference (check notice for reference, <= 10 lines)

Check the reference part in the appendix. What is different from our problem? Is there anything to learn from or to apply on our situation? If there is, how does it affect your solution?

(You can research further upon the given reference and write it here.)

Cuckoo hashing을 이 문제에 적용할 수는 없지만, Cuckoo hashing을 통해 배울 수 있는 점은 있습니다.

Cuckoo hashing에서는 같은 해시 값을 갖는 데이터가 여러 번 들어오게 되어서 사이클이 생기게 되었을 때,

Rehash를 통해서 사이클을 제거하고 효율을 늘릴 수 있는데, 이 과정에서 모든 키를 다시 저장해야 하므로

$O(N)$ 의 시간을 소요하게 됩니다. 이번 PA에서는  $N$ 이 큰 값이기 때문에 Rehash의 과정을 거쳐서는

시간 안에 문제를 해결할 수 없습니다. 따라서, 같은 데이터가 여러 번 들어왔을 때 그것들을 count해주는 방식

으로 Cuckoo hashing의 단점을 피해갈 수 있습니다.

Continued next page

### 3. Solution explanation (<= 12 lines)

Which data structures and algorithms are used in your codes? Explain your solution.

Should the subproblems be implemented with different solutions? If they should, why so and what made them different?

Else, why should not?

Major, Semester, Credits, Number of satisfied graduation requirements를 key로 갖는 map을 사용하기 위해 4차원 배열을 사용했습니다. 입력을 받으면서 cnt1(4차원 배열)에 각 key값들을 만족하는 학생의 수를 카운트 해줍니다. 예를 들어, a b c d tsa가 입력으로 들어왔다면, [0][0][0][0], [0][0][0][d], [0][0][c][0], ..., [a][b][c][0], [a][b][c][d]의 16가지 경우를 모두 카운트해주는 것입니다. 이렇게 해서 전체 학생들을 필터링한 결과를 cnt1에 저장하고, 모든 쿼리를 입력받은 다음 각 쿼리를 tsa순으로 정렬해줍니다. tsa순으로 정렬된 쿼리들을 순서대로 돌며 이번 쿼리의 tsa가 어떤 학생의 tsa 이상일 동안 학생들의 정보를 다시 조회하며 cnt2에 count해줍니다. 여기서, 만약 이번 쿼리가 HIGHER이라면 이번 쿼리의 결과는 cnt1(이번 쿼리 필터) – cnt2(이번 쿼리 필터)이고, 이번 쿼리가 FIND또는 LOWER라면 이번 쿼리와 학생의 tsa가 같은 경우까지 모두 cnt2에 count해줍니다. 이 과정에서 지금 쿼리의 필터를 만족하는 학생이 있다면 이번 쿼리의 학생이 존재한다는 것을 기록해줍니다. 위의 과정이 모두 끝났다면, 쿼리가 FIND일 경우에는 위에 존재여부를 기록해둔 것을 이용하고, LOWER라면 cnt2(이번 쿼리 필터)가 이 쿼리의 정답입니다.

---

### 4. Time complexity calculation (<= 9 lines)

Calculate your time complexity by calculating your actual number of operations with worst N, M, and some constants.

Does your big-O notation seem correct? Is calculated value small enough for this problem?

Check the attached appendix to see example of this question.

쿼리를 정렬한 후 모든 쿼리와 모든 학생들을 모두 조회하며 count해주므로 평균적으로  $O(NM)$ 이 걸릴 수 있으나, 쿼리 정렬을 퀵소트로 진행했기 때문에 만약  $M^2 > NM$ 이라면, 최악의 경우에  $O(M^2)$ 일 수 있습니다.

---

Continued next page

**Question 5 and Question 6 shares their line limitation.**

**You can freely distribute the total line less than or equal to 18 lines.**

---

**5. Edge case analysis (Q5. + Q6. <= 18 lines, feel free to distribute limitation)**

Think about edge cases that possibly lead to additional optimizations or cause an error on others' code.

Show some examples of edge cases that you can think of, and explain how your codes/solution dealt with it properly.

만약 저와 같은 방식으로 진행하였다면, HIGHER 쿼리는 tsa가 같은 학생들을 처리하기 전에 진행해야 하고, LOWER또는 FIND쿼리는 tsa가 같은 학생들을 count해준 뒤에 진행해야 하므로 쿼리들을 그냥 tsa에 대한 오름차순으로만 정렬한 경우에 쿼리의 tsa가 같을 경우에 LOWER또는 FIND가 HIGHER보다 먼저 진행된다면 HIGHER의 값이 잘못 나올 수 있습니다. 따라서, HIGHER은 LOWER또는 FIND보다 먼저 처리해주어야 하므로 쿼리를 정렬할 때 tsa가 같다면 HIGHER가 LOWER또는 FIND보다 먼저 오도록 해야합니다.

**6. Discussion about extended case (Q5. + Q6. <= 18 lines, feel free to distribute limitation)**

Think about what if this problem situation is extended further with additional premises A~C. Why and how solution should be changed? What will be the time/space complexity?

Check the appendix attached to notice to see the detailed description of this question.

**A. (Given problem) + (Inserting data and querying occur randomly, not sequentially)**

제 코드와 같은 방식에서는 6-A가 문제가 되지 않습니다. 원래와 같은 방식으로 모든 데이터와 모든 쿼리를 입력받은 후 일정한 기준에 따라 정렬한 후 같은 과정을 진행해주면 됩니다.

**B. (6-A) + (Data is not sorted)**

이것 또한 6-A와 같은 방식으로 진행해주면 됩니다.

**C. (6-A) + (We cannot know beforehand which is dominant, query or data.)**

쿼리의 양이 더 많은 경우와 데이터의 양이 더 많은 경우를 동일하게 처리해주었기 때문에 문제 없이 같은 방식으로 진행할 수 있습니다.

---

**Continued next page**

**7. Others (<= 10 lines, not for grading)**

Letter to professor or TAs, further discussions needed, remarks of this assignments, and others.

Your writings in this question are not for grading.

---

---

**Congratulations on finishing it!**