AtCoder Grand Contest 017 解説

writer: semiexp

2017 年 7 月 9 日

For International Readers: English editorial starts on page 6.

A: Biscuits

A_i がすべて偶数の場合は，どのように袋を選んでも食べるビスケットの枚数は偶数である．よって，$P = 0$ の場合は答えは 2^N，$P = 1$ の場合は答えは 0 である．

A_i の中に奇数が含まれているものとすると，A_k が奇数であるような k をとることができる．k 番目の袋以外の $N-1$ 個の袋の選び方は 2^{N-1} 通りである．これらの $N-1$ 個の袋から選んだ袋のビスケットの枚数の合計を S とするとき，

- S が奇数であれば，k 番目の袋を選ぶと合計は偶数，選ばないと合計は奇数.
- S が偶数であれば，k 番目の袋を選ぶと合計は奇数，選ばないと合計は偶数.

であるから，これらの 2^{N-1} 通りの選び方のそれぞれについて，k 番目の袋を選ぶか選ばないかでちょうど 1 通り，合計を偶数にする方法および奇数にする方法が存在する．よって，この場合の答えは P によらず 2^{N-1} である．

B: Moderate Differences

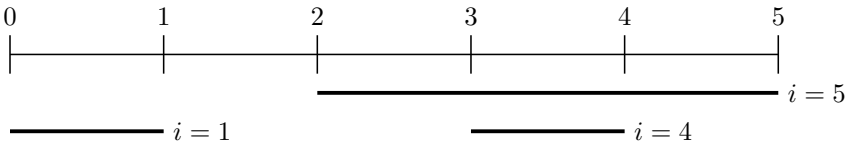
左から i 番目のマスに書く整数を X_i とする ($X_1 = A, X_N = B$)．$Y_i = X_{i+1} - X_i$ とおく ($i = 1, 2, \ldots, N-1$)．このとき，隣接するマスの整数の差の条件は,「各 $i = 1, 2, \ldots, N-1$ に対して，$-D \leq Y_i \leq -C$ または $C \leq Y_i \leq D$ が成り立つ」と言い換えられる．また，$\sum_{i=1}^{N-1} Y_i = \sum_{i=1}^{N-1} (X_{i+1} - X_i) = X_N - X_1 = B - A$ が成り立つ．逆に，これを満たすような整数列 $\{Y_i\}$ が存在すれば，$X_2 = X_1 + Y_1, X_3 = X_2 + Y_2, \ldots$ と順次定めていくことにより，条件をみたす $\{X_i\}$ を得ることができる．

$-D \leq Y_i \leq -C$ を満たす i が m 個存在するとする．このとき，他の $N-1-m$ 個の i については $C \leq Y_i \leq D$ が成り立つ．これらの条件式を各 i について足し合わせることで，$C(N-1-m) - Dm \leq \sum_{i=1}^{N-1} Y_i \leq -Cm + (N-1-m)D$ が成り立つ．よって，$C(N-1-m) - Dm \leq B - A \leq -Cm + (N-1-m)D$ が必要であるが，逆にこれが成り立つとき，条件をみたすような $\{Y_i\}$ をとることができる．実際，最初 Y_i の値を $-D$ または C にしておき，総和が $B - A$ 未満である限り値を $Y_1, Y_2, Y_3, \ldots, Y_{N-1}, Y_1, Y_2, \ldots$ の順で増やしていけばよい．

m を固定した際に条件をみたす書き方が可能かどうかは上のように $O(1)$ で判定できるので，$m = 0, 1, \ldots, N-1$ をすべて試し，1 つでも可能なものがあれば `YES`，いずれも不可能であれば `NO` を出力すればよい．

C: Snuke and Spells

整数 i が書かれたボールの個数を N_i とする．下図のように，各 $i = 1, 2, \ldots, N$ に対して，区間 $[i - N_i, i]$ を考える (下図は，ボールに書かれている整数が $1, 4, 5, 5, 5$ である場合に対応する)．すると，魔法を何回か唱えることでボールをすべて消滅させられることと，これらの区間が $[0, N]$ を覆い尽くすことは同値である．



整数 i が書かれたボールを j に変更するのは，区間 $[i-N_i,i]$ を縮めて $[i-N_i+1,i]$ にし，区間 $[j-N_j,j]$ を伸ばして $[j-N_j-1,j]$ にすることに相当する．特に，1 回の変更操作では，新たに覆える $[x-1,x]$ は高々 1 箇所である．よって，最初の状態で覆われていない $[x-1,x]$ が L 箇所あるとき，答えは L 以上である．

