たのしい家庭菜園
(Growing Vegetables is Fun)

JOI Spring Camp 2014 Day-1

waf

問題概要

- IOI草が列になってたくさん (n個) 生えている。
- IOI草は以下のどちらかの条件が満たされれば枯れずにすむ。
  - その草より左側に、その草より高い草が存在しない。
  - その草より右側に、その草より高い草が存在しない。
- うまくIOI草を並べ替えて、すべてのIOI草が枯れないようにする。
- 最小で何回の並べ替えをすれば良いか計算してください。

とりあえず全探索

- とりあえずよくわからないので**全探索**をする。
- 許される配置を列挙して、それぞれの配置の並べ替え回数を計算する。

全探索の実装をしてみる

- 考えられる配置を std::next_permutation(C++) / 再帰関数 などを使って適当に列挙したあと、条件を満たしているかどうか確認する。
- 端から順番に揃えていくと、無駄な並べ替えが起こらず並べ替え回数が最小となるので、並べ替えを実際にシミュレートして、並べ替え回数を計算する。
- 考えられる配置の列挙に $O(n!)$ かかって、並べ替え回数の計算に $O(n^2)$ かかるので、全体で時間計算量 $O(n! \times n^2)$ となって小課題1までは解けそう。

許される配置を考える

- どういう配置が許されるのかもっと分かりやすい言い換えを考える。
- いちばん大きな草は最終的にどの位置にいても枯れることはない。
- 二番目に大きな草は、いちばん大きな草たちの間に挟まれなければ良い。つまり、いちばん大きな草のうち、最も左側にある草のさらに左側か、最も右側にある草のさらに右側に配置すれば良い。
- どの草もそれより大きな草より**外側**に配置すれば良い。

わかりやすい図

問題の整理

- 整数列が与えられる。
- 隣り合った要素を入れ替えることを繰り返す。
- ある要素までは**広義単調増加**、そこから先は**広義単調減少**、となるように整数列をうまく並び替えたい。
- 最小で何回の入れ替えが必要か計算してください。

頭の良い解法？

- 初期状態の列を、ある一点で切り分けて、左側/右側のそれぞれの列でソートして、最後に連結する。
- 切り分ける点をすべて試して最小の並べ替え回数を計算する。
- 素朴に実装すると $O(n^3)$ かかるが、並べ替え回数を計算するところを高速化すれば、満点が取れそう。

