단계별 모의고사 2회차 해설

문제	
1	작업
2	영상처리
3	문제 추천 시스템 Version 1
4	가운데에서 만나기
5	문자열 제거
6	부품 대여장
7	연산 최대로
8	문제 추천 시스템 Version 2

1. 작업

1. 작업

- 작업 X를 하기 위해서 어떤 작업들을 먼저 해야하는지 찾아내고 그 작업들의 개수를 구하는 문제입니다.
- 작업 순서는 방향 간선을 이용하여 표시되어 있습니다. 이 간선의 방향을 반대로 뒤집으면 작업 X를 하기 위해 먼저 끝내야하는 작업들을 알 수 있습니다.
- 다시말해서, 간선을 뒤집고 작업 X에서 출발하여 그래프 탐색을 할 때, 방문한 정점의 개수(작업 X 제외)를 구하면 됩니다.
- 시간복잡도: $O(N)$

2. 영상처리

2. 영상처리

- 단순한 영상 이진화를 구현하여 흑백이미지를 구하고 컴포넌트(Component)의 개수를 구하는 문제입니다.
- 아래 순서대로 진행하면 문제를 해결할 수 있습니다.
  - 1. 각 픽셀에서 RGB 값을 평균내어 T보다 작다면 0을, 크거나 같다면 255로 변환합니다.
  - 2. 값이 0 또는 255로 변환된 픽셀들에서 Component를 구하는 알고리즘을 이용하여 물체의 개수를 세면 됩니다. 이때, Component를 구하는 알고리즘은 DFS 또는 BFS로 사용해도 됩니다.
- Component를 구하는 문제는 다양하게 많으니 다른 문제도 풀어보시면 됩니다. (ex. BOJ 2667, BOJ 11724, BOJ 4963)

3. 문제 추천 시스템 Version 1

3. 문제 추천 시스템 Version 1

- 문제에서 주어지는 대로 각 함수를 구현하면 되는 문제입니다.
- 주어진 함수를 제한시간 안에 돌 수 잘 구현하려면 자료구조를 잘 활용해야합니다.
- Java 는 TreeMap 또는 PriorityQueue 를, C++ 은 std::set 또는 priority_queue, Python 는 heapq 를 사용하여 문제를 해결하면 됩니다.
- 푸는 방법은 두 가지 (Python 은 한가지) 가 있습니다.

3. 문제 추천 시스템 Version 1

- 우선순위 큐를 이용하여 푸는 방법을 소개해드리겠습니다.
- **가장 어려운 문제**를 맨 위에 올리는 max heap과 **가장 쉬운 문제**를 맨 위에 올리는 min heap, 현재 추천 시스템(heap)에 있는 문제 번호 P의 난이도 L에 대한 정보를 저장하고 있는 배열로 관리하면 됩니다.
- 먼저, 각 문제들에 대한 난이도를 기록하는 배열을 제한에서 벗어난 난이도 값을 하나를 정하여 초기화를 해둡니다. (난이도가 최소 1이기 때문에 보통 0으로 초기화합니다.)
- 처음에 주어지는 추천 문제 리스트에 대한 정보를 max heap와 min heap 두 자료구조에 정보를 담고 현재 주어진 문제 P에 대한 난이도를 배열에 기록해둡니다. 이는 명령어 add를 처리하는 것과 동일합니다.

3. 문제 추천 시스템 Version 1

- ✓ 명령어 solved가 주어지면 문제 P에 대한 난이도를 문제 제한에서 벗어난 값으로 초기화를 하여 추천 문제 리스트에서 문제 P가 없다는 표시를 해둡니다.
- ✓ 명령어 recommend가 주어지면 x가 1이면 max heap에서, x가 -1이면 min heap에서 문제를 뽑아서 출력합니다. 만약, 해당 heap에서 전에 지워진 문제라면 꺼내서 지워줍니다. 이를 추천 문제 리스트에 있는 문제일때까지 반복합니다.

3. 문제 추천 시스템 Version 1

- C++의 std::set와 Java의 TreeMap을 이용한 풀이입니다.
- 우선순위 큐와 다르게 TreeMap 하나와 우선순위 큐에서 사용했던 문제에 대한 정보를 담는 배열 두 가지만 있으면 됩니다.
- 배열은 우선순위 큐 풀이에서 설명한 것과 동일합니다.
- 명령어 recommend가 주어졌을 때 x가 -1이라면 first, x가 1이라면 last 메소드를 이용하여 문제를 뽑으면 됩니다.
- 명령어 add는 특별한 것이 없고 TreeMap에 문제에 대한 정보를 담으면 됩니다. 배열에 문제 P에 대한 난이도 L을 저장하는 것을 잊으면 안됩니다 !
- 명령어 solved가 주어진다면 TreeMap에서 해당 원소를 찾아 지워야합니다. 이때, 위에서 말한 배열에 담긴 문제 P에 대한 난이도 L를 이용하여 해당 원소를 remove를 하면 됩니다.

