단계별 모의고사 7회차 해설

문제		백준번호
1	데이터 체커	BOJ22942
2	수	BOJ22943
3	죽음의 비	BOJ22944
4	팀 빌딩	BOJ22945
5	원 이동하기 1	BOJ22946
6	실행 시간	BOJ22947
7	원 이동하기 2	BOJ22948
8	회전 미로 탐색	BOJ22949

1. 데이터 체커

1. 데이터 체커

- 해당 문제는 원의 중심이 x좌표 축 위에 존재하고 한점 이상 접한 원이 존재하는지 확인하는 문제입니다.
- 관찰이 필요한 문제로 하나의 원과 x축과 만나는 두 끝점 일부를 남기고 지우면 괄호 두 개 모양 “( )”이 됩니다.
- 주어진 원에 번호를 매긴 후 위의 작업을 통해 나온 괄호에 번호를 매긴다고 해봅시다. 그러면 올바른 괄호 문자열인지 확인하는 알고리즘을 이용하여 괄호의 쌍이 맞는지 스택을 이용하여 확인하면 됩니다. 괄호의 쌍이 맞다는 것은 닫힌 괄호와 열린 괄호가 순차적으로 만나야하며 번호까지 같아야함을 의미합니다.
- 또한 괄호들이 같은 위치에 존재하는 경우도 조건에 위배되므로 NO를 출력하면 되고 그 것이 아니면 올바른 괄호 문자열인지 확인결과를 출력하면 됩니다.

2. 수

2. 수

- K 가지의 숫자를 선택하여 수를 만들어 지문에 있는 두 가지 조건에 해당하는지 체크하는 문제입니다.
- K 가지의 숫자를 이용하여 수를 만들 때 최대값은 10^5 인 것을 알 수 있습니다.
- 10^5 이하의 소수를 미리 뽑아서 1번 조건을 만족하는지 2번 조건을 만족하는지 구현하면 됩니다.
- 소수들을 뽑는 시간복잡도는 에라토스테네스의 체를 이용하면 $O(N\log\log N)$ 이고 루트 시간만에 루트인지 확인하는 알고리즘을 이용하면 $O(N\sqrt{N})$ 만에 전처리를 할 수 있습니다.

3. 죽음의 비

3. 죽음의 비

- 시작점 S에서 출발하여 안전하게 도착점 E까지 도달할 수 있는지 여부와 도달할 수 있다면 얼마나 최소 횟수로 이동하는지 구하는 문제입니다.
- 지도에 벽이 존재하지 않기 때문에 각 Point 간에 이동거리는 택시 기하학을 이용하여 계산하면 됩니다.
- 안전하게 도착지점까지 가기 위해서는 최대한 비를 안맞도록 우산을 쓰고 가는 것이 이득입니다. 근데 어떤 우산을 어떤 순서로 가져가야하는지 또는 가져가지 않고 바로 가도 되는지는 어떻게 구할까요 ?
- 우산의 개수는 최대 10개가 존재하기 때문에 우산의 순서까지 고려한 경우의 수는 어떻게 구할까요 ?

3. 죽음의 비

- 총 경우의 수를 구하는 식은 $\Sigma_{k=0}^{10}{}_{10}P_k$ 와 같습니다.
- 이를 계산해보면 약 10^7 보다 작은 수가 나옵니다.
- 또한 이동경로를 계산하는 과정은 point의 개수와 같기 때문에 총 약 10^8 번의 연산을 합니다.
- 따라서, 완전탐색 같은 경우는 시간안에 돌 수 있음을 파악할 수 있습니다.
- 탐색 중에 다음 point로 가는 도중 사망한다는 판단이 되면 중간에 커팅을 하여 백트래킹으로 풀면 상당히 빠르게 풀 수 있습니다.

