


제2회 초콜릿컵 에디토리얼

bubbler

special thanks to startlink

A	Easy	3단 초콜릿 아이스크림	... <u>3</u>
B	Easy	초콜릿 보관함	... <u>5</u>
C	Medium	알록달록 초콜릿 만들기	... <u>7</u>
D	Medium	초콜릿 보물 찾기	... <u>9</u>
E	Medium	초콜릿 트리 만들기	... <u>13</u>
F	Hard	초콜릿의 맛은 몇 점?	... <u>16</u>
G	Hard	화이트, 다크, 민트 초콜릿	... <u>20</u>
H	Challenging	초콜릿 비긴즈	... <u>25</u>
	Impossible	하이퍼 가짜 초콜릿	... <u>32</u>

A. 3단 초콜릿 아이스크림

implementation, string

출제자: **bubbler** / 난이도: **Easy**

First Solve: **aqua3g** (3분)

A. 3단 초콜릿 아이스크림

- ❖ 지문에 주어진 3단 아이스크림의 정의대로 구현하면 됩니다.
- ❖ 예를 들어 Python으로 구현을 하는 경우, 각각의 연산은 다음과 같이 구현할 수 있습니다.
 - ❖ S' 의 길이: $(\text{len}(s) + 2) // 3$
 - ❖ S' : $s[:(\text{len}(s) + 2) // 3]$
 - ❖ rev 함수: $s[::-1]$
 - ❖ tail 함수: $s[1:]$

B. 초콜릿 보관함

implementation, sorting

출제자: **bubbler** / 난이도: **Easy**

First Solve: **patata22** (5분)

B. 초콜릿 보관함

- ❖ 초콜릿이 들어있는 칸들을 시계 방향 또는 시계 반대 방향으로 모아서 하나의 문자열을 만들 수 있습니다.
- ❖ 하지만 이 상태로 연속된 O들을 세면 8번째 칸과 첫 번째 칸이 모두 O일 때 두 개의 서로 다른 구간으로 잘못 세게 됩니다.
- ❖ 이를 해결하려면, 첫 번째 칸이나 8번째 칸 중 하나에 X가 오도록 문자열을 회전하면 됩니다. 예외적으로, 모든 칸이 O인 경우는 그냥 처리하면 됩니다.
- ❖ 언어에 따라 예외 처리를 하지 않을 경우 RE 또는 WA를 받을 수 있습니다. (e.g. Python에서 예외 처리 없이 `.index('X')`를 사용할 경우 RE)
- ❖ 마지막으로, 각 구간의 O의 개수를 센 다음 정렬해서 화면의 값과 비교하면 됩니다.

C. 알록달록 초콜릿 만들기

math, binary-search

출제자: **bubbler** / 난이도: **Medium**

First Solve: **dabbler1** (21분)

C. 알록달록 초콜릿 만들기

- 주어진 무늬에서 n 번째 줄을 R_n 이라고 하고, 맨 윗 줄을 R_1 이라고 합시다.
- R_1, R_2, R_3, \dots 에 있는 초콜릿의 개수는 $1, 0, 1, 2, 1, 2, 3, 2, 3, \dots$ 로 규칙적으로 증가하며, 각 줄의 민트 초콜릿은 그 줄의 $1, 3, 2, 1, 3, 2, \dots$ 번째 칸에서 시작해서 3칸마다 놓여집니다. R_{3k} 까지의 모든 줄에 있는 민트 초콜릿의 수의 합은 $2 + 5 + \dots + (3k - 1) = \frac{3k(k+1)}{2} - k$ 입니다.
- 위의 규칙을 이용해서 각 무늬에서 번호가 n 번째로 작은 민트 초콜릿의 번호를 구하려면, 먼저 그 민트 초콜릿이 몇 번째 줄에 있는지를 이분탐색으로 찾고, 그 줄에서 몇 번째 칸에 오는지를 구합니다.
- 마지막으로 a 번째 줄의 b 번째 칸의 번호 $\frac{a(a-1)}{2} + b$ 를 출력하면 됩니다.

