제2회 보라매컵 예선 풀이

2023년 9월 10일

문제		의도한 난이도	출제자
A	포스터 만들기	**Easy**	yooyou7
B	시간 외 근무 멈춰!!	**Medium**	99asdfg
C	기지방호	**Medium**	junhyung9985
D	부대 창설 행사	**Hard**	yooyou7
E	감시 초소	**Hard**	99asdfg
F	합동 훈련	**Challenging**	99asdfg

A. 포스터 만들기

`implementation, constructive`

출제진 의도 – **Easy**

- 제출 78번, 정답 25명 (정답률 35.897%)
- 처음 푼 사람: **xiaowuc1**, 5분
- 출제자: yooyou7

Subtask 1

- 가능한 입력은 예제 입력 1에 이미 나와 있습니다.
- 예제 출력 1을 그대로 출력해도 되고, 다른 마크를 만들어서 출력해도 됩니다.

Subtask 2

- $Y = 3, X = 5$가 고정되므로, 가능한 입력은 모두 8개입니다.
- X가 나올 수 있는 칸은 테두리를 제외하면 3칸입니다.
- 해당 칸에 WYW를 넣는 경우에만 유효한 마크를 만들 수 있습니다.
- 따라서, 입력되는 X가 3개인 경우에만 YES + 위의 마크를 출력하면 됩니다.

Subtask 3 + Subtask 4

- 테두리를 제외한 칸은 원하는 색으로 자유롭게 색칠할 수 있습니다.
- 그러나 Y 와 X 가 유동적이기 때문에, 올바른 마크를 크기에 무관하게 만들기 위해선 몇 가지 관찰이 필요합니다.

Subtask 3 + Subtask 4

- **관찰 1.** 주어진 용지의 왼쪽 절반을 칠하면, 오른쪽 절반은 자동으로 결정됩니다.
- 여기서 왼쪽 절반은 X가 짝수라면 왼쪽 $X/2$줄, 짝수라면 $(X+1)/2$줄입니다.
- 마크는 좌우대칭이어야 하므로, 절반만 색깔을 결정해 주면 나머지 절반은 이를 대칭해 주는 것으로 색깔을 정할 수 있습니다.

Subtask 3 + Subtask 4

- 이 중 왼쪽 절반의 맨 오른쪽 줄을 **중앙 구역**으로 지정합시다.
- **관찰 2.** 노란색 문양은 최소 1칸이 **중앙 구역**에 칠해져야 합니다.
- **관찰 3.** 하얀색 문양은 단 1칸도 **중앙 구역**에 칠해져서는 안됩니다.
- 이는 문양을 대칭시켰을 때 문양의 개수가 **중앙 구역**의 유무에 연관이 있기 때문입니다.
- 특히, 하얀색 문양이 **중앙 구역**에 칠해지는 경우, 설령 문양의 개수가 2개더라도 두 문양이 중앙을 기준으로 대칭을 이룰 수 없게 됩니다.
- 이를 토대로 **중앙 구역** 전체를 노란색, 그 옆줄을 하얀색으로 칠하는 마크 또한 유효한 마크이며, **Subtask 3 + 4** 의 조건을 만족합니다.

Subtask 5

- 파란색으로 색칠이 고정되는 칸이 생기므로, 파란색을 제외한 색의 색칠을 최소화하는 것이 이상적입니다.
- 노란색 문양은 **중앙 구역**에 최소 1칸이 칠해져야 합니다. 이 1칸을 먼저 칠했다고 생각해 봅시다.
- 하얀색 문양은 노란색 문양에 인접하게 칠해지면서 **중앙 구역**을 피해야 합니다. 해당 칸은 각 노란색 1칸에 대해 유일하게 결정됩니다.
- 파란색을 제외한 각 색깔을 1칸씩만 색칠했으므로, 해당 방식으로 문양을 확정하고 나머지를 파란색으로 칠하는 것이 이외 색을 최소화하는 방법입니다.