4. 팀 빌딩

4. 팀 빌딩

- 해당 문제는 두 포인터로 해결할 수 있는 문제입니다.
- 기존에 풀었던 두 포인터 문제들은 두개의 포인터가 맨 왼쪽에서 시작하여 문제를 해결하는 방식이였다면 이 문제에서는 구간을 늘려가며 탐색할 수 없습니다.
- 이러한 문제는 시작점을 하나의 포인터는 맨 왼쪽, 다른 포인터는 맨 오른쪽, 즉 양 끝점에서 시작하여 탐색을 해야합니다.
- 두 개의 포인터 중 값이 작은 포인터를 안쪽(왼쪽에 있는 포인터라면 오른쪽으로, 오른쪽에 있는 포인터라면 왼쪽으로)으로 이동하면서 탐색하면 됩니다.

5. 원 이동하기 1

5. 원 이동하기 1

- 해당 문제를 좌표평면에서 해결하려면 어려운 문제입니다. 이를 원을 하나의 노드로 보고 서로 다른 원 간의 포함관계를 간선으로 표현한 그래프를 생각해보면 좀 더 쉽게 볼 수 있습니다.
- 여기서 원 A를 포함하는 원들이 존재한다고 할 때 이 중에서 반지름이 가장 작은 원만 간선을 이어야 합니다. 그 이유는 반지름이 가장 작은 원이 아닌 다른 원으로 가기 위해서는 반드시 그 원을 지나가야 하기 때문입니다.
- 이런식으로 그래프를 그려보면 좌표평면을 루트로 하는 트리가 그려짐을 알 수 있습니다.
- 두 원의 포함관계를 계산하여 그래프로 그리는 과정은 반지름 기준으로 정렬을 하는데 $O(NlogN)$ 또는 $O(N^2)$ 로 해결가능하고 $O(N^2)$ 으로 원관의 포함관계를 파악하여 그래프로 만들 수 있습니다.

5. 원 이동하기 1

- 문제를 다시 정리해보면 좌표평면을 루트로 하는 트리에서 루트를 제외한 나머지 서로 다른 임의의 두 노드를 선택하여 사이의 거리를 구할 때 그 값들 중 최댓값을 구하는 것이 문제입니다.
- 이는 트리에서 지름을 구하는 문제로 알려져 잘 알려져 있으며 이는 BFS를 두 번 돌리거나 Tree DP를 이용하여 풀 수 있습니다.
- 트리의 지름을 구하는 문제는 두 문제(BOJ1167, BOJ1967)를 추가로 풀어보면 좋습니다.

6. 실행 시간

6. 실행 시간

- 작업 K 개를 잘 골라서 모든 작업이 끝나는 시간을 최소화할 때 최소 시간을 구하는 문제입니다.
- K는 최대 3개이고 N 개 중에 K 를 고르는 경우의 수는 $\binom{N}{K}$ 로 최대 $161\,700$ 임을 알 수 있습니다. 이때, $\binom{N}{K}$ 는 Combination으로 $\frac{N!}{K! \times (N-K)!}$ 입니다.
- 위상정렬을 이용하여 모든 작업이 끝나는 시간을 구하는 방법을 알아야합니다.
- indegree가 0인 곳부터 작업을 시작할 수 있으므로 indegree가 0인 곳부터 작업을 시작하고 다음 작업에 작업이 언제 끝났는지 넘겨주면 됩니다. 하나의 작업 전에는 여러가지 작업이 있을 수가 있기 때문에 그 값들 중 최댓값만 갱신하도록 구현하면 됩니다.
- 모든 작업이 끝나는 시간을 구하는 것은 $O(N+M)$ 으로 구할 수 있습니다.

6. 실행 시간

- 따라서, 작업 K 개를 고른 후 작업 시간을 0 으로 바꾼 후 모든 작업이 끝나는 시간을 계산하면 됩니다.
- 가능한 조합을 선택하고 계산하는 모든 과정은 $O(\binom{N}{K}(N+M))$ 으로 해결할 수 있습니다.
  최대 가능한 조합의 수는 $161\,700$ 이였으므로 약 9.7×10^7 의 계산을 하고 제한시간 2 초안에 돌아갈 수 있음을 계산할 수 있습니다.
- 위상정렬로 모든 작업이 끝나는 시간을 계산하는 문제를 더 풀어보고 싶다면 BOJ1516 를 풀어보시면 됩니다.