D. 초콜릿 보물 찾기

ad-hoc

출제자: **bubbler** / 난이도: **Medium**

First Solve: **jh01533** (32분)

D. 초콜릿 보물 찾기

- ❖ 아무 칸을 파서 보물상자의 일부가 있으면 최대 3번의 추가 퀴리로 답을 확정할 수 있습니다.
- ❖ 보물상자의 두 번째 칸은 주변 4칸 중 한 칸에 있으므로, 그 중 3칸을 시도해 보고 있으면 ok, 없으면 4번째 칸이 정답입니다.
- ❖ 첫 칸이 모서리에 있으면 2번, 꼭짓점에 있으면 1번의 추가 퀴리를 하면 됩니다.
- ❖ 체크무늬로 50칸을 파면 보물상자의 일부를 반드시 볼 수 있습니다.
- ❖ 하지만 50번째로 시도한 칸에 보물상자의 일부가 있으면 추가로 퀴리할 기회가 없으므로, 마지막에는 다른 처리가 필요합니다.

D. 초콜릿 보물 찾기

- ❖ 체크 무늬 50칸 중에서 내부에 있는 32칸을 먼저 시도합니다. 그 중에 보물상자의 일부가 있으면 $32 + 3 = 35$ 번 쿼리 이내에 성공합니다.
- ❖ 그 다음에는 모서리에 있는 16칸을 시도합니다. 그 중에 보물상자의 일부가 있으면 $48 + 2 = 50$ 번 쿼리 이내에 성공합니다.
- ❖ 그 다음에는 남아있는 두 꼭짓점 중 한 칸을 시도합니다. 여기에 보물상자의 일부가 있으면 $49 + 1 = 50$ 번 쿼리 이내에 성공합니다.
- ❖ 지금까지 보물상자가 발견되지 않았다면, 마지막 꼭짓점에는 반드시 보물상자가 있으므로 그 칸을 파볼 필요는 없고, 대신 이웃한 두 칸 중 하나를 파서 어느 쪽인지 판별하면 됩니다.

D. 초콜릿 보물 찾기

- ❖ note: 이 문제의 제한을 49로 바꾸면 풀 수 없습니다. 증명은 다음과 같습니다.
- ❖ 10×10 크기의 격자는 50개의 가로 도미노로도 채울 수 있고, 50개의 세로 도미노로도 채울 수 있습니다.
- ❖ 49번의 쿼리를 해서 보물상자가 발견되지 않았다면, 50개의 가로 도미노 중에도 쿼리되지 않은 도미노가 반드시 존재하고, 50개의 세로 도미노 중에도 마찬가지로 존재합니다.
- ❖ 보물상자가 묻힌 곳의 후보가 최소 2개가 남아있게 되므로, 49번의 쿼리를 어떻게 하더라도 정답을 알 수 없는 경우가 존재합니다.

E. 초콜릿 트리 만들기

dp

출제자: **bubbler** / 난이도: **Medium**

First Solve: **dabbler1** (44분)

E. 초콜릿 트리 만들기

❖ 다음과 같은 DP식을 생각합니다.

❖ $dp[h][x]$: 리프의 깊이가 h 이고 루트가 x 인 트리에서 한별이에게 빌려야 하는 초콜릿의 개수의 최소값

❖ 편의상 $A = \{a_1, a_2, \dots, a_k\}$ 라고 합시다.

❖ $h = 0$ 인 경우 (루트 노드만 있는 경우)

❖ $dp[0][x]$ 는 $x \in A$ 이면 0, 아니면 1입니다.

E. 초콜릿 트리 만들기

❖ $h > 0$ 인 경우

❖ $\text{dp}[h][x]$ 의 x 를 고정해놓고 양쪽 서브트리의 가능한 경우들을 생각해보면, 왼쪽 서브트리의 루트로 $0 \leq y < n$ 의 n 가지 경우가 가능합니다. 오른쪽 서브트리는 $(x - y) \bmod n$ 으로 고정됩니다.

❖ 각각의 경우에 대해, 루트를 제외하고 빌려야 하는 총 개수의 최소값은 $\text{dp}[h - 1][y] + \text{dp}[h - 1][(x - y) \bmod n]$ 이 됩니다.

❖ 이 값을 모든 y 에 대해 구해서 최소값을 취한 다음, $x \notin A$ 이면 1을 더해 주면 $\text{dp}[h][x]$ 의 값이 됩니다.

❖ 최종적으로, 모든 x 에 대한 $\text{dp}[h][x]$ 값들의 최소값을 출력하면 됩니다.

❖ 시간 복잡도는 $\mathcal{O}(HN^2)$ 입니다.

