# 第8回 日本情報オリンピック本選 解説[1]

2010 年1 月30 日

情報オリンピック日本委員会

# 1

# IOIOI

本問題は，文字列 $s$ と正整数 $n$ が与えられた時，$s$ に文字列 $P_n = \mathrm{IOIO}\cdots\mathrm{OI}$ (O が $n$ 個) が何個含まれているかを答える問題である．ここでは，全ての場所に対して，そこから始まる長さ $n$ の文字列が $P_n$ になっているかを 1 文字ずつ調べる素朴な解法と，$P_n$ の性質を利用して工夫して数える高速な解法を紹介する．

以下，$s$ の長さを $m$ とおく．また，$s$ の $k$ 文字目を $s_k$ とおき，$s$ の $k$ 文字目から始まる長さ $2n+1$ の部分文字列を $t_k$ とおく．ただし，$k = 1, 2, \cdots, m-2n$ とする．

## 素朴な解法

各 $t_k$ について，先頭から 1 文字ずつ見てゆき，全ての奇数番目の文字が I，全ての偶数番目の文字が O になっているかを調べればよい．

このアルゴリズムでは，$m-2n$ 箇所について，$2n+1$ 文字を調べる必要があり，時間計算量は $O((m-2n)n)$ となる．この解法では，$n \leqq 100, m \leqq 1000$ を満たす 50%のデータについては実行時間制限内で答えを出すことが可能であるが，残るデータについては，実行時間制限に間に合わない可能性が高い．

## 高速な解法

$s$ の $k$ 文字目を終端として I が 1 文字おきで連続している個数を $I_k$，O が 1 文字おきで連続している個数を $O_k$ とおく．表 1 は $s$=OOIOIOIOIIOII の場合の例である．

| $k$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $s_k$ | O | O | I | O | I | O | I | O | I | I | O | I | I |
| $I_k$ | 0 | 0 | 1 | 0 | 2 | 0 | 3 | 0 | 4 | 1 | 0 | 2 | 1 |
| $O_k$ | 1 | 1 | 0 | 2 | 0 | 3 | 0 | 4 | 0 | 0 | 1 | 0 | 0 |

表 1: $I_k$ と $O_k$

ここで，$I_k$ は下のように計算することができる．ただし，$k < 1$ では $I_k = 0$ とする．

$$I_k = \begin{cases} I_{k-2} + 1 & (s_k = I) \\ 0 & (s_k = O) \end{cases}$$

$s_k$ が I ならば 2 文字前までに連続している数+1 で，O ならば 0 ということである．$O_k$ についても同様に計算できる．計算の際は，前から順に計算してゆけば良い．

$t_k$ が $P_n$ と等しい条件は，$O_{i+2n-1} \geqq n$ かつ $I_{i+2n} \geqq n+1$ である．例えば，上述の例で，$n = 2$ とすると，$O_6(=3) \geqq n(=2)$ かつ $I_7(=3) \geqq n+1(=3)$ なので，$t_3$ は $P_n$ になっていると分かる．

| $k$ | $\cdots$ | $k$ | $k+1$ | $k+2$ | $k+3$ | $\cdots$ | $k+2n-1$ | $k+2n$ | $\cdots$ |
|---|---|---|---|---|---|---|---|---|---|
| $s_k$ | $\cdots$ | I | O | I | O | $\cdots$ | O | I | $\cdots$ |
| | | | | $\updownarrow$ | | | | | |
| I について | $\cdots$ | I | | I | | $\cdots$ | | I | $\cdots$ |
| O について | $\cdots$ | | O | | O | $\cdots$ | O | | $\cdots$ |

表 2: $t_k = P_n \iff O_{i+2n-1} \geqq n$ かつ $I_{i+2n} \geqq n+1$ であることの視覚的説明

よって，$m$ 箇所について $I_k, O_k$ の値を計算し，$m-2n$ 箇所について条件を満たしているかを調べれば答えが求まる．時間計算量は $O(m)$ となり，このアルゴリズムを用いれば，全ての採点用データで実行時間制限を満たすことができる．

## 補足

一般に，文字列 $s$ に含まれている文字列 $t$ の個数を数えることは，KMP 法などのアルゴリズムを用いることで，時間計算量 $O(|s|+|t|)$ で行うことが可能である．ただし，$|s|, |t|$ はそれぞれ $s, t$ の長さである．

# 2 ピザ

## 素朴な解法

1 件の注文が与えられると，各店舗からその注文先までの道のりをすべて調べ，どの店舗からその注文先へ配達するのが最も近いかを調べる．これを $m$ 件の注文について繰り返す．

この方法の時間計算量は $O(nm)$ に達する（1 件の注文を処理するのに $O(n)$ の時間がかかる）．この方法では 40 点しか得られない．

## 解法

まず前処理として，全ての店舗を，その環状線における位置の順番でソートする．そして，1 件の注文が与えられると，どの店舗からその注文先へ配達するのが最も近いかを，二分探索法を用いて調べる．

