2025 HICON
홍익대학교 프로그래밍 경진대회 풀이

Official Solutions

by

HICON

2025년 9월 27일

문제		의도한 난이도	출제자
A	와우산 스탬프 투어	**Easy**	pauljjang410
B	후문으로	**Easy**	hssong
C	체크 박스	**Medium**	xhdtlsid2
D	불안정한 탑	**Medium**	smjun04
E	실력과 열정	**Medium**	dosawas
F	와우 네트워크	**Advanced**	smjun04
G	도치균형의 배열	**Advanced**	dosawas
H	순회공연	**Hard**	xhdtlsid2
I	사과 농장	**Hard**	xhdtlsid2

A. 와우산 스탬프 투어

`implementation`

출제진 의도 – **Easy**

- ✓ 제출 49번, 정답 32명 (정답률 65.306%)
- ✓ 처음 푼 참가자: **왕밤빵왕밤빵왕밤빵왕밤빵왕밤빵**, 1분
- ✓ 출제자: `pauljjang410`

A. 와우산 스탬프 투어

- ✓ 지문에 제시된 과정을 그대로 구현하면 됩니다.
- ✓ N 은 3 이상인지, 5 와 같은지로 조건을 설정합니다.
- ✓ W 는 $1\,000$ 초과인지, 아닌지로 조건을 설정합니다.

B. 후문으로

`implementation`

출제진 의도 – **Easy**

- ✓ 제출 155번, 정답 26명 (정답률 16.774%)
- ✓ 처음 푼 참가자: **왕밤빵왕밤빵왕밤빵왕밤빵왕밤빵**, 6분
- ✓ 출제자: hssong

B. 후문으로

- ✓ 각 건물을 방문할 때마다 층수의 차이를 누적하여 해결할 수 있습니다.
- ✓ 헷갈리는 부분은 0층이 없다는 것입니다. 이는 각 층수를 zero-based로 변환하여 차이를 구해 누적한 후, 다시 한국식의 층수로 변환하여 출력하는 방식으로 해결할 수 있습니다.
- ✓ 구체적으로, 한국식 층수에서 지상층인 경우에만 1을 빼 생각하면, 0층부터 시작하는 층수 시스템이 됩니다.
- ✓ 유럽에서는 실제로 0층부터 시작하는 층수 표기를 볼 수 있다고 합니다. 신기하네요.

C. 체크 박스

`case_work, ad_hoc`

출제진 의도 – **Medium**

- ✓ 제출 106번, 정답 23명 (정답률 21.698%)
- ✓ 처음 푼 참가자: **akdlxm39**, 16분
- ✓ 출제자: `xhdtlsid2`

C. 체크 박스

- ✓ 전체 체크 조작을 두 번 이상 사용하는 건 무의미합니다.
- ✓ 모든 데이터 행의 체크 박스가 체크된 상태가 반복되기 때문입니다.
- ✓ 마찬가지로, 전체 해제 조작을 두 번 이상 사용하는 것도 무의미합니다.

- ✓ 또한, 전체 해제 이후 전체 체크를 하는 것도 무의미합니다.
- ✓ 전체 해제를 하려면 이미 전체가 체크된 상태였어야 하기 때문입니다.

C. 체크 박스

- 따라서 최적의 조작 방식은 다음 세 가지 경우 중 하나입니다.
- 1. 토글만 사용
- 2. 전체 체크 후 토글만 사용
- 3. 전체 체크 후 전체 해제 후 토글만 사용

- 각 경우에 대해서 필요한 조작 횟수를 구하고, 셋 중 최솟값을 구하면 됩니다.
- 2, 3번의 경우, 처음부터 전체가 체크된 상태라면 전체 체크 조작을 하지 않아도 된다는 점에 유의해야 합니다.