F. 초콜릿의 맛은 몇 점?

bruteforcing, bitmask, recursion

출제자: **bubbler** / 난이도: **Hard**

First Solve: **aeren** (217분)

F. 초콜릿의 맛은 몇 점?

- ❖ 딱히 수학적 구조가 없으므로 정직하게 모든 polyomino에 대한 XOR 값을 모두 구해서 더해야 합니다.
- ❖ 주어진 제한은 폴리오미노가 약 10^7 개까지 나올 수 있는 제한이며, 입력이 4×7 크기일 때 발생합니다. 따라서 모든 bitmask를 생성해서 bfs를 돌리는 방법으로는 통과하기 어렵고, 정확히 모든 polyomino만을 생성하는 방법을 찾아야 합니다.

F. 초콜릿의 맛은 몇 점?

- 🍬 이는 다음과 같은 재귀 함수를 작성하여 달성할 수 있습니다. 아래 pseudocode에서 x 는 현재 polyomino의 XOR값, C 는 바로 다음에 추가할 수 있는 칸들의 집합 (현재 polyomino와 바로 인접한 칸들 중에서 사용 가능한 칸들), F 는 미래에 추가할 수 있는 칸들의 집합입니다. $C \cap F = \emptyset$ 입니다.

```
function recurse(x, C, F):
    ret = x
    for c in C:
        remove c from C
        C2 = copy of current C
        F2 = copy of current F
        for c2 in 4-way neighbors of c:
            if c2 in F:
                add c2 to C2; remove c2 from F2
        ret += recurse(x ^ value of c2, C2, F2)
    return ret
```

F. 초콜릿의 맛은 몇 점?

- ❖ c에서 c를 뺀 후 다시 c를 넣지 않는 것이 중요합니다. 현재의 polyomino와 c칸을 모두 포함하는 polyomino를 모두 처리했으면, 그 다음부터는 c를 포함하지 않는 것만 처리하도록 해 줌으로써 중복 처리를 방지해 줍니다.
- ❖ 위의 함수 recurse는 1칸 이상의 polyomino가 이미 있고 그에 맞게 c를 주었을 때만 동작하므로, 모든 polyomino에 대한 답을 구하려면 각각의 칸에서 시작해서 얻은 답을 모두 더하면 됩니다. 이때, 같은 원리로 칸 c를 포함하는 모든 polyomino에 대해 계산이 끝났다면 다음 칸부터는 c를 c와 F에서 제외해야 합니다.
- ❖ 마지막으로, recurse 내에서 c2와 F2를 만들 때 array copy가 일어나므로 이를 $O(1)$ 로 만들기 위해 bitmask를 사용할 수 있습니다.

G. 화이트, 다크, 민트 초콜릿

ad-hoc, segment-tree, lucas-theorem

출제자: **bubbler** / 난이도: **Hard**

First Solve: **sedev57** (69분)

G. 화이트, 다크, 민트 초콜릿

- ❖ 문제에서 주어진 대로 아랫줄의 두 초콜릿의 색을 받아서 그에 해당하는 윗줄의 초콜릿의 색을 주는 함수를 $f(x, y)$ 라고 합시다.
- ❖ 손으로 여러 가지 경우를 시도해 보면 다음의 사실을 관찰할 수 있습니다.
 - ❖ 길이 4인 문자열 $x_0x_1x_2x_3$ 의 3줄 위에 등장하는 초콜릿은 $f(x_0, x_3)$ 입니다.
- ❖ 이를 확장하여 여러 가지 관찰이 가능하며, 찾은 관찰에 따라 적어도 3가지 풀이가 가능합니다.

G. 화이트, 다크, 민트 초콜릿

- 관찰: 길이 $3^k + 1$ 인 문자열 $x_0x_1\dots x_{3^k}$ 의 3^k 줄 위에 등장하는 초콜릿은 $f(x_0, x_{3^k})$ 입니다.
- 풀이
 - $n \geq 3^k + 1$ 인 k 를 찾아서 $n - 3^k$ 길이의 문자열을 계산하는 것을 길이가 1이 될 때까지 반복합니다. 그러면 최초 상태의 답을 $\mathcal{O}(n)$ 에 구할 수 있습니다.
 - 쿼리를 수행하려면, 위의 계산 과정을 모두 기록해 놓습니다. 원래의 문자열에서 1개의 문자가 바뀌면, 각 중간 과정에서 최대 2개의 문자가 바뀌므로 각 쿼리를 $\mathcal{O}(\log n)$ 에 구할 수 있습니다.
 - 시간 복잡도는 $\mathcal{O}(n + Q \log n)$ 입니다.

