코드포스 6주차 문제 풀이

문제 난이도

	문제 번호	문제 이름	난이도
A	27914	인터뷰	S2
B	22871	징검다리 건너기 (large)	S1
C	16568	엔비스카의 영혼	S1
D	2218	상자 안의 구슬	G3
E	27635	Konstrukcija 스페셜 저지	G2
F	9029	정육면체	G2
G	5042	나무 옮기기	P5
H	2906	폭발 사고	P2

A. 인터뷰

수학, 다이나믹 프로그래밍

A. 인터뷰

- 이 문제의 목표는 줄을 서 있는 학생들 중에서 은호와 같은 학년의 학생이 한 명도 없도록 연속한 학생들을 선택하는 방법의 수를 구하는 것입니다.

- dp[i] = dp[i - 1] + cnt
- cnt = 같은 학년의 학생이 없는 연속된 학생 수를 세기 위한 카운터
- 주욱 순회하다가 같은 학년이면 cnt = 0으로 초기화, 아니면 cnt++ 입니다.

- 시간복잡도 : $O(N + Q)$

B. 징검다리 건너기 (LARGE)

다이나믹 프로그래밍, 이분 탐색

B. 징검다리 건너기 (large)

- 이 문제는 왼쪽에서 오른쪽으로 돌을 건너갈 때 필요한 힘K의 최솟값을 찾는 것입니다.
- dp[i] = 가장 왼쪽 돌에서 시작하여 i번째 돌에 도달하기 위해 필요한 최소의 힘
- 0 -> j -> i (0 < j < i) 가능한 경로 중, (i - j) * (1 + abs(A[i] - A[j])) 의 최소값을 구해서 dp[i]에 저장합니다.
- 시간복잡도 : $O(N^2)$

C. 엔비스카의 영혼

다이나믹 프로그래밍, 그래프 이론, 그래프 탐색, 너비 우선 탐색

C. 엔비스카의 영혼

- 이 문제는 줄을 서 있는 사람들 앞에 한길이가 서기 위해 필요한 최소 시간을 계산하는 문제입니다.

- dp[i] = 한길이가 i번째 사람 앞에 서기 위해 필요한 최소 시간
- 1초 기다리는 경우 : dp[i - 1] + 1
- a명 새치기 하는 경우 : dp[i - a - 1] + 1
- b명 새치기 하는 경우 : dp[i - b - 1] + 1
- 이 세 가지 경우 중 최소를 dp[i]에 저장합니다

- 시간복잡도 : $O(N)$

D. 상자 안의 구슬

다이나믹 프로그래밍

D. 상자 안의 구슬

- ✓ 이 문제는 세 가지 작업을 해서 얻을 수 있는 최대 점수를 구하는 것입니다.

- ✓ dp[i][j] = 상자 A에서 처음 i개, 상자 B에서 처음 j개의 구슬을 사용했을 때 얻을 수 있는 최대 점수
- ✓ 1번 연산(A에서 빼기) : dp[i - 1][j]
- ✓ 2번 연산(B에서 빼기) : dp[i][j - 1]
- ✓ 3번 연산(A, B에서 빼고 점수 더하기) : dp[i - 1][j - 1] + score[boxA[i]][boxB[j]]
- ✓ 이 세 가지 중 최대 값을 dp[i][j]에 저장합니다.

- ✓ 시간 복잡도 : $O(AB)$

E. KONSTRUKCIJA 스페셜 저지

다이나믹 프로그래밍, 그래프 이론, 위상 정렬, 방향 비순환 그래프

E. Konstrukcija 스페셜 저지

- ✓ 원래 Konstrukcija 문제의 tns(1, N)값을 출력해야 합니다.
- ✓ DAG 이므로 위상정렬을 먼저 하고 그래프 탐색을 합니다.
- ✓ 초기값 : dp[1][0] = -1
- ✓ 순회하면서 dp[i][0] += 1 합니다
- ✓ dp[i][0] : 노드 i까지 진입하는 경로의 tension 값의 홀수 번째 항목
- ✓ dp[i][1] : 짝수 번째 항목
- ✓ 다음 노드의 dp[next][0] 값 = 현재 노드의 dp[cur][1]
- ✓ 다음 노드의 dp[next][1] 값 = 현재 노드의 dp[cur][0]

