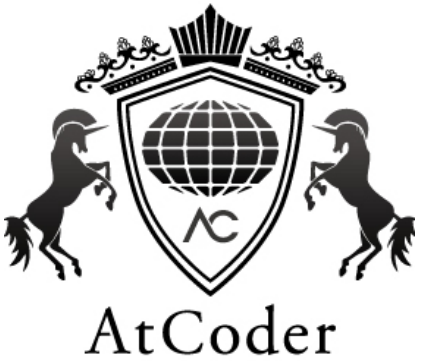
AtCoder Beginner Contest 026
解説



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　掛け算の最大値

A問題　問題概要

AtCoder

- 偶数Aが与えられる。
- X + Y = AとなるX,Yのうち、X × Yが最大になるものを選び、その値を出力せよ

- 制約
- 2 ≦ A ≦ 100

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、A という1つの整数
  - 問題で与えられた処理を行う
    - 今回は、X*Yの最大化
  - 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つの文字列を、標準入力から受け取る
    - Cであれば、scanf(“%d”, &A);　など
    - C++であれば、cin >> A;
    - 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

- 今回の問題は、X*Yの最大値を出力する
- X*Yが最大となるX,Yってどう求めればいいの？
  - よくわからない場合は全通り調べれば良い！
  - ループを回して、全通り調べましょう。

  - 例えばこんな感じ

```
for(int X = 0; X <= A; X++){
        int Y = A – X:
        ans = max(ans, X * Y);
}
```

A問題　アルゴリズム

AtCoder

- 実は、相加相乗平均より、X = Y = A / 2の時、X*Yが最大となることがわかる
- それが解っていれば、その1通りだけ調べれば良い

```
int X = A / 2;
int ans = X * X;
```

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”, ans); (C)
    - cout << ans << endl; (C++)
    - System.out.println(ans); (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　N重丸

1. 問題概要
2. アルゴリズム

B問題　問題概要

AtCoder

- N個の円が与えられる。これらの円はすべて中心が原点であり、半径が違う
- 外側から赤白交互に色を塗っていく
- 赤く塗られた部分の面積を求めなさい

- 制約
- $1 \leqq N \leqq 1{,}000$
- $1 \leqq Ri \leqq 1{,}000$

B問題　アルゴリズム

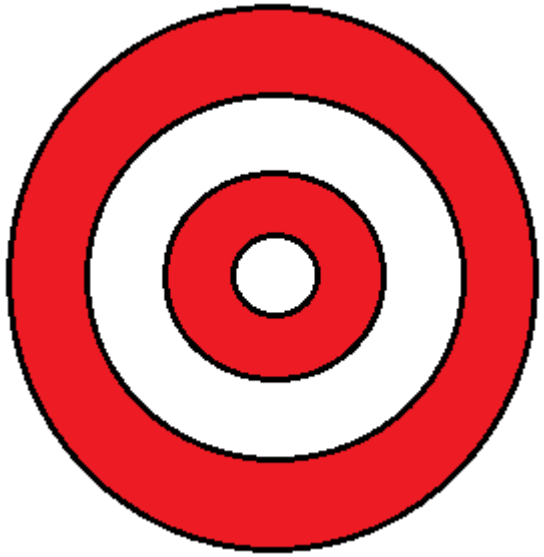
AtCoder

- 入力
  - 整数Nを受け取る
  - 各円の半径Riをn個受け取る
    - 今回はスペース区切りではなく、改行区切りなので、何度も受け取るだけ
    - 詳しくは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

