

“초등부 4번. 양손에 V” 문제 풀이

작성자: 조승현

부분문제 1

두 번의 V자 색칠하기를 하는 모든 경우를 다 시도해 볼 수 있다. 흰 격자를 선택하는 방법은 $O(NM)$ 가지이고, V자 색칠하기를 시뮬레이션 하는 시간은 $O(\min(N, M))$ 이므로 첫 V자 색칠하기, 두번째 V자 색칠하기, 시뮬레이션까지 고려한 시간복잡도는 $O(NM \cdot NM \cdot N) = O(N^2M^2 \min(N, M))$ 이다.

부분문제 2

i 행 j 열의 격자를 편의상 (i, j) 로 표기하기로 한다.

(i, j) 에서 V자 색칠하기를 했을 때 색칠되는 격자의 수를 $C[i][j]$ 라 하면, (i, j) 가 흰색일 때 $C[i][j]$ 는 $((i, j)$ 부터 왼쪽 위 대각선으로 연속된 흰색 격자의 수) + $((i, j)$ 부터 오른쪽 위 대각선으로 연속된 흰색 격자의 수) - 1이 된다.

- $L[i][j]$: (i, j) 부터 왼쪽 위 대각선으로 연속된 흰색 격자의 수
- $R[i][j]$: (i, j) 부터 오른쪽 위 대각선으로 연속된 흰색 격자의 수

(i, j) 가 흰색인 경우 $L[i][j] = L[i-1][j-1] + 1$, 아닌 경우 $L[i][j] = 0$ 이며, $R[i][j]$ 도 유사하게 계산할 수 있다. 따라서, 모든 $L[i][j]$ 및 $R[i][j]$ 값을 계산할 수 있고, $C[i][j]$ 테이블 역시 $O(NM)$ 시간복잡도로 채울 수 있다.

첫 V자 색칠하기를 하는 각각의 경우에 대해, 첫 V자 색칠하기 후 상태에서 모든 $C[i][j]$ 를 $O(NM)$ 시간에 구해 놓으면 문제를 해결할 수 있다. 이 때 총 시간복잡도는 $O(N^2M^2)$ 이다.

부분문제 4

$i + j$ 가 짹수인 (i, j) 를 짹수 칸, $i + j$ 가 홀수인 (i, j) 를 홀수 칸이라 하자. V자 색칠하기를 하는 두 칸의 기우성에 따라 경우의 수를 나눌 수 있다. 먼저, 짹수 칸 하나와 홀수 칸 하나에서 했다면 두 V자 색칠하기는 서로에게 영향을 끼치지 않는다. 즉, 초기 상태에서 $C[i][j]$ 를 계산해 놓고, 짹수 칸의 최댓값과 홀수 칸의 최댓값의 합을 구하면 이 케이스에서 최적이다.

첫 번째 V자 색칠하기에서 색칠된 칸들의 집합을 S_1 두 번째 V자 색칠하기로 색칠된 칸들의 집합을 S_2 라 할 때, 다음 셋 중 하나가 성립한다:

1. 정수 k 가 존재하여 S_1 의 모든 격자 (i, j) 는 $i + j \leq k$, S_2 의 모든 격자 (i, j) 는 $i + j > k$ 가 성립한다. (또는 그 반대)
2. 정수 k 가 존재하여 S_1 의 모든 격자 (i, j) 는 $i - j \leq k$, S_2 의 모든 격자 (i, j) 는 $i - j > k$ 가 성립한다. (또는 그 반대)
3. 선택한 두 격자에 대해 둘 중 위쪽인 격자를 중심으로 대각선을 그어 격자판을 4개의 영역으로 나누면 다른 격자는 아래 영역에 포함되며, 이 때 두 V자 색칠은 서로 영향을 끼치지 않는다.

1번 또는 2번 케이스는 두 V자가 서로 $i + j = k$ 나 $i - j = k$ 직선 기준으로 나누어진 두 영역에 각각 포함됨을 뜻하며, 3번 케이스는 두 V자 중 하나는 위쪽, 하나는 아래쪽 V자가 되어 서로 독립적인 경우이다.

1, 2번 케이스는 다음에 정의되는 두 테이블 DL, DR 의 값을 계산하여 해결할 수 있다.

- $DL[i][j] : (i, j)$ 에서 출발하여 원쪽 아래 대각선으로 가다가 원쪽 위 대각선으로 가는 흰색 칸으로만 이루어진 경로의 최대 칸 수
- $DR[i][j] : (i, j)$ 에서 출발하여 오른쪽 아래 대각선으로 가다가 오른쪽 위 대각선으로 가는 흰색 칸으로만 이루어진 경로의 최대 칸 수

DL 테이블은 $DL[i][j] = \max(DL[i+1][j-1] + 1, L[i][j])$ 의 점화식으로 $O(NM)$ 시간에 계산할 수 있고, DR 도 동일한 방법으로 가능하다. 그리고 DL, DR, C 테이블을 이용하면 $i + j \leq k$ 인 칸들만 사용하는 V자의 최대 길이, $i + j > k$ 인 칸들만 사용하는 V자의 최대 길이 등을 빠르게 계산할 수 있다. 모든 k 에 대해 $i + j \leq k$ 인 칸에서의 V자, $i + j > k$ 에서의 V자, $i - j \leq k$ 인 칸에서의 V자, $i - j > k$ 에서의 V자의 최대 길이를 모두 구하면 충분하고, 따라서 이 케이스는 $O(NM)$ 시간에 처리 가능하다.

3번 케이스는 $M[i][j] = \max(C[i][j], M[i-1][j-1], M[i-1][j], M[i-1][j+1])$ 로 정의되는 DP 테이블 M 를 채우면 두 V자 중 아래가 (i, j) 를 선택한 것일 때 위쪽 V자는 $M[i-1][j]$ 칸만큼 색칠하는 것이 최적임을 알 수 있다(직사각형에서 prefix max를 사용한 것과 동일하다). 따라서, $O(NM)$ 시간에 처리 가능하다.

종합하여, 다이나믹 프로그래밍으로 $O(NM)$ 시간에 문제를 해결할 수 있다. 구현을 효율적으로 하지 못하여 $O(N^2M)$ 이나 $O(NM^2)$ 로 문제를 해결하는 코드를 작성하면 부분문제 3까지만 점수를 받을 수 있다.