2014年 JOI 本選
問題 1
JOI 紋章

解説担当: 城下 慎也(@phidnight)

最初に

- 本選お疲れ様でした！
- 今から本選の解説が行われます。

問題概要

- 2×2サイズの JOI 紋章が指定されるので、1 箇所の書き換えを必要ならば実行して、JOI旗にできるだけ多くの JOI 紋章を含むようにしたときの JOI 紋章の個数の最大値を求める。
- 制約
  - $2 \leqq N \leqq 1{,}000$
  - $2 \leqq M \leqq 1{,}000$

具体例

- JOI 旗と JOI 紋章の例

JOI 旗

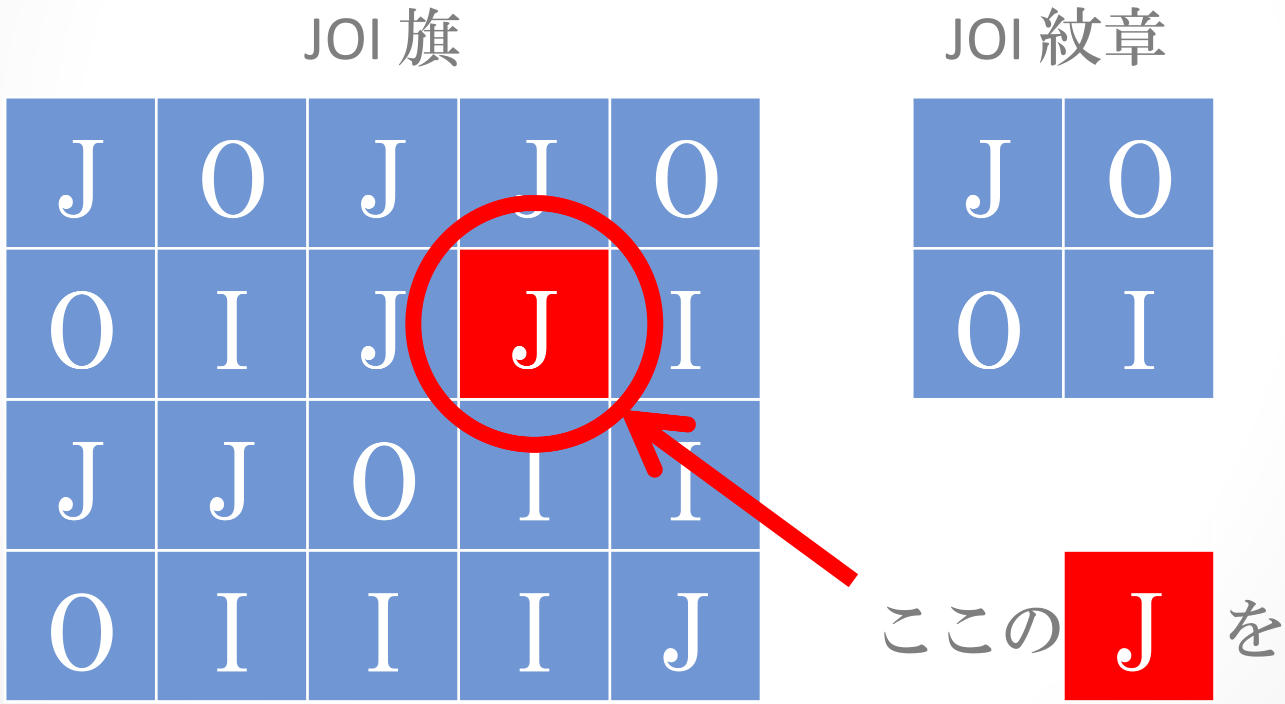
J	O	J	J	O
O	I	J	J	I
J	J	O	I	I
O	I	I	I	J

JOI 紋章

J	O
O	I

具体例

- JOI 旗と JOI 紋章の例



具体例

- JOI 旗と JOI 紋章の例

JOI 旗

J	O	J	J	O
O	I	J	O	I
J	J	O	I	I
O	I	I	I	J

JOI 紋章

J	O
O	I

O にする

具体例

- JOI 旗と JOI 紋章の例
- １つ目

JOI 旗

J	O	J	J	O
O	I	J	O	I
J	J	O	I	I
O	I	I	I	J

JOI 紋章

J	O
O	I

具体例

- JOI 旗と JOI 紋章の例
- １つ目　２つ目

JOI 旗

J	O	J	J	O
O	I	J	O	I
J	J	O	I	I
O	I	I	I	J

JOI 紋章

J	O
O	I

具体例

- JOI 旗と JOI 紋章の例
- 1つ目　2つ目　3つ目

JOI 旗

J	O	J	J	O
O	I	J	O	I
J	J	O	I	I
O	I	I	I	J

JOI 紋章

J	O
O	I

合計 3 個が最大となる。

文字列の問題

- 本問題は文字列を用いた問題である。

文字列の問題

- 本問題は文字列を用いた問題である。
- 文字列の操作には、例えば C++ なら char 配列や string がある。(他の言語にもいろいろある)

文字列の問題

- 本問題は文字列を用いた問題である。
- 文字列の操作には、例えば C++ なら char 配列や string がある。(他の言語にもいろいろある)
- 文字列はエンバグしやすい筆頭
  - バグの原因が文字列操作であるとかは悲しい
  - 貴重なコンテスト時間を失ってしまう

文字列の問題

- 本問題は文字列を用いた問題である。
- 文字列の操作には、例えば C++ なら char 配列や string がある。(他の言語にもいろいろある)
- 文字列はエンバグしやすい筆頭
  - バグの原因が文字列操作であるとかは悲しい
  - 貴重なコンテスト時間を失ってしまう
- 慣れない操作はするべきではない（重要）
  - 事前に構造を把握したもの以外の使用は要注意

文字列の問題

- 本問題は文字列を用いた問題である。
