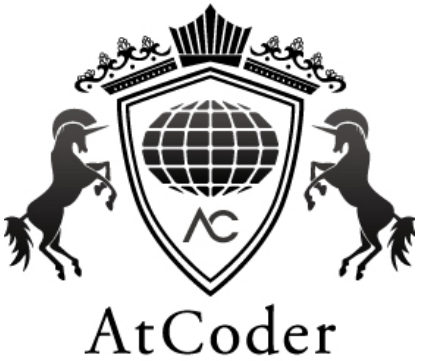
AtCoder Beginner Contest 011
解説



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　来月は何月？

1. 問題概要
2. アルゴリズム

A問題　問題概要

AtCoder

- 今月の月を表す整数Nが与えられる
- 来月は何月か出力

- 制約
- $1 \leqq N \leqq 12$

A問題　アルゴリズム

AtCoder

- 基本的なプログラムの流れ
  - 標準入力から、必要な入力を受け取る
    - 今回の場合は、N という1つの整数
  - 問題で与えられた処理を行う
    - 今回は、Nから、来月の値を算出
  - 標準出力へ、答えを出力する

A問題　アルゴリズム

AtCoder

- 入力
  - 1つの整数を、標準入力から受け取る
    - Cであれば、scanf(“%d”, &N);　など
    - C++であれば、cin >> N;
    - 入力の受け取り方は、下記の練習問題に記載があります。
      - http://practice.contest.atcoder.jp/tasks/practice_1

A問題　アルゴリズム

AtCoder

- 処理部分
  - 今回は、与えられた整数が表す月の、次の月を計算する
  - 計算方法は色々
    - オーソドックスな方法
      - まず、Nに1を足す。
      - Nが13になってしまったら、Nを1にする
    - ちょっと器用な方法
      - N %= 12; N += 1;
      - 12で割った余りを計算すると、12月が0月っぽくなる。
        - » こっちの方がかっこいいから個人的に好きだけれども、別にこんなことしなくても良い。
    - 1を足すが思いつかなかった人用
      - 配列に来月の月の番号を全部書いちゃう。
      - Int[] nextMonth = new int[]{-1, 2,3,4,5,6,7,8,9,10,11,12,1};
      - おすすめはしません。

A問題　アルゴリズム

AtCoder

- 出力
  - 求めた答えを、標準出力より出力する。
  - 言語によって違います。
    - printf(“%d¥n”, N); (C)
    - cout << N << endl; (C++)
    - System.out.println(N); (Java)
    - 各言語の標準出力は、下記の練習問題に記載があります。
      - http://practice.contest.atcoder.jp/tasks/practice_1

B問題　名前の確認

1. 問題概要
2. アルゴリズム

B問題　問題概要

AtCoder

- 文字列が与えられる。
- 先頭文字を大文字に、残りを小文字に変換しなさい。

- 制約
- $1 \leqq |S| \leqq 12$

B問題　アルゴリズム

AtCoder

- 入力
  - 文字列Sを受け取る
    - 解らない場合はpracticeで確認しよう！
      - http://practice.contest.atcoder.jp/tasks/practice_1

B問題　アルゴリズム

AtCoder

- 処理
  - 先頭文字を大文字に変換する
  - 残りの文字を小文字に変換する

  - やるべき処理は2つ。
    - 今見ている文字が、大文字か小文字か確認する。
    - 必要であれば、大文字を小文字に変換する、または小文字を大文字に変換する

B問題　アルゴリズム

AtCoder

- 大文字、小文字の確認
  - 言語の標準ライブラリについている場合もある。
    - isLowerなど。
  - ついていない場合は、文字コードを利用しよう！
    - If(c >= ‘a’ && c <= ‘z’)などで、小文字かどうかを判定できる。
    - ASCII文字コード表　http://e-words.jp/p/r-ascii.html
      - ‘A’が65, ‘B’が66, … ‘Z’が90
        - » よって、’A’以上’Z’以下なら大文字
      - ‘a’が97, ‘b’が98, … ‘z’が122
        - » よって、’a’以上’z’以下なら小文字

B問題　アルゴリズム

AtCoder

- 大文字、小文字の変換
  - 標準ライブラリについている場合もある。
    - toLowerなど
      - そもそもこれがあれば、大文字小文字の確認も要らない。
  - 小文字から大文字に変換したい場合
    - ‘a’と’A’の差は32、‘b’と’B’の差は32、‘z’と’Z’の差も32
    - つまり32を引けば、小文字は大文字になり、32を足せば、大文字は小文字になる。
      - といいつつも、連続していることさえ分かれば、数字を覚える必要は全くない
        - » C = (C + ‘A’ - ‘a’);　大文字への変換
        - » C = (C + ‘a’ - ‘A’);　小文字への変換
        - » ‘A’と’a’の差さえ分かれば計算可能なので、上のように書ける