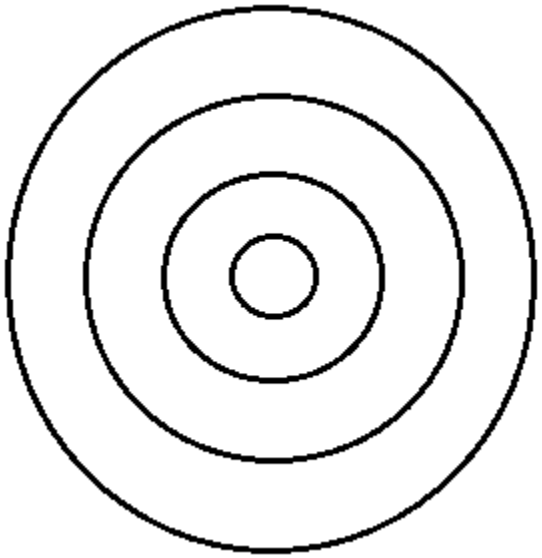
- 処理
  - 色を塗られた部分の面積を求めなければならない



B問題　アルゴリズム

AtCoder

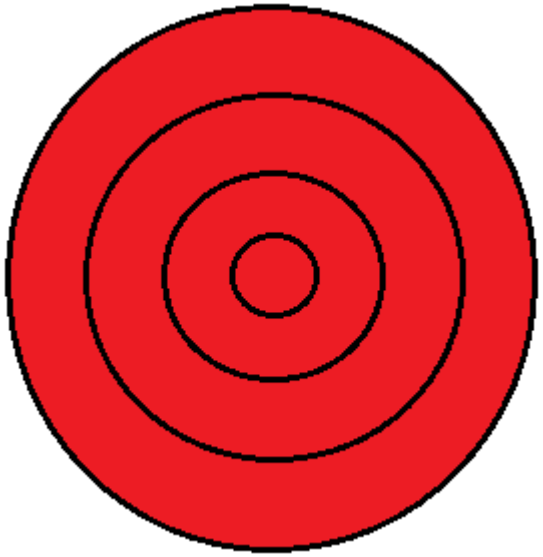
- 処理
  - 色を塗られた部分の面積を求めなければならない
  - まず、最初は何も塗られていない状態



B問題　アルゴリズム

AtCoder

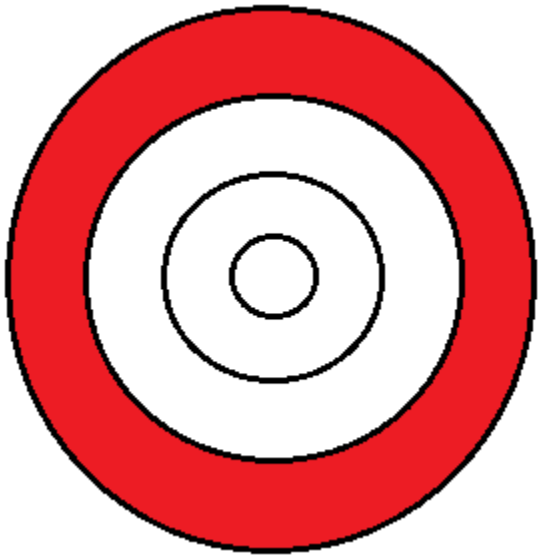
- 処理
  - 色を塗られた部分の面積を求めなければならない
  - まず、最初は何も塗られていない状態
  - そこから、まず一番外側の円を赤く塗る



B問題　アルゴリズム

AtCoder

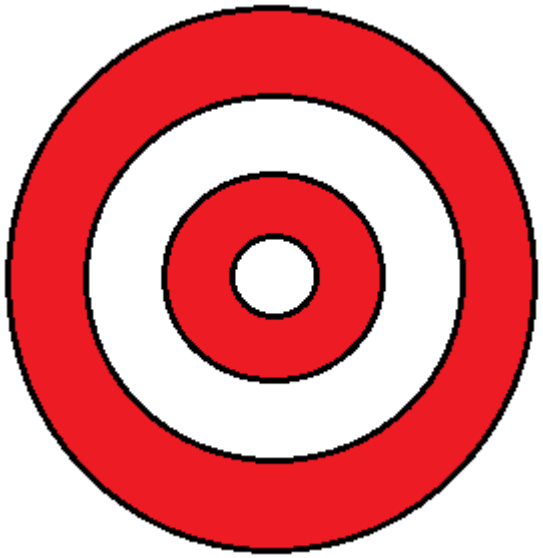
- 処理
  - 色を塗られた部分の面積を求めなければならない
  - そこから、まず一番外側の円を赤く塗る
  - 次に、2番目の円を白く塗る



B問題　アルゴリズム

AtCoder

- 処理
  - 色を塗られた部分の面積を求めなければならない
  - そこから、まず一番外側の円を赤く塗る
  - 次に、2番目の円を白く塗る
  - これを繰り返すことで、全ての色を塗ることが出来る



