AtCoder Grand Contest 002 解説

A : Range Product

次のような場合分けをします.

- $0 < A \leq B$ のとき : `Positive`
- $A \leq B < 0$ のとき
  - $B - A + 1$ が偶数のとき : `Positive`
  - $B - A + 1$ が奇数のとき : `Negative`
- $A \leq 0 \leq B$ のとき : `Zero`

B : Box and Ball

次のように問題を言い換えると考えやすいかもしれません.

> N 個のコップがある．最初，1 番目のコップには赤い水が 1 L 入っている．また，2〜N 番目のコップには無色の水が 1 L ずつ入っている．M 回の操作を順に行う．i 回目の操作では，x_i 番目のコップから水を 1 L すくい，それを y_i 番目のコップへ移す．すべての操作を終えた後，赤い水が入っているコップは何個か求めよ.

言い換えた後の問題は次のようにシミュレートできます．まず，必要な配列を

- $num[i] := i$ 番目のコップに入っている水の量
- $red[i] := i$ 番目のコップに赤い水が入っているか

と定義します．その後，次の擬似コードのような処理を行います.

```
for i = 1 to N do
  num[i] ← 1
  red[i] ← false
end for
red[1] ← true

for i = 1 to M do
  if red[x_i] then
    red[y_i] ← true
  end if
  num[x_i] ← num[x_i] − 1
  num[y_i] ← num[y_i] + 1
  if num[x_i] = 0 then
    red[x_i] ← false
  end if
end for
```

シミュレーションが終わった後，配列 red のうち true の個数が答えです．

C : Knot Puzzle

最後に結び目 i をほどくとすると，$a_i + a_{i+1} \geq L$ でなければなりません．条件を満たす i が存在しない場合，明らかに答えは `Impossible` です．逆に，条件を満たす i がひとつでも存在する場合，答えは `Possible` であることが示せます．結び目をほどく順番は次のように構成できます．

まず，$a_i + a_{i+1} \geq L$ を満たす i をひとつ選びます．次に，結び目 $1, 2, \ldots, i-1$ をこの順にほどいた後，結び目 $N-1, N-2, \ldots, i+1$ をこの順にほどきます．最後に，結び目 i をほどきます．こうすると，まだ結び目が残っているひと繋がりのロープには，必ず長さ $a_i + a_{i+1}$ の区間が含まれます．よって，各操作で選ぶひと繋がりのロープは，必ず長さの総和が L 以上となります．

D : Stamp Rally

まずは，「j 組目の兄弟のスコアは i 以下か？」という判定問題を考えてみましょう．この問題は次のように解くことができます．

兄弟が辺 $1, 2, \ldots, i$ のみを通って訪れられる頂点を数え上げ，それが z_i 以上か判定すればよいです．そのために，辺 $1, 2, \ldots, i$ のみからなるグラフを構築します．このグラフの連結成分のうち，頂点 x_i または頂点 y_i を含む連結成分が，兄弟が訪れられる頂点ということになります．例えば Union-Find を用いることで，辺 $1, 2, \ldots, i$ を追加するのを $\mathrm{O}(N+M)$ 時間で，兄弟が訪れ

られる頂点を数え上げるのを $\mathrm{O}(1)$ 時間で行えます．

以上の判定問題において i を二分探索することで，j 組目の兄弟のスコアを $\mathrm{O}((N+M)\log M)$ 時間で求められます．これを Q 組の兄弟について別々に行うことで，すべての兄弟のスコアを $\mathrm{O}(Q(N+M)\log M)$ 時間で求められます．もちろん，この方法では計算時間が掛かりすぎるので，高速化の必要があります．

Q 組の兄弟について別々に二分探索を行う場合，「グラフに辺 1, 2, . . . , M を追加していく」というまったく同じ処理を $\mathrm{O}(Q\log M)$ 回も繰り返すことになり，非効率的です．そこで，Q 組の兄弟について並列に二分探索を行うことを考えます．つまり，「グラフに辺 1, 2, . . . , M を追加していく」というパスを $\mathrm{O}(\log M)$ 回だけ繰り返し，各パスの適切なタイミングで 1, 2, . . . , Q 組目の兄弟が訪れられる頂点を数え上げます．この「適切なタイミング」は過去の二分探索の結果から決まります．この方法の計算時間は $\mathrm{O}((N+M+Q)\log M)$ となり，十分に高速です．