G. 화이트, 다크, 민트 초콜릿

- 관찰: 길이 $3k + 1$ 인 문자열 $x_0x_1\dots x_{3k}$ 이 있을 때 꼭짓점에 등장하는 초콜릿은 $x_0x_3x_6\dots x_{3k}$ 와 같습니다.
- 풀이
 - $n \geq 3k + 1$ 인 k 를 찾아서 0..2줄을 계산한 후 1/3로 압축하는 것을 길이가 1이 될 때까지 반복합니다.
 - 앞의 풀이와 같은 방법으로 쿼리를 처리할 수 있으며, 시간 복잡도도 같습니다.

G. 화이트, 다크, 민트 초콜릿

관찰: $W = 0, D = 1, M = 2$ 라고 하면, $f(x, y) = -(x + y) \bmod 3$ 입니다.

풀이

$x_0 x_1 \dots x_{n-1}$ 이 주어질 때, $n - 1$ 줄 위의 꼭짓점의 초콜릿의 번호는

$$(-1)^{n-1} \sum_{i=0}^{n-1} \binom{n-1}{i} x_i \bmod 3$$

이 됩니다. 따라서 이항 계수 $\bmod 3$ 을 모두 구해 놓으면 각 쿼리를 $\mathcal{O}(1)$ 에 구할 수 있습니다.

뤼카 정리를 쓰면 간단하게 $\mathcal{O}(n \log n)$ 에 전처리가 되고, 뫼비우스 정리를 모르더라도 $i!$ 를 나누는 최대 3^k 의 지수와 $\frac{i!}{3^k} \bmod 3$ 을 가지고 모든 이항 계수 $\bmod 3$ 을 구할 수 있습니다.

시간 복잡도는 $\mathcal{O}(n \log n + Q)$ 입니다.

H. 초콜릿 비긴즈

ad-hoc, dp-bitfield

출제자: **bubbler** / 난이도: **Challenging**

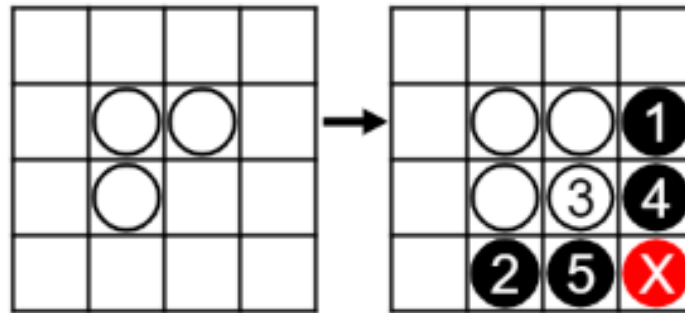
First Solve: **aurora2001** (287분)

H. 초콜릿 비긴즈

- ❖ 현대의 시간 복잡도 낚시 문제(?!)입니다.
- ❖ 일반성을 잃지 않고 $N \leq M$ 이라고 합시다. 결론부터 말하면, $N \leq 3$ 일 때만 비트 DP를 돌리면 되고, $N \geq 4$ 일 경우에는 답이 커지지 않고 반복된다는 사실을 이용해서 답을 출력하면 됩니다.
- ❖ 비트 DP는 다음과 같이 구성합니다.
 - ❖ $dp[l][p_1][p_2][k]$: 가로 길이가 l 칸이고 마지막 2개 세로줄의 패턴이 p_1, p_2 (각각 비트마스크 형태)이면서 전체 화이트 초콜릿의 개수가 k 개인 비긴 보드의 수 ($0 \leq k \leq 3l, 0 \leq p_1, p_2 < 2^N$)
 - ❖ 이렇게 하면 각 상태마다 상태 전이하는 시간이 2^N 만큼 소요되므로 전체 시간 복잡도는 $\mathcal{O}(2^{3N} M^2)$ 인데, 실제로 일어나는 상태 전이는 훨씬 적기 때문에 가능한 세로줄의 상태($N = 3$ 일 경우 6가지)와 모든 세로줄 조합에 대한 상태 전이(28가지)를 전처리하면 됩니다.

