월간 향유회 2023. 12.

공식 해설

문제		의도한 난이도	출제자
A	민주주의	**Bronze**	kiwiyou
B	교환 분배법칙	**Silver**	heeda0528
C	주식 시뮬레이션	**Silver**	utilforever
D	그래프 변환	**Gold**	moonlit
E	언젠가 정렬이 될 수 있으면 좋겠네.	**Gold**	heeda0528
F	Finding Array Tutorial	**Gold**	rustiebeats
G	귤나무	**Platinum**	pjshwa, heeda0528
H	Road To The LegenD,	**Platinum**	kiwiyou
I	C.S.G.	**Platinum**	snrnsidy
J	역삼각형	**Platinum**	bnb2011

A. 민주주의

`string, implementation`

출제진 의도 — **Bronze**

- 제출 413번, 정답 331명 (정답률 80.63%)
- 처음 푼 참가자: **riroan**, 0분
- 출제자: `kiwiyou`

A. 민주주의

- 찬성의 수가 $\frac{M+1}{2}$표 이상인 문제를 세면 됩니다.
- 문제마다 주어진 기록을 보고 0의 개수를 세면 찬성의 수를 알 수 있습니다.
- 찬성의 수가 $\frac{M+1}{2}$ 이상일 때마다 답에 1을 더해주면 답을 구할 수 있습니다.
- 한 줄을 읽고 0의 개수를 세는 데 $\mathcal{O}(M)$ 시간이 걸리므로 전체 시간복잡도 $\mathcal{O}(NM)$에 문제를 풀 수 있습니다.

B. 교환 분배법칙

`math, brute_force`

출제진 의도 — **Silver**

- 제출 437번, 정답 189명 (정답률 43.478%)
- 처음 푼 참가자: **ainta**, 3분
- 출제자: `heeda0528`

B. 교환 분배법칙

- 식을 정리해봅시다.
- $a + (b \times c) = (a + b) \times (a + c).$
- $a + bc = a^2 + ab + ac + bc.$
- $a = a^2 + ab + ac.$
- $a(a + b + c - 1) = 0.$

B. 교환 분배법칙

- 따라서 $a = 0$ 이거나 $a + b + c = 1$ 이라면 식이 성립합니다.
- $a = 0$ 일 때는 모든 b, c에 대해 성립하므로 가능한 쌍이 $(2N + 1)^2$개입니다.
- $a + b + c = 1$ 일 때는 아무 두 변수의 값을 고정하고 식을 만족하는 다른 한 변수의 값이 범위 내에 들어오는지 확인하면서 이중 포문을 돌면 됩니다.
- 이중 포문 내에서 $a = 0$ 일 때의 예외 처리가 필요합니다.
- 전체 시간복잡도는 $\mathcal{O}(N^2)$입니다.

C. 주식 시뮬레이션

`implementation, simulation`

출제진 의도 — **Silver**

- 제출 524번, 정답 99명 (정답률 18.893%)
- 처음 푼 참가자: **mjhmjh1104**, 13분
- 출제자: `utilforever`

C. 주식 시뮬레이션

- 지문에서 설명한 대로 구현하면 되는 문제입니다.
- 구현 과정에서 주의할 점
  - 계산 결과가 32비트 범위를 초과할 수 있습니다.
  - 5번 메뉴를 처리할 때 일의 자리에서 버림하는 수식을 잘 세워야 합니다.
- 전체 시간복잡도는 $\mathcal{O}(NQ)$입니다.

D. 그래프 변환

`dynamic_programming, math`

출제진 의도 — **Gold**

- 제출 343번, 정답 71명 (정답률 20.991%)
- 처음 푼 참가자: **cocoa_chan**, 12분
- 출제자: `moonlit`

D. 그래프 변환

- x_i를 그래프를 i번 변환했을 때의 정점 개수로 정의하겠습니다.
- y_i를 그래프를 i번 변환했을 때 한 정점이 가진 간선의 개수로 정의하겠습니다.
- 그래프를 변환하더라도 정점 간의 대칭이 유지되기 때문에 각 정점이 가진 간선의 개수는 항상 동일합니다.
- $x_0 = n$, $y_0 = n - 1$입니다.

