아니메컵 OVA 풀이

Official Editorial

by

아니메컵 제작위원회

2023년 8월 26일

문제		의도한 난이도	출제자
A	하늘의 타일링	Impossible	altarimoo
B	나노의 머리 속 시곗바늘	Hard	bubbler
C	치노의 라떼 아트 만들기	Hard	bubbler
D	THE iDEM@STER (M@STER VERSION)	Challenging	sait2000
E	소수 세기	Hard	index
F	하이퍼 주사위 굴리기	Hard	index

OVA-A. 하늘의 타일링

`linear-algebra`, `exponentiation-by-squaring`, `combinatorics`
출제진 의도 – **Impossible**

- 제출 7번, 정답 0명 (정답률 0.000%)
- 처음 푼 사람: -
- 출제자: `altarimoo`

- 위키피디아$_W$ 링크는 이와 같이 밑줄이 그어져있습니다.

- 각기둥 모양 격자의 각 칸을 정점으로 하고, 인접한 칸 사이를 연결하는 간선이 있는 무방향 그래프를 생각해봅시다.
- 도미노를 놓는 것을 해당 그래프에서 간선을 선택하는 것으로 생각하면, 타일링은 그래프에서의 완전 매칭(perfect matching)과 대응함을 알 수 있습니다.
- 따라서 문제는 특정 그래프에서 완전 매칭의 수를 세는 문제와 동치입니다.
- 그래프 G의 완전 매칭의 수를 $\mathrm{Pm}(G)$로 표기하도록 하겠습니다.

- 주어진 그래프를 기술하기 위한 용어를 정의합니다.
- C_n 를 정점의 수가 n 개인 사이클이라 정의합니다.
- P_n 를 정점의 수가 n 개인 일직선 형태의 그래프로 정의합니다.
- 두 그래프 G_1, G_2 에 대해서 두 그래프의 Cartesian product$_W$ 를 $G_1 \square G_2$ 로 표기하고, 다음과 같이 정의합니다.
  - $G_1 \square G_2$ 의 정점의 집합은 $V(G_1) \times V(G_2)$ 가 됩니다.
  - 두 정점 (u_1, u_2) 와 (v_1, v_2) 가 인접할 필요충분조건은, G_1 에서 u_1 과 v_1 이 인접하거나 G_2 에서 u_2 와 v_2 가 인접한 것입니다.

- 주어진 그래프는 $C_M \square P_N$ 으로 표현할 수 있습니다.
  - 참고로 이 형태의 그래프를 stacked prism graph로 부르기도 한다고 합니다.
- cf) https://mathworld.wolfram.com/StackedPrismGraph.html

- 어떤 무방향 단순 그래프에 대해서 각 간선에 대해 방향을 부여한 그래프를 그래프의 orientation$_W$이라고 부릅니다.
  - 즉, 원래 그래프에 방향 없는 간선 (u, v)가 있었다면, orientation에는 (u, v) 또는 (v, u) 중 하나가 있게 됩니다.

- 그래프 G의 orientation G^e가 주어졌을 때 각 정점이 1 이상 n 이하의 정수로 인덱스가 붙어있다고 하면 skew-adjacency matrix $A(G^e)$를 다음과 같이 정의합니다.

$$A(G^e) = (a_{ij})_{n\times n},$$
$$a_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E(G^e), \\ -1 & \text{if } (v_j, v_i) \in E(G^e), \\ 0 & \text{otherwise.} \end{cases}$$

- 즉 $A(G^e)$는 G의 인접 행렬에서 각 간선에 대해 orientation과 같은 방향은 그대로 1로 두고, 반대방향을 -1로 바꾼 형태입니다.

- 아래의 lemma가 알려져있습니다.
- Lemma 1. 정점의 수가 짝수인 무방향 단순 그래프 G가 주어졌을 때, G의 임의의 orientation G^e에 대해 다음이 성립합니다.

$$\det A(G^e) \leq \mathrm{Pm}(G)^2$$

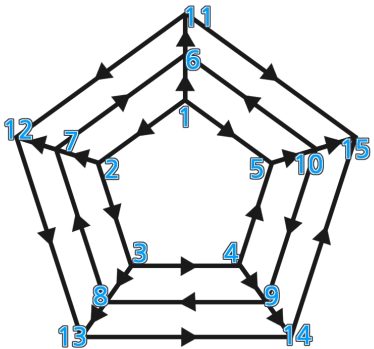
  - 이때, 등호는 G^e가 Pfaffian orientation$_W$일 때 성립합니다.
- Lemma의 증명과 Pfaffian orientation의 정의는 생략합니다.
- cf) https://doi.org/10.1016/j.laa.2012.01.019

- 정점의 수가 홀수인 경우에는 $\mathrm{Pm}(G) = 0$ 이므로, 이 뒤부터는 정점의 수가 짝수인 경우, 즉 $N \bmod 2 = 0$ 또는 $M \bmod 2 = 0$ 이 성립하는 경우만 다루겠습니다.
- Lemma 1 에서부터 다음과 같은 작전을 세울 수 있습니다.
  1. $C_M \times P_N$ 의 어떤 Pfaffian orientation $(C_M \times P_N)^e$ 하나를 구성한다.
  2. $\det\left(A\left((C_M \times P_N)^e\right)\right)$ 를 계산한다.

- Pfaffian orientation을 구성하기 위해서 FKT algorithm$_W$을 사용합니다.
- $C_M \times P_N$은 N 개의 동심원에 대해서 각 원의 원주에 M 개의 점을 찍은 다음 원 위에서 인접한 점 사이에 간선을 잇고, 이웃한 두 원 사이의 점들을 순서대로 짝을 지어 간선을 잇는 방식으로 평면 위에 그릴 수 있습니다. 즉, 이 그래프는 평면 그래프입니다.

- 가장 작은 동심원의 정점들을 반시계방향으로 1 부터 M 까지 번호를 매깁니다.
- 그 다음으로 작은 동심원에서 1 에 연결된 정점부터 반시계방향으로 $M+1$ 부터 $2M$ 까지 번호를 매깁니다.
- 마찬가지 방법으로 모든 정점에 번호를 매깁니다.
- 이제 가장 작은 동심원에서 $(M, 1)$ 을 제외한 $M-1$ 개의 간선들과, 서로 다른 동심원을 연결하는 간선들을 모두 고릅니다. 그러면 고른 간선들이 $C_M \times P_N$ 의 스패닝 트리가 됨을 알 수 있습니다.
- 이제 고른 간선들 모두에 정점 번호가 커지는 방향으로 방향을 정합니다. 즉 가장 작은 동심원에서 $(M, 1)$ 을 제외한 모든 간선은 반시계방향으로, 서로 다른 동심원을 연결하는 간선들은 바깥방향으로 방향을 정합니다.

- 이제 모든 면들에 대해 시계방향 간선이 홀수 개가 되도록 해야 하므로, $(M, 1)$을 1에서 M 방향으로 방향을 준 다음, 다른 동심원들의 간선은 바로 안 동심원과 반대 방향으로 방향을 주면 됩니다.



