A. Announcements

A - Standing Sign 解説 by heno239

まず、京都大学を訪れるのは時刻Tの倍数の直前だけである、と考えて問題ありません。

例えば、$T = 5$ だったときに、時刻 12 に京都大学を訪れるよりも時刻 14.9 に京都大学を訪れる方がより多くの看板を見ることができます。(時刻 12 と時刻 14.9 の間では看板の撤去が行われないからです。)

そうすると、各 i について、時刻 $[i \times T, i \times (T+1)]$に看板が設置される場合は時刻 $i \times (T+1) - 0.1$ に京都大学を訪れ、そうでない場合は京都大学を訪れない、とするのが最適な京都大学の訪れ方の一つになります。

上記のことは、例えば $[S[i]/T]$ の種類数に一致します。

```
All solutions are from:
atcoder.jp/contests/kupc2021/editorial
```

B. Build The Grid / Sol.1

B - Painting with Many Orders 解説 by ≫yamunaku

$N = i$ について条件を満たすグリッドを G_i とします。

次のグリッドは G_1 です。

```
.
```

また、次のグリッドは G_2 のひとつです。

```
#.
..
```

G_i が存在したとき、G_{i+2} は次のようにして構成できます。

```
#.#####
.......
#.┌───┐
#.│   │
#.│G_i│
#.│   │
#.└───┘
```

例えば、上に従って G_6 を構成すると次のようになります。

```
#.####
......
#.#.##
#.....
#.#.#.
#.#...
```

B. Build The Grid / Sol.2

B - Painting with Many Orders 解説 by heno239

渦巻を作ります。

```
.
```

```
#.
..
```

```
.##
.#.
...
```

```
.###
.#..
.##.
....
```

```
.####
.#...
.#.#.
.###.
.....
```

```
.#####
.#....
.#.##.
.#..#.
.####.
......
```

C. Coins and Boxes

C - Gacha 解説 by nxteru

すべてのガチャとコインを通らなければなりません。またある地点について、座標 0 からその地点までの（コインの枚数）−（ガチャの個数）が負の場合には、上の値が 0 以上になる地点まで行ったあとまた戻ってくる必要があります。

ガチャまたはコインのある地点ごとに座標を区切って隣り合う地点同士の間を区間と呼びます。（コインの枚数）−（ガチャの個数）が負の区間については1.そこを通り過ぎて、2. また戻ってきて、3. その後また座標が正の方向に進む、ため区間の長さの 3 倍の移動距離が必要です。ただし1. のときに最も大きい座標まで到達していたなら 3. を省略することができるため区間の長さの 2 倍になります。

上のことから最適な行動はある最終地点 f があり、fより以前では負の区間は 3 回、正の区間は 1 回通る。f 以降では最後まで到達したあと f まで戻ってくるため、負の区間は 2 回、正の区間も 2 回通るという構造になります。

よって最終地点 f を全ての地点について全探索すればよいです。累積和を用いると最終地点を決め打ったときの移動距離を $O(1)$ で求めることができるので、全体で $O(N)$ で解くことができました。

D. Destructive Game

D - Stones 解説 by ≫moririn2528

解答

grundy 数を考えます。石の数 a_i の山、数 b_i が与えられたとき、この山の grundy 数は

- b_i が奇数のとき、$a_i \mod 2$
- b_i が偶数、$a_i \mod (b_i+1) = b_i$ のとき、2
- それ以外のとき、$(a_i \mod (b_i+1)) \mod 2$

となります。

道筋

2 人対称ゲームであるので、Grundy 数をまず疑います。パッと考えて規則性がわからなければ小さい数で実験しましょう。すると、0,1 が多く出てきて、2 がたまに出てくることがわかります。どういうときに 2 が出るのか、どういう規則性になっているのかを考えて、それをサッと証明できれば終了です。

解説

b_i が奇数のときは、b_i^k が奇数となるので、石の数が偶数のとき操作後は奇数に、奇数ならば偶数になります。よって、帰納的に石の数が x の時の grundy 数が $x \mod 2$ となることがわかります。

b_i が偶数のときは、書き出していくと $0, 1, 0, 1, \ldots, 0, 1, 2$、長さ b_i+1 の周期であるように見えます。これを示します。 石の数を x とすると、$0 \leq x \leq b_i - 1$ のときは操作が 1 減らすことのみであるので明らかです。また、$x = b_i$ のときも明らかです。 $b_i + 1 \leq x$ の時について、$b_i^k \mod (b_i+1)$ は $-1, 1$ のいずれかになります。また、$k = 0, 1$ の操作は可能なので、操作後の石の数を y とすると、$y \equiv x-1$ か $y \equiv x+1 \mod (b_i+1)$ となり、任意の方に操作可能です。よって、帰納的に示すことができます。 以上により、解答に書かれた、それぞれの山に対する grundy 数が証明されました。すべての山の grundy 数を xor した値が 0 でなければ先手の `Alice` が、0 のときは後手の `Bob` が勝ちます。

E. Edges, Colors and MST

E - PERMST 解説 by ≪yamunaku

辺 i の重みを w_i とします。 G の全域木 T に対して、次が成り立つことが知られています。

> 「T は G の最小全域木である」⇔「T に含まれない G の任意の辺 x をとると、T 上の u_x と v_x を結ぶパスに含まれる任意の辺 y について $w_y \leq w_x\ (*)$ 」