- 文字列の操作には、例えば C++ なら char 配列や string がある。(他の言語にもいろいろある)
- 文字列はエンバグしやすい筆頭
  - バグの原因が文字列操作であるとかは悲しい
  - 貴重なコンテスト時間を失ってしまう
- 慣れない操作はするべきではない（重要）
  - 事前に構造を把握したもの以外の使用は要注意
- 本問題（ここで解説する解法）では特に使用しないが、文字列特有のアルゴリズムとかある。
  - 興味があったら調べてみよう。

部分点解法

- 全てのマス（$N \times M$マス）について J,O,I それぞれの変更を実行し、変更後の JOI 旗に含まれる JOI 紋章を 1 個ずつ数える。

部分点解法

- 全てのマス（$N \times M$マス）について J,O,I それぞれの変更を実行し、変更後の JOI 旗に含まれる JOI 紋章を 1 個ずつ数える。
- 30点のテストケース→正解
- 70点のテストケース→TLE(時間超過)

部分点解法

- 全てのマス（$N \times M$マス）について J,O,I それぞれの変更を実行し、変更後の JOI 旗に含まれる JOI 紋章を 1 個ずつ数える。
- 30点のテストケース→正解
- 70点のテストケース→TLE(時間超過)

　→どうして満点が得られないのか？

部分点解法

- 全てのマス（$N \times M$マス）について J,O,I それぞれの変更を実行し、変更後の JOI 旗に含まれる JOI 紋章を 1 個ずつ数える。
- 30点のテストケース→正解
- 70点のテストケース→TLE(時間超過)

→どうして満点が得られないのか？

→計算量がO(N^2M^2)だから！

→計算量とは一体？

計算量とは

- プログラムの実行において、比較、代入、加減などの初等動作を行う回数にまつわる関数。

計算量とは

- プログラムの実行において、比較、代入、加減などの初等動作を行う回数にまつわる関数。
- よく O() などの形式で表される。

計算量とは

- プログラムの実行において、比較、代入、加減などの初等動作を行う回数にまつわる関数。
- よく O() などの形式で表される。
- 細かい説明とかは今回省きますが、理解に最低限必要な要点だけまとめると、
  - 実行時間が変数にどう比例するか見積もれる。
  - 主に最高次のものが使われる
  - これによって実行時間を予想できる。
  - 例 : O(N^2), O($N\log N$), O(2^N) などなど
  - 値を代入して 10 億超えてたりすると非常に遅い
  - 1 億は切っておきたい(可能なら数百万あたりまで)

部分点解法

- 全てのマス（$N \times M$マス）について J,O,I それぞれの変更を実行し、変更後の JOI 旗に含まれる JOI 紋章を 1 個ずつ数える。
- 30点のテストケース→正解
- 70点のテストケース→TLE(時間超過)

