tags: `AtCoder` ログ 解説記事 競プロ

ARC099(ABC101) 振り返り

ARC094 (https://beta.atcoder.jp/contests/arc094/tasks)以来の悪夢再来。

- ABCリンク (https://beta.atcoder.jp/contests/abc101/tasks)
- ARCリンク (https://beta.atcoder.jp/contests/arc099/tasks)

A - Eating Symbols Easy (記号を食べる高橋君: Easy)

問題リンク (https://beta.atcoder.jp/contests/abc101/tasks/abc101_a)

概要

+ と - からなる文字列Sが与えられる。0に対して、Sの頭から順に

- 文字が + なら1増やす
- 文字が - なら1減らす

という操作を行ったとき、最終的に得られる値を求めよ。

制約

- $|S| = 4$

解説

実際にSを頭から見ていって操作を行えばいい。文字列の処理方法に注意ってところだろうか。

ちなみに操作自体は順不同なので、別にどこから見ていっても結果は変わらない(やる意味はないと思うが)。

回答コード

```
 1  using namespace std;
 2
 3  /* ---------- ここまでマクロ ----------*/
 4
 5  int main() {
 6      string S;
 7      cin >> S;
 8
 9      int ans = 0;
10      // この値に記号を作用させていく
11
12      // Sを頭から順に見ていく
13      // 記法については後ほど
14      for (char c : S) {
15          if (c == '+') {
16              ans++;
17          } else {
18              ans--;
19          }
20      }
21
22      cout << ans << endl;
23      return 0;
24  }
```

日頃から使ってない人には14行目の

```
for (char c : S) {
    ...
}
```

という記法は見慣れないかもしれない。これは「**範囲for文**」と呼ばれていて、本質的には

```
for (int i = 0; i < S.size(); i++) {
    char c = S[i];
    ...
}
```

とやってることはほとんど同じ。PythonやRuby経験者なら慣れていると思うが、C++だけって方の中には今まで知らなかったという方もいるかもしれない[1]。mapの要素を全列挙したいときとかに便利なので、身につけておくべし。

B - Digit Sums (桁和)

問題リンク (https://beta.atcoder.jp/contests/abc101/tasks/abc101_b)

概要

自然数Nが与えられる。Nを10進法表記したときの各桁の和を$S(N)$としたとき、Nが$S(N)$で割り切れるか判定せよ。

制約

- $1 \leq N \leq 10^9$

解説

実際に$S(N)$を計算して、割り切れるかどうか判定するのが早いし色々考えなくていいので楽。普通に実装すれば$O(\log N)$で行けるはず。

回答例

```
using namespace std;

/* ---------- ここまでマクロ ----------*/

// digit sum: Nの桁和を返す
int dsum(int N) {
    int ret = 0;

    while (N > 0) {
        ret += N % 10;
        N /= 10;
        // 1の位をretに足してNを10で割る、を繰り返すだけ
    }

    return ret;
}

int main() {
    int N;
    cin >> N;

    int d = dsum(N);
    if (N % d == 0) {
        cout << "Yes" << endl;
    } else {
        cout << "No" << endl;
    }
    return 0;
}
```

桁和を求める問題は競プロでも非常によく出るので、スラスラ書けるようにしておきたいところ。以下のように再帰関数で書くともっと楽。

```
int dsum(int N) {
    return N == 0 ? 0 : N % 10 + dsum(N / 10);
}
```

再帰関数は上のようにとても短く書けるのが魅力的だが、

- 慣れていないと読みづらい
- 終端判定を間違えやすい
- 裏で尋常じゃない呼び出し量になってしまうことがある(Fibonacci数列とか)

のが短所でもある。逆にこれらさえ気をつければとても有用な武器となりうる。

ちなみにこの問題は後のD問題の伏線的な問題でもあるのだが、それはまた後ほど。

C - Minimization (最小化)

問題リンク (https://beta.atcoder.jp/contests/arc099/tasks/arc099_a)

概要

数列$(1, 2, \ldots, N)$**を並び替えた**長さNの数列A_iがある。
この数列に対して以下の操作を何度か行うことで、全ての要素を同じにしたい。