E : Candy Piles

まず，配列 a を a_i の降順にソートしておきます．以降は $a=(7,7,7,6,4,4,4,2,2)$ を例に説明します．

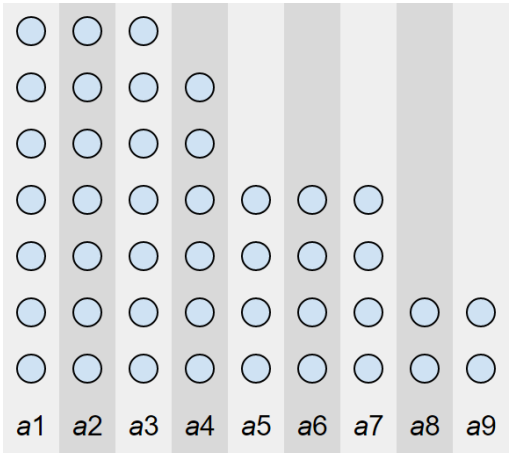

図 1　$a=(7,7,7,6,4,4,4,2,2)$ の例

「キャンディが最も多く残っている山をひとつ選び、その山のキャンディをすべて食べる」という操作は，最も左の列を消すことと等価です．また，「キャンディが残っているすべての山から、1個ずつキャンディを食べる」という操作は，最も下の行を消すことと等価です．これらの操作を交互に行い，すべてのキャンディを消したプレイヤーが負けです．

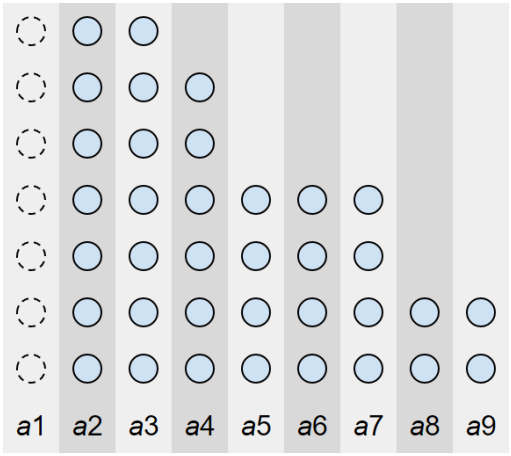

図 2　1 つ目の操作

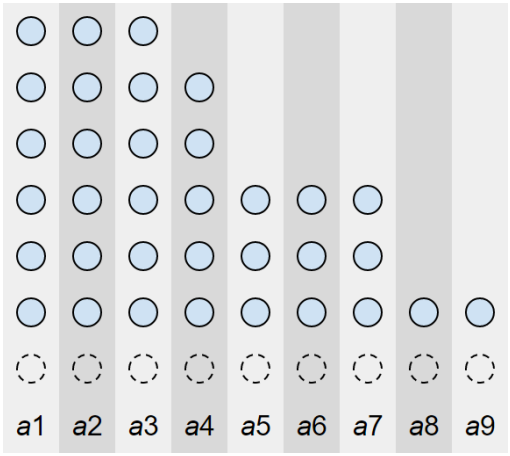

図 3　2 つ目の操作

このゲームは，さらに次のように言い換えられます．配列 a に対応するヤング図形を考えます．最初，原点（左下の角）に駒が置かれています．2 人のプレイヤーは交互に，駒をひとつ上かひとつ右へ動かします．駒をヤング図形の縁へ動かしたプレイヤーが負けです．

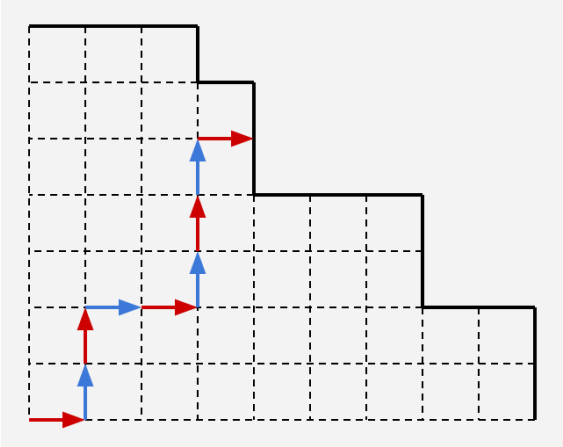

図 4　ヤング図形上のゲーム

ヤング図形上の各座標について，「その座標に駒が置かれた状態でターンが回ってきた場合，自分は勝つか負けるか」を ○ / × で書き込んでみます．まず，縁にはすべて ○ を書き込みます．それ以外の座標については，ひとつ上の座標とひとつ右の座標がともに ○ ならば × を書き込み，そうでなければ ○ を書き込みます．書き込みが終わった後，原点の ○ / × を見れば，答えが分かります．しかし，この方法の計算時間は $\mathrm{O}(\sum a_i)$ で，制限時間に間に合いません．