– 이때 가장 작은 동심원의 정점들에 대해서 skew-adjacency matrix $A_C = A({C_M}^e)$ 를 구해보면 다음과 같은 형태가 됩니다.

$$A_C = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 & 1 \\ -1 & 0 & 1 & \cdots & 0 & 0 \\ 0 & -1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ -1 & 0 & 0 & \cdots & -1 & 0 \end{pmatrix}$$

– 전체 skew-adjacency matrix $A\left((C_M \square P_N)^e\right)$를 구해보면 다음과 같은 형태가 됩니다.

$$A\left((C_M \square P_N)^e\right) = \begin{pmatrix} A_C & I_M & O & O & \cdots \\ -I_M & -A_C & I_M & O & \cdots \\ O & -I_M & A_C & I_M & \cdots \\ O & O & -I_M & -A_C & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

- 이때 $\det\left(A\left((C_M \times P_N)^e\right)\right)$는 다음과 같습니다.

$$\begin{vmatrix} A_C & I_M & O & O & \cdots \\ -I_M & -A_C & I_M & O & \cdots \\ O & -I_M & A_C & I_M & \cdots \\ O & O & -I_M & -A_C & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{vmatrix} = (-1)^{M\lfloor N/2 \rfloor} \begin{vmatrix} A_C & I_M & O & O & \cdots \\ I_M & A_C & -I_M & O & \cdots \\ O & -I_M & A_C & I_M & \cdots \\ O & O & I_M & A_C & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{vmatrix}$$

- 이때 전 슬라이드의 마지막 행렬은 Kronecker product$_W$를 사용해서 다음과 같이 표현할 수 있습니다.

$$\begin{pmatrix} A_C & I_M & O & O & \cdots \\ I_M & A_C & -I_M & O & \cdots \\ O & -I_M & A_C & I_M & \cdots \\ O & O & I_M & A_C & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} = B_P \otimes I_M + I_N \otimes A_C$$

where

$$\begin{pmatrix} 0 & 1 & 0 & 0 & \cdots \\ 1 & 0 & -1 & 0 & \cdots \\ 0 & -1 & 0 & 1 & \cdots \\ 0 & 0 & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} = B_P$$

- $B_P \otimes I_M + I_N \otimes A_C$ 을 복소수 계수 행렬로 생각하면, 해당 행렬의 고유값을 모두 곱하면 행렬값을 구할 수 있습니다.
- Lemma 2. A_C 의 고유값을 중복을 포함하여 $\alpha_1, \alpha_2, \cdots, \alpha_M$ 이라고 합시다. 또, B_P 의 고유값을 중복을 포함하여 $\beta_1, \beta_2, \cdots, \beta_N$ 이라고 합시다.
  그러면 $B_P \otimes I_M + I_N \otimes A_C$ 의 고유값은 모두 $\alpha_i + \beta_j$ 의 형태입니다. 또, 다음이 성립합니다.

$$\det\left(B_P \otimes I_M + I_N \otimes A_C\right) = \prod_{i=1}^{M} \prod_{j=1}^{N} \left(\alpha_i + \beta_j\right)$$

- Proof: 복소수 계수 행렬은 언제나 삼각화할 수 있음이 알려져 있으므로, A_C 와 B_P 각각을 삼각화해봅시다.
- $P^{-1}A_CP$ 와 $Q^{-1}B_PQ$ 가 상삼각행렬인 P 와 Q 를 생각합시다. 이때 다음이 성립합니다.

$$\begin{aligned}
&(Q\otimes P)^{-1}\left(B_P\otimes I_M + I_N\otimes A_C\right)(Q\otimes P)\\
&= \left(Q^{-1}\otimes P^{-1}\right)\left(B_P\otimes I_M + I_N\otimes A_C\right)(Q\otimes P)\\
&= \left(Q^{-1}B_PQ\otimes P^{-1}I_MP + Q^{-1}I_NQ\otimes P^{-1}A_CP\right)\\
&= \left(Q^{-1}B_PQ\otimes I_M + I_N\otimes P^{-1}A_CP\right)
\end{aligned}$$

- $Q^{-1}B_PQ\otimes I_M + I_N\otimes P^{-1}A_CP$ 는 상삼각행렬이므로, $Q\otimes P$ 가 $B_P\otimes I_M + I_N\otimes A_C$ 를 삼각화함을 알 수 있습니다.

- 삼각화 전후의 행렬은 고유값 및 행렬값이 같으며, 삼각화한 후 행렬의 대각성분들이 고유값이 됩니다.
- 일반성을 잃지 않고 $P^{-1}A_CP$의 대각성분들을 순서대로 $\alpha_1, \alpha_2, \cdots, \alpha_M$라고 하고, $Q^{-1}B_PQ$의 대각성분들을 순서대로 $\beta_1, \beta_2, \cdots, \beta_N$라고 합시다.
- 그러면 $Q^{-1}B_PQ \otimes I_M + I_N \otimes P^{-1}A_CP$의 대각성분은 순서대로 다음과 같습니다.

$$\begin{gathered}\alpha_1+\beta_1, \alpha_2+\beta_1, \cdots, \alpha_M+\beta_1, \\ \alpha_1+\beta_2, \alpha_2+\beta_2, \cdots, \alpha_M+\beta_2, \\ \vdots \\ \alpha_1+\beta_N, \alpha_2+\beta_N, \cdots, \alpha_M+\beta_N\end{gathered}$$

- 앞에서 구한 대각성분을 모두 곱하면 $B_P \otimes I_M + I_N \otimes A_C$ 의 행렬값을 구할 수 있으므로,

  $\det\left(B_P \otimes I_M + I_N \otimes A_C\right) = \prod_{i=1}^{M}\prod_{j=1}^{N}\left(\alpha_i + \beta_j\right)$ 이 성립합니다. □

- Lemma 2의 우변을 계산하기 위해서 $\underline{\text{resultant}_W}$를 사용합니다.
- 두 0이 아닌 다항식 $f = a_0x^{n_f} + a_1x^{n_f-1} + \cdots + a_{n_f}$와 $g = b_0x^{n_g} + b_1x^{n_g-1} + \cdots + b_{n_g}$에 대해서, resultant $\text{res}(f, g)$를 다음 슬라이드와 같은 행렬식으로 정의합니다.
  - 행렬은 $(n_f + n_g) \times (n_f + n_g)$ 형태로, 앞 n_g열이 f의 계수, 뒤 n_f열이 g의 계수가 됩니다.
  - 단, $n_f = n_g = 0$일 경우 $\text{res}(f, g) = 1$로 정의합니다.

$$\operatorname{res}(f,g) = \begin{vmatrix} a_0 & 0 & \cdots & 0 & b_0 & 0 & \cdots & 0 \\ a_1 & a_0 & \cdots & 0 & b_1 & b_0 & \cdots & 0 \\ a_2 & a_1 & \ddots & 0 & b_2 & b_1 & \ddots & 0 \\ \vdots & \vdots & \ddots & a_0 & \vdots & \vdots & \ddots & b_0 \\ a_{n_f} & a_{n_f-1} & \cdots & \vdots & b_{n_g} & b_{n_g-1} & \cdots & \vdots \\ 0 & a_{n_f} & \ddots & \vdots & 0 & b_{n_g} & \ddots & \vdots \\ \vdots & \vdots & \ddots & a_{n_f-1} & \vdots & \vdots & \ddots & b_{n_g-1} \\ 0 & 0 & \cdots & a_{n_f} & 0 & 0 & \cdots & b_{n_g} \end{vmatrix}$$

- f와 g가 복소수 계수 다항식인 경우, $f = a_0 \prod_{i=1}^{n_f}(x - r_i), g = b_0 \prod_{i=1}^{n_g}(x - s_i)$와 같이 인수분해가 됩니다.
- 이때 다음 lemma가 알려져있습니다.
- Lemma 3. 다음이 성립합니다. 증명은 생략합니다.

$$\mathrm{res}(f, g) = a_0^{n_g} b_0^{n_f} \prod_{i=1}^{n_f} \prod_{j=1}^{n_g} (r_i - s_j)$$

- Lemma 2의 식과 lemma 3의 식을 비교하면, $\alpha_1, \alpha_2, \cdots, \alpha_M$ 을 근으로 갖는 다항식과, $-\beta_1, -\beta_2, \cdots, -\beta_N$ 을 근으로 갖는 다항식을 찾아서 resultant를 계산해야 함을 알 수 있습니다.
- 그러한 다항식은 특성다항식$_W$을 이용해서 찾을 수 있습니다.
- 정사각행렬 A에 대한 특성다항식을 $p_A(x) = \det(xI - A)$로 표기하겠습니다. 그러면 $p_{A_C}(x)$는 $\alpha_1, \alpha_2, \cdots, \alpha_M$를 근으로, $p_{-B_P}(x)$는 $-\beta_1, -\beta_2, \cdots, -\beta_N$를 근으로 가집니다.

