AGC 012 解説

writer : camypaper

2017 年 4 月 1 日

For international readers: English editorial starts from page 8.

A : AtCoder Group Contest

結論から言えば，「$2, 4, 6, \ldots, 2N$ 番目に強い参加者の強さの和」が答えである．参加者を弱い順に並べたとき、最初の N 人を N 個のチームの最も弱いメンバーとすることができる．このとき $N+1$ 番目の参加者はどのチームに入ってもチームで 2 番目に強いメンバーとなる．すると，$N+2$ 番目の人は $N+1$ 番目の参加者と同じチームにするのがよいことが分かる．このようにしてチームを作っていくと，上記の値が得られる．

計算量は，強さの順にソートする部分が最も重く， $O(N \log N)$ である．

B : Splatter Painting

操作を逆順に見ていくことにする．このとき，頂点の色が上書きされるという条件は一度でも色が塗られた頂点の色が変わることはない，と言い換えられる．すると，「頂点 v から距離 d 以内にある頂点たちの色を塗る」という操作が一度でも行われたならば，(操作を逆順に見たときの) それ以降の操作によって頂点 v から距離 d 以内にある頂点の色が変わることはないことがわかる．これを利用して動的計画法によりこの問題を解くことが可能である．

$\mathrm{dp}(v, d)$ を「頂点 v から距離 d 以内にある頂点たちの色を塗る」という操作が行われた (操作を逆順に見たときの) 最初の時点，とする．頂点 v から距離 d 以内にある頂点の色を塗るという操作が行われたとき，v と隣接している頂点 u について「頂点 u から $d-1$ 以内にある頂点を塗る」，という操作が行われる．また，「頂点 v から距離 $d-1$ 以内にある頂点の色を塗る」という操作も行われていると考えてよい．

最終的な頂点の色は $\mathrm{dp}(v, 0)$ から何番目の操作によって色が塗られたかを知ることで求められる．ただし，操作が行われていない場合頂点の色は 0 である．これらの処理は $O(Q + (N + M) \max d_i)$ の計算量で達成可能であり，十分高速である．

C : Tautonym Puzzle

p を $(1, 2, 3, \ldots, n)$ を並び替えた数列として，$s = (p_1, p_2, p_3, \ldots, p_n, 1, 2, 3, \ldots, n)$ とする．このとき，s の部分列であって，良い文字列であるようなものの個数は p の増加部分列の数 $f(p)$ で表される．以下のような問題を解くことが可能ならば，元の問題も解くことが可能である．

> 長さ 100 以下の順列 p であって，$f(p) = N$ を満たすような p の一例を示せ．

ここで，$p = (1, 2, 3, \ldots, n)$ のとき，$f(p) = 2^n - 1$ である．p の $i-1$ と i の間に $n+1$ を挿入した数列を p'_i としたとき，$f(p'_i) = 2^n - 1 + 2^{i-1}$ となり，$f(p)$ から 2^i だけ増加する．さらに，$p = (1, 2, 3, \ldots, n)$ に $n+1, n+2, \ldots$ を上記のように挿入していくとき，新たに挿入された数たちが降順に並ぶように挿入すると，「$f(p)$ を 2^i だけ増加させる」という独立な操作として扱うことが可能である．これを利用すると，$2^n - 1 + 2^a + 2^b + \ldots = N$ となるように順列を構成すればよいことが分かる．以下の C++ のソースコードで表されるように構成を行うと，長さ $2 \log N$ 以下での構成が可能である．$N \leq 10^{12}$ より，構成された順列 p の長さは 80 以下となり，必ず条件を満たす．

```
vector<int> construct(long long N){
    int n;
    vector<int> p;
    for(n=40;n>0;n--){
        if(N>=(1LL<<n)-1)break;
    }
    for(int i=1;i<=n;i++)
        p.push_back(i);
    N-=(1LL<<n)-1;

    for(int i=n-1;i>=0;i--){
        if(N<1LL<<i)continue;
        n++;
        p.insert(p.begin()+i,n);
        N-=1LL<<i;
    }
    return p;
}
```

D : Colorful Balls

ボールの色が 1 種類の場合，答えは 1 である．以降，ボールの色は 2 種類以上存在すると仮定する．一般に，3 つのボール a, b, c が存在して，a と b，a と c の位置がどちらも交換可能ならば，b と c の位置も必ず交換可能である．これは，図 1 に示されるように「a と b の位置を交換」，「a と c の位置を交換」，「a と b の位置を交換」という手順で操作を行うことで達成可能である．