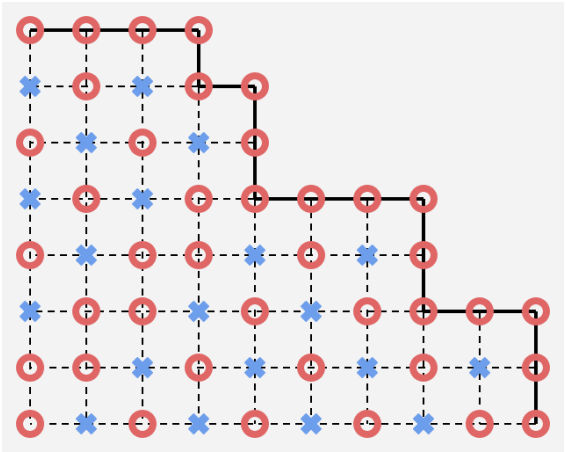

図 5　各座標の ○ / ×

上図を観察すると，縁以外の ○ / × は斜め方向に並んでいることが分かります．この性質を利用すると，原点の ○ / × を次のように求めることができます．まず，原点から右上向きへ，縁に達する直前まで矢印を伸ばします．次に，そこから右向きと上向きへ，縁に達する直前まで矢印を伸ばします．右向きと上向きの矢印の長さがともに偶数ならば，原点は × です．そうでなければ，原点は ○ です．この方法の計算時間は $\mathrm{O}(N)$ で，十分に高速です．

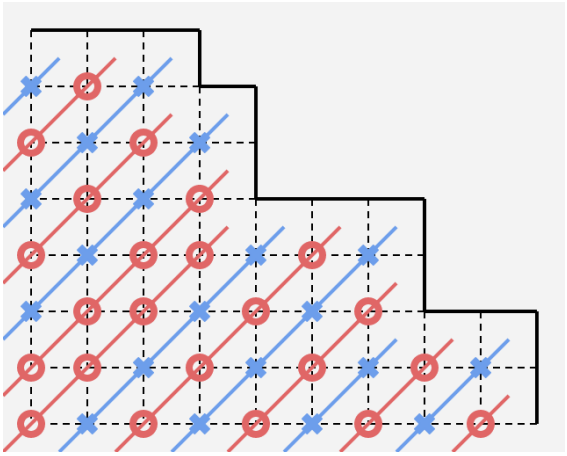

図 6　斜め方向に並ぶ ○ / ×

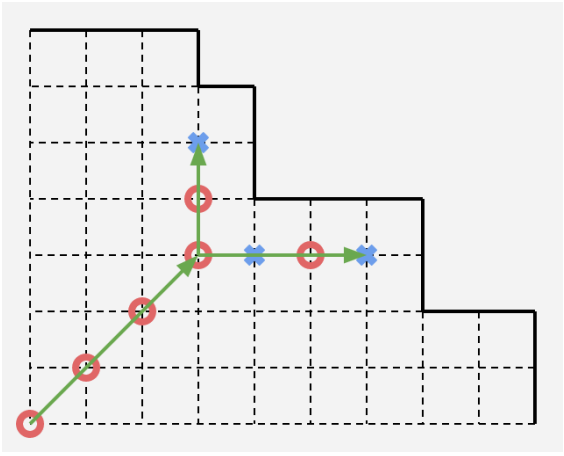

図 7　原点の ○ / × を高速に求める

F : Leftmost Ball

$K = 1$ の場合，答えは 1 通りです．以降は，$K \geq 2$ の場合のみを考えます．さらに，塗り替え後の色の列のうち，色 $1, 2, \ldots, N$ がこの順に現れるようなものだけを数え上げることにします．この個数に $N!$ を掛ければ答えが求まります．