- 정리하면 다음의 theorem이 성립합니다.
- Theorem 4. 다음이 성립합니다.

$$\det\left(B_P \otimes I_M + I_N \otimes A_C\right) = \operatorname{res}\left(p_{A_C}\left(x\right), p_{-B_P}\left(x\right)\right)$$

- 유도 과정에서 복소수 계수를 사용했지만, theorem 4의 식은 정수 계수만을 사용하는 식임을 확인할 수 있습니다. 따라서, 계산할 때 계수들을 P로 나눈 나머지를 통해서 계산할 수 있습니다.

- $p_{-B_P}(x)$는 N차 다항식이므로 문제의 범위에서 실제로 구할 수는 없습니다.
- resultant를 효율적으로 계산하기 위해 다음의 성질들을 사용합니다. 이 성질들은 행렬식의 성질로부터 유도할 수 있습니다. 이때 h는 어떤 다항식 q가 존재해 $h = g - qf \neq 0$과 $n_h = \deg h \leq n_g$를 만족하는 다항식입니다.
  - $\operatorname{res}(g, f) = (-1)^{n_f n_g} \operatorname{res}(f, g)$
  - $\operatorname{res}(f, h) = a_0^{n_g - n_h} \operatorname{res}(f, g)$
  - $n_f = 0$일 때 $\operatorname{res}(f, g) = a_0^{n_g}$
  - $n_f \geq 1$일 때 g가 f로 나누어떨어지면 $\operatorname{res}(f, g) = 0$

- 앞 슬라이드의 성질들을 사용하면 유클리드 호제법으로 차수를 줄여가면서 resultant를 계산할 수 있습니다.
- 특히, 처음부터 $p_{-B_P}(x)$ 대신에 $p_{-B_P}(x) \bmod p_{A_C}(x)$ 를 구해서 계산을 할 수 있습니다.

- 먼저 $p_{-B_P}(x)$를 계산합니다. B_P는 1과 -1이 번갈아 나오기 때문에 행렬 크기에 수학적 귀납법을 적용하기 어려운 형태이므로, 조금 더 일반적인 경우를 증명합니다.
- Lemma 5. $n \times n$ 형태의 행렬 A에 대해서 $1 \leq i \leq n-1$에 대해 $A_{i,i+1} = A_{i+1,i} \in \{1, -1\}$이고 나머지 값은 0이라고 합시다. 그러면 이런 조건을 만족하는 모든 행렬에 대해서 특성다항식 $p_A(x) = \det(xI - A)$가 같고, 그 다항식을 $f_n(x)$로 표기하면 $f_n(x)$에 대해 다음과 같은 점화식이 성립합니다.

$$f_1(x) = x$$
$$f_2(x) = x^2 - 1$$
$$f_{k+2}(x) = x f_{k+1}(x) - f_k(x)$$

- Proof. $n = 1$과 $n = 2$의 경우는 생략합니다.
- $n = k + 2$이라고 하고, n 미만의 모든 양의 정수에 대해 점화식이 성립한다고 합시다.
- 임의의 A에 대해 $xI - A$는 다음과 같은 형태입니다. 단 $s_i \cdots \in \{1, -1\}$이고, 빈 부분은 0입니다.

$$\begin{pmatrix} x & s_1 & & \\ s_1 & x & s_2 & \\ & s_2 & x & \ddots \\ & & \ddots & \ddots \end{pmatrix}$$

– 첫 행과 첫 열에 여인수 전개를 반복해서 적용하면 다음과 같이 정리할 수 있습니다.

$$\begin{vmatrix} x & s_1 & & \\ s_1 & x & s_2 & \\ & s_2 & x & \ddots \\ & & \ddots & \ddots \end{vmatrix} = x \begin{vmatrix} x & s_2 & & \\ s_2 & x & \ddots & \\ & \ddots & \ddots & \end{vmatrix} - s_1 \begin{vmatrix} s_1 & s_2 & & \\ & x & \ddots & \\ & \ddots & \ddots & \end{vmatrix}$$

$$= x f_{k+1}(x) - s_1^2 \begin{vmatrix} x & \ddots \\ \ddots & \ddots \end{vmatrix}$$

$$= x f_{k+1}(x) - s_1^2 f_k(x) = x f_{k+1}(x) - f_k(x)$$

- $\begin{pmatrix} f_{k+2}(x) \\ f_{k+1}(x) \end{pmatrix} = \begin{pmatrix} x & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} f_{k+1}(x) \\ f_k(x) \end{pmatrix}$ 가 성립하므로, $p_{-B_P}(x) = f_N(x)$ 를 분할 정복을 이용한 거듭제곱을 통해 구할 수 있습니다.

- 다음의 lemma도 lemma 5와 비슷한 방법으로 증명할 수 있습니다.
- Lemma 6. $n \times n$ 형태의 행렬 A에 대해서 $1 \leq i \leq n-1$에 대해 $A_{i,i+1} = -A_{i+1,i} \in \{1, -1\}$이고 나머지 값은 0이라고 합시다. 그러면 이런 조건을 만족하는 모든 행렬에 대해서 특성다항식 $p_A(x) = \det(xI - A)$가 같고, 그 다항식을 $g_n(x)$로 표기하면 $g_n(x)$에 대해 다음과 같은 점화식이 성립합니다.

$$g_1(x) = x$$
$$g_2(x) = x^2 + 1$$
$$g_{k+2}(x) = xg_{k+1}(x) + g_k(x)$$

- Theorem 7. $p_{A_C}(x) = xg_{M-1}(x) + 2g_{M-2}(x) + (-1)^M + 1$ 입니다.
- Proof. $xI - A_C$ 는 다음과 같은 형태입니다. 여기에 Theorem 5와 비슷하게 첫 행과 첫 열에 여인수 전개를 반복해서 적용합니다.

$$\begin{pmatrix} x & -1 & & & -1 \\ 1 & x & -1 & & \\ & 1 & x & \ddots & \\ & & \ddots & \ddots & -1 \\ 1 & & & 1 & x \end{pmatrix}$$

$$\begin{vmatrix} x & -1 & & & -1 \\ 1 & x & -1 & & \\ & 1 & x & \ddots & \\ & & \ddots & \ddots & -1 \\ 1 & & & 1 & x \end{vmatrix} = x \begin{vmatrix} x & -1 & & \\ 1 & x & \ddots & \\ & \ddots & \ddots & -1 \\ & & 1 & x \end{vmatrix} - (-1) \begin{vmatrix} 1 & -1 & & \\ & x & \ddots & \\ & \ddots & \ddots & -1 \\ 1 & & 1 & x \end{vmatrix}$$

$$+ (-1)^{M-1} \times (-1) \begin{vmatrix} 1 & x & -1 & \\ & 1 & x & \ddots \\ & & \ddots & \ddots \\ 1 & & & 1 \end{vmatrix}$$

$$
\begin{aligned}
&= xg_{M-1}(x) + \left(1 \times \begin{vmatrix} x & \ddots & \\ \ddots & \ddots & -1 \\ & 1 & x \end{vmatrix} + (-1)^{M-2} \times 1 \times \begin{vmatrix} -1 & & \\ x & \ddots & \\ \ddots & \ddots & -1 \end{vmatrix} \right) \\
&+ (-1)^{M} \left(1 \times \begin{vmatrix} 1 & x & \ddots \\ & \ddots & \ddots \\ & & 1 \end{vmatrix} + (-1)^{M-2} \times 1 \times \begin{vmatrix} x & -1 & \\ 1 & x & \ddots \\ & \ddots & \ddots \end{vmatrix} \right) \\
&= xg_{M-1}(x) + \left(g_{M-2}(x) + (-1)^{M-2} \times (-1)^{M-2} \right) \\
&+ (-1)^{M} \left(1 + (-1)^{M-2} g_{M-2}(x) \right) \\
&= xg_{M-1}(x) + 2g_{M-2}(x) + (-1)^{M} + 1
\end{aligned}
$$