D. 그래프 변환

- 정점이 x개인 그래프(G)에서 정점 하나에 y개의 간선을 가진 그래프가 가진 간선의 총 개수는 악수 정리에 의해 $\frac{xy}{2}$입니다.
- 모든 간선은 새로운 그래프(G')의 정점이 되므로 $x_{i+1} = \frac{x_i y_i}{2}$입니다.
- 계산 과정에서 $\frac{1}{2} \equiv 5 \cdot 10^8 + 4 \pmod{10^9 + 7}$임을 이용할 수 있습니다.

D. 그래프 변환

- 한 정점이 y개의 간선을 가졌을 때, 특정 간선을 기준으로 보면 양쪽에 $y-1$개씩의 간선이 인접해 있습니다.
- 따라서 새로운 그래프(G')의 각 정점은 $2(y-1)$개의 간선을 가지므로 $y_{i+1} = 2(y_i - 1)$입니다.
- 위 정보를 바탕으로 x_k를 구하면 $\mathcal{O}(n)$의 시간복잡도로 문제를 해결할 수 있습니다.

E. 언젠가 정렬이 될 수 있으면 좋겠네.

`greedy`

출제진 의도 — **Gold**

- 제출 202번, 정답 48명 (정답률 24.752%)
- 처음 푼 참가자: **ainta**, 22분
- 출제자: `heeda0528`

E. 언젠가 정렬이 될 수 있으면 좋겠네.

- 사전 순 최소는 앞 원소부터 가능한 작은 수로 채우는 것이 최적입니다.
- 현재 사전 순 최소인 수열을 k번째까지 완성했고, 다음 원소를 결정할 차례라고 해봅시다.
- $T_i =$ 남은 수 중 A_i보다 앞에 있으면서 A_i와 서로소가 아닌 수의 개수 라고 정의하겠습니다.
- 처음에 T 배열은 $\mathcal{O}(N^2 \log X)$에 구성할 수 있습니다.

E. 언젠가 정렬이 될 수 있으면 좋겠네.

- 자신보다 앞에 있는 수 중 자신과 서로소가 아닌 것이 있다면 자신은 맨 앞으로 이동할 수 없습니다.
- 따라서 남은 수 중 다음 원소가 될 수 있는 것은 $T_i = 0$ 인 A_i입니다.
- 이러한 A_i들 중 가장 작은 수를 다음 원소로 결정하는 것이 최적입니다.

E. 언젠가 정렬이 될 수 있으면 좋겠네.

- A_p를 다음 원소로 결정했다면, $p < q$ 이고 A_p와 A_q가 서로소인 모든 q에 대해 T_q를 1 감소시킬 수 있습니다.
- 이를 총 N번 반복하면 사전 순으로 최소인 수열을 얻을 수 있습니다.
- 전체 시간복잡도는 $\mathcal{O}\left(N^2 \log X\right)$입니다.

E. 언젠가 정렬이 될 수 있으면 좋겠네.

- 삽입 정렬과 유사하게 진행하는 풀이도 있습니다.
- 수열 A의 앞 $k-1$개의 원소들만으로 사전 순 최소를 만들었고, A_k를 적절한 위치에 끼워넣는 상황을 생각해봅시다.
- A_k가 A_i보다 작으면서 그 사이의 모든 수가 A_k와 서로소라면 A_k를 A_i 자리에 삽입할 수 있고, 이러한 위치 중 가장 앞을 택하는 것이 최적입니다.
- 이를 총 N번 반복하면 사전 순으로 최소인 수열을 얻을 수 있습니다.
- 위의 증명은 독자의 몫으로 남깁니다.

