
Merry Christmas!


Merry Christmas!
Merry Christmas!

# JOI2019春合宿 Day 4 - ふたつのケーキ

解説: 清水郁実

# 問題概要

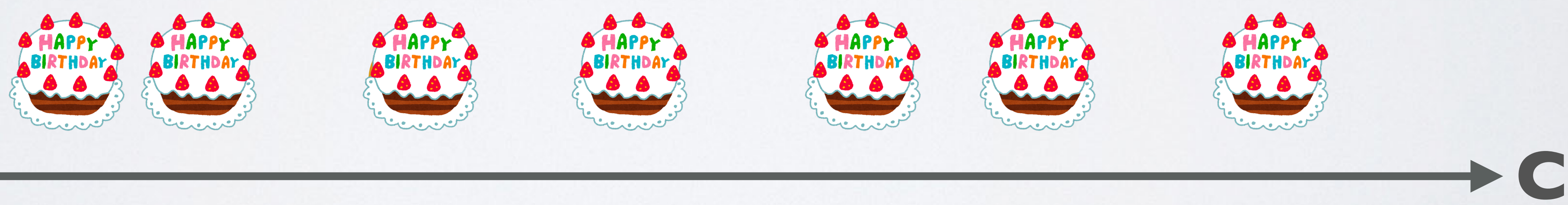
- 整数**N**, **M**とN個のケーキの情報(**Vi**, **Ci**)が与えられる
- このうちM個のケーキを選んで自由な順に円環状に並べたときの、**(選んだケーキのVの値の総和)** **–** **(隣り合う2つのケーキのCの値の差の絶対値の総和)**
  の最大値を答えよ
- 3≦M≦N≦200,000
  0≦Vi, Ci≦10^9

# 自明な考察

- M個のケーキを選び終えたときに、**(隣り合う2つのケーキのCの値の差の絶対値の総和)**を最小化するような並び替えを考える

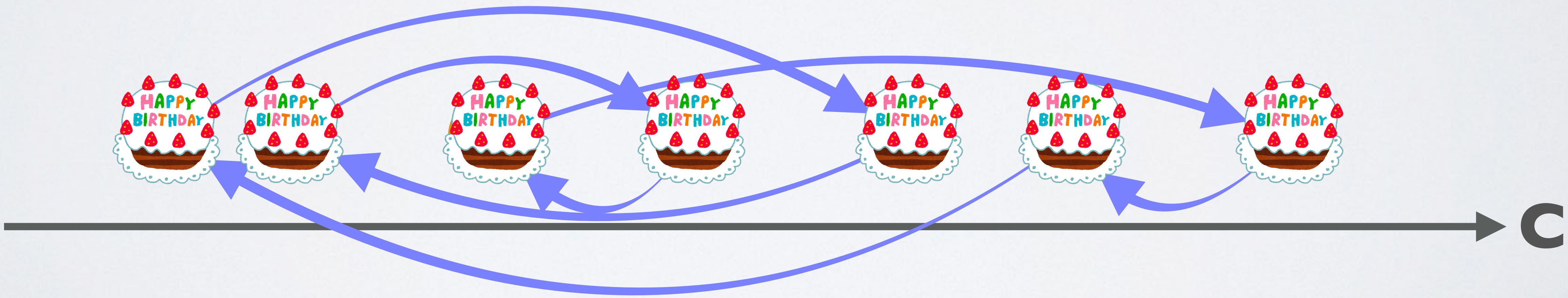
# 自明な考察

- M個のケーキを選び終えたときに、**(隣り合う2つのケーキのcの値の差の絶対値の総和)**を最小化するような並び替えを考える



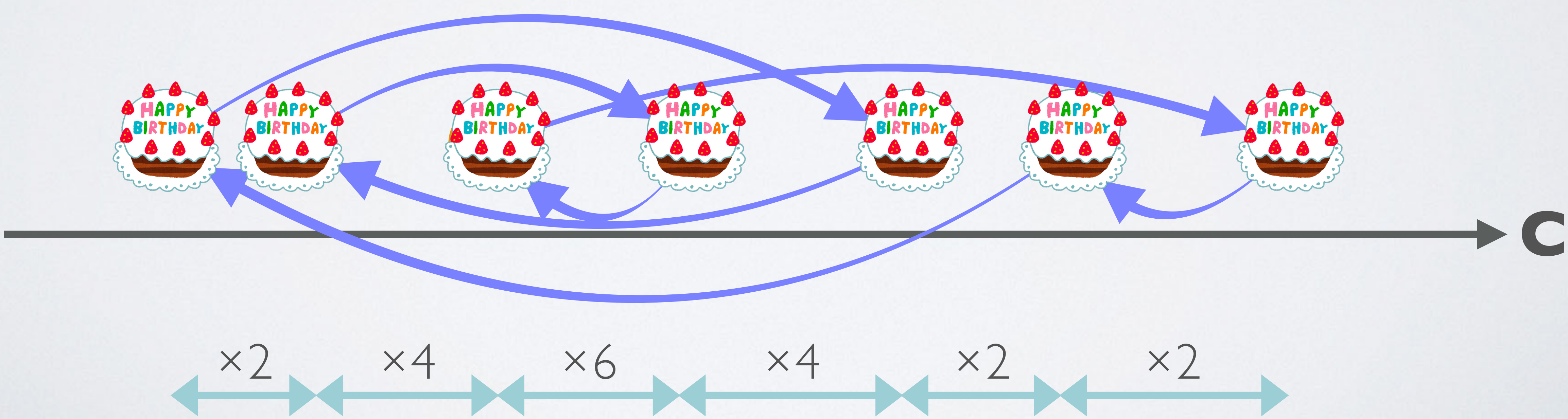
# 自明な考察

- M個のケーキを選び終えたときに、**(隣り合う2つのケーキのcの値の差の絶対値の総和)**を最小化するような並び替えを考える



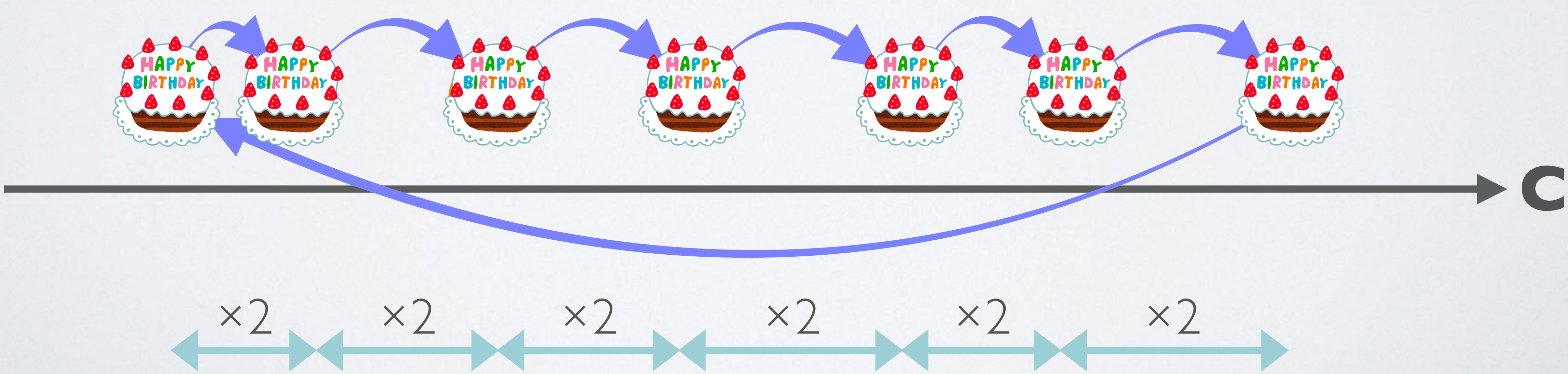
# 自明な考察

- M個のケーキを選び終えたときに、**(隣り合う2つのケーキのcの値の差の絶対値の総和)**を最小化するような並び替えを考える



# 自明な考察

- M個のケーキを選び終えたときに、**(隣り合う2つのケーキのcの値の差の絶対値の総和)**を最小化するような並び替えを考える



# 自明な考察

- Cが小さい(or 大きい)順に並べるのが最適で、

  このときの値は **2×(最大値−最小値)**

# 自明な考察

- Cが小さい(or 大きい)順に並べるのが最適で、
  このときの値は **2×(最大値−最小値)**
- Cの値を予めすべて2倍しておくと2倍の係数がなくなって楽

# 問題の言い換え

- 整数N, MとN個のケーキの情報(Vi, Ci)が与えられる
- このうちM個のケーキを選んだときの

  **(選んだVの値の総和) − (選んだCの最大値 − 最小値)**

  を最大化する

# 小課題1: N≦100 (5点)

- Cの最小値と最大値を決めて、その間に収まるケーキのうちVの値が大きい方からM個取る
- 実装方針によってO(N^3) or O(N^3 logN)

# 小課題2: N≦2,000 (24点)

- このままだと見通しが悪いので、ケーキを(Ci,Vi)の順にソートしておく
- すべての長さM以上の区間について、
  区間内部のVの値最大M個の和から(C[右端]−C[左端])を引いた値の最大値を求めたい

# 小課題2: N≦2,000 (24点)

- 区間の左端を固定して、右端を右方向に動かしていく
- (多重)集合に値を追加していって、毎回大きい方からM個の和がわかるようにしたい
- これは優先度付きキュー（priority_queue）等を使ってよしなに実装するとできて、全体O(N^2 logN)

# 小課題2: N≦2,000 (24点)

- 区間の左端を固定して、右端を右方向に動かしていく
- (多重)集合に値を追加していって、毎回大きい方からM個の和がわかるようにしたい
- これは優先度付きキュー（priority_queue）等を使ってよしなに実装するとできて、全体O(N^2 logN)
- ここまでは流石に解けてほしい

