CTF writeup

TSG CTF 2020

slowest_decryption

@kurenaif

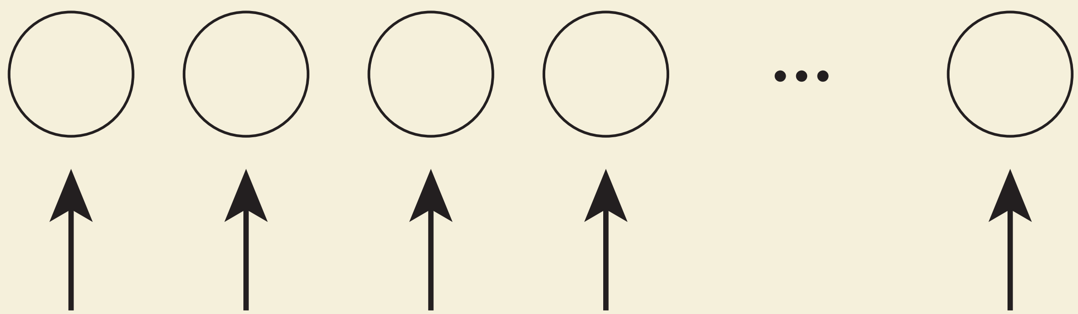
問題概要

赤：サーバー
青：クライアント

N = 20000 くらいの数字が与えられる（3桁くらいの数字）
P[0]~P[19999] とする

20000回数字を選ぶ。

20000個



P[0~19999]
の中から好きに選ぶ。（重複あり）

問題概要

選んだ数字を c[0]...c[19999] とする。(0~19999 の数字　重複あり)

選んだ数字の中身は P[c[0]]...P[c[19999]] になる。

gcd(c) * (0*P[c[0]] + 1*P[c[1]] + 2*P[c[2]] + ... + 19999*P[c[19999]])

を求める。

考えられるすべての c の選び方で、↑の式を求めて、その合計値を求める。

gcd(0,0,...,0) * (0*P[0] + 1*P[0] + 2*P[0] + ... + 19999*P[0]) +

gcd(0,0,...,1) * (0*P[0] + 1*P[0] + 2*P[0] + ... + 19999*P[1]) + ...

gcd(19999,19999,...,19999) * (0*P[19999] + 1*P[19999] + 2*P[19999] + ...)

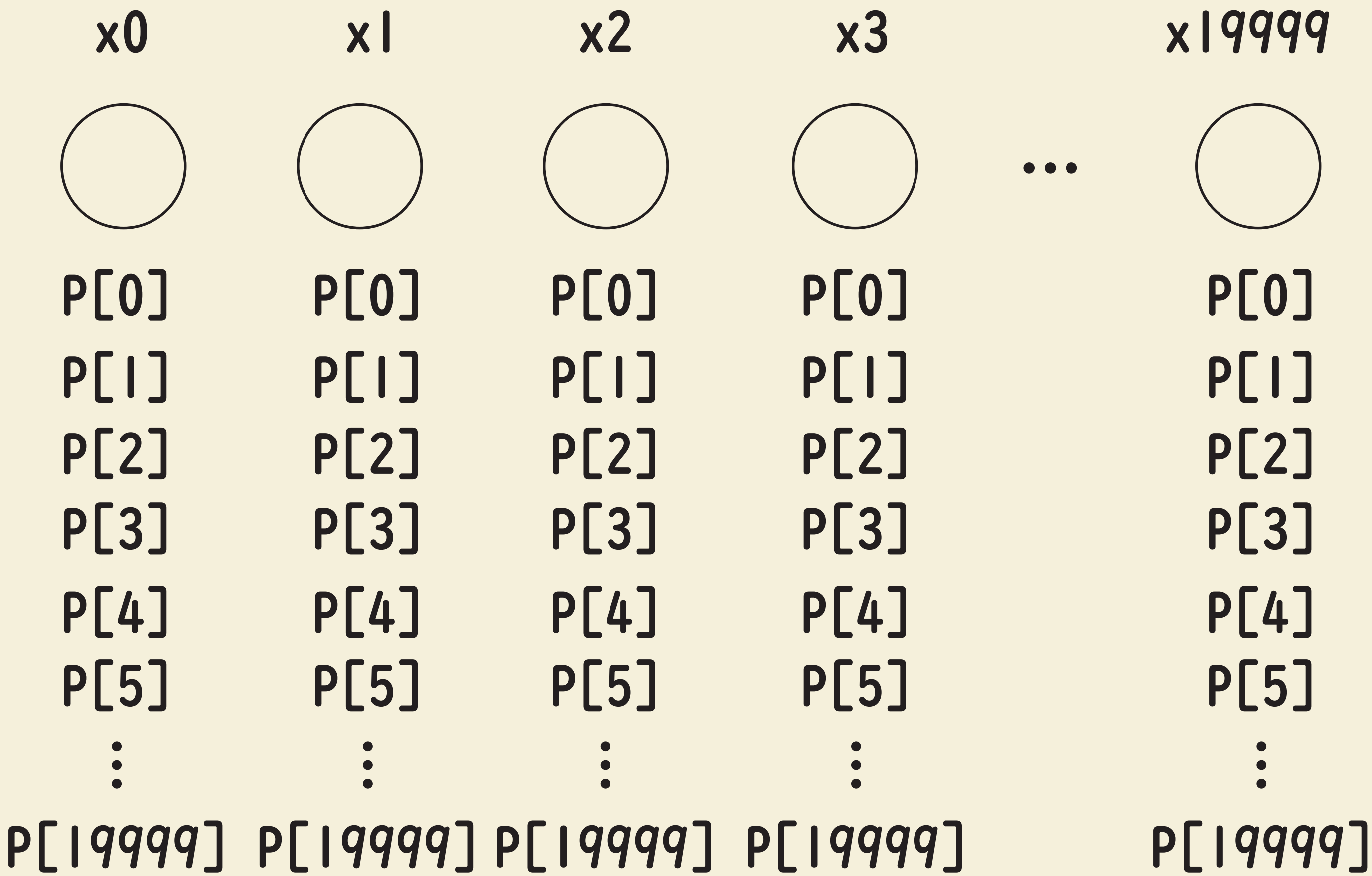
問題概要

N**N の計算量が必要で、コンテストに間に合いそうにない。

gcd を考慮しない場合

```
(0*P[0] + 1*P[0] + 2*P[0] + ... + 19999*P[0]) +
(0*P[0] + 1*P[0] + 2*P[0] + ... + 19999*P[1]) + ...
(0*P[19999] + 1*P[19999] + 2*P[19999] + ...)
```

