

Lemon Cup 풀이

by

Lemon Cup 출제진

문제	의도한 난이도	출제자
A 레몬 게임	Easy	pk661
B 레몬 상들리에	Normal	pk661
C 레몬이 스니켓의 위험한 대결	Challenging	woohyun_jng
D 레몬티처럼 달콤한 입술	Hard	woohyun_jng
E 레몬 경로	Easy	realpsdoingdamyoo
F 레몬컵 출제하기	Easy	kuhyaku
G 레몬컵 문제 배치하기	Hard	realpsdoingdamyoo
H 레몬레몬 왕국	Challenging	kuhyaku
I 레몬 왕국의 용사, 비타로	Hard	kuhyaku
J Lemon Tree	Normal	realpsdoingdamyoo
K 레몬향의 마흐트	Hard	kuhyaku
L 레몬 박물관	Normal	pk661

A. 레몬 게임

필요한 개념: greedy, math

예상 난이도: **Silver III**

- ✓ 제출 734회, 정답 255명
- ✓ 처음 푼 사람: qwersdfzxc!, 1분
- ✓ 출제자: pk661

A. 레몬 게임

부분문제 1 ($N \leq 3$)

- ✓ 가능한 모든 경우를 따져봅시다.
- ✓ $N = 1$ 이면 가능한 경우는 없습니다.
- ✓ $N = 2$ 이면 $(A_1, A_2) = (1, 2), (2, 1)$ 일 때만 가능합니다.
- ✓ $N = 3$ 이면 $(A_1, A_2, A_3) = (1, 1, 1)$ 일 때만 가능합니다.

A. 레몬 게임

만점 풀이

- ✓ 초기 상태에서 숫자 1이 적힌 레몬의 개수를 P , 숫자 2가 적힌 레몬의 개수를 Q 라고 정의합니다.
- ✓ 모든 레몬의 합은 $P + 2Q$ 이며, 게임을 완전히 끝내려면 이 합이 3의 배수여야 합니다.
- ✓ 삭제되는 숫자 2의 개수는 삭제되는 숫자 1의 개수를 초과할 수 없으므로 $P \geq Q$ 가 성립합니다.
- ✓ 위 두 조건, 즉 $P \geq Q$ 이고 $P + 2Q \equiv 0 \pmod{3}$ 이면 모든 레몬을 제거할 수 있음을 증명할 수 있습니다.

B. 레몬 종합선물세트

필요한 개념: math, combinatorics

예상 난이도: **Diamond V**

- ✓ 제출 57회, 정답 24명
- ✓ 처음 푼 사람: functional, 49분
- ✓ 출제자: pk661

B. 레몬 종합선물세트

- ✓ 원을 일직선으로 펼쳤을 때, 각 색 c 에 대해 가장 왼쪽 위치를 L_c , 가장 오른쪽 위치를 R_c 라고 하자.
- ✓ $[L_c, R_c]$ 구간들은 완전히 포함되거나 서로 겹치지 않는 laminar 구조를 이룬다.
- ✓ 그러므로 모든 색 c 에 대해 L_c, R_c 가 정해지면, 나머지 칸들의 색도 모두 자동으로 결정된다.

B. 레몬 종합선물세트

- ✓ 사용되는 색의 개수를 K 라고 하자.
- ✓ N 개의 색 중 K 개를 고르고, laminar 구조를 이루는 K 개의 구간을 선택하는 경우의 수는

$$\binom{N}{K} \times \frac{K!}{N} \binom{N}{K} \binom{N}{K-1}$$

이다.

- ✓ $K = 1, 2, \dots, N$ 에 대해 위 식의 값을 모두 더한 후, 정리하면 답이

$$(N-1)!(N!)^2 \sum_{K=1}^N \frac{1}{(K-1)!K!((N-K)!)^2(N-K+1)!}$$

임을 알 수 있다.

C. 레몬이 스니켓의 위험한 대결

필요한 개념: greedy, priority-queue, tree, smaller-to-larger

예상 난이도: **Ruby V**

- ✓ 제출 38회, 정답 9명
- ✓ 처음 푼 사람: sadtreap, 47분
- ✓ 출제자: woohyun_jng

C. 레몬이 스니켓의 위험한 대결

부분문제 1 ($1 \leq N \times K \leq 10$)

✓ 가능한 경우의 수는

$$\frac{(N \cdot K)!}{(N!)^K}$$

이며, 이 값의 최대 수준이 $10! = 3\,628\,800$ 이므로 완전 탐색으로도 충분히 계산이 가능합니다.

C. 레몬이 스니켓의 위험한 대결

부분문제 2 ($K = 2, 1 \leq N \leq 5,000$)

- ✓ 동적 계획법(DP)을 이용하여 효율적으로 해결할 수 있습니다.
- ✓ $dp[x][y]$ 를 "1번째 사람이 x 까지, 2번째사람이 y 까지 이동했을 때의 최소 비용" 으로 정의합니다.
- ✓ 점화식은 다음과 같습니다:

$$dp[x][y] = \min(dp[x-1][y], dp[x][y-1]) + A_{1,x} \times A_{2,y}$$

- ✓ 모든 상태를 채우는 데 $O(N^2)$ 의 시간 복잡도가 소요되며, 제한 조건 하에서 충분히 통과할 수 있습니다.