□

- 구현은 다음과 같이 하면 됩니다. 모든 계산은 $\bmod P$ 로 진행합니다.
  1. $p_{A_C}(x)$ 를 구한다.
  2. $p_{-B_P}(x) \bmod p_{A_C}(x)$ 를 다항식을 원소로 하는 행렬의 거듭제곱을 통해 구한다.
  3. Resultant를 계산한다.
- 1번 과정은 M 의 제한이 작으므로 $\mathcal{O}\left(M^2\right)$ 에도 할 수 있고, 문제 범위에서 계수의 절대값이 10^9 이하이므로 미리 계산해서 하드코딩할 수도 있습니다.
- 2번 과정은 $\mathcal{O}\left(M^2 \log N\right)$ 에 할 수 있습니다.
- 3번 과정은 나눗셈을 위해 모듈로 역원을 구하는 것까지 고려하면 $\mathcal{O}\left(M^2 + M \log P\right)$ 에 할 수 있습니다.
- 따라서 시간복잡도는 $\mathcal{O}\left(M^2 \log N + M \log P\right)$ 입니다.

OVA-B. 나노의 머리 속 시곗바늘

`math, ad-hoc, case work`
출제진 의도 – **Hard**

- 제출 77번, 정답 6명 (정답률 7.792%)
- 처음 푼 사람: **yijw0930**, 50분
- 출제자: `bubbler`

- 먼저, 문제에서는 각 시계에 적힌 수를 $1..N$ 과 $1..M$ 으로 줬지만, $0..N-1$ 과 $0..M-1$ 로 써도 같은 문제가 되고, 이렇게 쓰면 K 초 후에 시계가 가리키는 값을 더 간단하게 표현할 수 있습니다.
- 시계 A 가 가리키는 값은 1 칸 이동할 때마다 $\bmod N$ 으로 1 감소하고, 시계 B 는 $\bmod M$ 으로 1 증가합니다. 따라서, K 칸 이동 후에는 각각 $(-K) \bmod N$ 과 $K \bmod M$ 를 가리키게 됩니다.
- 따라서 이 문제는 $K > 0$ 이면서 $(-K) \bmod N = K \bmod M$ 인 최소의 K 를 찾는 것과 같은 문제입니다.

- 어떤 K'에 대해 $(-K') \bmod N > K' \bmod M$이고 양변의 홀짝이 같다면, $K > K'$이면서 $(-K) \bmod N = K \bmod M$을 만족하는 최소의 K 값은 $K = K' + \frac{(-K') \bmod N - K' \bmod M}{2}$ 입니다. 이때 $K' \bmod M + \frac{(-K') \bmod N - K' \bmod M}{2} < M$이어야 합니다.
- 홀짝이 다르거나 M이 충분히 크지 않다면, 조건을 만족하는 K는 적어도 다음 N이나 M의 배수 이상입니다.
- 또한, $x \bmod N \leq N - 1$, $x \bmod M \leq M - 1$이므로, $(-K) \bmod N \leq M - 1$, $K \bmod M \leq N - 1$이어야 합니다.
- 위 관찰들을 종합하여 조심스럽게 케이스 분류를 하면, 모든 케이스에 대해 $\mathcal{O}(1)$ 시간에 답을 찾는 알고리즘을 얻을 수 있습니다.

- Case 1: N 이 짝수, $M > \frac{N}{2}$
  - $K' = 1$ 로 두면, $(-K') \bmod N = N - 1$ 과 $K' \bmod M = 1$ 의 홀짝이 같고 M 이 충분히 크므로 $K = \frac{N}{2}$ 에서 두 값이 일치하게 되며, 이것이 답이 됩니다.

- Case 2: $N < M$
  - N 이 짝수이면 Case 1 에 해당되므로 N 은 홀수입니다.
  - $0 < K \leq N$ 이면 $(-K) \bmod N = N - K$, $K \bmod M = K$ 의 홀짝이 다르므로 $K > N$ 이고, 이를 만족하면서 $K \bmod M \leq N - 1$ 이 되려면 한 바퀴를 돌아 다시 0 으로 만들어야 하므로 $K \geq M$ 이 됩니다.
  - $K' = M$ 일 때, $(-K') \bmod N = N - M \bmod N$, $K' \bmod M = 0$ 입니다.
  - $N - M \bmod N$ 이 짝수이면 $K = M + \frac{N - M \bmod N}{2}$ 가 답이 됩니다.
  - 홀수이면, K' 를 다음 N 의 배수인 $M + N - M \bmod N$ 으로 둡니다. 그러면 $K' \bmod M = N - M \bmod N$ 으로 홀수가 되어, $K = K' + \frac{M \bmod N}{2}$ 이 답이 됩니다.

- Case 3: $N > M$
  - 먼저 $(-K) \bmod N \leq M - 1$를 이용해 $K' = N - M$으로 둡니다. 이때 $(-K') \bmod N = M$, $K' \bmod M = N \bmod M$입니다.
  - 둘의 홀짝이 같으면 $K = \frac{M - N \bmod M}{2}$입니다.
  - 둘의 홀짝이 다르면, K'를 다음 M의 배수인 $N - N \bmod M$으로 둡니다. 그러면 $(-K') \bmod N = N \bmod M$, $K' \bmod M = 0$입니다. 여기서 M이 홀수이면 가정에 의해 $N \bmod M$이 짝수가 되어 $K = K' + \frac{N \bmod M}{2}$가 답입니다.

- Case 3: $N > M$ (cont.)
  - M 이 짝수이면, $N \bmod M$ 은 홀수이고, 따라서 N 도 홀수입니다. N 이 다음 바퀴에 진입하지 않으면 양쪽의 홀짝이 같을 수가 없으므로, N 을 한 바퀴 더 돌려서 $K' = 2N - M$ 으로 설정합니다.
  - 그러면 $(-K') \bmod N = M$, $K' \bmod M = 2N \bmod M$ 이 되고, 둘 다 짝수이므로 $K = K' + \frac{M - 2N \bmod M}{2}$ 이 답이 됩니다. □

- 이렇게 모든 경우를 직접 분석하지 않고, 앞의 관찰들을 루프에 넣어서 푸는 방법도 있습니다.
- 루프가 동작하는 최대 횟수는 조건 분기의 깊이의 최댓값과 같고, 따라서 이것도 역시 $\mathcal{O}(1)$에 동작합니다.
- 다만, 구현 방법에 따라 특정 케이스에서 무한 루프가 발생할 수 있으니, 작은 범위의 모든 입력에서 결과가 잘 출력되는지 확인하여야 합니다.
- 출제자의 풀이도 이 방식으로 동작하며, 구체적인 알고리즘은 다음 페이지에 있습니다. 이 알고리즘은 N과 M이 1보다 큰 홀수 공약수 g를 갖는 경우에 무한루프에 빠지게 되며, $\frac{N}{g}$과 $\frac{M}{g}$에 대한 답을 구하고 g를 다시 곱해주는 것으로 해결할 수 있습니다.

- 출제자의 알고리즘은 다음과 같습니다.
- 편의상 x 보다 크면서 y 의 배수인 가장 작은 수를 $f(x, y)$ 라고 합시다.