gcd を考慮しない場合



(0*P[c[0]] + 1*P[c[1]] + 2*P[c[2]] + ... + 19999*P[c[19999]])

gcd を考慮しない場合

x0	x1	x2	x3		x19999
	P[2]			…	
P[0]	P[0]	P[0]	P[0]		P[0]
P[1]	P[1]	P[1]	P[1]		P[1]
P[2]	P[2]	P[2]	P[2]		P[2]
P[3]	P[3]	P[3]	P[3]		P[3]
P[4]	P[4]	P[4]	P[4]		P[4]
P[5]	P[5]	P[5]	P[5]		P[5]
⋮	⋮	⋮	⋮		⋮
P[19999]	P[19999]	P[19999]	P[19999]		P[19999]

x1 に P[2] が入るパターン数は、N**(N-1)

答え += P[2] * N**(N-1)

gcd を考慮しない場合

x0	x1	x2	x3	…	x19999
			P[5]	…	
P[0]	P[0]	P[0]	P[0]		P[0]
P[1]	P[1]	P[1]	P[1]		P[1]
P[2]	P[2]	P[2]	P[2]		P[2]
P[3]	P[3]	P[3]	P[3]		P[3]
P[4]	P[4]	P[4]	P[4]		P[4]
P[5]	P[5]	P[5]	P[5]		P[5]
⋮	⋮	⋮	⋮		⋮
P[19999]	P[19999]	P[19999]	P[19999]		P[19999]

x3 に P[5] が入るパターン数は、N**(N-1)

答え += P[5] * 3 * N**(N-1)

gcd を考慮しない場合

```
答え = (P[0] * 0 + P[0] * 1 + P[0] * 2 + ... + P[0] * 19999) * N**(N-1) +
(P[1] * 0 + P[1] * 1 + ... + P[1] * 19999) * N**(N-1) +
...
(P[19999] * 0 + ... + P[19999] * 19999) * N**(N-1)

∴
答え = (P[0] + ... + P[19999]) * (N * (N-1)/2) * (N**(N-1))
```

gcd を考慮する場合

gcd が 1 のときの答え +

gcd が 2 のときの答え +

gcd が 3 のときの答え +

...+

gcd が 19999 のときの答え

のように、分解して考える。

gcd を考慮する場合： gcd が 1 のものの個数？

小さなケースで考えてみる

(1,1), (1,2)

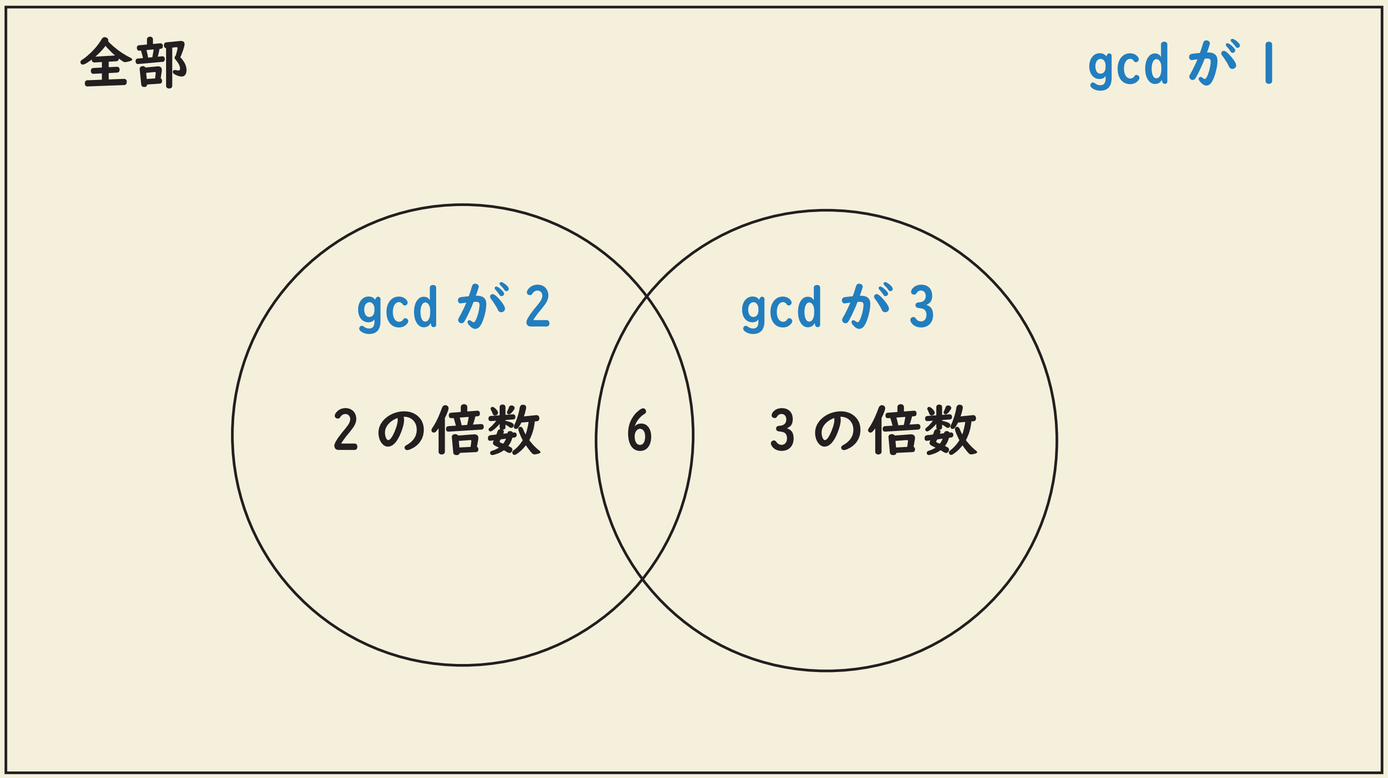
(2,1), (2,2)

gcd が 1 のもの = 全部 - 要素がすべて 2 の倍数のもの

[(1,1), (1,2), (2,1), (2,2)] - [(2,2)]

