包除原理の数え方

Counting with PIE: The Principle of Inclusion-Exclusion

Mizar/みざー

VRC競プロ部 @ VRC技術・学術系イベントHUB × #Vketステージ 2026-07-22

包除原理で解く部分問題

有限集合 U とその部分集合 $A_1, A_2, \ldots, A_M$ が与えられる。つまり、各 $i = 1, 2, \ldots, M$ について、 $A_i \subseteq U$ である。

以下は定数時間で計算できるとする。

- 集合 $X, Y \subseteq U$ に対する $Z = X \cap Y$ ： 2 つの集合の共通部分の集合 Z を求める
- 集合 $X \subseteq U$ に対する $C = \mathrm{count}(X)$ ： 集合 X の要素数 C を求める
- 整数 A, B に対する $A + B, A - B$ ： 2 つの整数の和と差を求める

求めたいのは、 $A_1, A_2, \ldots, A_M$ のうち、少なくとも 1 つに入る要素の数、つまり、$A_1, A_2, \ldots, A_M$ の和集合の要素数である。

$$|A_1 \cup A_2 \cup \cdots \cup A_M| = |\{x \in U \mid \exists i \in \{1, \ldots, M\},\ x \in A_i\}|$$

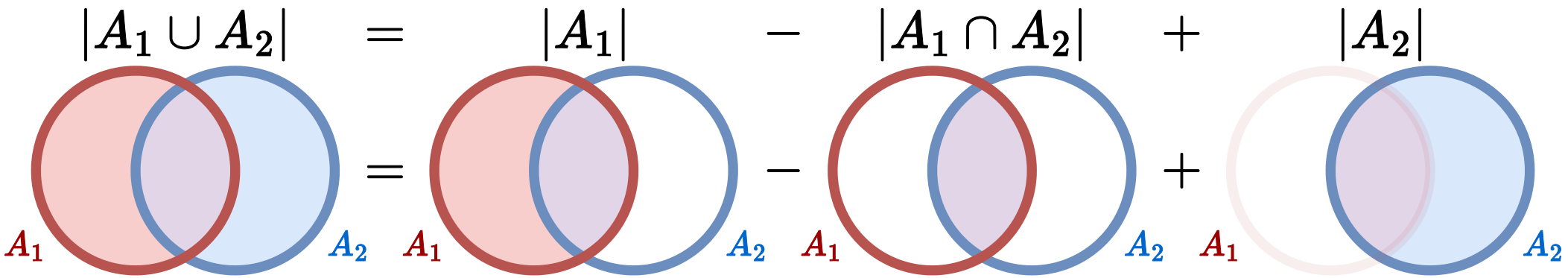
制約： $1 \leq M \leq 20$

2 個の集合を数える

A_1 と A_2 の共通部分に入っているものは、単純に足すと 2 回数えてしまう。

共通部分を 1 回引く。

$$|A_1 \cup A_2| = |A_1| - |A_1 \cap A_2| + |A_2|$$

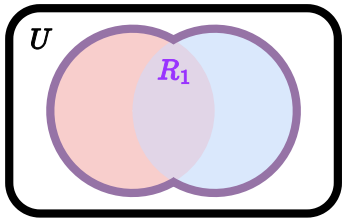


複数の集合の和集合の要素数: 再帰分解で考える

求めたい値: $|R_1|$

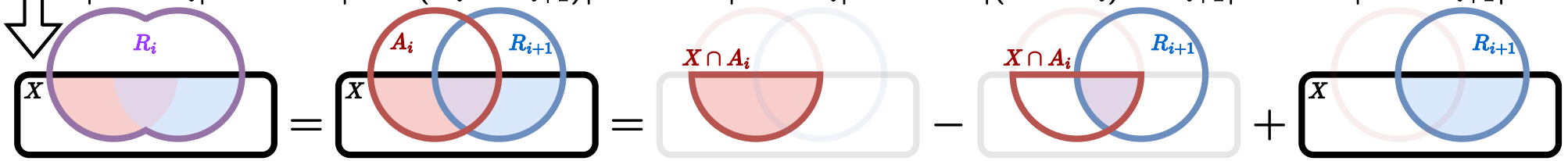
$R_1 \subseteq U \Longrightarrow |R_1| = |U \cap R_1|$

