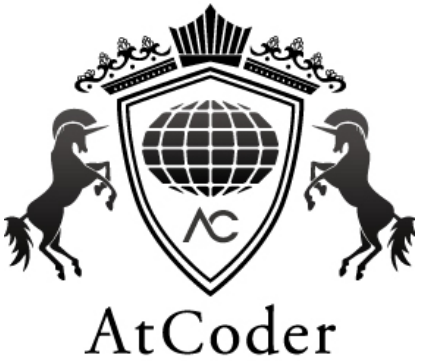
AtCoder Beginner Contest 007
解説



AtCoder株式会社

競技プログラミングを始める前に

AtCoder

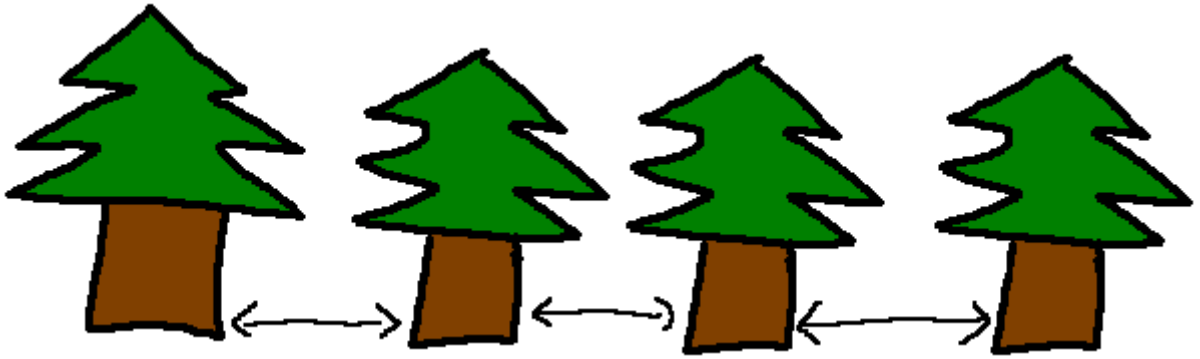
- 競技プログラミングをやったことがない人へ
  - まずはこっちのスライドを見よう！
  - 過去のコンテストのスライドですが、導入説明があります。
    - http://www.slideshare.net/chokudai/abc004

A問題　植木算

A問題　概要

AtCoder

- 一直線に木が並んでいる
- 木の本数が与えられる
- 隣り合う木々の間の数を数えなさい



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、木の本数を表す整数nを受け取る
  - 問題で与えられた処理を行う
    - 今回は、整数nに対して、木の間の数を算出する
  - 標準出力へ、答えを出力する

A問題　アルゴリズム

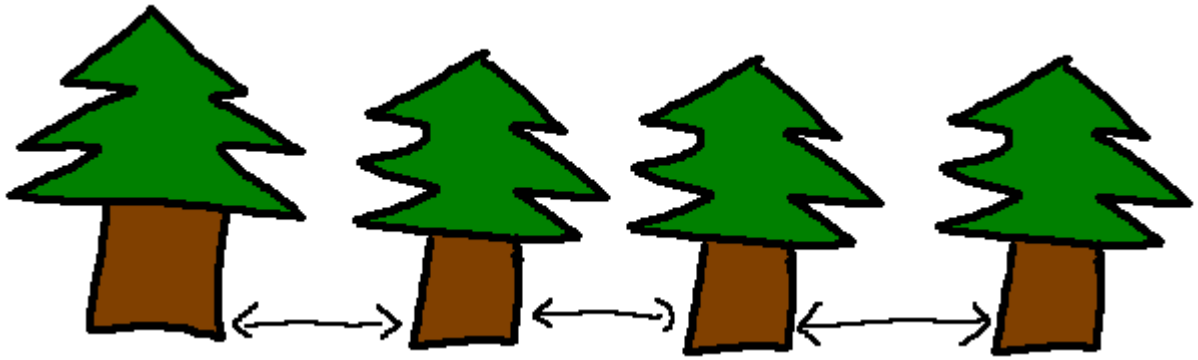
AtCoder

- 入力
  - 1つの数字を、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。
      - AtCoderで使用可能な殆どの言語に対して、入出力の方法が記述されています。http://practice.contest.atcoder.jp/tasks/practice_1