# ちなみに

- 一応dpをするとO(N^2)でできる
  - ここからのdp高速化による改善は多分うまくいかないor見通しが悪いので深入りしません

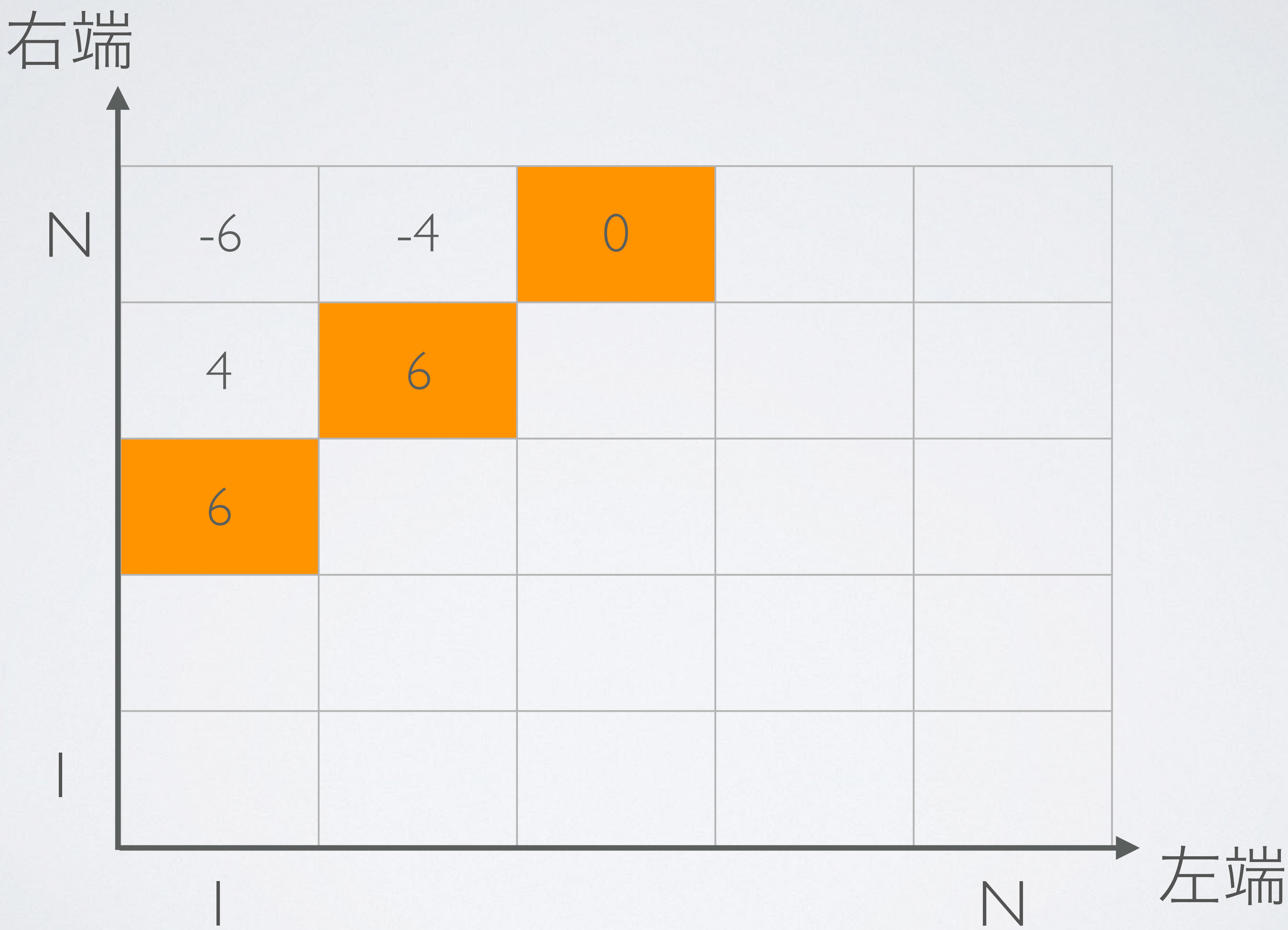
# 小課題3: N≦200,000 (100点)

- ここからが本番
- O(N^2)個ある区間を全部見ようと思うと大変なので、これをO(N)個ぐらいに減らしたい

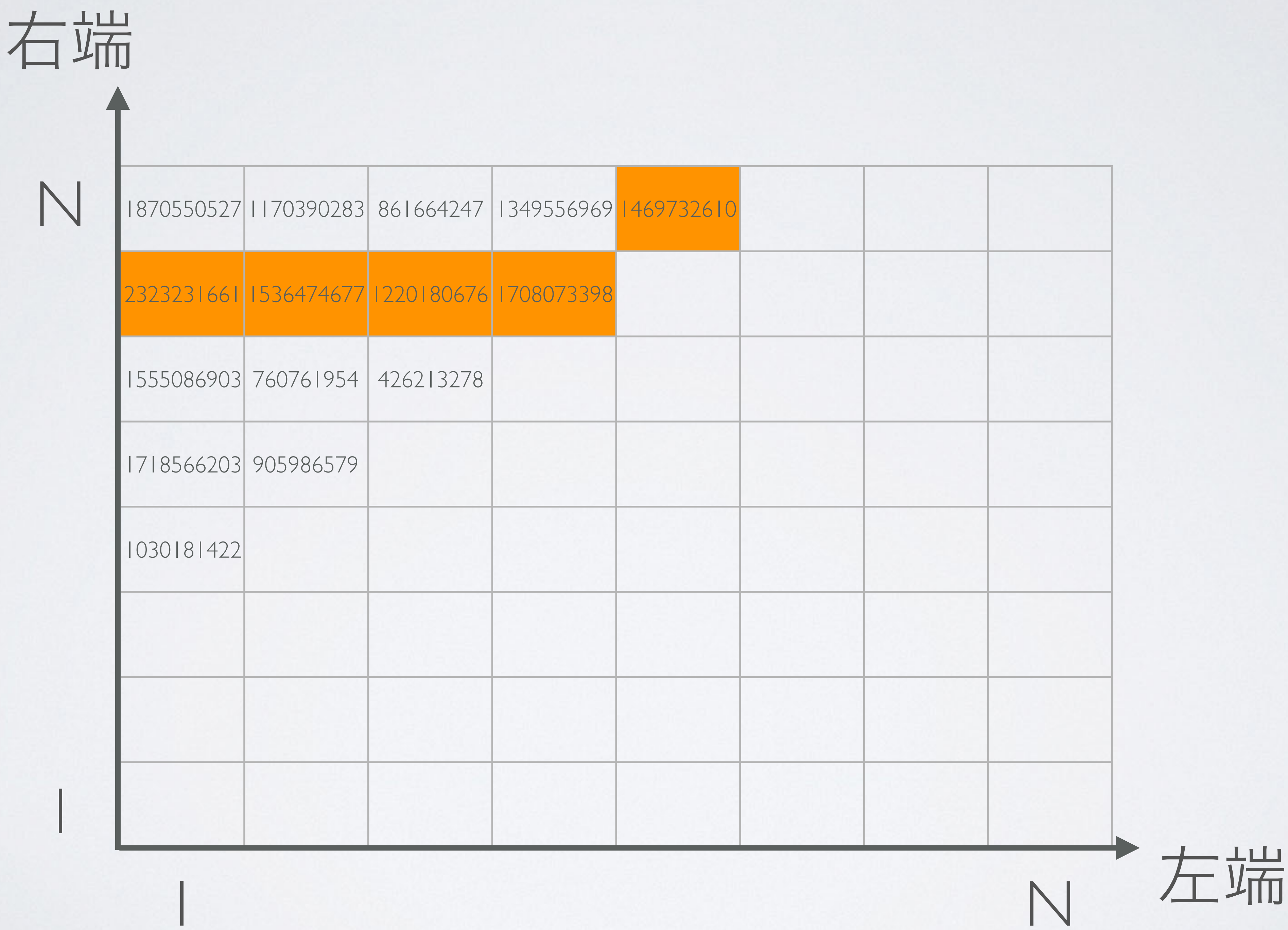
# 小課題3: N≦200,000 (100点)

- ここからが本番
- O(N^2)個ある区間を全部見ようと思うと大変なので、これをO(N)個ぐらいに減らしたい
- 具体的には、各左端に対して、最適値を取る右端の位置のみを調べられると望ましい

# 観察 - 入力例1



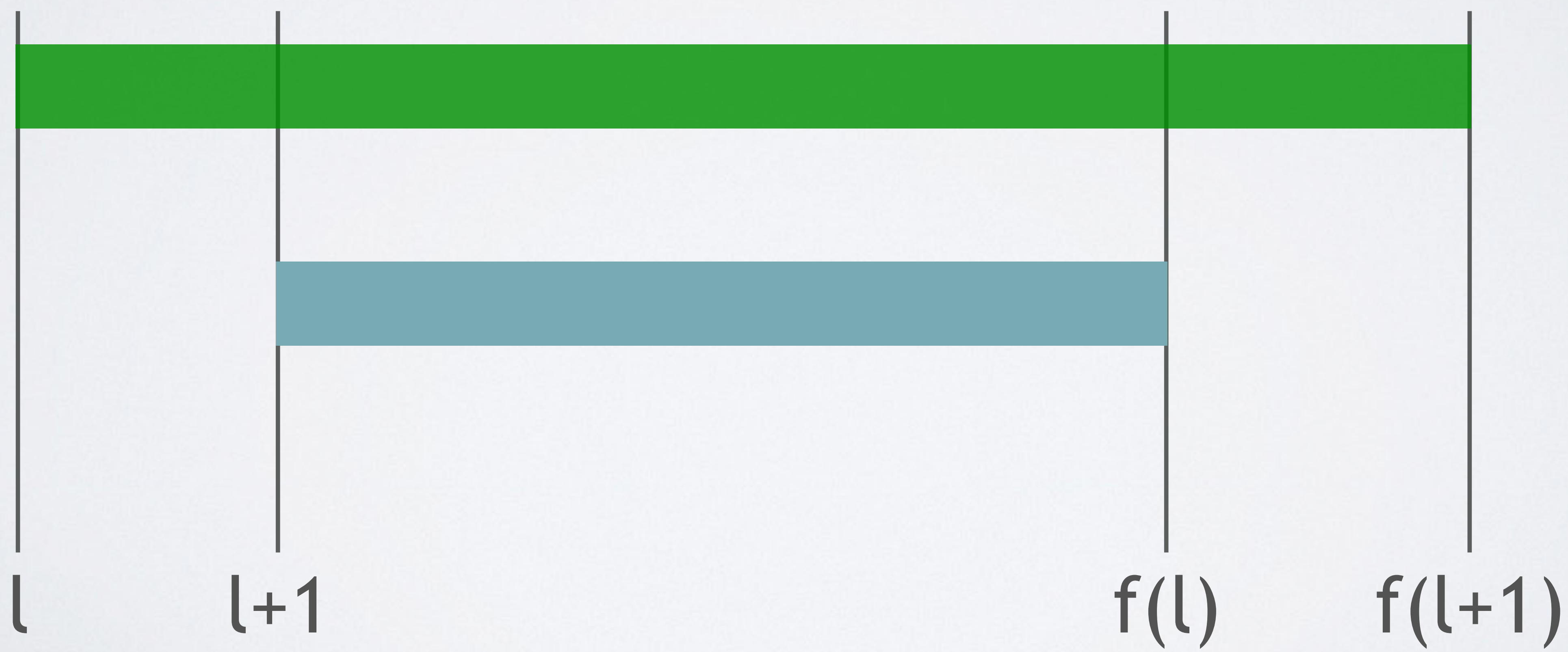
# 観察 - 入力例2


右端
N
1870550527
1170390283
861664247
1349556969
1469732610
2323231661
1536474677
1220180676
1708073398
1555086903
760761954
426213278
1718566203
905986579
1030181422
1
1
N
左端

