단계별 모의고사 5회차 해설

문제		백준번호
1	피로도	BOJ22864
2	가장 먼 곳	BOJ22865
3	탑 보기	BOJ22866
4	종점	BOJ22867
5	산책 (small)	BOJ22868
6	징검다리 건너기 (small)	BOJ22869
7	산책 (large)	BOJ22870
8	징검다리 건너기 (large)	BOJ22871

1. 피로도

1. 피로도

- 일과 휴식을 잘 분배하여 최대한 가능한 많은 일을 할 수 있는지 구하는 문제입니다.
- 하루에 할 수 있는 일을 구하라 하였고 휴식 또는 일을 선택하는 두 가지가 있기 때문에 완전탐색으로 문제를 해결할 수 있음을 확인할 수 있습니다.
- 이를 시간복잡도로 계산해보면 $O(2^{24})$ 으로 시간제한 안에 충분히 돌 수 있음을 확인할 수 있습니다.
- 또는 그리디 알고리즘으로도 해결할 수 있습니다.

2. 가장 먼 곳

2. 가장 먼 곳

- 친구 A, B, C가 있는 곳들을 기준으로 가장 멀리 떨어진 곳을 선택하는 문제입니다.
- 이는 친구 A, B, C가 있는 곳에서 각각 출발해서 최단거리로 이동하는 최단거리를 구한 후 어떤 지역이 가장 먼 곳인지 구하면 됩니다.
- 이때, 최단거리를 구할 때 사용해야하는 알고리즘은 다익스트라 알고리즘을 써야합니다.
- 다익스트라를 3번 돌린 후 가장 먼 곳을 계산하면 풀 수 있습니다.

2. 가장 먼 곳

- 이를 잘 생각해보면 다익스트라 한번으로도 가능합니다.
- 처음에 우선순위 큐에 넣을 때 친구 A, B, C를 동시에 다 넣고 다익스트라를 돌리면 친구 A, B, C에서 가장 먼 곳이 어디인지 단 한번의 다익스트라로 구할 수 있습니다.

3. 탑 보기

3. 탑 보기

- 탑에서 보이는 다른 탑들을 구하고 만약 보이는 탑이 존재한다면 그 탑들 중 번호가 가장 작은 것을 구하는 문제입니다.
- 이는 monotone stack을 이용하여 해결할 수 있습니다.
- 이와 비슷한 문제로는 BOJ2493, BOJ17298, BOJ17299, BOJ6198 등이 있습니다.
- 왼쪽 또는 오른쪽 방향으로 볼 수 있는 건물들을 계산한 후 반대 방향을 계산하면 됩니다.
- 또는, 왼쪽 방향을 먼저 봐서 볼 수 있는 건물들이 있는지 여부와 해당 건물을 저장을 했다면 오른쪽 방향을 볼땐, 저장되지 않은 건물들만 업데이트하면 됩니다.
- 시간복잡도 $O(N)$으로 해결할 수 있습니다.

4. 종점

4. 종점

- 문제를 요약하면 종점에 출입기록을 보고 종점에 최대 버스가 몇 개 존재하는지 구하는 문제입니다.
- 이를 구할 때 사용되는 알고리즘은 스위핑이라는 기법으로 비슷한 문제는 BOJ19598입니다.
- 종점에 출입하는 시각을 +1로 두고 종점에서 나가는 시각을 -1으로 할 때 모든 시각을 시간순으로 정렬 후 표시했던 값을 누적하면서 값의 최대값을 구하면 됩니다. 이러한 방법을 스위핑이라고 부릅니다.
- 주어지는 시각정보를 보면 이를 초로 바꾸어 파싱해야할 것처럼 보이지만 파싱을 안하고 풀 수도 있습니다.
- 정렬하는데 가장 많은 시간이 걸리므로 시간복잡도는 $O(NlogN)$ 입니다.

5. 산책 (small)

5. 산책 (small)

- 정점 1에서 출발하여 정점 N으로 도착하는 최단경로를 구하고 정점 N 에서 출발하여 정점 1로 도착하는 최단경로를 구하면 되는 문제입니다.
- 일단, 최단경로(또는 최단거리)를 구할 땐, BFS 또는 Dijkstra를 이용하는데 해당 문제에서 존재하는 간선의 가중치는 모두 동일하기 때문에 BFS를 이용하여 $O(N+M)$ 만에 구할 수 있음을 알 수 있습니다.
- 가능한 모든 최단경로를 구했다고 가정했을 때, 그 중 사전순으로 가장 먼저 오는 최단경로를 어떻게 빠르게 구할 수 있을까요?
- 이는 각 정점에서 연결된 다른 정점들을 정렬한 후 탐색하여 가장 먼저 도착지점에 도달하는 경우가 사전순으로 보았을 때 가장 빠른 경로임을 알 수 있습니다.