B問題　アルゴリズム

AtCoder

- 出力
  - A問題と同じく、答えを出力するだけ
  - Print(S)みたいな感じ

- 注意点
  - 多くの言語では、string型の文字列を直接一字一字変換することは出来ない！
    - 答えを求めるために、新たにstring型を作っておいた方が良い！
  - Char型の配列であれば、これは気にしないで良い。

C問題　123引き算

1. 問題概要
2. アルゴリズム

C問題　問題概要

AtCoder

- Nから、1,2,3のどれかを100回まで引ける。
- Nを0にできたらクリア。
- NG数字が3つ与えられ、NがNG数字に一度でもなったらダメ。
- クリア可能かどうかを判定しなさい。

- 制約
  - 1 ≦ N, NG1, NG2, NG3≦ 300
    - NがNG数字と一致することがあることにも注意！
      - » ジャッジ解これでバグってました、ごめんなさい＞＜

C問題　アルゴリズム

AtCoder

- まずは全探索を考えてみる。
  - 100回、1,2,3のどれかを引くことを試みる。
  - 3つの分岐が100回。計算回数は3^100回ほどになる。
    - 地球が爆発するまで待っても間に合わない！
  - よって、何か工夫する必要がある。

C問題　アルゴリズム

AtCoder

- ダメなケースを考える
  - N以下のNG数字が3連続続いている場合
    - N=8, NG= {2,3,4}など
  - Nが大きく、NG数字が邪魔をして、1回じゃ間に合わない場合
    - N=300, NG={297,294,291};
      - 300->298の時点で、あと99手で減らせる最大数は297なので、間に合わない。
  - これらを場合分けするのは非常に大変！
    - » 無理ではない。が、やりたくない。
    - できれば全探索したい。

C問題　アルゴリズム

AtCoder

- 解法1 動的計画法を使った解法
  - 全探索は難しいので、全探索を上手くまとめてあげる。
    - 数字kに辿り着くまでに、最低何手必要かを、上手く処理してあげる。
  - dpをINFで初期化;
  - dp[N] = 0;
  - For i : N to 0
    - If(NG(i)) continue;
    - For j: 1 to 3
      - dp[i - j] = min(dp[i] + 1, dp[i – j]);
  - こんな感じで上から処理してあげる。
  - dp[0]が100以下ならYES、そうでないならNO

C問題　アルゴリズム

AtCoder

- 解法2 貪欲法を使った解法
  - そもそも、もし3を引いても大丈夫な時に、2や1を引く必要があるのか？
    - 2を引いて行ける範囲は、3を引いて行ける範囲より狭い。
    - 1も同様
  - よって、引ける数のうち、もっとも大きい数を引けば良い。
  - これを100回繰り返して、Nが0以下になっていればYES

AtCoder

- 注意点

Nが最初からNG
数字な時に注意しよう！！！

D問題　大ジャンプ

1. 問題概要
2. アルゴリズム

D問題　問題概要

AtCoder

- ランダムな4方向に、距離Dだけジャンプする
- N回のジャンプ後に、ゴール地点(X,Y)にいる確率を出力しなさい。

- 制約
  - 1≦N≦ 8(part1) , 30(part2), 1000(all)
  - 1≦X,Y,D≦10^9

D問題　部分点１　アルゴリズム

AtCoder

- 前処理
  - ジャンプの距離がDなのは面倒！1にしたい！
    - X /= D; Y /=D;しましょう。
    - 割り切れなかったら、確実に(X,Y)にはたどり着けないので、答えは0です。

D問題　部分点１　アルゴリズム

AtCoder

- 全探索について
  - 4方向に移動するので、パターン数は4 ^ N
  - 部分点1の制約なら、4^8は65536なので、計算可能
    - 全パターン調べて、(X , Y)に辿り着いたパターン数を、4^Nで割れば良い。
  - では、どうやって全パターンを調べるのか？
    - 深さ優先探索を使おう！

D問題　部分点１　アルゴリズム

AtCoder

- 深さ優先探索
  - 再帰を利用して、全パターンを列挙する
    - Int dfs(int count, int nowX, int nowY)みたいな関数を作る
      - CountがNであれば、X==nowX, Y==nowYの時に、1を返す
        - » それ以外は0を返す
      - CountがN以外であれば、4方向について探索する
        - » dfs(count+1, nowX + 1, nowY);
        - » dfs(count+1, nowX - 1, nowY);
        - » dfs(count+1, nowX, nowY + 1);
        - » dfs(count+1, nowX, nowY - 1);
        - » Forループで書ける形にするとちょっと楽。
      - この4つの結果を足し算した値をreturnする。
  - 幅優先探索などでも良いが、深さ優先探索の方が書き易い