A問題　アルゴリズム

AtCoder

- アルゴリズム
  - 木がN本あるとき(N≧1)　木の間はN-1箇所ある
  - Nを読み込んでN-1を出力すると正解



A問題　アルゴリズム

AtCoder

- 具体的な処理
  - int ret = N – 1;
  - N = N-1;
  - N--;
  - --N;
  - 好きなのでいいです！
    - もちろん出力時にN-1を出力するのもOK
    - 私は入力された整数を変更したくない派なので、1番目を推奨しています。

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

AtCoder

B問題　辞書式順序

B問題　概要

AtCoder

- 英小文字から成る文字列Aが与えられる
- 辞書順比較した際にAより小さい文字列を1つ
  どれでもよいので出力、存在しない場合は-1を出力

B問題　アルゴリズム

AtCoder

- 問題の流れ
- 入力
  - 文字列Aを受け取る
- 処理
  - 文字列Aより辞書順で小さい文字列を探す
- 出力
  - 上記処理で見つけた文字列を出力する
  - もし存在しなければ-1を出力する

B問題　アルゴリズム

AtCoder

- 最も良いアルゴリズム
  - “a”が一番小さいのでどんな文字列がきても”a”を出力すればよい
  - ただしA=“a”のときはそれより小さい文字列は存在しないので”-1”を出力する。

