출제 이유

- DP (dynamic programming) 기본기 다지기
- 1주차 2번째 문제 비슷한 유형 다시 풀어보기

참고 알고리즘

- DP의 Memoization(메모이제이션) : 배열에 저장하는 방식

- 11048 (이동하기) 문제의 경우 누적해야할 값과 배열이 눈에 띄지만 어려운 문제의 경우 “점화식”을 잘 생각해야함

코드 소개

```cpp
int main() {
    int n, m;
    cin >> n >> m;

    int candy[1001][1001] = {0};

    for (int i = 0; i < n; i++) {
        for (int j = 0 ; j < m; j++) {
            cin >> candy[i][j];
        }
    }

    // p * q 배열에서 반복문
    for (int p = 0; p < n; p++) {
        for (int q = 0; q < m; q++) {
            // 첫번째 방에서는 경우의 수가 없으므로 continue
            if (p == 0 && q == 0) continue;
            // 첫번째 행은 위쪽에 방이 없으므로 바로 왼쪽 방에 있는 사탕값을 더해주고 진행
            else if (p == 0) candy[p][q] += candy[p][q-1];
            // 첫번째 열은 왼쪽에 방이 없으므로 바로 위쪽 방에 있는 사탕값을 더해주고 진행
            else if (q == 0) candy[p][q] += candy[p-1][q];

            // 위쪽 방과 왼쪽 방이 값을 크기 비교한 뒤 더 사탕을 많이 가질 수 있는 방의 사탕값을 더해주면서 진행
            else {
                if (candy[p-1][q] >= candy[p][q-1]) candy[p][q] += candy[p-1][q];    // 위쪽 방의 사탕값이 큰 경우
                else candy[p][q] += candy[p][q-1];  // 왼쪽 방의 사탕값이 큰 경우
            }
        }
    }

    cout << candy[n-1][m-1] << endl;

    return 0;
}
```

핵심코드!!

시간 복잡도

- 메모제이션의 경우 시간복잡도는 O(n)