解法

赤い辺で構成された G の全域木を T とします。

w_i に 1 から M までの整数を 1 つずつあてはめて、条件を満たす $w(=p)$ の中で辞書順最小のものを作ることにします。

すべての w_i の値が決まるまで、以下の手続きを繰り返します。

手続き

id をまだ w_i の値を決めていない i の最小値、m をまだ使っていない整数の最小値とします。ここで、m 以上の整数はまだ使われていないことが、手続きの定義より帰納的に示せます。

$c_{id} = 1$ のとき（辺 id が赤い辺のとき）

$w_{id} = m$ としてもまだ条件を満たす w は存在します。例えば、まだ使っていない整数の中で小さいものをすべて赤い辺に、残ったものをすべて青い辺に割り当てればよいです。

$j < id$ について w_j の値は決まっているので、辞書順最小となる w は $w_{id} = m$ を満たします。

$c_{id} = 0$ のとき（辺 id が青い辺のとき）

T 上の u_{id} と v_{id} を結ぶパスに含まれる辺 y のなかで、w_y がまだ決まっていない y を昇順にならべて $y_1, y_2, \ldots, y_k$ とします。

$w_{y_i} = m + i - 1 (1 \leq i \leq k)$ および $w_{id} = m + k$ としてもまだ条件を満たす w は存在します。例えば、まだ使っていない整数の中で小さいものをすべて赤い辺に、残ったものをすべて青い辺に割り当てればよいです。

もし $w_{id} < m + k$ とすると、$w_{id} < w_{y_i}$ となる y_i が存在してしまうため、上の $(*)$ を満たさず T は全域木ではなくなってしまいます。

$j < id$ について w_j の値は決まっているので、辞書順最小となる w は $w_{y_i} = m + i - 1 (1 \leq i \leq k)$ および $w_{id} = m + k$ を満たします。

実装

上の解法を愚直に実装すると $O(MN)$ かかってしまいますが、Union Find などのデータ構造や “データ構造をマージする一般的なテク” というテクニックを用いると $O(M\alpha(N))$ や $O(M \log N)$ などで解くことができます。

F. Flatland Currency

F - One Yen Coin 解説 by nxteru

1 回の会計で 1 つの商品を購入するのが最適です。また持っている 1 円玉は支払わない方が良いです。よって A 円の商品は価値 $5 - A\%5$、重み $A + 5 - A\%5$ として、容量が $X - X\%5$ のナップザック問題に帰着できます。(値段が 5 の倍数の商品は無視します)

この問題では価値が 1〜4 と小さいことを利用して高速に解きます。各価値ごとに商品を重みの昇順にソートします。ここで各価値ごとに使う価値の合計を 12 で割った余りを決め打ちます。各価値ごとに下から決め打った余りの分だけ使うと、残りの商品は下から順に価値 12 ごとにまとめることができます。後は重さの小さい順に容量いっぱいまで価値 12 のまとまりを詰めていけばよいです。

価値 1〜4 それぞれ 12 で割った余りを決め打って $\frac{12}{1} \times \frac{12}{2} \times \frac{12}{3} \times \frac{12}{4}$ 通りを全探索すると $O(N logN)$ で解くことができます。

G. Game with Balls and Boxes

G - Two Step Sort 解説 by へnxteru

説明を簡単にするため問題を言い換えます。（言い換えなくても解けると思います。）また ここでは $(1, 2, \ldots, n)$ を並び替えた数列を順列と表すことにします。

順列 $P = (P_1, P_2, \ldots, P_n)$ と順列 $Q = (1, 2, \ldots, n)$ が与えられる。コスト A_i をかけて P の i 番目の値を自由に書き換えることができる。コスト B_i をかけて Q の i 番目の値を自由に書き換えることができる。書き換えた後 P と Q が同じ順列になっていなければならない。

この問題で書き換えないで残すもののコストを最大化したいと言い換えます。まず P と Q で同じ場所なのに異なる数である場合その 2 つは同時には残せません。またある数 i が P と Q で異なる場所にある場合もその 2 つは同時には残せません。この 2 つの条件を満たせばあとはすべて残すことができます。同時に残せないペアを辺で結ぶと円環になっています。適当なところで切って順番に並べると、数列上で隣り合うものを同時に選べないという問題になるので dp で解くことができます。計算量は $O(N)$ です。

H. High Powers / Sol.1

H - Symmetric 解説 by ≫shakayami

以下、$0^0 = 1$とします。

答えを$f(n, m)$とします。

このとき、

$$f(n, m) = s \cdot f(n-1, m) - t \cdot f(n-2, m) + u \cdot f(n-3, m)$$

$$f(n, m) = s \cdot f(n, m-1) - t \cdot f(n, m-2) + u \cdot f(n, m-3)$$

また、

$$\begin{pmatrix} f(0,0) & f(0,1) & f(0,2) \\ f(1,0) & f(1,1) & f(1,2) \\ f(2,0) & f(2,1) & f(2,2) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}$$

であるため、

$$\begin{pmatrix} f(n,m) & f(n,m+1) & f(n,m+2) \\ f(n+1,m) & f(n+1,m+1) & f(n+1,m+2) \\ f(n+2,m) & f(n+2,m+1) & f(n+2,m+2) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ u & -t & s \end{pmatrix}^n \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & u \\ 1 & 0 & -t \\ 0 & 1 & s \end{pmatrix}^m$$