Subtask 5

- 해당 방법으로 색칠하기 위해선, **중앙 구역**을 포함한 가운데 3 4개의 세로줄 중 최소 1개의 가로줄이 XXX 형태여야 합니다.
- 해당 3 4칸이 확보된다면 그 칸을 WYW 또는 WYYW로 칠하고, 나머지를 B로 칠하는 마크가 유효합니다.
- 그런 가로줄이 하나도 없다면, 마크를 만드는 것이 불가능하므로 NO를 출력합니다.
- 마크의 가능성 확인은 $\mathcal{O}(Y)$에 준하게 처리할 수 있지만, 입출력의 시간 복잡도인 $\mathcal{O}(YX)$에 밀립니다.

B. 시간 외 근무 멈춰!!

`greedy, sorting, simulation`
출제진 의도 – **Medium**

- 제출 40번, 정답 3명 (정답률 7.500%)
- 처음 푼 사람: **kes0716**, 62분
- 출제자: 99asdfg

Subtask 1

- $N = 1, A, B, M = 0$ 이므로, 주어진 하나의 작업을 최소 시간 외 근무만을 사용하여 진행하면 됩니다.
- 단순히 사칙연산 식을 세우면 $\mathcal{O}(1)$ 로 해결할 수 있습니다.
- 그 외에도, 해당 작업에 대한 시뮬레이션을 직접 돌려보며 $\mathcal{O}(\max(d_i))$ 로 문제를 해결할 수도 있습니다.

Subtask 2

- 마감 기한이 더 늦는 작업을 미리 진행하느라 마감 기한이 더 빠른 작업을 진행하지 못하는 일이 발생하면 안 되므로, 마감 기한이 빠른 작업부터 차례로 진행하는 것이 항상 최적입니다.
- 또한 $A, B, M = 0$ 이므로, 평일 시간 외 근무와 주말 시간 외 근무 횟수에 제한이 없습니다.
- 각 작업을 마감 기한의 오름차순으로 정렬해 둔 뒤, 1 일부터 $\max(d_i)$ 일까지 평시 근무와 평일 시간 외 근무만으로 모든 작업을 완료할 수 있는지 시뮬레이션해 봅시다.
- 만약 평시 근무와 평일 시간 외 근무만으로 작업을 완료할 수 없다면, 주말 시간 외 근무를 진행하는 시뮬레이션을 다시 돌리며 최소 주말 시간 외 근무 시간을 찾읍시다.
- 총 시간복잡도는 $\mathcal{O}\left(N \log N + \max(d_i)\right)$ 입니다.

Subtask 3

- 이제 $0 \leq A = B$ 이므로, 진행할 수 있는 시간 외 근무 횟수에 제한이 발생합니다.
- 만약 $A = B = 0$ 을 만족한다면, **Subtask 2**와 동일하게 문제를 해결할 수 있습니다.
- 만약 $A > 0$ 이고 M 이 A 로 나누어떨어지지 않는다면, 가점을 정확히 M 점을 받는 것은 불가능합니다.
- 그 외의 경우에는, 총 M/A 회의 시간 외 근무를 평일과 주말에 나누어서 진행하며 주말 시간 외 근무의 최솟값을 찾아야 합니다.

Subtask 3

- **Subtask 2**와 유사하게 1 일부터 차례대로 시뮬레이션을 진행하되, 평일 시간 외 근무를 최대한 넣어준 뒤, 더 넣어야 하는 주말 시간 외 근무 시간을 채워 넣어 줍시다.
- 평일 및 주말 시간 외 근무를 M/A 회 정확히 채웠음에도 작업이 모두 종료되지 않았다면, 평시 근무로 남은 작업들을 채워주는 시뮬레이션을 돌립시다.
- 총 시간복잡도는 **Subtask 2**와 마찬가지로 $\mathcal{O}\left(N \log N + \max(d_i)\right)$ 입니다.

Subtask 4

- $M \leq 100$이므로, 모든 (A, B) 쌍에 대해 $Ax + By = M$을 만족하는 모든 (x, y) 쌍을 찾아줍시다.
- 이때, A, B, M이 0인지 아닌지에 따라 x값 및 y값이 무한히 커질 수 있다는 점에 유의합시다.
- $A \leq B$라는 조건 덕분에 $y = 0$이며 $x \neq 0$인 상황은 발생하지 않습니다.
- 따라서, 무한이 아닌 평일 시간 외 근무 $\rightarrow$ 무한이 아닌 주말 시간 외 근무 $\rightarrow$ 무한인 평일 시간 외 근무 $\rightarrow$ 남은 작업에 대한 평시 근무 $\rightarrow$ 무한인 주말 시간 외 근무 순서로 시뮬레이션을 진행해 주면 **Subtask 3**과 유사한 방식으로 문제를 해결할 수 있습니다.
- 총 시간복잡도는 $\mathcal{O}\left(N \log N + M \max(d_i)\right)$입니다.