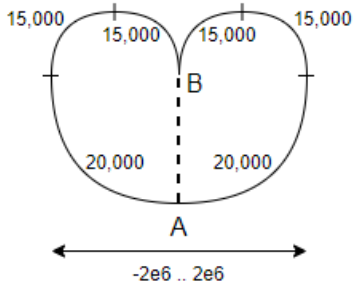
1. K 를 1로 초기화
2. $(-K) \bmod N = K \bmod M$ 이 될 때까지 다음을 반복:
   - 2.1 $(-K) \bmod N \geq M$ 이면, K 보다 크고 $(-K) \bmod N < M$ 인 최소의 K 값으로 업데이트. 이 값은 $f(K, N) - M + 1$ 입니다.
   - 2.2 아니고 $N \leq K \bmod M$ 이면, K 보다 크고 $K \bmod M = 0$ 인 최소의 K 값으로 업데이트. 이 값은 $f(K, M)$ 입니다.
   - 2.3 아니고 $(-K) \bmod N > K \bmod M$ 이면서 두 값의 홀짝이 같으면, 그 둘이 만나도록 K 를 업데이트하고 종료
   - 2.4 아니면 $\min(f(K-1, N), f(K-1, M)) + 1$ 로 업데이트

OVA-C. 치노의 라떼 아트 만들기

`number-theory`, `constructive`
출제진 의도 – **Hard**

- 제출 9번, 정답 4명 (정답률 44.444%)
- 처음 푼 사람: **yijw0930**, 99분
- 출제자: `bubbler`

- 모든 꼭짓점의 좌표가 정수로 제한되어 있으므로, 모든 변의 기울기는 유리수입니다.
- 하트를 다음 그림처럼 여러 개의 원호 비슷한 영역으로 나누고 하나의 영역에 N 개의 변을 사용한다고 하면, 서로 다른 양의 분수가 N 개 필요하고, 양 끝점의 x, y 좌표의 차이는 이 모든 분수를 기약분수로 나타냈을 때 분자와 분모 각각의 합이 됩니다.



- 따라서 이 문제는 분자와 분모의 합이 최소가 되도록 N 개의 분수를 만드는 문제가 되고, 이것은 여러 가지 방법으로 해결할 수 있습니다.
  - 분자와 분모의 합을 늘려가면서 분자와 분모가 서로소인 N 개의 분수를 선택
  - Stern-Brocot tree로 분수를 생성해 가면서 분자와 분모의 합에 대한 min priority queue를 사용하여 N 개의 분수를 추출
- 좌표 범위를 실제로 맞추려면 각각의 영역에 들어가는 변의 개수를 미세조정해야 할 수 있습니다.

OVA-D. THE iDEM@STER (M@STER VERSION)

`math`

출제진 의도 – **Challenging**

- 제출 1번, 정답 0명 (정답률 0.000%)
- 처음 푼 사람: -
- 출제자: `sait2000`

– 아니메컵 2쿨에 나왔던 BOJ 27457의 연장선인 문제입니다.

- 문제에서 주어진 문법을 만족하는 프로그램 S를 "올바른 프로그램"이라고 하고, 이 프로그램의 결과값을 $V(S)$라고 합시다.
- 그리고 $V(S) = N$인 S들 중에서 가장 짧은 것을 "최적의 프로그램"이라고 합시다. 어떤 N에 대해 최적의 프로그램은 여러 개일 수도 있습니다.
- 편의상 S의 부분 문자열이면서 주어진 문법을 만족하는 부분을 "부분 프로그램"이라고 하고, PPP $\cdots$ @@@ 에서 PPP와 @@@가 괄호 쌍일 때, 이 부분 프로그램을 "루프"라고 하겠습니다.

- 어떤 올바른 프로그램 S에 대해, S의 글자 순서를 뒤집고 `@`를 `P`로, `P`를 `@`로 치환한 프로그램을 S의 "반전", 또는 $inv(S)$라고 합시다. 그러면 다음이 성립합니다.
- Lemma 1. 프로그램 S의 반전 $inv(S)$에 대해 $V(inv(S)) = -V(S)$가 성립합니다.
- 증명은 다음 페이지에 있습니다.
- Lemma 1에 의해, 양의 정수 N에 대한 최적의 프로그램의 집합 $C = \{S_1, S_2, \cdots\}$를 알고 있다면, $-N$에 대한 최적의 프로그램의 집합은 C의 프로그램들의 반전의 집합 $inv(C) = \{inv(S_1), inv(S_2), \cdots\}$와 같습니다. 따라서 $|N|$에 대해서 C를 구한 다음, N이 양수이면 C에서, N이 음수라면 $inv(C)$에서 사전 순으로 빠른 프로그램을 고르면 됩니다.

– Proof. 모든 올바른 프로그램은 빈 문자열, @, P에서 시작해서, 두 올바른 프로그램을 이어 붙이거나 PPP$\cdots$@@@로 감싸는 방법으로 만들 수 있습니다. 각각의 경우에 대해 주어진 명제를 증명하면 구조적 귀납법에 의해 모든 올바른 프로그램에 대해 주어진 명제가 성립합니다.

- $S = \epsilon$: $V(S) = 0, inv(S) = \epsilon, V(inv(S)) = 0 = -V(S)$
- $S =$ P: $V(S) = 1, inv(S) =$ @, $V(inv(S)) = -1 = -V(S)$
- $S =$ @: $V(S) = -1, inv(S) =$ P, $V(inv(S)) = 1 = -V(S)$
- $S = S_1S_2$: $V(S) = V(S_1) + V(S_2), inv(S) = inv(S_2)inv(S_1)$, $V(inv(S)) = V(inv(S_2)) + V(inv(S_1)) = -V(S)$
- $S =$ PPPS_1@@@: $V(S) = 3V(S_1), inv(S) =$ PPP$inv(S_1)$@@@, $V(inv(S)) = 3V(inv(S_1)) = -V(S)$ □

- Lemma 2. 부분 프로그램 a, b, c, d, e에 대해, $S = a$PPPb@@@cPPPd@@@e는 최적의 프로그램이 아닙니다. (조건에 의해 b와 d는 각각 루프 안에 들어있는 형태입니다.)
- Proof. c가 빈 문자열이 아니라면, c는 반드시 P로 시작해서 @로 끝나고, e는 빈 문자열이거나 P로 시작하므로, $S_1 = a$PPPb@@@PPPd@@@ce는 올바른 프로그램이면서 $V(S) = V(S_1)$입니다. c가 빈 문자열이면 $S = S_1$입니다.
- 그리고 S_1에서 b는 빈 문자열이거나 P로 끝나고, d는 빈 문자열이거나 @로 시작하므로, 두 루프를 하나로 합친 $S_2 = a$PPPbd@@@ce 또한 올바른 프로그램이며, $V(S_2) = V(S)$이고, S_2는 S보다 길이가 6 짧습니다. 따라서 S는 최적의 프로그램이 아닙니다. □

- Lemma 2에 의해, 주어진 N에 대한 최적의 프로그램은 최상위 루프가 없거나 하나만 있을 수 있고, 루프가 있다면 $S = a\text{PPP}b@@@c$일 때 b도 재귀적으로 이 조건을 만족해야 함을 알 수 있습니다.
- 규칙에 의해 루프 내부의 부분 프로그램 b는 빈 문자열이거나, @로 시작해서 P로 끝나야 합니다. 이 제약사항이 있을 때의 최적의 프로그램은 해당 제약사항이 없을 때와 다를 수 있으므로, 두 가지를 따로 고려해야 합니다.
- 빈 문자열이거나 @로 시작해서 P로 끝나면서 결과값이 N인 프로그램의 최소 길이를 $in(N)$, 시작과 끝 글자에 제한이 없는 경우의 최소 길이를 $out(N)$ (지문에서는 $f(N)$)이라고 합시다.

- $in(0) = out(0) = 0$ 입니다.
- Lemma 1 에 의해, $in(-N) = in(N)$, $out(-N) = out(N)$ 이 성립합니다.
- 따라서 N 이 음수이면 $|N|$ 의 경우를 통해서 값을 알 수 있으므로, 이 뒤로는 N 이 양수인 경우만 고려합니다.

– 풀이의 흐름은 다음과 같습니다.

