三井住友信託銀行プログラミングコンテスト 解説

E869120, square1001

2019 年 12 月 1 日

For International Readers: English editorial will be published in a few days.

A: November 30

この問題では、4 つの値 M_1, D_1, M_2, D_2 を入力します。そこで、一部のみなさんは 「M_1, D_1 だけで良いのでは？」 と思うでしょう。しかし、実は M_2, D_2 を利用した解法など、一つの簡単な問題にもいろいろな解法のバリエーションがあります。

以下、この問題の 3 通りの解法を書いていきます。

★ 解法 1 - M_1, D_1 だけを使って “普通に” 判定

2019 年にはうるう年がないので、「月末日」 であるかは以下のように判定できます。

- $M_1 = 1, 3, 5, 7, 8, 10, 12$ のとき：$D_1 = 31$ だと 「月末日」、それ以外だと 「月末日でない」
- $M_1 = 4, 6, 9, 11$ のとき：$D_1 = 30$ だと 「月末日」、それ以外だと 「月末日でない」
- $M_1 = 2$ のとき：$D_1 = 28$ だと 「月末日」、それ以外だと 「月末日でない」

あとは、条件分岐 (多くのプログラミング言語では if-else 文) を使うことで実装ができます。普通に実装すると 2 重に if-else 文を書くことになりますが、出力の部分に 「3 項演算子」 (知らない人は調べてみましょう) を使うことで実装が簡略化できます。

＜実装例 (Python 3) ＞ https://atcoder.jp/contests/sumitrust2019/submissions/8712722

★ 解法 2 「2 つの月が違えば月末日」

例えば、11 月 30 日の次は 12 月 1 日です。このように、「月末日」 の次の日は月が替わっています！そのように、$M_1 \neq M_2$ であれば 「月末日」、それ以外だと 「月末日でない」 と判定できます。

＜実装例 (Python 3) ＞ https://atcoder.jp/contests/sumitrust2019/submissions/8712762

★ 解法 3 - 「月末日の次の日は “X 月 1 日”」

11 月 30 日の次が 12 月 “1 日” であるように、「月末日」 の次の日は 「月のスタート」、つまり X 月 1 日のようになっています。よって、$D_2 = 1$ であれば 「月末日」、それ以外だと 「月末日でない」 と判定できます。

＜実装例 (Python 3) ＞ https://atcoder.jp/contests/sumitrust2019/submissions/8712801

さて、あなたが一番気に入った解法はどれでしょうか？

B: Tax Rate

現実社会でも、税込み価格だけが分かっている時に、税抜き価格も知りたいと思ったことがある、という人も多いと思います。

この問題では主に二つの方針があります。一つは 「全探索で答えを見つける」、もう一つは 「計算で答えを決め打ちする」 方法です。それぞれについて説明していきたいと思います。

★ 解法 1 - 「全探索で答えを見つける」

アップルパイの税抜き価格を X として、「$X \times 1.08$ を整数に切り捨てた値」 が N であるようなものを見つければ良いです。税抜き価格が N 以下なのは明らかなので：

- 税抜き価格は 1 でしょうか？ (つまり、1×1.08 を整数に切り捨てた値は N か？)
- 税抜き価格は 2 でしょうか？ (つまり、2×1.08 を整数に切り捨てた値は N か？)
- 税抜き価格は 3 でしょうか？ (つまり、3×1.08 を整数に切り捨てた値は N か？)

：　：　：

- 税抜き価格は N でしょうか？ (つまり、$N \times 1.08$ を整数に切り捨てた値は N か？)

この中に “合っているもの” があればこれが税抜き価格ですので、これを出力しましょう。もしなければ、税抜き価格としてありうるものはないので、”:(” と出力しましょう。

$X = 1, 2, 3, \cdots, N$ すべてに対して 「税抜き価格が X か」 を判定する必要があります。これは、「繰り返し処理 (多くのプログラミング言語では for 文) を書くことで実装できます。

＜実装例 (Python 3) ＞ https://atcoder.jp/contests/sumitrust2019/submissions/8716074

★ 解法 2 - 「計算で答えを決め打ちする」

実は、全探索をしなくても、計算で答えを決め打ちすることができます。

アップルパイの税抜き価格を X とすると、税込み価格は $X \times 1.08$ を整数で切り捨てた値です。これが N 円になるためには、$N \le X \times 1.08 < N + 1$ が条件となります。

ここで、条件の値を全部 1.08 で割ってみると、$\frac{N}{1.08} \le X < \frac{N+1}{1.08}$ となります。ですので、$\frac{N}{1.08}$ 以上 $\frac{N+1}{1.08}$ 未満に整数があるならこれが税抜き価格となり、なければ税抜き価格としてありうるものがない、ということが分かります。

例えば、$N = 1001$ の場合、$926.851851 \cdots \le X < 927.777777 \cdots$ となるので、税抜き価格が 927 円であることが分かります。税抜き価格を 「$\frac{N}{1.08}$ を整数に切り上げた値」 と決め打ちして、税込み価格が N に一致するかどうかを判定する、というような実装もできます。

＜実装例 (Python 3) ＞ https://atcoder.jp/contests/sumitrust2019/submissions/8716275

C: 100 to 105

この問題では、「100 円から 105 円」 の性質を使って一発で判定する解法と、動的計画法 (DP) を使う全く違った解法があります。コンテストの時はどちらかで解いた人が多いと思いますが、違った解法を見てみることでも知見が広がります。ですので、ぜひ読んでみましょう。