# 觀察 - 自作入力

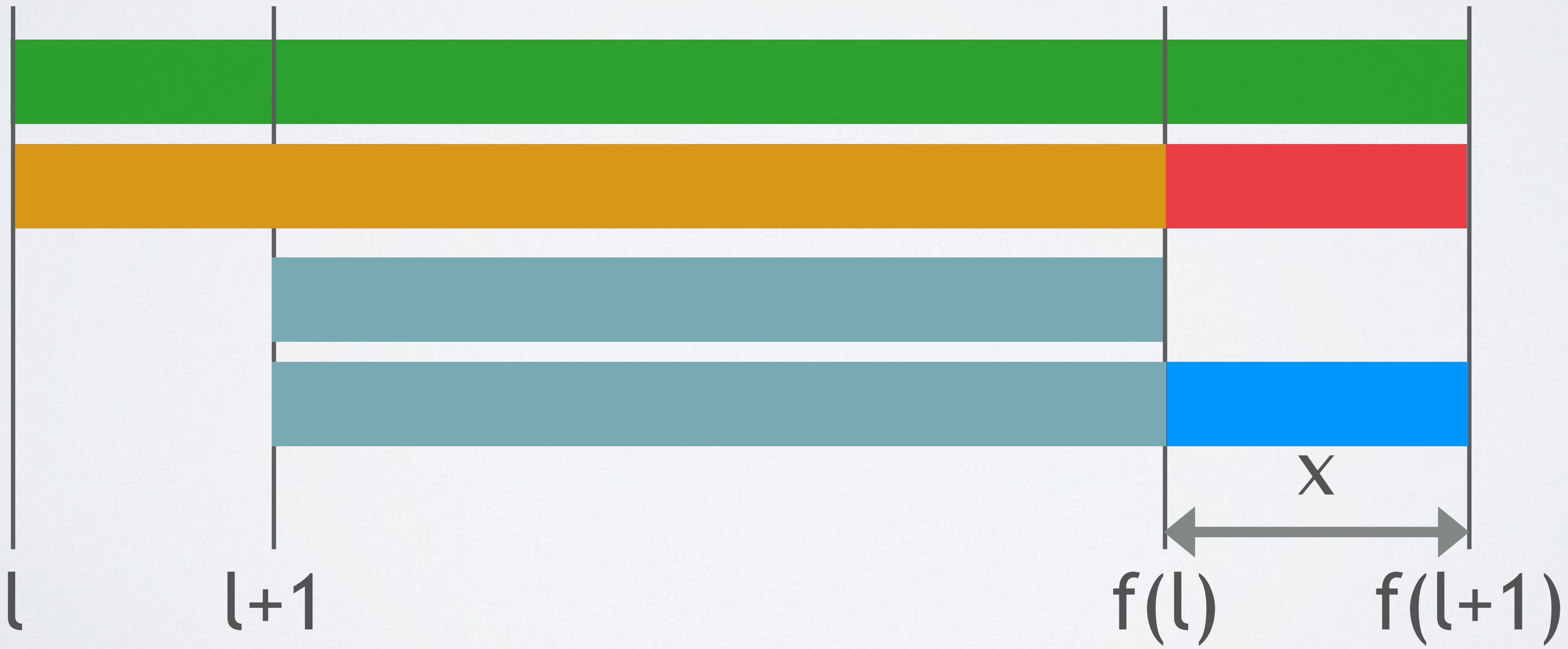
# 観察

- ↓が最適と仮定すると…



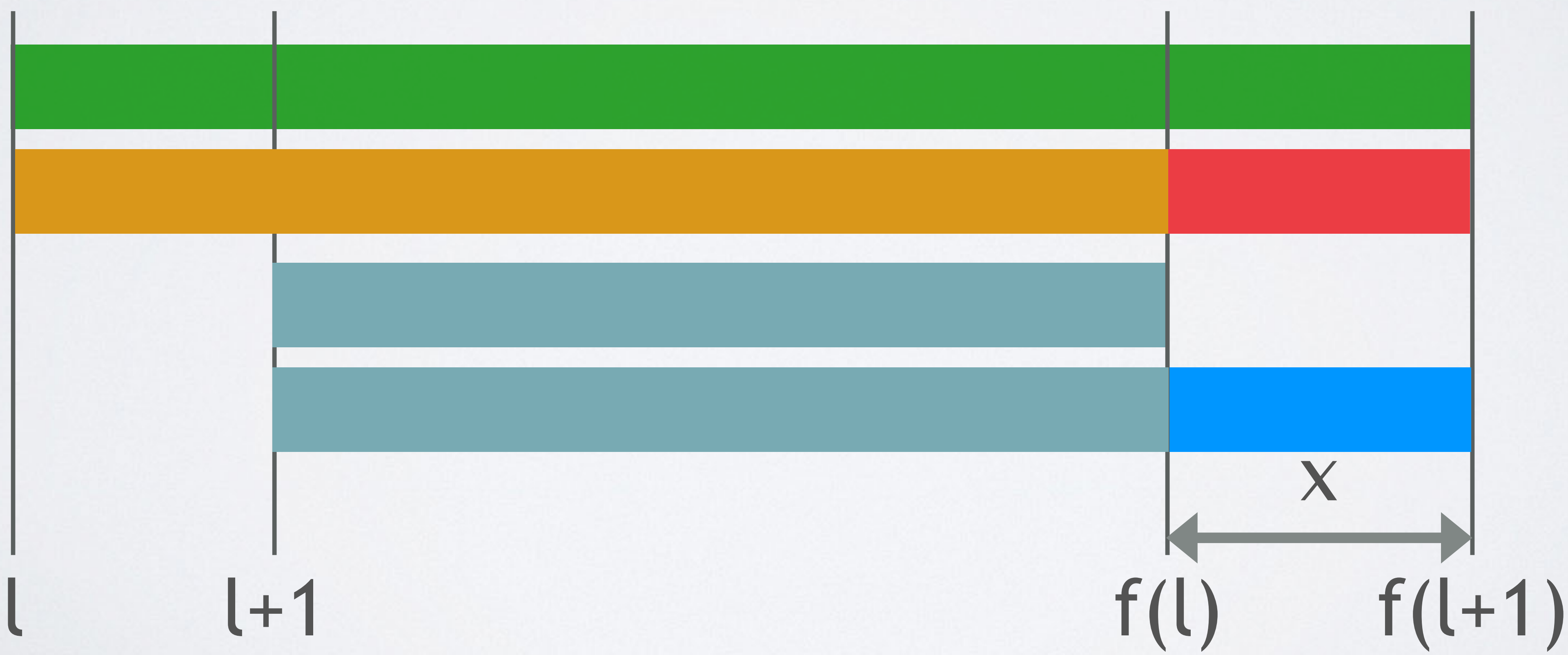
# 観察

- ↓が最適と仮定すると…



# 観察

- ↓が最適と仮定すると…
  - に を加えるとM個の最大和はx以上増える

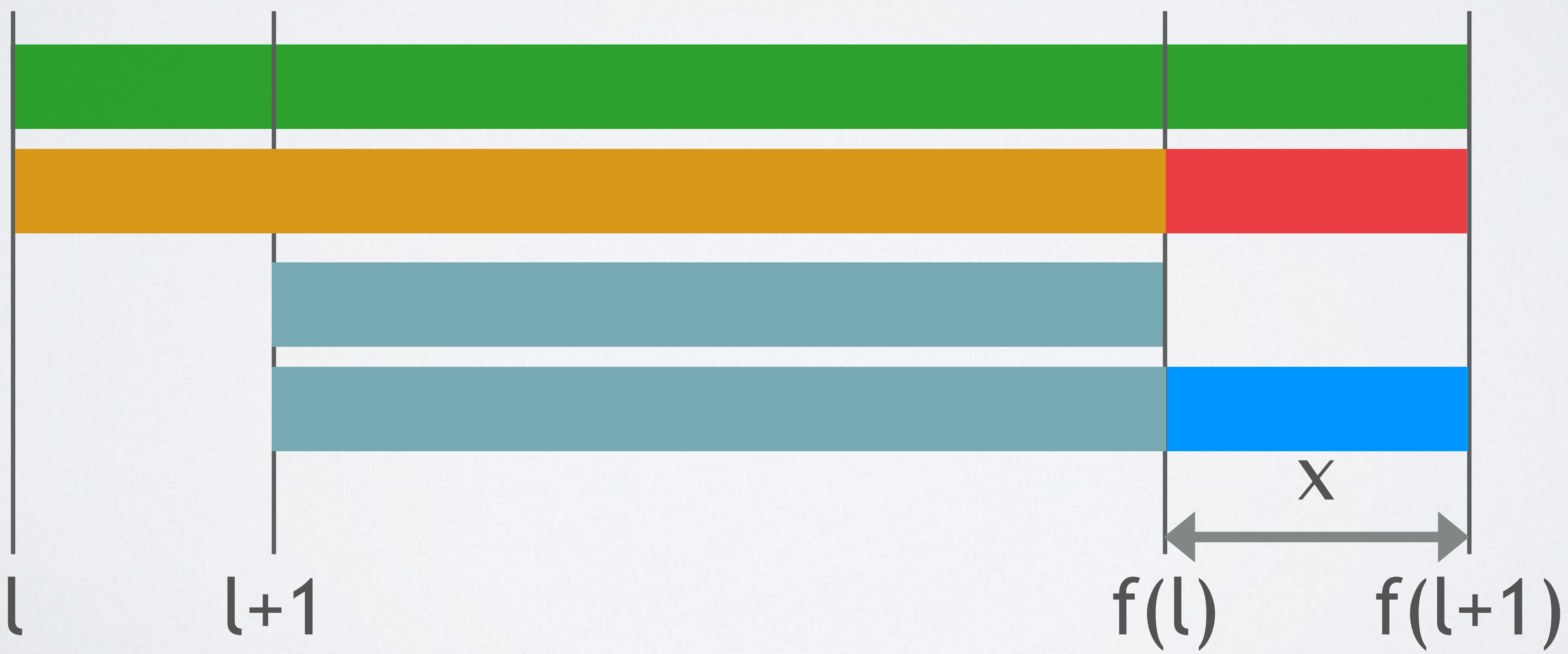


# 観察

- ↓が最適と仮定すると…

に を加えるとM個の最大和はx以上増える

⇒ に を加えても最大和はx以上増える ( ⊂ より)