$$
\begin{aligned}
R_1 &= A_1 \cup A_2 \cup A_3 \cup \cdots \cup A_M \\
&\vdots \\
R_i &= A_i \cup A_{i+1} \cup A_{i+2} \cup \cdots \cup A_M \\
R_{i+1} &= A_{i+1} \cup A_{i+2} \cup \cdots \cup A_M \\
&\vdots \\
R_M &= A_M \\
R_{M+1} &= \varnothing
\end{aligned}
$$



R_1 を含むような集合 U があれば、
同形の式で始められる
（もしくは X に U を渡さない形で展開）

$$
|X \cap R_i| = |X \cap (A_i \cup R_{i+1})| = |X \cap A_i| - |(X \cap A_i) \cap R_{i+1}| + |X \cap R_{i+1}|
$$



$R_i = A_i \cup R_{i+1}$
R_i から A_i を括り出す

1 回毎に 2 分岐するため、
$X \cap A_i$ を求める回数は $(2^M - 1)$

R_{i+1} が空 $\varnothing$ になるまで帰納的に処理

$$
R_{i+1} = \varnothing \Longrightarrow |(X \cap A_i) \cap R_{i+1}| = |X \cap R_{i+1}| = 0
$$

選択優先 DFS の擬似コード（再帰版）

再帰版では、次の分解をそのまま計算している。

$$\left|X \cap R_i\right| = \left|X \cap (A_i \cup R_{i+1})\right| = \left|X \cap A_i\right| - \left|(X \cap A_i) \cap R_{i+1}\right| + \left|X \cap R_{i+1}\right|$$

```
def dfs_rec(X: Set, A: Ref[List[Set]], i: int):  # A は集合リストの参照、X は共通部分、i は次に分解する添字
    acc = 0  # この状態以下の寄与
    if i < len(A):  # まだ見ていない集合がある場合
        Y = intersect(X, A[i])  # A[i] を重ねたときの共通部分 X ∩ A[i]
        acc += count(Y)  # A[i] 側を足す
        acc -= dfs_rec(Y, A, i + 1)  # Y と後続側の重なりを引く
        acc += dfs_rec(X, A, i + 1)  # A[i] を重ねずに後続側へ進む
    return acc

# 集合全体を使う場合 -- 積集合: 2^M-1回、count: 2^M-1回
ans = dfs_rec(集合全体, A, 0)

# 集合全体を使わない場合 -- 積集合: 2^M-M-1回、count: 2^M-1回
ans = sum((count(A[i]) - dfs_rec(A[i], A, i + 1)) for i in range(len(A)))
```

選択優先 DFS の擬似コード（非再帰版）

```
# 空集合による枝刈り有り -- 最大で 積集合: 2^M-M-1回、count: 2^M-1回
def dfs_nonrec(A: Ref[List[Set]]):  # A は集合リストの参照
    acc = 0
    for i in range(len(A)):
        acc += count(A[i])  # 各集合に対して、まずはその集合の要素数を足す
        stack = [(A[i], i + 1, -1)]  # stackの要素は (現在の共通部分, 次に見る添字, 次に重ねる符号)
        while stack:  # 未処理の状態がある間くり返す
            (X, j, k) = stack.pop()
            if j < len(A):  # まだ見ていない集合がある場合
                stack.append((X, j + 1, k))  # A[j] を重ねない側を先入れ(後回し)
                Y = intersect(X, A[j])  # A[j] を重ねたときの共通部分 X ∩ A[j]
                cnt = count(Y)  # A[j] を重ねた共通部分の要素数
                if cnt == 0:  # cnt == 0 なら共通部分は空集合で、これ以上重ねても寄与はない
                    continue
                acc += k * cnt  # A[j] を重ねた共通部分を符号 k で足す
                stack.append((Y, j + 1, -k))  # A[j] を重ねた側を後入れ(先に処理する)
    return acc

ans = dfs_nonrec(A)
```