一方，L 回の操作で $[0,N]$ を覆い尽くすことは可能である．これを示すには，1 回の操作により覆われていない $[x-1,x]$ を 1 つ減らせることを示せばよい．$[j-1,j]$ が覆われていないような $j(1\leq j\leq N)$ をとる．$i-N_i<0$ であるような i があれば，i が書かれたボールを 1 個 j に変更すればよい．そのような i が存在しなければ，2 つ以上の区間で覆われているような $[r-1,r]$ が存在する．そのような r のうち最小のものを取ってくると，$i-N_i=r-1$ となるような (すなわち，$r-1$ を起点とするような) 区間が少なくとも 1 つ存在するから，この i が書かれたボールを 1 個 j に変更すればよい．

以上の議論より，$[x-1,x]$ が覆われていないような $x=1,2,\ldots,N$ の個数を数えれば答えが得られることがわかる．この問題では，複数の変更クエリを処理する必要があるが，そのためには「各 x に対し，$[x-1,x]$ を覆う区間の数」「$[x-1,x]$ が覆われていないような $x=1,2,\ldots,N$ の個数」を順次更新していけばよい (前者を更新する際に，後者の値も適切に更新する)．区間を 1 縮める，伸ばす操作でこの更新は $O(1)$ でできるから，全体として更新クエリは $O(M)$ で処理できる．初期化コスト $O(N)$ と合わせて，この問題は $O(N+M)$ で解くことができる．

D: Game on Tree

木に対して，その木から始めてゲームをしたときの Grundy 数を考える．

根 (頂点 1) から複数本の辺が出ている場合は，それぞれの辺ごとに別の根を持ってきて，(より辺の本数が少ない) 複数の木に対するゲームにすることができる．Grundy 数の性質より，元の木の Grundy 数は，このようにして得た木の Grundy 数すべてを XOR したものになる．

根からちょうど 1 本の辺が出ている場合は，Grundy 数に対して次の定理が成り立つ．

定理. 根付き木 T の *Grundy* 数を $G(T)$ で表す．T の根 r から新たに 1 本の辺を生やし，その辺の端点のうち r でないほうを新たに根とした木を T' とするとき，$G(T')=G(T)+1$ が成り立つ．

証明. 数学的帰納法により示す．

T が 1 頂点のみである場合は，$G(T)=0,G(T')=1$ が容易に確かめられるからよい．

T が $n-1$ 頂点以下の場合示されたと仮定して，T が n 頂点の場合を示す．T からある辺を取り除き，根を含まない連結成分も取り除いてできる木は，辺の選び方により $n-1$ 通り考えられるが，これらを $S_1,S_2,\ldots,S_{n-1}$ とおく．また，S_i の根から新たに 1 本の辺を生やし，その辺の r でない端点を根とした木を S_i' とおく．すると，帰納法の仮定より，$G(S_i')=G(S_i)+1$ が成り立つ．また，T' から遷移可能な木は，1 頂点のみからなる木 I (辺 rr' を取り除いた場合) または S_i' のいずれか (それ以外の辺を取り除いた場合) である．

Grundy 数の定義より，$G(S_1),\ldots,G(S_{n-1})$ の中には，$0,1,\ldots,G(T)-1$ はすべて含まれるが，$G(T)$ は含まれない．よって，$G(I),G(S_1')=G(S_1)+1,\ldots,G(S_{n-1}')=G(S_{n-1})+1$ の中には，$0,1,\ldots,G(T)$ はすべて含まれるが，$G(T)+1$ は含まれない．ゆえに $G(T')=G(T)+1$ である．よって示された． □

上の議論により，根から DFS を行いつつ各点を根とした部分木の Grundy 数を求めることで，木全体の Grundy 数を求めることができる．

E: Jigsaw

ピース i の**型**を，次のようにして定める．

- $C_i = 0$ ならば，$l = +_{A_i}$ とする．$C_i \neq 0$ ならば，$l = -_{C_i}$ とする．
- $D_i = 0$ ならば，$r = +_{B_i}$ とする．$D_i \neq 0$ ならば，$r = -_{B_i}$ とする．
- このとき，ピース i の型は (l, r) である．

$+_1, \ldots, +_H$ をまとめて $+$，$-_1, \ldots, -_{H-1}$ をまとめて $-$ で表すことにする．すると，型 (l, r) のピースのすぐ右に型 (l', r') のピースを置けるための必要十分条件は，次のいずれかを満たすことであるとわかる (*)：

- r, l' がともに $+$.
- ある k に対し，$r = +_k, l' = -_k$.
- ある k に対し，$r = -_k, l' = +_k$.

