IOI 列車で行こう
Take the 'IOI' train

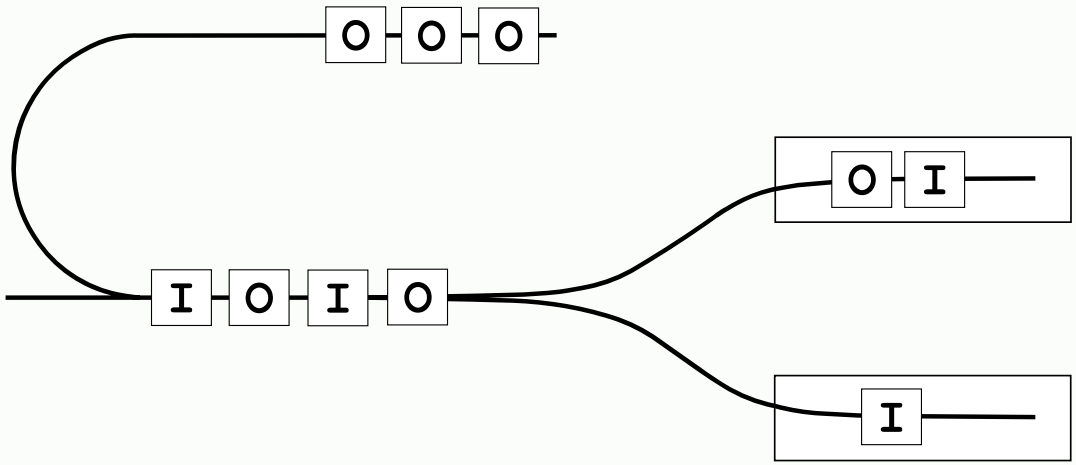
今西 健介 (@japlj)

2013/02/10, 情報オリンピック本選

問題概要

Take the 'IOI' train

- 2つの車庫に車両がたくさん入っている
  - 車両 I, O の2種類がある
- 順に取り出して IOIO...I と連結していく
  - 最初にいくつかの車両を退避させてもOK
  - 最長の列車は？



解法説明の前に……

Take the 'IOI' train

- なにやら面倒そうな条件
  - 列車編成前に**好きなだけ**車両を待機用レールに移してOK
  - 車両を使いきらずに**好きなだけ**残してOK

→　**問題の言い換え**をしよう!!!

問題の言い換え

Take the 'OI' train

- 退避とか使い残しとか気にしない
- 全部一列に並べてしまう

出来た列からできるだけ長い

IOIOIOI...OI をとってくると考える

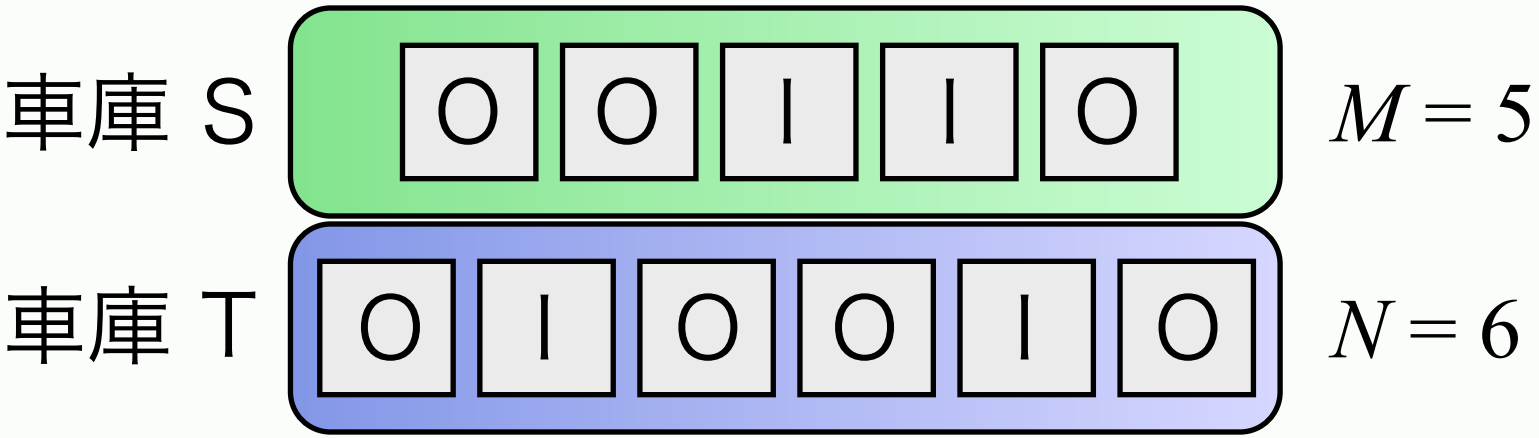
→ この列の作り方が問題になる

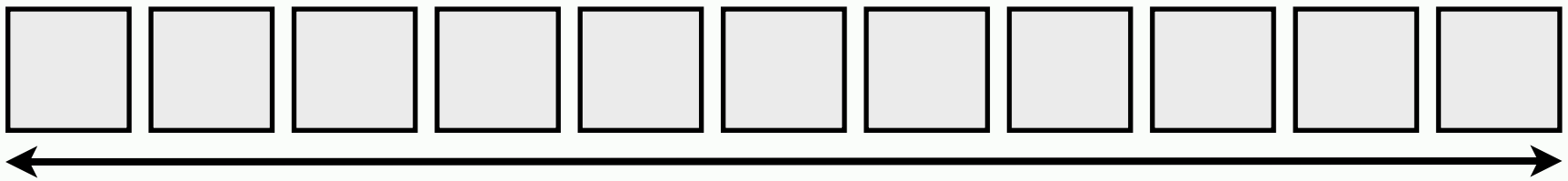
20点解法('⊙ˬ⊙')