B問題　アルゴリズム

AtCoder

- 処理
  - 色を塗られた部分の面積を求めなければならない
  - そこから、まず一番外側の円を赤く塗る
  - 次に、2番目の円を白く塗る
  - これを繰り返すことで、全ての色を塗ることが出来る

  - では、どう計算すればいいか？
    - 赤く塗る、の部分で、塗った面積を足す
    - 白く塗る、の部分で、塗った面積を引く
  - 上記を繰り返すことにより、赤い部分の面積を求めることが出来る

B問題　アルゴリズム

AtCoder

- 解法まとめ
  - まずはRiの値を大きい順にソートする
    - ソートアルゴリズムは、大体の言語で標準で実装されている
    - 普通は昇順（小さい順）にソートされるので注意
      - 小さい順に処理をすると混乱しやすいが、別にできなくはない
  - 次に、外側の円から順番に、面積を計算していく
    - 赤・白が交互に出てくることに注意
    - 円の面積は、(半径)×(半径)×(円周率)
    - 円周率は、3.14じゃダメ！標準で入っている言語が多いです。
      - 入っていない場合は、調べて十分な精度までコードに埋め込みましょう。
  - 最後に回答を出力

B問題　アルゴリズム

AtCoder

- 出力
  - 今回は、小数を出力する必要がある
  - 小数の出力方法も、言語によって違う
    - 練習ページには書いていないので、ほかの人の提出を見よう！
  - 普通に出力すると、様々な問題があります。
    - 桁数が大きいときに、指数表示で出力されてしまう
    - 値が小さい時に、十分な精度の出力が行われない
  - 以上のようなことを避けるため、出力フォーマットを明記しましょう。
    - C/C++なら、printf(“%.14f¥n”, ans);など

C問題　高橋君の給料

1. 問題概要
2. アルゴリズム

C問題　問題概要

AtCoder

- 社員がN人いる
- 社長以外の社員は、上司を一人だけ持っている
- 社員の給料は、直属の部下の給料の最大値・最小値から算出できる
- 社長の給料を求めよ

- 制約
- $1 \leqq N \leqq 20$

C問題　アルゴリズム

AtCoder

- やるべきことは、問題文に書かれている処理を忠実に書くだけ
  - その書き方が難しい！
- どうして難しいか？
  - 自分の給料を決めるのは、「部下の給料」を知る必要があるのに、「部下の一覧」は入力で与えられない
    - 「部下の一覧」ではなく「上司」が与えられる
  - 高橋君の給料を求めようと思っても、まず部下の給料を求めないといけない！
    - 計算順序とかをちゃんと考えないといけない
- 部下がいない時だけ計算式が違う

C問題　アルゴリズム

AtCoder

- 気付くべきこと
  - 上司の番号は、必ず自分の番号より小さい
    - つまり、部下の番号は、必ず自分の番号より大きい
    - 自分より大きい社員番号の社員を、全部調べることにより、自分の給料を算出できる
    - つまり、社員番号の大きい社員から順番に、給料を求めれば良い
  - 上司の番号が与えられているなら、それを適当に処理すれば、部下の一覧も作れる
    - ListやVectorなどの配列を使うことで、適当に対応できる！

C問題　アルゴリズム

AtCoder

- 実装方法
  - 実装１：ループによる実装（もらう型）
  - 実装２：ループによる実装（配る側）
  - 実装３：再帰による実装

C問題　アルゴリズム

AtCoder

- 実装１：ループによる実装（もらう型）
  - 予め部下リストを作る
  - 逆順に、部下リストから最大値と最小値を求め、給料を求めていく

```
for(int i = N - 1; i >= 0; i--){
    if(sub[i].size()==0){P[i] = 1; continue;}
    maxP[i] = 0; minP[i] = (int)1e9;
    for(int j: sub[i]){
        maxP[i] = max(maxP[i], P[j]);
        minP[i] = min(minP[i], P[j]);
    }
    P[i] = maxP[i] + minP[i] + 1;
}
```