まずは，色の列が与えられたとき，それが塗り替え後の色の列としてあり得るか判定する問題を考えてみましょう．もちろん，色 0 がちょうど N 個，色 1, 2, ..., N がちょうど $K-1$ 個ずつ含まれることが必要条件です．さらに，次のような必要条件も思いつきます．

- 各 $1 \leq i \leq N$ について，(左から i 番目の色 0) $<$ (最も左の色 i) という位置関係が成り立つ．

例えば，$(0, 1, 0, 2, 1, 2, \underline{3}, \underline{0}, 3)$ という色の列は，(最も左の色 3) $<$ (左から 3 番目の色 0) という位置関係なので，塗り替え後の色の列としてあり得ません．実は，これらの必要条件を満たす色の列は，常に塗り替え後の色の列としてあり得ます．というのも，色 0 を左から順に色 1, 2, ..., N へ塗り替えると，塗り替え前の色の列が構成できるからです．

以上より，問題は次のように言い換えられます．

> 色 0 をちょうど N 個，色 1, 2, ..., N をちょうど $K-1$ 個ずつ，左から右へ並べる．ただし，各 $1 \leq i \leq N-1$ について，最初の色 i を並べた直後から色 $i+1$ を並べ始められる．また，各 $1 \leq i \leq N$ について，i 番目の色 0 を並べた直後から色 i を並べ始められる．色の列は何通りか？

この問題の答えは，次のようなグラフをトポロジカルソートする方法の個数として表せます．

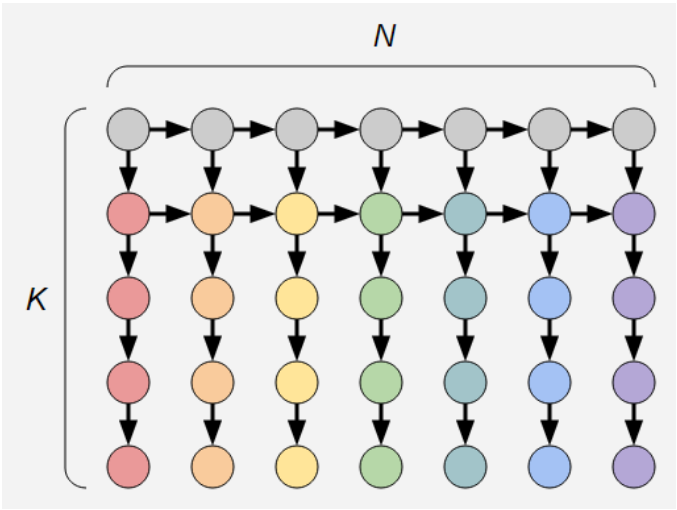

図 8　色の位置関係を表すグラフ

トポロジカルソートする方法の個数は，次のような DP で数え上げられます．下図のようなグラフをトポロジカルソートする方法を $dp[i][j]$ 通りと定義します．下図の場合，先頭に並べられる頂点は 2 通りあります．黒の頂点を先頭に並べる場合，残りの頂点をトポロジカルソートする方法は $dp[i-1][j]$ 通りです．一方，オレンジの頂点を先頭に並べる場合，残りの頂点をトポロジカルソートする方法の個数はどうなるでしょうか？ オレンジの頂点を先頭に並べる場合，残りのオレンジの頂点もあらかじめ位置を決めてしまうことにします．残りのオレンジの頂点は $K-2$ 個で，残りの全色の頂点は $i+j(K-1)-1$ 個なので，位置を決める方法は ${}_{i+j(K-1)-1}C_{K-2}$ 通りです．よって，残りの頂点をトポロジカルソートする方法は ${}_{i+j(K-1)-1}C_{K-2} \cdot dp[i][j-1]$ となります．このような漸化式にしたがって $dp[N][N]$ を計算すればよいです．計算量は $\mathrm{O}(N^2)$ です．

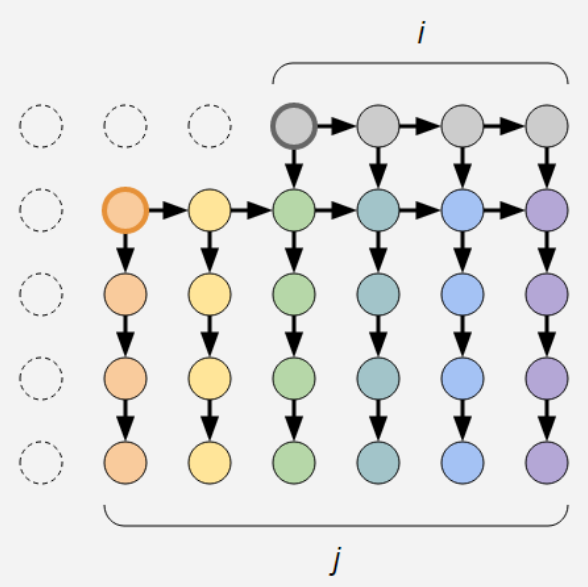

図 9　$dp[i][j]$ に対応するグラフ

AtCoder Grand Contest 002 Editorial

A : Range Product

Consider the following cases:

