

AtCoder Beginner Contest 038

解説



AtCoder株式会社 代表取締役
高橋 直大

-
- 競技プログラミングをやったことがない人へ
 - まずはこっちのスライドを見よう！
 - <http://www.slideshare.net/chokudai/abc004>

A問題 お茶

1. 問題概要
2. アルゴリズム

- 文字列 S が与えられる
- S の末尾の文字が T であるか判定せよ
- 制約
 - $1 \leq (Sの長さ) \leq 50$
 - S は英大文字のみからなる

- 方法1

- Lを文字列の長さとする、Sの末尾の文字はSのL番目の文字のこと

C++では、`S[S.size()-1]`

- 方法2

- 文字列をreverseすると、先頭の文字が末尾の文字になっている

- その他:

- ライブラリの関数を呼び出すなど

C++だと、`(*S.rbegin())`なんて表し方もあります

B問題 ディスプレイ

1. 問題概要
2. アルゴリズム

- 2つのディスプレイを横に並べて置き、高さをそろえることができるか判定
- ただし、ディスプレイは90度回転させることができる
- 制約
 - $1 \leq (\text{高さ} \cdot \text{幅}) \leq 10^5$

- どのようなとき、高さをそろえることができるか...
 - 高さを x [mm]で揃えるとき、片方のディスプレイの高さと幅のどちらかが x [mm]で、もう片方のディスプレイの高さと幅のどちらかが x [mm]でなければならない
 - すなわち、片方のディスプレイの高さ,幅は、もう片方のディスプレイの高さ,幅と共通の長さを1つは持つ・・・①
 - 逆に、共通の長さを持つとき、その長さが高さとなるように回転させて置けばOK
- これで、①の条件で判定すればいいことが分かった

- 共通の長さを持つか判定するために、この問題では
2*2の4通り試せばOK
 - すなわち...
 - if $H1=H2$ or $W1=W2$ or $H1=W2$ or $W2=H1$ YES
 - else NO

C問題 単調増加

1. 問題概要
2. アルゴリズム

- 長さ N の数列が与えられます。 i 番目の数は a_i
- 数列の l 番目から r 番目までが単調増加となるような (l,r) の数を求めよ
- 制約
 - $1 \leq a_i \leq 10^5$
 - $1 \leq N \leq 10^5$

- 部分点1:

- 全ての1以上N以下の l に対し、 (l,r) が条件を満たすような r の数を求めてみる
- (l,l) は常にOK
- $(l,l+1)$ は $a_l < a_{l+1}$ ならOK
- $(l,l+2)$ は $(l,l+1)$ がOKで $a_{l+1} < a_{l+2}$ ならOK...
- といったように、数列のある部分が条件を満たすかは、最後の値がその1つ前の値より大きく、かつ最後の値以外も単調増加になっているかに等しい
- 長さを1ずつ増やしながらか、 r が条件を満たすか求めつつ、条件を満たすなら答えを増やす
 - l が N 通り、長さも1から $N+1-l$ まで増やすので、計算量は $O(N^2)$

- 満点解法:
 - ある (l,r) が条件を満たすとき...
 - $l < r$ なら、 $(l,r-1)$ もまた条件を満たす
 - これより、全ての1以上 N 以下の l に対し、 (l,r) が条件を満たす最大の r' が分かれば、 a_l から始まる数列で条件を満たすものの数は、 $r'-l+1$
 - r' を求めるために数列を l 番目から探していくと、数列全体が単調増加なときにまだ $O(N^2)$ かかってしまう。

- ある l に対し、 (l,r) が条件を満たす最大の r を r' とおく
- このとき、 $(l+1,r)$ が条件を満たす最大の r は r' 以上
 - l 番目から r' 番目は単調増加なので、数列の $l+1$ 番目から r' 番目までの部分は単調増加
- このことから、以下のようなコードが組める:

```
l:=1, r:=1, ans:=0
```

```
while l <= N
```

```
  while  $(l,r+1)$  が条件を満たす r:=r+1
```

```
  # この時点で $r$ は $(l,r)$ が条件を満たす最大の $r$ 
```

```
  answer:=answer+(r-l+1)
```

```
  l:=l+1
```

```
end
```

- この計算量は？
- l は1から N まで増え、 r はその中でたくさん増えるかも
 - でも、 r も全体で1から N までしか増えない
 - そのため、 r を増やすwhileは、高々 N 回しか回らないことが分かる
- 内側のwhileが N 回しか回らないことが分かったので、 $O(N)$
 - しゃくとり法と呼ばれる手法

D問題 プレゼント

1. 問題概要
2. 考察
3. アルゴリズム

- 長方形が N 個あり、横が $h_i[\text{cm}]$ と縦が $w_i[\text{cm}]$ と決まっている。
- 最大で何重の入れ子にできるか？
- 制約
 - $1 \leq N \leq 10^5$
 - $0 \leq h_i, w_i \leq 10^5$

- 部分点解法:
 - DPを考えてみよう！
 - $dp[i]$: i 番目の箱を最も外側とするとき、最大で何重の入れ子とできるか
 - $dp[i] := \max(dp[j]) + 1$ ただし、 i 番目の箱は j 番目の箱を入れることができる
 - これをメモ化再帰として実装すればOK
 - ある箱は直接または間接的にその箱自身を含むことはできないため、DPの遷移で閉路が生じない
 - $O(N^2)$

- 満点解法
- h_i がすべて互いに異なる場合をひとまず考えよう
 - $dp[i] := \max\{dp[j] \mid h_j < h_i \text{ かつ } w_j < w_i\} + 1$ というDPの遷移だった
 - $\max\{f(x) \mid \text{条件}\}$ というのは、条件を満たすような x での $f(x)$ の最大値という意味
 - 箱を h_i が昇順となるようにソートすると、 $dp[i] := \max\{dp[j] \mid j < i \text{ かつ } w_j < w_i\} + 1$
 - よって、 i を 1 から N まで増やしながらか DP を求めていくことにすると、 w がより小さいもののうちの最大値が求めれば OK

- w がある値以下という条件での最大値を高速に求めるデータ構造が欲しい
 - > BIT(Binary Indexed Tree, Fenwick Tree)が手軽かつ便利
 - 解説は諸サイト・スライドに譲ることにします
 - 蟻本にも載ってる！
- BITは区間の和を持つ方法が最も基本的だが、列で1番目から k 番目の最小値を求める操作と列の値の更新の操作も行うことができる
 - 今回はこの操作を使います

- BITで扱う列の*i*番目を最も外側の箱の横の長さが*i*となる入れ子の数の最大値とする
- `query(i)`で列のうち*i*番目まで最大値、`update(i,a)`を列の*i*番目を*a*で更新、とすると、
以下のような感じ
for `i` from 1 to `N`
 `dp[i]:=bit.query(wi-1)+1`
 `bit.update(wi, dp[i])`
最終的に、`dp[i]`のうち最大値が答え
- BITの1回の更新/最小値クエリーはどちらも $O(\log N)$ なので、 $O(N \log N)$

- h_i に同じ数があるとき...
 - h_i が同じ箱は w_i の降順にソートするとそのままのコードでうまくいく
 - h が同じ箱はどれも互いに入れ子にならないので、 h が同じもので入れ子を作らないように並んでいて欲しい
 - w_i の降順に並べると、先にdpの値が計算された箱を含むような入れ方はないため、 h が同じ箱で入れ子になることがなく、正しい答えが得られる
 - その他に、 h が同じ箱を計算している間はBITをupdateせず、 h が変わるとそこでupdateを全て行うという方法もある