また，一番左のピースの型 (l, r) について，l は $+$ であり，一番右のピースの型 (l', r') について，r は $+$ である $(\star)$.

ここで，$2H$ 個の頂点 $u_1, \ldots, u_H, v_1, \ldots, v_H$ を持った有向グラフを考える．このグラフは，多重辺，自己辺なども許すものとする．このグラフの辺を，次のようにして与える：

- このグラフには辺 $1, 2, \ldots, N$ の N 個の辺がある．
- ピース i の型が (l, r) であるとき，辺 i の起点は
  - $l = +_k$ なら u_k.
  - $l = -_k$ なら v_k.
- ピース i の型が (l, r) であるとき，辺 i の終点は
  - $r = +_k$ なら v_k.
  - $r = -_k$ なら u_k.

グラフの辺をたどることを，そのピースを今までで最後に置いたピースの右の位置に置くことに対応させる．すると，(*) で示した 3 つの場合の並べ方は，グラフ上で次のたどり方に対応する：

- r, l' がともに $+$ の場合：最後に通った辺の終点が v_* であるとき，ある u_* に移動して，その点から出ている辺をたどる．
- ある k に対し，$r = +_k, l' = -_k$ の場合および，ある k に対し，$r = -_k, l' = +_k$ の場合：最後に通った辺の終点から出ている辺をたどる．

また，$(\star)$ に対応して，最初に通る辺の始点は u_*，最後に通る辺の終点は v_* である．

すなわち，グラフ上で「v_* から u_* に辺を 0 本以上増やすことで，ある u_* からある v_* までのオイラー路が作れるか」を判定すればよいことがわかる．

グラフに新たな頂点 w を追加する．v_* から u_* に辺を生やすときは，v_* から w の辺と w から u_* の辺を 1 本ずつ増やすことにする．さらに，最初に通る辺の始点を u_s，最後に通る辺の終点を v_t として，v_t から w，w から u_s の辺も 1 本ずつ増やすことにする．すると，元の条件が成り立てば，この新しいグラフ上で w を含むオイラー閉路が存在する．一方，このようなオイラー閉路が存在すれば，閉路中の $v_* \to w$, $w \to u_*$ の形の辺を 1 本ずつ取り除き，残る $v_* \to w \to u_*$ の辺をすべて $v_* \to u_*$ に変更することで，元の条件のオイラー路が作れることがわかる．

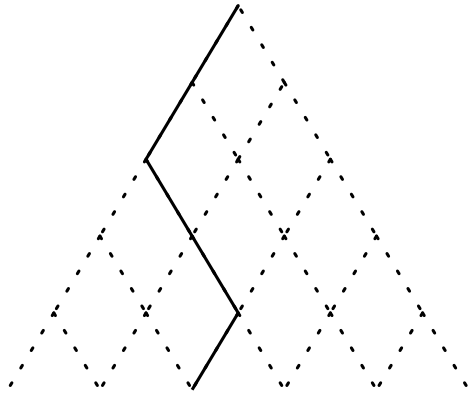
さて，有向グラフにおいてオイラー閉路が存在するためには，全体が (強) 連結でありかつ，すべての頂点において入次数と出次数が一致していることが必要十分である．v_* からは w への辺を増やすことしかできないので，v_* において出次数が入次数より大きい場合は不可能である．出次数が入次数以下の場合は，次数の差だけ w へ辺を増やす必要がある．u_* についても同様である．一方，この議論から，辺を増やした後のグラフは一意に定まる．また，このようにすると，w 以外のすべての頂点において入次数と出次数が一致するため，w においてもこれらは一致する．よって，次数の条件から不可能でない場合は，辺を増やした後，w を含む連結成分がすべての辺を含むかどうか判定すればよい．