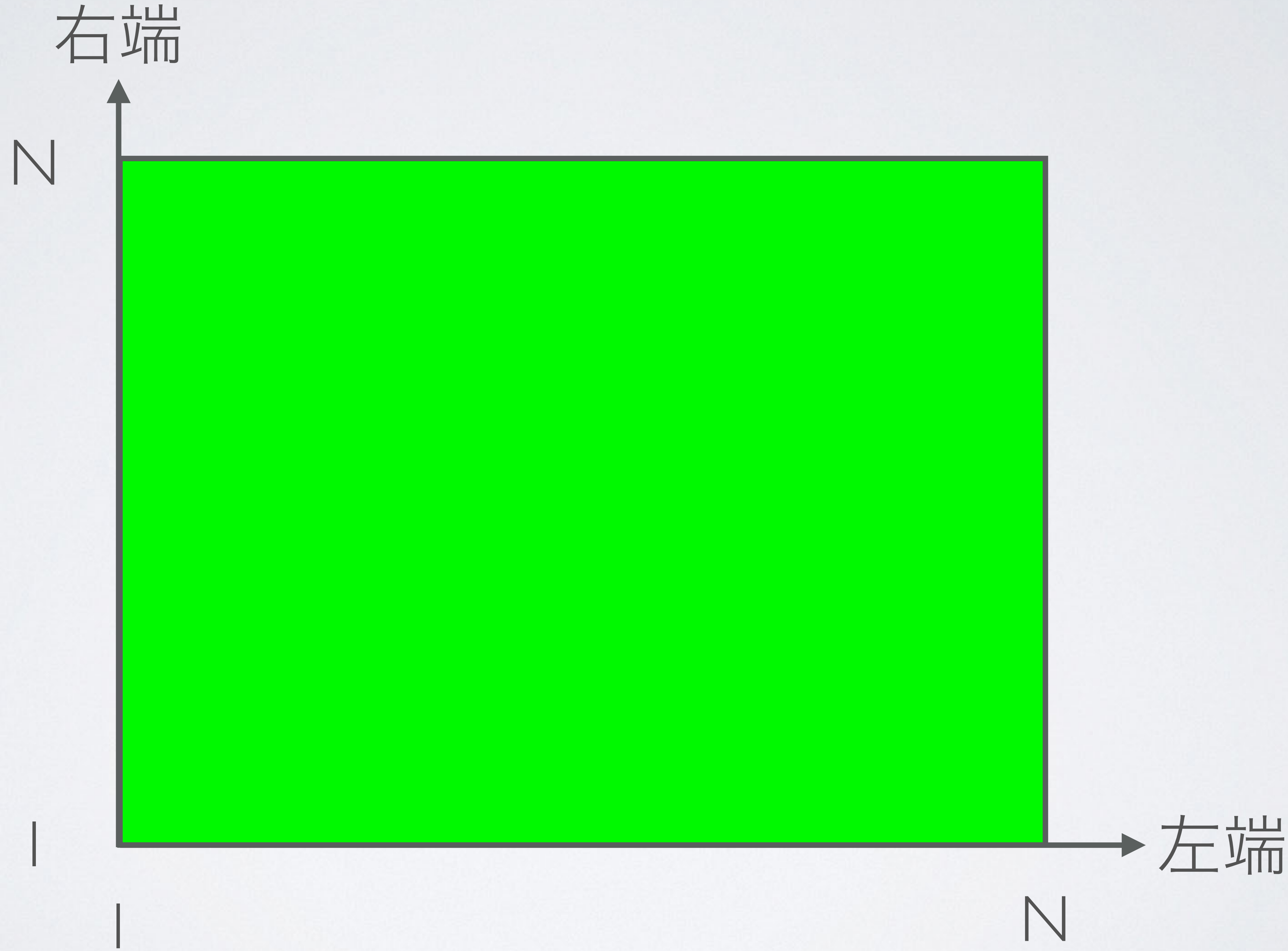
# 観察

- iを左端としたときの最適な選び方での右端の位置をf(i)とすると、**f(i)は広義単調増加**と考えてよい

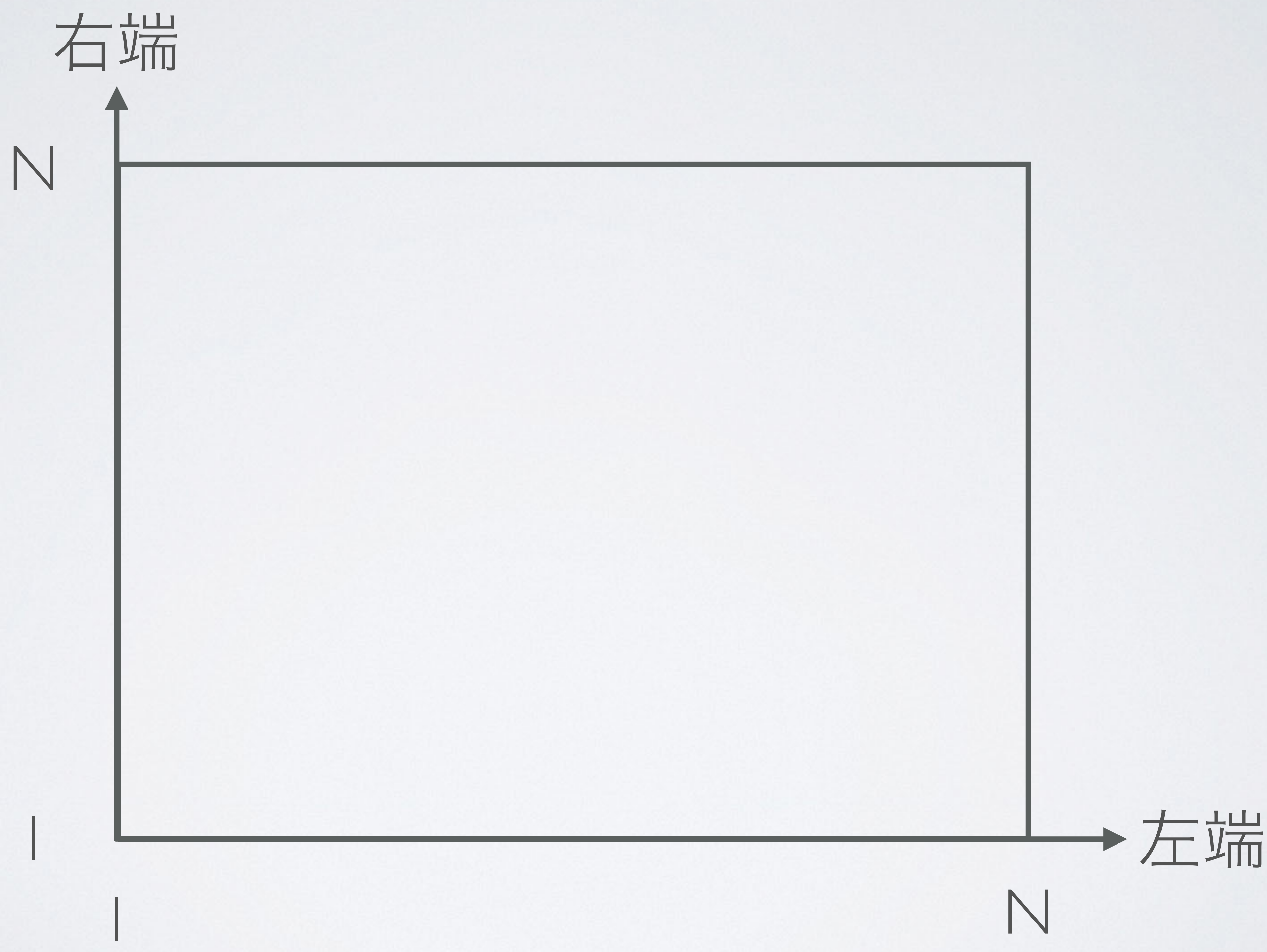
...?