二分探索法は，ソートされたデータに対して用いるものである．そのために最初に店舗をソートしている．この際「本店から時計回りに測った道のり」順にソートすればよいが，本店は「本店から時計回りに測った道のり」が 0 の店と $d$ の店，という 2 軒の店舗として扱うとよいであろう．

この方法で時間計算量は $O((n+m)\log n)$ となる（前処理のソートに $O(n\log n)$ の時間がかかり，また 1 件の注文を処理するのに $O(\log n)$ の時間がかかる）．この方法を用いると満点を得ることが出来る．

## 別解

前処理として，すべての店舗を位置順でソートしておく（本店については，上で述べたようにして扱う）．さらにそれとは別に，すべての注文も，その注文先の位置順でソートしておく．

ソートされた順番で 1 件ずつ注文を処理していく．1 件目の注文先に最も近い店舗を，1 軒目の店舗（こちらもソートされている）から順に調べていく．これが $t_1$ 軒目の店舗だったとしよう．このとき，2 件目の注文先に最も近い店舗は，$t_1$ 軒目の店舗から順に調べていくと必ず見つかる．これが $t_2$ 軒目の店舗だったとしよう．このとき，3 件目の注文先に最も近い店舗は，$t_2$ 軒目の店舗から順に調べていくと必ず見つかる．$\cdots$

これを続けていくと，全ての注文先に最も近い店舗を決定することが出来るが，この方法で時間計算量は $O(n\log n + m\log m)$ となる（注文先から店舗までの道のりを調べた回数は $O(n+m)$ 回であり，前処理のソートよりも軽い）．この方法でも満点を得ることが出来る．

# 3 あみだくじ

## 解法

この問題に対する一番素朴なアプローチは，横棒を 1 本ずつ隠しながら線をたどる方法であろう．しかしこの方法では「(隠す線の本数 $m$) $\times$ (たどる縦棒の本数 $k$)」を必要とするため，$O(mk)$ の時間がかかってしまう．つまりこの方法では満点を取ることができない．

この問題の重要なポイントは同時に隠すことができる横棒が 1 本ということである．つまり，各横棒についてなにがしかの処理を行えば，$O(m)$ 時間で処理することが可能になるというわけである．

## 横棒の情報を作る

まず全ての横棒を高さでソートする．その後，各横棒について「上にたどっていた先の番号」と「下にたどっていった先のスコア」を左・右それぞれについて求める．

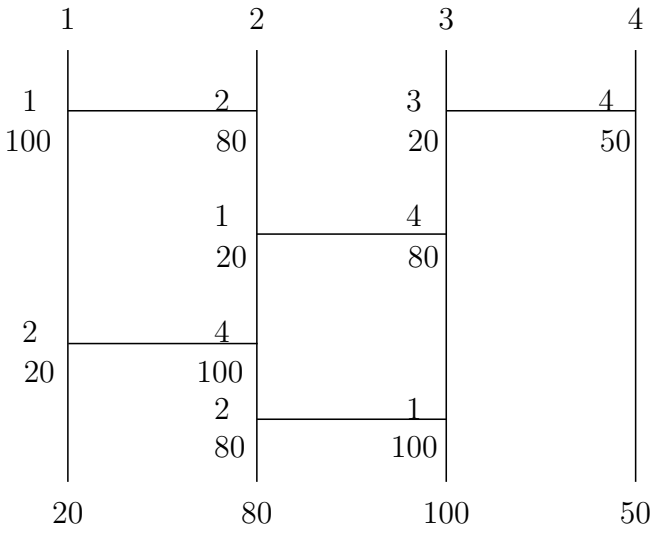


図: 横棒についての情報作成

上の図は，問題文中の入出力の例 1 について，それらの情報を求めたものである．各横棒の左上，右上に縦棒の番号を，左下，右下にスコアを書いている．

この情報は，長さが縦棒の数の配列に，縦棒の番号を入れ，横棒が現れるたびに交換する操作で作ることができる．よって，この操作は $O(m)$ 時間で可能である．

## 仕上げ

先ほど作った図を利用すると，線を隠したときのスコアがどのようになるかがわかる．たとえば，一番左上の線についてみると「1 番と 80 点，2 番と 100 点」が繋がることになる（クロスしたデータを見ている）．横棒をとった場合は「1 番と 100 点，2 番と 80 点」が繋がることになる（縦にデー

タを見ている）．後はこの情報を頼りに，J 君が選ぶ $(1 \ldots k)$ に属す番号と，属さない番号がペアになっている横棒の中からスコアが一番小さくなるもの探すことで解が求められる．こちらも $O(m)$ 時間で処理可能である．

## 今回のテストデータ

この問題では部分点が何段階かに分けて設定されているので，想定解法以外でも 30% もしくは 60% の点をとることは十分に可能であろう．

| 番号 | $n$ | $m$ | $h$ | $k$ | 備考 |
|---|---|---|---|---|---|
| 1 | 10 | 15 | 5 | 5 | |
| 2 | 2 | 9 | 10 | 1 | 削除なし |
| 3 | 10 | 9 | 10 | 1 | 削除なし |
| 4 | 100 | 400 | 20 | 50 | |
| 5 | 200 | 800 | 20 | 195 | $n-k$ が小さい |
| 6 | 200 | 1000 | 30 | 100 | |
| 7 | 500 | 10000 | 100 | 249 | |
| 8 | 600 | 20000 | 100 | 300 | |
| 9 | 800 | 50000 | 200 | 400 | |
| 10 | 1000 | 100000 | 1000 | 500 | |