D. 불안정한 탑

`math, greedy`

출제진 의도 – **Medium**

- ✓ 제출 54번, 정답 23명 (정답률 44.444%)
- ✓ 처음 푼 참가자: **akdlxm39**, 22분
- ✓ 출제자: `smjun04`

D. 불안정한 탑

- 문제를 처음 접하면, 불안정한 탑을 찾아 높이를 낮추는 과정을 반복하는 그리디한 접근을 떠올리기 쉽습니다.
- 하지만 한 탑의 높이를 수정하면 인접한 탑들의 안정성 조건이 연쇄적으로 바뀌기 때문에, 이러한 접근은 매우 복잡해집니다.
- 이 문제의 핵심적인 관찰은, **최소 비용으로 모든 탑을 안정시킨 최종 상태는 모든 탑의 높이가 서로 같다**는 것입니다.

D. 불안정한 탑

- ✓ 모든 탑이 안정된 후의 높이를 $B_1, B_2, \ldots, B_N$ 이라고 합시다. 안정성 조건은 다음과 같습니다.
  - $B_1 \leq B_2$
  - $B_N \leq B_{N-1}$
  - $B_k \leq \dfrac{B_{k-1} + B_{k+1}}{2} \quad (2 \leq k \leq N-1)$
- ✓ 세 번째 조건을 변형하면 $2B_k \leq B_{k-1} + B_{k+1}$, 즉 $B_k - B_{k-1} \leq B_{k+1} - B_k$ 가 됩니다.
- ✓ 인접한 탑의 높이 차이를 $d_i = B_{i+1} - B_i$ 로 정의하면, 위 식은 $d_{k-1} \leq d_k$ 가 됩니다. 즉, 높이 차이로 이루어진 수열은 감소하지 않습니다.

D. 불안정한 탑

- 수열 $d_i = B_{i+1} - B_i$ 는 감소하지 않음을 보였습니다. ($d_1 \le d_2 \le \cdots \le d_{N-1}$)
- 한편, 첫 번째 안정성 조건($B_1 \le B_2$)에서 $d_1 = B_2 - B_1 \ge 0$ 입니다.
- 두 번째 안정성 조건($B_N \le B_{N-1}$)에서 $d_{N-1} = B_N - B_{N-1} \le 0$ 입니다.
- 이를 모두 종합하면 $0 \le d_1 \le d_2 \le \cdots \le d_{N-1} \le 0$ 이라는 부등식이 성립합니다.
- 위 부등식을 만족하는 유일한 경우는 모든 d_i 가 0인 경우입니다. 따라서 모든 i 에 대해 $B_{i+1} = B_i$ 이고, 모든 탑의 높이는 같아야 합니다.

D. 불안정한 탑

- ✓ 증명을 통해 최종적으로 모든 탑의 높이가 어떤 값 H 로 같아져야 함을 알았습니다.
- ✓ 문제 조건에 따라 탑의 높이는 낮출 수만 있으므로, 최종 높이 H 는 모든 초기 높이 A_i 보다 작거나 같아야 합니다.

$$H \leq \min(A_1, A_2, \ldots, A_N)$$

- ✓ 총비용은 $\sum_{i=1}^{N}(A_i - H) = (\sum_{i=1}^{N} A_i) - N \times H$ 입니다.
- ✓ 이 비용을 최소화하려면 H 를 최대로 만들어야 합니다. H 가 될 수 있는 최댓값은 $\min(A_1, A_2, \ldots, A_N)$ 입니다.

D. 불안정한 탑

- ✓ 따라서 모든 탑을 초기 높이의 최솟값으로 맞추는 것이 최소 비용을 달성하는 방법입니다.
- ✓ $O(N)$ 시간 복잡도로 모든 탑의 높이 합과 최솟값을 구한 뒤, $(\sum A_i) - N \times \min(A_i)$ 를 계산하면 정답입니다.
- ✓ N 과 A_i 의 최댓값을 고려하면 합이 32비트 정수형의 범위를 초과할 수 있으므로, `long long` 과 같은 64비트 정수 자료형을 사용해야 합니다.