1. $in(N)$ 하나의 값을 구합니다.
2. $in(N)$ 의 연속합을 구합니다.
3. $out(N)$ 하나의 값을 구하고, $in(N)$ 을 통해 $out(N)$ 을 표현합니다.
4. $out(N)$ 의 연속합을 구합니다.

- 이제 $in(N)$ 하나의 값을 구해봅시다. (@로 시작해서 P로 끝나야 하는 경우)
- 편의상 부분 프로그램 a를 k번 반복한 프로그램을 $(a)_k$라고 쓰겠습니다.
- 루프가 없는 경우의 최적의 프로그램은 $(\texttt{@PP})_N$이고, 이때 길이는 $3N$입니다.
- 루프를 포함하는 경우의 최적의 프로그램은 $N \bmod 3$에 따라 달라집니다. 수식의 간결함을 위해, 여기서부터 $\frac{N}{3}$은 N을 3으로 나눈 몫, 즉 $\lfloor \frac{N}{3} \rfloor$을 의미합니다. 이때 프로그램의 길이는 루프 안의 결과값에 따라 결정됩니다.

– $N \bmod 3 = 0$인 경우

가능한 방법	$in(N)$ 후보
@PPP$(ins(\frac{N}{3}))$@@@P	$in(\frac{N}{3}) + 8$
@@P@@P@@PPP$(ins(\frac{N}{3}+1))$@@@P	$in(\frac{N}{3}+1) + 15$
$(@@P@@P@@P)_k$@@P@@P@@PPP$(ins(\frac{N}{3}+1+k))$@@@P	$in(\frac{N}{3}+1+k) + 15 + 9k$
@PPP$(ins(\frac{N}{3}-1))$@@@PP@PP@PP	$in(\frac{N}{3}-1) + 15$
@PPP$(ins(\frac{N}{3}-1-k))$@@@PP@PP@PP$(@PP@PP@PP)_k$	$in(\frac{N}{3}-1-k) + 15 + 9k$

– $N \bmod 3 = 1$ 인 경우

가능한 방법	$in(N)$ 후보
@PPP$(ins(\frac{N}{3}))$@@@PP	$in(\frac{N}{3}) + 9$
@PPP$(ins(\frac{N}{3} - k))$@@@PP$(@PP@PP@PP)_k$	$in(\frac{N}{3} - k) + 9 + 9k$
@@P@@PPP$(ins(\frac{N}{3} + 1))$@@@P	$in(\frac{N}{3} + 1) + 12$
$(@@P@@P@@P)_k$@@P@@PPP$(ins(\frac{N}{3} + 1 + k))$@@@P	$in(\frac{N}{3} + 1 + k) + 12 + 9k$

– $N \bmod 3 = 2$인 경우

가능한 방법	$in(N)$ 후보
@PPP$(ins(\frac{N}{3}))$@@@PP@PP	$in(\frac{N}{3}) + 12$
@PPP$(ins(\frac{N}{3} - k))$@@@PP@PP$(\text{@PP@PP@PP})_k$	$in(\frac{N}{3} - k) + 12 + 9k$
@@PPP$(ins(\frac{N}{3} + 1))$@@@P	$in(\frac{N}{3} + 1) + 9$
$(\text{@@P@@P@@P})_k$@@PPP$(ins(\frac{N}{3} + 1 + k))$@@@P	$in(\frac{N}{3} + 1 + k) + 9 + 9k$

- Theorem 3. 모든 $N \geq 1$에 대해, $in(N+1) - in(N) \in \{3, 1, -1\}$ 입니다.
- 어떤 미지의 값이 특정 집합의 원소 중 하나만 될 수 있을 때, 수식에서 편의상 이를 그 집합으로 표현하겠습니다. 예를 들어, 위 Theorem의 수식은 $in(N+1) - in(N) = \{3, 1, -1\}$, $in(N+1) = in(N) + \{3, 1, -1\}$로 쓸 수 있습니다.
- Theorem 3의 증명은 다음 페이지에서 시작합니다.

– Proof. (강한 귀납법) 먼저 $1 \leq N \leq 6$인 경우, 가능한 방법을 직접 구해 보면 다음과 같습니다.

N	가능한 방법	$in(N)$
1	@PP	3
2	@PP@PP	6
3	@PP@PP@PP	9
4	@PP@PP@PP@PP, @PPP@PP@@@PP	12
5	@PP@PP@PP@PP@PP, @PPP@PP@@@PP@PP, @@PPP@PP@PP@@@P	15
6	@PPP@PP@PP@@@P	14

– 따라서 $1 \leq N \leq 4$일 때 $in(N+1) - in(N) = 3$이고, $N = 5$일 때는 -1입니다.

- 이제 $N \geq 6$인 경우를 봅시다.
- 앞서 $in(6) = 14 < 18$이고 루프가 없는 프로그램의 앞 18글자를 $ins(6)$으로 치환하여 항상 더 짧은 프로그램을 얻을 수 있으므로, 루프가 없는 프로그램은 더 이상 고려할 필요가 없습니다.
- 강한 귀납법의 가정을 이용해 $in(N)$의 후보를 더 좁힐 수 있습니다.

$$
\begin{aligned}
N \bmod 3 = 0 \Rightarrow in(N) &= in(\tfrac{N}{3}) + 8 \\
N \bmod 3 = 1 \Rightarrow in(N) &= \min(in(\tfrac{N}{3}) + 9, in(\tfrac{N}{3} + 1) + 12) \\
&= \min(in(\tfrac{N}{3}) + 9, in(\tfrac{N}{3}) + \{3, 1, -1\} + 12) = in(\tfrac{N}{3}) + 9 \\
N \bmod 3 = 2 \Rightarrow in(N) &= \min(in(\tfrac{N}{3}) + 12, in(\tfrac{N}{3} + 1) + 9) \\
&= \min(in(\tfrac{N}{3}) + 12, in(\tfrac{N}{3}) + \{3, 1, -1\} + 9) = in(\tfrac{N}{3} + 1) + 9
\end{aligned}
$$

– 이제 앞의 결과를 이용해 각 경우의 $in(N+1) - in(N)$을 구할 수 있습니다.

$$N \bmod 3 = 0 \Rightarrow in(N+1) - in(N) = 1$$
$$N \bmod 3 = 1 \Rightarrow in(N+1) - in(N) = in(\tfrac{N}{3}+1) - in(\tfrac{N}{3}) = \{3, 1, -1\}$$
$$N \bmod 3 = 2 \Rightarrow in(N+1) - in(N) = -1$$

– 따라서 모든 $N \geq 1$에 대해 $in(N+1) - in(N) = \{3, 1, -1\}$입니다. □

– Corollary 4. $N \geq 1$에 대해 $in(N)$은 다음과 같이 재귀적으로 $\mathcal{O}(\log N)$에 구할 수 있습니다.

$$in(N) = \begin{cases} 3N, & \text{if } N \leq 5 \\ in(\frac{N}{3}) + 8, & \text{if } N > 5 \text{ and } N \bmod 3 = 0 \\ in(\frac{N}{3}) + 9, & \text{if } N > 5 \text{ and } N \bmod 3 = 1 \\ in(\frac{N}{3} + 1) + 9, & \text{if } N > 5 \text{ and } N \bmod 3 = 2 \end{cases}$$

– $k \geq 3$에 대해 $\sum_{i=8}^{3k+1} in(i)$은 다음과 같이 구할 수 있습니다.

$$\begin{aligned}\sum_{i=8}^{3k+1} in(i) &= \sum_{j=3}^{k} (in(3j-1) + in(3j) + in(3j+1)) \\ &= \sum_{j=3}^{k} ((in(j)+9) + (in(j)+8) + (in(j)+9)) \\ &= 3\sum_{j=3}^{k} in(j) + 26(k-2)\end{aligned}$$

- Corollary 4를 이용해 끝부분에서 값을 하나씩 따로 구하는 방식으로 위 아래 양 범위를 조정해가며 재귀적으로 구하면 $\mathcal{O}\left(\log^2 N\right)$에 $\sum_{i=1}^{N} in(i)$을 구할 수 있습니다.
- Challenge: $\mathcal{O}\left(\log N\right)$에 구할 수 있을까요?