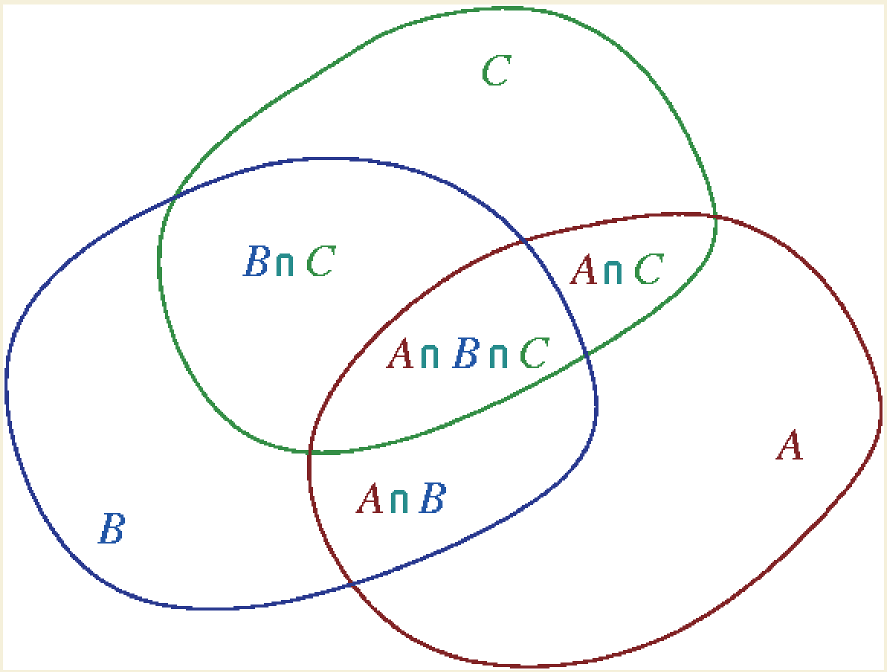
gcd を考慮する場合： gcd が 1 のものの個数？

gcd が 1 ＝ 全部 − 2 の倍数 − 3 の倍数 ＋ 6 の倍数



gcd を考慮する場合：gcd が1のものの個数？

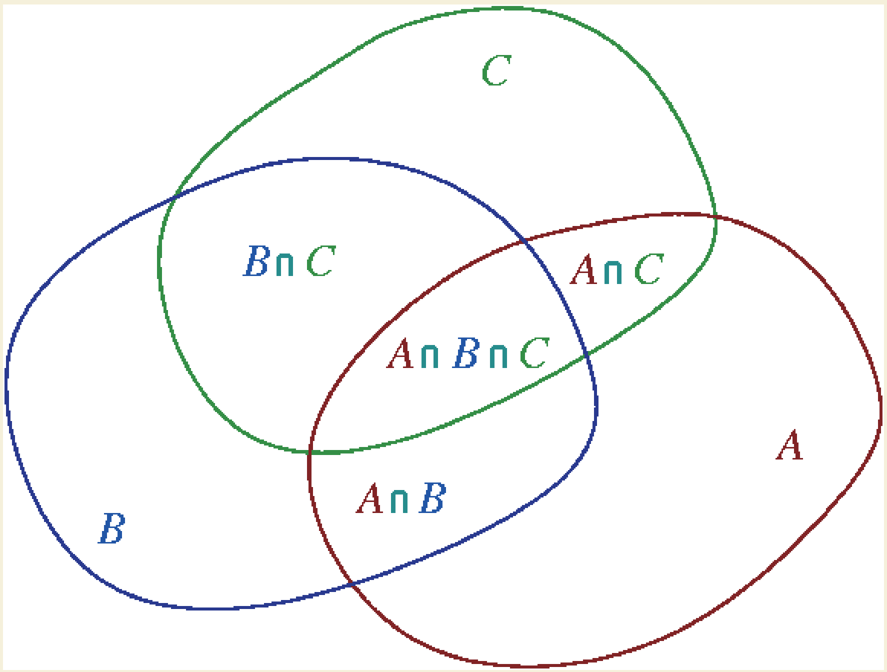
$$
\begin{aligned}
|A \cup B \cup C| = &|A| + |B| + |C| \\
&- |A \cap B| - |B \cap C| - |C \cap A| \\
&+ |A \cap B \cap C|
\end{aligned}
$$



引用：https://ja.wikipedia.org/wiki/%E5%8C%85%E9%99%A4%E5%8E%9F%E7%90%86

gcd を考慮する場合：gcd が 1 のものの個数？

$$
\begin{aligned}
|A \cup B \cup C| = & |A| + |B| + |C| \\
& - |A \cap B| - |B \cap C| - |C \cap A| \\
& + |A \cap B \cap C|
\end{aligned}
$$