C. 레몬이 스니켓의 위험한 대결

부분문제 3 ($K = 2$)

- ✓ 두 명의 이동 순서는 길이 $2N$ 의 이진 수열(1이 N 개, 2가 N 개)로 표현할 수 있습니다.
- ✓ 수열의 i 번째 원소가 x 이고, 이전에 동일한 값 x 이 총 y 회 등장했다면, 이를 " x 번째 사람이 현재 높이 y 에서 1만큼 위로 이동한다"로 해석합니다.
- ✓ 각 이동에 대해

$$B_i = A_{x,y+1} - A_{x,y}, \quad C_i = A_{x,y+1} \quad (1 \leq i \leq 2N)$$

으로 정의하고, 시작 전 상태를 $B_0 = A_{1,0} + A_{2,0}$ 로 둡니다.

C. 레몬이 스니켓의 위험한 대결

✓ 전체 비용은

$$\sum_{i < j} B_i \times C_j - \sum A_{1,i} \times A_{1,i+1} - \sum A_{2,i} \times A_{2,i+1}$$

로 나타낼 수 있습니다.

- ✓ 위의 풀은 **Faker's Algorithm**¹이 적용 가능한 구조이므로, 이를 이용해 최소 비용을 계산할 수 있습니다.
- ✓ 루트 정점 아래에 길이가 N 인 체인 2개를 연결하고 해당 알고리즘을 적용합니다.
- ✓ 이 외에도 i 번째 대원마다 볼록 꺾질을 만들어 잘 합쳐주는 풀이도 존재합니다.
- ✓ 백준의 이동하기 4 문제의 정해와 비슷한 그리드를 이용하면 이 풀이를 유도해 낼 수 있습니다.

C. 레몬이 스니켓의 위험한 대결

만점 풀이

- ✓ $K \neq 2$ 인 경우에도 동일한 아이디어로 해결할 수 있습니다.
- ✓ 루트 정점 아래에 길이가 N 인 체인 K 개를 연결한 뒤, Faker's Algorithm을 적용하면 됩니다.

D. 레몬티처럼 달콤한 입술

필요한 개념: dp, segment-tree

예상 난이도: **Diamond III**

- ✓ 제출 50회, 정답 1명
- ✓ 처음 푼 사람: moonrabbit2, 685분
- ✓ 출제자: woohyun_jng
- ✓ NMIXX - Kiss (원곡: 레인보우)

D. 레몬티처럼 달콤한 입술

부분문제 1 ($1 \leq N, Q \leq 5,000$)

- ✓ 스택을 이용한 풀이 등 다양한 접근법이 존재하지만, 여기서는 동적 계획법(DP)을 이용한 방법을 소개하겠습니다.
- ✓ 이 문제는 몇 가지 핵심적인 관찰을 통해 최적해의 형태를 제한하고, 그 위에서 DP를 수행하는 방식으로 해결할 수 있습니다.

D. 레몬티처럼 달콤한 입술

관찰 1

- ✓ 두 조명등이 비추는 영역에 겹치는 부분이 있다고 가정합니다.
- ✓ 두 조명등이 각각 높이 h_1, h_2 를 가지며, 지면의 $[x, y]$ 구간을 모두 비추고 있다고 하겠습니다.
- ✓ 이 경우, $[x, y]$ 구간의 중간 지점에 높이 $h_1 + h_2$ 의 조명등을 하나 설치하면, 기존 두 조명등이 커버하던 모든 구간을 비출 수 있습니다.
- ✓ 따라서, 어떤 두 조명등도 비추는 영역이 겹치지 않는 최적해가 항상 존재한다고 볼 수 있습니다.

D. 레몬티처럼 달콤한 입술

관찰 2

- ✓ 두 개 이상의 조명등이 설치되어 있고, 그 중 하나가 $[0, N]$ 구간 바깥을 비추고 있다고 가정합니다.
- ✓ 이 경우, 조명등의 조건 $A_i \leq \min(N - i, i)$ 에 따라 동일한 비용으로 해당 조명등을 $[0, N]$ 구간 안으로만 비추도록 위치를 조정할 수 있습니다.
- ✓ 관찰 1을 이용하면, 조명등이 2개 이상 설치된 경우에는 항상 조명등 간의 비추는 구간이 겹치지 않고, 모두 $[0, N]$ 구간 내에 포함되는 최적해가 존재함을 알 수 있습니다.
- ✓ 한편, 조명등이 1개만 설치되는 경우에는 $\left\lfloor \frac{N}{2} \right\rfloor$ 위치에 높이 $\left\lceil \frac{N}{2} \right\rceil$ 인 조명등을 설치하면 전체 구간을 비출 수 있으므로, 이 경우는 예외 처리해주면 됩니다.

D. 레몬티처럼 달콤한 입술

관찰 3

- ✓ 앞서 살펴본 관찰들을 바탕으로, 최적해의 구조를 다음과 같이 강제할 수 있습니다.
- ✓ 인접한 조명등들이 만드는 삼각형은 끝점이 1씩 차이 나며, 맨 왼쪽 삼각형의 왼쪽 끝점은 반드시 $x = 0$, 맨 오른쪽 삼각형의 오른쪽 끝점은 반드시 $x = N$ 에 위치합니다.
- ✓ 이 사이에 조명등이 비추지 않는 구간 $[i, i + 1]$ 이 존재할 수 있는데, 이러한 구간을 **split-edge**라고 정의합니다.
- ✓ split-edge들의 집합을 S 라고 할 때, 전체 조명등 설치에 필요한 개수는 $\frac{N - |S|}{2}$ 입니다.
- ✓ 따라서 $|S|$ 를 최대화하는 것이 곧 최적해를 구하는 것이 됩니다.