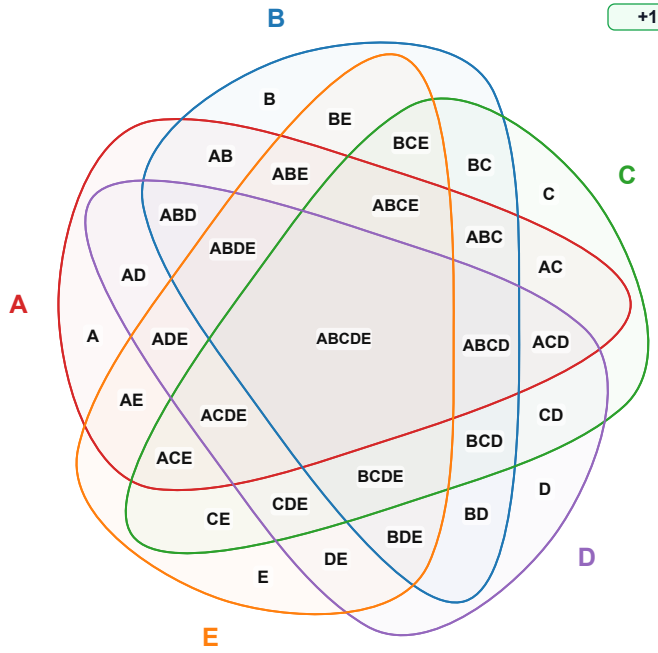
包除原理の数え方

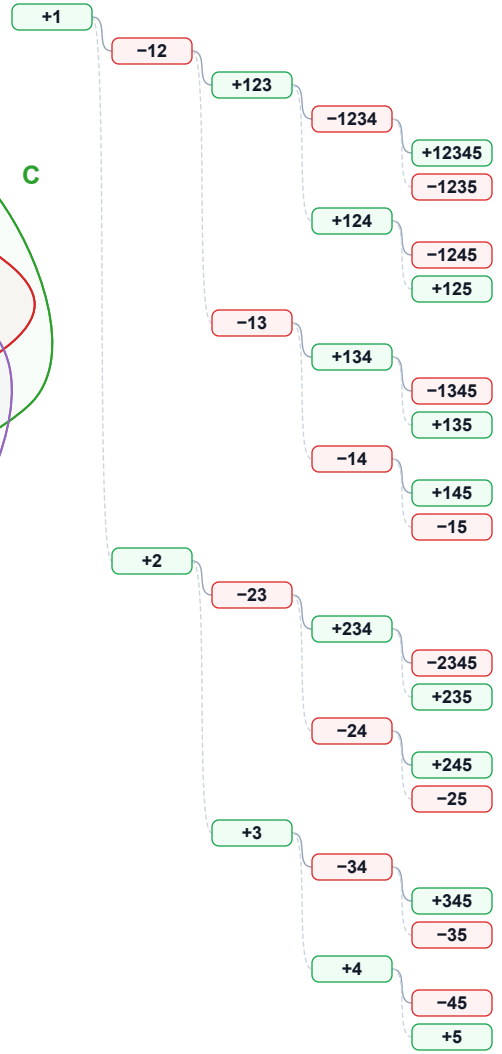
$M = 5$ の処理順

$A_{123} = A_1 \cap A_2 \cap A_3$ のように略記する。

$$
\begin{aligned}
&|A_1 \cup A_2 \cup A_3 \cup A_4 \cup A_5| \\
&= |A_1| - |A_{12}| + |A_{123}| - |A_{1234}| \\
&\quad + |A_{12345}| - |A_{1235}| + |A_{124}| - |A_{1245}| \\
&\quad + |A_{125}| - |A_{13}| + |A_{134}| - |A_{1345}| \\
&\quad + |A_{135}| - |A_{14}| + |A_{145}| - |A_{15}| + |A_2| - |A_{23}| \\
&\quad + |A_{234}| - |A_{2345}| + |A_{235}| - |A_{24}| + |A_{245}| - |A_{25}| \\
&\quad + |A_3| - |A_{34}| + |A_{345}| - |A_{35}| + |A_4| - |A_{45}| + |A_5|
\end{aligned}
$$