となります。 よって、行列累乗によって$O(\log n + \log m)$で計算できます。

H. High Powers / Sol.2

H - Symmetric 解説 by ≫shakayami

以下、$0^0 = 1$とします。 答えを$f(n, m)$とします。

このとき、x, yについて以下の形式的べき級数を考えます。

$$\sum_{n=0}^{\infty} \sum_{m=0}^{\infty} f(n, m) x^n y^m$$

ここで、

$$\sum_{n=0}^{\infty} \sum_{m=0}^{\infty} a^n b^m x^n y^m = \frac{1}{1 - ax} \cdot \frac{1}{1 - by}$$

であることに注意すると、

$$\sum_{n=0}^{\infty} \sum_{m=0}^{\infty} f(n, m) x^n y^m = \frac{xy^2 - yx^2}{(1 - \alpha x)(1 - \beta x)(1 - \gamma x)(1 - \alpha y)(1 - \beta y)(1 - \gamma y)}$$

となります。 また、

$$(1 - \alpha x)(1 - \beta x)(1 - \gamma x) = 1 - (\alpha + \beta + \gamma)x + (\alpha\beta + \beta\gamma + \gamma\alpha)x^2 - \alpha\beta\gamma x^3 = 1 - sx + tx^2 - ux^3$$

であるため、

$[x^n] f(x)$を$f(x)$のx^nの係数とすると、

$$f(n, m) = \left([x^n] \frac{x}{1 - sx + tx^2 - ux^3}\right) \left([y^m] \frac{y^2}{1 - sy + ty^2 - uy^3}\right) - \left([x^n] \frac{x^2}{1 - sx + tx^2 - ux^3}\right) \left([y^m] \frac{y}{1 - sy + ty^2 - uy^3}\right)$$

となります。それぞれの級数の値についてはBostan-Mori algorithmなどを使うと$O(\log n + \log m)$で計算できます。

H. High Powers / Sol.3

H - Symmetric 解説 by ≫shakayami

はじめに

この解説に出てくる行列は、特に断りがない場合は3x3です。

また、Iは単位行列,Oは零行列です。

つまり、

$$I=\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix},O=\begin{pmatrix}0&0&0\\0&0&0\\0&0&0\end{pmatrix}$$

事前知識

対称式を行列で管理することを考えます。

$$A=\begin{pmatrix}0&1&0\\0&0&1\\u&-t&s\end{pmatrix}$$

という行列は、固有多項式が$\lambda^3-s\lambda^2+t\lambda-u=(\lambda-\alpha)(\lambda-\beta)(\lambda-\gamma)$であるため、固有値が$\alpha,\beta,\gamma$であることがわかります。ここで、$A$は固有多項式が重根を持たないため対角化可能となります。

$$\Lambda=\begin{pmatrix}\alpha&0&0\\0&\beta&0\\0&0&\gamma\end{pmatrix}$$

としたとき、ある正則行列Pが存在して、

$$A=P\Lambda P^{-1}$$

と書けます。

ここで、有理数係数多項式$f(x)=a_0+\cdots+a_nx^n$があったとき、

$$f(A)=\sum_{j=0}^{n}a_jA^j=\sum_{j=0}^{n}a_jP\Lambda^jP^{-1}=Pf(\Lambda)P^{-1}=P\begin{pmatrix}f(\alpha)&0&0\\0&f(\beta)&0\\0&0&f(\gamma)\end{pmatrix}P^{-1}$$

よって、$f(A)$の固有値は$f(\alpha),f(\beta),f(\gamma)$となります。

また、Aと$f(A)$は、$P^{-1}AP,P^{-1}f(A)P$がともに対角行列となることから、同時対角化可能となります。

また、有理数係数多項式$f(x)$があったとき、$f(\alpha)+f(\beta)+f(\gamma)$を求めたいならば$f(A)$のトレースを、$f(\alpha)f(\beta)f(\gamma)$を求めたいならば$f(A)$の行列式を求めればよいことがわかります。

また、 f,gという2つの多項式があったとき、 $f(A)+g(A)$の固有値は

$$f(\alpha)+g(\alpha),f(\beta)+g(\beta),f(\gamma)+g(\gamma)$$

$f(A)g(A)$の固有値は

$$f(\alpha)g(\alpha), f(\beta)g(\beta), f(\gamma)g(\gamma)$$

となります。（これもAと同時対角化可能）

さらに、$f(\alpha), f(\beta), f(\gamma) \neq 0$ の場合、$f(A)$の逆行列は固有値が$1/f(\alpha), 1/f(\beta), 1/f(\gamma)$となって、$A$と同時対角化可能となります。

解法

以上を踏まえて、この問題を解くことを考えます。

これは、

$$-\frac{\beta^m-\gamma^m}{(\beta-\gamma)}\cdot\frac{\alpha^n}{(\alpha-\beta)(\alpha-\gamma)}, -\frac{\gamma^m-\alpha^m}{(\gamma-\alpha)}\cdot\frac{\beta^n}{(\beta-\gamma)(\beta-\alpha)}, -\frac{\alpha^m-\beta^m}{(\alpha-\beta)}\cdot\frac{\gamma^n}{(\gamma-\alpha)(\gamma-\beta)}$$