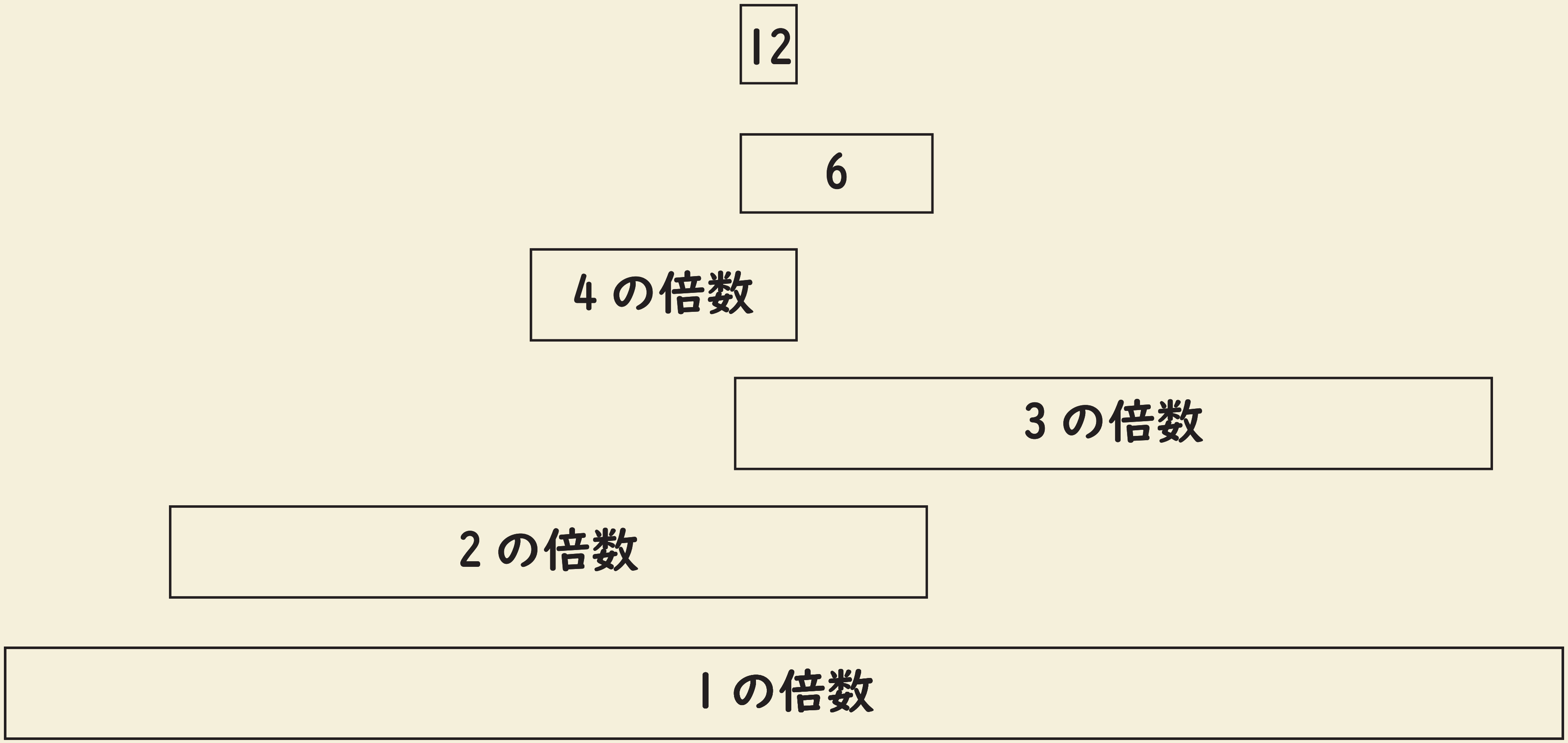


偶数のときがマイナスで

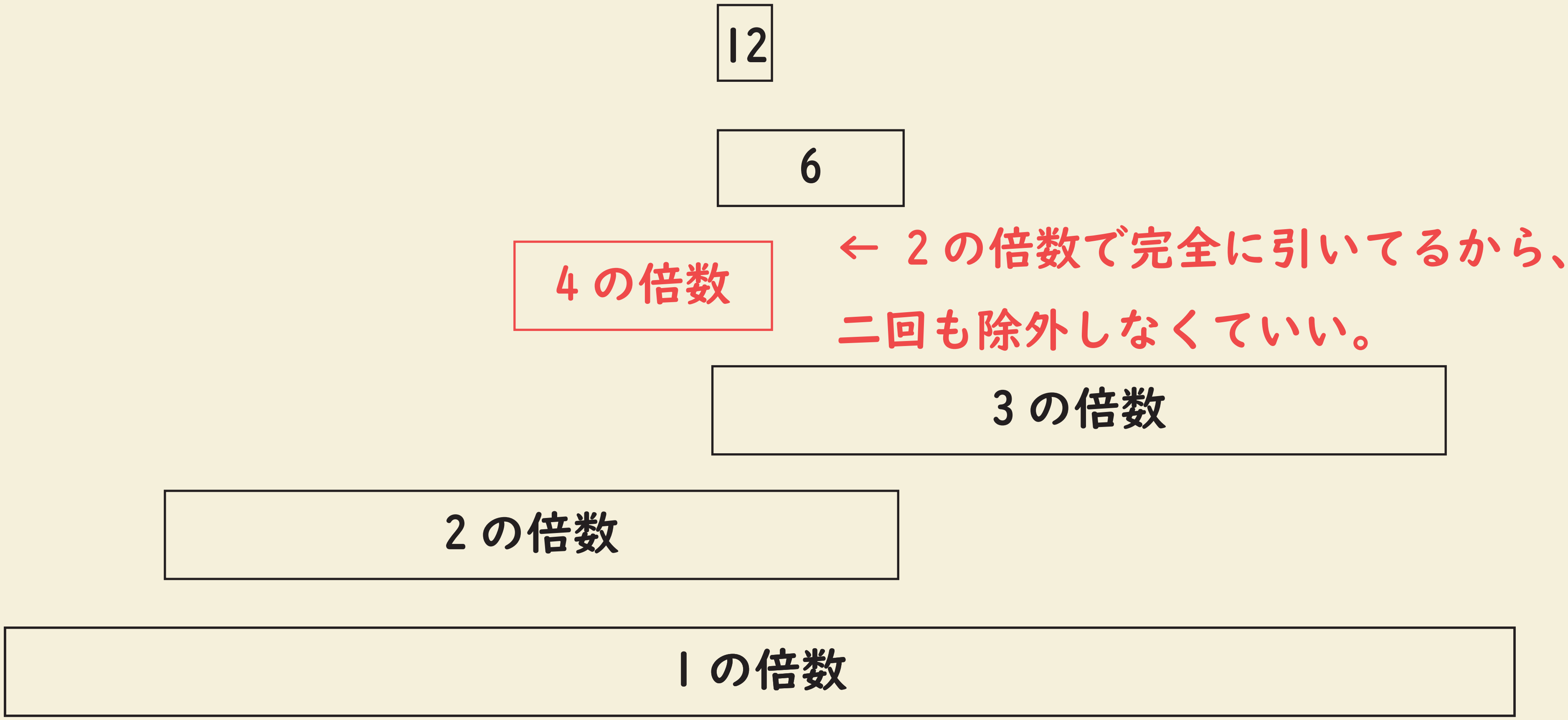
奇数のときがプラス

引用：https://ja.wikipedia.org/wiki/%E5%8C%85%E9%99%A4%E5%8E%9F%E7%90%86

gcd を考慮する場合：gcd が 1 のものの個数？



gcd を考慮する場合：4 の倍数を考慮する？



gcd を考慮する場合： 12 の倍数を考慮する？

← 6 の倍数で完全に引いてるから、
二回も除外しなくていい。

12

6

4 の倍数

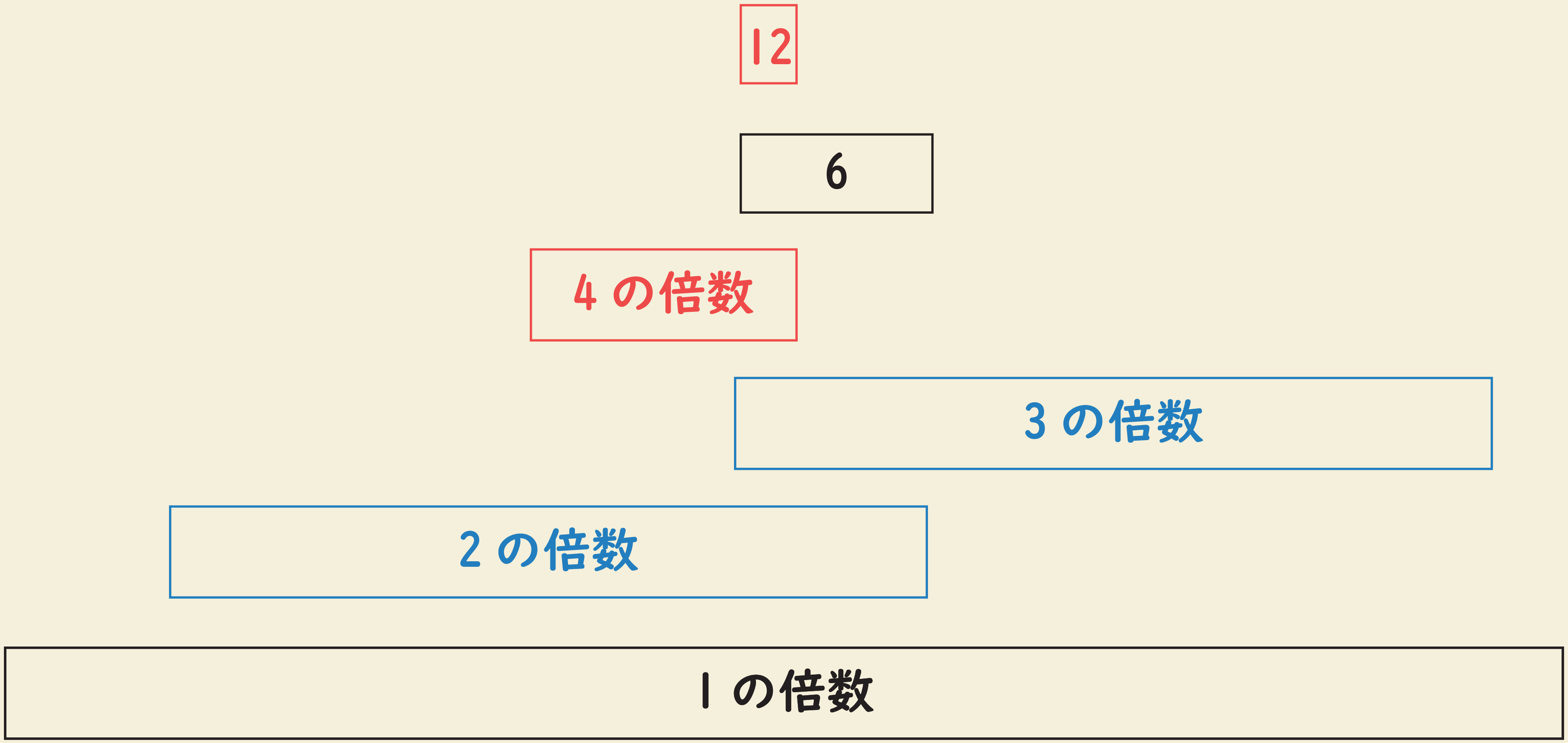
3 の倍数

2 の倍数

1 の倍数

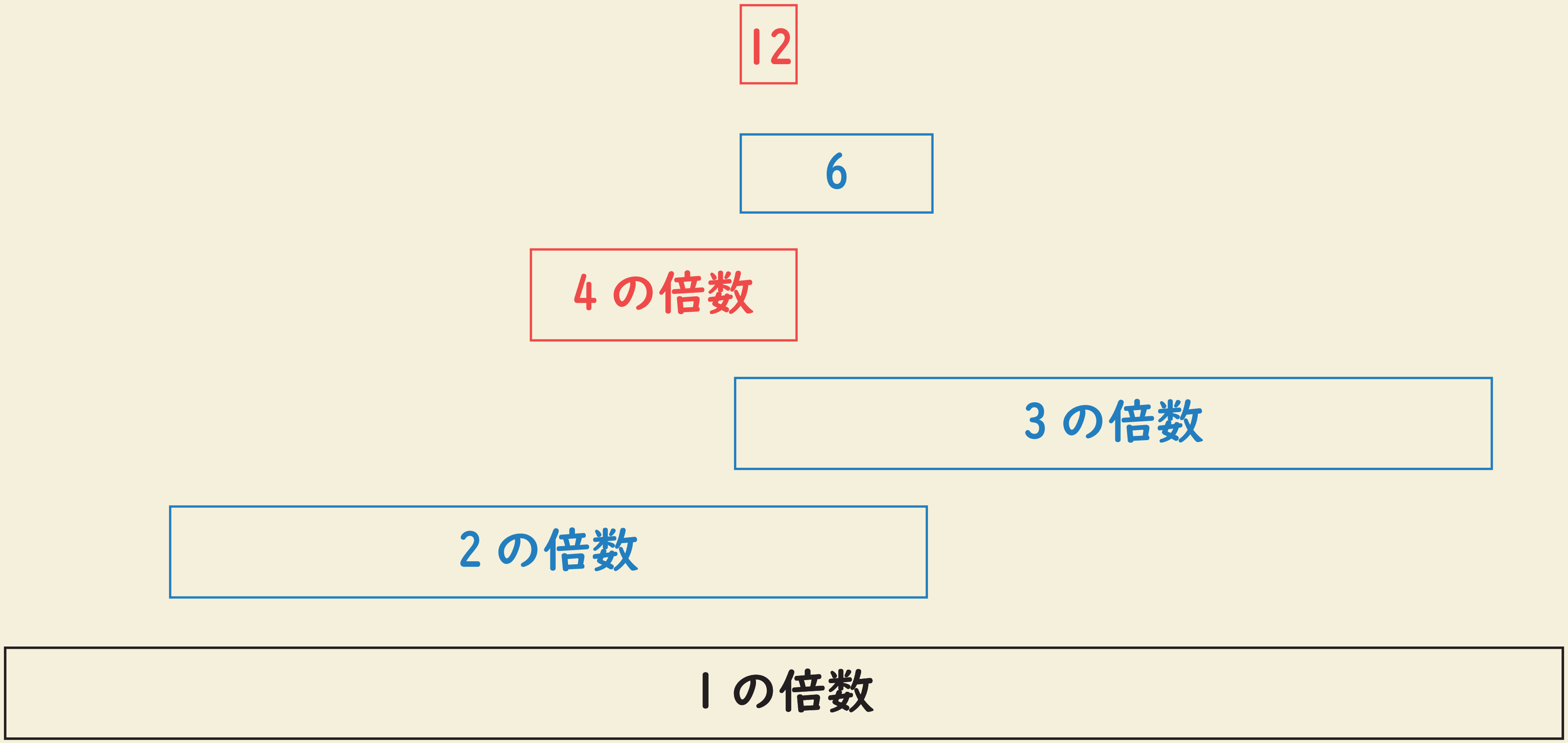
紅の魔術工房
Atelier kurenaif

gcd を考慮する場合： 2 と 3?