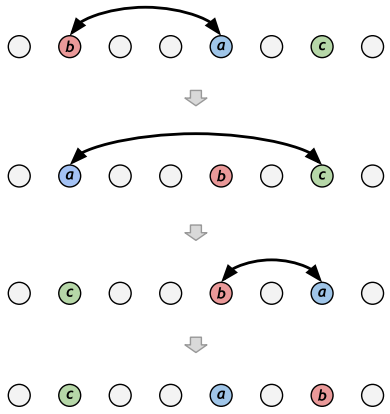


図 1　a と b，a と c が位置を交換可能なときに，b と c の位置を交換する操作

ある色 i に着目しているとき，色が i であるボールのうち最も軽いボールを x_i とする．このとき，色が i であって，操作 1 で x_i と交換可能なボールの重さは x_i の重さと等しいとしてもよいことが示せる．x_i と操作 1 により操作が交換可能なボール a，x_i と操作 2 により位置が交換可能なボール b があるとする．先程の事実から a と b の位置も交換可能であることが分かる．よって，操作 2 によって a と位置の交換が可能どうかは x_i の重さによってのみ定まることが言える．

操作 2 による位置の交換については N 個のボールのうち最も軽いボール y と，y と色が異なるボールのうち最も軽いボール z についてのみ考えれば十分である．これは，以下のことから示すことが可能である．

- y と色が異なり，かつ y と位置の交換が不可能なボールは，他のどのボールとも操作 2 では位置の交換が不可能．
- y と色が同じで，かつ z と位置の交換が不可能なボールは，他のどのボールとも操作 2 では位置の交換が不可能．

ここで，y から y と操作 2 により位置が交換可能なボールに辺を張り，z についても同様の操作を行ったグラフを考える．このグラフ上の同じ連結成分に含まれるボールたちは互いに位置の交換が可能なことが示せる (詳細は略)．最終的に互いに位置が交換可能なボールたちの並べ方を調べればよく，これは容易に達成可能である．全ての処理は $O(N)$ で達成可能であり，十分高速である．

E : Camel and Oases

まず，ラクダが N 箇所のオアシスを全て訪れることは可能か？という問題について考える．ラクダがまた訪れていないオアシスであって，歩いて到達可能なオアシスが存在するならば，先にそのようなオアシスを訪れた方がよいのは明らかである．ここから，ラクダが大ジャンプをするとき，現在の位置から歩いて到達可能なオアシスは必ず存在しないと考えてよい．

ラクダが大ジャンプをするとき，オアシスから移動する必要はないことも明らかであろう．すると，ラクダが大ジャンプ可能な回数を K とすると，K は $O(\log V)$ 程度となる．$V \leq 2 \times 10^5$ より，K は高々 18 以下である．以上のことをまとめると，この問題は以下のように言い換えられる．

> $d_0 = V, d_i = \lfloor \frac{d_{i-1}}{2} \rfloor$ を満たす長さ K の数列 d が与えられる．数直線上に並べられた N 個の点の間にいくつか仕切りを入れて $k(k \leq K)$ 個のグループに分割したい．グループには $0, 1, 2, \ldots, k-1$ の番号を割り振る．左から i 番のグループに割りふられた番号を p_i として，以下の制約を満たすように分割することは可能か？
>
> - 左から i 番のグループに含まれるどの隣り合った二点間の距離も d_{p_i} 以下である．
> - 左から $i-1$ 番目のグループの右端の点と，i 番目のグループの左端の点の距離は d_{p_i} より大きい．
> - 左から i 番目のグループの右端の点と，$i+1$ 番目のグループの左端の点の距離は d_{p_i} より大きい．

グループに左から順番に番号を割り振っていくときに K 個以下のグループで N 個の点を全てグループに割り振ることが可能かどうかを調べることにする．これは， dp(使った番号の集合) = 次のグループの始点として選べる点のうち，最も右側にある点，という bitDP が解ければよい．各状態において今いるグループに割り当てる番号の数は $O(K)$ 個である．さらに，次のグループの始点がどこになるかは $O(NK)$ で前計算しておけば $O(1)$ で調べられる． であるから，$O((N+V)\log V)$ でこの問題を解くことが可能である．