- 이제 $out(N)$을 구해봅시다.
- 루프가 없는 경우의 최적의 프로그램은 P, PP, 또는 PP(@PP)$_{N-2}$ 입니다.
- 루프를 포함하는 경우의 최적의 프로그램은 역시 $N \bmod 3$에 따라 달라집니다.

Cases	가능한 방법	$out(N)$ 후보
$N \bmod 3 = 0$	PPP$(ins(\frac{N}{3}))$@@@	$in(\frac{N}{3}) + 6$
$N \bmod 3 = 1$	PPP$(ins(\frac{N}{3}))$@@@P	$in(\frac{N}{3}) + 7$
	@@PPP$(ins(\frac{N}{3}+1))$@@@	$in(\frac{N}{3}+1) + 8$
$N \bmod 3 = 2$	PPP$(ins(\frac{N}{3}))$@@@PP	$in(\frac{N}{3}) + 8$
	@PPP$(ins(\frac{N}{3}+1))$@@@	$in(\frac{N}{3}+1) + 7$

- 이제 Theorem 3의 증명 과정과 비슷한 방법으로 $out(N)$을 구할 수 있습니다.
- $1 \leq N \leq 6$인 경우, 가능한 방법을 직접 구해 보면 다음과 같습니다.

N	가능한 방법	$out(N)$
1	P	1
2	PP	2
3	PP@PP	5
4	PP@PP@PP	8
5	PP@PP@PP@PP, PPP@PP@@@PP	11
6	PPP@PP@PP@@@	12

- $N = 6$의 경우로부터 $N \geq 6$일 때 루프가 없는 프로그램은 고려할 필요가 없음을 알 수 있습니다.

– 이제 $N \geq 6$인 경우 각각에 대해 $out(N)$을 계산해보면 다음과 같습니다.

$$
\begin{aligned}
N \bmod 3 = 0 \Rightarrow out(N) &= in(\tfrac{N}{3}) + 6 = in(N) - 2 \\
N \bmod 3 = 1 \Rightarrow out(N) &= \min(in(\tfrac{N}{3}) + 7, in(\tfrac{N}{3} + 1) + 8) \\
&= \min(in(\tfrac{N}{3}) + 7, in(\tfrac{N}{3}) + \{3, 1, -1\} + 8) \\
&= in(\tfrac{N}{3}) + 7 = in(N) - 2 \\
N \bmod 3 = 2 \Rightarrow out(N) &= \min(in(\tfrac{N}{3}) + 8, in(\tfrac{N}{3} + 1) + 7) \\
&= \min(in(\tfrac{N}{3} + 1) - \{3, 1, -1\} + 8, in(\tfrac{N}{3} + 1) + 7) \\
&= \min(in(\tfrac{N}{3} + 1) + \{5, 7, 9\}, in(\tfrac{N}{3} + 1) + 7) \\
&= in(\tfrac{N}{3} + 1) + \{5, 7, 7\} = in(N) - \{4, 2, 2\}
\end{aligned}
$$

- 앞 슬라이드의 계산에서 $N \geq 6$ 이면 많은 경우에 $out(N) = in(N) - 2$ 임을 알 수 있습니다.
- 예외는 $N \bmod 3 = 2$ 이고 $in(\frac{N}{3} + 1) - in(\frac{N}{3}) = 3$ 인 경우에 발생합니다.
  - 이때는 $out(N) = in(N) - 4$ 입니다.

- $d_3(N) = \begin{cases} 1, & \text{if } in(N+1) - in(N) = 3 \\ 0, & \text{otherwise} \end{cases}$ 로 정의하면, $N \geq 6$에 대해 다음이 성립합니다.

$$\sum_{i=6}^{N} out(i) = \sum_{i=6}^{N} (in(i) - 2) - 2 \sum_{\substack{6 \leq i \leq N \\ i \bmod 3 = 2}} d_3(\tfrac{i}{3})$$

$$= \sum_{i=6}^{N} (in(i) - 2) - 2 \sum_{j=2}^{\frac{N-2}{3}} d_3(j)$$

- in의 연속합과 d_3의 연속합을 구하면 문제를 해결할 수 있습니다.

- Corollary 4를 적용하면 d_3을 다음과 같이 구할 수 있습니다. 증명은 생략합니다.
- Theorem 5. $N \geq 1$에 대해 다음이 성립합니다.

$$d_3(N) = \begin{cases} 1, & \text{if } 1 \leq N \leq 4 \\ 0, & \text{if } N = 5 \\ d_3(\frac{N}{3}), & \text{if } N > 5 \text{ and } N \bmod 3 = 1 \\ 0, & \text{if } N > 5 \text{ and } (N \bmod 3) \in \{0, 2\} \end{cases}$$

- $N \geq 6$에 대해 $\sum_{i=6}^{N} d_3(i)$은 다음과 같이 구할 수 있습니다.

$$\begin{aligned}\sum_{i=6}^{N} d_3(i) &= \sum_{\substack{6 \leq i \leq N \\ i \bmod 3 = 1}} d_3\left(\frac{i}{3}\right) \\ &= \sum_{j=2}^{\frac{N-1}{3}} d_3(j)\end{aligned}$$

- in의 부분합과 마찬가지로 아래쪽 범위를 조정해가면서 재귀적으로 구하면 $\mathcal{O}(\log N)$에 $\sum_{i=1}^{N} d_3(i)$를 구할 수 있습니다.

OVA-E. 소수 세기

`number-theory`
출제진 의도 – **Hard**

- 제출 89번, 정답 27명 (정답률 32.584%)
- 처음 푼 사람: **xiaowuc1**, 15분
- 출제자: `index`

- 우선 다이나믹 프로그래밍을 통해 접근해봅시다.
- 먼저, $N = 3\,000\,000$에 대해 에라토스테네스의 체를 통해 2 이상 N 이하의 소수를 구합니다.
- 2부터 N까지의 소수들을 오름차순으로 확인하며 문제의 답을 계산해나갑니다.
- 소수 p에 대해 문제의 답을 $f(p)$라 하면, $f(p)$는 i와 $p-1-i$가 동시에 소수인 i에 대해 $f(i)+f(p-1-i)+1$의 최댓값입니다. 만약 그런 i가 없다면 $f(p)$는 1이 됩니다.
- i의 값을 매번 1부터 $p-1$까지 돌며 찾는다면 시간 복잡도는 $\mathcal{O}\left(N^2/\log(N)\right)$이 됩니다.
- 그러나 N 제한이 크기 때문에 이 풀이로는 제한 시간 내에 돌기 어렵습니다.

OVA-E. 소수 세기

- 처음에 소수 P에서 시작해서 모든 과정이 끝났을 때 칠판에 남아있는 수들을 생각해봅시다.
- 남아있는 수의 개수를 m이라 하고, 각 수를 $p_1, p_2, \cdots, p_m$이라 합시다.
- 이때 $\sum_{i=1}^{m} p_i = P - m + 1$임을 수학적 귀납법으로 증명할 수 있습니다.
- 아래와 같이 정리하면 $m \leq \left\lfloor \frac{P+1}{3} \right\rfloor$임을 보일 수 있습니다.

$$\begin{aligned} P + 1 &= \sum_{i=1}^{m} p_i + m \\ &\geq \sum_{i=1}^{m} 2 + m \\ &= 3m \end{aligned}$$

- 한편, 소수를 적는 전체 횟수는 $2m-1$이 됨을 보일 수 있습니다.
- 따라서, $f(P) \leq 2\left\lfloor \frac{P+1}{3} \right\rfloor - 1$입니다.
- f의 값을 일부 확인해보면 위 식의 상한을 달성함을 확인할 수 있습니다.
- 소수 p에 대해, $f(p) = 2\left\lfloor \frac{p+1}{3} \right\rfloor - 1$일 것이라 추측해봅시다.