C. 기지방호

graphs, dp

출제진 의도 – **Medium**

- 제출 23, 정답 11명 (정답률 47.826%)
- 처음 푼 사람: **snrnsidy**, 17분
- 출제자: junhyung9985

Subtask 1

- $N = 1, M = 1$ 이므로 현재 시연하고 있는 진법과 다음에 시연할 진법이 다르면 피로도가 1 아니면 0 씩 누적됩니다.
- 또한, $T = 1$ 이라서 매일 시작하는 진법을 시연한 후에 바로 마지막으로 시연할 진법을 시연해야합니다.
- 그러므로, 날마다 시작하는 진법과 마지막으로 시연할 진법이 다르면 피로도를 1 씩 더해서 출력하면 됩니다.
- 총 시간복잡도는 $\mathcal{O}(L)$ 입니다.

Subtask 2

- **Subtask 1**와 다른 점은 N과 M의 제한이 풀렸습니다.
- 따라서, 현재 시연하고 있는 진법과 다음에 시연할 진법간에 누적되는 피로도를 매번 계산해서 더해줘야 합니다.
- 총 시간복잡도는 $\mathcal{O}(NML)$입니다.

Subtask 3

- **Subtask 2**와 다른 점은 L의 제한이 상당히 커졌습니다.
- 그래서 날마다 누적되는 피로도를 따로 계산하고 계속 더해주면 시간초과가 납니다.
- 그래도 진법들의 개수가 매우 작으므로, 모든 $(C[i], C[j])(1 \le i, j \le F)$ 쌍에 대해서 $C[i]$를 시연하고 다음에 $C[j]$를 시연할 시에 드는 피로도를 미리 계산하여 그래프 형태로 저장합시다. 이는 $\mathcal{O}\left(NMF^2\right)$ 내로 가능합니다.
- 이후 날마다 시작하는 전술과 마지막으로 시연할 전술간의 피로도를 그래프에서 찾아 더하면 됩니다.
- 총 시간복잡도는 $\mathcal{O}\left(NMF^2 + L\right)$입니다.

Subtask 4

- **Subtask 3**에서의 모든 진법들의 쌍에 대해서 누적되는 피로도를 미리 계산하는 것까지는 동일합니다.
- 다만, T의 범위가 커짐에 따라서, 날마다 시작하는 진법과 마지막으로 시연할 진법들의 사이에 $T-1$개의 진법들을 시연해서 누적되는 피로도를 줄일 수 있습니다.
- 중간에 다른 진법들을 시연하여 누적되는 피로도를 줄일 수 있는 경우가 존재한다는 것은 다음과 같은 예시로 쉽게 확인할 수 있습니다.

Subtask 4

- 예를 들어, $C[A]$ 진법을 시연하고 다음에 $C[B]$ 진법을 시연해야한다고 합시다.
- $C[A]_{ij}$ 와 $C[B]_{ij}$ 가 다른 모든 (i, j) 쌍의 개수가 $a+b$개라고 하면 누적되는 피로도는 $(a+b)^2 = a^2 + 2ab + b^2$ 입니다.
- 또한, $C[X]_{ij}$ 와 $C[A]_{ij}$ 가 다른 모든 (i, j) 쌍의 개수가 a개, $C[X]_{ij}$ 와 $C[B]_{ij}$ 가 다른 모든 (i, j) 쌍의 개수가 b개인 $C[X]$ 라는 진법이 존재한다고 합시다.
- 그렇다면 $C[A]$ 를 시연한 후 $C[B]$ 를 바로 시연하지 않고, $C[A] \to C[X] \to C[B]$ 순서대로 시연을 할 수 있다면, 누적되는 피로도를 $a^2 + b^2$ 로 줄일 수가 있습니다.