を固有値に持つような行列を作って、それのトレースを求めればよいです。

そしてこの行列は以下の行列の積によって構成されます。

- 固有値が$\frac{\beta^m-\gamma^m}{(\beta-\gamma)}, \frac{\gamma^m-\alpha^m}{(\gamma-\alpha)}, \frac{\alpha^m-\beta^m}{(\alpha-\beta)}$となるような行列
- 固有値が$-\alpha^n, -\beta^n, -\gamma^n$となるような行列
- 固有値が$\frac{1}{(\alpha-\beta)(\alpha-\gamma)}, \frac{1}{(\beta-\gamma)(\beta-\alpha)}, \frac{1}{(\gamma-\alpha)(\gamma-\beta)}$となるような行列

これらについて順番に考えていきましょう。

1. $\frac{\beta^m-\gamma^m}{(\beta-\gamma)}$について

   条件を満たすような行列をS_mと書くことにします。

   結論から言うと、以下のように再帰的に計算することができます。

   - $S_0 = O$
   - $S_{2k} = ((\mathrm{tr}A^k)\cdot I - A^k)S_k \quad (k \geq 1)$
   - $S_{2k+1} = ((\mathrm{tr}A^{k+1})\cdot I - A^{k+1})S_k + (tI - sA + A^2)^k \quad (k \geq 0)$

   このとき、この部分の計算量は$O(\log m)$となって十分間に合います。

   ▼ 証明
   偶奇で場合分けします。

   $m = 2k$のとき、 これは

   $$\beta^{2k-1} + \beta^{2k-2}\gamma + \cdots + \beta\gamma^{2k-2} + \gamma^{2k-1}$$

   であり、これは因数分解ができて

   $$= (\beta^k + \gamma^k)(\beta^{k-1} + \beta^{k-2}\gamma + \cdots + \beta\gamma^{k-2} + \gamma^{k-1})$$

   ここで、$\beta^k + \gamma^k, \ldots$を固有値に持つような行列は、$(\mathrm{tr}A^k)\cdot I - A^k$と書けます。（$I$は単位行列） ま

た、$\beta^{k-1}+\cdots+\gamma^{k-1}$は$S_k$となります。

$m=2k+1$のとき、 これは

$$\beta^{2k}+\beta^{2k-1}\gamma+\cdots+\beta^k\gamma^k+\cdots+\beta\gamma^{k-1}+\gamma^k$$

となり、これは

$$(\beta^{k+1}+\gamma^{k+1})(\beta^{k-1}+\beta^{k-2}\gamma+\cdots+\beta\gamma^{k-2}+\gamma^{k-1})+\beta^k\gamma^k$$

と書けます。 ここで、$tI-sA+A^2$は$\beta\gamma, \gamma\alpha, \alpha\beta$を固有値に持つため、 、$(tI-sA+A^2)^k$は$\beta^k\gamma^k, \gamma^k\alpha^k, \alpha^k\beta^k$を固有値に持ちます。

2. $-\alpha^n$について

   これについては$-A^n$とすればよいです。
   計算量についても、二分累乗のテクニックを使えば$O(\log n)$で計算できて十分間に合います。

3. $\dfrac{1}{(\alpha-\beta)(\alpha-\gamma)}$について

   結論から言うと、

   $$(3A^2-2sA+tI)^{-1}$$

   となります。逆行列の存在性も保証されます。
   計算量についてもこの部分はn, mに対して定数時間で計算することができます。

   ▼ 証明
   これについては、固有値が

   $$(\alpha-\beta)(\alpha-\gamma), (\beta-\gamma)(\beta-\alpha), (\gamma-\alpha)(\gamma-\beta)$$

   であるような行列の逆行列を求めればよいです。 $\phi(x)=x^3-sx^2+tx-u$とします。 ここで、

   $$(\alpha-\beta)(\alpha-\gamma)=\lim_{x\to\alpha}\frac{(x-\alpha)(x-\beta)(x-\gamma)}{x-\alpha}=\lim_{x\to\alpha}\frac{\phi(x)-\phi(\alpha)}{x-\alpha}=\phi'(\alpha)$$

   ここで、$\phi'(x)=3x^2-2sx+t$であるため、

   $$3A^2-2sA+tI$$

   は$\phi'(\alpha), \phi'(\beta), \phi'(\gamma)$を固有値に持ちます。 また、$\alpha, \beta, \gamma$は相異なることから、この行列は正則であり、逆行列が存在します。

以上の1,2,3を踏まえると

$$-S_mA^n(3A^2-2sA+tI)^{-1}$$

のトレースが答えとなります。全体の時間計算量も$O(\log n+\log m)$となるため十分間に合います。

補足

行列の掛け算の順番についてですが、$S_m, A^n, (3A^2 - 2sA + tI)^{-1}$はどれも$A$についての有理式で書くことができて、これらは互いに可換となります。 よって行列の掛け算の順番については入れ替わっても構いません。

I. Items and Heroes

I - Good LACK 解説 by heno239

各頂点 i について、その頂点を根とする部分木を R_i とし、$S_i = \Sigma_{v \in R_i}(A_v - C_v)$ とおきます。

全ての i について $0 \leq S_i$ となることが、全ての目標を同時に達成するための必要十分条件となっています。

木をheavy light decompositionします。heavy light decompositionによってできたpathに適当に順番付けをし、i 個目のpath P_i に対して $X_i = \min_{v \in P_i}(S_v)$ と置きます。