  →どうして満点が得られないのか？

    →計算量がO(N^2M^2)だから！

      →計算量とは一体？

      →制約の式N^2M^2に上限(1,000ずつ)を代入すると…

部分点解法

- 全てのマス（$N \times M$マス）について J,O,I それぞれの変更を実行し、変更後の JOI 旗に含まれる JOI 紋章を 1 個ずつ数える。
- 30点のテストケース→正解
- 70点のテストケース→TLE(時間超過)

→どうして満点が得られないのか？

→計算量がO(N^2M^2)だから！

→計算量とは一体？

→制約の式N^2M^2に上限(1,000ずつ)を代入すると…

→$1000^4 = 10^{12}$ (1 兆くらい！)

部分点解法

- 全てのマス（$N \times M$マス）について J,O,I それぞれの変更を実行し、変更後の JOI 旗に含まれる JOI 紋章を 1 個ずつ数える。
- 30点のテストケース→正解
- 70点のテストケース→TLE(時間超過)

　→どうして満点が得られないのか？

　　→計算量がO(N^2M^2)だから！

　　　→計算量とは一体？

　　　→制約の式N^2M^2に上限(1,000ずつ)を代入すると…

　　　　→$1000^4 = 10^{12}$ (1 兆くらい！)

　　　　　→これでは間に合わない…

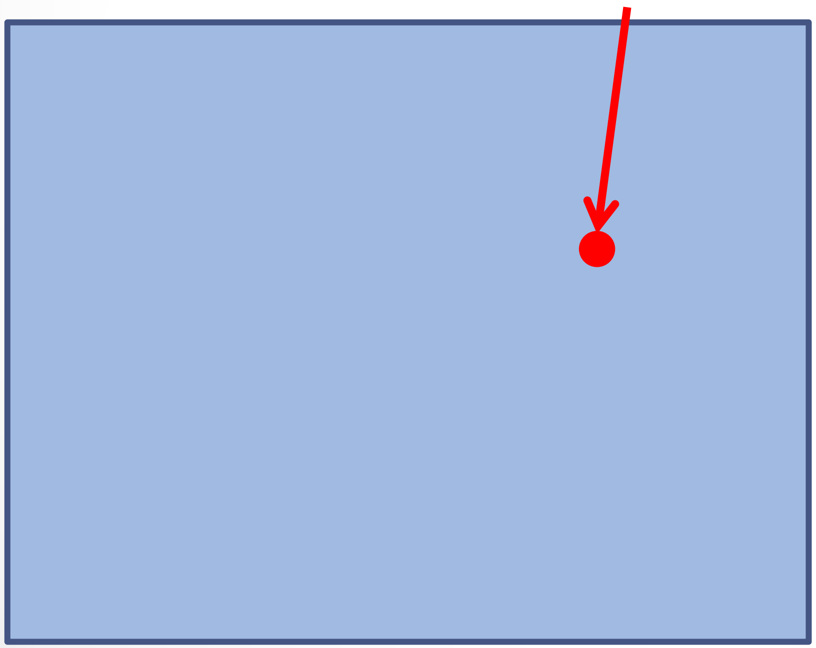
満点解法へのアプローチ

- 部分点解法に何か改善の余地は？

満点解法へのアプローチ

- 部分点解法に何か改善の余地は？
- 実は各書き換えにおいて、わざわざ全部の 2 × 2区間を見る必要がない！

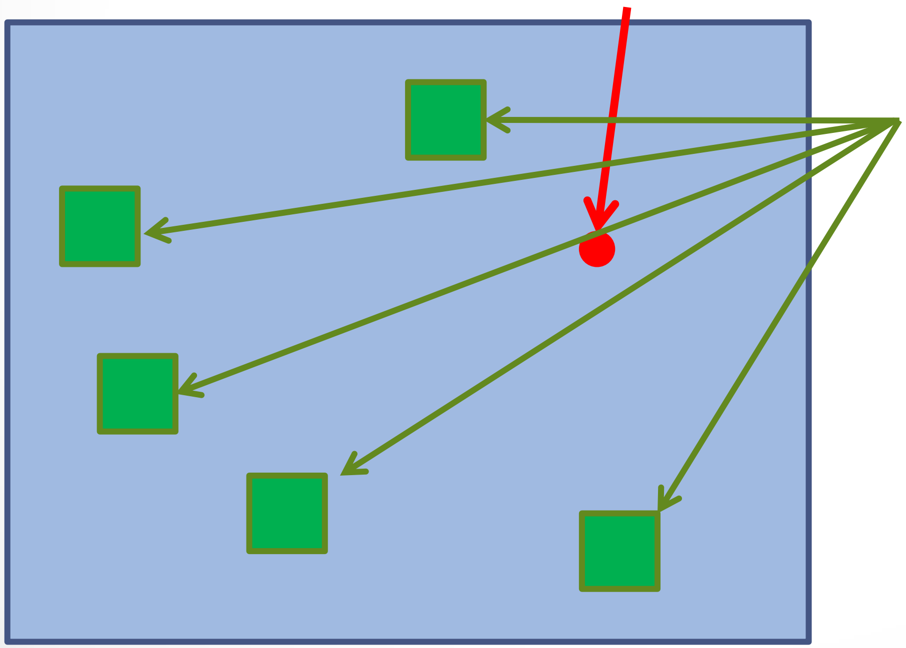