5. 산책 (small)

- ✓ 정점 1에서 출발하여 정점 N 으로 가는 경로를 구했다면 해당 경로에서 사용된 간선을 체크해두고 정점 N 에서 정점 1로 도착하는 최단경로를 탐색할 때 체크된 간선을 사용하지 않도록 해야합니다.
- ✓ 총 시간복잡도 $O(NlogN)$ 만에 풀 수 있습니다.

6. 징검다리 건너기 (small)

6. 징검다리 건너기 (small)

- 주어지는 조건을 만족하면서 가장 왼쪽에서 가장 오른쪽으로 이동할 수 있는지 계산하는 문제입니다.
- 이는 그래프 탐색 알고리즘(BFS, DFS)을 이용하여 문제를 해결할 수 있지만, 다이나믹 프로그래밍으로 푸는 방법으로도 풀 수 있습니다.
- 점화식으로는 DP$[i]$: i번째까지 도달할 수 있는지 여부(1은 도달 가능 0은 도달 불가능)으로 세울 수 있습니다.
- DP$[i]$를 계산할 때 왼쪽에 있는 모든 돌을 한번씩 보고 첫 번째 돌에서 출발하여 도달할 수 있는지 계산을 하면 됩니다.
- 그래프 탐색, 다이나믹 프로그래밍 둘다 $O(N^2)$으로 문제를 해결할 수 있습니다.

7. 산책 (large)

7. 산책 (large)

- Small 버전과 간선의 가중치가 주어진다는 것을 제외하면 같습니다.
- Small 버전 풀이 설명에서 언급했듯이 존재하는 간선의 가중치는 모두 다르기 때문에 해당 문제에서는 BFS가 아닌 Dijkstra 알고리즘을 이용해야합니다.
- 해당 문제는 Small 버전에서 했던 것처럼 각 정점과 연결된 정점들을 정렬하면 사전순으로 가장 먼저오는 경로를 구할 수 있을까요? 아쉽게도 이는 반례가 존재합니다.
- 이 문제에서는 정점 1 $\rightarrow$ 정점 N와 정점 N $\rightarrow$ 정점 1의 최단경로를 dijkstra 알고리즘으로 구한 뒤 계산된 거리배열을 이용하여 사전순으로 가장 먼저 오는 계산해야합니다.
- 사전순으로 가장 먼저 오는 최단경로를 구했으면 이때 사용한 간선들을 체크해두고 정점 N $\rightarrow$ 정점 1의 최단경로를 구하면 됩니다.
- 시간복잡도 $O(MlogN)$으로 문제를 해결할 수 있습니다.

8. 징검다리 건너기 (large)

8. 징검다리 건너기 (large)

- 해당 문제는 Small 문제와 다르게 K의 최솟값을 구하는 것이 문제입니다.
- 이는 다이나믹 프로그래밍으로도 풀 수 있지만, Small에서 사용했던 아이디어와 이분탐색을 이용하여 푸는 방법으로 설명하겠습니다.
- 결정해야하는 값은 K 이고 K 값은 1이상 10^6 이하인 정수 중에 하나라는 것을 예측할 수 있습니다.
- K 값에 따라 가장 오른쪽에 있는 돌로 이동할 수 있는지 여부를 계산해보면 어느 한 지점을 기준으로 왼쪽은 불가능, 오른쪽은 가능임을 확인할 수 있습니다.

8. 징검다리 건너기 (large)

- 따라서 K에 대해 이분탐색을 진행하여 조건에 맞는 최소 K를 구할 수 있습니다.
- 이분탐색하는 과정에서 가장 오른쪽으로 도달할 수 있는지 여부를 어떻게 구할까요?
- 이는 K를 결정하고 구하는 것이기 때문에 징검다리 건너기 (small) 버전에서 사용된 풀이를 이용하면 됩니다.
- 이분탐색을 이용하면 $O(N^2 \log 10^6)$만에 해결할 수 있고, 만약 다이나믹 프로그래밍으로 푼다면 $O(N^2)$으로 풀 수 있습니다.