いま、$0 \leq \min_i(X_i)$ となることが、全ての目標を同時に達成するための必要十分条件となっています。

ここで、値 C_x を変更したとします。このとき S_v に変更のある頂点 v は C_x の先祖のみとなっており、したがって X_i に変更のある添え字 i は $O(logN)$ 個 となっています。

各pathについて遅延セグ木を持っておくことで、それぞれのpathについて X_i の値を $O(logN)$ で更新することができるため、X_i の値を全てmultiset等で保管しておくことで、$\min_i(X_i)$ の値を高速に更新することができます。

時間計算量は $O(NlogN + Q(logN)^2)$ です。

J. Juggler's Trick

J - Delete Balls 解説 by へse1ka2

まず白いボールが存在しないとき、列のボールをすべて消すことができる必要十分条件について考えてみましょう。

列に含まれている赤いボールの数を R、青いボールの数を B とします。

このとき、$R = r \times k, B = b \times k$ を満たす整数 k が存在しなければ、明らかにすべてのボールを消すことはできません。

逆に、上を満たす k が存在すればすべてのボールを消すことができることを数学的帰納法で示しましょう。

$k = 0$ ならすべてのボールが消えています。

$k \geq 1$ とします。 列を左から $r + b$ 個ずつ k 個に分割します。以降これらを区間と呼びます。

ある区間に赤いボールがちょうど r 個含まれていればその部分を消せるので、帰納法の仮定よりすべてのボールを消すことができます。

そのような区間が存在しなければ、鳩ノ巣原理より、赤いボールが r 個より多く含まれている区間と r 個未満含まれている区間がそれぞれ一つ以上存在します。 よってある隣り合う区間が存在し、一方には赤いボールが r 個より多く、もう一方には赤いボールが r 個未満含まれています。 この区間の一方から開始し、ボール一つ分ずつ区間をスライドさせていくと、各時点で区間に含まれている赤いボールの数は高々一つずつ変わっていきます。よってある時点で赤いボールがちょうど r 個になるので、その部分を消すと帰納法の仮定よりすべてのボールを消すことができます。

これですべてのボールを消すことができる必要十分条件が分かりました。

さて、元の問題の解説に戻りましょう。

列の最初から i 番目までの部分列を s_i、s_i に対する最適値を d_i とします。

各 i に対し、s_i の最適値を達成する操作において、 i 番目のボールは消えているか残っています。

残っている場合、$d_i = d_{i-1}$ です。

消えている場合は、白いボールを適切に塗り分けることで、 $j + 1$ 番目から i 番目までの部分列に含まれる赤いボールの数が $r \times k$ 個、青いボールの数が $b \times k$ 個となるような j, k が存在し、$d_i = d_j + k$ となります。

以上より $d_i = max\{d_{i-1}, max\{d_j + k \mid j = i - (r + b) \times k$、$j$と$k$は上の条件を満たす$\}\}$ となります。

各 $i, k, j = i - (r + b) \times k$ に対し、上の条件が成り立つかどうかは適当な前処理をすると定数時間で判定できるので、全体の計算量は $O(N^2/(r + b))$ となります。

分割統治平面走査を行うことで、計算量を $O(N(logN)^2)$ に改善することができます。

K. King's Palace

K - Three Coloring 解説 by heno239

まず、壁を A 個と B 個に分けます。便宜上、前者をグループA、後者をグループBと呼びます。

簡単に言えば、グループAについて 3^A 通り全探索を行い、それに合うようなグループBの個数を $O(1)$ で求められるようにするのが目標です。以下に詳細を説明します。

まず、グループBの壁をそれぞれ 3 色のうちいずれかで塗ることを考えます。このとき、グループBの壁の塗り方を $3*B$ bitで次のように表現します。(便宜上、これをグループBの表現と呼びます。)

```
はじめ全てのbitは 0 になっている。グループBの i 番目の壁を、赤色としたときは 3*i 番目のbitを
1 に、緑色としたときは 3*i+1 番目のbitを 1 に、青色としたときは 3*i+2 番目のbitを 1 にする。
```

次に、グループAの壁をそれぞれ 3 色のうちいずれかで塗ることを考えます。このとき、グループAの壁の塗り方を $3*B$ bitで次のように表現します。(便宜上、これをグループAの表現と呼びます。)

```
はじめ全てのbitは 0 になっている。グループAの既に色で塗られている壁と見比べたとき、グループの
i 番目の壁を、赤色としても問題ないときは 3*i 番目のbitを 1 に、赤色としても問題ないときは 3*i
+1 番目のbitを 1 に、赤色としても問題ないときは 3*i+2 番目のbitを 1 にする。
```

このように表現したとき、数え上げるべきものは次のように言い表せます。

```
グループAの色の塗り方とグループBの色の塗り方の組であって、以下の 3 条件
- グループA内の壁同士で条件を破らない
- グループB内の壁同士で条件を破らない
- グループAの表現である 3*B bitが、グループBの表現である 3*B bitを包含する
を全て満たすもの。
```