E. Konstrukcija 스페셜 저지

- ✓ next가 N이면 dp[next][0] = dp[cur][1], dp[next][1] = dp[cur][1]을 하지 않고 break합니다.
- ✓ 위와 같이 하는 이유와 dp[1][0] = -1인 이유는 1과 N을 제외한 모든 부분 수열의 홀수 길이, 짝수 길이를 찾아야 하기 때문입니다.
- ✓ dp[1][0] = -1 같이 설정하면 그래프 탐색할 때 dp[1][0]++ -> dp[1][0] = 0으로 되기 때문에 1이 제외됩니다.
- ✓ 탐색하면서 다음 노드로 자신의 정보를 넘겨줘야 하는데, 이 때 자식 중 N이 아닌 하나의 자식 노드로 자신의 정보를 넘겨줍니다.
- ✓ 시간 복잡도 : $O(N+M)$

F. 정육면체

다이나믹 프로그래밍

F. 정육면체

- ✓ 이 문제는 주어진 나무 조각을 정육면체 조각들로 자를 때 필요한 절단 횟수를 최소화하여 정육면체 조각들의 개수를 구하는 것입니다.

- ✓ dp[w][l][h] : 가로 길이가 w, 세로 길이가 l, 높이가 h인 나무 조각을 모든 조각이 정육면체가 되도록 자를 때 필요한 최소 절단 횟수
- ✓ w = l = h 인 경우 : dp[w][l][h] = 1
- ✓ 가로로 자르는 경우 (1 < cutPoint < w) :
  dp[cutPoint][l][h] + dp[w - cutPoint[l][h]
- ✓ 세로로 자르는 경우 (1 < cutPoint < l) :
  dp[w][cutPoint][h] + dp[w][l - cutPoint][h]
- ✓ 수평으로 자르는 경우 (1 < cutPoint < h) :
  dp[w][l][cutPoint] + dp[w][l][h - cutPoint]

F. 정육면체

- ✓ 위의 가능한 모든 가짓수 중에 가장 작은 값을 dp[w][l][h] 에 저장합니다.
- ✓ dp 초기화가 오래 걸리기 때문에 미리 200*200*200 dp배열을 T입력 전에 전처리하고 쿼리를 처리합니다.
- ✓ 시간복잡도 : $O(200^4)$

G. 나무 옮기기

다이나믹 프로그래밍, 정렬, 기하학

G. 나무 옮기기

- ✓ 이 문제는 도로의 왼쪽에 심어진 나무를 도로의 양쪽에 균등하게 나눠 심어야 할 때, 나무를 옮겨야 하는 최소 이동 거리를 구하는 문제입니다.
- ✓ dp[left][right] : 왼쪽에 left개의 나무를, 오른쪽에 right개의 나무를 심었을 때 옮겨야 하는 최소 이동 거리
- ✓ 나무를 놓아야 할 위치가 고정이기 때문에, 나무의 간격을 계산합니다.
- ✓ 나무의 위치를 기준으로 오름차순 정렬합니다.
- ✓ 나무를 처음부터 탐색하면서 이 나무를 left에 놓았을 때, right에 놓았을 때의 최소값을 dp[left][right] 배열에 저장합니다. (탑다운 방식)