E. 실력과 열정

`dp, greedy`

출제진 의도 – **Easy**

- ✓ 제출 51번, 정답 1명 (정답률 1.961%)
- ✓ 처음 푼 참가자: **nadori**, 51분
- ✓ 출제자: dosawas

E. 실력과 열정

- 최종 점수를 최대화 하기 위해 dp table을 다음과 같이 만들어봅시다.
- $dp[i][a][b]$: i 일차에 실력이 a, 열정이 b 일 때, 최대 누적 점수
- 실력과 열정의 합을 일정하므로 dp 식을 더 간단하게 2차원 배열로 나타내봅시다.
- $dp[i][a]$: i 일차에 실력이 a 일 때, 최대 누적 점수
- $S = A_0 + B_0$ 라고 하면 실력이 a 로 결정되었을 때, 열정은 자동으로 $S - a$ 로 구할 수 있으므로 dp 식을 2차원으로 줄일 수 있습니다.

E. 실력과 열정

- ✓ i 일차에 실력이 a 인 상태가 되기 위해서는 $i-1$ 일차의 어떤 상태에서 전이가 되었을 것입니다.
- ✓ 1. 공부 : 실력이 늘어남.
  - $dp[i][a] = \max(dp[i-1][a-x]) + a * (S-a)$
  - x 는 $K \leq x \leq a$
- ✓ 2. 휴식 : 실력이 줄어듦.
  - $dp[i][a] = \max(dp[i-1][a+x]) + a * (S-a)$
  - x 는 $K \leq y \leq S-a$

위와 같은 점화식을 $i = N$ 일차까지 진행하고 $dp[N]$ 을 순회하면 점수의 최댓값을 찾을 수 있습니다. 시간복잡도는 $O(NS^2)$ 입니다.

- ✓ 별해)
- ✓ dp식을 1차원배열로 해결하고 시간복잡도를 $O(NS)$ 로 줄일 수 있습니다.
- ✓ greedy + bruteforce로 해결할 수도 있습니다.

F. 와우 네트워크

`disjoint_set`, `sweeping`

출제진 의도 – **Advanced**

- ✓ 제출 39번, 정답 12명 (정답률 33.333%)
- ✓ 처음 푼 참가자: **왕밤빵왕밤빵왕밤빵왕밤빵왕밤빵**, 41분
- ✓ 출제자: `smjun04`

- ✓ 1초부터 T 초까지 매초의 연결 요소(Connected Components) 개수를 모두 더해야 합니다.
- ✓ T 가 최대 10^9 으로 매우 크기 때문에, 모든 시각을 하나씩 시뮬레이션하는 것은 불가능합니다.
- ✓ 네트워크의 연결 상태는 **새로운 라우터가 켜지는 시점**에만 변합니다.
- ✓ 따라서 이벤트가 발생하는 시점들 사이의 긴 구간 동안에는 연결 요소의 개수가 일정하게 유지됩니다.



- ✓ 이 관찰을 바탕으로, 시간의 흐름에 따라 이벤트를 처리하는 **스위핑(Sweeping)** 기법을 사용합니다.
- ✓ 먼저, M 개의 라우터 정보를 (`s, u, v`) 형태의 이벤트로 보고, 시작 시간 ‘s‘ 기준으로 오름차순 정렬합니다.
- ✓ 정렬된 이벤트를 순서대로 처리하며, 각 이벤트 사이의 구간별 점수를 계산하여 누적합니다.
- ✓ 이 과정에서 연결 요소의 개수를 효율적으로 관리하기 위해 **Disjoint Set Union (DSU)** 자료구조를 사용합니다.

F. 와우 네트워크

- ✓ 현재 연결 요소의 개수를 `components` (초기값 N), 마지막 이벤트 시각을 `last_time` (초기값 1)으로 둡니다.

- ✓ 정렬된 각 이벤트 (`s, u, v`)에 대해 다음을 반복합니다.
- ✓ **이전 구간 점수 합산**
  - `[last_time, s-1]` 구간의 점수를 더합니다.
- ✓ **네트워크 상태 갱신 (at time s)**
  - `union(u, v)`를 호출합니다.
  - 만약 두 정점이 다른 집합이었다면, `components`를 1 감소시킵니다.
- ✓ **시간 갱신**
  - `last_time = s;`

- ✓ M 개의 이벤트를 정렬하는 데 $O(M \log M)$ 이 소요됩니다.
- ✓ DSU 연산은 경로 압축 최적화 시 거의 상수 시간이므로, 총 시간 복잡도는 $O(M \log M)$ 입니다.
- ✓ 답이 32비트 정수형의 범위를 초과할 수 있으므로, `long long`과 같은 64비트 정수 자료형을 사용해야 합니다.