そこで、あらかじめ グループB側を全探索してから高速ゼータ変換することで、各グループAの塗り方に対して $O(1)$ で組むことができるグループBの塗り方の総数を求めることができます。

時間計算量は $O(3^A + B2^{3B})$ となっており、$N = 22$ では $A = 15, B = 7$ とすると十分高速に動きます。

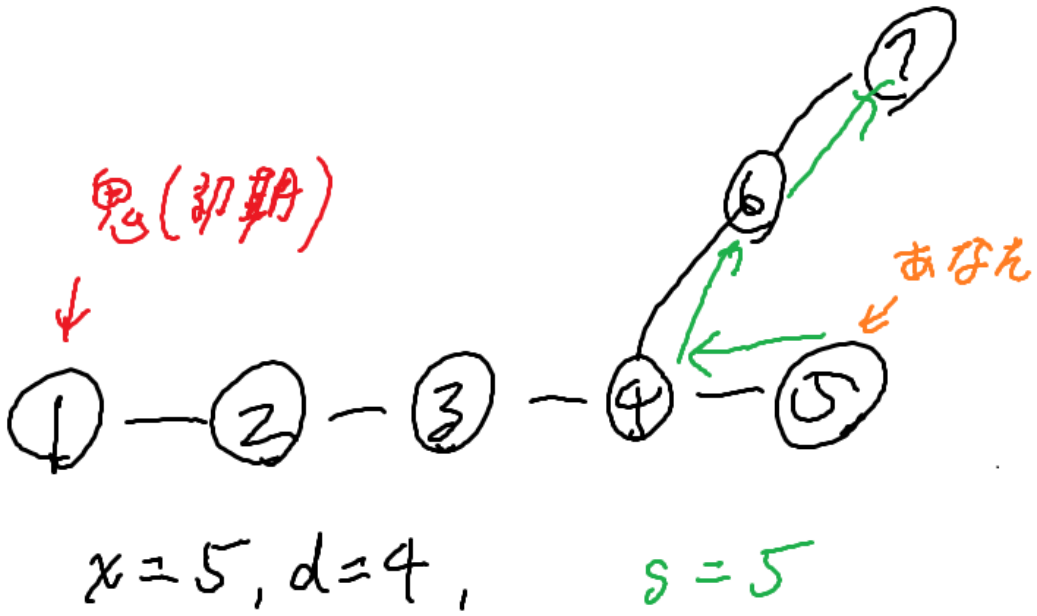
L. Lion and Zebra

L - Tag Game 解説 by ≫moririn2528

以降あなたを貴とする。貴の初期頂点を x, 初期の鬼までの距離を d_0 とする。 詳しい解法は略解の次に書かれている。

略解

貴は x から一番遠い点に向かって出来るだけ進む。やばくなったら一歩戻ってそこから出来るだけ遠いところに逃げる。やばい条件は、鬼との距離が 2 以下のとき。ただし、図 1 のようなとき、(d_0 が偶数で鬼と距離 2 になったときの貴の頂点から深さ d_0 の木が生えているとき)、矢印の方向に進む。 以上は直径の端点からの LCA と全方位木 DP を用いると実装できる。



詳細解法

$d_0 = 1$ のときは、答えは 1。$d_0 = 2$ のときは置いといて、$3 \leq d_0$ のときを考える。

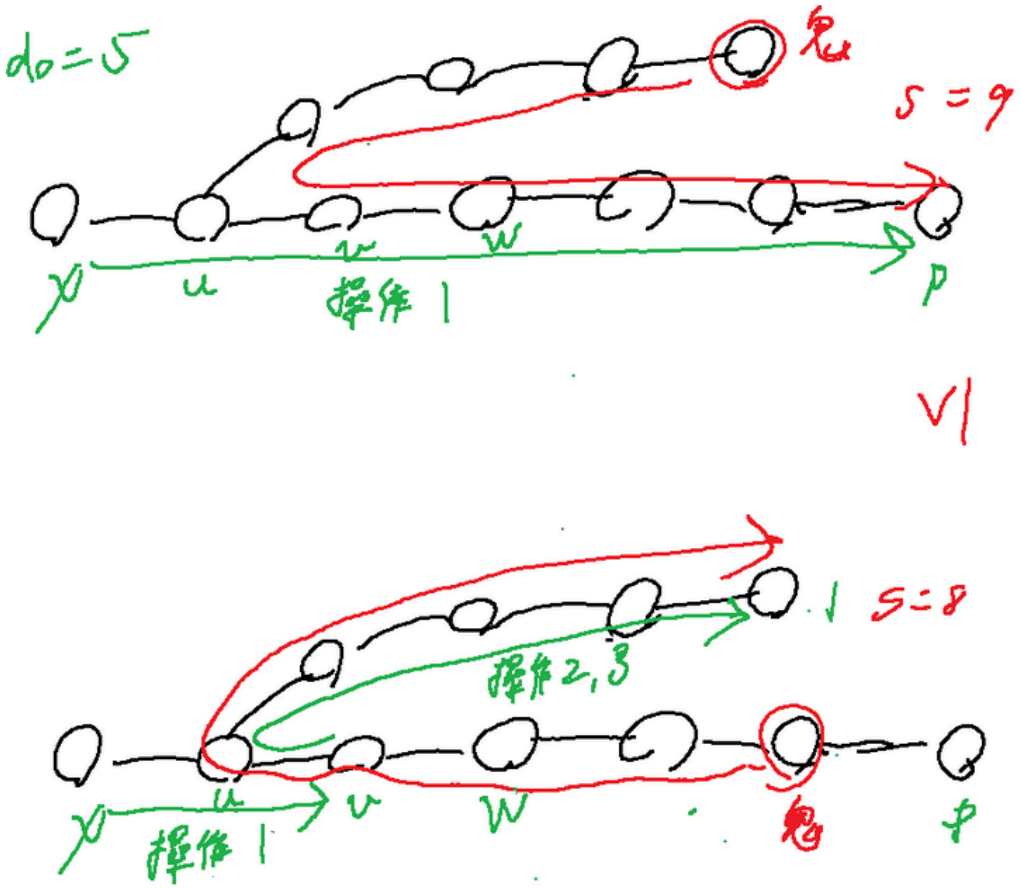
適当な木の直径の端点のうち x から遠い方の点を p とする。すると、p は x から一番遠い頂点の一つとなる。これは木の直径を求めるアルゴリズムからもわかる。貴は以下の操作を行うとき最適となる。 t 秒経った後の鬼までの距離を $d(t)$ とする。