さて，元の問題では始点を全て求める必要があった．これは左から bitDP をした結果と，右側から bitDP した結果の 2 つを組み合わせることで解くことが可能である．0 番以外の番号付けをしたときに，残った点たちが存在するかどうか，残った点たちを 1 つのグループにすることが可能かどうか，の 2 つを調べればよい．これは $O(V)$ で可能である．よって，全体として $O((N+V)\log V)$ でこの問題を解くことが可能である．

F: Prefix Median

簡単のため, 与えられる数列が順列である場合から考える．このとき, $b_i < i$ あるいは $b_i > 2N-i$ を満たすような b は明らかに存在しない．その他，図 2 に示されるように，$2i-1$ 個の要素からなる集合にどのように 2 つ新しい要素を追加しても，以下のどちらかを満たすことが分かる．

- 新しい中央値も元の中央値のままである．
- 新しい中央値は元の中央値に隣接した位置にある．

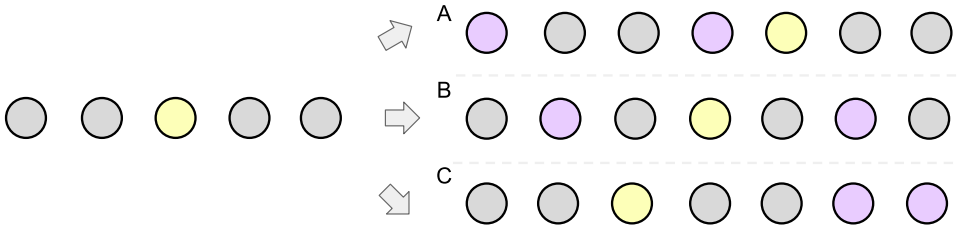


図 2　元の中央値と新しい中央値の位置関係

これらの事実は，数列 b に以下のような制約を与える．

- 制約 $1: i \leq b_i \leq 2N-i (1 \leq i \leq N)$
- 制約 $2: b_j < b_i < b_{j+1}$ なる $i, j (i < j)$ は存在しない．
- 制約 $3: b_j > b_i > b_{j+1}$ なる $i, j (i < j)$ は存在しない．

制約 $2, 3$ のうちどちらかに違反したとすると，上述した性質に違反するため，そのような数列が存在しないことが示せる．逆に，これらの性質を満たす数列であれば，実は必ず構成可能である．数学的帰納法を用いて示す．任意の自然数 k について長さ $2k-1$ の順列 a が与えられたとき，上の性質を満たす数列 b が全て構成可能である，と仮定する．$k=1$ のときは明らかに成立する．$k=i$ のときに成立するならば $k=i+1$ のときも成立することを示す．$k=i+1$ のとき，$b_{i+1}=i+1$ である．このとき，上の制約より b_i は $i, i+1, i+2$ のいずれかである．それぞれ場合分けをして考える．

$b_i = i$ のとき

図 3 に示されるように，$1, 2, 3, \ldots, i$ と $i+1, i+2, \ldots, 2i-1$ の 2 つのグループに分ける．後者のグループから $b_1, b_2, \ldots, b_{i-1}$ に含まれない値であって，最も i に近いものから 2 つ取り除く．後者のグループの大きさは $i+1$ であるため，必ず取り除くことが可能である．制約 2, 3 より，上のように 2 つの数を取り除いた集合において，b_{i-1} と b_i が等しい，あるいは b_{i-1} と b_i が隣接するという条件を満たすようにすることが可能である．$b_i = i+2$ の場合もほぼ同様である．