H. 초콜릿 비긴즈

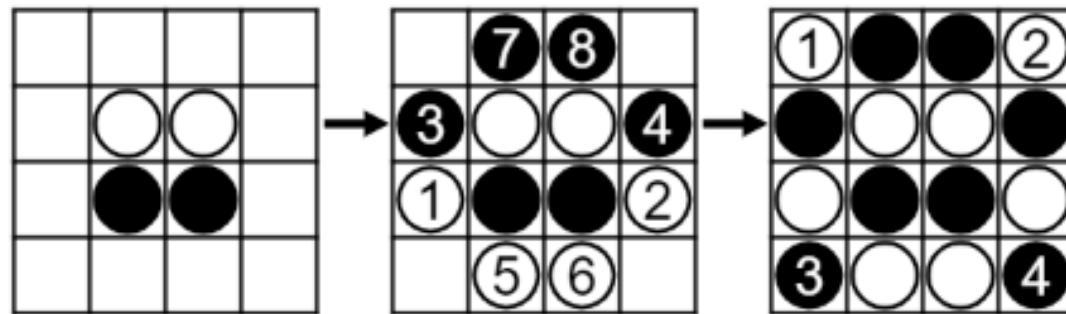
- ❖ $N \geq 4$ 일 때의 답의 증명은 다음과 같습니다.
- ❖ 먼저 다음과 같이 보드 중간에 같은 색의 삼각형이 나오는 경우를 생각해 봅시다.



- ❖ 이 경우 보드를 채우다 보면 x의 자리에 어떤 초콜릿을 놓아도 연속 3개가 만들어집니다. 따라서 비긴 보드의 중간에는 위와 같은 모양이 나올 수 없습니다.
- ❖ 이는 위 모양을 돌리거나 검은 초콜릿으로 바뀌어도 마찬가지입니다. 따라서 보드 중간의 모든 2×2 영역에는 흰색과 검은색 초콜릿이 2개씩 포함되어야 합니다.

H. 초콜릿 비긴즈

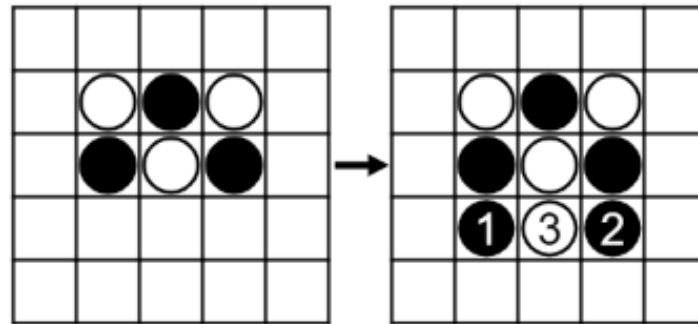
🍫 이제 다음의 경우를 생각해 봅시다.



🍫 이 모양이 보드 중간에서 한 번이라도 등장하면, 상하좌우 모든 방향으로 패턴이 확장되면서 보드 전체를 채우는 방법의 수가 하나뿐임을 알 수 있습니다.

H. 초콜릿 비긴즈

🍫 $N, M \geq 5$ 라면, 다음에 의해 모든 비긴 보드가 위 패턴을 포함하게 됩니다.

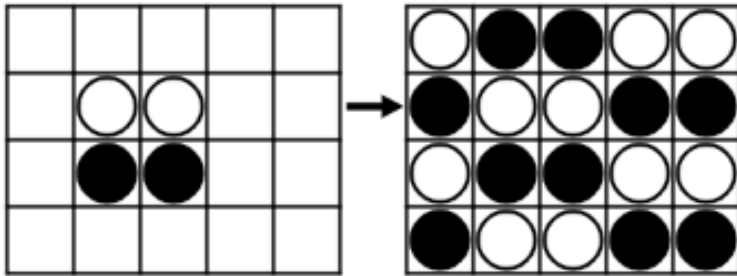


- 🍫 따라서 이 반복 패턴을 가로 또는 세로로 놓고 $N \times M$ 영역을 추출했을 때 화이트 초콜릿의 개수가 다크 초콜릿과 같거나 하나 많은 경우의 수를 세면 됩니다.
- 🍫 답은 항상 8 이하이고, 가로나 세로 방향으로 4칸 확장하면 항상 화이트와 다크 초콜릿이 같은 개수만큼 늘어나므로, $N, M \leq 8$ 인 경우를 비트DP 코드로 구해서 $N \bmod 4$ 와 $M \bmod 4$ 로 답을 골라 출력하면 됩니다.