- $0 < A \leq B$: `Positive`
- $A \leq B < 0$
  - $B - A + 1$ is even : `Positive`
  - $B - A + 1$ is odd : `Negative`
- $A \leq 0 \leq B$: `Zero`

B : Box and Ball

You can restate the problem as follows:

> There are N cups. Initially, the cup 1 contains 1 liter of red water, and each of the other cups contains 1 liter of transparent water. You perform M operations. In the i-th operation, you take 1 liter of water from the cup x_i and pour it into the cup y_i. After you perform all the operations, compute the number of cups that contain red water.

After the restatement, you can solve the problem by the following simulation. First, define two arrays:

- $num[i]$:= The amount of water in the cup i
- $red[i]$:= Whether the cup i contains red water or not

Then, perform the following operations:

```
for i = 1 to N do
  num[i] ← 1
  red[i] ← false
end for
red[1] ← true

for i = 1 to M do
  if red[x_i] then
    red[y_i] ← true
  end if
  num[x_i] ← num[x_i] − 1
  num[y_i] ← num[y_i] + 1
  if num[x_i] = 0 then
    red[x_i] ← false
  end if
end for
```

The answer is the number of "true" in the array red after the simulation.

C : Knot Puzzle

If that the knot i is untied last, $a_i + a_{i+1} \geq L$ must be satisfied. When there is no such i, the answer is obviously `Impossible`. On the other hand, when there exists such i, we can prove that the answer is `Possible`. We can construct the ordering of knotting in the following way:

Choose an i that satisfies $a_i + a_{i+1} \geq L$. First, untie the knots $1, 2, \ldots, i-1$ in this order. Then, untie the knots $N-1$, $N-2$, $\ldots$, $i+1$ in this order. Finally, untie the knot i. This way, the ropes with knots always form a single connected component and this component contains an interval of length $a_i + a_{i+1}$. Thus, in each operation, the length of the chosen rope os at least L.

D : Stamp Rally

Consider the following decision problem: "Is the score of the j-th pair of brothers at most i?" This problem can be solved in the following way.

It is sufficient to count the number of vertices that are reachable by the brothers by using edges $1, 2, \ldots, i$, and check if this is at least z_i. In order to do this, construct a graph with only edges $1, 2, \ldots, i$. The brothers can visit a vertex if this vertex is in the same connected

component of this graph as vertices x_i or y_i. For example, if you use disjoint-set union structure, you can add the edges $1, 2, \ldots, i$ in almost $\mathrm{O}(M)$ time (strictly speaking, there is an extra factor of inverse-ackerman function but we can ignore it) and count the number of vertices that can be visited by the brothers in $\mathrm{O}(1)$ time.

In this decision problem, by using a binary search on i, we can compute the score of the j-th brothers in $\mathrm{O}(M \log M)$ time. If you do this for Q pairs of brothers independently, you can compute the scores for all pairs in $\mathrm{O}(QM \log M)$ time. Of course, we need to improve it.

If you do the binary search for Q pairs of brothers independently, you repeat exactly the same sequence of operations "add edges $1, 2, \ldots, M$ to the graph in this order" $\mathrm{O}(Q \log M)$ times and it looks inefficient. We want to perform the binary search for Q pairs of brothers in parallel. That is, you can repeat the sequence of operations $\mathrm{O}(\log M)$ times and count the number of vertices visited by each pair of brothers at appropriate time. This "appropriate time" can be determined using the results of former binary search.

This way the algorithm works in $\mathrm{O}((M + Q) \log M)$ time.

E : Candy Piles

First, sort the array a in decreasing order. In the remaining part of this editorial, we use an example $a = (7, 7, 7, 6, 4, 4, 4, 2, 2)$.