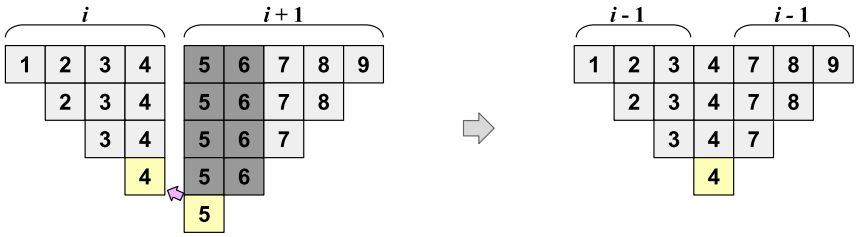


図 3　$b_i = i$ のときの 2 つの数の取り除き方

$b_i = i+1$ のとき

図 4 に示されるように，$1, 2, \ldots, i$ と $i+2, i+3, \ldots, 2i-1$ のグループに分ける．2 つのグループから $b_1, b_2, \ldots, b_{i-1}$ に含まれない値であって，最も $i+1$ に近いものから 1 つずつ取り除く．2 つのグループの大きさはどちらも i であるため，必ず取り除くことが可能である．制約 2, 3 より，上のように 2 つの数を取り除いた集合において，b_{i-1} と b_i が等しい，あるいは b_{i-1} と b_i が隣接するという条件を満たすようにすることが可能である．

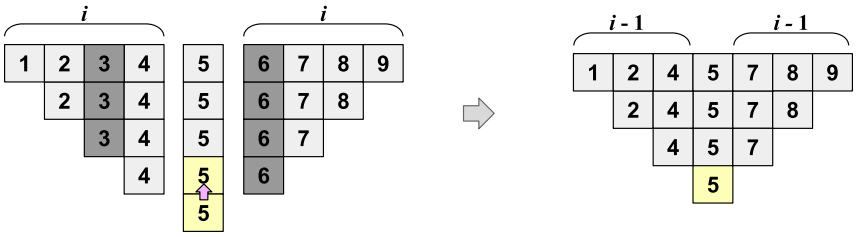


図 4　$b_i = i+1$ のときの 2 つの数の取り除き方

その後，残った数に $1, 2, 3, \ldots, 2i-1$ となるように数を振り直すことにすると，$k = i$ の場合に帰着させることが可能である．b_i の値に関わらず $k = i$ の場合に帰着させることが可能であり，$k = i$ の場合は構成可能であると仮定したので，$k = i+1$ も構成可能であることが示された．

以上より，与えられる数列が順列である場合，この問題は以下のように言い換えられる．

整数 N が与えられる．以下の条件を満たす数列 b はいくつあるか？

- $i \leq b_i \leq 2N - i (1 \leq i \leq N)$
- $b_j < b_i < b_{j+1}$ なる $i, j (i < j)$ は存在しない．
- $b_j > b_i > b_{j+1}$ なる $i, j (i < j)$ は存在しない．

これらの制約から， b が以下の特徴を持つことが分かる．

- b_i が取りうる値の候補の数は b_{i+1} のものから 2 ずつ増える．
- 一度飛び越した候補は 2 度と使うことができない．

以上のことから，$\mathrm{dp}(i, j, k) = i$ 項目以降が確定した時点で，j 個の候補が残っており，候補のうち k 番目の値が b_iであるような場合の数，という動的計画法で解くことが可能である．これを $\mathrm{dp}(N, 1, 1)$ から順番に埋めていけばよい．遷移は以下の 3 通りである．

- $\mathrm{dp}(i, j, k)$ から $\mathrm{dp}(i - 1, j + 2, k + 1)$ に加算
- $\mathrm{dp}(i, j, k)$ から $\mathrm{dp}(i - 1, j', k - |j + 2 - j'|)$ へ加算 $(j + 3 - k \leq j' \leq j + 2)$
- $\mathrm{dp}(i, j, k)$ から $\mathrm{dp}(i - 1, j', k + 2)$ へ加算 $(k + 2 \leq j' \leq j + 2)$

状態の数が $O(N^3)$，遷移が $O(N)$ となり，合わせて $O(N^4)$ で十分高速である．

最後に与えられる数列 a が順列でない場合について考える．a はソート済みであるとしても一般性を失わない．一般の数列の場合，値が同じもののうち，N に最も近い位置だけ中央値として採用されうる，とすると順列の場合とほぼ同様である．結論として以下の条件を満たす数列を数え上げればよい．

- $a_i \leq b_i \leq a_{2N-i} (1 \leq i \leq N)$.
- $b_j < b_i < b_{j+1}$ なる $i, j (i < j)$ は存在しない．
- $b_j > b_i > b_{j+1}$ なる $i, j (i < j)$ は存在しない．

b_i が取りうる値の候補の数が b_{i+1} のものから最大で 2 つ増えることがある，という条件のもと先程と同様に数え上げればよい．こちらについても計算量は $O(N^4)$ となる．

AGC 012 Editorial

writer : camypaper

April 1st, 2017

A : AtCoder Group Contest

Without loss of generality, we can assume that $a_1 \geq a_2 \geq \cdots \geq a_{3N}$. We prove that the answer is $a_2 + a_4 + a_6 + \cdots + a_{2N}$.