4. 가운데에서 만나기

- **지문에서 나온 조건들을 잘 정리해야합니다.** 문제를 정리하면 아래와 같습니다.
- 임의의 도시 X 하나를 골라 준형이와 친구들이 있는 각 도시에서 출발하여 최단시간 계산합니다. 도시 X에서 출발해서 최단시간로 이동했을 때 그 중 가장 오래걸리는 시간을 계산합니다. 이런식으로 모든 도시를 한번씩 계산을 하고 값이 가장 작은 도시들을 찾는 문제입니다.
- 최단거리를 구하는 알고리즘은 여러가지가 존재하지만 해당 문제는 N 이 200밖에 안되기 때문에 플로이드 알고리즘을 이용하여 문제를 해결하면 됩니다.
- 플로이드 알고리즘을 이용하면 $O(N^3)$ 만에 모든 서로 다른 두 쌍의 도시 간의 최단시간을 계산할 수 있게 됩니다.

5. 문자열 제거

5. 문자열 제거

- 주어진 문자열 S와 지울 수 있는 문자열와 지웠을 때 얻는 점수가 주어질 때 얻을 수 있는 최대 점수를 구하는 문제입니다. 알파벳 하나를 지웠을 때 점수 1점을 얻을 수 있다는 것을 놓쳐서는 안됩니다.
- 해당 문제는 그리디하게 풀 수 있게 보이지만 그리디하게 풀 수 없습니다.
- 생각해볼 수 있는 그리디는 $\frac{\text{얻을 수 있는 점수}}{\text{지울 문자열의 길이}}$ 가 가장 큰 값부터 처리하는 것을 생각할 수 있습니다.
- 하지만, 이는 주어진 문자열 S 가 abcabc이고 지울 수 있는 문자열은 abc, bc와 점수는 각각 $8, 6$와 같은 케이스는 통과할 수 없습니다.

5. 문자열 제거

- 해당 문제는 그리디가 아닌 다이나믹 프로그래밍을 이용하여 풀어야합니다.
- $DP[i]$: i 번째까지 문자열을 제거하여 얻은 최대 점수라고 가정해봅시다.
- $DP[i] = max(DP[i - x_j] + y_j, DP[i - 1] + 1)$
- (x_j, y_j) 는 i번째 있는 문자를 끝으로 지울 수 있는 문자열 x_j 를 지워서 얻은 점수 y_j 를 의미합니다.
- 위에서 세운 점화식을 기준으로 구현하면 해결할 수 있습니다.

6. 부품 대여장

6. 부품 대여장

- 해당 문제는 문자열을 잘 파싱하고 자료구조를 잘 이용하여 주어진대로 구현하면 되는 문제입니다.
- 모든 문제가 그렇듯 해당문제를 풀면서 값이 중간에 int 범위를 벗어나지 않을까라는 의문을 해야합니다.
- 또한, 한 사람이 한 가지 부품만 빌리는 것이 아니라 두 가지 이상의 부품을 동시에 빌리고 있는 상태일 수 있습니다.
- "이름"과 "부품"을 하나의 문자열로 이어붙여서 관리를 하거나 key가 사람 정보만 담는 HashMap안에 부품 정보를 담는 HashMap을 넣어서 관리해도 됩니다.

7. 연산 최대로

7. 연산 최대로

- 해당 문제는 정수와 연산자의 순서를 적절히 배치하고 괄호를 원하는 만큼 이용하여 식을 통해 만들 수 있는 값의 최댓값을 구하는 문제입니다. **연산자의 우선순위는 없음을 조심해야합니다.**
- 여기서 정수와 연산자의 순서를 배치하는 것은 완전탐색을 통해 배치하면 된다는 것은 알 수 있습니다. 하지만, 괄호를 어떻게 잘 써야 수식의 결과가 최대가 될까요?
- 답은, 주어지는 정수를 보면 양의 정수만 주어지는 것을 알 수 있습니다. 양의 정수만 존재할 경우 더하기 연산을 한 후 곱하기 연산을 하면 값이 가장 커집니다.
- 그러면, 괄호는 더하기 연산을 먼저하고 곱하기 연산을 나중에 하도록 어떻게 배치를 해야할까요?
- 괄호는 무수히 사용해도 된다는 조건이 있기 때문에 어떻게든 연산 순서를 정할 수 있습니다.

7. 연산 최대로

- 예를 들어, $1+2+4+7\times5+8$가 주어졌을 때 더하기 연산을 먼저하고 곱하기 연산을 나중에 하도록 괄호를 치는 방법은 여러가지가 있습니다.
- $(1+2+4+7)\times(5+8)$
- $(((1+2)+4)+7)\times(5+8)$
- 따라서, 정수와 연산자를 완전탐색으로 가능한 배치를 하고 계산 순서는 더하기 먼저하고 곱하기를 나중에 하도록 계산하면 됩니다.

8. 문제 추천 시스템 Version 2

8. 문제 추천 시스템 Version 2

- 해당 문제는 3번 문제 추천 시스템 Version 1를 확장한 문제입니다.
- 이 문제도 마찬가지로 우선순위 큐 또는 TreeMap을 이용하여 문제를 해결할 수 있습니다.
- 알고리즘 분류 G가 추가됐지만 G의 범위를 확인해보시면 최대 100인 것을 알 수 있습니다. 이는 길이가 100인 TreeMap 또는 길이가 100인 우선순위 큐로 관리하면 된다는 것을 알 수 있습니다.
- 이처럼 문제에서 주어지는 제한을 확인을 해보면 힌트를 얻을 수도 있는 문제입니다.
- 나머지 풀이는 문제 추천 시스템 Version 1과 동일합니다.