G. 도치균형의 배열

`ad_hoc, constructive`

출제진 의도 – **Advanced**

- ✓ 제출 14번, 정답 6명 (정답률 42.857%)
- ✓ 처음 푼 참가자: **왕밤빵왕밤빵왕밤빵왕밤빵왕밤빵**, 66분
- ✓ 출제자: dosawas

- ✓ 다양한 풀이가 있는데 그 중 두 가지를 살펴봅시다.
- ✓ sol 1) 어떤 배열 A 의 $\sum D$ 는 그 배열을 뒤집은 배열 A' 의 $\sum U$ 와 같습니다.
- ✓ 따라서 앞에서 읽으나 뒤에서 읽으나 같은 팰린드롬 ($A = A'$) 이면 도치균형의 배열이 됩니다.
- ✓ $1 \cdots M\ M \cdots 1$ 와 같이 배열을 구성하면 됩니다.

G. 도치균형의 배열

- ✓ sol 2) 먼저 1 부터 M 까지의 수를 모두 쓰기 위해 나열해봅시다. $(1 \cdots M)$
- ✓ $\sum U = M(M-1)/2, \sum D = 0$ 가 됩니다.
- ✓ 그리고 이 상황에서 맨 앞에 M 을 추가하면 $\sum D$ 값이 $M-1$ 만큼 늘어납니다.
  - M이 짝수인 경우)
    - M 을 맨 앞에 $M/2$ 개 두면 $\sum D = (M/2) * (M-1) = \sum U$ 입니다.
  - M이 홀수인 경우)
    - 크기가 $2M$ 이하인 배열을 만들면 되고 배열의 원소는 10억이하의 수를 넣으면 되니 짝수로 바꿔서 풀면 됩니다.
    - 맨 뒤에 M+1을 추가하고 짝수인 경우처럼 해결합니다. 이 경우 $M+1+(M+1)/2 \leq 2M$ 이어야하고 부등식을 풀면 $M \geq 3$ 인 홀수에서 모두 답을 구할 수 있습니다. ($M=1$ 은 예외처리 하면 됩니다.)

H. 순회공연

`functional_graph`, `binary_search`, `sparse_table`

출제진 의도 – **Hard**

- ✓ 제출 11번, 정답 1명 (정답률 9.091%)
- ✓ 처음 푼 참가자: **nadori**, 92분
- ✓ 출제자: `xhdtlsid2`

- ✓ 각 도시와 도로를 그래프로 모델링하면 함수형 그래프가 됩니다.
- ✓ $N-1$ 일차에는 모든 가수가 사이클 안에 들어와 있게 됩니다.
- ✓ 따라서, 밤샘 공연이 일어난다면 늦어도 $N-1$ 일차에는 일어남을 알 수 있습니다.

- ✓ 1 일차부터 $N-1$ 일차까지 공연을 시뮬레이션할 경우 $O(N^2)$ 이 걸려 시간 제한 내에 풀 수 없습니다.

- ✓ 어떤 도시에서 가수들이 한번 모이면 그 이후로도 계속 함께 이동하게 됩니다.
- ✓ 즉, 밤샘 공연이 한번 일어나면 그 이후로도 계속 일어나게 됩니다.
- ✓ 따라서, 첫 밤샘 공연이 언제 일어나는지 이분 탐색을 이용하여 구할 수 있습니다.

H. 순회공연

- ✓ 이분 탐색을 하려면 x 일차에 밤샘 공연이 일어나는지 알 수 있어야 합니다.
- ✓ 희소 배열을 이용하면 어떤 가수가 x 일차에 어느 도시에 있는지 $O(\log N)$ 에 알 수 있습니다.
- ✓ 가수마다 x 일차 위치를 계산해서 $O(N \log N)$ 에 밤샘 공연 여부를 판별할 수 있습니다.
- ✓ 이분 탐색 범위가 $O(N)$ 이므로 최종 시간 복잡도는 $O(N \log^2 N)$ 입니다.