20点解法(全探索)

Take the 'IOI' train

- 作れる列を全部試すことを考えよう



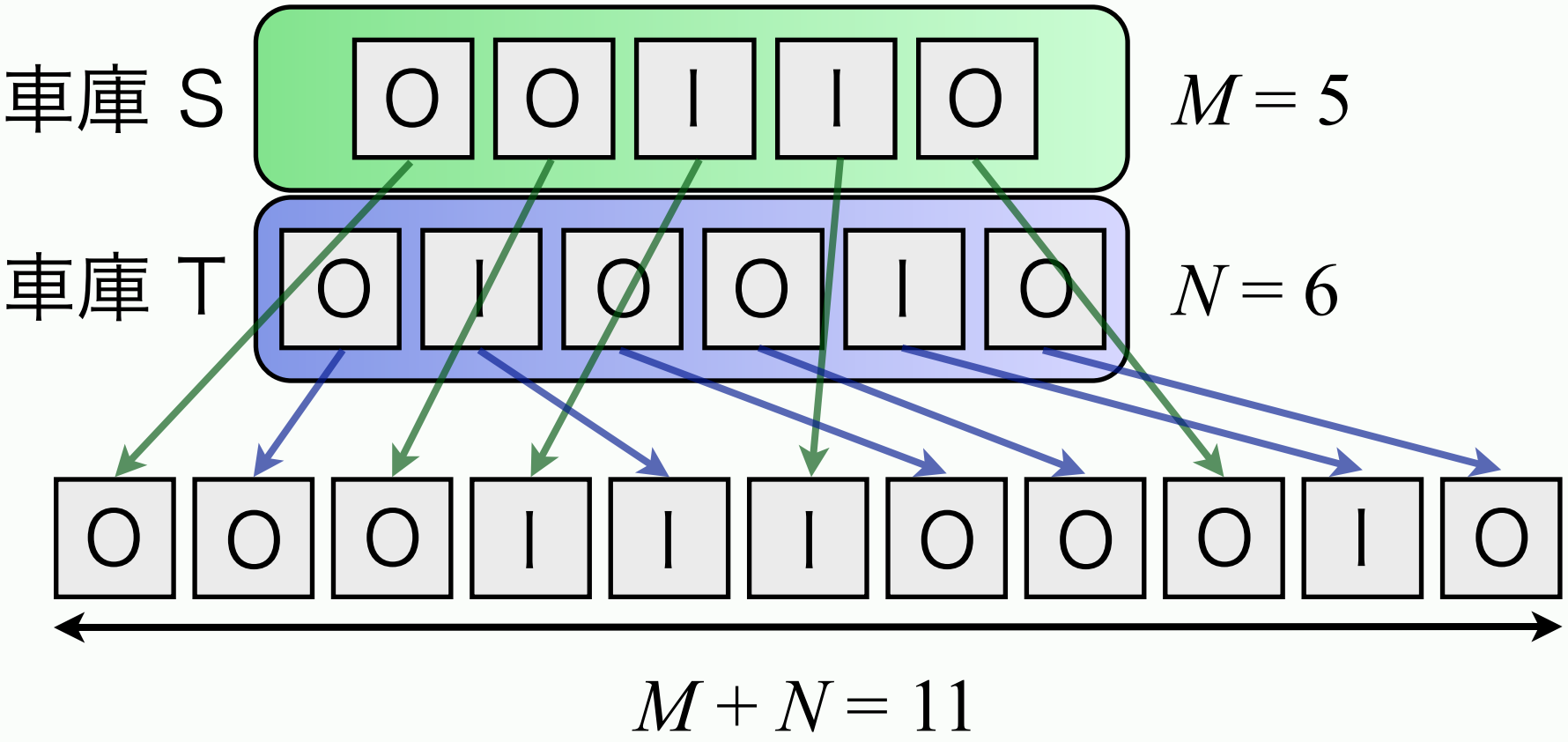


$$M + N = 11$$

20点解法(全探索)