> 連続するK個の要素$A_i, A_{i+1}, \ldots A_{i+K-1}$を選ぶ。これら全てを、その中の最小値で置き換える。

このとき、必要な操作の最小回数を求めよ。

制約

- $2 \leq K \leq N \leq 10^5$

解説

概要でも太字にしたが、数列A_iが数列$(1, 2, \ldots, N)$を並び替えたものであるということを見逃すとどツボにハマる。というのも、この条件から**全要素の最終的な値が1である**ことが決まるからだ。これは、1という要素はどのような操作を行っても消すことができないことから明らかである。

では実際にどういった操作をすればいいのか。結論を言ってしまうと、操作をした区間全体の集合は、

- 区間の和は途中で途切れてはならない

- 区間の和は数列全体を覆っていなくてはならない

という2つを満たしていなければならない。

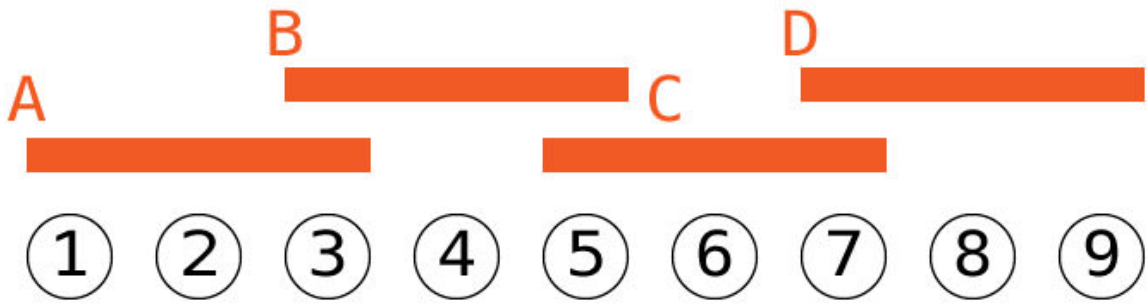
これは、上の操作を「区間全体に1を広げる操作」と考えると直感的にもわかりやすいだろう。各条件が満たされていないと、

- 区間が区切れているところから先に1を広げることができない
- 覆われていない要素に1を広げることはできない

ことになり、条件を達成することができない。

逆に、上の2条件さえ満たされていれば適用順序を工夫することで必ず全要素を1にできる。具体的には、1が含まれている区間から順番に操作していけばいい。

抽象的すぎて分からないという方は、以下の図が参考になれば幸いである。



N = 9, K = 3 の場合

$A_2 = 1 \rightarrow$ A, B, C, D

$A_5 = 1 \rightarrow$ C, B, A, D

$A_8 = 1 \rightarrow$ D, C, B, A

では実際に答えを求めよう。区間の数=答えだから、区間同士の重なりは1にするのが最も効率的である。このとき、m個の区間が覆える範囲は$[1, (K-1)m+1]$となる。これが区間$[1, N]$を覆っていればいいわけだから、答えは

$$(K-1)m+1 \geq N \quad \Leftrightarrow m \geq \frac{N-1}{K-1}$$

を満たす最小のmとなる。これは、左辺の分数を切り上げることで計算できる。

回答例

```
using namespace std;

/* ---------- ここまでマクロ ----------*/

int main() {
    ll N, K;
    cin >> N >> K;

    ll ans = (N - 2) / (K - 1) + 1;
    // a/b の切り上げは (a-1)/b+1 で求まる
    cout << ans << endl;
    return 0;
}
```

切り上げ処理に関してはコード中に書いた通り。覚えておくと楽なのだろうが、私はいつも導出している。

そして今まで見てきた通り、実は答えはNとKのみに依存していて**具体的な数列の要素は一切不要**である。こういう場合、上のように標準入力を受け取らずにコードを終了しても問題はない。

D - Snuke Numbers (すぬけ数)

問題リンク (https://beta.atcoder.jp/contests/arc099/tasks/arc099_b)