F. Finding Array Tutorial

`interactive, prefix_sum`

출제진 의도 — **Gold**

- 제출 110번, 정답 38명 (정답률 34.545%)
- 처음 푼 참가자: **ainta**, 26분
- 출제자: `rustiebeats`

F. Finding Array Tutorial

- 원본 수열 A에서 A_i가 고유한 원소이기 위해서는 부분 배열 $[A_1, A_2, \cdots, A_{i-1}]$과 $[A_{i+1}, \cdots, A_{N-1}, A_N]$에 A_i와 같은 값이 존재하지 않아야 합니다.
- 배열 p_i에 "? 1 i" 질문에 대한 답을 저장해 둡시다.
- 배열 s_i에 "? i N" 질문에 대한 답을 저장해 둡시다.

F. Finding Array Tutorial

- A_i가 고유한 원소이기 위한 필요충분조건은 $p_i = p_{i-1} + 1$과 $s_i = s_{i+1} + 1$을 동시에 만족하는 것입니다.
- 따라서 총 $2N$개의 질문으로 문제를 해결할 수 있습니다.

G. 귤나무

`simulation, ad_hoc`

출제진 의도 — **Platinum**

- 제출 123번, 정답 44명 (정답률 36.585%)
- 처음 푼 참가자: **ainta**, 29분
- 출제자: `pjshwa, heeda0528`

G. 귤나무

- 현재 귤나무에 남은 귤의 수를 M이라고 하겠습니다.
- M이 A의 최솟값보다 작아지기 전까지는 적어도 한 마리의 곰곰이가 귤을 딸 수 있습니다.
- 따라서 문제를 다음과 같이 표현할 수 있습니다.
  - $a := 0$ 으로 두고, A를 순회하며 $a + A_i \leq M$ 인 경우 A_i를 a에 더합니다.
  - M을 a만큼 감소시킵니다.
  - 이를 $M < \min(A)$이 될 때까지 반복합니다.

G. 귤나무

- 첫 과정에서 순회를 마친 후 a가 결정된 상황을 생각해봅시다.
- 이제 M에서 a를 빼고 다시 a를 구해야 하는데, 사실 M이 a보다 작아지기 전까지는 a의 값이 변하지 않습니다.
- M이 a 이상이라면 기존에 더해진 A_i는 항상 더해지고, M은 계속 감소하기 때문에 새로운 A_i가 더해질 일이 없기 때문입니다.
- 따라서 M을 a만큼 감소시키는 대신, M에 $M \bmod a$를 할당해도 됩니다.

G. 귤나무

- M 이하의 수 a로 M을 나눈 값의 나머지는 항상 $\left\lfloor \frac{M}{2} \right\rfloor$ 이하이기 때문에, 위 시행은 최대 $\log M$번만 이루어집니다.
- 따라서 전체 시간복잡도 $\mathcal{O}(N \log M)$ 에 문제를 풀 수 있습니다.

H. Road To The LegenD,

`dijkstra, coordinate_compression, segtree, lazy_prop`

출제진 의도 — **Platinum**

- 제출 43번, 정답 13명 (정답률 32.558%)
- 처음 푼 참가자: **ainta**, 67분
- 출제자: `kiwiyou`

H. Road To The LegenD,

- 1번 마을부터 각 마을까지의 최단거리 중 가장 긴 것을 찾아야 합니다.
- 가중치가 0 이상인 방향 그래프에서의 단일 출발점 최단경로 문제이므로 데이크스트라 알고리즘을 떠올릴 수 있습니다.
- 다만 고행의 길로 인해 간선이 최대 $\mathcal{O}(N^2)$개 존재하므로, 미리 간선을 만든다면 시간 혹은 메모리가 부족합니다.
- 그래프의 특성을 이용하거나 자료 구조를 활용하여 시간과 메모리를 개선할 수 있습니다.