2 の倍数や 3 の倍数など、ずれている者同士で計算することに意味がある。

gcd を考慮する場合：6?



ダブルカウント防止のために、2 の倍数 +3 の倍数 -6 の倍数をするために
6 は必要。

gcd を考慮する場合： 考慮しない者たちは？

$4 = 2^2$

$12 = 2^2 * 3$

のように、2 乗以上のものが含まれていると、

すでに計算されていることになる。

（それより上位のものが存在していて、それによりすでに計算済みになっている）

12

4 の倍数

2 の倍数

gcd を考慮する場合：足し算と引き算は？

$$
\begin{aligned}
|A \cup B \cup C| =& |A| + |B| + |C| \\
& - |A \cap B| - |B \cap C| - |C \cap A| \\
& + |A \cap B \cap C|
\end{aligned}
$$

2 の倍数：足し算

3 の倍数：足し算

6 の倍数 =2*3 の倍数：引き算

素因数分解して、

偶数個なら足し算

奇数個なら引き算

のようにしてやれば良い。

紅の魔術工房
Atelier kurenaif

メビウス関数

実はこれはメビウス関数という名前がついている。

0 を含めない自然数において、メビウス関数 $\mu(n)$ は全ての自然数 n に対して定義され、n を素因数分解した結果によって -1、0、1 のいずれかの値をとる。

メビウス関数は次のように定義される（ただし 1 は 0 個の素因数を持つと考える）：

- $\mu(n) = 0$ （n が平方因子を持つ（1以外の平方数で割り切れる）とき）
- $\mu(n) = (-1)^k$ （n が相異なる k 個の素因数に分解されるとき）
  - n が相異なる偶数個の素数の積ならば $\mu(n) = 1$
  - n が相異なる奇数個の素数の積ならば $\mu(n) = -1$

https://ja.wikipedia.org/wiki/%E3%83%A1%E3%83%93%E3%82%A6%E3%82%B9%E9%96%A2%E6%95%B0