Sort the teams in the descending order of their strengths, and let p be the second strongest member in the i-th strongest team. Then, we know that at least $2i - 1$ people are stronger than p: two members each in the first $i - 1$ teams and the strongest member in the i-th team. Thus, the strength of p is at most a_{2i}, and the sum of team strengths is at most $a_2 + a_4 + a_6 + \cdots + a_{2N}$.

On the other hand, we can achieve this sum by forming teams in the following way: (a_1, a_2, a_{3N}), $(a_3, a_4, a3N - 1)$, (a_5, a_6, a_{3N-2}), $\cdots$ (i.e., when you form a new team, always choose the two strongest remaining people and the weakest reaming people).

This solution works in $O(N \log N)$.

B : Splatter Painting

Perform the operations in the reverse order: when you perform the operation i, you should color all uncolored vertices that are within a distance d_i from vertex v_i, in color c_i. (You never overpaint vertices). This will give the same answer as the original problem.

How do you paint all vertices that are within a distance d from vertex v? When you do this, you call a function $func(v, d)$. When $d > 0$, a vertex is within a distance d from v if one of the following is satisfied:

- The vertex is within a distance $d - 1$ from vertex v.
- The vertex is within a distance $d - 1$ from one of neighbors of v.

Thus, when $d > 0$, $func(v, d)$ recursively calls $func(v, d - 1)$ and $func(u, d - 1)$ for each adjacent vertex u. You should memoize this solution: when $func(v, d)$ is called, you memoize this fact, and never perform the recursions again when this is called again. The final color of a vertex v can be obtained by the first time the function $func(v, 0)$ is called.

This solution works in $O(Q + M \max d_i)$.

C : Tautonym Puzzle

Let p be a permutation of $(1, 2, 3, \ldots, n)$, and let $s = (p_1, p_2, p_3, \ldots, p_n, 1, 2, 3, \ldots, n)$.

How many good subsequences does s have? If tt is a subsequence of s, t must be a subsequence of both $(p_1, \ldots, p_n)$ and $(1, \ldots, n)$. Thus, the number of good subsequences of s is the same as the number of (non-empty) increasing subsequences of $(p_1, \ldots, p_n)$.

Let $f(p)$ be the number of (possibly empty) increasing subsequences of $(p_1, \ldots, p_n)$. We should solve the following problem:

> Construct one permutation of length up to 100 such that $f(p) = N + 1$.

We use the following facts:

- $f(emptysequence) = 1$
- For any permutation q of length k, $f(q_1, \ldots, q_k, k+1) = 2f(q_1, \ldots, q_k)$.
- For any permutation q of length k, $f(k+1, q_1, \ldots, q_k) = f(q_1, \ldots, q_k) + 1$.

You define a function that returns p such that $f(p) = X$ for a given X. When X is an even (or odd) number, it recursively calls the function with the parameter $X/2$ (or $X - 1$). This way we can construct a permutation whose length is up to $2lgN = 80$.

D : Colorful Balls

Construct a graph with N vertices. We add an edge between two balls s and t if s and t are swappable.

For three balls a, b, c, when a and b are swappable, and a and c are also swappable, we can prove that b and c are also swappable:

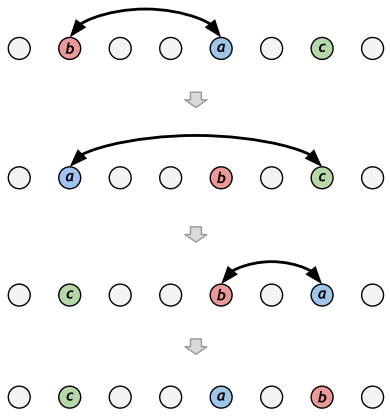


図 1　Swapping b and c

By performing possibly multiple operations, you can arbitrarily permute balls in the same connected component in this graph. Thus, our main objective in this problem is to find connected components in this graph. If we add edges between all pairs of swappable balls, the number of edges can be as large as N^2. Instead, we reduce the number of edges while keeping the connected components.