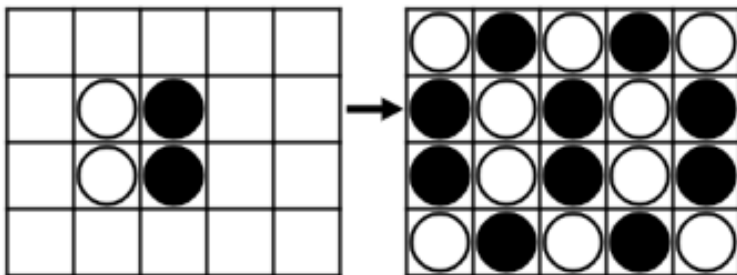
H. 초콜릿 비긴즈

🍫 $N = 4, M \geq 5$ 인 경우는 다음과 같이 경우가 나뉘집니다.

🍫 가로 패턴 4가지

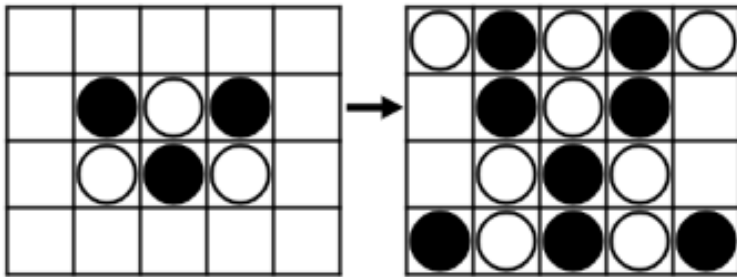


🍫 아래 모양과 그 흑백 반전 포함 2가지



H. 초콜릿 비긴즈

- 아래 모양에서 왼쪽 끝 2칸과 오른쪽 끝 2칸을 각각 ○● 또는 ●○로 채울 수 있고, 흑백을 반전할 수 있으므로 총 8가지



- 따라서 이 경우 답은 M 과 상관 없이 14입니다.
- $N = M = 4$ 인 경우는 직접 계산해 볼 수 있고, 답은 18입니다.
- Bonus: $N = 2, 3$ 일 때의 답을 $N = 1$ 의 DP table로 구할 수 있을까요?

. 하이퍼 가짜 초콜릿

number-theory, constructive

출제자: **bubbler** / 난이도: **Impossible**

First Solve: **aurora2001** (148분)

.하이퍼 가짜 초콜릿

- ❖ 가짜 소수, 즉 카마이클 수는 잘 알려진 개념이고, BOJ 13319와 같은 문제도 이미 있습니다. (모든 소인수가 500을 초과하는 카마이클 수를 아무거나 찾는 문제)
- ❖ 하지만 카마이클 수가 소수 11개의 곱이 되어야 한다면 이야기가 달라집니다.
 - ❖ OEIS A338443이 있긴 하지만, 2^{64} 미만인 수만 나열되어 있고 $2^{64} \ll 10^{70}$ 이므로 여기에는 답이 없습니다.
- ❖ Wikipedia에 나와 있는 Chernick의 공식 $(6k + 1)(12k + 1)(18k + 1)$ 을 확장하는 것을 시도해 볼 수 있습니다. 하지만 문제 조건 상 가장 큰 소인수는 가장 작은 소인수의 10배 미만이어야 한다는 점 때문에 가능성이 적고, 이 점을 무시하더라도 9개 이상의 소수를 사용한 공식은 찾을 수 없었습니다.