Subtask 4

- 따라서, 이 문제는 날마다 시작되는 진법과 마지막으로 시연할 진법 사이에 최대 $T-1$ 개의 진법을 넣어 누적되는 피로도를 줄이는 문제로 치환이 됩니다.
- 이를 구하는 방법은 $C[A]$ 가 시작되는 진법이고 $C[B]$ 를 마지막으로 시연하는 진법이라 할때, 중간에 t 개의 진법을 거쳐서 누적되는 피로도를 최소로 만드는 순서가 $C[A] \to \cdots \to C[X] \to C[B]$ 라고 한다면
- $C[A] \to \cdots \to C[X]$ 는 $C[A]$ 를 시연하고 $C[X]$ 를 시연해야 할 때, 중간에 $t-1$ 개의 진법을 거쳐서 누적되는 피로도를 최소로 만드는 순서라는 것을 활용해서 DP 식을 세우면 됩니다.

C. 기지방호

Subtask 4

- $dp[i][j][turn]$ 을 $C[i]$ 를 시연한 후, $C[j]$ 를 시연하기 전에 최대 $turn-1$ 개의 진법들을 시연하여 누적되는 최소 피로도라고 합시다.
- $dp[i][j][1]$ 은 **Subtask 3**에서 미리 구한 $C[i]$ 를 시연한 후 $C[j]$ 를 시연할 때 누적될 피로도 값으로 초기화 됩니다.
- 또한, $dp[i][j][turn] = min(dp[i][j][turn-1], dp[i][prev][turn-1] + dp[prev][j][1])$ 이러한 점화식을 세워 DP 테이블을 채울 수 있습니다.
- 이를 계산한 후에 날마다 시작되는 진법과 마지막으로 시연할 진법에 대해서 DP 테이블에서 최소 피로도를 구하여 더하면 됩니다.
- 총 시간복잡도는 $\mathcal{O}\left(NMF^2 + TF^3 + L\right)$ 입니다.

Subtask 5

- **Subtask 4**의 풀이와 동일하지만, 그냥 T 번을 모두 본다면 시간초과나 메모리초과가 날 것입니다.
- 그런데, T 번을 모두 봐야할까요?
- 모든 진법의 쌍에 대해서 누적되는 피로도를 중간에 얼마든지 다른 몇 개의 진법들을 시연하여 누적되는 피로도를 최소로 줄이려면 몇 번까지만 보면 될까요?
- 정답은 F 번만 보면 됩니다. F 번 이상을 보는 순간부터는 모든 진법의 쌍에 대해서 누적되는 피로도가 최소로 줄인 값이 됩니다.

Subtask 5

- 이러한 이유로는 모든 진법의 쌍에 대해서 누적되는 피로도가 음이 아닌 정숫값이라는 것에 있습니다. 즉, 만약에 두 진법 간에 드는 피로도를 $F-2$개의 진법들을 거쳐서 최소로 만들었을 때, 다른 진법들을 추가해도 최대 0만큼만 줄일 수 있습니다.
- 그래서 $F \leq T$를 만족한다면, F번까지만 보면 됩니다. 그러면 DP 시간복잡도는 $\mathcal{O}\left(min(TF^3, F^4)\right)$으로 줄일 수가 있습니다.
- 이렇게만 최적화를 했을 때는 총 시간복잡도는 $\mathcal{O}\left(NMF^2 + min(TF^3, F^4) + L\right)$입니다.

Subtask 5

- 앞의 DP식 최적화까지만 해도 시간 내로 충분히 풀리지만 여기서 시간복잡도를 좀 더 줄일 수 있습니다.
- 만약에 $F \le T$를 만족할 때는 모든 진법의 쌍에 대해서 드는 피로도를 최소로 줄인 값이 되므로, 그냥 플로이드-워셜을 활용해도 됩니다. 그렇게 되면 $\mathcal{O}\left(F^3\right)$까지 줄일 수가 있습니다.
- 그래서 플로이드-워셜까지 활용하여 총 시간복잡도를 $F \le T$를 만족하면 $\mathcal{O}\left(NMF^2 + F^3 + L\right)$, 아니면 $\mathcal{O}\left(NMF^2 + TF^3 + L\right)$가 되도록 더 줄일 수가 있습니다.