D. 레몬티처럼 달콤한 입술

split-edge의 조건

- ✓ split-edge가 되기 위해서는 다음의 조건들을 만족해야 합니다.
 1. 다른 split-edge와 인접해서는 안 됩니다.
 2. S 에 속하는 모든 split-edge를 제거한 뒤 남는 연속한 구간은 길이가 모두 짝수여야 합니다.
 3. $[0, 1]$ 또는 $[N - 1, N]$ 는 split-edge가 될 수 없습니다.
 4. 주민이 있는 위치에 조명등을 설치했을때 비춰지는 영역에 포함되면 안 됩니다.
- ✓ 각 조건의 이유는 앞서 정리한 최적해의 구조를 고려하면 쉽게 이해할 수 있습니다.

D. 레몬티처럼 달콤한 입술

- ✓ 이제 $|S|$ 의 최댓값을 구하기 위한 동적 계획법(DP)을 정의해봅시다.
- ✓ $dp[n]$ 을 다음과 같이 정의합니다: 마지막 split-edge가 $[i - 1, i]$ 이고 i 의 홀짝성이 n 과 동일할 때, 구간 $[0, n]$ 내에서 만들 수 있는 split-edge의 최대 개수.
- ✓ $[n - 1, n]$ 구간이 split-edge의 후보인 경우, 점화식은 다음과 같습니다:
$$dp[n] = \max(dp[n - 2], dp[i - 3] + 1, 1_{n \text{ is odd}})$$
- ✓ 그렇지 않은 경우에는 단순히 $dp[n] = dp[n - 2]$ 입니다.
- ✓ 단, 조건 4에 의해 $[i - A[i], i + A[i]]$ 범위 내에는 split-edge를 둘 수 없으므로, 해당 범위들을 미리 전처리해줘야 합니다.
- ✓ 이 전처리는 imos법을 이용해 $O(N)$ 에 수행할 수 있고 DP 수행은 $O(N)$ 으로 가능하므로 전체 시간복잡도는 $O(QN)$ 입니다.

D. 레몬티처럼 달콤한 입술

부분문제 2 ($A_i \geq 1$)

- ✓ 조건 4에 의해 split-edge가 존재할 수 없으므로, $|S|$ 의 최댓값은 항상 0입니다.
- ✓ 따라서 전체 구간을 커버하기 위해 $\left\lfloor \frac{N}{2} \right\rfloor$ 위치에 높이 $\left\lceil \frac{N}{2} \right\rceil$ 의 조명등을 하나 설치하는 것이 최적입니다.
- ✓ 각 쿼리마다 $\left\lceil \frac{N}{2} \right\rceil$ 을 출력하면 됩니다.

D. 레몬티처럼 달콤한 입술

부분문제 3 ($A_i = 0$)

- ✓ 이 경우, 모든 구간 $[i - 1, i]$ 가 split-edge의 후보가 될 수 있습니다.
- ✓ 가능한 split-edge들을 2씩 건너뛰며 배치하는 것이 최적입니다.
- ✓ 따라서 최대한 많은 split-edge를 배치하면 정답은 $\left\lfloor \frac{N}{3} \right\rfloor + 1$ 입니다.

D. 레몬티처럼 달콤한 입술

만점 풀이

- ✓ DP를 더 빠르게 수행하기 위해, 세그먼트 트리를 활용한 최적화를 적용합니다.
- ✓ 이러한 유형에서는 보통 **행렬 기반 세그먼트 트리**를 이용하는 것이 효과적입니다.
- ✓ 각 구간 $[l, r]$ 마다 다음 세 가지 상태를 고려할 수 있습니다:
 - 가장 왼쪽 split-edge의 인덱스의 홀짝성
 - 인접한 split-edge의 존재 여부
 - $[l, l + 1]$ 구간에 split-edge가 존재하는지 여부
- ✓ 동일하게, 가장 오른쪽 split-edge에 대해서도 위 상태를 추적해야 하므로, 각 구간마다 총 3×3 개의 상태가 존재합니다.

D. 레몬티처럼 달콤한 입술

- ✓ 따라서 DP 전이를 3×3 크기의 행렬로 표현하고, 세그먼트 트리를 통해 이를 병합할 수 있습니다.
- ✓ 두 구간의 행렬을 병합할 때는 다음 세 가지 경우를 고려해야 합니다:
 1. 좌측 구간 끝과 우측 구간 시작 모두에 split-edge가 없는 경우
 2. 둘 중 하나에만 split-edge가 있는 경우
 3. 양쪽 모두에 존재하지만 인접하지 않은 경우

D. 레몬티처럼 달콤한 입술

- ✓ $A[i]$ 가 변경된다는 것은, split-edge로 사용할 수 없는 구간이 동적으로 바뀐다는 의미입니다.
- ✓ 이러한 정보를 효율적으로 관리하기 위해, **화성지도 세그먼트 트리²**를 사용합니다.
- ✓ 행렬 세그먼트 트리와 화성지도 세그먼트 트리를 적절히 결합하면, 이 문제를 처리할 수 있는 강력한 자료구조를 구성할 수 있습니다.
- ✓ 이를 통해 각 쿼리에 대해 $O(27 \log N)$ 시간에 업데이트와 질의가 가능하며,
- ✓ 전체 시간 복잡도는 전처리 $O(27N \log N)$, 쿼리 처리 $O(27Q \log N)$ 으로 해결됩니다.