全体に対してここが変化したとき…



満点解法へのアプローチ

- 部分点解法に何か改善の余地は？
- 実は各書き換えにおいて、わざわざ全部の 2 × 2区間を見る必要がない！

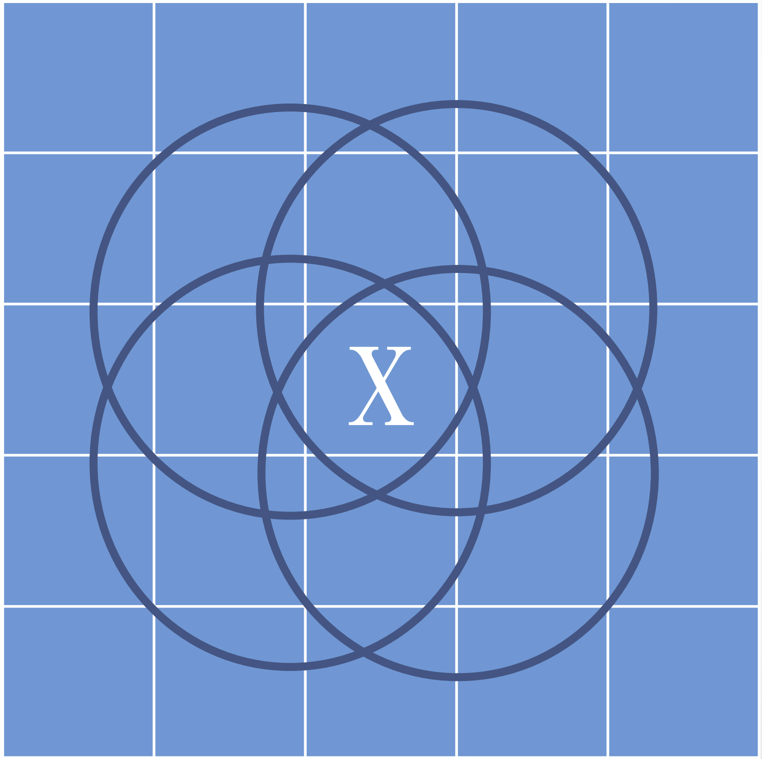
全体に対してここが変化したとき…



こんなところとかは調べる必要がない！（書き換えによる変化がない）

満点解法へのアプローチ

- より細かく言うと、下記のマスXを書き換えた場合にはXを含む4つの場所以外はXの中身によらず同じ結果となっているはず。



満点解法へのアプローチ

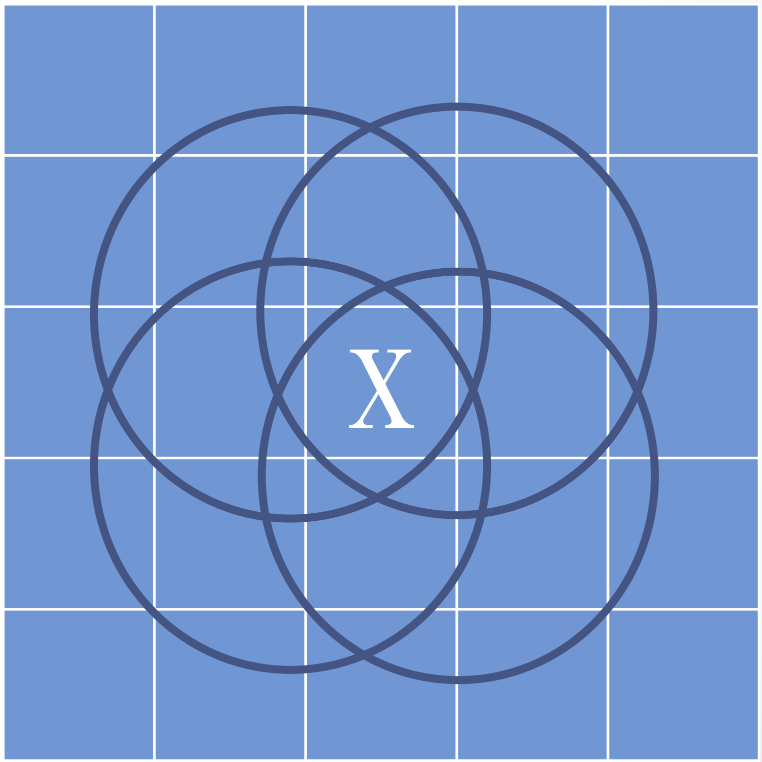
- より細かく言うと、下記のマス X を書き換えた場合には X を含む 4 つの場所以外は X の中身によらず同じ結果となっているはず。
- ということは、

(初期状態のJOI 紋章の個数)

＋

(書き換えによる変化量)

によって計算できる！



満点解法

- 満点解法は以下の手順で計算できる

満点解法

- 満点解法は以下の手順で計算できる

1.変更前の JOI 紋章の個数を計算する。

2.全ての$1 \leqq i \leqq N, 1 \leqq j \leqq M$に対し、次の処理を行う。

- マス(i, j)を J,O,I それぞれに書き換えてみる
- 周囲 4 箇所について紋章の増減の最大値を覚える (増減は -4 から +4 までの範囲の値を取る)

3.手順2で得られた最大値に手順1の値を加算して答え。

満点解法

- 満点解法は以下の手順で計算できる

1.変更前の JOI 紋章の個数を計算する。
　- 全ての領域を探索してO(NM)