全部で $2^5 - 1 = 31$ 個の空でない共通部分を見ている。





包除原理の項の奇偶

$$
\begin{aligned}
&|A_1 \cup A_2 \cup A_3 \cup A_4 \cup A_5| \\
&= |A_1| - |A_{12}| + |A_{123}| - |A_{1234}| + |A_{12345}| - |A_{1235}| + |A_{124}| - |A_{1245}| + |A_{125}| - |A_{13}| \\
&\quad + |A_{134}| - |A_{1345}| + |A_{135}| - |A_{14}| + |A_{145}| - |A_{15}| + |A_2| - |A_{23}| + |A_{234}| - |A_{2345}| \\
&\quad + |A_{235}| - |A_{24}| + |A_{245}| - |A_{25}| + |A_3| - |A_{34}| + |A_{345}| - |A_{35}| + |A_4| - |A_{45}| + |A_5| \\
&= |A_1| + |A_2| + |A_3| + |A_4| + |A_5| \\
&\quad - |A_{12}| - |A_{13}| - |A_{14}| - |A_{15}| - |A_{23}| - |A_{24}| - |A_{25}| - |A_{34}| - |A_{35}| - |A_{45}| \\
&\quad + |A_{123}| + |A_{124}| + |A_{125}| + |A_{134}| + |A_{135}| + |A_{145}| + |A_{234}| + |A_{235}| + |A_{245}| + |A_{345}| \\
&\quad - |A_{1234}| - |A_{1235}| - |A_{1245}| - |A_{1345}| - |A_{2345}| \\
&\quad + |A_{12345}|
\end{aligned}
$$

重ねた集合の数で分けると、奇数のときは足し、偶数のときは引く。

M 個の集合の包除原理

$$
\begin{aligned}
&|A_1 \cup A_2 \cup A_3 \cup A_4 \cup A_5| \\
&= |A_1| + |A_2| + |A_3| + |A_4| + |A_5| \\
&\quad - |A_{12}| - |A_{13}| - |A_{14}| - |A_{15}| - |A_{23}| - |A_{24}| - |A_{25}| - |A_{34}| - |A_{35}| - |A_{45}| \\
&\quad + |A_{123}| + |A_{124}| + |A_{125}| + |A_{134}| + |A_{135}| + |A_{145}| + |A_{234}| + |A_{235}| + |A_{245}| + |A_{345}| \\
&\quad - |A_{1234}| - |A_{1235}| - |A_{1245}| - |A_{1345}| - |A_{2345}| \\
&\quad + |A_{12345}|
\end{aligned}
$$

愚直に計算すると積集合の回数は以下の通り。

$$\sum_{i=1}^{M} (i-1)\binom{M}{i} = (M-2)2^{M-1} + 1$$

$$\left| \bigcup_{i \in \{1,\ldots,M\}} A_i \right| = \sum_{\varnothing \neq I \subseteq \{1,\ldots,M\}} (-1)^{|I|+1} \left| \bigcap_{i \in I} A_i \right|$$

$$= \sum_i |A_i| - \sum_{i<j} |A_i \cap A_j| + \sum_{i<j<k} |A_i \cap A_j \cap A_k| - \cdots$$

回転対称な n-集合 Venn 図と素数

n 個の集合を $360^\circ/n$ ずつ回転させて配置し、回転によって集合が巡回的に入れ替わる Venn 図を考える。

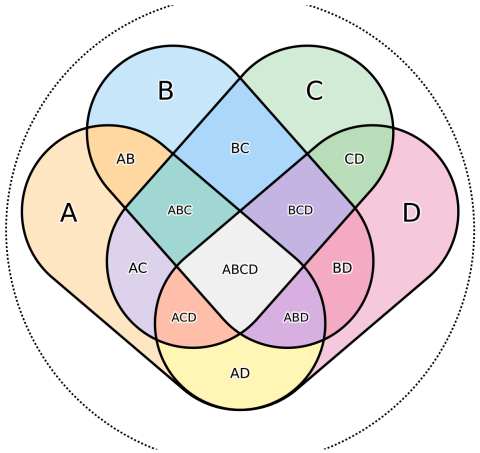
ちょうど k 個の集合に属する atom は、属する集合の選び方に対応するため ${}_nC_k$ 個存在する。一方、回転対称な図では、これらの atom は回転によって n 個ずつ同じ軌道を作る。したがって、すべての $1 \leq k < n$ に対して n が ${}_nC_k$ の約数であることが必要である。