D. 부대 창설 행사

ad hoc, constructive, greedy, data structures
출제진 의도 – **Hard**

- 제출 6번, 정답 5명 (정답률 83.333%)
- 처음 푼 사람: **xiaowuc1**, 34분
- 출제자: yooyou7

Subtask 1 + Subtask 2

- 문제 지문에 나온 대로 N 개의 무대에 대해 순서를 고정해 봅시다.
- 무대마다 P 명의 학생을 확인하면서, 이번 무대에 들어가는지 및 최소 인원을 달성하는지 확인하면 됩니다.
- 가능한 무대의 경우의 수는 $N!$ 이고, $8! = 40\,320$ 이므로 P 를 곱하면 시간 안에 모든 경우의 수를 탐색할 수 있습니다.
- 총 시간복잡도는 $\mathcal{O}(N!NP)$ 이며, **Subtask 1**은 이보다 여유롭게 $\mathcal{O}\left(N^{N}NP\right)$ 등으로도 통과할 수 있도록 설정되어 있습니다.

Subtask 3

- N 의 크기가 200으로, 모든 경우의 수를 탐색하는 것은 불가능합니다.
- 다만, 여러 경우의 수를 탐색하지 않고도 $\mathcal{O}(NP)$ 안에 탐색하는 풀이가 존재합니다.

Subtask 3

- 마지막 무대 X 를 제외한 모든 무대의 순서가 결정되었다고 가정해 봅시다.
- 이때, X 가 성공적으로 진행되기 위해선 X 만 희망한 병사의 수가 최소 인원을 만족해야 합니다.
- 이는 다른 무대를 같이 희망한 병사의 경우 앞 순서에서 이미 연습으로 빠졌을 것이기 때문입니다.
- 이를 토대로 첫 상태에서 희망 무대가 1개인 병사만 모았을 때, 최소 인원을 만족하는 무대를 미리 정리할 수 있습니다.
- 가능한 후보가 0개인 경우 / 1개인 경우 / 2개 이상인 경우로 나눠서 진행해 봅시다.

Subtask 3

- **Case 1. X 의 후보가 0개**
- 이는 앞의 모든 무대를 어떤 식으로 배열하더라도, 마지막 무대에서 무대가 실패한다는 것과 동치입니다.
- 따라서, 가능한 무대의 순서는 없습니다.

D. 부대 창설 행사

Subtask 3

- **Case 2. X 의 후보가 1개**
- 해당 무대를 마지막 무대로 확정해 줍시다.
- 이제 P 명의 병사들의 희망 목록에서 X 를 지워줄 수 있습니다. 이는 X 가 더 이상 병사들의 희망 무대에 영향을 미칠 수 없기 때문입니다.
- 방금 선택된 무대를 논외로 두고, 남은 무대 중 "마지막 무대"를 재귀적으로 탐색할 수 있습니다.
- X 를 지운 뒤 희망 무대 개수가 0개가 되는 병사의 경우, 직전에 지운 무대로 연습하러 갔다고 생각할 수 있습니다.
- 이후 희망 무대 개수가 1개가 되는 병사들만을 모아 다시 한번 최소 인원을 만족하는 무대들을 정리하면 됩니다.

Subtask 3

- **Case 3. X 의 후보가 2개 이상**
- X 의 정의상, 각 무대는 다른 무대에 독립적으로 최소 인원을 채운 무대입니다.
- 그렇기에 어떤 무대를 먼저 마지막 무대로 선정하더라도, 나머지 무대는 이후 "마지막 무대"로 편입될 수 있습니다.
- 따라서, **Case 2** 를 참조하여 가능한 후보 중 하나를 마지막 무대로 선정하고, 동일한 과정을 거쳐 X 를 정리하고 새로운 "마지막 무대" 후보를 정리해줍니다.
- 이후 새롭게 갱신된 후보 중 하나를 다시 선정하여 반복하면 됩니다.

Subtask 3

- N 개의 무대가 모두 결정되었다면, 마지막 무대로 결정된 역순으로 출력하면 됩니다.
- 무대 순서를 결정하는 중 언제라도 가능한 마지막 무대가 없는 순간이 나온다면, -1 을 출력하면 됩니다.
- 시간 복잡도는 $\mathcal{O}(NP)$ 로, $\mathcal{O}\left(N^2+NP\right)$, $\mathcal{O}(NNP)$ 등의 풀이도 빠듯하게나마 통과할 수 있도록 제한이 잡혀있습니다.