F: Zigzag

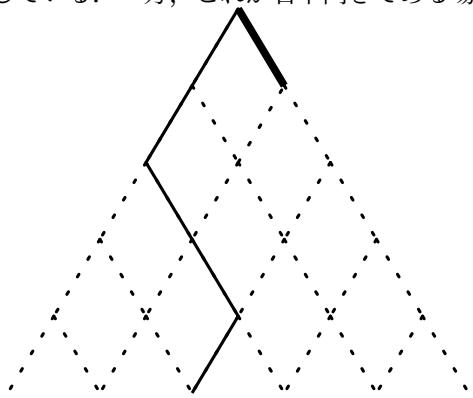
折れ線 1 から順に，現在の折れ線が直前の折れ線よりも左に来ることがないように，折れ線を 1 回目，2 回目，…，$N-1$ 回目の移動の順番で決定していくことを考える．

1 つの方法として，状態として「現在の折れ線の番号 i」「現在の折れ線で今までに行った移動の回数 j」「前回の折れ線と，現在の折れ線において，j 回目までの移動の後の横方向の位置の差 」「現在の折れ線で j 回目までの移動の向き」「直前の折れ線で，$j+1$ 回目以降の移動の向き」を持ってビット DP を行う方法がある．ここで,「現在の折れ線で j 回目までの移動の向き」「直前の折れ線で，$j+1$ 回目以降の移動の向き」は合わせて $N-1$ 箇所の向きであるから，この 2 つによる状態数は合わせて $O(2^n)$ である．よって，このビット DP の状態数は $O(n^3 2^n)$ であり，計算量が大きすぎる．

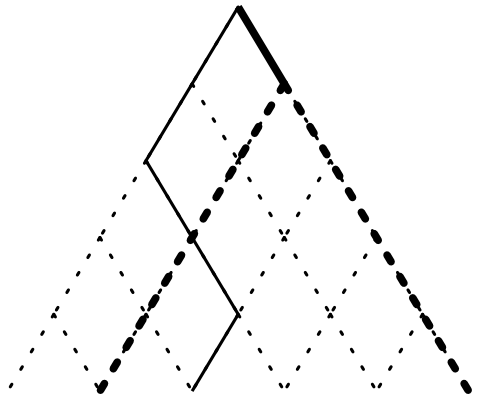
ここで，現在の折れ線が超えては行けない限界 (左の折れ線の位置) を，現在の折れ線の先端の観点から考える．下の図において，実線で示した折れ線が前回の折れ線であるとし，これより左に来ないように現在の折れ線を描くことを考える．



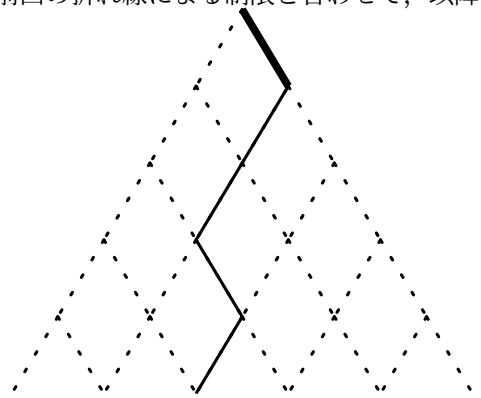
現在の折れ線の最初の移動が左下向きである場合は，ここまでの移動において，前回と今回の折れ線の先端位置は一致している．一方，これが右下向きである場合は，先端位置が 1 ずれる．



このとき，現在の先端から到達可能な部分を図示すると，下の太点線で囲まれた領域になる．



前回の折れ線による制限と合わせて，以降の移動において超えられない限界は，下の細実線で示した折れ線になる．



このようにすると，折れ線の先端が前回とずれていても，現在の先端を起点とする別な折れ線を限界とすることができる．この新たな限界は，次のようにして求められる．

- 元々の限界における最初の移動と同じ向きに移動している場合は，元々の限界と同じ．
- 元々の限界における最初の移動と異なる向きに移動している場合は,「元々の限界では左下，今回の移動では右下」という移動に限られる．元々の限界において右下の移動がない場合は，新たな限界は「最後まで左下の移動を繰り返す」折れ線である．一方，右下の移動がある場合は，その中で一番最初の移動を単に取り除いてできるものが新たな限界の折れ線となる．