E. 레몬 경로

필요한 개념: dp, bfs

예상 난이도: **Gold I**

- ✓ 제출 385회, 정답 102명
- ✓ 처음 푼 사람: urd05, 8분
- ✓ 출제자: realpsdoingdamyoo

E. 레몬 경로

서브태스크

- ✓ 서브태스크는 크게 중요하지 않으므로 짧게 설명하겠습니다.
- ✓ 서브태스크 1은 각각의 점에 대해 1번 점과 연결되는 간선의 가중치를 출력하면 됩니다.
- ✓ 서브태스크 2는 1번 점에서부터의 BFS를 통해 구한 최단 경로의 길이를 출력하면 됩니다.
- ✓ 서브태스크 3은 가능한 모든 레몬 경로를 구해 답을 구하면 됩니다.
- ✓ 서브태스크 4는 만점 풀이와 거의 차이가 없으므로 생략하겠습니다.

E. 레몬 경로

만점 풀이

- ✓ 1번 점과 p번 점 사이의 **레몬 거리**를 구하기 위해 필요한 정보는 다음과 같습니다.
 - p번 점까지 최소 개수의 간선을 거쳐 이동하는 경우의 수
 - p번 점까지 최소 개수의 간선을 거쳐 이동하는 경로들의 가중치 합

E. 레몬 경로

- ✓ BFS를 돌리며, 거리가 d 인 점에서 $d + 1$ 인 점으로 위의 두 값을 전이해주면 됩니다.
- ✓ BFS를 두 번 돌려 거리를 구하는 과정, 값을 전이하는 과정을 각각 진행해도 되지만, 한 번으로도 가능합니다.
- ✓ BFS를 돌리다가 거리가 d 인 점 p 를 방문했다고 해봅시다.
- ✓ 이때 거리가 $d - 1$ 인 점들은 모두 방문한 상태이므로, 점 p 와 이웃한 정점들 중 1번 점과의 거리가 $d - 1$ 인 점들에게서 정보를 받아오면 됩니다.
- ✓ BFS + 정답 출력의 과정을 $O(M + N \log X)$ 에 완료할 수 있습니다.

F. 레몬컵 출제하기

필요한 개념: bitmask, bfs

예상 난이도: **Platinum V**

- ✓ 제출 234회, 정답 86명
- ✓ 처음 푼 사람: dong5995, 18분
- ✓ 출제자: kuhyaku

F. 레몬컵 출제하기

부분문제 1 ($N \leq 500$)

- ✓ 간단히, 지금까지 나온 모든 문제와 전부 비교해보면 됩니다.
- ✓ 시간복잡도는 $O(N^2)$ 입니다.

F. 레몬컵 출제하기

만점 풀이

- ✓ 어떤 알고리즘 조합이 유효한지 아닌지 여부를 저장하는 크기 2^K 의 bool 배열을 관리합니다.
- ✓ 새로 알고리즘이 추가될 때, BFS하듯이 갱신해줄 수 있습니다.
- ✓ 이미 어떤 상태를 방문했다면, 그 상태의 submask도 방문처리가 되어 있을 것이기 때문입니다.
- ✓ 시간복잡도는 $O(N + K2^K)$ 입니다.

G. 레몬컵 문제 배치하기

필요한 개념: constructive, case-work

예상 난이도: **Diamond III**

- ✓ 제출 8번, 최고점 0점 (정답률 0.00%)
- ✓ 처음 푼 사람: -
- ✓ 출제자: realpsdoingdamyoo

G. 레몬컵 문제 배치하기

- ✓ 해 구성에 앞서 수열 뒤집기가 불가능한 경우부터 알아봅시다.
- ✓ 길이 m 의 구간을 뒤집으면, 뒤집기 전 구간의 inversion 개수를 I 라고 할 때 뒤집은 이후 inversion의 개수는 $\frac{m(m-1)}{2} - I$ 가 됩니다.
- ✓ 구간을 2개 뒤집으므로, 각 구간의 inversion을 I_1, I_2 라고 할 때 연산 이후 inversion은 $m(m-1) - I_1 - I_2$ 개가 됩니다.
- ✓ 따라서, 전체 수열의 inversion 개수의 홀짝성이 유지된다는 것을 관찰할 수 있습니다.
- ✓ 초기 inversion은 0개이므로, 뒤집힌 수열의 inversion 개수 $(\frac{n(n-1)}{2})$ 가 짝수인 $n = 4k$, $n = 4k + 1$ 의 경우만 가능합니다.

G. 레몬컵 문제 배치하기

✓ 해 구성은 다음과 같은 4가지의 경우로 나누어 생각해볼 수 있습니다.

- $N = 8k$

- $N = 8k + 1$

- $N = 8k + 4$

- $N = 8k + 5$

✓ 각각의 경우에 대해 문제를 해결해 봅시다.

✓ 이 문제 풀이의 맨 아래에 정해 코드를 첨부했습니다. 각각의 경우에 대해 중간 과정을 확인해보면 풀이 이해에 도움이 됩니다 :)

G. 레몬컵 문제 배치하기

- ✓ $N = 8k$ 일 때는, fig 1과 같은 연산을 똑같이 2번 시행하면 순서를 뒤집을 수 있습니다.(총 10회)
- ✓ $N = 8k + 1$ 일 때는, 먼저 1번부터 $N - 1$ 번 문제까지의 순서를 모두 뒤집습니다.
- ✓ 그 후, fig 2와 같이 6회의 연산을 추가해 N 번 문제를 맨 앞으로 옮길 수 있습니다.(총 16회)

- ✓ fig 1에서 구간 1, 2, ..., 8은 길이가 k 로 동일합니다.
- ✓ fig 2에서 구간 1, 2, 3, 4는 길이가 $2k$ 로 동일합니다. 2A, 3B는 길이 1, 2B와 3A는 길이 $2k-1$ 의 쪼개진 구간을 의미합니다.
- ✓ X' 는 구간 X 가 뒤집혔다는 것을 의미합니다.