Subtask 4

- N 과 P 의 범위가 크지만, 대신 x_i 의 크기에 제한이 걸려있습니다.
- $\mathcal{O}(NP)$ 의 주 시간 소모는 마지막 무대를 희망한 병사의 목록 갱신 및 후보 재검토에 사용됩니다.
- 따라서, 임의의 병사가 희망했던 무대 또는 임의의 무대를 희망했던 병사의 길이를 답변의 길이 $\mathcal{O}(L)$ 에 비례하게 호출할 수 있다면, $\mathcal{O}(NP)$ 보다 훨씬 작은 $\mathcal{O}(\Sigma x_i)$ 로 문제를 해결할 수 있습니다.
- 앞으로 Σx_i 를 X 로 표현하겠습니다.

Subtask 4

- 최적화를 위해 사용할 수 있는 자료 구조는 생각보다 다양합니다.
- `tree set` 의 경우, 목록에 추가 및 삭제하는 과정이 $\mathcal{O}(\log X)$ 이므로, 최대 $\mathcal{O}(X \log X)$ 로 해결할 수 있습니다.
- `hash set` 의 경우, 추가 및 삭제 과정이 평균적으론 $\mathcal{O}(1)$ 이나 해시 충돌 등으로 인한 최악의 경우 $\mathcal{O}(X)$ 이므로 희망 무대 번호가 무작위라면 최대 $\mathcal{O}(X^2)$ 시간초과의 여지가 존재합니다.
- 다만, 이 문제의 경우 병사의 희망 무대 목록이 정렬되어 주어지는 등 해시 충돌로 인해 시간이 과도하게 소모되지 않는 입력을 받을 수 있게 보정되어 있습니다.
- 이외에도 `vector` 로 완료된 무대 목록을 보관하면서 $\mathcal{O}(X)$ 로 최적화하는 풀이 또한 존재합니다.

E. 감시 초소

dp
출제진 의도 – **Hard**

- 제출 52번, 정답 2명 (정답률 3.846%)
- 처음 푼 사람: **xiaowuc1**, 71분
- 출제자: 99asdfg

Subtask 1

- $N \le 15$로 제한이 매우 작으므로, 비트마스킹을 활용하여 감시초소를 설치하는 모든 2^N 개의 경우를 전부 순회할 수 있습니다.
- 모든 경우의 수를 순회하며 해당 경우의 총 비용과, 가능 여부를 기록한 뒤 그 중 최소를 출력하면 됩니다.
- 총 시간복잡도는 $\mathcal{O}\left(N2^N\right)$ 입니다.

Subtask 2

- $dp[i]$ 를 1 번째 지역부터 i 번째 지역까지 모든 지역을 감시하기 위한 최소 비용으로 정의합시다.
- 또한, $c(l, r)$ 을 구간 $[l, r]$ 을 모두 감시할 수 있는 개별 감시 초소들 중 최소 비용으로 정의하면 dp 식을 다음과 같이 세울 수 있습니다.
- $dp[i] = \min(dp[j] + c(j+1, i))(j < i)$
- 각각의 (l, r) 쌍에 대해, $c(l, r)$ 은 $\mathcal{O}(N)$ 으로 계산할 수 있습니다.
- 따라서, 약 N^2 개의 모든 (l, r) 쌍에 대한 $c(l, r)$ 값을 미리 계산해둔 뒤 위의 dp 식을 계산하면, 총 시간복잡도 $\mathcal{O}(N^3)$ 으로 문제를 해결할 수 있습니다.

