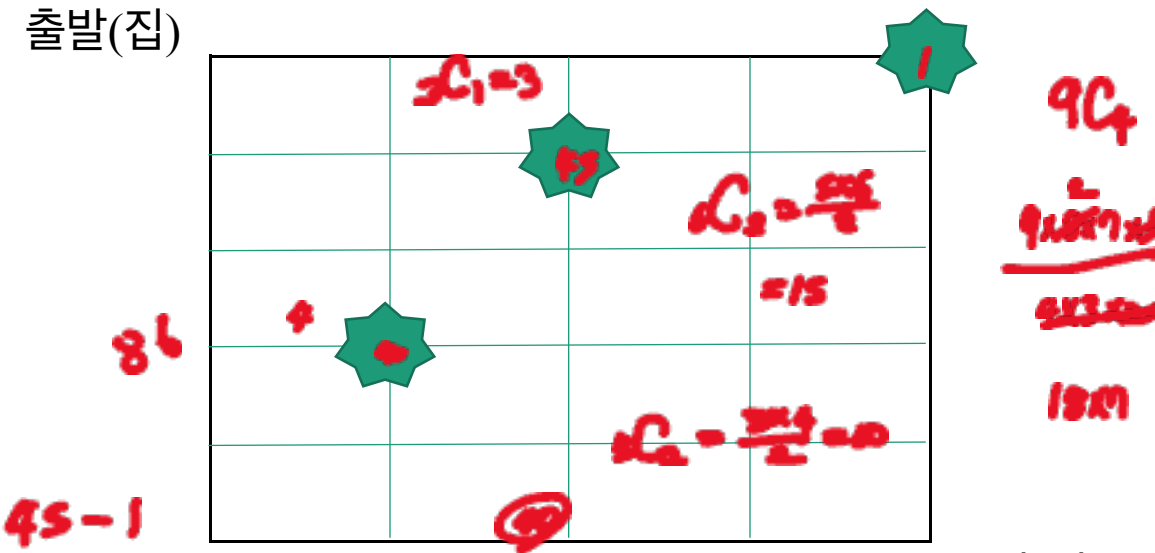
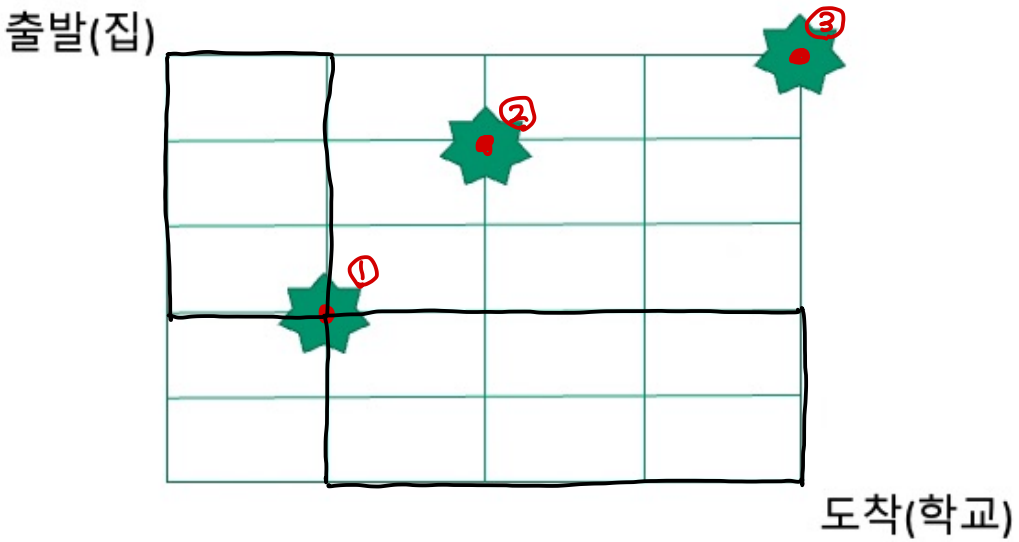
동적 계획 문제 1

명지대학교
컴퓨터공학과
이 충기 교수

문제 1: 통학길의 수

길동은 집에서 학교까지 걸어서 간다 . 그는 항상 최단 경로를 이용한다. 아래 그림은 그의 집에서 학교까지의 지도이다. 별표로 표시된 공사중인 곳은 지나갈 수 없다. 그는 왼쪽 위의 집에서 오른쪽 아래의 학교에 도착해야 한다. 그가 통학길로 선택할 수 있는 경로는 모두 몇 개인가?