.하이퍼 가짜 초콜릿

- ❖ 역시 Wikipedia에 나와 있는 Korselt의 조건(또는 판별법)은 다음과 같습니다.
 - ❖ 양의 정수 n 이 카마이클 수이다 $\Leftrightarrow n$ 은 squarefree이고 (소인수가 모두 다르고) n 의 모든 소인수 p 는 $p - 1 \mid n - 1$ 을 만족한다.
- ❖ 이를 이용하면 다음의 방법으로 카마이클 수를 생성할 수 있습니다. (Erdős construction, Erdős, 1956)
 - ❖ 약수가 많은 합성수 $\Lambda = \prod_{i=1}^r q_i^{h_i}$ 를 적당히 고릅니다.
 - ❖ $\mathcal{P} = \{p \text{ prime} : p - 1 \mid \Lambda, p \nmid \Lambda\}$ 를 만족하는 집합 \mathcal{P} 를 만듭니다.
 - ❖ 서로 다른 $p_1, p_2, \dots, p_k \in \mathcal{P}$ 를 골라 $n = \prod_{i=1}^k p_i$ 라고 했을 때, $n \equiv 1 \pmod{\Lambda}$ 이면 n 은 카마이클 수입니다.

.하이퍼 가짜 초콜릿

- ❖ 이것이 성립하는 이유는 다음과 같습니다.
 - ❖ n 의 모든 소인수 p_i 에 대해 $p_i - 1 \mid n - 1$ 을 증명하면 Korselt 조건에 의해 n 은 카마이클 수입니다.
 - ❖ \mathcal{P} 의 정의에 의해 $p_i - 1$ 은 Λ 의 약수입니다. 한편, $n \equiv 1 \pmod{\Lambda}$ 이므로 $n - 1$ 은 Λ 의 배수입니다. 따라서 $p_i - 1 \mid \Lambda \mid n - 1$ 이 성립합니다.
- ❖ Erdős의 원본 article은 찾기 어렵고, 대신 Alford et al, 2013의 초반부에 관련 내용과 적절한 Λ 를 선정하는 방법이 나와 있습니다.

.하이퍼 가짜 초콜릿

- ❖ 이 문제에서는 $10^7 \leq p < 10^8$ 로 소인수가 제한되어 있으므로, 이 범위 내에 소수가 적당히 많이 들어가도록 Λ 를 선정해야 합니다.
- ❖ 출제자의 풀이 코드는 $\Lambda = 2^{10}3^55^37^211^1$ 을 사용했습니다.
- ❖ 이 경우 범위 내에 소수가 43개가 있고, 그 중 11개를 조합하는 방법의 수는 $\binom{43}{11} = 5752004349 > 3483648000 = \varphi(\Lambda)$ 으로 정답이 있을 가능성이 높다고 예상할 수 있습니다.
- ❖ $\prod p_i \bmod \Lambda$ 가 $\varphi(\Lambda)$ 가지의 값 중 하나를 균일한 확률로 갖는다고 가정했을 때, 11개의 소수를 단순히 랜덤하게 골라서, 또는 순차적으로 골라서 원하는 값(1)을 얻기에는 필요한 시행 횟수가 너무 많습니다.

.하이퍼 가짜 초콜릿

- ❖ 이를 가속하기 위해, 이산 로그를 구할 때 쓰는 baby-step giant-step 알고리즘의 변형을 사용할 수 있습니다.
- ❖ 랜덤성을 최대한 유지하기 위해 5개의 곱과 6개의 곱으로 나누어서 두 개의 해시 테이블에 나누어 저장합니다.
- ❖ 각각 랜덤으로 뽑아서 곱이 $\text{mod } \Lambda$ 로 1이 나오는 상대 조합이 있는지를 확인합니다. 상대 조합이 존재하면서 소수의 목록이 겹치지 않으면 두 조합에 포함된 11개의 소수를 출력합니다. 아니라면 해시 테이블에 그 값을 추가합니다.
- ❖ 출제자가 사용한 Λ 값과 비슷한 크기의 값들을 써 본 결과, 수만 번 이내의 iteration 내에 조건에 맞는 결과가 출력됨을 확인하였습니다.

.하이퍼 가짜 초콜릿

 pseudocode로 표현하면 다음과 같습니다.

Lam = large, highly composite number

L = list of primes derived from Lam

S1 = {}; S2 = {}

loop:

 p1to5 = list of 5 primes randomly picked from L

 inv = inverse of product of p1to5 modulo Lam

 if inv in S2 and S2[inv] is disjoint to p1to5:

 output S2[inv] + p1to5; break

 S1[product of p1to5 mod Lam] = p1to5

 p6to11 = list of 6 primes randomly picked from L

 inv = inverse of product of p6to11 modulo Lam

 if inv in S1 and S1[inv] is disjoint to p6to11:

 output S1[inv] + p6to11; break

 S2[product of p6to11 mod Lam] = p6to11