AtCoder Beginner Contest 030
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　勝率計算

1. 問題概要
2. アルゴリズム

A問題　問題概要

AtCoder

- 4つの整数 A, B, C, D が与えられる.
- B/A と D/C を比較
- 制約
  - 1 ≦ A, B, C, D ≦ 1000

A問題　アルゴリズム

AtCoder

- 解法1
  - B/A , D/C の両方にA*Cを掛けると

    B*C, A*D となるので、この2つを大小比較
    - 整数同士の掛け算なので安全（ロバスト）です
- 解法2
  - B/A, D/C を小数にして比較
    - 比較には浮動小数点誤差に影響されないように、微小な数EPSを用意し、
      - x<y → x+EPS<y
      - x==y → abs(x-y)<EPS
    - と計算しましょう

B問題　時計盤

1. 問題概要
2. アルゴリズム

B問題　問題概要

AtCoder

- アナログ時計がある。長針と短針のなす角を求めよ
- 制約
  - $0 \leqq n \leqq 23$
  - $0 \leqq m \leqq 59$

B問題　アルゴリズム

AtCoder

- 長針は1時間ごとに360/12で30度、1分ごとに360/12/60で0.5度動きます。
- 短針は1分ごとに360/60で6度動きます。

- 長針と短針が12時の方向から見てどれだけ進んでいるかを0～360度の小数で表し、差を取ります。
- ただし、180度より大きいとき、大きい方の角を見てしまっているため、360度から差の角を引いたものが答え。
  - 整数/整数は言語によっては切り捨てられてしまうため注意！

C問題　飛行機乗り

1. 問題概要
2. アルゴリズム

C問題　問題概要

AtCoder

- 空港A, Bを出発する飛行機がそれぞれN, M本ある
  - Aを出発する時刻は a_i, Bを出発する時刻は b_j
- A→BにはX時間、B→AにはY時間かかる
- 時刻0に空港Aにいるので、なるべく多く空港間を往復したい
- 制約
  - 1<= N, M <=10^5, 1<=a_i, b_j <=10^9
  - 30点: 1<=a_i, b_j <=10^5

C問題　アルゴリズム

AtCoder

- どの飛行機に乗っても、空港間を往復する時間は変わらない
- このとき、なるべく多く往復したいなら、乗れる飛行機のうち最も早いものに乗るのが最適
  - 最も早いもの以外に乗ったとき、到着時刻もそれだけ遅れ、次に乗る飛行機の選択肢が減る（ことがある）。
  - 最も早い飛行機以外に乗る最適な乗り方があるとすれば、最も早いものに乗り、時間の差だけ次の空港で待てばいいため、最も早いものに乗っても最大の往復回数を実現できる。
    - →→最も早いものに乗っても損は絶対ない。得もたまにある

C問題　アルゴリズム

AtCoder

- 30点解法
  - 空港A, Bについて、時刻t(0～10^5)に出発する飛行機があるかどうかを全て配列に初めにメモしておく
  - (今いる空港, 今の時刻) を覚えて、ちょうど出発する飛行機に乗れるなら乗る、というシミュレーションを繰り返す
  - 空港Aに戻ってきた回数が答え。
- 満点解法
  - (今いる空港, 今の時刻) を覚えて、次に出発する最も早い飛行機を二分探索で探す
    - 配列から(今の時刻)以上で最小の値を探すだけなので、簡単に(C++ならstd::lower_bound)求まります
  - あとは部分点と同様。

D問題　へんてこ辞書

1. 問題概要
2. 解法

D問題　問題概要

AtCoder

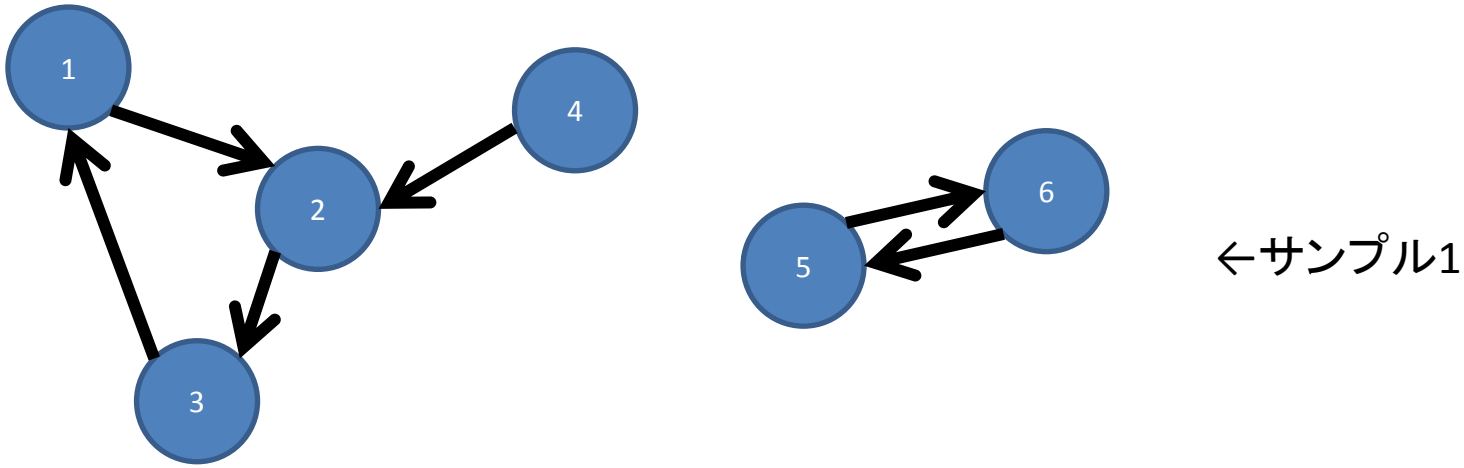
- 1からNの整数 i に対し、$i \rightarrow b_i$ という移動が決まっている
- 初め a にいるものとする。k 回移動したとき、どこにいるだろうか？
- 制約
  - 2<=N<=100000
  - 20点: k<=100000
  - 80点: $k<=10^{100000}$