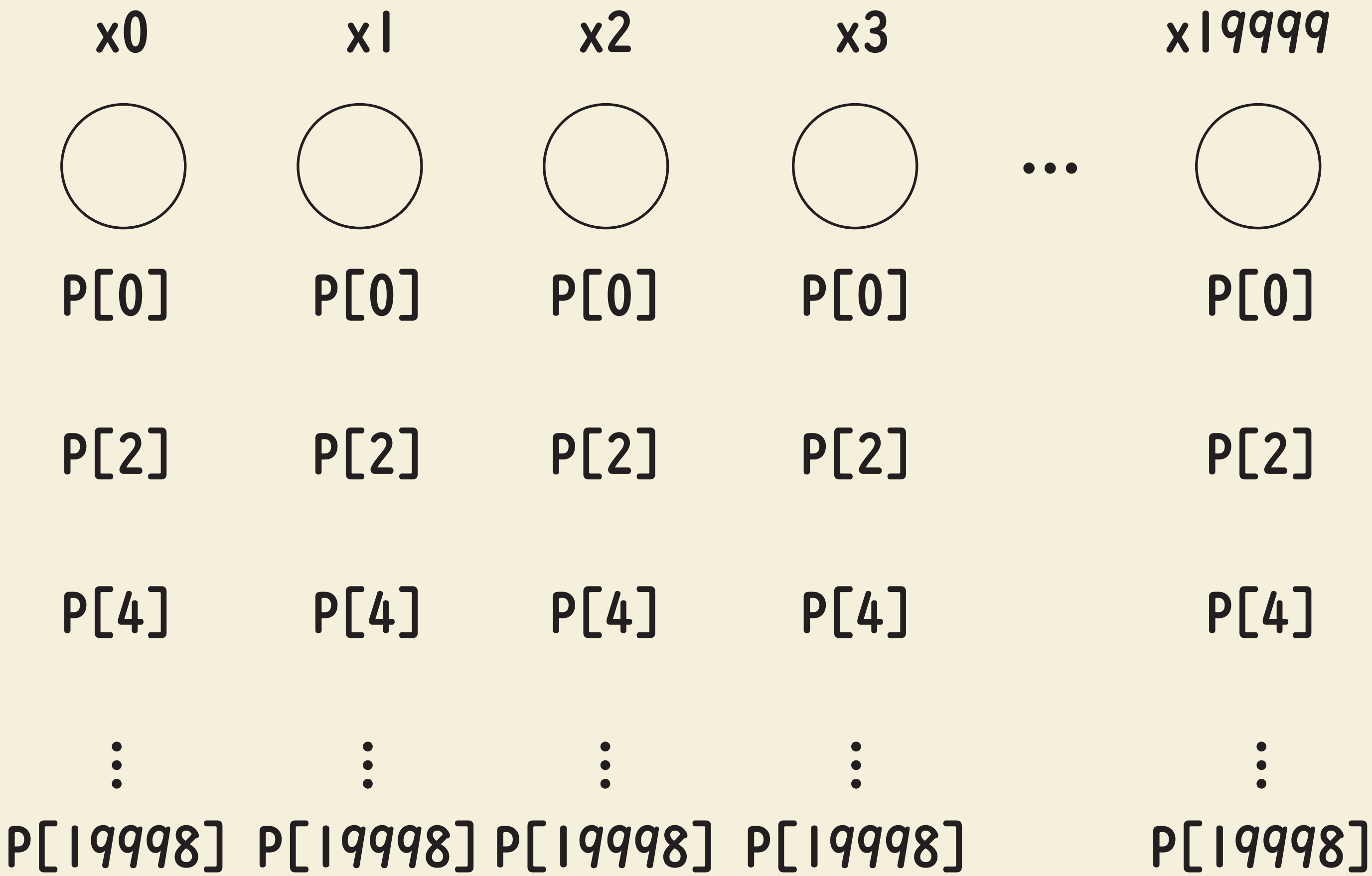
最終的な計算方法

答えの中で gcd が 1 となるものは ...

i = 1~19999

ans += μ(i) * ((i の倍数だけで選んだときの gcd を考慮しない総和))