n が素数なら、 $1 \leq k < n$ に対して ${}_nC_k$ の分子は n を因数に持ち、分母は持たないので、この整除条件は常に成立する。

反対に n が合成数で、 p をその素因数の1つとすると、 n は ${}_nC_p$ の約数ではない。そのため、ちょうど p 個の集合に属する atom を n 個ずつ回転対称に配置できず、Venn 図は作れない。

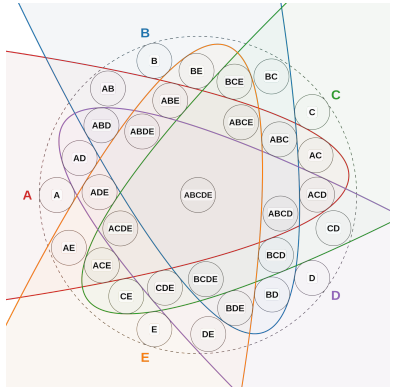
したがって、一般の単純閉曲線を許すと、

$\boxed{n\text{回回転対称な } n-\text{集合 Venn 図が存在する} \iff n\text{は素数}}$ となる。



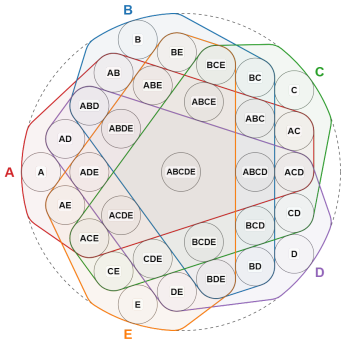
包除原理の数え方

単純閉曲線を原点中心に 5 回回転対称に配置して 5 集合Venn図を構成し、その 31 個の非空 atom のそれぞれに、原点を中心とする半径 1 の基準円の内部にも完全に収まる半径 R の小円を 1 個ずつ内包させ、これら 31 個の小円も原点中心に 5 回回転対称に配置するとき、R の最大値はいくつか。



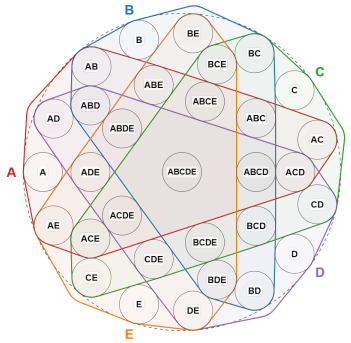
$R \approx 0.110598$

楕円を用いた5集合Venn図。大きく基準円をはみ出す構成例。



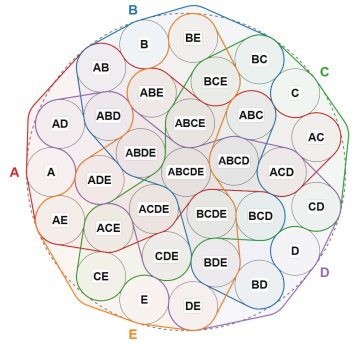
$R = \dfrac{\sqrt{5}}{16+\sqrt{5}} \approx 0.122618$