これを用いると，ビット DP の状態として「現在の折れ線の番号 i」「現在の折れ線で今までに行った移動の回数 j」「現在，折れ線が超えてはいけない限界」を持っておけばよく，状態数は $O(n^2 2^n)$ となる．ビット DP における更新は，各状態から高々 2 つの別の状態に値を足し込めばよいが，ここで「新たな限界」を表すビット列を求める必要がある．これはビット演算を用いると効率よく行うことができる．現在の限界が左下向き，次の折れ線の向きが右下向きである場合のみ考える (他の場合は，限界を更新する必要がない)．ビット列の Least Significant Bit (LSB) から順に 1 回目，2 回目，...，$N-1$ 回目の移動に対応させ，0 を左下への移動，1 を右下への移動に対応させる．現在の限界が `m` であるとき,「j ビット目以降で最初の 1 を 0 にし，j ビット目を 1 にした」ビット列が新たな限界であるから，これは `((m - 1 << j) & m) | (1 << j)` で得られる．

よって，各状態からの更新も $O(1)$ で行えるから，全体で $O(n^2 2^n)$ で問題を解くことができる．

AtCoder Grand Contest 017 Editorial

writer: semiexp

July 9th, 2017

A: Biscuits

If all A_i are even numbers, the sum is always even. Thus, if $P = 0$ the answer is 2^N, and otherwise the answer is 0.

Otherwise, there exists an integer k such that A_k is odd. There are 2^{N-1} ways to choose a subset of $N-1$ bags (except for the k-th bag). Let S be the total number of biscuits in this subset.

- If S is odd, if you choose the k-th bag, the total number of biscuits will be even. Otherwise the total number of biscuits will be odd.
- If S is even, if you choose the k-th bag, the total number of biscuits will be odd. Otherwise the total number of biscuits will be even.

In both cases, for a given subset of $N-1$ bags, regardless of the value of P, there is exactly one way to make the total number P modulo 2. Therefore, if at least one bag contains odd number of biscuits, the answer is always 2^{N-1}.

B: Moderate Differences

Let X_i be the integer in the i-th square ($X_1 = A, X_N = B$). Let $Y_i = X_{i+1} - X_i$.

- For each i, $-D \leq Y_i \leq -C$ or $C \leq Y_i \leq D$ must be satisfied.
- $\sum_{i=1}^{N-1} Y_i = B - A$ must be satisfied because $\sum_{i=1}^{N-1} Y_i = \sum_{i=1}^{N-1} (X_{i+1} - X_i) = X_N - X_1 = B - A$.

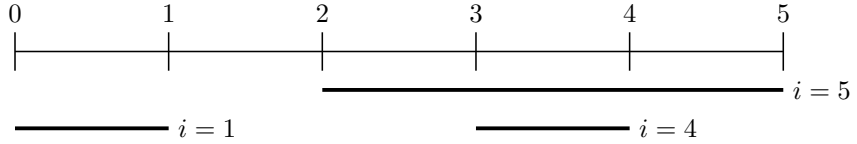
If you can find Y_i that satisfies the conditions above, you can construct a desired sequence by $X_2 = X_1 + Y_1, X_3 = X_2 + Y_2, \ldots$. Thus, these conditions are sufficient.

Suppose that m of Y_i satisfy $-D \leq Y_i \leq -C$ and the remaining $N-1-m$ Y_i satisfy $C \leq Y_i \leq D$. By adding these inequalities, we get $C(N-1-m) - Dm \leq \sum_{i=1}^{N-1} Y_i \leq -Cm + (N-1-m)D$. Thus, $C(N-1-m) - Dm \leq B - A \leq -Cm + (N-1-m)D$ must be satisfied. On the other hand, when this inequality holds, we can find an example of Y_i. We can first set the values of Y_i to $-D$ or C, and while the sum is smaller than $B - A$, we can increase integers one by one in the order $Y_1, Y_2, Y_3, \ldots, Y_{N-1}, Y_1, Y_2, \ldots$.

We can try all values of $m = 0, 1, \ldots, N-1$, and for each m we can check the inequality in $O(1)$ as we described above. If we find at least one m that satisfies the inequality, the answer is `YES`, otherwise the answer is `NO`.

C: Snuke and Spells

Suppose that we have N_i balls with an integer i. For each i, consider the interval $[i - N_i, i]$. For example, the following diagram shows the case with $1, 4, 5, 5, 5$.



We can vanish all balls by the spells if and only if these intervals entirely fill $[0, N]$. We claim that, when the total length of uncovered parts is L, we need L modifications (the proof is attached later).

Therefore, we can solve the problem by counting the number of x such that $[x - 1, x]$ is not covered. In order to handle multiple queries, we should keep the following values:

- For each x, the number of intervals that covers $[x - 1, x]$.
- The total length of uncovered parts.