7. 원 이동하기 2

7. 원 이동하기 2

- 해당 문제는 원 A 내부에서 출발하여 원 B 내부로 도착할 때 방문한 구간의 개수를 구하는 문제입니다.
- 이 문제는 원 A와 원 B에서 각각 출발해서 좌표평면에서 만날때까지 상위 원을 찾아 계속 나가면 됩니다.
- 이때 상위 원을 찾는 방법은 주어진 모든 원은 만나지 않기 때문에 반지름이 작은 순서대로 정렬한 후 반지름이 자신보다 큰 원들을 하나씩 보면서 포함관계에 있는지 체크하면 됩니다.
- 이런식으로 좌표평면까지 이동했다면 원 B 내부에서 출발하여 좌표평면으로 이동한 경로를 뒤집어서 원 A 내부에서 출발하여 좌표평면으로 이동한 경로와 합치면 됩니다.
- 시간복잡도 $O(N)$ 으로 해결할 수 있습니다.

8. 회전 미로 탐색

8. 회전 미로 탐색

- 흔히 알고 있는 미로탐색 문제에서 4×4 크기의 구역들로 나누어 각 구역을 회전시켜 만든 문제입니다.
- BFS로 푸는 문제이지만 BFS에 대해 상당히 잘 이해를 하고 있어야하고 탐색 상태를 잘 나타내야합니다.
- 현재 위치한 구역을 제외한 나머지 구역들은 원래 상태로 돌아가고 회전을 하지 않고 현재 위치한 구역만 행동(정지, 상하좌우로 한칸 이동)을 할 때마다 회전이 됩니다.
- 일반적인 미로탐색에서 탐색 상태공간을 어떻게 정의했는지 생각해보면 visited$[i][j]$와 같이 (i, j)에 방문했는지 여부만 체크했었습니다.
- 하지만, 이 문제는 각 구역이 회전을 하기 때문에 탐색 상태를 visited$[k][i][j]$과 같이 정의를 해야합니다.

8. 회전 미로 탐색

- 탐색 상태 visited$[k][i][j]$는 현재 (i, j)에 있는데 현재 있는 구역이 원래 상태에서 시계방향으로 k번 90도 회전한 경우를 의미합니다. k가 4일 경우 k가 0인 경우와 동일하기 때문에 k는 최대 4, i와 j는 최대 500이기 때문에 메모리제한보다 상당히 적은 메모리로 관리할 수 있습니다.
- 또한, 모든 상태를 탐색하는 시간복잡도는 $O(4N^2)$로 예상이 됩니다. 그런데 과연 그럴까요? 구현 방법에 따라 시간복잡도가 다릅니다.
- 어떤 행동을 할때마다 구역이 회전하기 때문에 하나의 행동을 선택할때마다 구역이 회전하는 비용이 발생합니다.

8. 회전 미로 탐색

- 근데 탐색할 때 계속 구역을 회전시켜야할까요? 각 구역이 회전한 경우를 미리 저장하고 있으면 됩니다. 그러면, 탐색할 때 구역을 회전시키지 않고도 바로 다음에 어디로 가야하는지 알 수 있습니다.
- 가로, 세로의 길이를 N 으로 할 때, 각 구역을 회전시키는데 $O(4N^2)$ 의 시간복잡도가 발생하고 탐색시 구역을 회전하는 시간복잡도를 $O(1)$ 로 줄일 수 있습니다.
- 각 구역을 회전시킨 배열을 미리 계산을 했다면 BFS 알고리즘을 이용하여 미로탐색을 진행하면 되는데 여기서 어떤 행동을 할 때 다음에 가야할 위치가 어떻게 되는지 구하는게 혼란이 올 수 있습니다.
- 이는 반드시 종이에 직접 그려보면서 어떤식으로 돌아가는지 계산 후에 코드를 짜는 과정을 거쳐야합니다.