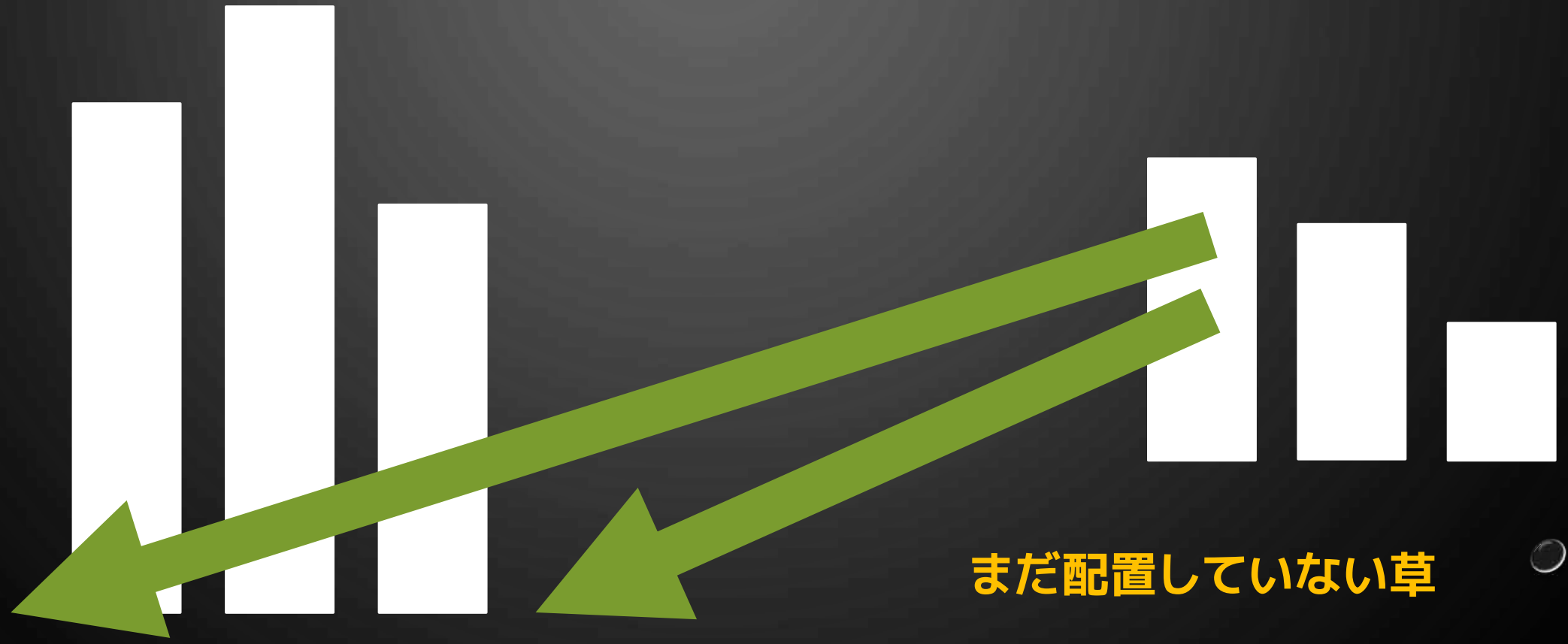
というのは嘘で

- [6, 5, **1**, 4, 3, 2] というテストケースに対して正しく答えることができない。
- 正しい答えは [**1**, 6, 5, 4, 3, 2] で 2回 となるが、先ほどの嘘解法では 3回 となってしまう。
- ちゃんと考えなおしましょう。

ちょっと頭の良い全列挙

- 最終的に許される配置のみをすべて列挙していく。
- まずいちばん高い草を中央に配置する。
- より高い草から順番に、すでに出来上がっている配置の左右どちら側に配置するか、で**全探索**していく。
- 並べ替え回数の計算のアルゴリズムは先ほどのものと一緒で良い。
- 列の列挙に $O(2^n)$ 、並べ替え回数の計算に $O(n^2)$ 、高い順に草をリストアップするのに $O(n \log n)$ かかって、全体で $O(2^n \times n^2)$ となるので小課題2までは解けそう。

わかりやすい図 その2



すでに配置が決まった草

まだ配置していない草

並べるときの注意点

- 同じ高さの草が**複数**ある場合を考慮する必要がある。
- 同じ高さの草たちを入れ替えるのは明らかに無駄であるから、同じ高さの草たちだけを取り出した場合、それらの中では入れ替えは必要ない。
- 同じ高さの草が M 個あるとき、初期状態で列の左端に近いものから順に L 個選んで、**順番を保ったまま**、すでに出来上がってる列の左側に挿入する。残った $M - L$ 個も、**順番を保ったまま**、すでに出来上がってる列の右側に挿入する。これをすべての L に対して試す。
- 時間計算量は変わらない。

まだまだ無駄がたくさんある

- 高い草から順番に配置しているので、ある草を配置しようとしているとき、それ以前に配置した草たちの順列は**無視**してしまって良い？
- それ以前に配置した草たちの数が一緒ならば、そのなかで並べ替え回数が最小のもの以外の配置のしかたは、すべて切り捨ててしまって良い？

かんたんなDP

- $dp[i] \coloneqq i$ 番目に大きな草まで並べたときの最小の並べ替え回数
- $a[i] \coloneqq i$ 番目に大きな草を列に加えるときの最小の並べ替え回数
- $dp[i] = dp[i\ -1] + a[i]$
- $a[i]$ を計算するには……

無理なら逆から並べてみる

- 列が中央から出来上がっていくので並び替え回数を数えるのが難しい。
- それ以前の草の配置を無視したらダメそう。
- 高い草から順番に配置してダメなら、低い草から順番に配置してみる。
- 列が左端/右端から出来上がっていくのでうまくいきそう。
- ある草を配置しようとしているとき、それ以前に配置した低い草たちの順列は完全に無視してしまって良い。
- 小さい草から動かして、端から列を構築していくイメージ。

わかりやすい図 その3 (整列途中)

ふつうのDP

- $dp[i] \coloneqq i$ 番目に**小さな**草まで並べたときの最小の並べ替え回数
- $a[i] \coloneqq i$ 番目に**小さな**草を列に加えるときの最小の並べ替え回数
- $dp[i] = dp[i\ -1] + a[i]$
- $a[i]$ は計算できそう。