Take the 'OI' train

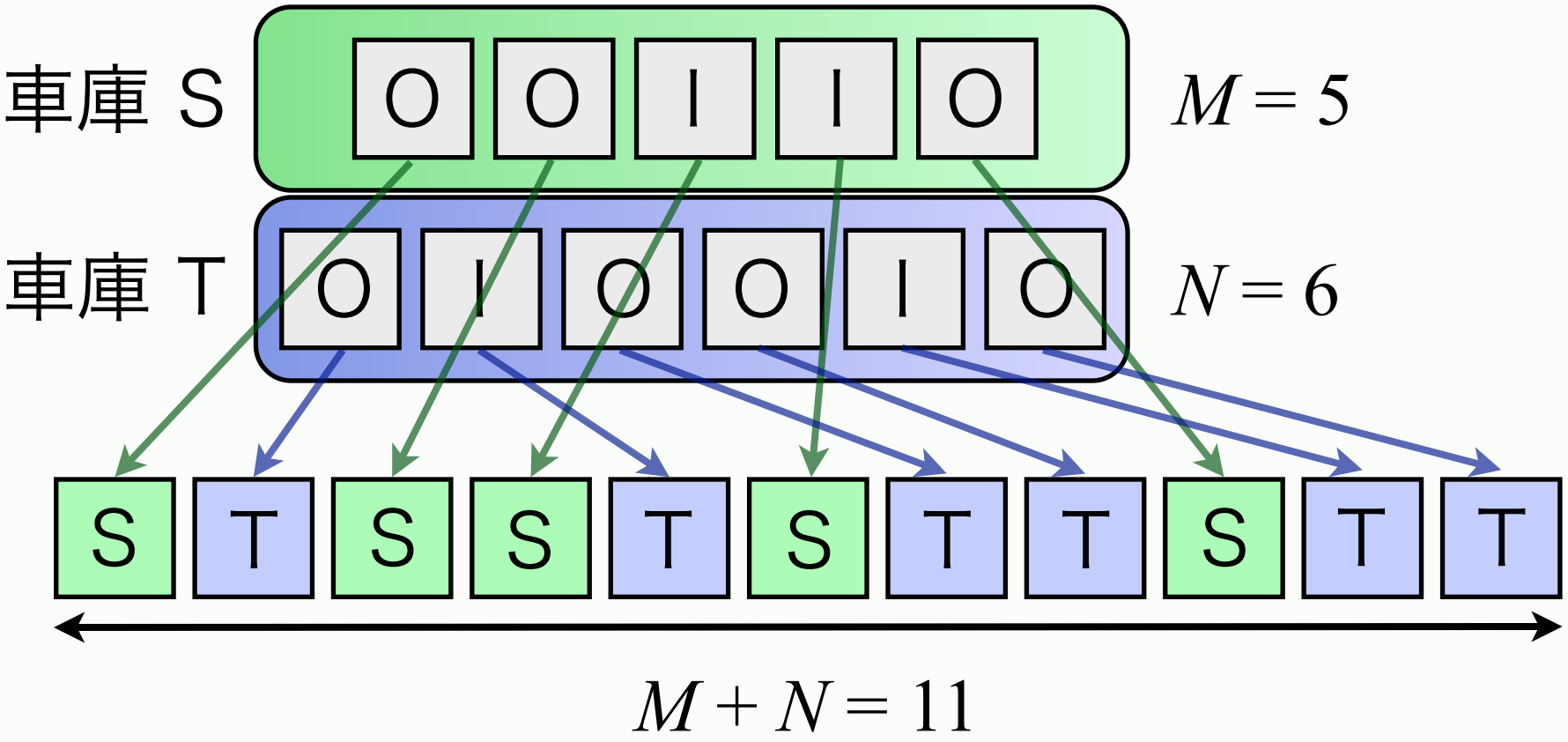
- 作れる列を全部試すことを考えよう



20点解法(全探索)

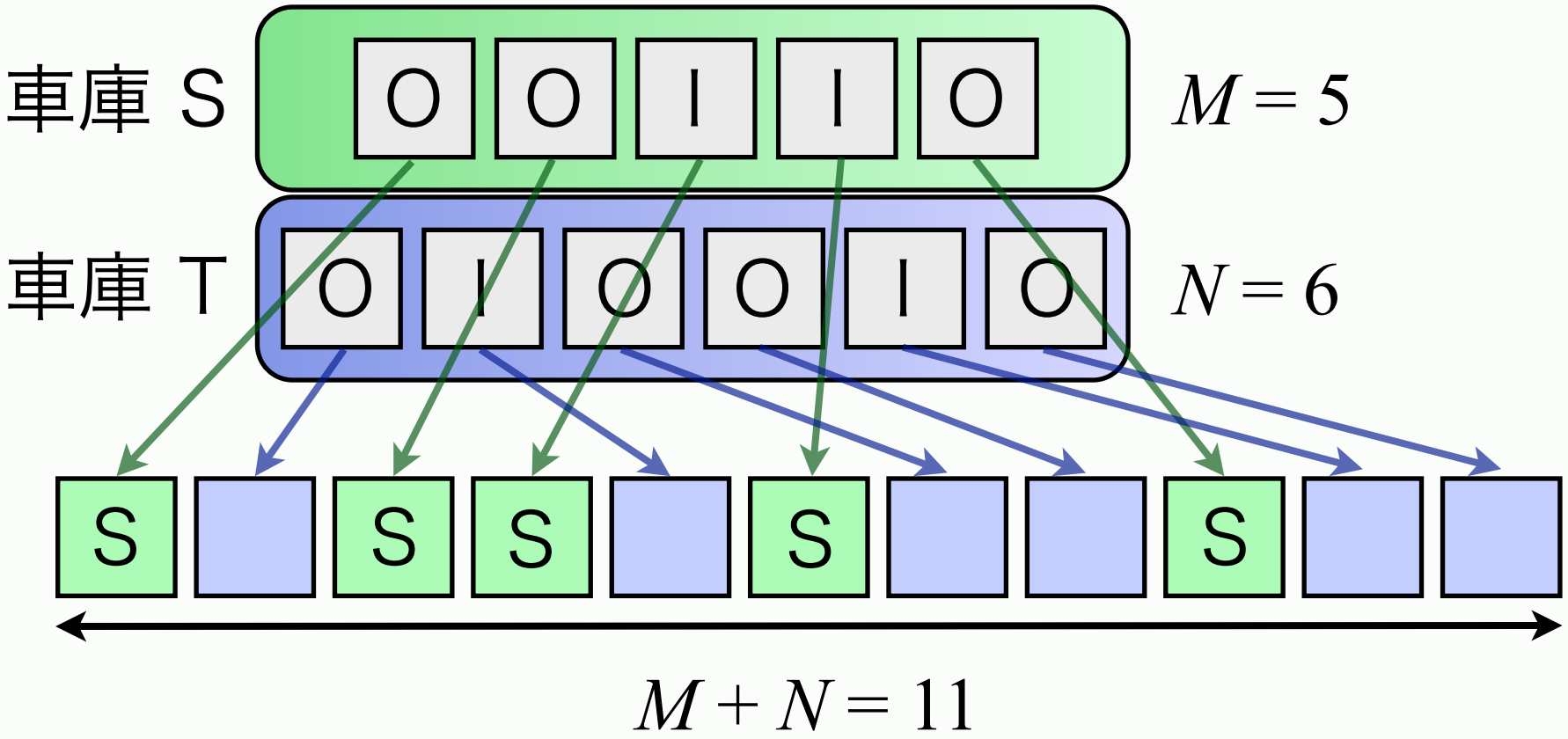
Take the 'IOI' train

- S, Tのどっちから来るかが決まると列も決まる



20点解法(全探索)