C問題　アルゴリズム

AtCoder

- 実装２：ループによる実装（配る型）
  - 逆順に処理していく
  - 上司の最大値と最小値を更新し、、給料を求めていく

```
for(int i = 0; i < N; i++){maxP[i] = 0; minP[i] = (int)1e9;}
for(int i = N - 1; i >= 0; i--){
    if(maxP[i]==0) P[i] = 1;
    else P[i] = maxP[i] + minP[i] + 1;
    maxP[boss[i]] = max(maxP[boss[i]], P[j]);
    minP[boss[i]] = min(minP[boss[i]], P[j]);
}
```

C問題　アルゴリズム

AtCoder

- 再帰による実装
  - 部下リストを予め作る
  - 再帰で更新する

```
int dfs(int id){
    if(sub[id].size() == 0) return 1;
    int minP = (int)1e9; int maxP = 0;
    for(int i: sub[id]){
        int P = dfs(i);
        minP = min(minP, P);
        maxP = max(maxP, P);
    }
}
```

C問題　アルゴリズム

AtCoder

- おまけ　部下リストの作り方
  - boss[i]が、iの上司がboss[i]であることを示す
  - 上司の部下リストに自分を追加する、ということを繰り返すことで作れる
  - 擬似コードは↓のような感じ

```
List<int> sub[N];
for(int i=0;i<N;i++){
        sub[boss[i]].add(i);
}
```

C問題　アルゴリズム

AtCoder

- 考察
  - 「答えが非常に大きくなる」というが、どれくらいか？
    - 1の部下が2, 2の部下が3…というような構造になってる時が、最も大きくなる
    - この時、2^n – 1が高橋君の給料になる。
      - 今回の問題だと、1048575とかなので、int型で十分
  - 計算量も気にしないで良い
    - 紹介したどの実装でもO(N)、かつNが20以下なので余裕
    - 1億を超えるようなときだけ気を付けよう！

D問題　高橋君ボール1号

1. 問題概要
2. アルゴリズム

D問題　問題概要

AtCoder

- 高橋君が、高橋君ボールを投げる
- 投げた時間に対する距離f(t)は、以下の数式で表せる
  - $f(t) = At + B\sin(Ct\pi)$
- f(t) = 100となるtを1つ求めよ

- 制約
  - $1 \leqq A,B,C \leqq 100$

D問題　アルゴリズム

AtCoder

- そもそもどんな関数なのかな？
  - 二つに分けて考えよう！
    - At
      - tに対してAだけ増加する直線
    - Bsin(Ctπ)
      - 周期2/CのsinカーブをB倍したもの
  - つまり、「直線」と「sinカーブ」を足し合わせた関数
  - f(t)=100を、数学的に直接求めようとするのは難しそう！
    - であれば、何か工夫して求めてみよう！

D問題　アルゴリズム

AtCoder

- 関数のイメージ
  - 大体こんな感じ！
    - うねうねしながら増えてく

D問題　アルゴリズム

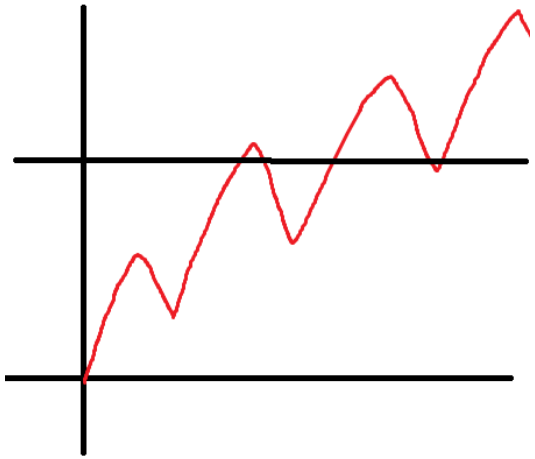
AtCoder

- 直接答えを求められないときはどうすれば良いか？
  - 二分探索を使えば良い！
    - f(t)のtに、適当な値を入れる
      - f(t)が100より大きければtはそれより小さく、100より小さければ、tはそれより大きいことを利用する
    - 具体的なアルゴリズム
      - 答えがt=0からt=10000の間にあるとする
      - ちょうど真ん中のt=5000を試す
      - f(t)<100なら、答えは5000から10000の間にあり、そうでないなら、0から5000の間を調べれば良い
      - これを繰り返し、十分な精度になるまで答えを半分にしていく
    - これで本当に良い？
      - 関数が単調増加な関数でないので、これでは、f(t)=100のtの最小値などは求められない
      - しかし、f(t)=100となる、tの1つを求めるだけなら、これで十分！