First, consider Operation 1, and in particular, consider balls with color i. Let x_i be the lightest ball of color i, and y_i, z_i be other two balls of color i. When $y_i + z_i \leq X$, we should add an edge betwen y_i and z_i. However, in this case, there are edges between x_i and y_i, and x_i and z_i. It means that even if we ignore the edge between y_i and z_i, it doesn't change the connected components. In Operation 1, we only need to consider pairs that involve the ball x_i, and we need at most $O(N)$ edges.

Next, consider Operation 2. Without loss of generality, we can assume the following:

- The colors are numbered 1 through C.
- For each i, the lightest ball with color i is a_i.
- $a_1 \leq a_2 \leq \cdots$

We claim that we only need to consider pairs that involve either a_1 or a_2.

When there is an edge between two balls s and t (i.e., their colos are different and $weight(s)+weight(t) \leq Y$),

- If $color(s) \neq 1$ and $color(t) \neq 1$, there is a path $s - a_1 - t$.
- If $color(s) \neq 2$ and $color(t) \neq 2$, there is a path $s - a_2 - t$.
- If $color(s) = 1$ and $color(t) = 2$, there is a path $s - a_2 - a_1 - t$.

This way, we again need only $O(N)$ edges.

The entire solution works in $O(N)$.

E : Camel and Oases

First, the camel visits some oases by walking. The set of visited oases in this step is an interval, and for each adjacent pair of two oases in this interval, the distance between them must be at most V.

Then, the camel jumps to another place and starts walking again. The set of visited oases in this step is again an interval, and this time the distance between adjacent oases must be at most $floor(V/2)$. The camel jumps again and starts walking, and so on.

In summary, the problem can be restated as follows:

> Let $\{v_0, v_1, \ldots, v_{k-1}\}$ be the sequence $\{V, floor(V/2), floor(V/4), \ldots, 0\}$ (it stops after the first appearance of zero). Can we divide the N points (oases) into k disjoint intervals such that for each i, the distance between two adjacent points in this interval is at most v_{p_i} (here p is some permutation of length k)? Also, for each point, determine if the point can be in the interval with v_0.

We do the following bitmask DP: for each subset S of $\{v_1, v_2, \ldots, v_{k-1}\}$, compute the length of the longest prefix that can be covered by the intervals corresponding to elements in S. Call it $dpL[S]$. Similarly, do the same thing for suffixes: call it $dpR[S]$. Since $k = O(logV)$, we can compute these arrays in $O(2^k k) = O(VlogV)$.

How to check if the camel can start from a particular point p? Let $[l, r)$ be the interval of points that can be reached only by walking from the point p. The answer is "Possible" if we can divide the set $\{v_1, v_2, \ldots, v_{k-1}\}$ into two disjoint sets S_1 and S_2, such that $dpL[S_1] \geq l$ and $dpR[S_2] \geq N - r$.

In total, this solution works in $O((N + V) \log V$.

F: Prefix Median

For simplicity, consider the case where $a_i = i$ for each i.

Obviously, $i \leq b_i \leq 2N - i$. Also, we can prove that for each i, b_i and b_{i+1} are either the same or adjacent in the sequence $a_1, \ldots, a_{2i+1}$. See the following picture:

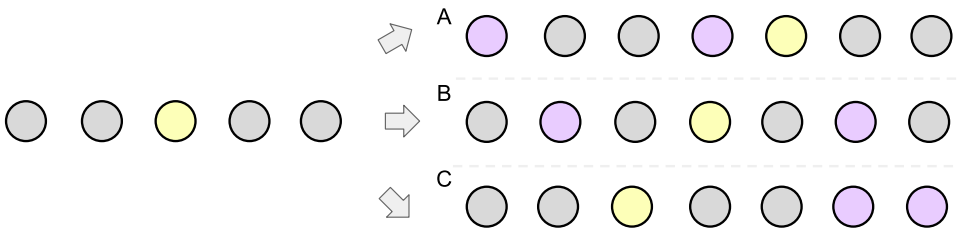


図 2　The median of a sequence and the median of new sequence after adding two elements

Thus, b must satisfy the following conditions:

1. $i \leq b_i \leq 2N - i (1 \leq i \leq N)$
2. No $(i, j)(i < j)$ satisfies $b_j < b_i < b_{j+1}$.
3. No $(i, j)(i < j)$ satisfies $b_j > b_i > b_{j+1}$.