방법-1

기본적인 생각은 다음과 같다: 공사장이 없을 경우 전체 가짓수에서 막힌 점을 거쳐 가는 경우를 뺀다.

전체 가짓수: 가로 4, 세로 5 → ${}_9C_4$ 가지 $= \frac{9\times\overset{2}{\cancel{8}}\times7\times\cancel{6}}{\cancel{4}\times3\times\cancel{2}\times1} = 9\times2\times7 = 126$ 가지

①번 공사장을 거쳐 가는 경우: 가로 1, 세로 3 먼저 이동 후 가로 3, 세로 2 이동하므로 ${}_4C_1 \times {}_5C_3 = 4\times\frac{5\times\overset{2}{\cancel{4}}\times3}{\cancel{3}\times\cancel{2}\times1}$

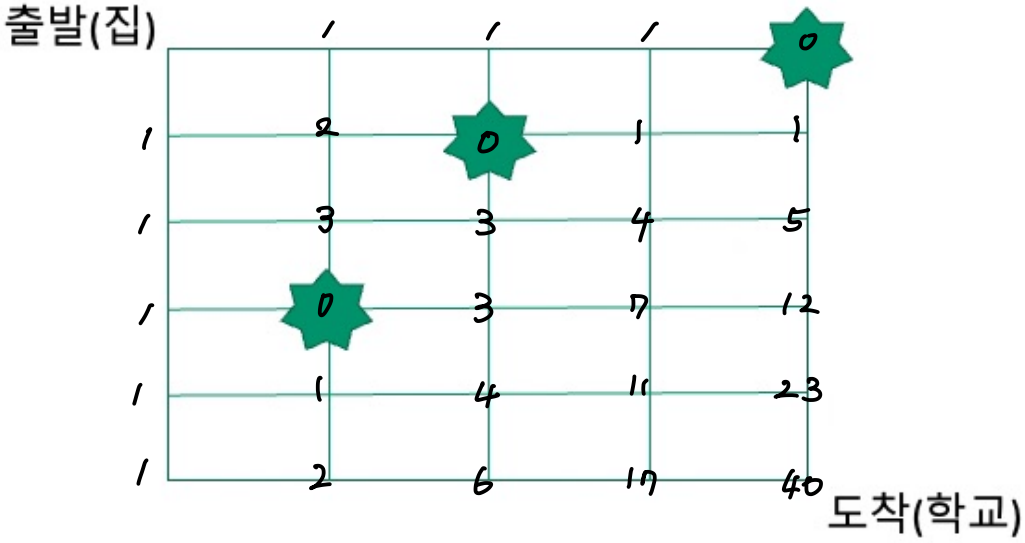
$= 40$가지

②번 공사장을 거쳐 가는 경우: 가로 2, 세로 1 먼저 이동 후 가로 2, 세로 4 이동하므로 ${}_3C_1 \times {}_6C_2 = 3\times\frac{\overset{3}{\cancel{6}}\times5}{\cancel{2}}$

$= 45$가지

③번 공사장을 거쳐 가는 경우: 가로 4 먼저 이동 후 세로 5 이동하므로 ${}_4C_4 \times {}_5C_5 = 1$ 가지

$126 - 40 - 45 - 1 = 126 - 86 = 40$ 가지



방법-2

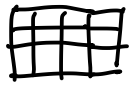
막힌 공사장을 가는 경우의 가짓수를 0으로 놓고 경로마다 가짓수 계산

위와 같이 계산된다.

따라서 도착지 까지는

40가지의 경우의 수 존재

문제 2: 통학길의 수 구하기



길동은 집에서 학교까지 최단 경로를 이용하여 걸어서 간다. 그의 집에서 학교까지의 격자형 모양의 지도가 있다. 지도에는 동서로 ***m*** 개의 도로가 있고 남북으로 ***n*** 개의 도로가 있다. 지도에서 별표로 표시된 공사중인 교차점은 지나갈 수 없다. 집과 학교는 각각 지도의 왼쪽 위 모서리와 오른쪽 아래 모서리에 있다.