同じ向きの2つの正五角形上に20個と10個のatom代表点を均等に配置、内側の辺の延長が外側の辺の4等分点。



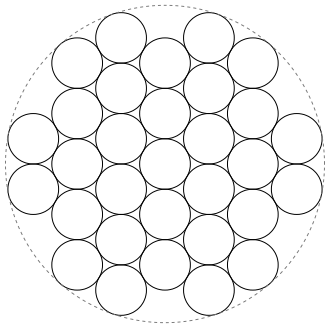
$R \approx 0.123261$

基準円側に寄せるatomを増やすことで小円半径 R が 約0.5% 改善。



$R \approx 0.154109$

5回回転対称に並べる集合図形に凹を許すことで小円半径 R が 約25% 改善。



$R \approx 0.158944541560$

Venn図と5回回転対称を課さない一般の31円充填では、既知の最良値は
$R \approx 0.158944541560$
である。

Binary String Generators

長さ N の文字列 $S_1, S_2, \ldots, S_M$ が与えられる。各 S_i は `0`, `1`, `?` からなる。

1 つの S_i を選び、各 `?` を独立に `0` または `1` に置き換え、 `0` と `1` からなる文字列を作る。この `0` と `1` からなる文字列の種類数を求める。

制約には、次のようなバリエーションがある。実行時間制限は 2 秒である。

Hard の K は、すべての S_i に含まれる `?` の文字数の総和である。

- (Easy): $1 \leq N \leq 10,\ 1 \leq M \leq 20$
- (Medium): $1 \leq N \leq 60,\ 1 \leq M \leq 20$
- (Hard): $1 \leq N \leq 60,\ 1 \leq M \leq 100,\ 0 \leq K \leq 400$

原問題: 競プロキャンプ2026関東 F: Binary String Generators

https://mofecoder.com/contests/kyoprocamp2026east/tasks/kyoprocamp2026east_f

入出力形式

入力は次の形で与えられる。

$$
\begin{array}{|l|}
\hline
N\ M \\
S_1 \\
S_2 \\
\vdots \\
S_M \\
\hline
\end{array}
$$

N は文字数、M は文字列本数。

各 S_i は長さ N の文字列。S_i の各文字は `0`, `1`, `?` のいずれか。

出力は、作れる完成済みの文字列の種類数を 1 行に出す。

サンプル 1

入力例: 文字数 $N = 3$, 文字列本数 $M = 3$, それぞれの文字列 S_i は $S_1 = \mathtt{000},\ S_2 = \mathtt{101},\ S_3 = \mathtt{1??}$。

```
3 3
000
101
1??
```

出力例: 5通り。

```
5
```

$$\left|\{\mathtt{000}, \mathtt{100}, \mathtt{101}, \mathtt{110}, \mathtt{111}\}\right| = 5$$

サンプル 2

入力例: 文字数 $N = 5$, 文字列本数 $M = 5$, それぞれの文字列 S_i は
$S_1 = 0000?$, $S_2 = 000?0$, $S_3 = 00?00$, $S_4 = 0?000$, $S_5 = ?0000$。

```
5 5
0000?
000?0
00?00
0?000
?0000
```

出力例: 6通り。

```
6
```

$$\left|\{00000, 00001, 00010, 00100, 01000, 10000\}\right| = 6$$

サンプル 3

入力例: `?` のみからなる長さ $N = 60$ の文字列 S_1 が $M = 1$ 本与えられる。

```
60 1
????????????????????????????????????????????????????????????
```

出力例: 115京2921兆5046億0684万6976通り。

```
11529215046068469 76
```

$$2^{60} = 1152921504606846976$$

サンプル 4

入力例: VRCPROCON = VRC競プロ部。

```
59 9
11111111111111111111111111111111111111111111111111111
?1?0???0?0000???0000?0000??0000???000???0000??000??0???0?1?
11?0???0?0???0?0????0???0?0???0?0???0?0????0???0?00??0?11
?1?0???0?0???0?0????0???0?0???0?0???0?0????0???0?0?0?0?1?
11?0???0?0000??0????0000??0000??0???0?0????0???0?0??00?11
?1??0?0??0?0???0????0????0?0???0???0?0????0???0?0???0?1?
11??0?0??0??0??0????0????0??0??0???0?0????0???0?0???0?11
?1???0???0???0??0000?0????0???0??000???0000??000??0???0?1?
11111111111111111111111111111111111111111111111111111
```