概要

正整数nに対して、nを10進法表記したときの各桁の和を$S(n)$とし、関数gを$g(n) = \dfrac{n}{S(n)}$と定義する。

そして任意の正整数$m > n$に対して$g(n) \leq g(m)$が成立するようなnを「すぬけ数」と呼ぶことにする。

このとき、すぬけ数を小さい方から順にK個出力せよ。

制約

- $K \geq 1$
- K番目のすぬけ数は10^{15}以下である。

解説

ちょっと制約が変わっているが、これはすぬけ数が具体的にどれくらいの分布になっているかを悟られないようにするためのものだろう。

この問題の方針だが、大きく分けて2つある。1つはちゃんと計算してすぬけ数を順番に求める、言わば「正統派」。もう1つはある程度すぬけ数を求めるプログラムを作ってそこから法則を強引に見出す、言わば「エスパー派」である。

ここでは前者の方針をEditorialにしたがって解説するが、すぬけ数は列挙してみると結構面白い規則性があるらしいので調べてみるといいだろう。

方針

この問題でまず躓くのが、最初にどう一歩を踏み出すかというところである。

結論を言ってしまうと、今回の場合は以下のように**漸化的に**すぬけ数を求めるのがベターな方針らしい。

- $snuke(n)$ =「n以上で最小のすぬけ数」という関数$snuke$を作る、
- 1はすぬけ数なのでまず$n = 1$とする。
- $n \leftarrow snuke(n+1)$として、nを更新する。

言われてみればそりゃそうだって感じだが、言われなければ踏み出せない。こういった方針立ても経験からくるものなのだろう。

というわけで次の課題は、上でサラッと出てきた関数$snuke(n)$を作るということになる。

問題の有限化

すぬけ数の定義には「任意の正整数」が使われているが、これをプログラミングで扱うことはもちろんできない。
そこで、$snuke(n)$の**上限**というのを考えてみる。

mがある程度大きいとき、$100m$付近におけるgの最小値はm付近に比べて十分大きいことはなんとなく分かるだろうか。mの桁数をdとすると、100倍したところで桁和は高々20程度しか増えないので、分母のように100倍になるということはまずない[2]。

というわけで、具体的に$m \in [n, 100n]$の範囲で$g(m)$が最小となるmを求めればすぬけ数が求まることになる。随分大雑把な議論だが、実際これで正しく求まる。

この「**問題の有限化**」がこの問題を解く上での1つの鍵である。これによって、「任意の$m' > m \geq n$に対して$g(m) \leq g(m')$」というプログラムでは到底扱えない性質が、まだ範囲こそ大きいが「$m \in [n, 100n]$で$g(m)$が最小」というプログラムでも力技で扱える性質に帰着されたことになる。

ふるいにかける

上の議論によって「時間を気にしなければ一応は解ける」という状態にはなったが、まだ$g(m)$を求める対象が多すぎてとてもじゃないが現実的ではない。

そこで、すぬけ数となりうる値をある程度絞り込む必要がある。言うなれば調査対象を「ふるいにかける」というわけだ。

例えば、$n=1000$のとき$snuke(1000)=1234$となりうるだろうか?答えはNOである。これはなぜかというと、例えば1199という値は$1199<1234$かつ$S(1199)>S(1234)$を満たすため明らかに$g(1199)<g(1234)$を満たすからだ[3]。

このように、大半の値は「小さい方の桁は全部9」という値が言わば上位互換として存在する。逆に、この性質を満たす値には自明な上位互換というのは存在しないため調べる必要がある。

具体例として、$n=1234$のときの調査対象を列挙すると以下のようになる。

4桁	3桁	2桁	1桁	0桁
1234	1235	1249	1399	2999
-	1236	1259	1499	3999
-	…	…	…	…
-	1239	1299	1999	9999
-	-	-	-	10999
-	-	-	-	11999
-	-	-	-	…
-	-	-	-	122999

1行目はnとの一致桁数である。0桁が分かりにくいかもしれないが、注釈[2:1]でも述べた通り相当過剰に取ってるのでそこまで細かい値を気にする必要はない。一致する桁以降は全部9というのが一番重要である。