- というか S だけ決めれば T も決まる



20点解法(全探索)

- 作れる列は全部で S の決め方と同じだけある
- これは $M+N$ 個から M 個を選ぶ**組み合わせ**

n 個のものから k 個を選ぶ方法の総数

$$
{}_nC_k = \binom{n}{k} = \frac{n!}{k!(n-k)!}
$$

$$
= \frac{n \times (n-1) \times \cdots \times (n-k+1)}{k \times (k-1) \times \cdots \times 1}
$$

20点解法(全探索)

- 何通りの列を調べなければならないか？
  - 20点分のケースでは $M, N \leqq 10$
  - 50点分のケースでは $M, N \leqq 50$
  - 100点分のケースでは $M, N \leqq 2{,}000$
- ${}_{20}C_{10} = \mathbf{184{,}756}$ → いける
- ${}_{100}C_{50} > \mathbf{10^{29}}$ → やばい
- ${}_{4000}C_{2000} > \mathbf{10^{1202}}$ → やばすぎ

50点解法(｢◡｢)

動的計画法

Take the 'IOI' train

- 20点より多くとるには**動的計画法**を使う
  - Dynamic Programming, DP

- 動的計画法ってなんだろう？
- 調べてみよう！！！！

動的計画法

Take the 'IOI' train

動的計画法

動的計画法（どうてきけいかくほう、英: Dynamic Programming, DP）は、コンピュータ科学の分野において、ある最適化問題を複数の部分問題に分割して解く際に、そこまでに求められている以上の最適解が求められないような部分問題を切り捨てながら解いていく手法である。分割統治法がトップダウン的な手法であるのに対し、動的計画法はボトムアップ的な手法といえる。

(Wikipedia)

? ? ? ? ? ? ?

Take it easy

Take the 'OI' train

(少なくとも競技プログラミングで問題を解く時は)
次のような認識でOK

- 途中までの計算結果を次の計算に利用する
- 一段階小さい(一段階前の)**状態**から，その次の**状態**の解を求める
  - 基本的に状態の数は少ないほど効率が良い

状態のとり方

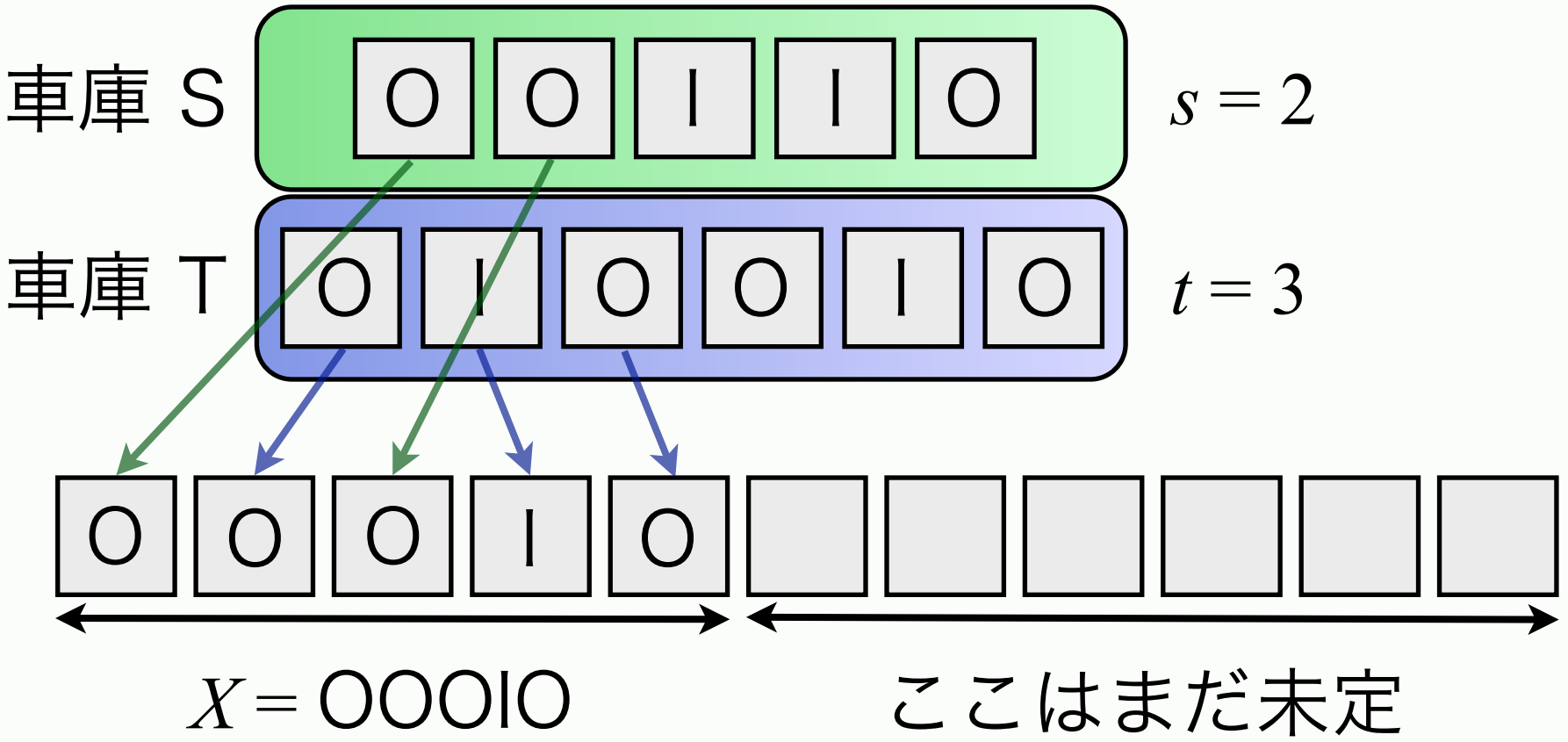
Take the 'IOI' train

- この問題で状態はどういう風にとれるだろう
  - たとえば次のわかりやすい状態のとり方はどうか？

- 車庫 S から先頭 s 両をすでに出していて
- 車庫 T から先頭 t 両をすでに出していて
- 今並んでいる車両の列が X である
- という状態

状態のとり方(例)