We claim that these conditions are sufficient. By induction, it is sufficient to prove the following:

Let $b_1, \ldots, b_i + 1$ be the sequence that satisfies the three conditions above. That is,

- b_{i+1} is the median of the $2i + 1$ numbers.
- b_i is one of the three middle numbers of the $2i + 1$ numbers.
- b_{i-1} is one of the five middle numbers of the $2i + 1$ numbers.
- :
- No $(p, q)(p < q)$ satisfies $b_q < b_p < b_{q+1}$.
- No $(p, q)(p < q)$ satisfies $b_q > b_p > b_{q+1}$.

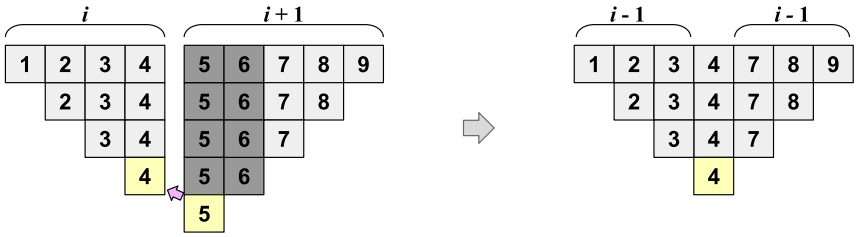
Then, we can remove two of numbers $1, 2, \ldots, 2i + 1$ such that:

- The numbers $b_1, \ldots, b_i$ remain after the removal of two numbers.
- b_i is the median of the remaining $2i - 1$ numbers.
- b_{i-1} is one of the three middle numbers of the remaining $2i - 1$ numbers.
- b_{i-2} is one of the five middle numbers of the remaining $2i - 1$ numbers.
- :

There are three cases: $b_{i+1} = i, i+1, i+2$. We consider each of these cases separately:

The case where $b_i = i$

Consider the numbers $\{i+1, \ldots, 2i+1\}$. Since there are $i+1$ numbers, at least two of these don't appear in $b_1, \ldots, b_{i-1}$. It is easy to prove that if we remove the two smallest such numbers, the conditions will be satisfied.

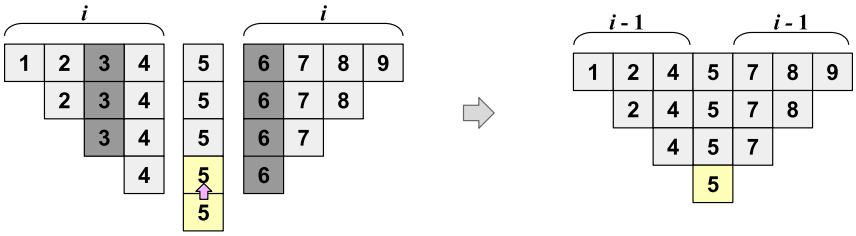


The case where $b_i = i + 2$

Similar to the case above.

The case where $b_i = i + 1$

Remove the greatest number in the set $\{1, \ldots, i\}$ that doesn't appear in $\{b_1, \ldots, b_{i-1}\}$. Also, remove the smallest number in the set $\{i+2, \ldots, 2i+1\}$ that doesn't appear in $\{b_1, \ldots, b_{i-1}\}$. Again, it is easy to prove that if we remove these two numbers, the conditions will be satisfied.



In general case (where a_i are not necessarily pairwise distinct), the conditions (and the proof) are very similar. Assume that $a_1 \leq a_2 \leq \cdots \leq a_{2N-1}$. Then,

1. b_i is one of the numbers $\{a_i, a_{i+1}, \ldots, a_{2N-i}\}$.
2. No $(i, j)(i < j)$ satisfies $b_j < b_i < b_{j+1}$.
3. No $(i, j)(i < j)$ satisfies $b_j > b_i > b_{j+1}$.

How to count the number of such sequences? Determine the sequence in the order $b_N, \ldots, b_1$.

Then, the last two conditions mean the following: when we "jump over" a number x, we can never use x later in the sequence.

This will lead to an easy DP. Define $dp[i][j][k]$ as the number of ways to choose the first i elements of the sequence (in the reverse order), such that there are j candidate numbers for the next element, and the number we chose the last (i.e., the i-th element) is the k-th smallest among the j candidate numbers. It works in $O(N^4)$.