H. Road To The LegenD,

- $h_v > H_u$를 만족하는 h_v가 존재하면 이를 h'_u라 하겠습니다.
- 한 마을에서 출발하는 고행의 길이 연결하는 마을들의 격은 항상 구간 $[h'_u, \max h]$ 꼴로 나타납니다.
- 이때 h'_u가 단조증가한다는 성질을 이용할 수 있습니다.
- 각 h_u에 대응하는 정점 $t(h_u)$과, 모든 u에 대해 간선 $(u, t(h'_u), p_u)$와 $(t(h_u), u, 0)$을 만듭니다.
- h'_u가 단조증가하므로, $h'_u < h'_v$이면서 가장 작은 h'_v를 구해 간선 $(h'_u, h'_v, 0)$을 만들면 모든 고행의 길을 $\mathcal{O}(N)$개의 간선으로 모델링할 수 있습니다.

H. Road To The LegenD,

- 그대로 데이크스트라 알고리즘을 사용하면 $\mathcal{O}((N+M)\log(N+M))$에 문제를 해결할 수 있습니다.
- 특수한 모델링 없이, 자료구조를 활용한 방법도 소개합니다.
- 데이크스트라 알고리즘을 진행하면서 각 간선을 한 번에 방문하는 방법을 찾아야 합니다.

- 데이크스트라에서 정점들까지의 거리를 업데이트할 때, 특정 위치가 아닌 구간을 업데이트할 수 있으면 됩니다.
- 우선순위 큐의 용도로 느리게 갱신되는 세그먼트 트리를 사용하면 구간 업데이트 연산을 지원할 수 있습니다.

H. Road To The LegenD,

- 모든 편한 길을 최대 한 번 방문하여 세그먼트 트리를 업데이트하고, 각 정점 방문 시에 구간 업데이트가 한 번 일어나므로 $\mathcal{O}((N+M)\log N)$에 문제를 풀 수 있습니다.
- 단, 이미 최단경로가 확정된 정점이 구간 업데이트의 영향을 받게 된다면 각 정점을 최대 $\mathcal{O}(N)$번 방문하게 되어 전체 시간복잡도가 $\mathcal{O}\big(\big(N^2+M\big)\log N\big)$이 됨에 주의해야 합니다.
- 이를 막기 위해 더 이상 업데이트하지 않음을 나타내는, 무한대를 나타내는 값보다 작은 값을 도입할 수 있습니다.

I. C.S.G.

`sprague_grundy, game_theory, dynamic_programming`
출제진 의도 — **Platinum**

- 제출 49번, 정답 6명 (정답률 14.286%)
- 처음 푼 참가자: **kes0716**, 132분
- 출제자: `snrnsidy`

- 이 문제를 풀기 위해서는 게임 이론 dp를 이용하는 방법이 있고, 스프라그 그런디 정리를 이용하는 방법이 있습니다. 여기서는 후자의 방법으로 설명하겠습니다.
- 이전 차례까지 선택한 카드에 적힌 수들의 소인수 집합을 S라고 할 때, 현재 차례에서 선택할 수 있는 카드는 아래 조건을 만족해야 합니다.
- 카드에 적힌 수 x의 소인수들의 집합 S_x와 S의 교집합이 공집합이여야 합니다.

- 위에서 말한 집합을 bitmask로 표현할 수 있으며, bitmask에 대한 grundy 수를 구해주면 됩니다.
- 이는, 위에서 말한 것처럼 $\mathrm{bitmask}\ \&\ \mathrm{mask}[x] = 0$을 만족하는 x를 선택했을 때의 만들어지는 상태 $\mathrm{bitmask} \mid \mathrm{mask}[x]$의 grundy 수 집합의 mex 값이 됩니다.

- 다만, N 이하의 소수를 전부 고려하는 경우, bitmask로 표현하는 것은 어렵기 때문에 $\frac{N}{2}$ 이하의 소수만 고려하면 됩니다.
- 이렇게 할 수 있는 이유는 $\frac{N}{2}$ 초과의 소수를 소인수로 갖는 수는 해당 소수가 유일하기 때문입니다.
- 그리고 $\frac{N}{2}$ 초과의 소수에 대해, 해당 수가 여러개가 등장하더라도 1 하나로 처리하면 됩니다.