1. $d(t) > 2$ のときは、p に向かって進み続ける。p に着けばそこで鬼と出会うまで停止、着く前に $d(t) \leq 2$ となったときは次の操作に進む。
2. $d(t) = 2$ のとき、貴が今いる頂点 a で木を分割する。それぞれの木の根は a とする。x を含む部分木の深さが $d/2 - 1$ であり、深さ $d/2$ の部分木があるとき、深さ $d/2$ の部分木の一番深い頂点に向かって、そこで静止。
3. 上記以外の時は x を含む部分木の一番深い頂点に向い、着いたら静止。

この一連の操作 S が最適であることを示す。 つまり、まず、操作 2,3 をするとき鬼と出会うまでの時間が最小(x, d_0 が同じで操作 1 のみで終わるときの出会う時間以下)となることを示し、次に、鬼の初期頂点がある頂点のとき、操作 S をすると、操作 2,3 をすることになり、ほかの操作は操作 S をしたときより早く、もしくは同じ時間で鬼と出会うことを示す。

頂点 x から頂点 p 方向に距離 $\lfloor \frac{d_0-1}{2} \rfloor - 1$ の頂点を u, 距離 $\lfloor \frac{d_0-1}{2} \rfloor$ の頂点を v, 距離 $d_0 - \lfloor \frac{d_0-1}{2} \rfloor$ の頂点を w とする。また、鬼の初期位置を z とする。頂点 a, b の距離を $dis(a, b)$ とする。

操作 2 で終わるときは、鬼と出会うまでの時間は $d_0 + 1$、操作 1 で終わるとき、貴は頂点 p まで移動した後 2 秒以上待つ。制約から $dis(x, p) \geq d_0$、操作 1 で終わるときより操作 2 で終わるときの方が短時間で鬼に捕まる。 操作 1 で終わるときより操作 3 の方が早く、もしくは同じ時間で捕まることについては、下図のように鬼の初期位置を変えることで、示される。



次に、ほかの操作についても考える。辺 (u, v) を切ったとき頂点 v を含む部分木上に鬼がいたとき、上記操作では、操作 2,3 で終わる。上記操作を行ったとき、$d(t) \leq 2$ となる最小整数の t を t_b とし、そのときに貴のいる頂点を $y(t_b)$ とすると、$y(t_b) = v$ である。 t_b 秒までの間で得られている情報は、頂点 v で分割し、頂点 u を含まない部分木集合 A のうち深さが $d_0 - \lfloor \frac{d_0-1}{2} \rfloor$ 以上の部分木上で頂点 v から距離 $d(t_b)$ にある頂点のいずれかに鬼がいる。ほかの操作をしても、t_b 秒以内では、必ず頂点 u を含む部分木上にいるので、この鬼の位置の候補を絞ることはできない。よって、どの操作をしても、この鬼の位置の候補がある方向には行けない(行くと操作 S を行うときよりも出会う時間が早くなる)。

d_0 が奇数の時、$d(t_b) = 1$ であり、A のうち深さ $\frac{d_0+1}{2}$ 以上の部分木には行けない。また、頂点 u を含む部分木の深さは $dis(x, v) = \frac{d_0-1}{2}$ 以上であるので、$y(t_b) = v$ のとき操作 3 が最適となる。また、$y(t_b) \neq v$ のときは行くことができる頂点が減少するので、操作 3 が最適となる。

d_0 が偶数の時、$d(t_b) = 2$ であり、A のうち深さ $\frac{d_0}{2} + 1$ 以上の部分木には行けない。また、頂点 u を含む部分木の深さは $dis(x, v) = \frac{d_0}{2} - 1$ 以上であるので、$y(t_b) = v$ のとき操作 2,3 が最適となる。また、$y(t_b) \neq v$ のときは行くことができる頂点が減少するので、操作 2,3 が最適となる。

よって示された。 $d_0 = 2$ のときは操作 2,3 と同様の操作をする。

実装については、事前に直径の端点を根とする LCA 2 つと、全方位木 DP でそれぞれの頂点に対しその頂点で分割したときの部分木の深さの集合を持って置き、LCA から u, v を計算し、部分木の深さの集合から深さ $d_0/2$ の部分木があるかどうかを調べればよい。 計算量は $O(q \log n)$