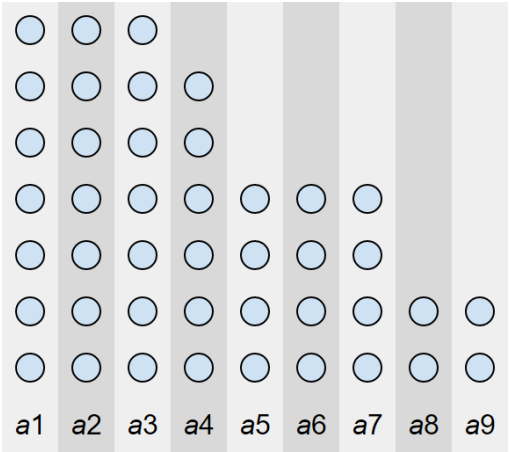

図 1　An example: $a = (7, 7, 7, 6, 4, 4, 4, 2, 2)$

The first type of operation (take all candies from the biggest pile) is equivalent to removing the leftmost column in the diagram. The second type of operation (take a candy from each non-empty pile) is equivalent to removing the bottommost row in the diagram. Two players perform these operations alternately, and the player who removes the last candy loses.

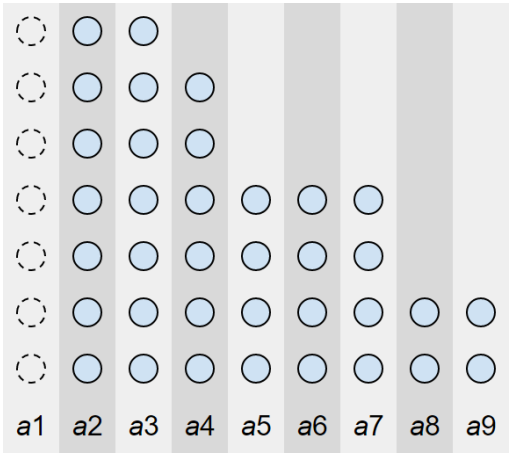

図 2　The first type of operation

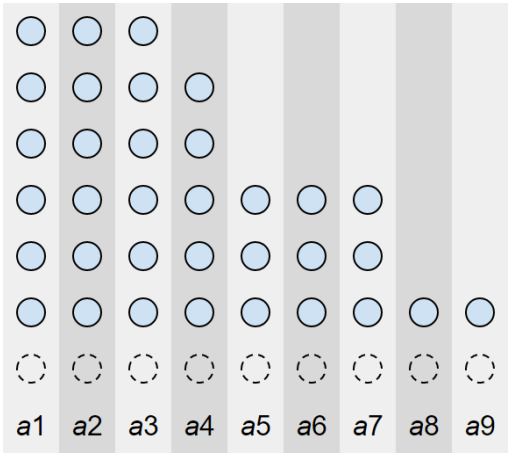

図 3　The second type of operation

We can further restate the statement as follows. Consider a young diagram that corresponds to the array a. Initially, you put a token on the left-bottom corner. Two players move the token by a unit distance to the right or to the up alternately, and the player who moves the token to the edge of the diagram loses.

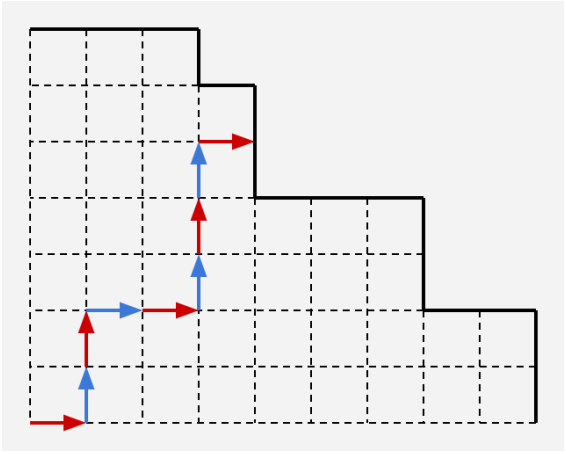

図 4　A game on young diagram

For each lattice point on the young diagram, write a 'o' if it's a winning state and write a 'x' if it's a losing state. First, write 'o' to each lattice point on the top-right boundary. For each other lattice point, write a 'x' if both of top-right adjacent lattice points are 'o', and otherwise write a 'o'. You can check the answer by seeing the bottom-left corner lattice. However, this solution works in $O(\sum a_i)$ and it's too slow.