G. 레몬컵 문제 배치하기

1	2	3	4	5	6	7	8
4'	3'	2'	1'	8'	7'	6'	5'
4'	3'	1	2	7	8	6'	5'
4'	3'	1	7'	2'	8	5	6
4'	3'	7	1'	8'	2	5	6
4'	3'	7	8	1	2	6'	5'

1	2A	2B	3A	3B	4	X
2A'	1'	2B	3A	3B	X'	4'
1	2A	2B	X	3B'	3A'	4'
1	2A	2B	X	3A	3B	4
1	X'	2B'	2A'	3A	4'	3B'
X	1'	2B'	2A'	3A	3B	4
X	1	2A	2B	3A	3B	4

G. 레몬컵 문제 배치하기

- ✓ $N = 8k + 4$ 일 때는 6회의 연산으로 문제 순서를 $\{N, N - 1, 3, 4, \dots, N - 3, N - 2, 2, 1\}$ 와 같이 만들 수 있습니다.
- ✓ 이후 $N = 8k$ 일 때와 같이 10회의 연산으로 남은 부분을 뒤집으면 됩니다.(총 16회)
- ✓ $N = 8k + 5$ 일 때에도 비슷하게 7회의 연산으로 문제 순서를 $\{N, N - 1, 3, 4, \dots, N - 3, N - 2, 2, 1\}$ 와 같이 만들 수 있습니다.
- ✓ 뒤집어야 할 구간의 길이가 $8k + 1$ 이지만, 양쪽 끝에 2개씩의 여유가 있으므로 10회의 연산으로 뒤집어줄 수 있습니다.(총 17회)
- ✓ 다만 이 경우 $N = 13$ 이 예외가 되는데, 이 경우는 적당한 그리디 등의 방법을 사용하여 17회 이내의 답안을 찾으면 됩니다.
- ✓ 정해 코드는 다음과 같습니다. MCS(cpp / py, 비밀번호 lemon)

H. 레몬레몬 왕국

필요한 개념: offline-query, segment-tree, sweeping

예상 난이도: **Diamond I**

- ✓ 제출 69회, 정답 8명
- ✓ 처음 푼 사람: mickeyjung, 128분
- ✓ 출제자: kuhyaku

H. 레몬레몬 왕국

부분문제 1 ($N, M, Q \leq 200$)

- ✓ 어떤 그래프가 원형임은, 모든 정점의 차수가 0 혹은 2임과 동치입니다.
- ✓ 각 쿼리마다, $O(M^2)$ 정도의 시간복잡도에 구하면 됩니다.
- ✓ 이중 for문을 돌면서, l 을 고정했을 때 r 을 하나씩 증가시키며 빠르게 판정할 수 있습니다
- ✓ 시간복잡도는 쿼리당 $O(M^2)$, 전체 $O(N + QM^2)$ 입니다.

H. 레몬레몬 왕국

부분문제 2 ($N, M, Q \leq 2000$)

- ✓ 미리 각 l, r 쌍이 원형인지 $O(M^2)$ 에 계산해둡니다.
- ✓ 2차원 누적합을 이용하면 쿼리당 $O(1)$ 에 처리할 수 있습니다.
- ✓ 시간복잡도는 $O(N + M^2 + Q)$ 입니다.

H. 레몬레몬 왕국

부분문제 3 ($Q = 1, L_i = 1, R_i = M$)

- ✓ L 을 x 축, R 을 y 축으로 생각한 2차원 평면을 생각합니다.
- ✓ 각 정점마다, 이 정점이 차수 조건을 만족할 L, R 범위를 이차원 평면 위의 몇 개의 직사각형으로 나타낼 수 있습니다. 또한, 총 직사각형 개수는 $O(N + M)$ 개입니다.
- ✓ 위 직사각형 중 N 개가 겹치는 점이 가능한 (L, R) 쌍입니다.
- ✓ 사실 정확히는 계단 모형이지만, 이는 추가로 $[i, i] \times [1, i]$ 직사각형 M 개를 추가한 후 직사각형 $N + 1$ 개가 겹치는 점으로 생각하면 됩니다.
- ✓ 어떤 점에서 직사각형은 최대 $N + 1$ 개까지 겹칠 수 있으므로, 화성지도 세그먼트 트리 등을 이용해 스위핑을 하면 풀 수 있습니다.
- ✓ 시간복잡도는 $O((N + M) \log M)$ 입니다.

H. 레몬레몬 왕국

만점 풀이

- ✓ 다음과 같은 연산을 지원하는 세그먼트 트리를 짜면 됩니다.
 - $L \leq i \leq R$ 에 대해 $A[i] := A[i] + k$
 - $A[i] = N + 1$ 에 대해 $B[i] := B[i] + 1$
 - $L \leq i \leq R$ 에 대해 $B[i]$ 합
- ✓ 즉, 값이 $N + 1$ 이 되는 위치들을 히스토릭하게 관리하면 됩니다.
- ✓ 이는 각 노드에 리프들의 $A[i]$ 의 최댓값과 그 개수, 자신의 왼쪽 서브트리와 오른쪽 서브트리의 max와 자신이 동일한지를 담고 이에 따라 적당히 잘 전파시키면 됩니다.
- ✓ 시간복잡도는 $O((N + M + Q) \log M)$ 입니다.
- ✓ 비슷한 세그먼트 트리를 사용하는 문제로는 BOJ 12827 Posters가 있습니다.