2.全ての$1 \leqq i \leqq N, 1 \leqq j \leqq M$に対し、次の処理を行う。
　・マス(i, j)を J,O,I それぞれに書き換えてみる
　・周囲 4 箇所について紋章の増減の最大値を覚える
　(増減は -4 から +4 までの範囲の値を取る)

3.手順2で得られた最大値に手順1の値を加算して答え。

満点解法

- 満点解法は以下の手順で計算できる

1.変更前の JOI 紋章の個数を計算する。

- 全ての領域を探索してO(NM)

2.全ての$1 \leqq i \leqq N, 1 \leqq j \leqq M$に対し、次の処理を行う。

- マス(i, j)を J,O,I それぞれに書き換えてみる
- 周囲 4 箇所について紋章の増減の最大値を覚える
(増減は -4 から +4 までの範囲の値を取る)

- 各操作でO(1)となり、手順 2 全体ではO(NM)となる。

3.手順2で得られた最大値に手順1の値を加算して答え。

満点解法

- 満点解法は以下の手順で計算できる

1.変更前の JOI 紋章の個数を計算する。

- 全ての領域を探索してO(NM)

2.全ての$1 \leqq i \leqq N, 1 \leqq j \leqq M$に対し、次の処理を行う。

- マス(i, j)を J,O,I それぞれに書き換えてみる
- 周囲 4 箇所について紋章の増減の最大値を覚える (増減は -4 から +4 までの範囲の値を取る)

- 各操作でO(1)となり、手順 2 全体ではO(NM)となる。

3.手順2で得られた最大値に手順1の値を加算して答え。

- 全体でO(NM)となる。

満点解法

- 満点解法は以下の手順で計算できる

1.変更前の JOI 紋章の個数を計算する。

- 全ての領域を探索してO(NM)