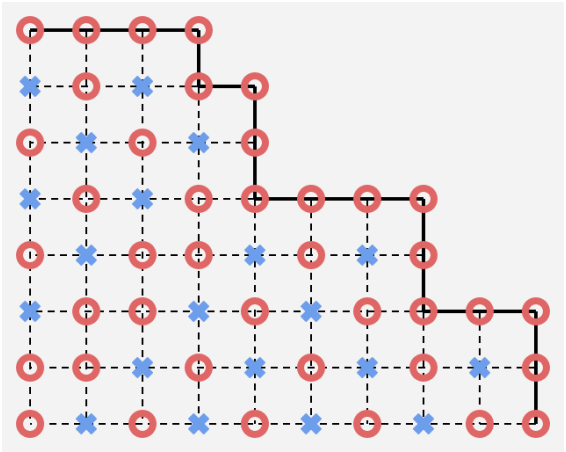

図 5　A diagram with ' ○' and ' × '

From the diagram above, you can see that in each diagonal the same symbol is written (except for the top-right boundary).

Proof: assume that the left-bottom corner is $(0, 0)$ and we introduce a coordinate system. If $(x + 1, y + 1)$ is inside the diagram and this is a losing state, both $(x, y + 1)$ and $(x + 1, y)$ will be winning states and thus (x, y) will be a losing state. Similarly, we can prove that when $(x+2, y+2)$ is inside the diagram and both $(x+2, y+2)$ and $(x+1, y+1)$ are winning states, (x, y) will be a winning state.

You can determine the symbol at the left-bottom corner as follows.

First, you start from the left-bottom corner and go to top-right diagonal one unit before the boundary. If the lengths of both arrows are even, the answer is 'x'. Otherwise the answer is 'o'. This way you can compute the answer in $O(N)$ time.

F : Leftmost Ball

When $K = 1$, the answer is obviously 1. Assume that $K \geq 2$. Also, assume that if we compare the leftmost occurrence of the colors $1, 2, ..., N$, the colors appear in this order. (We can get the answer by multiplying $N!$ to it.)

First, let's discuss how to determine if a given sequence of colors (0 to N) is valid or not. Of course, there must be exactly N occurrences of balls of color 0, and exactly $K - 1$ occurrences

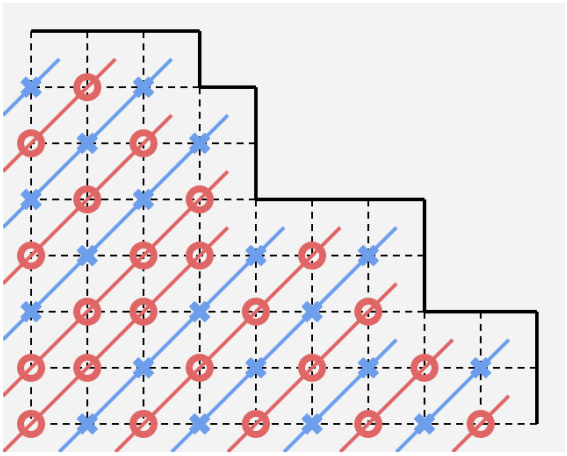

図 6　Diagonal ○ / ×

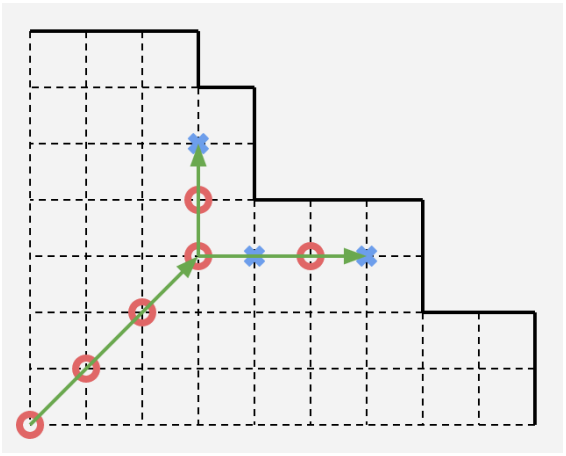

図 7　Compute the symbol at the origin quickly

each of colors $1, 2, ..., N$. Also, the following condition must be satisfied:

- For each $1 \leq i \leq N$, the i-th (from the left) ball of color 0 is to the left of the leftmost ball of color i.