- ここまでは決まっている，という状態



状態のとり方(例)

Take the 'IOI' train

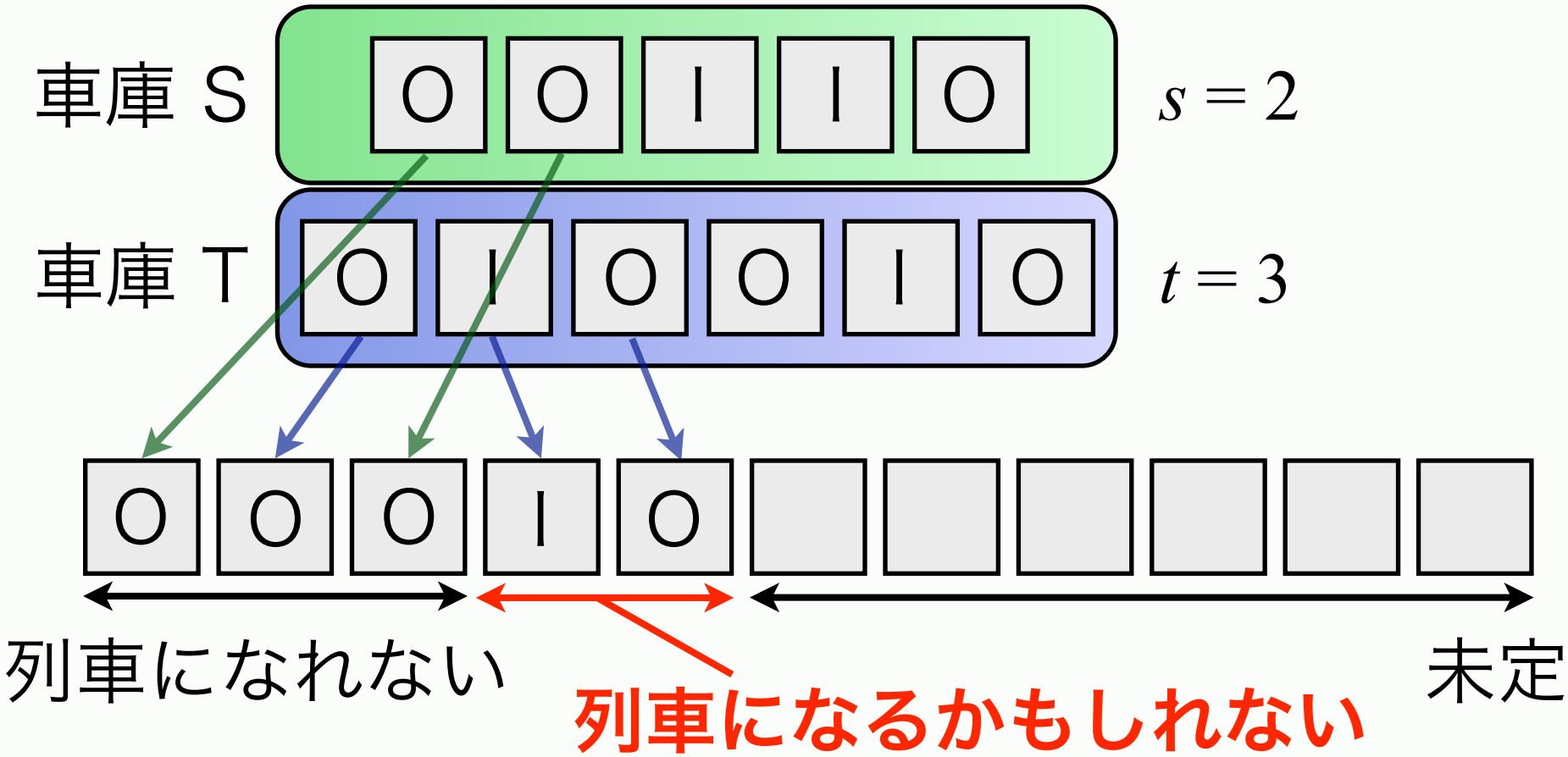
- これも立派な状態のとり方だが……？
- 状態数が**多すぎ**る！
  - X は結局列の総数ぐらいのパターンがある
  - 全探索するのと変わらない！

→　X から無駄を省けないか？

状態のとり方

Take the 'OI' train

- X の末尾の列車になりうる部分に注目



状態削減

Take the 'IOI' train

- 列車になれない部分はどうでもいい
- **今後の選択や解に影響を与えない**
  - こういう部分は状態から削減してしまって構わない

- X をそのまま状態とする必要はない！
- **「X の末尾の列車になりうる部分の長さ(L)」**
  さえ分かっていればOK！！！
  - IOIO... という形の列であることは分かっているから

動的計画法の立式

- 状態のとり方を決めた
- ある状態から次の状態を計算する式を立てる

- 初期状態は $s = t = L = 0$
  - 車庫 S からも T からも 1 つも車両を出していない
  - もちろん列車になれる部分の長さも 0

動的計画法の立式

- $dp(s, t, L)$:=
  - 初期状態から開始したときに
  - 車庫 S からは s 両
  - 車庫 T からは t 両の車両を出していて
  - 並べた列の列車になれる部分の長さが L
  - という状態に到達できるかどうか(true / false)
- もちろん $dp(0, 0, 0) = \text{true}$