G. 나무 옮기기

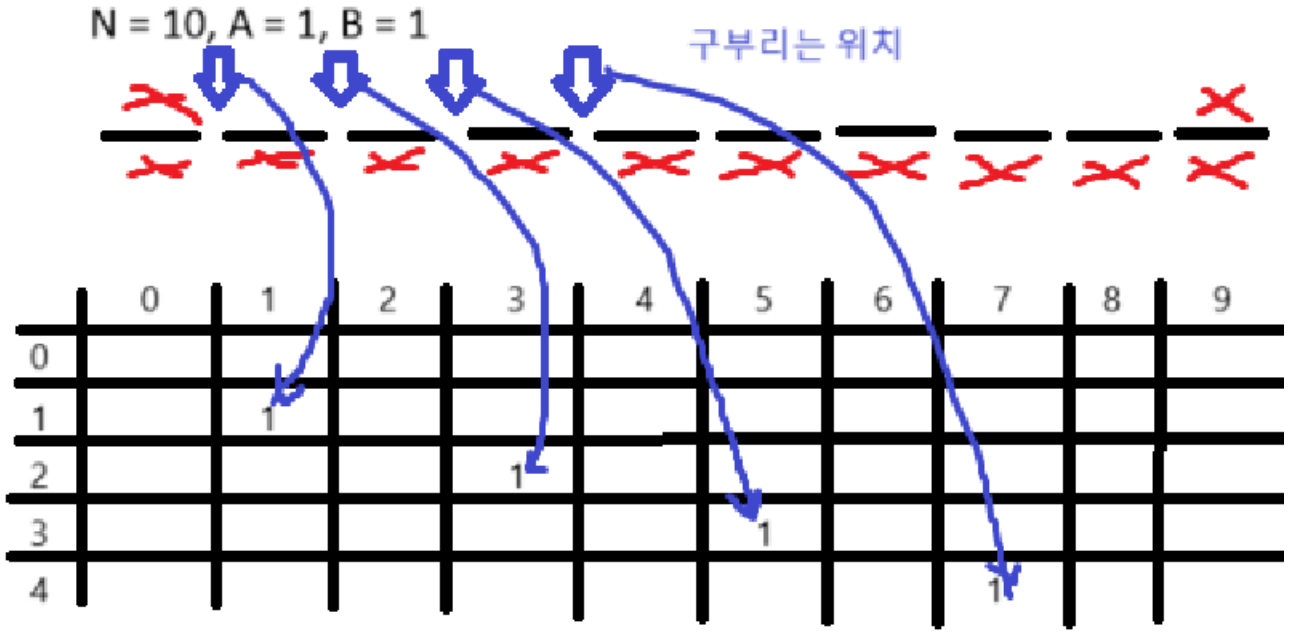
- ✓ left에 놓은 경우와 right에 놓은 경우 거리 계산에 유의해야 합니다.
- ✓ dp[0][0]을 출력하면 정답입니다.
- ✓ 시간복잡도 : $O(N^2)$

H. 폭발 사고

다이나믹 프로그래밍

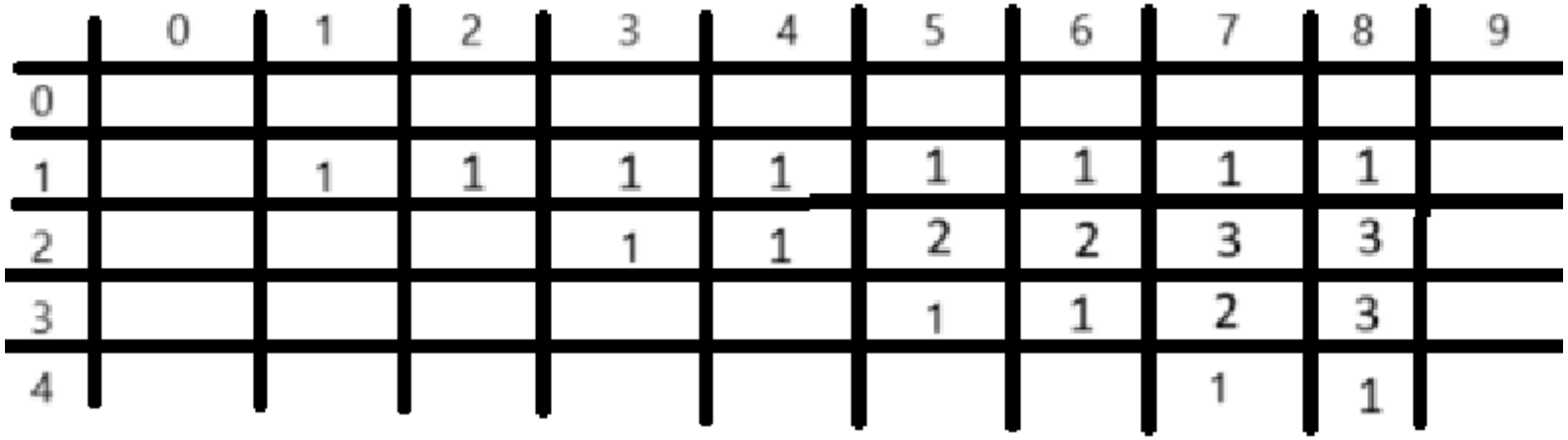
H. 폭발 사고

- ✓ 이 문제는 테이프를 180도로만 구부려서 테이프를 폭발시키지 않고 구부릴 수 있는 모든 경우를 세는 문제입니다.
- ✓ dp[i][j] = i번째 조각에서 왼쪽에 있는 조각의 길이가 j인 경우



- ✓ dp 순회 처음 부분입니다.

