백준 코테 연습 - DP (7/28)

🍬 11048_이동하기

방법 1 (올 수 있는 방법)

항상 아래와 오른쪽으로만 갈 수 있음

D[i][j] = (1, 1)에서 시작해서 (i, j)에 도착했을 때 사탕 개수의 최댓값

(i, j)에 도착했을 때 가능한 경우의 수 (D[i][j]를 만들 수 있는 방법 3가지):

case 1) (1, 1) → (i, j - 1) → (i, j) 이때, D[i][j] = D[i][j - 1] + A[i][j]

case 2) (1, 1) → (i - 1, j) → (i, j) 이때, D[i][j] = D[i - 1][j] + A[i][j]

case 3) (1, 1) → (i - 1, j - 1) → (i, j) 이때, D[i][j] = D[i - 1][j - 1] + A[i][j]

→ D[i][j] = max(case 1, case 2, case 3) + A[i][j]

범위 검사를 굳이 안해도 되는 이유: A[i][j] ≥ 0 이고 for문을 1부터 돌리기 때문

방법 2 (가는 방법)

(i, j)에서 갈 수 있는 경우의 수 (D[i][j]를 만들 수 있는 방법 3가지):

case 1) (1, 1) → (i, j) → (i, j +1) 이때, D[i][j + 1] = max(D[i][j] + A[i][j + 1], D[i][j + 1])

case 2) (1, 1) → (i, j) → (i + 1, j) 이때, D[i + 1][j] = max(D[i][j] + A[i + 1][j], D[i + 1][j])

case 3) (1, 1) → (i, j) → (i + 1, j + 1) 이때, D[i + 1][j + 1] = max(D[i][j] + A[i + 1][j + 1], D[i + 1][j + 1])

배열의 인덱스를 +1 해서 넉넉히 잡으면, 따로 범위 검사 필요 X

방법 3

대각선 이동은 오른쪽→아래 / 아래→오른쪽 두 방법보다 항상 사탕 개수가 적거나 같으므로 고려하지 않아도 됨 (A[i][j] ≥ 0 일 때만)

방법 4 (Top down) (방법 1 변형)

위 3가지 for문 방식은 Bottom up 방식이었다면,

재귀를 이용한 Top down 방식도 사용할 수 있음

이 방식에서 가장 중요한건 **메모이제이션** (사탕 개수가 0일수도 있으므로 배열을 -1로 초기화하기)

방법 5 (Top down) (방법 2 변형)

🏃11060_점프 점프

방법 1 (어디에서 올 수 있는지)

D[i] = i번 칸에 도착할 수 있는 최소 점프 횟수

j번째 칸에서 i번째 칸으로 갈 때(j < i), i - j ≤ Aj 임

D[i] = min(D[j]) + 1

i번 칸에 오기 전 어떤 j번 칸에서 와야 최소인지 알 수 없으므로 모든 칸에 대해 조사해야 함

방법 2 (어디로 갈 수 있는지)

점화식은 똑같음