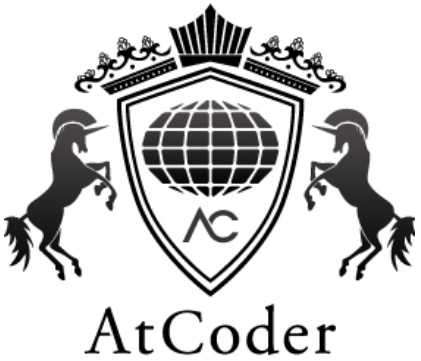
AtCoder Regular Contest 035
解説



AtCoder株式会社 代表取締役
高橋 直大

競技プログラミングを始める前に

AtCoder

- 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　高橋くんと回文

1. 問題概要
2. アルゴリズム

A問題　問題概要

AtCoder

- 文字列 *s* が与えられる.
- *を自由な文字に置き換えて，sを回文にできるか判定
- *を置き換える文字はそれぞれ異なっていてもOK
- 制約
  - 1 ≦ |s| ≦ 1000

A問題　アルゴリズム

AtCoder

- 解法1
  - 前と後ろで対応する文字が同じになりうるか調べる
  - つまり，i 番目と |s|+1-i 番目の文字で，どちらかが * であるか，どちらも同じ文字であるか調べる(1 <= i <= |s|)
  - 1つでも同じになりえない文字の組があればNO，そうでなければYES
- 解法2
  - sとsをひっくり返した文字列tの間で，sのi番目とtのi番目が同じ文字になりうるか調べる.
  - 結局やっていることは同じ．この方が少し楽かも.

B問題　アットコーダー王国のコンテスト事情

1. 問題概要
2. 考察＆解法

B問題　問題概要

AtCoder

- 問題がN個ある
- それぞれの問題を解くのにかかる時間を知っている
- 解いた時間に依存するペナルティがある
  - 問題ペナルティ=コンテスト開始からその問題を正解するまでの時間
  - コンテストペナルティ=問題ペナルティの総和
- 全完するときの最小コンテストペナルティと，それを達成する解き方の数を求めよ
- 制約
  - 1 ≦ N ≦ 100000
  - 1≦(各問題を解くのにかかる時間)≦ 1000

B問題　考察＆解法

AtCoder

- コンテストペナルティ最小を達成するためには，
  かかる時間の小さい問題から解けば良い

→かかる時間順でソートして，シミュレーションする

(最大ケースで答えが32bit整数を超えるので注意)

- かかる時間が同じ問題は，解く順序を入れ替えても
  最小ペナルティが変わらない．

→かかる時間が同じ問題がx個あったら，解き方はx!通り

- 全体の組み合わせ数としては，それらの積が答え
- 計算量
  - ソートにO(n log n) ←標準ライブラリを使いましょう
  - その他 O(n)

C問題　アットコーダー王国の交通事情

1. 問題概要
2. 考察
3. 解法

C問題　問題概要

AtCoder

- N頂点,M本の辺から成る重み付き無向グラフがある
- このグラフにK本無向辺を新たに追加する
- 追加する度に，全点対間の距離の和を求めよ

- 制約
  - 1 ≦ N ≦ 400, 1 ≦ M ≦ 1000, 1 ≦ K ≦ 400
  - 1≦(各問題を解くのにかかる時間)≦ 1000

C問題　考察

AtCoder

- 全点間最短距離を求めるワーシャルフロイド法を用いる
→O(N^3)で全点間距離テーブルが計算可能
→辺が1つ増える毎にテーブルをO(N^3)で更新していてはTLE
- ある辺を既存の辺より小さいコストの辺に変更するときO(N^2)でテーブル更新可能
→コストがcの辺(a,b)を追加時，頂点i,j間の最短距離cost[i][j]は以下のように更新される

cost[i][j] = min(cost[i][j], cost[i][a] + cost[b][j] + c,
cost[i][b] + cost[a][j] + c)

※cost[][]は全点間最短距離を記録した二次元テーブル

C問題　解法

AtCoder

- まとめると
  - 最初にO(N^3)で全点間最短距離テーブルを構築
  - 辺の追加時に毎回O(N^2)でテーブル更新
- を行うことで満点を得れる

- 計算量: O(N^3 + KN^2)

- 余談ですが，ワーシャルフロイドのアルゴリズムはO(N^3)ですが，アルゴリズムが非常に単純なので，多少頂点数が多くても高速に動作することが多いです．

D問題　高橋くんとマラソンコース

1. 問題概要
2. アルゴリズム

D問題　問題概要

AtCoder

- 南北・東西方向の道があり，交差点の間を北・東方向にのみ進める.
- 交差点上にN個のチェックポイントがある.
- 以下の2つのクエリーを処理
  - k_j 番目のチェックポイントを (a_j,b_j)に設定しなおす
  - チェックポイント$l1_j$からチェックポイント$r1_j$までの経路数と，チェックポイント$l2_j$からチェックポイント$r2_j$までの経路数のうち，どちらが多いか答える. ただし，多い方は少ない方の2倍以上.
- 制約
  - 2<=N<=2*10^5
  - 1<=(チェックポイントの座標)<=10^6

D問題　アルゴリズム

AtCoder

- 東西方向で東を正にX軸，南北方向に北を正にY軸を取ります．
- 隣り合うチェックポイントのx座標の差がdx，y座標の差がdyのとき，その2つのチェックポイント間の経路の数は

$$_{dx+dy}C_{dx} = (dx+dy)!/(dx!)/(dy!)$$

(dx個の→，dy個の↑の並べ替えの個数に等しいから)

- チェックポイントaからチェックポイントbまでの経路の数は，(aとa+1間の経路数)*(a+1とa+2間の経路数)*・・*(b-1とb間の経路数)
- 経路数はメチャクチャでかくなりそう．
  - (1,1) から (10^6,10^6) への経路数の時点でヤバイ
- 経路数の計算では掛け算・割り算しか出てこない．

D問題　アルゴリズム

AtCoder

- 普通に計算するのはよくない
- logを取ってみよう！！
  - 0<a<b のとき log(a)<log(b) なので，大小関係が維持される.
  - log(a*b)=log(a)+log(b) , log(a/b)=log(a)-log(b) から，経路数は楽に計算できそう.
  - 数がそこまででかくならない
    - (1,1) から (10^6,10^6) までの経路数をwとおくと，log(w)は10^6ぐらい.
  - 大と小で2倍差があれば，logを取った値はlog(2)≒0.69以上差があるので誤差もOK

D問題　アルゴリズム

AtCoder

- 部分点解法では，番号が隣り合う2つのチェックポイントの間の経路数にlogを取ったものを毎回計算し，それらの和の大小を比較する．
  - 予め階乗にlogを取ったものを求めておく必要がある
- 満点解法では，セグメント木を使い，区間の和を求められるようにしておく．
  - チェックポイントの更新では，その前後の経路数を更新
  - 比較では，和の大小を比較