# 分 割 統 治

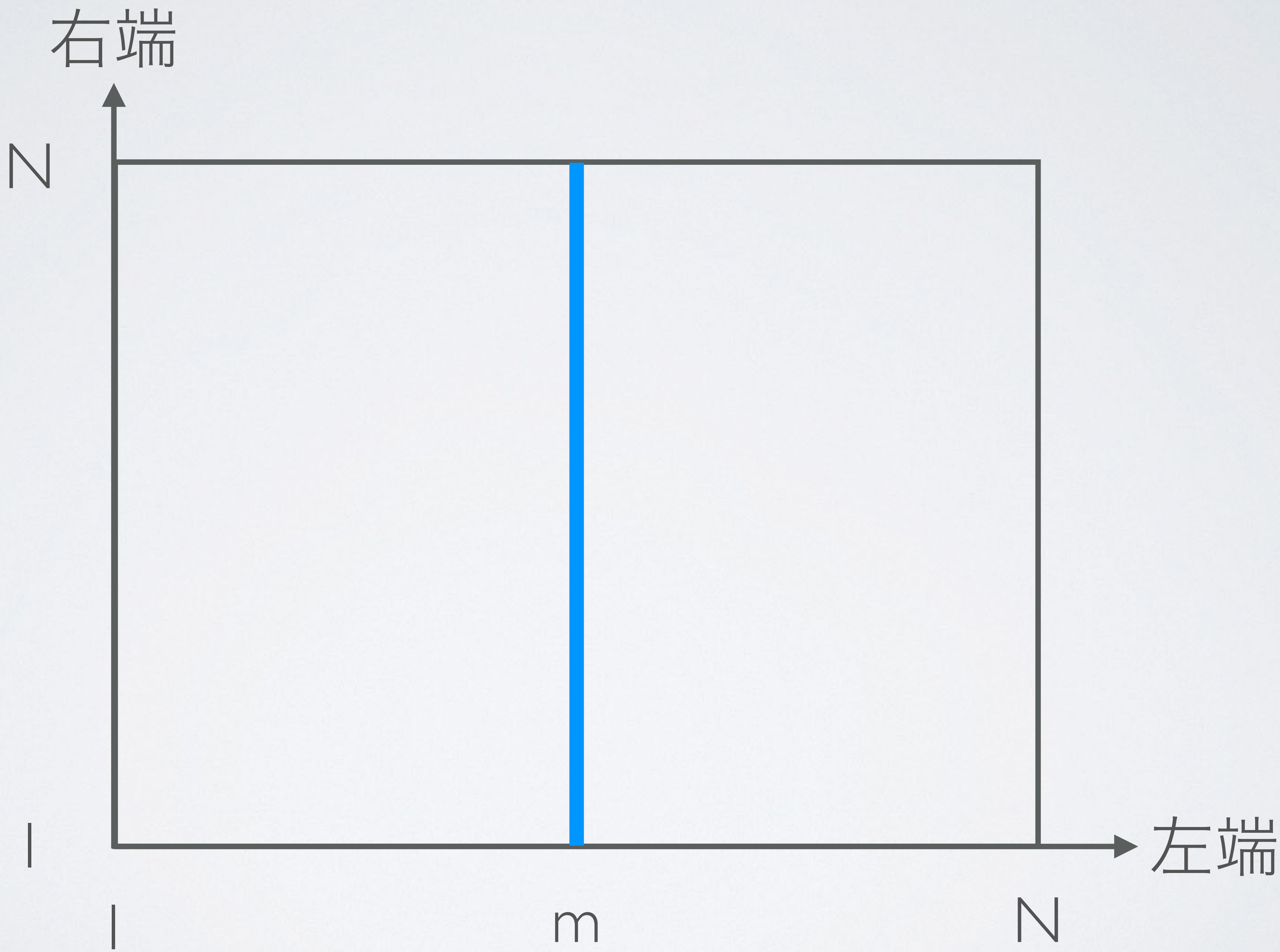
# 満点解法



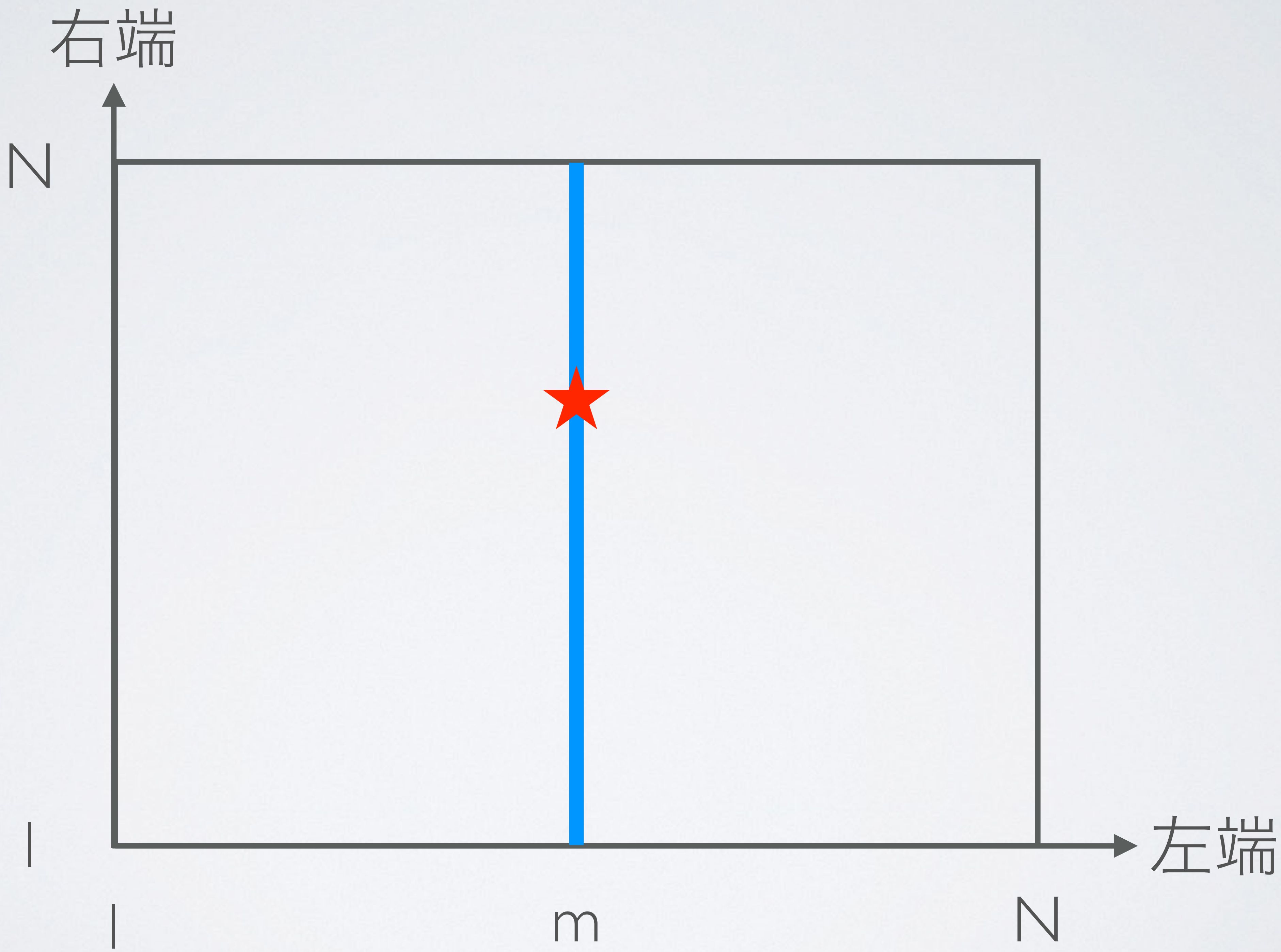
# 満点解法



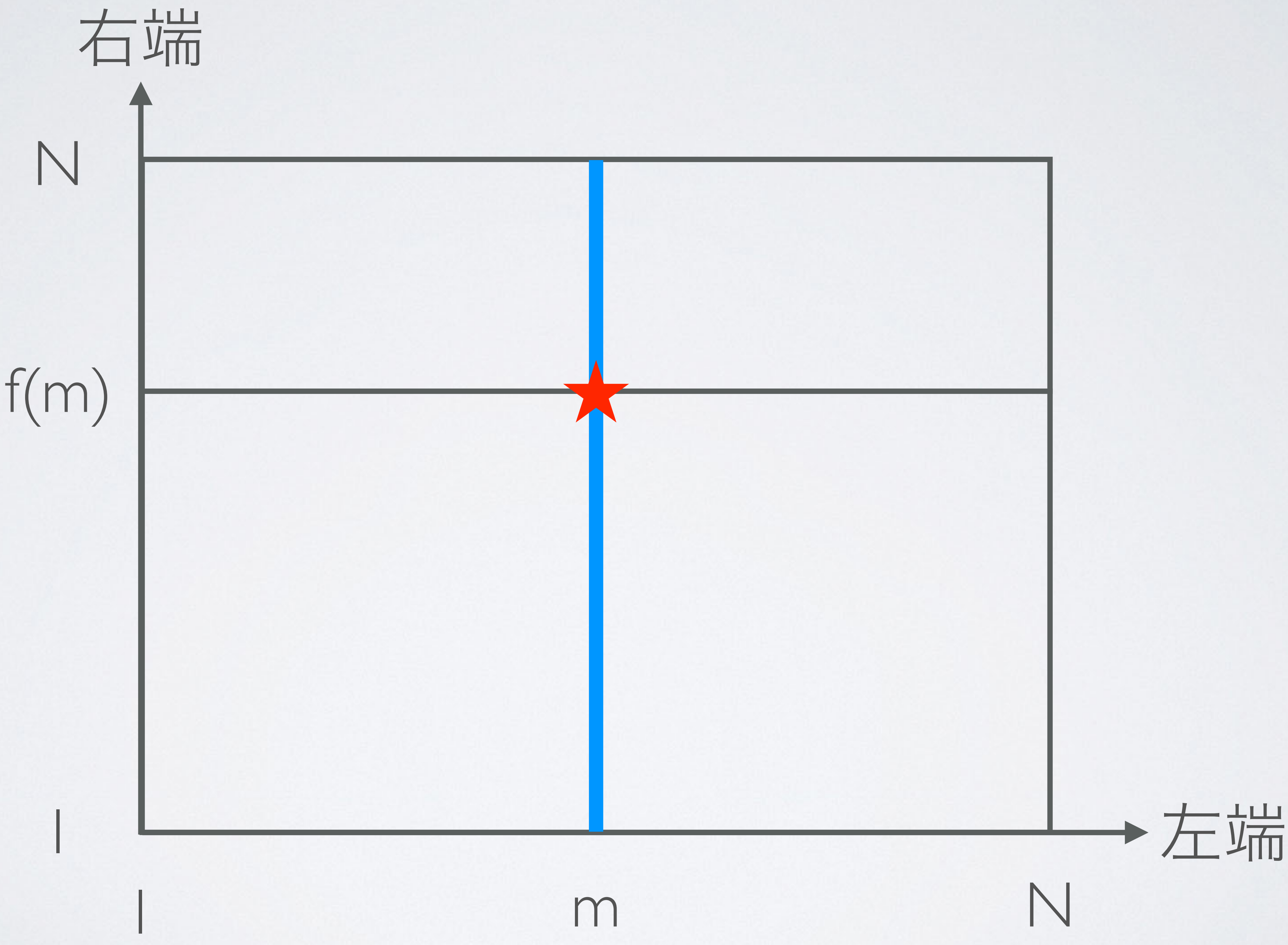
# 満点解法



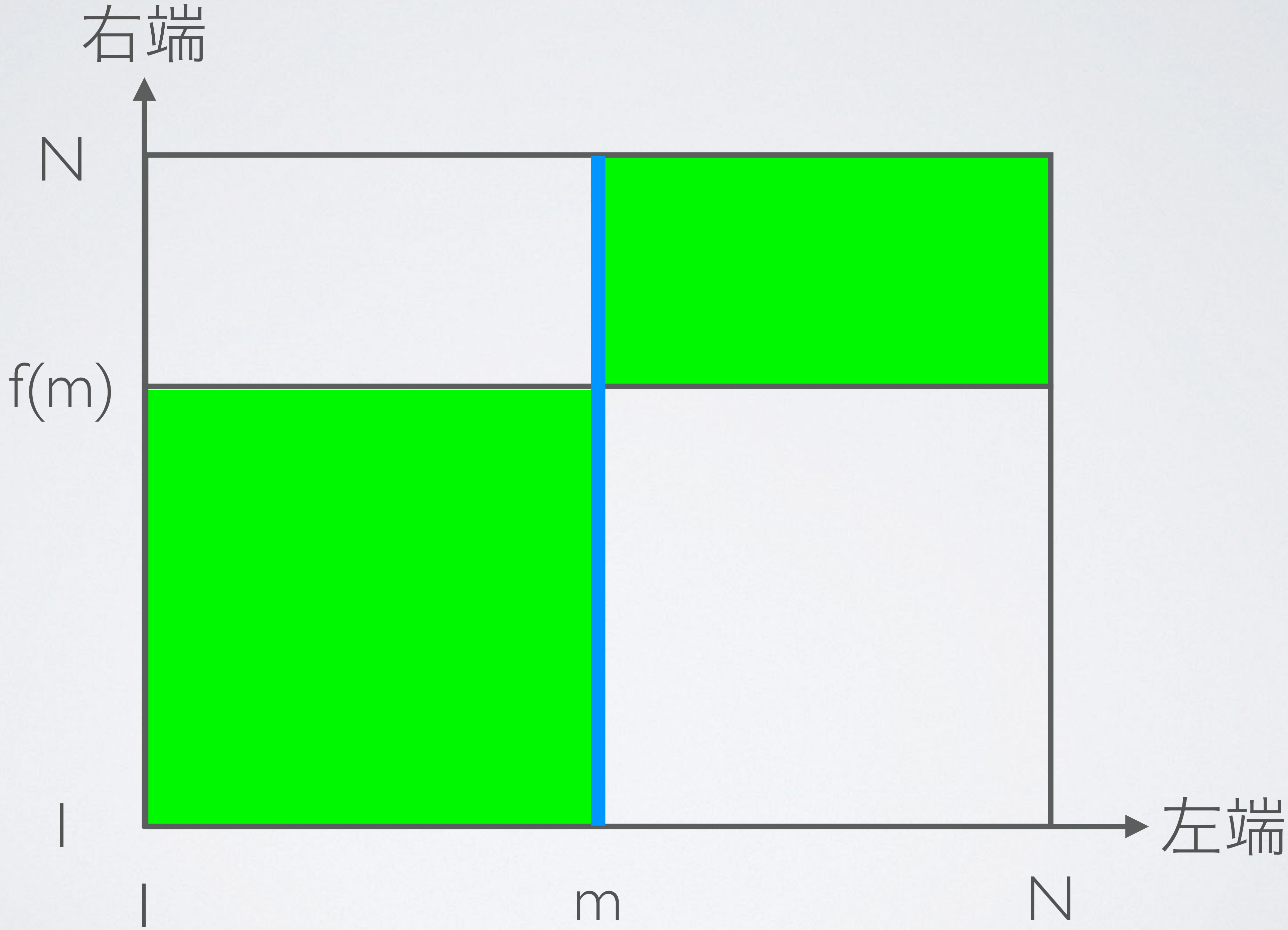
# 満点解法



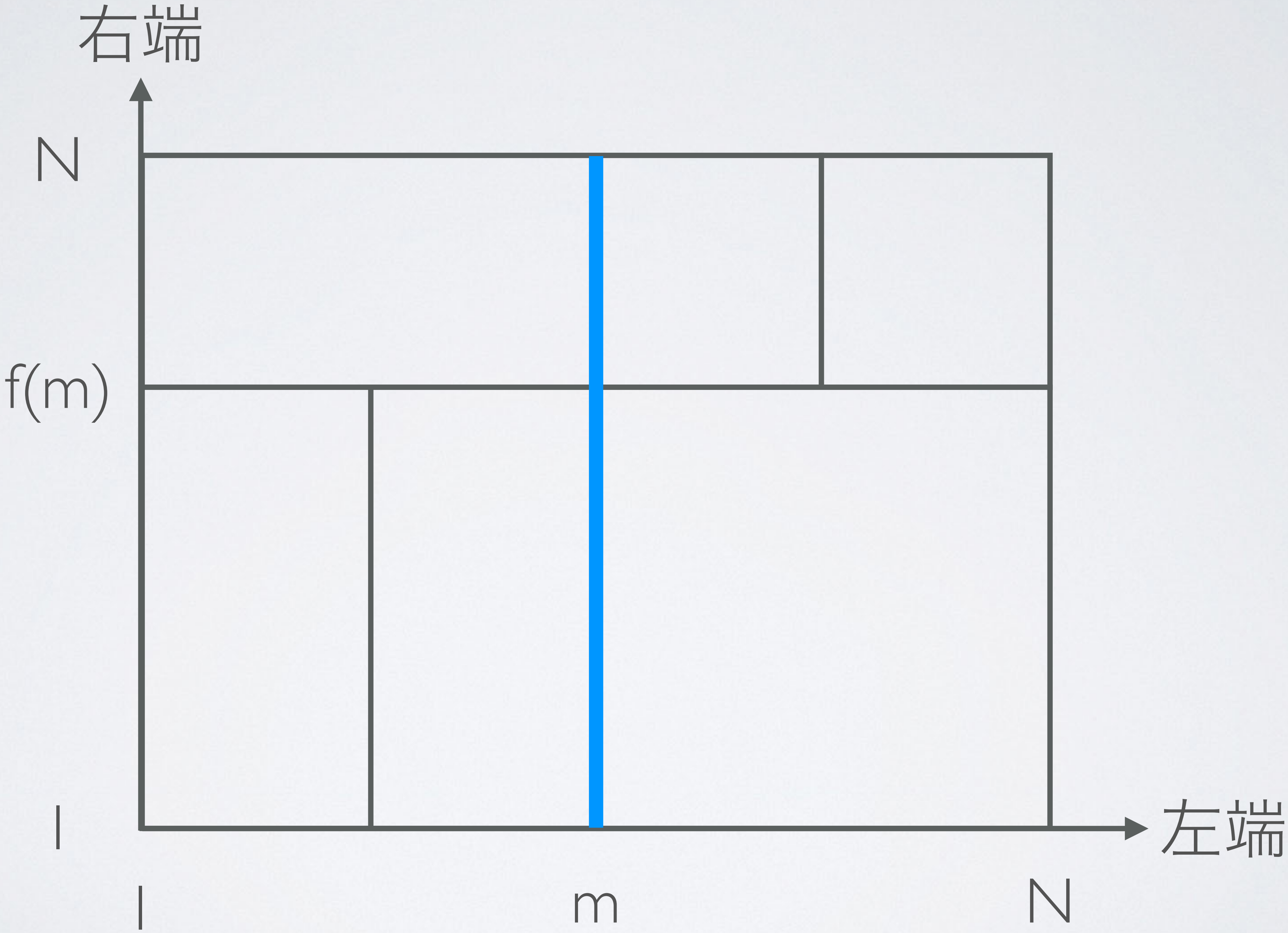
# 満点解法



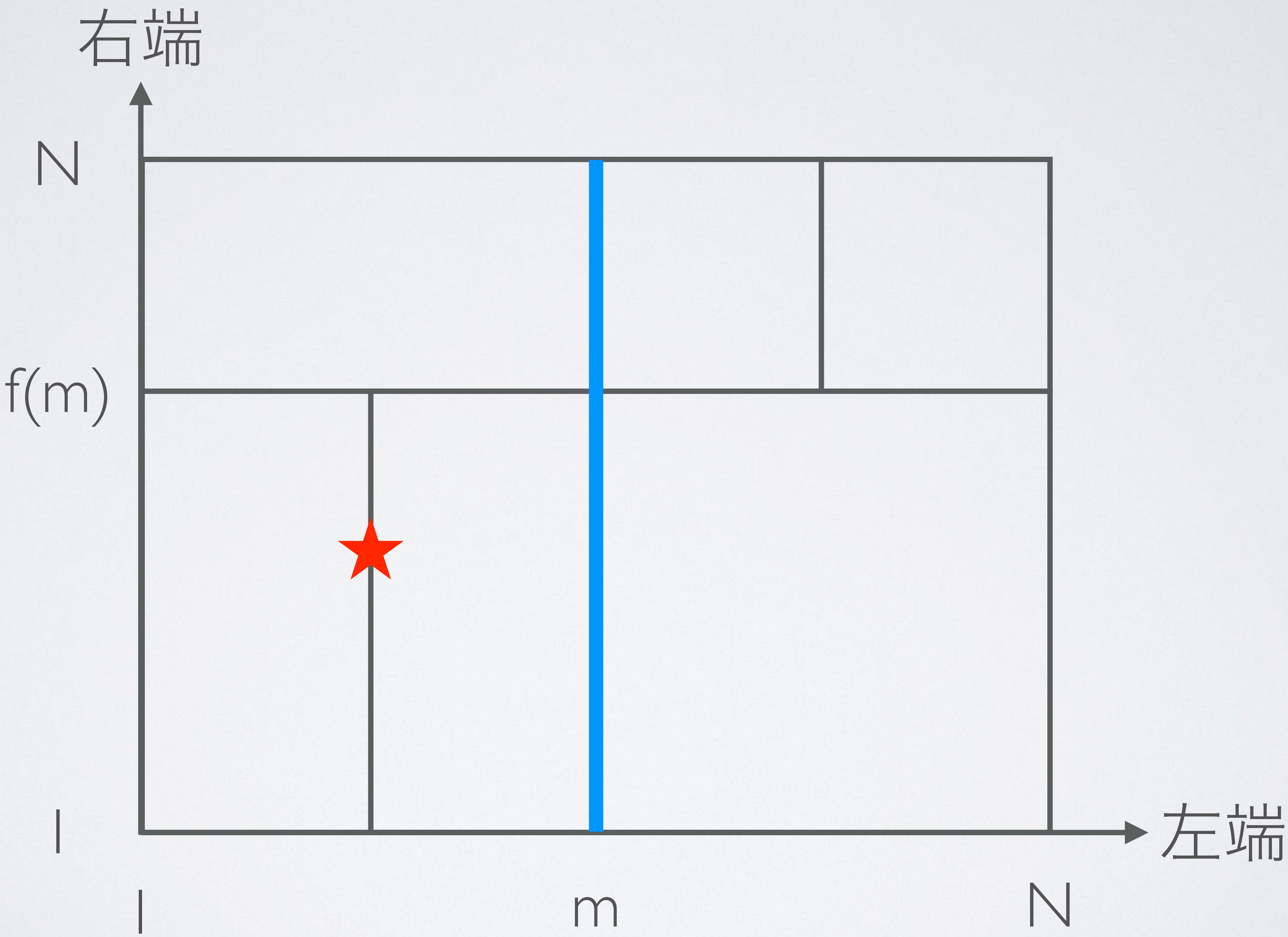
# 満点解法



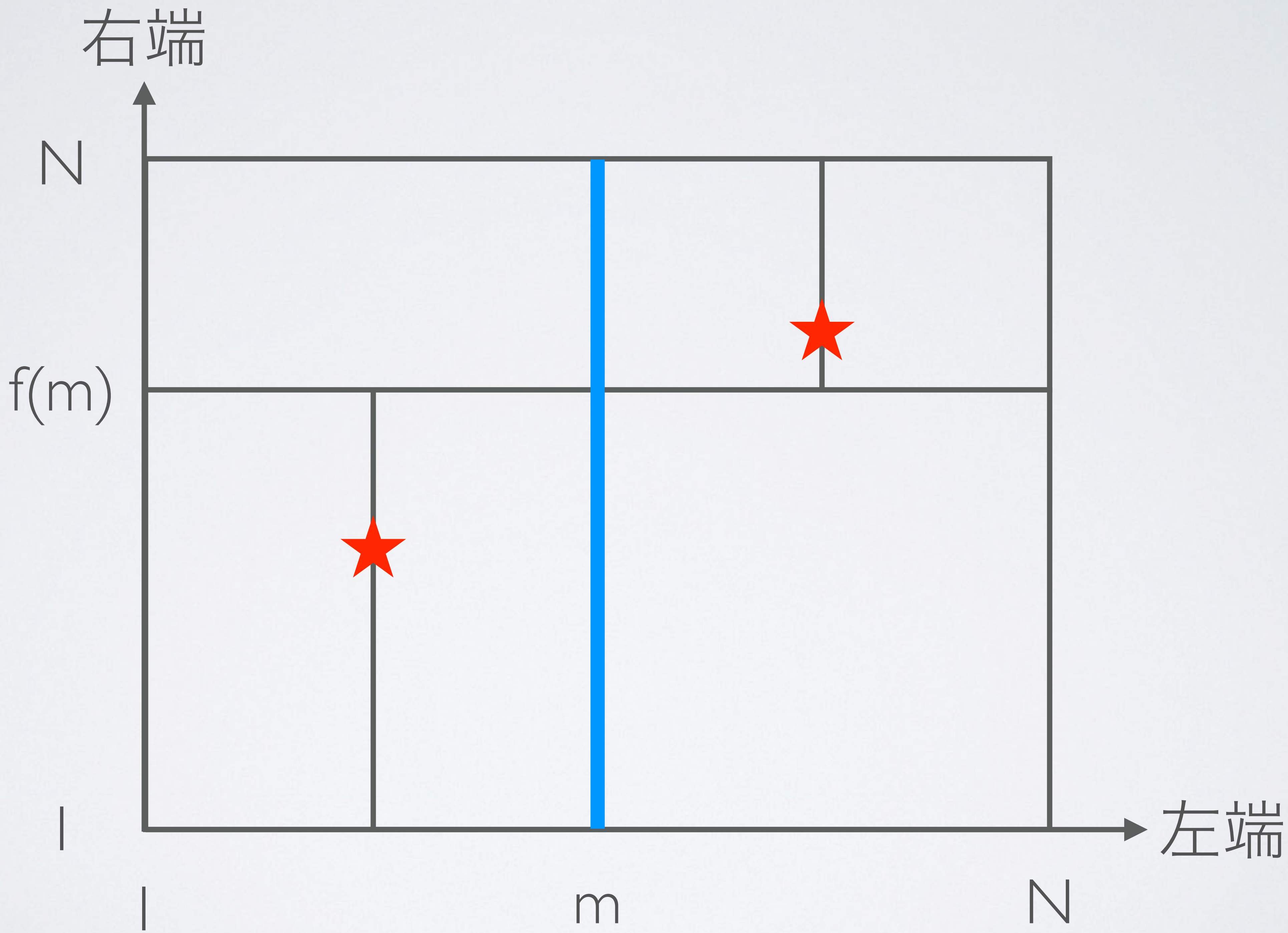
# 満点解法



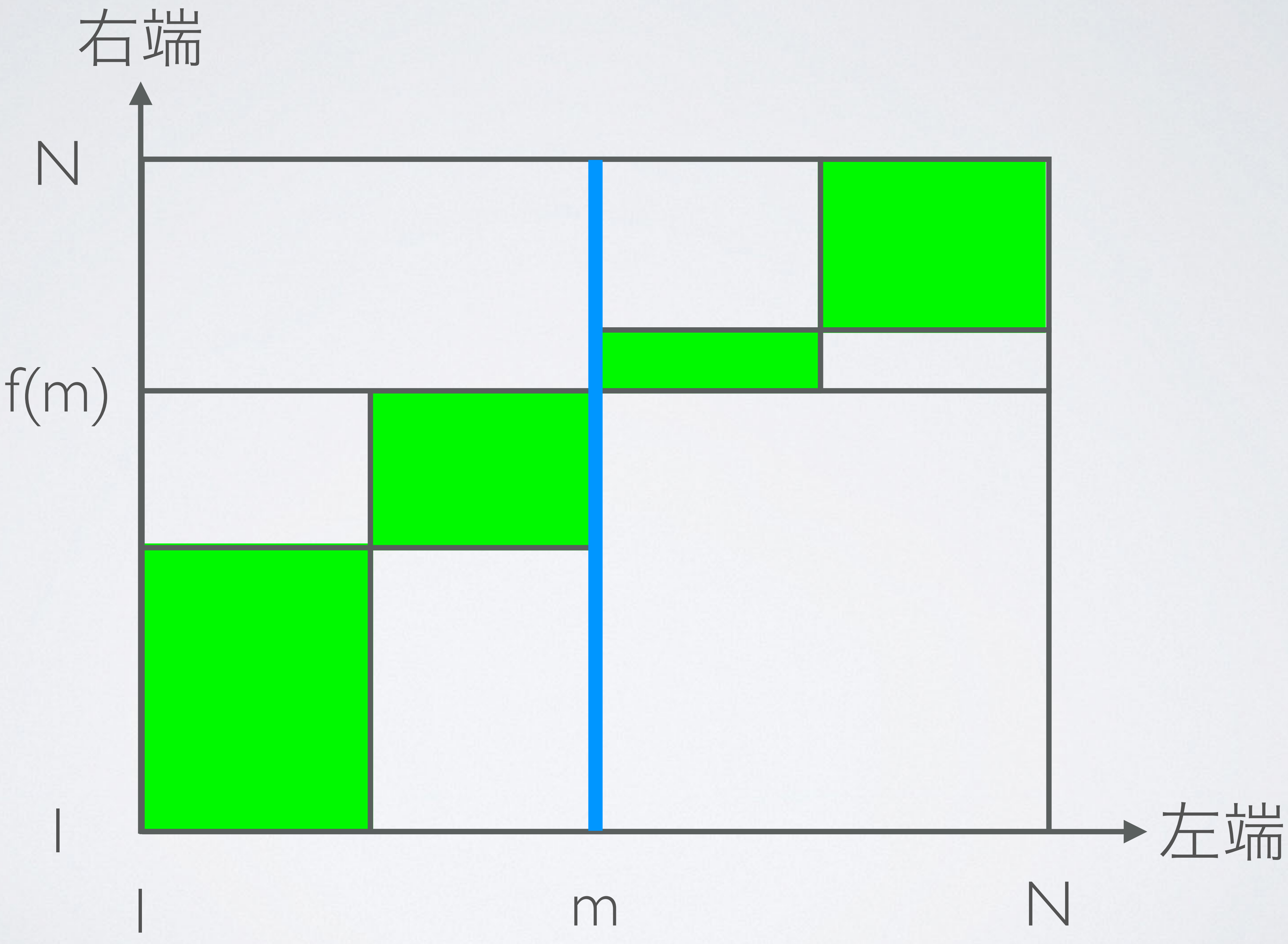