H. 폭발 사고

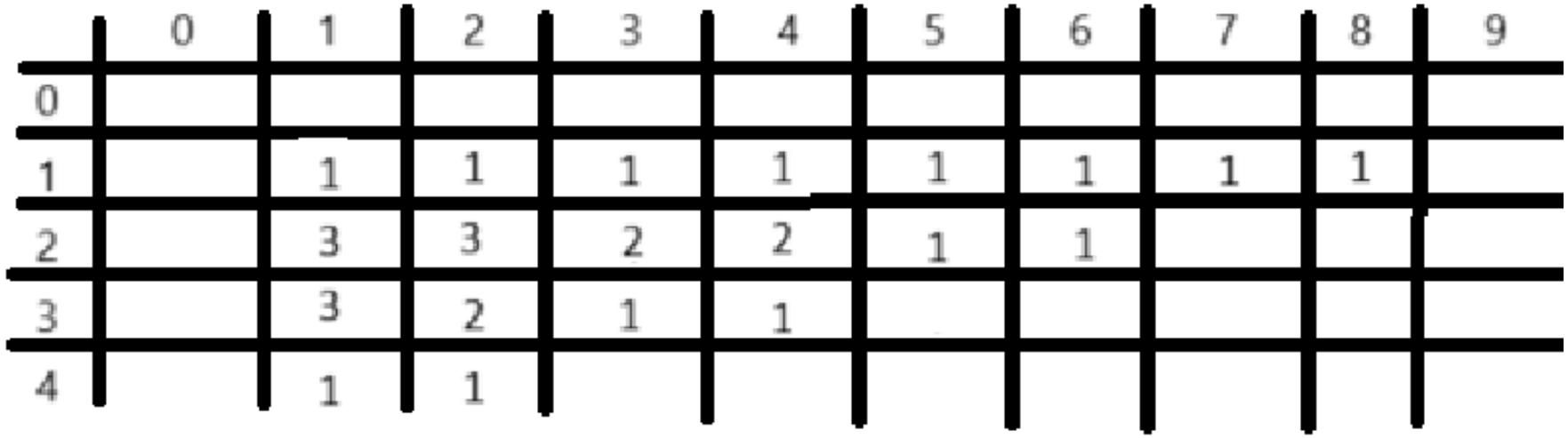


- ✓ 위의 dp 배열은 왼쪽에서 오른쪽으로 구부리는 모든 경우입니다.

- ✓ sum = { 0, 1, 2, 4, 6, 10, 14, 21, 29, 29 }
- ✓ dp를 계산한 것을 기반으로 누적합 배열을 만듭니다.

- ✓ sum[n] : dp[0] ~ dp[n] 을 모두 합한 것

H. 폭발 사고



- ✓ 누적합 배열을 만든 이유는 왼쪽에서 오른쪽으로 구부리는 경우에 왼쪽과 오른쪽을 동시에 구부리는 경우를 세기 위함입니다.

- ✓ left = { 0, 8, 7, 4, 4, 2, 2, 1, 1, 0 }

H. 폭발 사고

- ✓ 오른쪽에서 왼쪽으로 구부리는 경우의 수 : sum[N - 1]
- ✓ 왼쪽에서 오른쪽으로 구부리는 경우의 수 : left[0] ~ left[N - 1] 합
- ✓ left[i] * sum[i - 1] : 오른쪽과 왼쪽을 둘 다 구부리는 경우의 수
- ✓ 구부리지 않는 경우의 수 : 1

- ✓ 위 네 가지 경우를 전부 합하여 모듈러 연산하면 정답입니다.

- ✓ 값이 크기 때문에 중간중간 모듈러 연산을 해줘야 합니다.

- ✓ 시간 복잡도 : $O(N^2)$