[Baekjoon] 17485 - 진우의 달 여행 (Large)

공개여부	☑
작성일자	@November 4, 2021
태그	baekjoon

지구에서 달까지 가는데 드는 최소비용을 구하는 문제이다. 우주선은 아래 양 대각선과 아래로 이동할 수 있다. 정확한 위치가 없이, 달에만 도달하면 된다는 점과 연료가 전부 명시되어있는점을 미루어 보았을때, 그리고 행렬의 크기가 최대 1000 * 1000 인것으로 보아 100000만이 최대이고 원소값도 100이하의 자연수이기때문에, 100씩 1000번을 내려와도 10만임을 알 수 있다. DP문제라는것을 느낌적으로 바로 알 수 있었다.

바로 생각난 알고리즘은 간단했다. 1001*1001짜리 int 형 공간을 미리 할당해두고, N, M값에 맞게 입력을 받는다. M이 아래로 내려가는 선택지이고, N이 층의 역할이기 때문에, DP문제 답게 모든 M가지 선택지를 고려할 수 있게끔 할 수 있다.

예제입력1번으로 예시를 들자면, 0층 5 8 5 1과 그 아래 1층 3 5 8 4가 주어졌을때, 1층 3은 0층의 5, 8중에 작은 수에서 이동하게 된다. 1층의 5는 0층의 5, 8, 5중에서 가장 작은 수를 선택해 이동하게 된다. 이러면 매 층을 이동하려 할때마다, 이미 이동된 과거의 값을 통해서 최솟값

을 계속 구하며, DP형식으로 이동하게 되기 때문에 결과적으로 메모이제이션, DP로 해결한것이 된다.

문제는 근데 똑바로 **"끝까지"** 읽어야한다. 왜냐면 방금 끝까지 안읽고 이렇게만 풀었다가, 자꾸 29가 안나오고 27이 나오길레 뭐지..하고 문제를 봤는데, **같은방향으로 두번 연속으로 움직일 수 없다**고 한다. 제발 ***문제는 끝까지 읽자.***

```
#include <iostream>
#include <algorithm>

using namespace std;

int memo[1001][1001] = { 0, };

int main(void)
{
    ios::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);

    int N, M;
    int result = 100001;
    int past[1001];

    cin >> N >> M;
    for (int i = 0; i < N; i++) {
        for (int k = 0; k < M; k++)
            cin >> memo[i][k];
    }
    for (int k = 0; k < M; k++)
        past[k] = 2;

    for (int i = 1; i < N; i++) {
        for (int k = 0; k < M; k++) {
            if (k == 0) {
                if (memo[i - 1][k] > memo[i - 1][k + 1] && past[k] != -1) {
                    memo[i][k] += memo[i - 1][k + 1];
                    past[k] = 1;
                }
                else {
                    memo[i][k] += memo[i - 1][k];
                    past[k] = 0;
                }
            }
            else if (k == M - 1) {
                if (memo[i - 1][k] > memo[i - 1][k - 1] && past[k] != -1) {
                    memo[i][k] += memo[i - 1][k - 1];
                    past[k] = 1;
                }
                else {
                    memo[i][k] += memo[i - 1][k];
                    past[k] = 0;
```

```
                }
            }
            else {
                int idx, tmp = 101;
                for (int h = -1; h <= 1; h++) {
                    if (tmp > memo[i - 1][k + h] && past[k] != h * (-1)) {
                        tmp = memo[i - 1][k + h];
                        idx = h;
                    }
                }
                memo[i][k] += tmp;
                past[k] = idx;
            }
        }
    }

    for (int i = 0; i < M; i++)
        if (memo[N - 1][i] < result)
            result = memo[N - 1][i];

    cout << result;

    return 0;
}
```

근데 충격적이게도 이를 변경해도 27이 나오더라(????)

머리가 띵해져서 잘 봤는데, 아무리봐도 이상한게 없다.

시간을 가지고 좀 보았다. 그러다보니 문제를 하나 찾았는데, 생각보다 복잡한 문제였다.

1. 만약 비교하는 숫자들이 같을경우, 그 수가 어디서왔는지 어떻게 정할것인가?
2. 기존에 사용하던 past가 수정된다면, 그 다음값은 그 past를 그대로 사용하게된다.

1, 2번의 경우 둘다 위 방식으로는 해결 할 수 없다. 의도는 알겠으나, 이렇게 코드를 짜선 안된다는걸 깨달을 수 있었다. 각각 어디에서 왔는지를 생각해야하기 때문에, 나는 결국 memo배열을 1001 * 1001 * 3 구조로 공간을 할당하기로 하였다. 본인이 어디에서 왔는지 알기위해서는 이방법이 제일 맞다고 생각했다. 마지막 3번째가 본인이 어디에서 왔는지를 나타내는 값이니, 현재 값을 계산할때는 그것에 상반되는 값만 더하면 된다. 계산하는 형식은 내가 위에서 정리한 것과 같다.

```
#include <iostream>
#include <algorithm>
```

```
using namespace std;

int oil[1001][1001] = { 0, };
int memo[1001][1001][3] = { 0, };

int main(void)
{
    ios::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);

    int N, M;
    int result = 100001;

    cin >> N >> M;
    for (int i = 0; i < N; i++)
        for (int k = 0; k < M; k++)
            cin >> oil[i][k];

    for (int k = 0; k < M; k++) {
        memo[0][k][0] = oil[0][k];
        memo[0][k][1] = oil[0][k];
        memo[0][k][2] = oil[0][k];
    }

    for (int i = 1; i < N; i++) {
        for (int k = 0; k < M; k++) {
            if (k == 0) {
                memo[i][k][0] = 100001;
                memo[i][k][1] = memo[i - 1][k][2] + oil[i][k];
                memo[i][k][2] = min(memo[i - 1][k+1][0], memo[i - 1][k+1][1]) + oil[i][k];
            }
            else if (k == M - 1) {
                memo[i][k][0] = min(memo[i - 1][k-1][1], memo[i - 1][k-1][2]) + oil[i][k];
                memo[i][k][1] = memo[i - 1][k][0] + oil[i][k];
                memo[i][k][2] = 100001;
            }
            else {
                memo[i][k][0] = min(memo[i - 1][k-1][1], memo[i - 1][k-1][2]) + oil[i][k];
                memo[i][k][1] = min(memo[i - 1][k][0], memo[i - 1][k][2]) + oil[i][k];
                memo[i][k][2] = min(memo[i - 1][k+1][0], memo[i - 1][k+1][1]) + oil[i][k];
            }
        }
    }

    for (int k = 0; k < M; k++)
        for (int h = 0; h < 3; h++)
            if (result > memo[N - 1][k][h])
                result = memo[N - 1][k][h];

    cout << result;

    return 0;
}
```