# 満点解法



# 満点解法



# 満点解法



# 満点解法

- mを左端として固定して、右端を全部試すことで最適な位置を見つける
- →左右の領域それぞれについて再帰的に同じ問題を解く

# 満点解法

- mを中央で取るようにすると、探索空間の面積は毎回1/2ずつに減っていくので、
- 左端と右端が決まっているときの答えがO(X)で求まるとき、この問題は全体O(NX logN)で解ける

# 満点解法

- mを中央で取るようにすると、探索空間の面積は毎回1/2ずつに減っていくので、
- 左端と右端が決まっているときの答えがO(X)で求まるとき、この問題は全体O(NX logN)で解ける
- monotone minimaというらしい

# 類題

- IOI 2014 Holiday
- 代表を目指す皆さんならもちろん2011以降のIOIは埋めていますよね…？

# 満点解法

- **[l, r]内の値の大きい方からM個の総和は？**

# 満点解法

- **[l, r]内の値の大きい方からM個の総和は？**
- → 完全にオンラインクエリとして処理するのは無理そう

# 満点解法

- **[l, r]内の値の大きい方からM個の総和は？**
- → 完全にオンラインクエリとして処理するのは無理そう
  - WeavletMatrix…? vectorを載せたセグ木…? 二分探索…?

# 満点解法

- **[l, r]内の値の大きい方からM個の総和は？**

- → 完全にオンラインクエリとして処理するのは無理そう

  - WeavletMatrix…? vectorを載せたセグ木…? 二分探索…?

    - ➡ 全体が間に合うためには各クエリO(logN)近くで答えられないといけないが、見るからに不可能

# 満点解法

- **[l, r]内の値の大きい方からM個の総和は？**

- → 完全にオンラインクエリとして処理するのは無理そう

  - WeavletMatrix…? vectorを載せたセグ木…? 二分探索…?