出力例: 6兆4003億1512万5761通り。

```
6400315125761
```

サンプル 5

入力例: KYOPROCAMP 2026EAST = 競プロキャンプ2026関東。

```
59 13
1111111111111111111111111111111111111111111111111111111
0???0?0???0??000??0000??0000???000???0000??000??0???0?0000?
0??0???0?0??0???0?0???0?0???0?0???0?0?????0???0?00?00?0???0
000????0???0???0?0000??0000??0???0?0????00000?0?0?0?0000?
0??0????0???0???0?0?????0??0??0???0?0?????0???0?0???0?0????
0???0???0????000??0?????0???0??000???0000?0???0?0???0?0????
1111111111111111111111111111111111111111111111111111111
???????000???000???000???000??00000??000???0000?00000??????
??????0???0?0??00?0???0?0?????0?????0???0?0???????0????????
?????????0??0?0?0????0??0000??0000??00000??000????0????????
???????00???00??0??00???0???0?0?????0???0?????0???0????????
??????00000??000??00000??000??00000?0???0?0000????0????????
1111111111111111111111111111111111111111111111111111111
```

出力例: 114兆2041億0124万0833通り。

```
114204101240833
```

Medium の計算回数の目安

競技プログラミングでは、2 秒で処理できる回数には限りがある。

大ざっぱには、10^8 回程度なら間に合うことが多い。

一方で、10^9 回程度になると厳しいことが多い。

$$2^{30} \approx 10^9$$

$M \leq 20$ なら、重ね方は $2^{20} - 1 = 1{,}048{,}575$ 個。

これは 10^8 の約 $\dfrac{1}{95}$ である。

したがって、この方針は全探索の候補になる。

Hard の制約

最後に Hard を考える。

$$1 \le N \le 60, \qquad 1 \le M \le 100, \qquad 0 \le K \le 400$$

ただし、K はすべての S_i に含まれる `?` の文字数の総和である。

Medium と違って M が大きいので、2^M 通りの重ね方をすべて見ることはできない。

たとえば $M = 100$ では、次が成り立つ。

$$10^{30} < 2^{100}$$

10^8 回を大きく超える計算を 2 秒で行うことは現実的ではない。

したがって、Hard では別の方針が必要になる。

Hard の方針

Hard では、難しさが 2 つある。

- `?` が多い S_i は、作れる 01 文字列を列挙できない
- M が大きいので、すべての重ね方に包除原理を使えない

そこで、S_i を `?` が多い順に並べ、先頭を Heavy、残りを Light に分ける。

たとえば、`?` が多い方から 21 本を Heavy にする。

種類	特徴	処理
Heavy	`?` が多いが本数は少ない	包除原理
Light	`?` が少ない	01 文字列を列挙

Heavy 側を数える

Heavy 側は、Medium と同じく包除原理で数える。

空でない重ね方について、共通部分を作る。

矛盾しなければ、その共通部分のサイズを数える。

重ねた個数が奇数なら足す。

重ねた個数が偶数なら引く。

これで、Heavy のどれかから作れる 01 文字列の種類数が求まる。

Light 側を列挙する

Light は `?` が少ないので、実際に 01 文字列を列挙する。

ある Light パターン S に `?` が q 個あるとする。

その q 個の `?` に対して、`0` または `1` の割り当てをすべて試す。

列挙される個数は 2^q 個である。

列挙した 01 文字列を配列に入れ、`sort` して重複を消す。

`sort` すると同じ文字列が隣り合うので、前の要素と違うものだけを残せばよい。

Heavy と重なる Light を除く

答えは、**Heavy で作れる数に、Light で作れるが Heavy では作れない数を足して**求める。

Light から列挙した 01 文字列を x とする。

Heavy パターン S から x が作れる条件は、S の `0` と `1` の位置がすべて x と一致することである。

`?` の位置は一致条件に関係しない。

Heavy のどれかから作れる x はすでに数えているので、どの Heavy からも作れない x だけを Light 側として足す。

Hard の計算量

`?` が多い順に Heavy を取り、その本数を H、Light の列挙数を L、Light の `?` の最大個数と総数を B, R とする。

$$B \leq R \leq K - HB$$

$0 < a \leq b < B$ で $(2^a + 2^b) < (2^{a-1} + 2^{b+1})$ なので、`?` は可能な限り B 個ずつまとめたとき L が最大になる。 $B > 0$ なら、Light の列挙数 L は次のように抑えられる。

$$L \leq M - H + \left\lfloor \frac{R}{B} \right\rfloor (2^B - 1) + (2^{R \bmod B} - 1)$$