表: 問題 3 の採点用データ

# 4 さんぽ

## 素朴な解法

太郎君の散歩および交差点の文字の変化の規則を素直に実装し，太郎君の毎回の散歩の経路をシミュレートすれば，太郎君の $N$ 回目の散歩の経路を求めることはできる．

この方法を用いた場合，1 回の散歩のシミュレーションにかかる時間は $O(W+H)$ であり，そのシミュレーションを $N$ 回行うことになるので，時間計算量は $O(N(W+H))$ となる．配点の 30% にあたる $W, H \leqq 100$，$N \leqq 1000$ のデータでは，この方法で時間内に正解を得ることができる．

## 高速な解法

各々の交差点について,「太郎君が $N-1$ 回目までの散歩でその交差点を通る回数」を，以下で述べるように動的計画法で求めることができる．この情報がわかれば，$N-1$ 回目の散歩が終わった時点で各々の交差点に書かれている文字も求まり，$N$ 回目の散歩の経路が（1 回のシミュレートで）わかることに注意しよう．

交差点を通過する回数は以下のように求まる：

- 太郎君の住んでいる交差点 $(1,1)$ については，回数は $N-1$ である．
- そうでない場合，求める回数は「その交差点を北から訪れる回数」と「その交差点を西から訪れる回数」の和に等しい（北や西に交差点がない場合はそれに対応する回数は 0 と数える）．前者は，北隣の交差点から南へ移動する回数に等しく，それは，北隣の交差点を訪れた回数を $k$ とすれば，

$$\begin{cases} \dfrac{k}{2} & (k \text{ が偶数のとき}) \\ \dfrac{k+1}{2} & (k \text{ が奇数で，北隣の交差点に最初書かれていた文字が「南」のとき}) \\ \dfrac{k-1}{2} & (k \text{ が奇数で，北隣の交差点に最初書かれていた文字が「東」のとき}) \end{cases}$$

となる．後者も同様である．

時間計算量は $O(WH)$ となり，すべてのデータにおいて時間内に正解を得ることができる．

# 5 認証レベル

本問題は，$W_i \times H_i$ の長方形に並んだ正整数が 2 組与えられ，そのうちの 1 つずつが指定されたとき，それぞれのレベルを定め，レベルの値以下の連結成分の成分数を求めなければならない．その和を与えられた整数 $R$ 以上にする最小のレベルの和を解答する．

長方形に並んだ正整数の最大値を $L$ とおく．

## 長方形が 1 つの場合

レベルから連結成分の成分数を求めるのは，深さ優先探索・幅優先探索いずれでも $O(WH)$ 時間で可能である．また，連結成分が $R$ 以上か未満かを判定すればよく，$R$ 個たどった後探索を打ちきることで時間計算量は $O(R)$ となる．素朴な解法では $O(RL)$ 時間でで問題を解くことが可能であるが，レベルに関して成分数は単調増加であるから，この場合二分探索により $O(R \log L)$ 時間で解くことができる．

## 部分点の考えられる解法

さて，もとの問題の解法を考えよう．それぞれの長方形についての連結成分の成分数を，0 以上，1 以上，$\cdots, R$ 以上にするために必要なそれぞれのレベルを求めればよい．素朴な解法を用いる場合，これらを同時に求めることが可能であり，時間計算量は $O(RL)$ となる．レベルとして長方形に書かれている数のみを考えれば十分であることを用いると，時間計算量は $O(RWH)$ となるが，すべてのデータに対しては速度がまだ不十分である．

長方形ごとにレベルを定める必要があるため，上述の二分探索をそのまま用いることはできない（適切な順序を入れることができない）．それぞれの長方形について $R+1$ 回の二分探索を用いることもできるが，この時間計算量は $O(R^2 \log L)$ となってしまう．

## 高速な解法

連結成分の成分数を 0 以上，1 以上，$\cdots, R$ 以上にするために必要なレベルを同時に高速に計算することを考えよう．実は，これらを成分数の順に求めていくことが可能である．あるレベルの値以下の連結成分の成分数が $r$ となるとき，それら $r$ 成分の上下左右の成分のうちもっとも小さい数が，連結成分の大きさを $r+1$ 以上にする最小のレベルである．

この計算をするときに，連結成分の上下左右として考えられる数は $O(r)$ 個あるため，すべてを毎回調べると合計の時間計算量は $O(R^2)$ となるが，データ構造を工夫すると少ない時間計算量で解くことができる．いま，必要な処理は「上下左右の成分を追加すること」と「最小のものを取り出すこと」である．これは priority queue を用いることで 1 回あたりの操作の時間計算量が $O(\log r)$ となる．このとき，合計の時間計算量は $O(R \log R)$ となり，すべてのデータに対して制限時間内に正解できる．