M. Math String / Sol.1

M - Formula 解説 by heno239

式を左から見ていったときに、現在の状態を $A+B*C$ または $A+B*$ という風に持つことを考えます。

$A+B*C$ の状態で、

次に数字 d が来た場合は、$A+B*(10*C+d)$ という風に更新できます。

次に文字 $+$ が来た場合は、$(A+B*C)+1*$ という風に更新できます。

次に文字 $*$ が来た場合は、$A+(B*C)*$ という風に更新できます。

$A+B*$ の状態で、

次に数字 d が来た場合は、 $A+B*d$ という風に更新できます。

そこで、 i 文字目まで見た時の $A+B*C$ の状態の数を N_i 、この状態全ての $A, B, B*C$ の総和をそれぞれ A_i, B_i, D_i とし、 $A+B*$ の状態の数を N_i^*、この状態全ての A, B の総和をそれぞれ A_i^*, B_i^* とすると、$N_{i+1}, A_{i+1}, B_{i+1}, D_{i+1}, N_{i+1}^* A_{i+1}^*, B_{i+1}^*$ が $N_i, A_i, B_i, D_i, N_i^*, A_i^*, B_i^*$ の線形和で表せるようになります。

したがって行列累乗を用いることで $N_N, A_N, B_N, D_N, N_N^*, A_N^*, B_N^*$ を求められ、この問題を解くことができます。

時間計算量は $O(logN)$ です。

M. Math String / Sol.2

M - Formula 解説 by ≫shakayami

答えを$f(n)$としたとき、以下の形式的べき級数を考えましょう。

$$\sum_{n=0}^{\infty} f(n)x^n$$

ただし、便宜上$f(0) = 0$とします。

以下のように級数を定めます。

- $A(x)$：x^Nの係数が「文字数がNであって、1-9が使えるときの式全ての答えの総和」であるような級数
- $B(x)$：x^Nの係数が「文字数がNであって、1-9と*が使えるときの式全ての答えの総和」であるような級数
- $C(x)$：x^Nの係数が「文字数がNであって、1-9と*と+が使えるときの式全ての答えの総和」であるような級数
- $D(x)$：x^Nの係数が「文字数がNであって、1-9が使えるときの式の総数」であるような級数
- $E(x)$：x^Nの係数が「文字数がNであって、1-9と*が使えるときの式の総数」であるような級数

ここで、

$$C(x) = \sum_{n=0}^{\infty} f(n)x^n$$

について、$[x^n]C(x)$が求められれば十分です。

ここで、$[x^n]P(x)$を,$P(x)$のx^nの係数です。

A, B, C, D, Eのうち簡単なものから求めましょう。

すると、

$$A(x) = \frac{45x}{(1-9x)(1-90x)}$$

$$D(x) = \frac{9x}{1-9x}$$

となります。

$B(x)$については、k個の数があって、i番目はa_i桁$(1 \leq a_i)$であるような状況を考えます。

ここで、kの範囲としては、$1 \leq k, 2k-1 \leq N$

であり、さらに文字数としてのつじつま合わせを考えると

$$a_1 + \cdots + a_k = N - k + 1$$

となります。

すると、

$$[x^N]B(x) = \sum_{k} \sum_{\sum a_i = N-k+1} \prod_{i=1}^{k} [x^{a_i}]A(x)$$

であり、これは

$$[x^N]B(x) = \sum_{k} [x^{N-k+1}]A(x)^k = [x^{N+1}] \sum_{k} \{xA(x)\}^k$$

これを踏まえると、

$$[x^N]B(x) = [x^{N+1}]\frac{xA(x)}{1 - xA(x)}$$

であるため。

$$B(x) = \frac{A(x)}{1 - xA(x)} = \frac{45x}{1 - 99x + 765x^2}$$

同様の手法で、$E(x)$も計算できて、

$$E(x) = \frac{D(x)}{1 - xD(x)} = \frac{9x}{1 - 9x - 9x^2}$$

となります。

$C(x)$については、式がk個の和に分解されて、各項がa_i文字で表現されているような状況を考えます。

ただし、$1 \leq k, 2k - 1 \leq N, a_i \geq 1, \sum_{i=1}^{k} a_i = N - k + 1$です。

すると、

$$[x^N]C(x) = \sum_{k} \sum_{\sum a_i = N-k+1} \sum_{i=1}^{k} [x^{a_i}]B(x) \prod_{i \neq j} [x^{a_j}]E(x)$$

よって、

$$[x^N]C(x) = \sum_{k \geq 1} [x^{N-k+1}] \sum_{i=1}^{k} B(x)E(x)^{k-1}$$

$$[x^N]C(x) = [x^N] \sum_{k \geq 1} kB(x)(xE(x))^{k-1}$$

$$[x^N]C(x) = [x^N]\frac{B(x)}{(1 - xE(x))^2}$$

よって、

$$C(x) = \frac{45x - 810x^2 + 2835x^3 + 7290x^4 + 3645x^5}{1 - 117x + 2592x^2 - 17901x^3 + 2673x^4 + 215784x^5 + 247860x^6}$$

となります。

級数の値については、Bostan-Mori Algorithmなどを使えば高速に計算することができます。