For example, the sequence $(0, 1, 0, 2, 1, 2, \underline{3}, \underline{0}, 3)$ is invalid because the condition above is not satisfied for $i = 3$. On the other hand, we can prove that these conditions are sufficient. If we color the i-th ball of color 0 with color i, we can reconstruct the colors before the repainting to color 0.

Therefore, the problem can be restated as follows:

> You arrange N balls of color 0 and $K - 1$ balls each of colors $1, 2, ..., N$. For each $1 \leq i \leq N - 1$, you can put balls of color $i + 1$ only after you put at least one ball of color i. Also, for each $1 \leq i \leq N$, you can put balls of color i only after you put at least i balls of color 0. How many ways are there to arrange the balls?

The answer is the same as the number of topological orderings of the following graph: (The vertices on the top row corresponds to 1st, 2nd, ..., N-th ball of color 0 from let to right. The red vertices corresponds to 1s5, ..., $K - 1$-th ball of color 1 from top to bottom, and so on.)

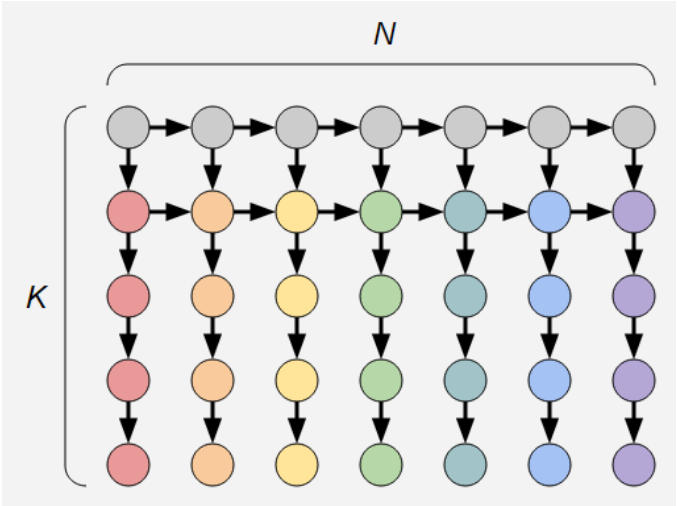

図 8　You count the number of topological orderings of this graph

The number of topolocial orderings can be computed using the following DP. Let $dp[i][j]$ be the number of topological orderings of the following graph.

There are two possible first vertex in a topolocial ordering: the bold black vertex or the bold orange vertex. When the bold black vertex comes first, the number of ways to order the remaining vertices is $dp[i-1][j]$. When the bold orange vertex comes first, the ordering of the remaining vertices can be decided by the topological ordering of non-orange vertices ($dp[i][j-1]$ ways) and the way you interleave orange vertices and non-orange vertices (${}_{i+j(K-1)-1}C_{K-2}$ ways because there are $K-2$ orange vertices and $i+j(K-1)-1$ vertices in total). Thus, the total number of topological orderings is ${}_{i+j(K-1)-1}C_{K-2} \cdot dp[i][j-1]$. (Here C denotes the binomial coefficient)

This way, we can compute the array $dp[][]$, and $dp[N][N]$ is the answer. The time complexity is $\mathrm{O}(N^2)$.

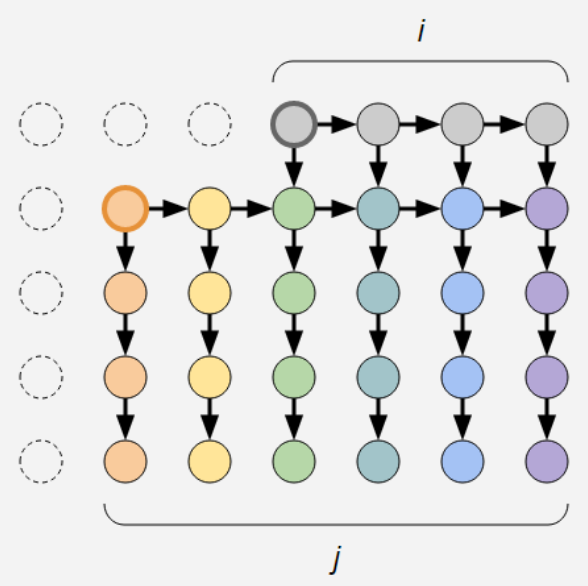

図 9　The graph corresponds to $dp[i][j]$