H. 레몬레몬 왕국

- ✓ 정해 이외에 의도하지 않은 $O((N + M + Q)\sqrt{M})$ 풀이가 있습니다.
- ✓ 적당한 XOR 해싱과 mo's 알고리즘을 이용해 풀 수 있습니다.

I. 레몬 왕국의 용사, 비타로

필요한 개념: greedy, priority-queue, disjoint-set

예상 난이도: **Diamond III**

- ✓ 제출 87회, 최고점 24점
- ✓ 가장 먼저 최고점에 도달한 사람: urd05, 653분
- ✓ 출제자: kuhyaku

I. 레몬 왕국의 용사, 비타로

부분문제 1 ($N \leq 500$) + 부분문제 2 ($N \leq 2,000$)

- ✓ 각 후보를 추가했을때 비용을 빠르게 계산해주면 됩니다.
- ✓ 비용을 (그 후보를 추가했을 때 추가되는 수) - (기존 추가되는 수) 처럼 계차로 계산하면 간편하게 계산할 수 있습니다.
- ✓ 구현에 따라 $O(N^3)$ 또는 $O(N^2)$ 입니다.

I. 레몬 왕국의 용사, 비타로

부분문제 3 (완전 이진 트리)

- ✓ ETT 풀이를 생각합니다.
- ✓ 어떤 i 를 추가로 탐험했을 때, 다른 후보들의 비용 변화를 관찰합니다.
- ✓ i 부터 부모를 타고 올라가면서, 그 서브트리에 있는 모든 후보에 적당한 상수를 더하기를 반복해 비용 갱신을 할 수 있음을 알 수 있습니다.
- ✓ $O(N \log^2 N)$ 에 풀 수 있습니다.

I. 레몬 왕국의 용사, 비타로

만점 풀이

- ✓ 부분문제 3의 풀이를 최적화합니다.

I. 레몬 왕국의 용사, 비타로

- ✓ 그대로 불가능한 이유가 뭘까요? 직선과 같이 깊이가 깊으면 문제가 됩니다.
- ✓ 직선의 경우에 끝까지 따라 올라가면 $O(N^2 \log N)$ 이 됩니다.
- ✓ 하지만 직선에선 굳이 하나하나 올라가지 않고 바로 루트로 올라가는 것을 생각할 수 있습니다.
- ✓ 즉, "경로 압축"을 이용할 수 있습니다.

I. 레몬 왕국의 용사, 비타로

- ✓ 만약 i 에서 i 의 부모로 올라갈 때, i 쪽 서브트리 말고 다른 쪽 서브트리에 남은 정점이 없다면 i 와 i 의 부모를 합칠 수 있습니다.
- ✓ 압축되지 않고 올라가는 경우를 "유효한 갱신"이라 합시다.
- ✓ 추가적으로, i 번 타고 올라갔을 때, 유효한 갱신이 이루어지려면 옆에 적어도 크기가 $i - L (= 10)$ 이상인 서브트리가 달려있어야 합니다.
- ✓ 유효 갱신 횟수를 m 이라 하면, $1 + 2 + 3 + 4 \dots (m - L) = O(m^2) \leq N$ (트리전체크기) 여야 합니다.
- ✓ 그러므로 유효한 갱신은 최대 $O(\sqrt{N} + L)$ 번 일어납니다.
- ✓ ETT 풀이에 적용하면 $O(N(\sqrt{N} + L) \log N)$ 풀이를 얻습니다.
- ✓ 이로 부분문제 4를 풀 수 있습니다.

I. 레몬 왕국의 용사, 비타로

- ✓ priority-queue 등의 자료구조를 이용해, 갱신 시간복잡도를 $O(\log N)$ 고정인 아닌, 자식 개수 비례로 바꿉니다.
- ✓ 이 경우, 시간복잡도가 $O(N\sqrt{N} + L \log N)$ 임을 증명할 수 있습니다.
- ✓ 자세한 증명은 생략합니다.

J. Lemon Tree

필요한 개념: math, trie

예상 난이도: **Platinum I**

- ✓ 제출 111회, 정답 8명
- ✓ 처음 푼 사람: moonrabbit2, 235분
- ✓ 출제자: realpsdoingdamyoo

J. Lemon Tree

- ✓ 서브태스크가 상당히 중요한 문제입니다. 차근차근 풀어봅시다!

서브태스크 1($Q = 1$)

- ✓ 깊이 i 인 점과 그 부모를 잇는 간선에 대해, 그 간선을 경로에 포함하는 두 점의 쌍은 $(2^{d+i-1} - 1)(2^{d+1} - 2^{d-i+1})$ 개 입니다.
- ✓ 또한 그러한 간선은 2^i 개 존재합니다.
- ✓ $\sum_{i=1}^d (2^{d+i-1} - 1)(2^{d+1} - 2^{d-i+1})2^i = (d - 2)2^{2d+2} + (d + 4)2^{d+1}$ 입니다.
- ✓ p 에 무관하게 $(d - 2)2^{2d+2} + (d + 4)2^{d+1}$ 를 출력하면 됩니다.