- もちろんこれ以外にも色々ある
  - 具体例は以下の通り
    - 文字列が2文字以上の場合は、1文字減らして出力
    - 1文字の場合は全探索

C問題　幅優先探索

C問題　概要

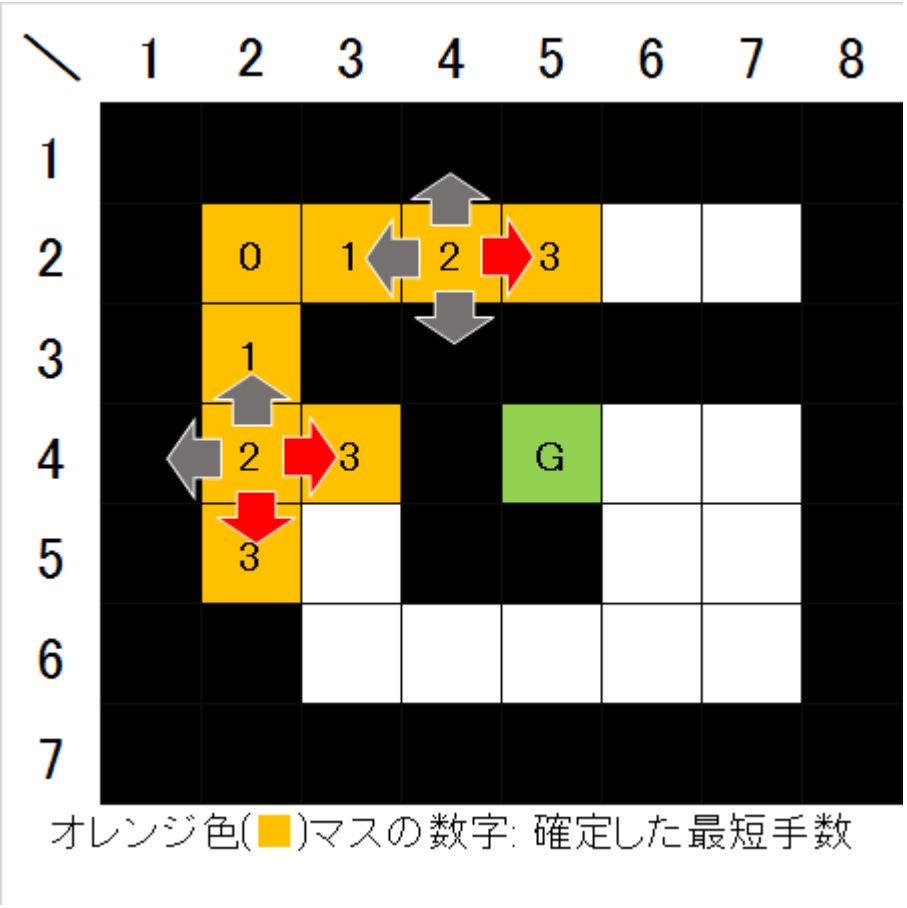
AtCoder

- 迷路とスタートとゴールが与えられる
- 上下左右に移動できるのでスタート-ゴール間の最短距離を求めよ

C問題　アルゴリズム

AtCoder

- 問題文に掛かれている通りに実装しよう！



C問題　アルゴリズム

AtCoder

- 実装Q&A
  - 上下左右に移動するのが面倒です！
    - こんな実装をしてあげよう！
      - int[] vx = { 0, 1, 0, -1};
      - Int[] vy = { 1, 0, -1, 0};
      - for(int i=0;i<4;i++) {
        - » int nx = x + vx[i];
        - » int ny = y + vy[i];
        - » …
    - このように、配列を使ってあげることによって、forループで纏めて処理することが出来ます！

C問題　アルゴリズム

AtCoder

- 実装Q&A
  - キューって何？
    - データの出し入れが出来るデータ構造です。
    - 先に入れたデータが先に出てきます。
  - 具体的にはこんな感じ
    - 1を入れる
    - 2を入れる
    - 数字を取り出す　→　1が出てくる
    - 3を入れる
    - 4を入れる
    - 数字を取り出す　→　2が出てくる
  - こんな感じで、データをどんどん入れて、取り出す処理を行うと、順番に取り出せる！

C問題　アルゴリズム

AtCoder

- 実装Q&A
  - 僕の言語にもキューって実装されてるの？
    - 大抵の言語にあります！
    - C++なら#include<queue>すれば、queueが。
    - JavaならJava.utilにQueueが。
      - Queue<Integer> = new LinkedList<Integer>();みたいになるので注意
    - C#なら、System.Collections.GenericにQueueが。
    - 他の言語にもきっとあります！
  - なかったら自分で実装しましょう！
    - 配列で出来ます！
      - 頭から順番に詰めていって、頭から取り出すだけ。
      - 長い配列と、頭の場所を表す整数、最後尾の場所を表す整数、の３つが必要
    - リストでも出来ます！

C問題　アルゴリズム

AtCoder

- 実装Q&A
  - 場所のデータをどうやってキューに入れるの？
    - 方法は3通り
    - X,Yの両方のデータが持てるデータ構造を使う
      - PairやTupleなど、複数のデータを入れられるものを使う
      - 構造体やclassを自分で作って入れる
    - 整数に押し込んでしまう。
      - X * 1000 + Yなどを入れる。
      - X = A / 1000; Y = A % 1000;などで取り出す。
        - » オーバーフローしないかどうかに注意！
        - » 速度がどうしても欲しい時や、メモリ消費量を抑えたい時用。競技プログラミング以外では、必要な時以外は使わないように！
    - キューを2個つくる！
      - X用のキューとY用のキュー

C問題　アルゴリズム

AtCoder

- 問題文に載っている幅優先探索を行えばよい
- キューを用いず、全てのマスに書かれている数字を更新操作の度に確認しても今回は間に合う
- 最短手数が1000手を超える盤面も存在することに注意せよ
- キューは自前で実装してもよいですが特にC++等は標準でqueueクラスが存在します

D問題　禁止された数字

D問題　概要

AtCoder

- AとBが与えられる
- A以上B以下で桁に4と9を含まない数字を数える

D問題　アルゴリズム(部分点 30点)

AtCoder

- 禁止された数字か判定された関数を作ることができたら愚直にB-A+1回のループを書けばよい
  - for(int i=A; i<=B; i++)みたいな感じ。
- 30点が得れる

- ある数字が含まれているかどうかは、ABC006 A問題でやったばっかり！
  - http://abc006.contest.atcoder.jp/tasks/abc006_1
  - 解説はこちらから
    - http://abc006.contest.atcoder.jp/tasks/abc006_1

D問題　アルゴリズム(4,9)