D問題　部分点２　アルゴリズム

AtCoder

- N = 30だと？
  - 深さ優先探索を単純にするだけでは、計算が間に合わない。
    - 何か工夫をしなければならない。
  - 「上、右」と移動した後と、「右、上」と移動した後で、X,Yに辿り着く確率は同じはず。
    - こうしたものを、上手く纏めることによって、計算を早くしよう！

D問題　部分点２　アルゴリズム

AtCoder

- メモ化再帰を使った高速化
  - dfs(count, nowX, nowY)のような関数を作った。
  - N = 30の時、これの取り得る値は？
    - Count は、0～N
    - nowX は、-N～N
    - nowYは、-N～N
    - ということは、O(N^3)通りしか、パターンが存在しない！
      - よって、これらの計算結果をメモしてあげることで、高速に計算することが出来る！
      - 計算結果をまとめただけで、O(4^N)からO(N^3)まで変わる。

D問題　部分点２　アルゴリズム

AtCoder

- メモ化再帰って？
  - DFSの計算結果をメモしておく手法！
  - 下のような一文を、冒頭に入れる！
    - If(dp[count, nowX, nowY] != -1) return dp[count, nowX, nowY] ;
  - 答えを返す時に、以下のように答えをメモする！
    - ret = ….（こたえ）
    - return dp[count, nowX, nowY] = ret ;
  - これだけで、同じ引数でdfs関数が呼び出された時に、2回目からは一瞬で答えを返せるようになります。
    - 今回の場合は、nowX, nowYが負の数になるので、配列を使う場合は注意！
      - X +=N, Y+=Nとしておけば、スタート地点をN,Nにすればok

D問題　完答　アルゴリズム

AtCoder

- N≦1000だと？
  - でかい。動的計画法を使ったところで間に合わない。
  - 仕方がないので、数学的に上手く計算してあげる必要がある！
    - とはいっても、直接的に計算するのは難しい。
  - 「上か下に移動する回数」と「左か右に移動する回数」に分ければ計算できないか？

D問題　完答　アルゴリズム

AtCoder

- N = 1000, X = 10, Y = 20の時
  - 例えば、「左右に動くのは600回」とする
    - とすると、「上下に動くのは400回」とわかる
  - 左右、上下の2択に分けたとして、左右に動くのが600回な組み合わせは、$_{1000}C_{600}$通り
  - ここで、「左に動くのは295回、右に動くのは305回」と解る
    - なぜなら、600回の動きでX=10に辿り着くにはこれしかありえない
    - (600 + 10) / 2 = 305のような形で求められる。
  - この組み合わせは、$_{600}C_{295}$通り
  - 同様に、上に動く回数、下に動く回数と、その組み合わせを計算出来る
  - これらを全て掛ければ、「左右に動く回数がK回の時の、X,Yに辿り着けるパターン数」が計算できる

D問題　完答　アルゴリズム

AtCoder

- K回左右に動くときのパターン数が計算可能
  - であれば、Kを0からNまでループさせてあげれば、全てのパターン数を計算することが可能！
- これを利用すれば、解くことが出来る！

D問題　完答　アルゴリズム

AtCoder

- 注意点
  - Double型は実は10^308までしか入らない！
    - なので、「全てのパターン数」を計算するのはやめたほうが良い。
      - long doubleとか使えば大丈夫？ダメ！！！
    - 「全パターン中、この組み合わせが選ばれる確率」も計算可能なので、そちらで上手く計算しよう！
      - nCk / 2^Nみたいな感じ。
      - パスカルの三角形を使って計算するのがお勧め。

D問題　完答　アルゴリズム

AtCoder

- パスカルの三角形を使ったCombinationの計算の仕方！
  - 「1個上の数」と、「1個上、1個左の数」を足し算する
  - すると、i行目j番目の数字が、iCjになっている。
  - 1
  - 1 1
  - 1 2 1
  - 1 3 3 1
  - 1 4 6 4 1

D問題　完答　アルゴリズム

AtCoder

- パスカルの三角形を使った確率の計算の仕方！
  - 「1個上の数」と、「1個上、1個左の数」を足し算する
    - これを2で割る
  - すると、i行目j番目の数字が、iCjになっている。
  - 1
  - 0.5　　0.5
  - 0.25　　0.5　　0.25
  - 0.125　　0.375　0.375　0.125
  - ……
- N個を選ぶ全ての組み合わせ中、K個を選ぶ組み合わせの割合などを計算する時に便利！