Duyệt bằng cách sử dụng đệ quy

Cách tệ nhất là ta phải duyệt toàn bộ các con đường từ (1, 1) -> (R, C). Với mỗi vị trí, ta phải duyệt 2 trường hợp: đi sang phải và đi lên trên. Và con đường ngắn nhất là ta phải đi qua (R + C) điểm. Như vậy, độ phức tạp với cách này ước tính vào khoảng $O(2^{R+C})$ (không hề tối ưu). Tuy nhiên, với cách này ta vẫn có thể ăn được 60% số test

Quy hoạch động

Nhược điểm của cách trên là tính đi tính lại các đường đi chung nhau mà không sử dụng lại. Với cách này, ta sẽ tiếp cận theo chiều ngược lại:

Gọi dp[i][j][k] là số cách đi từ ô (1, 1) tới ô (i, j) và đi qua k điểm trong tập P. Vì vậy, giả sử ta cần đạt đến trạng thái là vị trí (i, j) và đi qua k điểm trong tập P. Để đạt được trạng thái này ta chỉ có thể đi từ 2 trạng thái sau:

- Đi từ dưới lên:
  - dp[i-1][j][k-1] nếu (i, j) thuộc P
  - dp[i-1][j][k] nếu (i, j) không thuộc P
- Đi từ trái sang:
  - dp[i][j-1][k-1] nếu (i, j) thuộc P
  - dp[i][j-1][k] nếu (i, j) không thuộc P

Do chỉ có thể đi từ hai trạng thái trên, nên số cách đến trạng thái hiện tại (i, j, k) là tổng số cách đi từ 2 trạng thái trước đó. Như vậy, ta có công thức sau:

- Nếu (i, j) thuộc P:
  - $dp[i][j][k] = dp[i-1][j][k-1] + dp[i][j-1][k-1]$
- Nếu (i, j) không thuộc P:
  - $dp[i][j][k] = dp[i-1][j][k] + dp[i][j-1][k]$

Ta có thể đánh giá được điều kiện dừng của công thức trên là:

- dp[1][1][0] = 1 nếu (i, j) thuộc P, ngược lại là 0.
- dp[1][1][1] = 0 nếu (i, j) thuộc P, ngược lại là 1.
- (Đây là 2 trạng thái khởi đầu của tất cả các trạng thái sau này)

Như vậy, trạng thái ta cần tìm là dp[R][C][k] với k thuộc [0, K]

Với cách này, ta chỉ cần duyệt toàn bộ cặp (i, j, k) là có thể tính được toàn bộ đáp án. Độ phức tạp là $O((K+1).R.C)$