I. 사과 농장

`sweeping, bitset, prefix_sum`

출제진 의도 – **Hard**

- ✓ 제출 5번, 정답 0명 (정답률 0.000%)
- ✓ 처음 푼 참가자: -, -분
- ✓ 출제자: `xhdtlsid2`

- ✓ 농장 내 M 개의 열 중 하나를 골라 $y = 0$ 부터 $y = N$ 까지 이동한다고 생각해 봅시다.
- ✓ 어떤 영역의 가로변을 만나면 해당 영역 안으로 들어온 상태가 됩니다.
- ✓ 같은 영역의 가로변을 다시 만나면 해당 영역을 벗어난 상태가 됩니다.
- ✓ 즉, 현재 위치한 칸이 어떤 영역들에 속하는지 추적할 수 있습니다.

- ✓ 현재 칸이 어떤 영역들에 속하는지 알 때, 각 영역을 정한 사람들이 모두 서로 지인 관계인지 확인해야 합니다.
- ✓ 각 영역을 정한 사람들의 집합을 S라 합시다.
- ✓ S의 임의의 원소 u에 대해, u와 u의 모든 지인들의 집합을 T라 합시다.
- ✓ 각 영역을 정한 사람들이 모두 서로 지인 관계일 필요충분조건은 $S \subseteq T$ 입니다.

- ✓ $S \subseteq T$ 인지 확인하기 위해 비트 집합을 활용할 수 있습니다.
- ✓ 현재 칸이 어떤 영역들에 속하는지 추적하기 위해 K 개의 비트로 이루어진 비트 집합을 관리합시다.
- ✓ $i(\geq 0)$ 번째 비트는 현재 $i+1$ 번째 영역에 속해 있으면 1, 그렇지 않으면 0 으로 설정합니다.
- ✓ 또한, 각 사람별로 자신과 자신의 모든 지인들을 표시한 비트 집합을 미리 만들어 놓을 수 있습니다.
- ✓ $i(\geq 0)$ 번째 비트는 $i+1$ 번째 사람이 자기 자신이거나 지인인 경우 1, 그 외의 경우 0 으로 설정합니다.

- ✓ S를 나타내는 비트 집합을 S_b, T를 나타내는 비트 집합을 T_b라 합시다.
- ✓ $S \subseteq T$ 인지는 T_b와 S_b를 bitwise AND 한 결과가 S_b와 같은지 확인하여 알 수 있습니다.

- ✓ $S \subseteq T$ 인 경우, 현재 칸의 사과의 개수를 $|S|$로 나눠줍니다. (소수점 이하 절사)
- ✓ $S \not\subseteq T$ 인 경우, 현재 칸의 사과의 개수를 0으로 설정합니다.
- ✓ 농장 내의 모든 칸에 대해서 위와 같이 사과의 개수를 갱신합니다.

- 각 사람이 수확하게 되는 사과의 개수를 구해 봅시다.
- 어떤 영역에 대해, 농장 내 어떤 열을 지나는 해당 영역의 가로변들의 y 좌표를 오름차순으로 정렬한 것을 $y_1, y_2, \cdots, y_{2c}$ 라 합시다.
- 해당 열에서 수확하게 되는 사과의 개수는 $\sum_{i=1}^{c}$ (y_{2i} 와 y_{2i-1} 사이 칸들의 사과 개수의 합) 과 같고, 이는 앞에서 갱신한 A 배열의 누적 합을 이용하여 구할 수 있습니다.
- 각 열마다 계산한 값을 전부 더하여 해당 사람이 수확하게 되는 사과의 총 개수를 구할 수 있습니다.

- ✓ A 배열을 갱신할 때 각 칸마다 bitwise 연산을 수행할 경우 $O(NMK/w)$에 계산할 수 있습니다.
- ✓ 각 영역마다 가로변의 y좌표를 정렬하는 데 총 $O(K(M+N)\log(M+N))$이 소요됩니다.
- ✓ 따라서 최종 시간 복잡도 $O(NMK/w + K(M+N)\log(M+N))$에 문제를 해결할 수 있습니다.