ふつうのDPを実装する (1)

- i 番目に小さな M 個の草のなかで、初期状態で列の左端に近いものから順に L 個選んで、順番を保ったまま、左側に出来上がっている列の末尾に挿入する。
- 残った $M - L$ 個も、順番を保ったまま、右側に出来上がっている列の末尾に挿入する。
- すべての L について試して最小の入れ替え回数となるものを計算する。

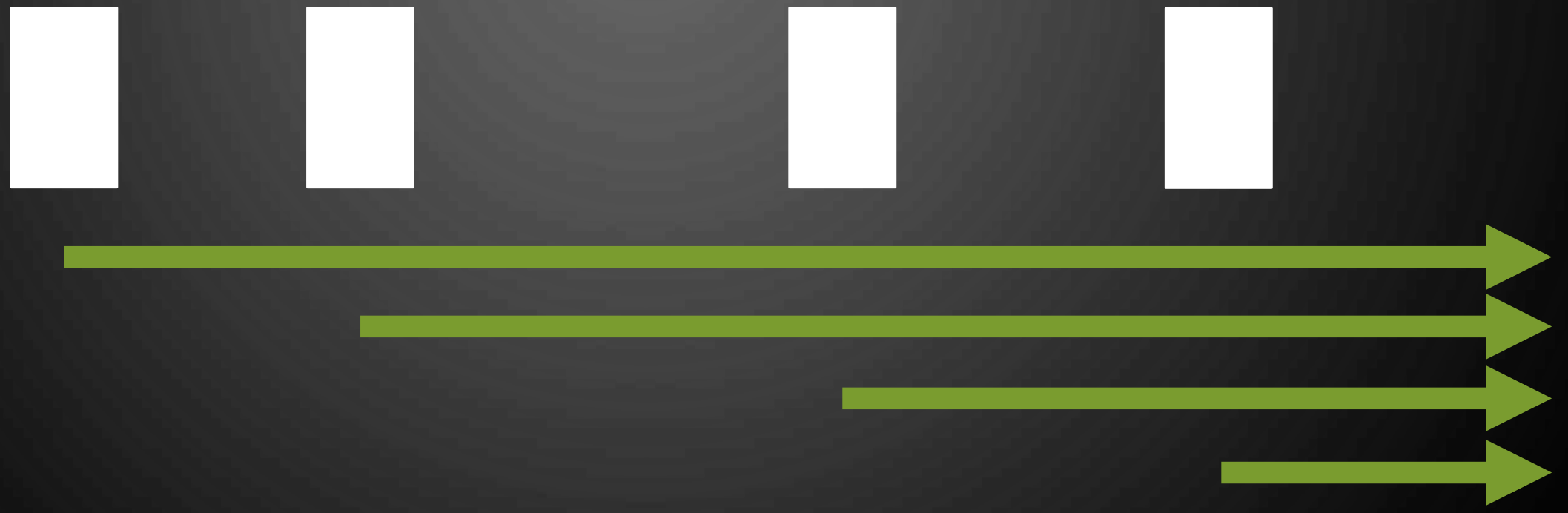
ふつうのDPを実装する (2)

- 高さが h で初期位置が x であるような草を左端の列に付け足すのに必要な入れ替え回数は、初期位置が $0,1,2,\ldots\ldots,x-1$ であるような草のなかで高さが h より高いものの個数と**一致**する。右端にもってくる場合も同様。
- 素朴に計算すると、各 L について $O(Mn)$ かかるので、ひとつの高さについて $O(M^2n)$ かかる。全体として $O(n^3)$ になる。
- 小課題3が解けない。つらい。