1. 집에서 학교까지 갈 수 있는 최단 경로의 수를 구하는 점화식을 적어라.
2. 집에서 학교까지 갈 수 있는 최단 경로의 수를 구하는 분할 정복 알고리즘을 작성하라.
3. 집에서 학교까지 갈 수 있는 최단 경로의 수를 구하는 동적 계획 알고리즘을 작성하라.

2-1. 점화식 (앞 문제 답의 방법2 채택)

M이라고 하는 공사장의 위치를 1로, 공사장이 없는 위치를 0으로 나타내는 배열이 있다고 가정하면, 각 경로마다 가는 방법의 가짓수를 나타내는 A라는 배열은 :

① M[i][j] = 1일 경우 (0≤ i< m) (0≤j < n)

A[i][j] = 0

② M[i][j] = 0인 경우 (0≤j <n) (0≤i<m)

A[0][0] = 1

A[i][0] = A[i-1][0] (0<i<m)

A[0][j] = A[0][j-1] (0<j <n)

A[i][j] = A[i-1][j] + A[i][j-1] (0<j <n)(0<i<m)

2-2. 분할 정복 알고리즘

```java
public static int ways1(int M[][], int m, int n){ // 1. 매개변수로 공사중인 위치를 나타낸 배열 M과, 동서로 뻗은 길의 개수 m, 남북으로 뻗은 길의 개수 n
    // 배열의 인덱스는 0 ~ m-1, 0 ~ n-1 까지 존재하므로
    if (M[m-1][n-1] == 1){ // 2-1. M 배열의 해당 인덱스가 1인 경우 공사장이 있다는 뜻이므로 0을 반환
        return 0;
    }
    else if(m - 1 == 0 && n - 1 == 0){ // 2-2. M 배열의 해당 인덱스가 0인 경우 1 반환
        return 1;
    }
    else if(m-1 == 0){ // 2-3. M 배열의 첫 번째 인덱스가 0인 경우 ways1(M, m, n-1) 반환
        return ways1(M, m, n: n-1);
    }
    else if(n-1 == 0){ // 2-4. M 배열의 두 번째 인덱스가 0인 경우 ways1(M, m-1, n) 반환
        return ways1(M, m: m-1, n);
    }
    else { // 2-5. 그 외의 경우는 ways1(M, m-1, n) + ways1(M, m, n-1) 반환
        return ways1(M, m: m-1, n) + ways1(M, m, n: n-1);
    }
}
```

2-3. 동적 계획 알고리즘

```java
    public static int ways2(int M[][], int m, int n){ // 1. 매개변수로 공사중인 위치를 나타낸 배열 M과, 동서로 뻗은 길의 개수 m, 남북으로 뻗은 길의 개수 n
        int[][] A = new int[m][n]; // 2. M과 같은 크기의 2차원 배열 A를 생성
        for(int i = 0; i < m; i++){ // 3. 지역변수 i가 0 ~ m-1까지 증가하는 동안
            for(int j = 0; j < n; j++){ // 4. 그 반복문 속에서 지역변수 j가 0 ~ n-1까지 증가하는 동안
                if(M[i][j] == 1){ // 4-1. M[i][j]가 1인 경우 A[i][j] = 0
                    A[i][j] = 0;
                }
                else if(i == 0 && j == 0){ // 4-2. i가 0이고 j가 0인 경우 A[i][j] = 1
                    A[i][j] = 1;
                }
                else if(i == 0){ // 4-3. i가 0인 경우 A[i][j] = A[0][j - 1]
                    A[i][j] = A[0][j - 1];
                }
                else if(j == 0){ // 4-4. j가 0인 경우 A[i][j] = A[i - 1][0]
                    A[i][j] = A[i - 1][0];
                }
                else { // 4-5. 그 외에는 A[i][j] = A[i - 1][j] + A[i][j - 1]
                    A[i][j] = A[i - 1][j] + A[i][j - 1];
                }
            }
        }
        return A[m-1][n-1]; // 5. 최종적으로 A배열의 맨 오른쪽 아래 모서리의 값인 A[m-1][n-1]를 반환
    }
}
```