動的計画法の立式

Take the 'IOI' train

- $dp(s, t, L) = \mathrm{true}$ のとき次の1手は？
  - $s < \mathrm{M}$ なら，車庫 S から次の車両を出せる
  - $t < \mathrm{N}$ なら，車庫 T から次の車両を出せる
- L がどう変化するかをしっかり考える
  - L が奇数 → 次に車両 O が来ると L が 1 増える
車両 I が来ると L が 1 になる
  - L が偶数 → 次に車両 I が来ると L が 1 増える
車両 O が来ると L が 0 になる

動的計画法の立式(まとめ)

Take the 'IOI' train

$dp(s, t, L) = \text{true}$ であれば

$s < M$ なら車庫 S の $s + 1$ 個目の車両を出す

その結果 L の値が L' に変わったとすると
$dp(\boldsymbol{s + 1}, t, \boldsymbol{L'}) \leftarrow \text{true}$

[tについても同じことをやる]

s, t, L について小さい方からループを回して

上の処理をやる

計算量

Take the 'IOI' train

- 状態数と，次の状態への計算量を考えよう
  - s, t はそれぞれ M, N 通り
  - L は $s + t \leqq M + N$ 通り
  - 次の状態への計算は $O(1)$ でできる
- 状態数は $O(MN(M+N))$, 遷移は $O(1)$
- 全部で $O(MN(M+N))$
  - 50点分のテストデータが処理できる
  - 100点分はまだ時間制限が厳しい

100点解法(・`д・´)