AtCoder

- 4,9を含むかどうか
  - やり方は何通りか存在する
    - nを文字列として持ち、文字4,9を含むか調べる
      - Findなどの、文字列検索を行うアルゴリズムを使う
      - Forループやforeachなどで1文字ずつ調べても良い
    - 1ケタずつ整数として調べる。
      - まず、%10を使い、1ケタ目の数字だけ取り出す
      - 次に、それが4,9であれば終了し、そうでなければ、/10して次の桁に移行する。

D問題　アルゴリズム(4,9)

AtCoder

- 1ケタずつ整数として調べる。
  - まず、%10を使い、1ケタ目の数字だけ取り出す
  - 次に、それが4か9であれば終了し、そうでなければ、/10して次の桁に移行する。

- 例えば、1342について調べる
  - 1342 % 10 = 2 ←4や9でないので、次の値は10で割って134
  - 134 % 10 = 4 ←4なので終了

D問題　アルゴリズム(満点)

AtCoder

- 動的計画法を用いる
- 0～Nに含まれる禁止された数の数をf(N)とする
- A～Bに含まれる禁止された数の数はf(B)-f(A-1)
  - 例えば、87から243を求めたいなら、0～243を求めて、0～86を引けば良い！
- f(N)を求めるためには、上位桁から、先行0を認めて各桁毎に決めていくいわゆる**桁DPが有効**
- 上位桁から決めていく際に、ある桁で一度,N未満が確定したらそれ以降はどんな値を入れても良い

  (0～Nだから)
- 状態数は (桁数) *(既に未満確定したかどうか(2通り)
- 桁数をNとして、O(N)

D問題　アルゴリズム(満点)　つづき

AtCoder

- 動的計画法を用いる
- たとえばF(1234)を求めるとき、

---- → 状態: (0桁目まで確定・未満フラグ: false)
1--- → 状態: (1桁目まで確定・未満フラグ: false)
12-- → 状態: (2桁目まで確定・未満フラグ: false)
10-- → 状態: (2桁目まで確定・未満フラグ: true)
11-- → 状態: (2桁目まで確定・未満フラグ: true)
108-→ 状態: (3桁目まで確定・未満フラグ: true)
101-→ 状態: (3桁目まで確定・未満フラグ: true)

こう言う感じの状態を考える

D問題　アルゴリズム(満点)　つづき

AtCoder

- F(1234)について考える

10--

11--

この二つは、2ケタ目まで確定した時点で、同じ扱いが出来る！！

  - 両方とも、それ以降の数字は、何を入れても良い。
- しかし、12—は、同じ扱いが出来ない！
  - 34以下を残りに入れないといけない。
- これを上手いこと纏めて、処理してあげる必要があります！

D問題　アルゴリズム(満点)　つづき

AtCoder

- 10—や、11—みたいな、「２桁目まで決まっていて」「小さいことが確定している」状態を、dp[1][1]のように表す。(0からなので、2ケタ目は1で表す)
- 12—のような、「２桁目まで決まっていて」「小さいことが確定していない」状態をdp[1][0]のように表す。

- このように、「何桁目まで確定しているか」「N未満であることが確定しているかどうか」の2つの状態を使って纏めてあげることによって、簡単に計算することが出来る！

D問題　アルゴリズム(満点)　つづき

AtCoder

- その他の解法
  - 8進数を使った解法について
    - 例えば、195435までの数字に対して、禁止されていない数を数えたい
      - 要するにf(195435)
    - 今回のルールで使える数字は、0,1,2,3,5,6,7,8の8通りしか存在しない！
      - 0->0, 1->1, 2->2, 3->3, 5->4, 6->5, 7->6, 8->7に変換するようにして、8進数として扱えば、1から変換後の数字までの数がいくつあるか数えるだけで良い！
      - じゃあ、4と9の場合はどうするの？
        - » F(195435)というのは、F(188888)と同じ！
        - » なぜなら、195435から188889は、全て9が含まれている！
        - » ということで、4や9を見つけたら、下の桁を全部8で埋めれば良い！
        - » F(12345678) = F(12338888)みたいになります！
        - » この処理を行った後に、先ほどの変換を行い、8進数から10進数にすれば良い！