D問題　アルゴリズム

AtCoder

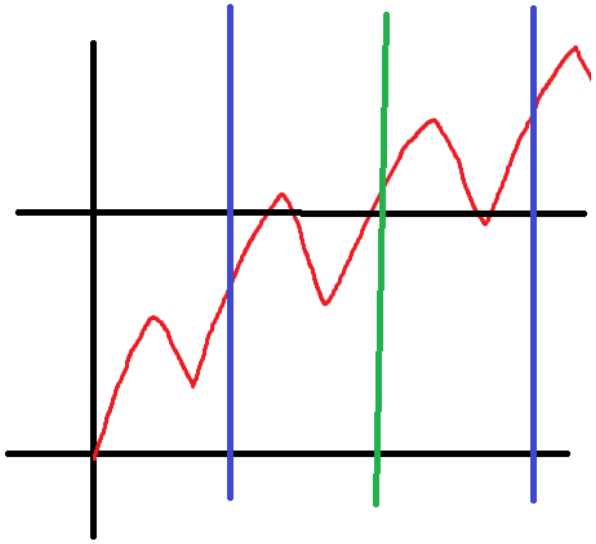
- なぜ二分探索で答えが求められるのか？
  - 例えば、f(t)=100がこの黒い線だったとする
    - 解となるtは、最低1つはあり、図のようにたくさんある場合もある



D問題　アルゴリズム

AtCoder

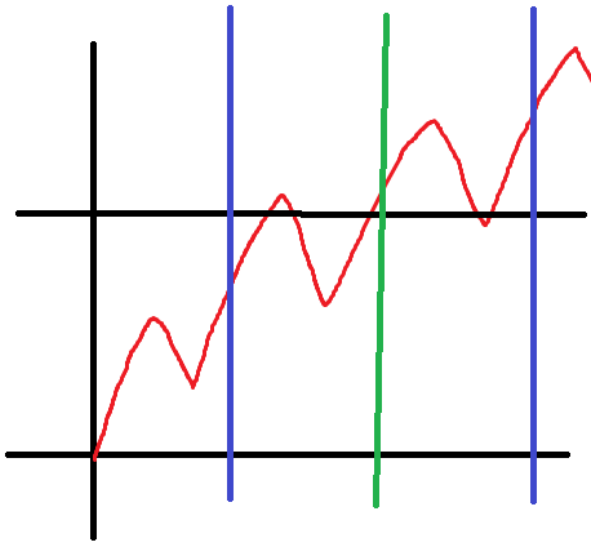
- なぜ二分探索で答えが求められるのか？
  - 例えば、f(t)=100がこの黒い線だったとする
    - 解となるtは、最低1つはあり、図のようにたくさんある場合もある
  - 今、青い線の間に答えがある、というところまで絞れており、緑の線の間に答えがあることが解っているとする
    - どちらを取っても解の取りこぼしが起こる？
  - 取りこぼしは発生するが、
    左の範囲を選択した時に、
    絶対に解の1つが間にある事が
    保証される！



D問題　アルゴリズム

AtCoder

- なぜ解が必ずあるか？
  - この関数f(t)は、tに対して連続である
  - f(左青)≦100≦f(緑)の時、f(t)は連続であるので、左青と緑の間に、必ずf(t)=100となるtが存在する！
    - 平均値の定理



D問題　アルゴリズム

AtCoder

- 擬似コードはおおよそこんな感じ
  - t≧200であれば、f(t)≧100は簡単に示せる

```
double low = 0; high = 200;
for(int i=0;i<100;i++){
        double mid = (low+high)/2;
        if(f(t)<100) low = mid;
        else high = mid;
}
print(low);
```

D問題　アルゴリズム

AtCoder

- 注意点
  - 誤差が非常に厳しいので気を付けましょう！
    - f(t)の誤差が10^-6まで許される
      - tが10^-9程度変わると、f(t)は10^5くらい変わったりする
      - よって、10^-11程度の精度はあった方が安心
      - 二分探索の打ち切り条件に気を付けよう！
      - 出力桁数にも注意しよう！