김범수

# 탐색 문제 풀이

## Search Solution

By POSCAT

# 차이를 최대로

백준 10819

https://www.acmicpc.net/problem/10819

- 이웃한 원소의 차이의 합의 최댓값을 출력하는 문제입니다. 이 때 수열의 순서는 임의로 변경할 수 있습니다.
- N의 범위가 매우 작으므로 모든 수열 조합을 만들어 계산한 다음 최댓값을 출력하는 브루트 포스 방법으로 계산할 수 있습니다.
- 수열의 종류는 N!이고, 수열에서 주어진 식을 계산하는 데 걸리는 시간은 N이므로, 총 N * N!번의 연산이 필요합니다.
- 하지만 N은 최대 8이므로, 8 * 8! = 약 30만으로 충분히 시간 안에 통과할 수 있습니다.

# 스택

백준 10828

https://www.acmicpc.net/problem/10828

- c++ stack을 사용해보는 문제입니다.
- 단순 구현 문제이므로 풀이는 건너뛰겠습니다.

# 큐

백준 10845

https://www.acmicpc.net/problem/10845

- c++ queue을 사용해보는 문제입니다.
- 단순 구현 문제이므로 풀이는 건너뛰겠습니다.

# DFS와 BFS

백준 1260

https://www.acmicpc.net/problem/1260

- DFS, BFS 실행 결과를 출력하는 문제입니다.
- 단순 구현 문제이므로 풀이는 건너뛰겠습니다.

# 토마토

백준 7576

https://www.acmicpc.net/problem/7576

- BFS를 사용하면, BFS 시작 지점 start에서 특정 지점까지의 최단 경로의 길이를 구할 수 있습니다. start에서 pos 지점까지의 거리를 dist[pos]라고 정의하겠습니다. 그러면 dist[start] = 0이 됩니다.
- 만약 BFS 과정에서 pos지점에서 next 지점으로 이동했다면, dist[next] = dist[pos] + 1로 저장을 해 줍니다. 이 방식을 이용하면 BFS가 끝났을 때 특정 지점에서 모든 지점까지의 최단 거리를 구할 수 있습니다.
- 토마토 문제에서 특정 토마토가 익었다면 다음 날에는 그 토마토의 상하좌우에 있는 토마토도 익게 됩니다. 이 때 모든 토마토가 익기 위해서는 며칠이 필요한 지를 출력하는 문제입니다.
- 익은 토마토는 1, 안익은 토마토는 0, 빈칸은 -1로 입력이 들어옵니다.

# 토마토

백준 7576

https://www.acmicpc.net/problem/7576

- 여기서는 편의를 위해 처음이 익은 토마토들의 dist를 1로 저장합니다.
- 가장 처음에 익은 토마토들의 dist를 1로 저장하고, queue에 익은 토마토의 위치를 넣어줍니다.
- BFS를 돌려주면서 dist[next] = dist[pos] + 1의 점화식으로 거리를 계산해줍니다. 이 과정을 queue가 빌 때까지 진행합니다.
- 만약 dist가 0인 토마토가 있다면, 이는 토마토가 벽에 막혀 익지 않았다는 뜻이 됩니다. 따라서 이 경우는 -1을 출력합니다.
- 모든 토마토가 익었다면, dist[pos] – 1은 pos 위치에 있는 토마토가 익는 데 걸린 시간을 의미하므로, 모든 토마토에 대해 max(dist[pos] – 1)가 모든 토마토가 익는 데 걸리는 시간이 됩니다.

# 단지번호붙이기

백준 2667

https://www.acmicpc.net/problem/2667

- 모든 지점을 훑어보면서 만약 1인 지점이 있다면 그 지점을 시작점으로 탐색(BFS, DFS 모두 가능)을 진행합니다. 이 때 탐색을 할 때, 탐색한 지점의 개수를 저장합니다. 또한 탐색한 지점의 숫자는 또 탐색하는 것을 방지하기 위해 0으로 변경합니다.
- 탐색이 종료되면 해당 단지에서 집의 개수가 계산되었을 것입니다. 이 값을 vector, 배열 등을 이용해 저장합니다.
- 모든 지점을 다 훑어보면 모든 지점이 0, 즉 모든 단지를 찾은 이후이므로, 배열을 정렬한 후 정답을 출력합니다.

# 마치며

- 구현 및 소스 코드는 아래의 주소에서 확인할 수 있습니다.

POSCAT

# Thank you :-)