When we update the former value, we should also update the latter value properly. Each query can be handled in $O(1)$, so this solution works in $O(N + M)$.

Proof.

- If we change an integer i to integer j, the interval $[i - N_i, i]$ will be shrunk to $[i - N_i + 1, i]$ and the interval $[j - N_j, j]$ will be enlarged to $[j - N_j - 1, j]$. In particular, in one modification, the total length of uncovered parts can decrease by at most one. Thus, we need at least L modifications.
- Suppose that some part of $[0, N]$ is not covered. In this case,
  - If there exists i such that $i - N_i < 0$, one of balls with an integer i is unnecessary.
  - Otherwise, choose the smallest integer r such that $[r, r + 1]$ is covered multiple times. There exists at least one i such that $i - B_i = r$, and one of balls with an integer i is unnecessary.

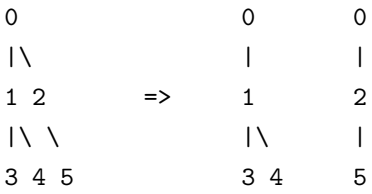
  If $[j-1, j]$ is not covered by any intervals, we can change an integer i to j and we can decrease the total uncovered length by one. By repeating this, we can cover $[0, N]$ in L modifications.

□

D: Game on Tree

We want to compute the Grundy Number of a given tree. If the Grundy Number is zero, Bob wins. Otherwise Alice wins.

Suppose that the root has $k > 1$ children. In this case, we can decompose the original tree into k smaller trees, and the Grundy Number of the original tree is XOR of the Grundy Numbers of new k trees.



If the root has exactly one child, we can compute the Grundy Number using the following theorem:

Theorem. *Let $G(T)$ be the Grundy Number of a rooted tree T. Let T' be a rooted tree obtained by attaching an edge to the root of T (and the new root is the new node attached to the old root). Then, $G(T') = G(T) + 1$.*

Proof. We can prove it by the induction on the number of vertices in T.

Let's compute the Grundy Number of T'.

- If we remove the new edge, we get a single node, and its Grundy Number is zero.
- Suppose that we remove one of other edges, and we get a new tree S. This tree is of the form "(one of trees reachable from T) + (one edge)". By the induction hypothesis, in the set of Grundy Numbers of these trees, $0 + 1, 1 + 1, \ldots, G(T) - 1 + 1$ will appear, but $G(T) + 1$ won't.

Therefore, $G(T') = G(T) + 1$. □

By combining the observations above, we can get the Grundy Number of the original tree by a simple DFS.

E: Blocks

Define the **type** of piece i as follows:

- If $C_i = 0$, $l = +_{A_i}$. If $C_i \neq 0$, $l = -_{C_i}$.
- If $D_i = 0$, $r = -_{B_i}$. If $D_i \neq 0$, $r = +_{B_i}$.
- The **type** of piece i is defined as (l, r).

We can put a piece of type (l', r') immediately to the right of a piece of type (l, r) if one of the following holds:

- l is positive and r' is negative.
- $r = l'$.

Also, when you arrange all intervals, the left part of the leftmost piece must be positive, and the right part of the rightmost part must be negative.

Now, consider a directed graph with $2H$ vertices. The vertices are labeled with $1, 2, \ldots, H$, and $-1, -2, \ldots, -H$. When we have a piece of type (l, r), we add an edge from l to r.

From the observation above, we want to determine if we can decompose this graph into one more more (directed) paths, such that each path starts from a positive vertex and ends at a negative vertex.

It turns out that this is equivalent to the following conditions:

- For each positive vertex, $outdegree \geq indegree$ holds.
- For each negative vertex, $outdegree \leq indegree$ holds.
- In each (weakly) connected component, there must be at least one vertex whose outdegree and indegree are different.

It is clear that these conditions are necessary. On the other hand, when these conditions hold, we can decompose the graph into a set of

- A path from a positive vertex to a negative vertex.
- A cycle.

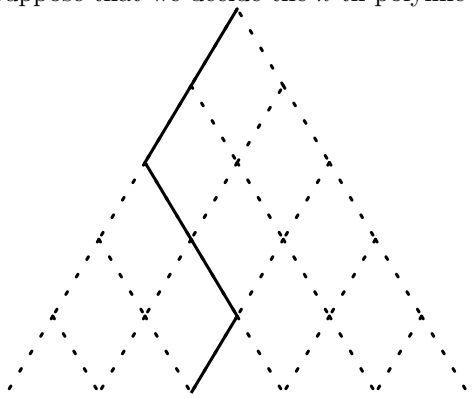
(This decomposition can be obtained by greedy connecting edges in any order).