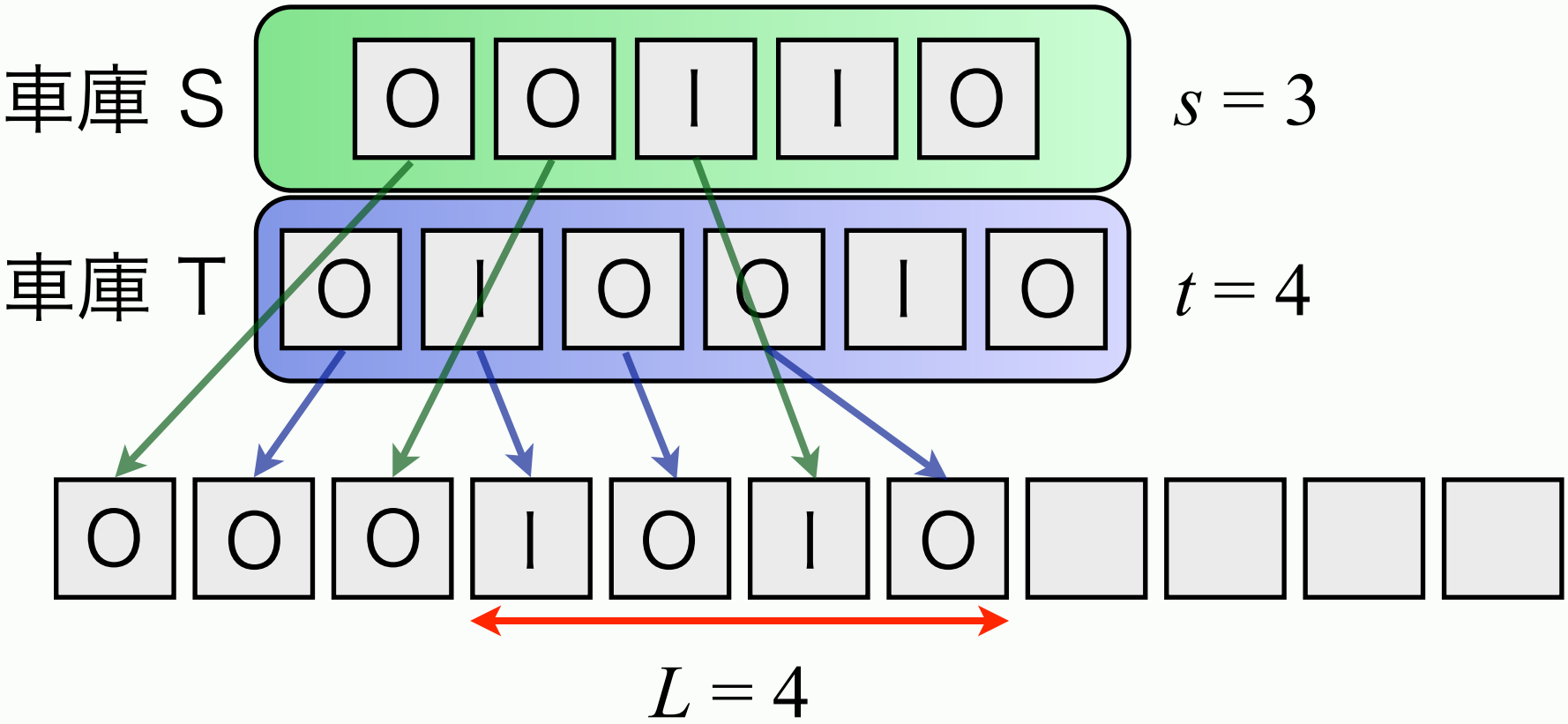
まだ無駄が？

Take the 'IOI' train

- 50点解法のDPにはまだ無駄がある？
- 次に示すふたつの状態を見てみよう
  - 列車になれる部分の長さに注目

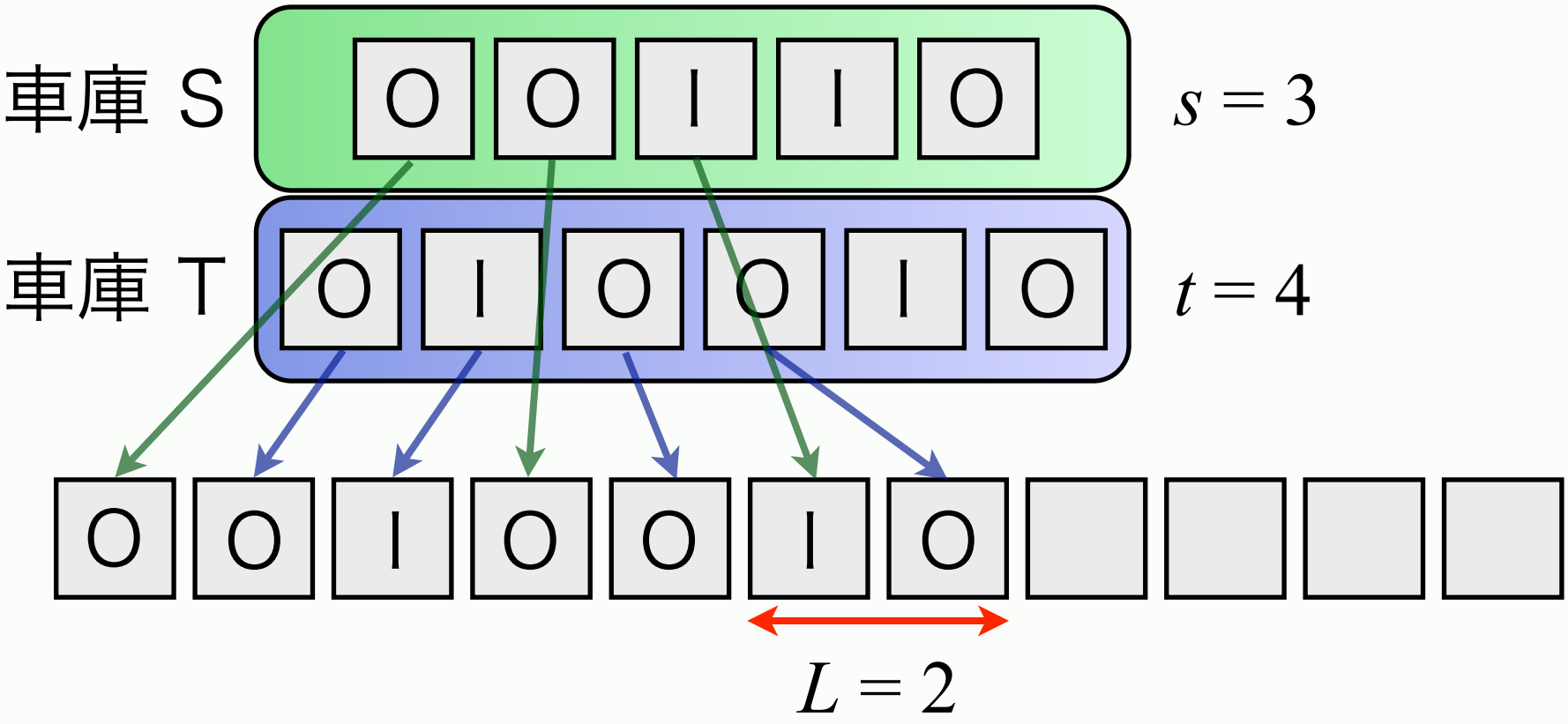
ある状態A

Take the 'IOI' train



別の状態B

Take the 'IOI' train



無駄はどこか？

Take the 'IOI' train

- 状態 A, B を比べよう
  - s, t の値は両方とも同じ
  - L の値が違う！(ただし偶奇は同じ)
- s, t が同じで L の偶奇も同じなら
  L は大きいほうがいいに決まっている！
- この無駄を省いた状態のとり方を考えよう

効率的な状態のとり方

Take the 'OI' train

- $dp(s, t, p)$:=
  - 車庫 S からは s 両
  - 車庫 T からは t 両の車両を出していて
  - 列車になれる部分の末尾の車両が p (I / O) のとき
  - 列車になれる部分の**長さ L の最大値**

動的計画法の立式

- $dp(s+1, t+1, \mathrm{I})$ を計算するときは
  - S の $s+1$ 番目の車両が I なら $dp(s, t+1, \mathrm{O})+1$ を考慮
  - T の $t+1$ 番目の車両が I なら $dp(s+1, t, \mathrm{O})+1$ を考慮
- して，考慮した値のうち大きい方を採用する

$res \leftarrow 0$

if $S[s+1] = \mathrm{I}$ then $res \leftarrow \max\{res, dp(s, t+1, \mathrm{O})+1\}$

if $T[t+1] = \mathrm{I}$ then $res \leftarrow \max\{res, dp(s+1, t, \mathrm{O})+1\}$