2.全ての$1 \leqq i \leqq N, 1 \leqq j \leqq M$に対し、次の処理を行う。

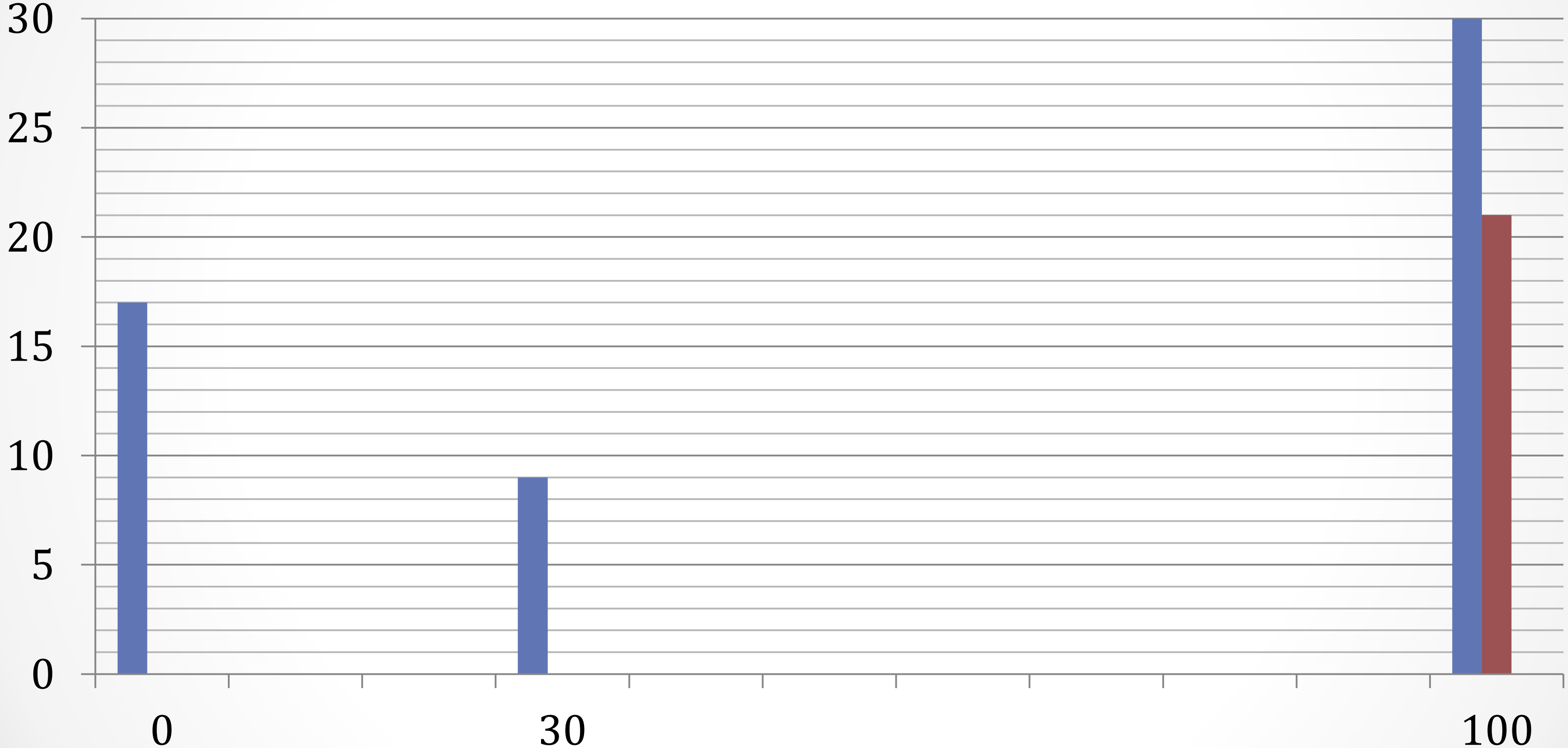
- マス(i, j)を J,O,I それぞれに書き換えてみる
- 周囲 4 箇所について紋章の増減の最大値を覚える
  (増減は -4 から +4 までの範囲の値を取る)

- 各操作でO(1)となり、手順 2 全体ではO(NM)となる。

3.手順2で得られた最大値に手順1の値を加算して答え。

- 全体でO(NM)となる。
- N, Mに1000を代入しても 100 万ほどで高速(満点)

得点分布



- 赤色は 31 人目以降を表しています。