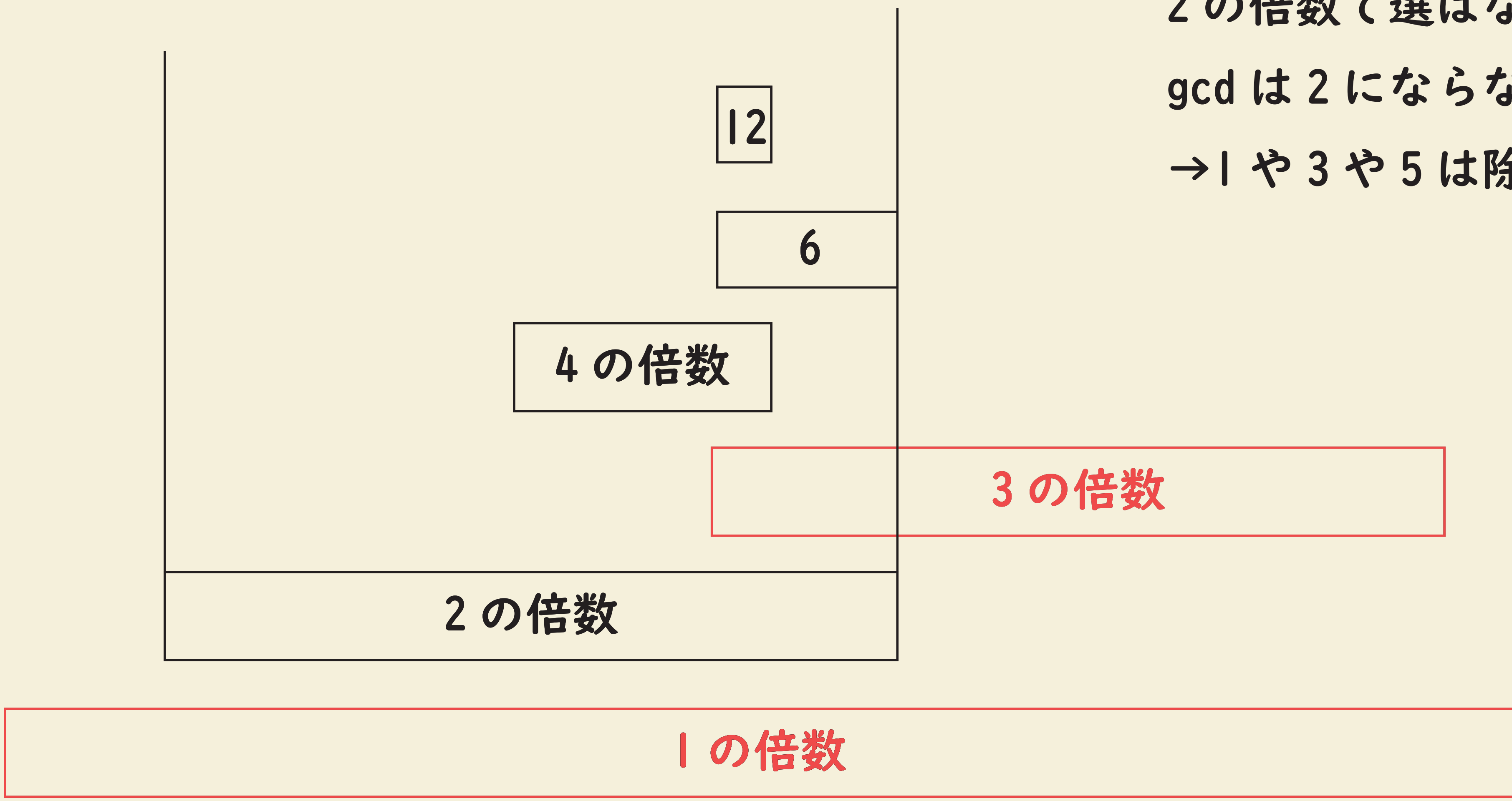
2の倍数の例



やり方はおなじ。

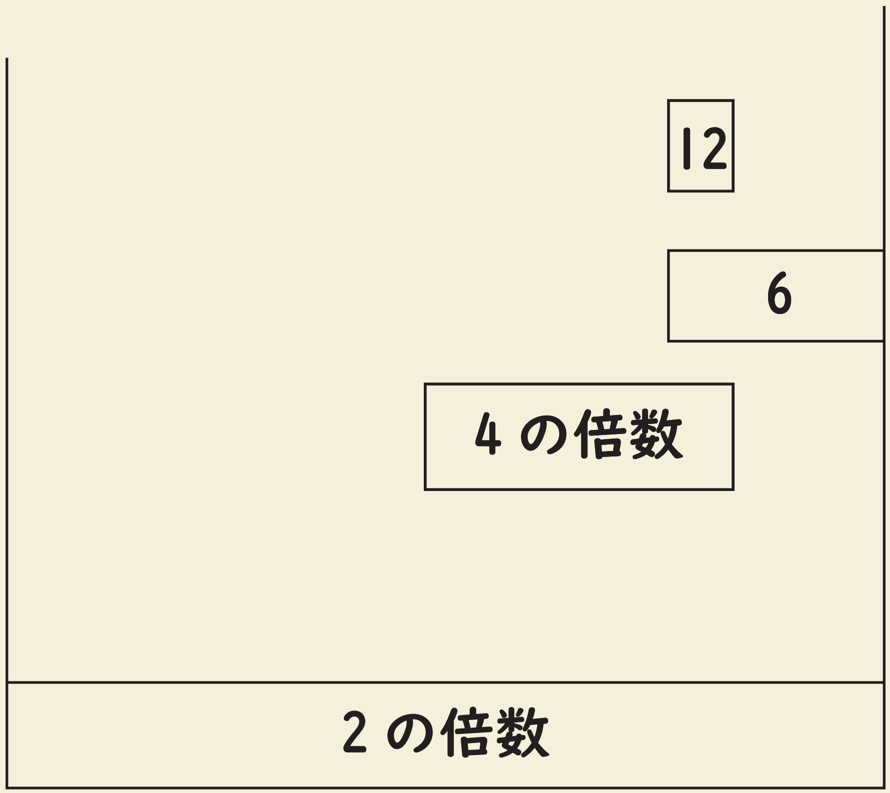
hoge * P[i] * (N/2) ** (N-1)

答えの中で gcd が 2 となるものは ...



2 の倍数で選ばなければ
gcd は 2 にならない。
→1 や 3 や 5 は除外

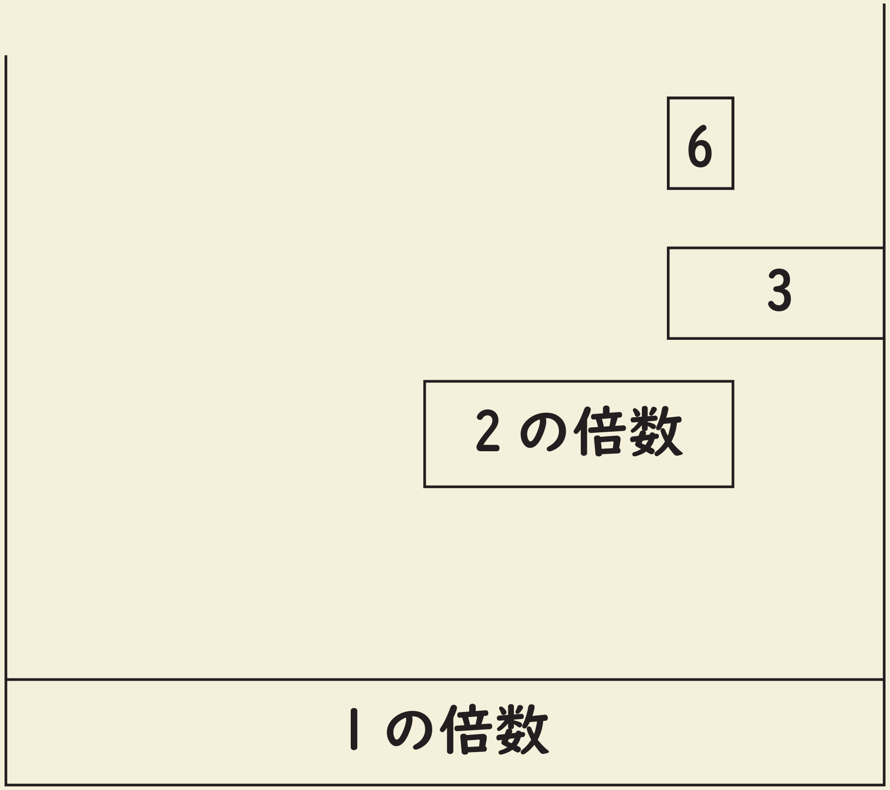
答えの中で gcd が 2 となるものは ...



4 の倍数の分は
まだ取り除かれていないので
考慮が必要。
12 も同様に考慮が必要。

2 の倍数だけで考えているので
全体で 2 で割って考える。

答えの中で gcd が 2 となるものは ...



4 の倍数の分は
まだ取り除かれていないので
考慮が必要。
12 も同様に考慮が必要。

2 の倍数だけで考えているので
全体で 2 で割って考える。

あとは gcd が 1 のときと同様。

最終的なアルゴリズム

最終的なアルゴリズム

g = 1~19999（求めたいgcd）

i = g ~ 19999（i は g の倍数でなければならないので g ずつで OK）

ans += g * μ(i/g) * ((i の倍数だけで選んだときの gcd を考慮しない総和）

O(Nlog(N)) ... かな？

多倍長整数重そうなので、C++ で実装。