N ビットの演算を定数時間とみなすと、全体の計算量は次の通り。

$$O(2^H + HL + L \log L)$$

$H = 21$ の場合の最大ケース

Heavy 側の列挙数 2^H は次の通り。

$$2^{21} = 2{,}097{,}152$$

Light 側の列挙数を L とする。$H = 21,\ M = 100,\ K = 400$ として B を全探索すると、L の上界は $B = 17$ のとき最大になる。このとき $R \leq 400 - 21 \cdot 17 = 43$ なので、Light 側は `?` が $17, 17, 9, 0, \ldots, 0$ 個の場合で評価できる。

Light 側の列挙数 L と、Heavy との重複チェック回数 HL は次の通り。

$$L \leq 2 \cdot 2^{17} + 2^9 + 76 = 262{,}732$$

$$HL \leq 21 \cdot 262{,}732 = 5{,}517{,}372$$

Hard のまとめ

`?` が多いものは、01 文字列を列挙しない。

本数が多いものには、包除原理を使わない。

そのために、文字列を Heavy と Light に分ける。

- Heavy は本数が少ないので包除原理
- Light は `?` が少ないので列挙
- Light のうち Heavy でも作れるものは足さない

この分割が Hard の中心アイデアである。

Hard の詳細な計算量

`?` が多い方から、次の本数 H だけ Heavy を取る。

$$H = \min\left(\left\lfloor \sqrt{K} + \log_2\left(\max\left(\sqrt[4]{K}, 1\right)\right)\right\rfloor, M\right)$$

Light の `?` の最大個数と総数を B, R とする。 $B \leq R \leq K - HB$ であり、$B > 0$ のとき、Light の列挙数 L は次のように抑えられる。

$$L \leq M - H + \left\lfloor \frac{R}{B} \right\rfloor (2^B - 1) + (2^{R \bmod B} - 1) < M - H + \frac{2^{\sqrt{K}+1}}{\sqrt[4]{K}}$$

N ビットの演算を定数時間とみなすと、全体の計算量は次の通り。

$$O\left(\sqrt[4]{K} 2^{\sqrt{K}} + M\sqrt{K} + M \log M\right)$$

Hard の詳細計算量の命題

light 側の列 $l_i \in \mathbb{N}\ (0 \le i < M-H)$ と heavy 側の列 $h_i \in \mathbb{N}\ (0 \le i < H)$ が与えられているとする。

- $M, K, H, R, B, L \in \mathbb{N}, \quad s, r \in \mathbb{R}$
- $K = \sum_{0 \le i < M-H} l_i + \sum_{0 \le i < H} h_i$
- $K = s^4, \quad 0 < K \implies s = 2^r$
- $\min(M, \lfloor s^2 + r \rfloor) \le H \le M$
- $H = M \implies B = 0$
- $H < M \implies B = \max_{0 \le i < M-H} l_i$
- $0 < H \implies B \le \min_{0 \le i < H} h_i$
- $\sum_{0 \le i < M-H} l_i \le R \le K - HB$
- $L = \sum_{0 \le i < M-H} 2^{l_i}$

を満たすとする。（空集合の総和は 0 とする）

このとき、次が成り立つ。

1. heavy 側の下界から $HB \le \sum_{0 \le i < H} h_i$。
2. $B = 0$ ならば $L + H = M$。
3. $B > 0$ ならば、以下の不等式列が成り立つ。

$$
\begin{aligned}
L + H &\le M + \left\lfloor \frac{R}{B} \right\rfloor (2^B - 1) + (2^{R \bmod B} - 1) \\
&\le M + \frac{R}{B}(2^B - 1) \\
&< M + \left(\frac{s^4}{B} - s^2 - r + 1 \right)(2^B - 1) \\
&< M + 2^{s^2 - r + 1}.
\end{aligned}
$$