そして上のリストに入る値は$n\leq 10^{15}$で300個にも満たないため、十分$g(m)$を調べることが可能である。以上でようやく準備が整った。

回答例

```
using namespace std;

template <typename T, typename U>
using P = pair<T, U>;
template <typename T>
using GPQ = priority_queue<T, vector<T>, greater<T>>;

using ll = long long;

#define DOUBLE(n) static_cast<double>(n)

#define FOR(i, a, b) for (ll i = (a); i < (b); i++)
#define REP(i, n) FOR(i, 0, n)
#define NREP(i, n) FOR(i, 1, n + 1)
// Usual REP runs from 0 to n-1 (R: n-1 to 0)
// Natural REP runs from 1 to n (R: n to 1)

/* ---------- ここまでマクロ ----------*/

// bのn乗を再帰的に求める
ll mypow(ll b, ll n) {
    return n == 0 ? 1 : b * mypow(b, n - 1);
}

// 桁和を求める(B問題を参照)
ll dsum(ll n) {
    ll ret = 0;
    while (n > 0) {
        ret += n % 10;
        n /= 10;
    }
    return ret;
}

// n以上で最小のすぬけ数を求める
ll snuke(ll n) {
    GPQ<P<double, ll>> val;
    // 逆順priority_queue, 一番上が最小の要素
    // (g(m), m)

    ll dig = 0;
    // 9が連続する桁数

    while (mypow(10, dig) <= n) {
        NREP(i, 9) {
            ll v = n - n % mypow(10, dig + 1) + i * mypow(10, dig) - 1;
            // 下dig桁が9
            // 1つ上は1~9
            // それより上はnと一致

            if (v >= n) {
                val.push(make_pair(DOUBLE(v) / dsum(v), v));
            }
        }
```

```
        dig++;
    }

    dig--;

    // 1桁もnと一致しないものを列挙
    NREP(i, 100) {
        ll v = i * mypow(10, dig) - 1;
        if (v >= n) {
            val.push(make_pair(DOUBLE(v) / dsum(v), v))
        }
    }

    auto ret = val.top();
    // g(m)とmが最小となる(g(m), m)の組

    return ret.second;
}


int main() {
    ll K;
    cin >> K;
    ll pre = 1;
    REP(_, K) {
        cout << pre << endl;
        pre = snuke(pre + 1);
    }
    return 0;
}
```

上のコードでは `snuke` の実装において、nと1桁も一致しない値の扱いが雑になっている。実際に計算すれば分かるのだが、このエリアの値が返り値になるのは$n = 99\ldots9$の場合くらいなもので、わりかし雑に扱っても答えは出せる。

あと調査対象の列挙方法も少しトリッキーではある。

```
ll v = n - n % mypow(10, dig + 1) + i * mypow(10, dig) - 1;
```

これは前半の `n - n % mypow(10, dig + 1)` でnの上位桁だけを取り出して、後半の `i * mypow(10, dig) - 1` で9が連なる値を加えている。正直自分でも気持ち悪いが、他にいい方法が浮かばなかった。

点数について

ちなみにこのD問題、500点である。「700点くらいない？」って感じだが、本来500点というのはそういうもんなんだろうか。「Two Sequences (https://beta.atcoder.jp/contests/abc091/tasks/arc092_b)」も500点だし。う〜ん、よく分からない。

しかしABCでRatedが21人通しているのがすごい。君たちもうARC行っていいよ。

感想

やはりDに尽きる。どうすりゃええねんこんなん。

ちなみに終わったあとでE問題も見たが、多分時間内に解くのは無理だっただろう。完全グラフの頂点同士は補グラフでは非接続(直接繋ぐ辺が存在しない)ってのが肝らしい。どっから補グラフ出てきた[4]。

まぁ全ては精進不足が元凶なので、経験を積むより他にない。もっと真面目な人間に育ちたかったと怠惰に暮らしながら常々思う。

1. 私はC++始めて半年くらいで初めて知りました。ググりにくいので意外と知る機会が少ない。 ↩

2. 厳密には100倍でも大きく取り過ぎで、実際は10倍くらいで十分である。ただ個人的に10倍では不安があったので、ここでは以降も100倍の範囲を取り続けることにする。 ↩ ↩

3. ここで違和感を覚える方は、問題の有限化によって「$snuke(n) =$『$m \in [n, 100n]$で$g(m)$が最小となるようなm』」と定義が変わっていることに注意していただくといいだろう。 ↩

4. とか思ってたら最近のこどふぉで似た傾向の問題 (http://codeforces.com/contest/990/problem/D)に遭遇してびっくり。 ↩