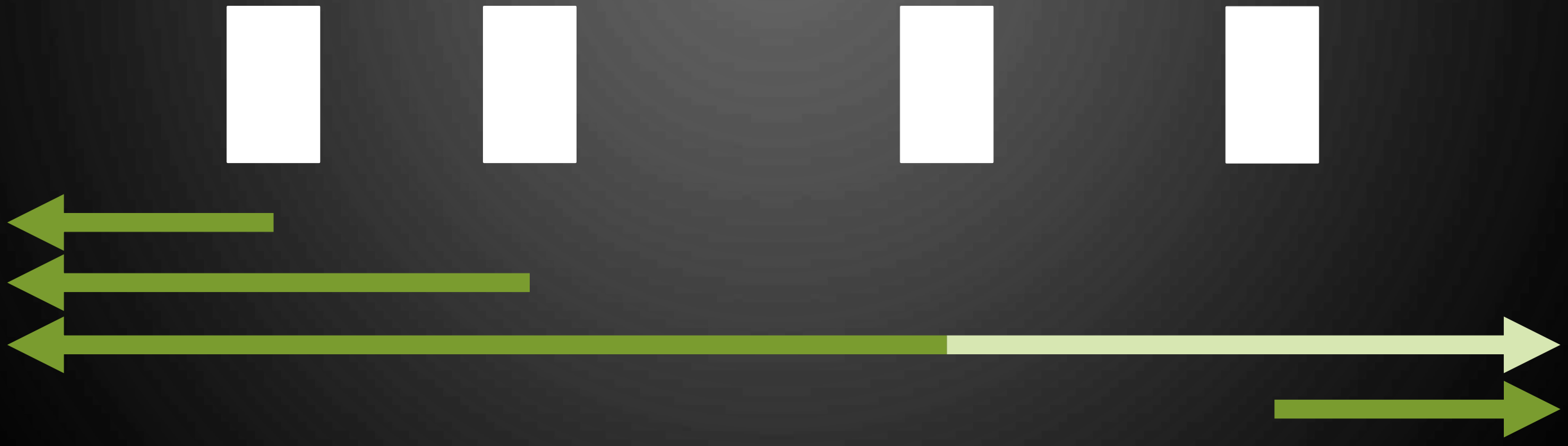
すこし高速化する

- 高さ h である M 個の草によって分割される $M+1$ 個の**区間**について考える。
- M 個すべての草を右側に配置する場合は、右側から $i+1$ 番目の区間を i 個の草が通過する。 このときの入れ替え回数は、各区間に存在する高さ h 以上の草の個数を数え上げることで $O(n)$ で計算できる。
- $L+1$ 個の草を左側に配置する場合の入れ替え回数は、(L 個の場合の入れ替え回数) – ($L+1$ 個目の草を右に配置するための入れ替え回数) + (左に配置するための入れ替え回数) と等しく、これは累積和を用いて、前計算に $O(n)$ かけることで $O(1)$ で計算できる。

わかりやすい図 その4（すべて右端）



わかりやすい図 その5 (1つ左に移す)



ふつうよりちょっと速いDP

- 高い順に草をリストアップするのに $O(n \log n)$ かかり、同じ高さの M 個の草についてまとめて $O(n)$ で計算できる。全体として $O(n^2)$ で計算できるので、小課題3までは解けるようになった。
- しかし小課題4を解くにはまだまだ遅い。

もっともっと高速化する

- 各区間に存在する高さ h 以上の草の個数を数えるところが遅い。
- この操作を高速に行えるデータ構造さえあれば……
- ん？ 区間に対する高速な操作？

Segment Tree

- JOI界では定番のデータ構造です。
- いろいろと応用が効く。
- 列に対する操作を高速化したいときに使えることがたまにある。
- ただし、慣れていないと実装が面倒である。

BITで実装する

- サイズ n のBITを用意する。まず、それぞれの草を初期位置に置いて、BITの各要素と対応付ける。 BITの各要素を1で初期化する。
- さきほどのDP解と同じように小さな草から順番にすべてを見ていく。
- 草を順番に見ていくとき、ついでにいま注目している高さの草に対応する要素をすべて0にする操作を行っておく。こうすることで、いま注目している草より高い草に対応するBITの要素のみ1である、という状態が**常に保たれる**。
- ある区間のなかにある、いま注目している草より高い草の個数は、BITのsumを用いることで効率的に求められる。

高速化の評価

- それぞれの草に対応する要素を順番に0にしていくので、全体として更新操作は $O(n \log n)$ で行える。
- 同じ高さの M 個の草について注目しているとき、すべて右側に配置した場合の回数を計算するのに $M+1$ 回、ひとつ左側に移し替えるのに 2回、sumが行われるので、全体として総和計算操作は$O(n \log n)$ で行える。
- 低い順に草をリストアップするのも $O(n \log n)$ で行える。
- 全体として $O(n \log n)$ で計算できるので、すべての小課題が解けそう。