D問題　解法

AtCoder

- 20点解法は、ミカミくんの移動をシミュレーションすればOK (O(k))

- グラフにして考えてみよう！
  - N個の頂点のある有向グラフで、それぞれの頂点からは1つのみ辺が出ている(入ってくる辺は複数あるかも)



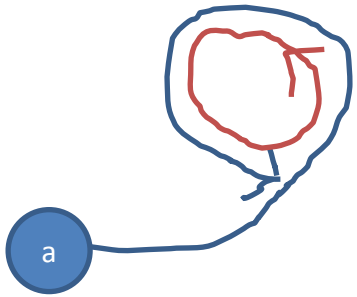
←サンプル1

D問題　解法

AtCoder

- どの頂点も、辺を辿って行くといくつか進んだ後は閉路に到達しその閉路の中をずっと辿り続ける
  - 頂点はN個しかないので、ずっと辺を進んでいくといつかは今まで辿ったことのある頂点に戻ってくる。どの頂点も出て行く辺はちょうど1本なので、戻ってきた後は全く同じ移動を繰り返すから

こんなイメージ→



    - (蛇足)全体のグラフは、連結成分(辺でつながっている頂点たち)ごとに分けると、連結成分のどの頂点も最後にぐるぐる回ることになる閉路と、その閉路に向けてつながる木の部分からなる

D問題　解法

AtCoder

- 閉路の長さをC、閉路に到達するまでの長さをT とおきます。
- k が十分大きく、k 回移動したときに閉路に到達しているものとします。
- このとき、k mod C (k を C で割った余り) さえ求まればOK
  - 閉路に到達しているぐらい十分大きいなら、C 回違ってもちょうど閉路1周分違うので、到達する頂点は変わらない
  - 始点aから k mod C 回動けばいい(厳密にはちょっと違いますが後述)
- ひとまず k mod C を求めてみる
  - 多倍長を使ってもOKですが、多倍長無しで簡単に求める方法があります(桁DPなどでよく使うので、多倍長の付いた言語を使っている人にも有用なテクです)

D問題　解法

AtCoder

- k の長さを L とおきます。
- k = (k の L-1 文字目までを整数として捉えたもの)*10+(kの最後の数字の数) となる
  - Ex. 334=33*10+4
  - Ex. 123456=12345*10+6
- このことに注目すると、(kのL-1 文字目まで) mod C が求まっていればよさそう。
- この1桁減らした数に帰結することを繰り返すと、大きい数の桁から見ていき、今までの数に10を掛け、今見ている桁の数を足す 操作を繰り返せば、k mod C が求まる。

D問題　解法

AtCoder

– kを文字列として
　保存するとこんな感じ→

```
int k_mod_C=0;
for(int i=0;i<L;++i){
  k_mod_C=(k_mod_C*10+k[i]-'0')%C;
}
```

- 始点 a からk mod C 回動けばいい
  - ただし、k回移動したときに閉路に入っていることを仮定していたので、k mod C が T より小さいとき、T以上になる(移動後の場所が閉路に入る)まで移動回数に C を足す必要がある
    - つまり、k を移動後の場所が変わらない範囲で小さくしたかった。k を k mod C にする操作は、kからCを引けるだけ引く操作なので、減らしすぎた分を戻す感覚
  - k mod C < C <= N なので、シミュレーションは高々O(N)

D問題　解法

AtCoder

- k 回動いた後に閉路に到達していないとき
  - 閉路に到達していないということは、同じ頂点を2回通っていない
  - 頂点はN個しかないので、高々N回しか移動していない！つまりk<=N
  - これならそのままシミュレーションすればOK
  - 逆に、k<=Nでも閉路に到達している場合もあるが、そのような場合もシミュレーションで答えは正しく求まる
  - なので、kがN以下のときはシミュレーション、それより大きければ閉路に入っていると仮定して解いてOK