- 그리고, 현재까지 선택한 카드 중에서의 1 개수의 홀짝성을 추가적으로 따져줘야 할 필요가 있습니다. 이를, parity라고 하겠습니다.
- 따라서, 위에서는 grundy 수를 구할 때 현재 상태 하나만 고려했다면 여기에 parity도 고려하여 grundy 수를 구해주면 됩니다.
- 따라서, 초기 상태로부터 grundy 수를 탑다운 방식으로 구하고 해당 값이 0이라면 bnb2011을 출력하고, 0이 아니면 amsminn을 출력하면 됩니다.

J. 역삼각형

`segtree, sweeping, math`

출제진 의도 — **Platinum**

- 제출 27번, 정답 11명 (정답률 48.148%)
- 처음 푼 참가자: **ainta**, 86분
- 출제자: `bnb2011`

J. 역삼각형

- 당연하지만, 조건을 만족하는 역삼각형의 개수는 $\mathcal{O}(N^3)$개이기 때문에 모든 a, b, c를 순회하며 조건을 만족하는지 검사하는 것은 불가능합니다.
- 역삼각형을 구성하는 세 점 a, b, c 중, b에 해당하는 점을 고정해 봅시다. a, b, c는 문제 본문에서 제시된 조건을 모두 만족하는 점들입니다.

J. 역삼각형

- 우선 역삼각형 하나의 넓이를 계산해봅시다. a와 b의 x 좌표 차를 p, b와 c의 x 좌표 차를 q라고 하고, 유사하게 y 좌표의 차도 r, s로 정의하겠습니다.
- b를 지나고 x축에 평행한 직선과 a, c를 각각 지나고 y축에 평행한 직선을 생각해봅시다. 이 직선들과 a, c를 잇는 선분은 사다리꼴을 이룹니다.
- 역삼각형의 넓이는 사다리꼴에서 양쪽의 직각삼각형 넓이를 뺀 것과 같습니다.
- 계산의 편의상 모든 넓이에 2를 곱한 것으로 생각합시다. 사다리꼴의 넓이는 $(p+q)(r+s)$ 이고, 직각삼각형의 넓이는 각각 pr, qs 이므로 역삼각형의 넓이는 $ps+qr$ 이 됩니다.

- b를 고정했으므로 가능한 a와 c에 해당하는 점을 고르는 것은 독립적입니다.
- a로 가능한 점의 개수를 L, c로 가능한 점의 개수를 R 이라고 합시다. 만들어질 수 있는 모든 역삼각형의 넓이의 합은 다음과 같습니다.
- $$\sum_{i=1}^{L}\sum_{j=1}^{R}\left(p_i s_j + q_j r_i\right)$$
  $$= \left(\sum_{i=1}^{L} p_i\right)\left(\sum_{j=1}^{R} s_h\right) + \left(\sum_{j=1}^{R} q_j\right)\left(\sum_{i=1}^{L} r_i\right)$$

- 결론적으로, 모든 점에 대해 아래 네 개의 값을 알고 있으면 총 $\mathcal{O}(N)$ 시간에 모든 역삼각형의 넓이 합을 구할 수 있습니다.

  1. 현재 점보다 왼쪽 위에 있는 모든 점과 x 좌표 차의 합
  2. 현재 점보다 왼쪽 위에 있는 모든 점과 y 좌표 차의 합
  3. 현재 점보다 오른쪽 위에 있는 모든 점과 x 좌표 차의 합
  4. 현재 점보다 오른쪽 위에 있는 모든 점과 y 좌표 차의 합

J. 역삼각형

- 네 개의 값 모두 스위핑과 세그먼트 트리를 이용해 전처리할 수 있습니다.
- 점들을 y 좌표 내림차순으로 정렬한 뒤 계산해주면 됩니다. 이때, 같은 y 좌표를 가지는 점들은 계산 결과에 포함하지 않도록 유의해야 합니다.
- 총 시간복잡도는 $\mathcal{O}(N \log N)$ 입니다.