Now, by the third condition, whenever we have a cycle, there exists at least one path or cycle that is adjacent to this cycle. Thus, we can merge them and we can decrease the number of cycles. By repeating this, we get a desired set of paths.

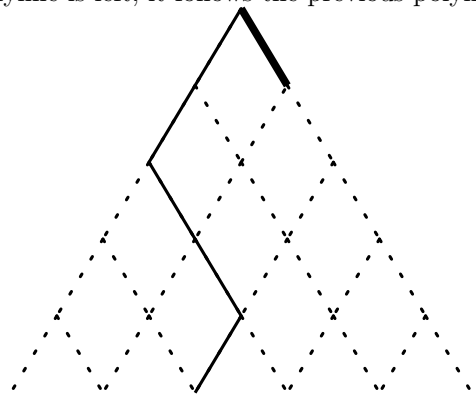
F: Zigzag

We can represent the shape of each polyline as a bitmask. We decide the polylines from polyline 1 in order. Define $dp[k][mask]$ as the number of ways to choose the first k polylines, such that the k-th polyline is $mask$. This DP will lead to $O(4^N * poly(N))$, or it can be improved to $O(2^N * N^3)$, but still it's too slow.

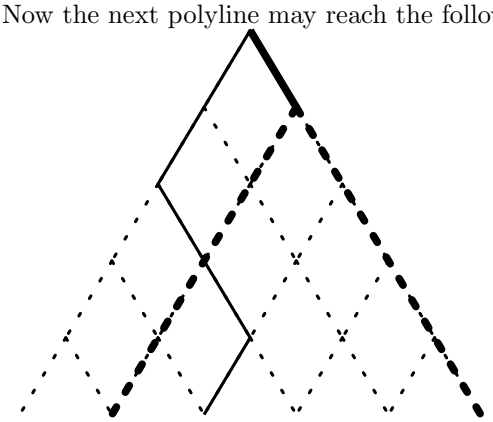
Suppose that we decide the k-th polyline as follows:



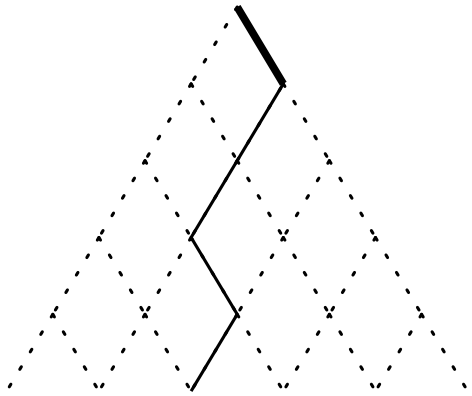
Now, we decide the $k+1$-th polyline from top to bottom. If the direction of the first (topmost) part of the next polyline is left, it follows the previous polyline. If the direction is right, it will look as follows:



Now the next polyline may reach the following part:



However, it is not allowed to go to the left of previous polyline. Thus, the region is limited as follows:



Now we can think as if the previous polyline was indeed this polyline. This way, even this case can be handled in the same way as the former case: now the $k+1$-th polyline follows the k-th polyline.

Let $P = (p_1, \ldots, p_{N-1})$ be the previous polyline, and let $Q = (q_1, \ldots, q_k)$ be the current polyline (it may not have reached the bottom yet). In general, we modify P as follows:

- If Q currently follows P, we don't modify P.
- Otherwise, $q_k = 1$ and $p_k = 0$. If there is no $i > k$ such that $p_i = 1$, we simply change p_k to 1 and now Q follows P. Otherwise, choose the smallest such i, change p_k to 1, and change p_i to 0.

All of these operation can be handled in constant time using bitwise operations.

Now, the state of the DP is $dp[i][j][mask]$. It means the number of ways to choose the first i polyline and the top j parts of the $i+1$-th polyline, with the following constraints:

- In the top j parts, the $i+1$-th polyline follows $mask$.
- In the future, the $i+1$-th polyline is not allowed to go to the left of $mask$.

Now we reduced the number of states to $O(n^2 2^n)$, and it works in $O(n^2 2^n)$ time.