$dp(s+1, t+1, \mathrm{I}) \leftarrow res$

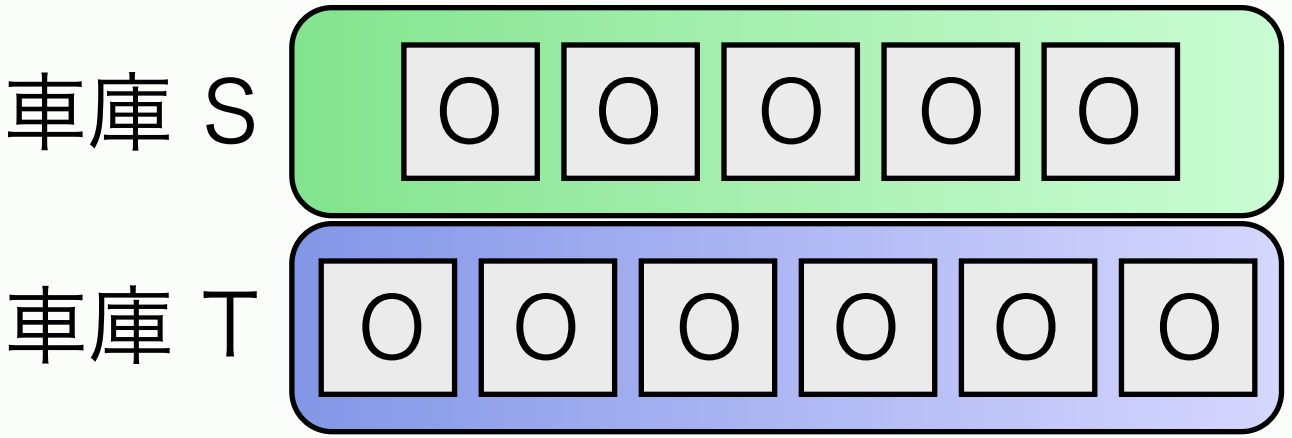
計算量

- 状態からは L が削減，その偶奇だけ考える
- よって状態数 $O(MN)$
- 遷移は変わらず $O(1)$
- 合わせて全体で $O(MN)$
- $M, N \leqq 2{,}000$ でも間に合う!!!
- ようやく 100 点
  - おつかれさまでした

コーナーケース

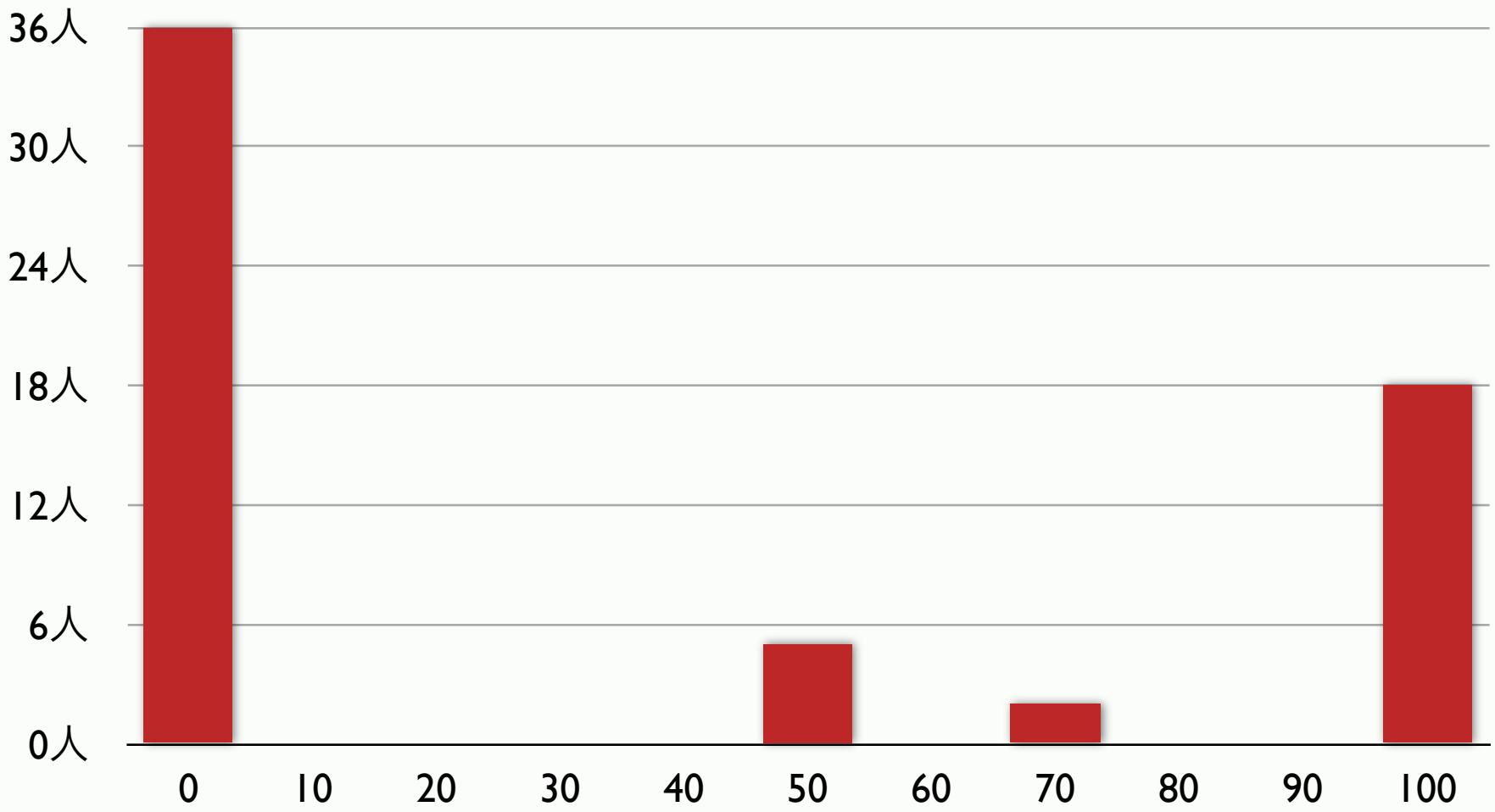
Take the 'IOI' train

- 鉄道事業を成功させる気がない
  - 問題文にはちゃんと書いてあります
  - 「列車が 1 つも編成できない場合は, 0 を出力せよ. 」



得点分布

Take the 'IOI' train



ちなみに

配るDP，貰うDP

Take the 'IOI' train

- 50点解法で使った方法を**配るDP**
- 100点解法で使った方法を**貰うDP**
  - と言うこともあります
  - この問題ではどっちの方が効率がよいということはなく，単に実装方法の違いです
  - 基本的には自分の書きやすい方でよいでしょう(もちろんどっちも書けたほうがいいです)

Further Reading

Take the 'IOI' train

- プログラミングコンテストでの動的計画法
  - http://www.slideshare.net/iwiwi/ss-3578511