OVA-E. 소수 세기

- 어떤 소수 p에 대해서 p보다 작은 모든 소수에 대해 추측이 성립한다고 합시다.
- $p \bmod 3 = 1$를 가정했을 때, i와 $p-1-i$가 동시에 소수인 i가 존재한다고 하면 아래와 같이 p에 대해 추측이 성립함을 보일 수 있습니다.

$$\begin{aligned} f(p) &\geq f(i) + f(p-1-i) + 1 \\ &= 2\left\lfloor \frac{i+1}{3} \right\rfloor + 2\left\lfloor \frac{p-i}{3} \right\rfloor - 1 \\ &= 2\left\lfloor \frac{i+1}{3} \right\rfloor + 2\left\lfloor \frac{1-i}{3} \right\rfloor + 2\left\lfloor \frac{p}{3} \right\rfloor - 1 \\ &= 2\left\lfloor \frac{p}{3} \right\rfloor - 1 = 2\left\lfloor \frac{p+1}{3} \right\rfloor - 1 \end{aligned}$$

- 세번째 등호는 $i \bmod 3$에 대해 경우를 나눠서 정리하면 유도할 수 있습니다.

- $p \bmod 3 = 2$를 가정했을 때, i와 $p-1-i$가 동시에 소수인 i가 존재한다고 하면 아래와 같이 정리할 수 있습니다.

$$
\begin{aligned}
f(p) &\geq f(i) + f(p-1-i) + 1 \\
&= 2\left\lfloor \frac{i+1}{3} \right\rfloor + 2\left\lfloor \frac{p-i}{3} \right\rfloor - 1 \\
&= 2\left\lfloor \frac{i+1}{3} \right\rfloor + 2\left\lfloor \frac{2-i}{3} \right\rfloor + 2\left\lfloor \frac{p}{3} \right\rfloor - 1 \\
&= 2\left\lfloor \frac{i+1}{3} \right\rfloor + 2\left\lfloor \frac{2-i}{3} \right\rfloor + 2\left\lfloor \frac{p+1}{3} \right\rfloor - 3
\end{aligned}
$$

– 이때 아래 등식이 성립하므로, $i \bmod 3 = 2$인 경우가 존재하면 p에 대해서 추측이 성립함을 보일 수 있습니다.

$$\left\lfloor \frac{i+1}{3} \right\rfloor + \left\lfloor \frac{2-i}{3} \right\rfloor = \begin{cases} \left\lfloor \frac{0+1}{3} \right\rfloor + \left\lfloor \frac{2-0}{3} \right\rfloor, & \text{if } i \bmod 3 = 0 \\ \left\lfloor \frac{1+1}{3} \right\rfloor + \left\lfloor \frac{2-1}{3} \right\rfloor, & \text{if } i \bmod 3 = 1 \\ \left\lfloor \frac{2+1}{3} \right\rfloor + \left\lfloor \frac{2-2}{3} \right\rfloor, & \text{if } i \bmod 3 = 2 \end{cases}$$

$$= \begin{cases} 0, & \text{if } i \bmod 3 = 0 \\ 0, & \text{if } i \bmod 3 = 1 \\ 1, & \text{if } i \bmod 3 = 2 \end{cases}$$

- 앞의 논의에서 문제 범위에서 다음과 같은 성질이 성립함을 확인하면 됨을 알 수 있습니다.
  1. $p \bmod 3 = 1$인 소수 p에 대해 i와 $p - 1 - i$가 동시에 소수인 i가 존재한다.
  2. $p \bmod 3 = 2$인 소수 $p \geq 5$에 대해 i와 $p - 1 - i$가 동시에 소수이고 $i \bmod 3 = 2$인 i가 존재한다.
- 위 성질이 성립하는지 확인하는 프로그램을 작성하면 문제 범위에서 성질이 성립함을 알 수 있습니다.

OVA-F. 하이퍼 주사위 굴리기

`geometry-hyper`, `simulation`
출제진 의도 – **Hard**

- 제출 10번, 정답 2명 (정답률 20.000%)
- 처음 푼 사람: **yijw0930**, 18분
- 출제자: `index`

- 명령으로 주어진 수를 v라 하고, 해당 명령을 수행하기 전에 $+x_i$ 방향 면에 쓰여진 수를 d_{+i}, $-x_i$ 방향 면에 쓰여진 수를 d_{-i} 라고 정의합니다. 문제의 조건에서 $d_{+i} + d_{-i} = 2N + 1$ 입니다.
- 문제 상황에서 명령을 수행하고 나면 $-x_1$ 방향에 v가 적힌 면이, $+x_1$ 방향에 $2N + 1 - v$가 적힌 면이 오게 됩니다.
- 명령을 수행하기 전에 v가 적힌 방향이 $+x_i$ 방향이었다면 명령을 수행하고 나면 $-x_i$ 방향에 d_{-1}이 적힌 면이, $+x_i$ 방향에 d_{+1}이 적힌 면이 오게 됩니다.
- 명령을 수행하기 전에 v가 적힌 방향이 $-x_i$ 방향이었다면 명령을 수행하고 나면 $-x_i$ 방향에 d_{+1}이 적힌 면이, $+x_i$ 방향에 d_{-1}이 적힌 면이 오게 됩니다.

OVA-F. 하이퍼 주사위 굴리기

- 앞 슬라이드에 이어서 명령을 수행한 후에 $+x_i$ 방향 면에 쓰여진 수를 d'_{+i}, $-x_i$ 방향 면에 쓰여진 수를 d'_{-i} 라고 정의합시다. 마찬가지로 $d'_{+i} + d'_{-i} = 2N + 1$ 입니다.
- 명령을 수행하기 전에 v 가 적힌 방향이 $\pm x_i$ 방향이었다고 할 때, 다음과 같습니다.

$$d'_{-j} = \begin{cases} v & \text{if } j = 1 \\ d_{\mp 1} & \text{if } j = i \\ d_{-j} & \text{otherwise} \end{cases}$$

$$d'_{+j} = \begin{cases} 2N + 1 - v & \text{if } j = 1 \\ d_{\pm 1} & \text{if } j = i \\ d_{+j} & \text{otherwise} \end{cases}$$

- 명령을 수행할 때 $d_{\pm i}$의 값이 바뀌는 방향의 수가 최대 4개임을 알 수 있습니다. 따라서 N이 크더라도 $d_{\pm i}$의 값이 변하는 총 횟수는 $\mathcal{O}(M)$ 번이기 때문에, 시뮬레이션을 통해 문제를 풀 수 있습니다.
- key가 수($d_{\pm i}$)이고, value에 해당 수가 적혀있는 면의 방향($\pm x_i$)을 저장하는 map 자료구조를 사용합니다. 이때 $d_{\pm i}$의 값이 바뀐 적이 있는 방향에 적힌 수들에 대해서만 저장합니다. 또, 명령을 수행할 때 map의 값들을 업데이트 하기 위해서 d_{-1}의 값을 저장하고 있어야 합니다.
- 최종적으로 원점과 주사위 중심 간 유클리드 거리의 제곱을 구해야 하므로, 주사위 중심의 좌표를 저장해야 합니다.

- map 연산의 횟수가 $\mathcal{O}(M)$ 이므로 시간 복잡도는 트리를 사용한 맵의 경우 $\mathcal{O}(M \log M)$, 해시를 사용한 맵의 경우 $\mathcal{O}(M)$ 입니다.
- 이외의 풀이로는 명령의 목록이 주어지면 계산에 필요한 좌표축의 목록을 알 수 있으므로, 필요한 좌표축만을 좌표 압축으로 남긴 다음 map 대신 배열을 사용하는 $\mathcal{O}(M \log M)$ 풀이가 있습니다.