E. 감시 초소

Subtask 3

- 구간 $[l, r]$ 을 모두 감시할 수 있는 감시초소들은 반드시 연속한 구간으로 나타낼 수 있습니다.
- 즉, 구간 $[l, r]$ 을 모두 감시하는 감시초소의 구간을 $[l_1, r_1]$ 으로 나타내면, l_1 및 r_1 값을 투 포인터 등을 활용하여 찾아낼 수 있습니다.
- 이때, l 값을 고정하고 r 값을 증가시키는 등의 방식으로 (l_1, r_1) 값을 찾아나가면 모든 (l, r) 쌍에 대한 (l_1, r_1) 쌍을 $\mathcal{O}\left(N^2\right)$ 으로 찾아낼 수 있습니다.
- 따라서, 해당 구간의 최솟값을 multiset 등의 자료구조로 관리하거나, 미리 N^2 개 쌍의 구간에 대한 비용의 최솟값 배열을 만들어 두는 방식으로 $\mathcal{O}\left(N^2 \log N\right)$ 또는 $\mathcal{O}\left(N^2\right)$ 으로 모든 (l_1, r_1) 쌍의 구간 비용 최소를 구할 수 있습니다.
- 위 값들을 활용하여 dp 식을 계산하면, 총 시간복잡도 $\mathcal{O}\left(N^2 \log N\right)$ 으로 문제를 해결할 수 있습니다.

Subtask 4

- 거리가 x인 지역을 추가로 감시하기 위해선 $x+1$만큼의 추가 인원이 필요하고, 한 감시 초소에 들어갈 수 있는 인원 수는 $P \le 100\,000$이므로, 임의의 l값에 대해 실제로 $c(l,r)$이 정의되는 r값의 개수는 최대 $2\sqrt{P}$개입니다.
- 따라서, 최대 $2\sqrt{P}$개의 (l,r) 쌍에 대한 (l_1,r_1) 쌍을 모두 구해줄 수 있습니다.
- 이때, set 등의 자료구조로 각각의 (l_1,r_1) 쌍의 최솟값을 구하면 $\mathcal{O}\left(N\log N\sqrt{P}\right)$이라는 시간복잡도로, 제한 $P \le N \le 100\,000$에는 통과할 수 없습니다.
- 대신, 각각의 l_1 값에 대해 구해야 하는 r_1 값이 최대 $2\sqrt{P}$라는 사실을 활용하여 미리 $2N\sqrt{P}$개의 쌍에 대한 최솟값을 저장하는 2차원 벡터를 만들어 두면, $\mathcal{O}\left(2N\sqrt{P}\right)$ 안에 dp값을 모두 계산할 수 있습니다.

Subtask 4

- 위의 풀이로도 시간 내에 문제를 해결할 수 있으나, 더욱 빠른 시간 내에 문제를 해결할 수 있습니다.
- 임의의 지역 x 에서, $x - \sqrt{P}$ 번 지역을 끝으로 하는 최대 감시 구역의 구간부터 $x + \sqrt{P}$ 번 지역을 끝으로 하는 최대 감시 구역의 구간까지 총 $2\sqrt{P}$ 개의 $[l, r]$ 구간들을 미리 저장해 둡시다.
- 이제, $dp[i]$ 를 계산하는 과정에서, 각각의 감시초소 설치 지역 i 에 대해 $dp[r] = \min(dp[l-1] + c[i])$ 및 $dp[i] = \min(dp[r])$ 로 계산해 줍시다.
- 이 경우, 전체적인 시간복잡도는 $\mathcal{O}\left(2N\sqrt{P}\right)$ 로 동일하나, 상수항이 많이 제거되며 공간복잡도도 줄어들어 더욱 빠른 시간 내에 정답을 구할 수 있습니다.

F. 합동 훈련

disjoint_set, offline_query, data_structure, smaller_to_larger

출제진 의도 – **Challenging**

- 제출 -번, 정답 -명 (정답률 -%)
- 처음 푼 사람: -, -분
- 출제자: 99asdfg

Subtask 1

- 단순히 서로 다른 부대들에 대한 N^2 개 쌍의 불만도를 기록해 주며, 불만도가 K 이상인 쌍이 하나라도 있다면 -1, 없다면 비용의 합을 출력하면 됩니다.
- N 제한이 작으므로 단순 2차원 배열을 사용하여 시뮬레이션으로 해결하면, 입력의 개수를 X 라 할 때 시간복잡도 $\mathcal{O}(X)$ 로 해결할 수 있습니다.

Subtask 2

- 각 부대를 노드, 부대를 차출할 때 필요한 비용을 노드의 가중치, 서로 다른 두 부대의 불만도 값을 가중치로 가지는 양방향 간선으로 이루어진 그래프를 구성해 봅시다.
- 그러면, 훈련 계획에서 일렬로 서 있는 순서는 간선의 가중치가 K 이하인 간선만을 따라가며 총 $2N$ 개의 노드를 선택하는 순서와 동일하게 표현할 수 있습니다.
- 이때, 간선의 가중치는 단순히 K 이상인지 미만인지에 따라 연결 가능성이 변화하므로, 가중치가 K 이하인 간선을 모두 제거합시다.
- 노드의 개수는 N 개이므로, 총 $2N$ 번 순회를 통해 임의의 노드에서 해당 연결 요소에 속한 모든 노드를 선택할 수 있음은 자명합니다.