J. Lemon Tree

서브태스크 2($Q = 2$)

- ✓ 두 쿼리가 들어오는 경우를, 한 서브트리가 다른 서브트리에 포함되는 경우와 그렇지 않은 경우로 나눌 수 있습니다.
- ✓ 각각의 서브트리 내의 점 쌍들에 대해서는 **서브태스크 1**과 같이 해결할 수 있으므로 두 서브트리에서 점을 하나씩 선택할 때의 경우만 생각해 봅시다.
- ✓ 두 서브트리를 서브트리 1, 서브트리 2라고 하고, 각각의 깊이를 d_1, d_2 , 크기를 cnt_1, cnt_2 로 정의합니다. cnt 는 각각 $2^{d_1+1} - 1, 2^{d_2+1} - 1$ 과 같습니다.
- ✓ 이제 두 경우에 대해 문제를 해결해 봅시다.

J. Lemon Tree

- ✓ 먼저 후자의 경우부터 풀어봅시다. 깊이가 각각 d_1, d_2 인 두 서브트리와, 그 사이를 잇는 길이 c 의 경로로 생각할 수 있습니다.
- ✓ 두 서브트리에서 한 점씩 뽑으므로 각각의 쌍을 잇는 경로는 **점1 - 루트1 - 루트2 - 점2**과 같이 나타납니다.
- ✓ 길이 c 의 경로 **루트1 - 루트2**는 $cnt_1 \times cnt_2$ 번 지나짐을 알 수 있습니다.
- ✓ 그리고 **점1 - 루트1**은 서브트리 1의 모든 점에 대해 cnt_2 번 지나집니다.
- ✓ 따라서 **점1 - 루트1**만을 고려했을 때 더해지는 값은 $\sum_{i=0}^{d_1} 2^i i = (d-1)2^{d+1} + 2$ 의 cnt_2 배입니다.
이 값을 dsm_1 으로 정의하겠습니다.
- ✓ 이 경우 두 번째 쿼리의 정답은 $cnt_1 \times cnt_2 \times c + cnt_1 \times dsm_2 + dsm_1 \times cnt_2$ 가 됩니다.

J. Lemon Tree

- ✓ 다음으로 전자의 경우도 생각해 봅시다. d_1, d_2, c 는 위와 같이 정의합니다.
- ✓ 이 경우도 비슷하게 생각할 수 있습니다. 일단 모든 경로가 **점1 - 루트1 - 루트2 - 점2**와 같다고 가정한 후, 불필요하게 카운트된 길이를 빼 주면 됩니다.
- ✓ 일반성을 잃지 않고 두 서브트리 중 위에 있는 것을 서브트리 1이라고 합시다.
- ✓ 이때 불필요하게 카운트된 경로 길이의 합은 $2 \sum_{i=0}^d 2^i (d - i) = 2^{d+2} - 2d - 4$ 입니다.
- ✓ 따라서 이 경우에 해당하는 Dsm_1 을 $dsm_1 - (2^{d+2} - 2d - 4) = 2^{d+1}(d - 3) + 2d + 6$ 으로 정의하면, 두 번째 쿼리의 정답은 $cnt_1 \times cnt_2 \times c + cnt_1 \times dsm_2 + Dsm_1 \times cnt_2$ 가 됩니다.
- ✓ 거의 다 왔어요!!

J. Lemon Tree

서브태스크 3($2^d p \leq 2^{21}$)

- ✓ 각각의 점들에 대한 cnt, dsm, Dsm 을 구하면, 이들을 적절히 합치며 사용할 수 있다는 것을 알 수 있습니다.
- ✓ 쿼리가 들어올 때마다, 새 서브트리보다 아래에 있는 서브트리들과의 거리 합, 새 서브트리보다 위에 있는 서브트리들과의 거리 합, 둘 모두 아닌 경우의 거리 합, 새 서브트리 내에서의 거리 합을 계산해 주면 됩니다.
- ✓ 새 쿼리를 처리할 때마다, 구한 dsm 값을 조상 노드에 적절히 더해 놓으면 $\log p$ 개의 정점만을 확인함으로써 앞의 세 값을 구해 줄 수 있습니다.
- ✓ $2^d p$ 가 작으므로, 완전 이진 트리를 직접 구성해 풀 수 있습니다.

만점 풀이

- ✓ 트리에서 꼭 필요한 점은 사실 p 들과 그 조상들밖에 없습니다.
- ✓ 따라서 각각의 쿼리에 따라 $\log p$ 개의 정점만을 만들도록 트라이를 만들면, $2^d p$ 의 크기 제한이 없어도 문제를 풀 수 있습니다.
- ✓ 총 시간복잡도는 $O(Q \log p)$ 입니다.

K. 레몬향의 마흐트

필요한 개념: hld, binary-search, heuristics

예상 난이도: **Diamond VI**

- ✓ 제출 93회, 정답 3명
- ✓ 처음 푼 사람: mickeyjung, 505분
- ✓ 출제자: kuhyaku

K. 레몬향의 마흐트

부분문제 1 (트리가 직선)

- ✓ N 번 노드에 10^5 의 마력을 흘려주면 바로 답을 알 수 있습니다.

부분문제 3 (성게)

- ✓ 섭테를 순서대로 밀지 않는 능력이 필요합니다.
- ✓ 사실 부분문제 2의 풀이는 어렵습니다.
- ✓ 매개 변수 탐색을 이용합니다.
- ✓ $w_i \geq M$ 인지 확인하려면, 모든 리프노드에 M 만큼의 유량을 흘리고 반환값이 $M \times (\text{리프수})$ 인지 확인하면 됩니다.