    ➡ 全体が間に合うためには各クエリO(logN)近くで答えられないといけないが、見るからに不可能

- 分割統治で必要になる値の[l, r]の順序をうまいこと利用して効率よく計算したい

# 満点解法

- 今使える道具は？
- 小課題2のアプローチを思い出す

# 満点解法

- 今使える道具は？
- 小課題2のアプローチを思い出す
  ➡ 区間[i, i]からスタートして右端を伸ばしていくことなら O(logN)でできる

# 満点解法

- 今使える道具は？
- 小課題2のアプローチを思い出す
  - ➡ 区間[i, i]からスタートして右端を伸ばしていくことなら O(logN)でできる
  - ➡ 左端を縮めることも…？

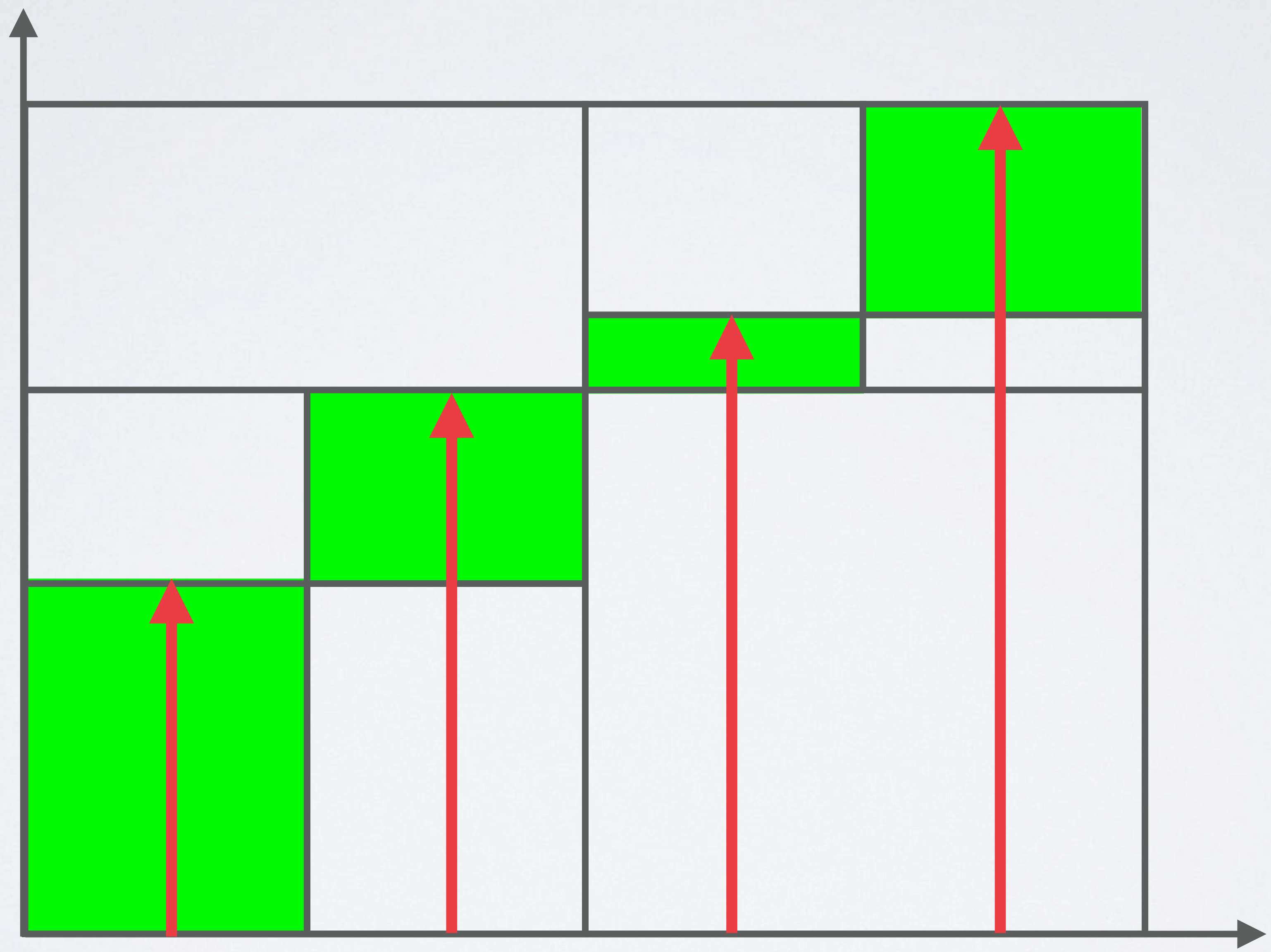
# 満点解法

- 今使える道具は？
- 小課題2のアプローチを思い出す

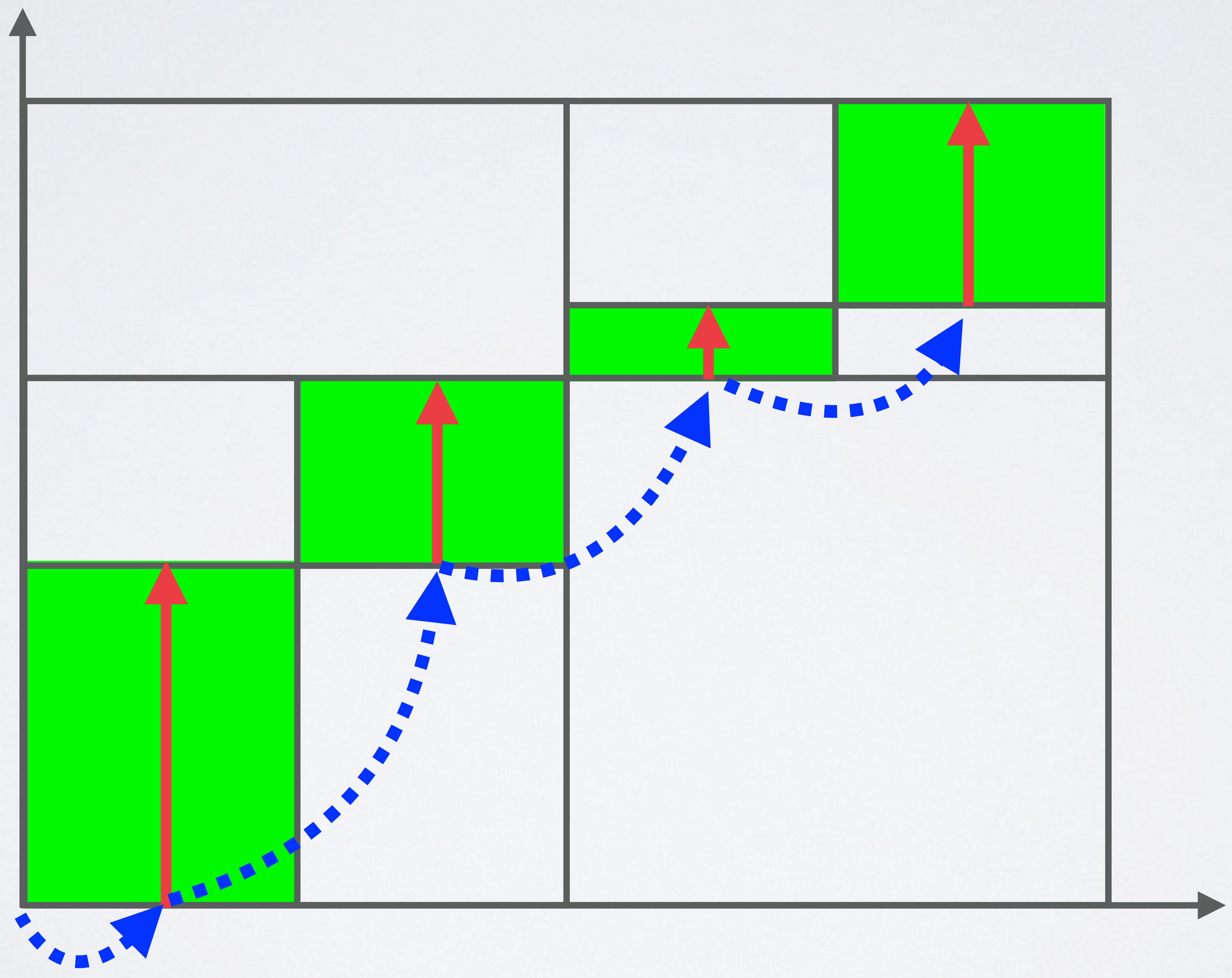
➡ 区間[i, i]からスタートして右端を伸ばしていくことなら O(logN)でできる

➡ 左端を縮めることも…？ **実はできる（後述）**

↑の長さのオーダーをかけてしまうとダメ



で右端を伸ばして で左端を縮める
長さの合計はO(N)



# 満点解法

- 区間[0, 0]からスタートして、右端を伸ばしたり左端を縮めたりしながら、区間内部の最大M個の和を管理しつつ、分割統治で必要な値を計算していく
- 一通り計算が終わったら、計算結果をもとに各探索領域をさらに半分に分割して同じ作業を繰り返す

# 満点解法

- 毎回横幅が1/2ずつに減っていくので、イテレーションの回数はlogN回
- 毎回のイテレーションで見なければいけない区間たちが左端,右端ともに単調増加になっていることが嬉しい性質
- parallel binary searchにかなり近いbfsをしたが、再帰でも同じ感じでできる

# 実装

## 満点解法

- 今使える道具は？
- 小課題2のアプローチを思い出す
  - ➡ 区間[i, i]からスタートして右端を伸ばしていくことならO(logN)でできる
  - ➡ 左端を縮めることも…？ **実はできる（後述）**

# 整理すると…

- 値Ciを（多重）集合に加える
- 値Ciを（多重）集合から削除する
- 追加されて残っている値のうち最大M個の総和を求める

- の3種類のクエリを処理できるデータ構造があればよい

# 平衡二分木解

- 値を追加/削除するたびに現在のM番目付近の値が特定できればよい
- std::setではk番目の値を求める操作がサポートされていないので、g++拡張のtreeとかいうやつを使う
  （参考: http://hogloid.hatenablog.com/entry/2014/09/23/132440）
- または、平衡二分木を自前で書く

# 平衡二分木解

- g++拡張は覚えないといけないおまじないの量が多いので、検索・ライブラリ持ち込み不可のJOIでは少し厳しい

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/tag_and_trait.hpp>

using namespace __gnu_pbds;

tree<T, null_type, less<T>, rb_tree_tag, tree_order_statistics_node_update> …
```

# 平衡二分木解

- g++拡張は覚えないといけないおまじないの量が多いので、検索・ライブラリ持ち込み不可のJOIでは少し厳しい

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/tag_and_trait.hpp>

using namespace __gnu_pbds;

tree<T, null_type, less<T>, rb_tree_tag, tree_order_statistics_node_update> …
```

- どのみち定数倍が怪しいのでおすすめしません

# セグ木解

- 一点更新・区間和の一般的なセグメント木を2本用いる
- Viの順に並べた長さNの数列を考えて、2本のセグメント木の各節点には、それぞれ区間内の生きている個数とVの総和を持たせる
- あとはセグメント木上の二分探索で右から個数の和がMになる位置を求めて、そこから末尾までのVの総和を普通に取ればよい
- 各クエリO(logN)で処理できる

# まとめ

- logN回のイテレーションで、毎回セグ木初期化→走査しつつ探索する区間を分割統治していけばOK
- セグ木への追加/削除は1回のイテレーションにつき各位置でちょうど1回ずつ行われるのでこれでO(N log^2N)

# まとめ

- logN回のイテレーションで、毎回セグ木初期化→走査しつつ探索する区間を分割統治していけばOK
- セグ木への追加/削除は1回のイテレーションにつき各位置でちょうど1回ずつ行われるのでこれでO(N log^2N)
- 実装はコードを書くとできます

# 得点分布

# 得点分布

# 得点分布

# 得点分布