Subtask 2

- 그러나, 그래프에서 간선을 제거하는 것은 간단한 작업이 아닙니다.
- 대신, 쿼리를 거꾸로 처리하며 간선의 가중치가 K 이하가 된 순간 해당 간선이 연결하는 두 노드를 분리 집합을 활용하여 연결하는 방식으로 문제를 해결합시다.
- 1 번 쿼리는 **Subtask 1**과 동일한 방식으로 순방향으로 처리해 주며, 각 부대 간 불만도를 저장해 둡시다.
- 또한, 2 번 쿼리가 들어오면 불만도만 증가시켜준 뒤 비용의 최댓값은 역방향으로 처리하는 과정에서 계산해 줍시다.

Subtask 2

- 순방향으로 쿼리를 모두 처리한 뒤, 각 쿼리를 통해 누적된 불만도 값들을 토대로 그래프를 구성합시다.
- $N \leq 1\,000$ 이므로, 단순히 N^2 쌍을 모두 순회하며 연결성 여부를 결정해줄 수 있습니다.
- 이때, 각각의 연결 요소에 대해 노드 가중치의 합을 저장해 두고, 해당 값들 중 최댓값을 구해줍시다.
- 비용은 쿼리 도중에 변화하지 않으므로, 단순히 서로 다른 두 연결 요소가 연결될 때마다 현재 시점의 최댓값과 비교하여 최댓값을 변경하는 방식으로 구현하면 각 쿼리당 $\mathcal{O}(1)$ 만에 최댓값을 구해줄 수 있습니다.
- 따라서, 총 시간복잡도는 약 $\mathcal{O}\left(N^2 \log N + Q \log N\right)$ 입니다.

Subtask 3

- 이제 비용이 변화할 수 있으므로, **Subtask 2**의 방식으로 최댓값을 구하는 대신 모든 값들을 multiset 등의 자료구조로 관리해 주면, 각 쿼리당 $\mathcal{O}(\log N)$ 안에 최댓값을 구할 수 있습니다.
- 또한, 각 연결 요소에 존재하는 비용값들 또한 map 등의 자료구조를 활용하여 관리해 줄 수 있습니다.
- N 제한이 작으므로, 비용값들을 관리하는 방식은 amortized $\mathcal{O}\left(N^2\right)$ 으로 관리해 줘도 무방하며, 이 경우 약 $\mathcal{O}\left(N^2 \log N + Q \log N\right)$ 으로 문제를 해결할 수 있습니다.

Subtask 4

- $N \le 100\,000$ 으로 N 제한이 상당히 커졌으므로, $N^2 \log N$ 의 시간복잡도로 그래프를 구성하는 것은 불가능합니다.
- 그러나, 입력으로 들어오는 수의 개수는 $500\,000$ 개 이하이므로, 임의의 두 노드를 잡았을 때 두 노드를 연결하지 않아야 하는 경우는 최대 $500\,000$ 번만 일어납니다.
- 따라서, 아직 아무런 연결 요소에도 속해있지 않은 노드들의 집합을 set 등의 자료구조로 관리하며 분리 집합으로 노드들을 연결하면 amortized $\mathcal{O}\left((N + 500\,000) \log N\right)$ 으로 모든 노드의 초기 연결 상태를 구성할 수 있습니다.
- 그 외의 구현은 **Subtask 2**와 동일하게 해줄 수 있습니다.

Subtask 5

- **Subtask 3**과 **Subtask 4**의 구현을 합쳐줍시다.
- 이때, N 제한이 **Subtask 3**보다 커졌으므로 비용값들을 관리할 때 smaller to larger 테크닉을 활용하면 더 효율적으로 비용값들을 관리해줄 수 있습니다.
- 총 시간복잡도는 약 $\mathcal{O}\left(N \log N + Q \log N\right)$ 입니다.