K. 레몬향의 마흐트

부분문제 2 (균형 이진 트리, $N \leq 3\,000$)

- ✓ 깊이마다 뽑아내면 됩니다.
- ✓ $k = 0, 1, 2 \dots$ 순서대로 순회하며, 깊이 k 에 위치한 간선들의 용량의 최솟값을 찾습니다.
- ✓ 깊이 $< k$ 인 간선들은 모두 강화 후, 부분문제 3을 풀듯이 매개 변수 탐색을 이용해 알아낼 수 있습니다.
- ✓ $\lceil \log N \rceil \lceil \log W \rceil$ 번의 마력 측정을 사용합니다.

K. 레몬향의 마흐트

부분문제 4 ($N \leq 3\,000$)

- ✓ HLD를 사용합니다.
- ✓ 1번 노드를 포함하는 체인은 부분문제 1처럼 알아냅니다.
- ✓ 그 이후 드러난 체인들에 대해 부분문제 1과 3을 풀듯이 알아냅니다.
- ✓ 이를 반복하면 체인을 뽑아내는 과정은 최대 $\lceil \log N \rceil$ 번 반복됩니다.
- ✓ $\lceil \log N \rceil \lceil \log W \rceil$ 번의 마력 측정을 사용합니다.

만점 풀이

- ✓ 각각의 체인들에 대해 이분 탐색을 진행하기 전, 지금까지 알아낸 최솟값보다 작은지 확인합니다.
- ✓ 체인들의 집합을 무작위로 섞은 후 이 과정을 진행하면, 평균적으로 $\log \log N$ 번의 이분 탐색을 진행하게 됩니다.
- ✓ 또한 $\lceil \log N \rceil = 14$ 에 대해, 이분 탐색 횟수가 10회 이상이 될 확률은 10^{-4} 이하입니다.
- ✓ 따라서 200번 이내의 마력 측정으로 문제를 해결할 수 있으며, 이외에도 다양한 방법이 존재합니다.

L. 레몬 박물관

필요한 개념: segment-tree

예상 난이도: **Platinum I**

- ✓ 제출 140회, 정답 19명
- ✓ 처음 푼 사람: urd05, 69분
- ✓ 출제자: pk661

L. 레몬 박물관

부분문제 1 ($N, Q \leq 1\,000$)

- ✓ 일단 $r - l + 1 \geq 2$ 조건을 배제합니다.
- ✓ 최소 횟수를 찾기 위해 imos법으로 생각합니다.
- ✓ $B_i = A_i - A_{i-1}$ 로 정의합니다.
- ✓ $[L, R]$ 에 1을 빼는 것은 B_L 을 1 감소, B_{R+1} 을 1 증가와 같습니다.
- ✓ 최소 연산을 달성하려면, 매 시행마다 B_i 가 양수인 지점을 하나 감소, 음수인 지점을 하나 증가하게 하면 됩니다.
- ✓ 즉, $\max(0, B_i)$ 합이 최소 횟수가 됩니다.
- ✓ 또한, 위의 조건이 있더라도 가능하다면 위 횟수가 항상 최소임을 보일 수 있습니다.

L. 레몬 박물관

- ✓ 가능 여부를 판정합니다.
- ✓ "서로 합칠 수 있는 연산" 이 존재하면 합치는 것이 이득임을 알 수 있습니다.
- ✓ 즉, $[L, M]$ 에 더하고 $[M + 1, R]$ 에 더하는 두 연산은 합쳐서 $[L, R]$ 로 만드는 것이 이득입니다.
- ✓ 위의 imos 얘기에서, 조건을 고려하지 않았을 때의 최소 횟수보다 많은 연산을 쓰는 방법은, 항상 "합칠 수 있는 연산"이 존재함을 보일 수 있습니다.
- ✓ 즉, 조건이 존재하더라도 항상 최소횟수로 연산을 하는게 최선임이 보여집니다.
- ✓ 최소 횟수를 달성하는 방법 중, 다음과 같은 방법이 제일 최선임을 알 수 있습니다.
- ✓ $B_i > 0$ 인 가장 작은 i 를 L , $B_i < 0$ 인 가장 작은 i 를 R 이라 했을 때, $[L, R - 1]$ 에 연산하는 걸 반복.

L. 레몬 박물관

- ✓ 그 과정에서, 구간 길이가 1 인 연산이 사용되지만 체크하면 됩니다.
- ✓ 이를 스택 등을 이용해 빠르게 시뮬레이션하면 됩니다.
- ✓ $O(NQ)$ 에 풀 수 있습니다.
- ✓ 좀 더 나아가, 더욱 간단한 판정법을 얻습니다.
- ✓ 모든 i 에 대해 $A_i \leq A_{i-1} + A_{i+1}$ (단, $A_0 = A_{N+1} = 0$)
- ✓ 위 조건을 위배하면 불가능함은 자명합니다.
- ✓ 역은, 만약 어떤 연산열에서 $[i, i]$ 연산이 존재한다면, 항상 이를 양 옆중 하나와 합칠 수 있거나, 어떤 $[L, R]$ 연산이 존재해 $[L, i], [i, R]$ 연산 2개로 바꿀 수 있음을 보일 수 있습니다. ($L < i < R$)
- ✓ 단순히 위 알고리즘에서 조건을 위배한 연산이 있을 경우를 식정리를 통해 구해도 됩니다.

L. 레몬 박물관

만점 풀이

- ✓ 위의 간단한 판정법을 세그먼트 트리를 이용해 관리하면 됩니다.
- ✓ $O((N + Q) \log N)$ 에 풀 수 있습니다.