★ 解法 1a - 「品物の個数を全探索」

品物の個数 C が決まっていると、合計価格としてありうるものは 「$100 \times C$ 円以上 $105 \times C$ 円以下」 すべてになります。ですので、判定が簡単です。

$C = 1, 2, 3, \cdots, X$ すべてに対して判定して、この中に 「ちょうど X 円の買い物をすることは可能」 と判定されたものがひとつでもあれば X 円の買い物ができ (その場合の答えは 「1」)、ひとつもなければどうやっても X 円の買い物ができません (その場合の答えは 「0」)。

★ 解法 1b - 「品物の個数を決め打ち」

実は、$C =$($\frac{X}{100}$ を整数で切り捨てた値) の値でちょうど X 円の買い物ができるならば答えは 「1」 で、そうでなければ答えは 「0」 です。なので、解法 1a の C を ($\frac{X}{100}$ を整数で切り捨てた値) と決め打ちして判定するだけでこの問題は解けるのです。

★ 解法 2 - 「2000 円以上ならどんな場合でも X 円の買い物ができる」

実は、2000 円以上ならどんな場合でも X 円の買い物ができます！ですので、$X \geq 2000$ の場合答えは必ず「1」 となります。

次に、$X \leq 1999$ の場合です。この場合、おにぎり・サンドイッチ・クッキー・ケーキ・飴・パソコンはそれぞれ 0 個以上 19 個以下買うことになります。ですので、$20 \times 20 \times 20 \times 20 \times 20 \times 20 = 6.4 \times 10^7$ 通りの買い方を全探索して、その中に合計価格が X 円となるものがあるか判定すれば良いです。コンピューターが 1 秒に計算できる回数は億単位ですので、適切な実装をすれば間に合います。

★ 解法 3 - 「動的計画法 (DP) でこの問題を解く」

dp_i を 「i 円の買い物ができるなら “Yes”、できないなら “No”」 とします。

- まず、dp_0 は “Yes” です。その後は、$dp_1, dp_2, dp_3, ..., dp_N$ の順に求められます。
- $dp_{i-100}, dp_{i-101}, dp_{i-102}, dp_{i-103}, dp_{i-104}, dp_{i-105}$ のどれかが “Yes” なら、これにひとつの品物を付け足した i 円の買い物ができるので dp_i が “Yes” になります。
- 6 つ全部 “No” な場合、i 円の買い物はどうやってもできないので dp_i は “No” となります。

このように、今まで求めた部分問題の答えを再利用するような方法を、「動的計画法 (DP)」 といいます。よく使われるアルゴリズムですので、知らない人は知っておきましょう。

D: Lucky PIN

この問題では、全探索による解法と、動的計画法による解法があります。

★ 解法 1 - 暗証番号を全探索

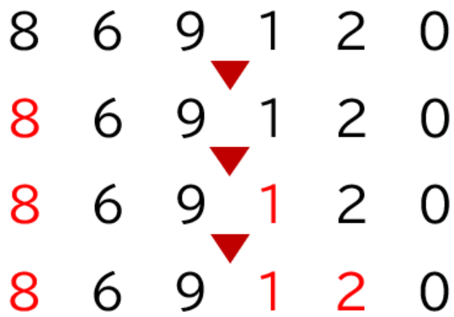
暗証番号 V を 000 から 999 まで決め打ちすることを考えます。すると、以下の問題を 1000 回解けばよいことになります。

- ラッキーナンバー S から、$N-3$ 桁を消して暗証番号 V を作ることができるか。

これは、以下のように貪欲法を用いて解くことができます。この解説では、S の i 文字目を S_i、V の j 文字目を V_j と表します。

1. $S_i = V_1$ となるような i の中で、最も小さい i を $p1$ とする。そのとき、$p1$ 文字目を残し、ステップ 2 へ進む。そのような i が無ければ暗証番号 V を作ることはできない。
2. $S_i = V_2$ となるような i であり、$p1+1$ 以上であるものの中で、最も小さい i を $p2$ とする。そのとき、$p2$ 文字目を残す。そのような i が無ければ暗証番号 V を作ることはできない。
3. $S_i = V_3$ となるような i であり、$p2+1$ 以上であるものの中で、最も小さい i を $p3$ とする。そのとき、$p3$ 文字目を残す。そのような i が無ければ暗証番号 V を作ることはできない。

例えば、$S = 869120, V = 812$ の場合、以下のようになり、暗証番号を作ることができます。



計算回数は $1000 \times N$ 回程度です。$N \leq 30000$ より、余裕を持って間に合います。

★ サンプルコード

- C++: https://atcoder.jp/contests/sumitrust2019/submissions/8695967

もう一つの解法を紹介します。

★ 解法 2 - 動的計画法による解法

以下の 3 つの値を記録して、文字列の左から動的計画法を行うことを考えます。

- pos: 最後にどの桁を見たか。
- len: 現時点で何文字を残すと決めたか。
- $current$: 現時点で残すと決まった部分の文字列。

例えば、$S = 869120$、$pos = 3$ の場合において、その時点で $1, 3$ 文字目のみを残すと決めた場合、$len = 2, current =$ ”89” となります。そのような状態を、

- $dp[pos][len][current]$ =(この状態にできるかどうかを 0 か 1 で表す)

というように多次元配列を用いて記録すると良いです。

また、動的計画法の遷移は「次の文字を残すか」「次の文字を消すか」の 2 通りしかないです。例えば $S = 869120$ において、$dp[3][2][89]$ からの遷移として考えられるのは、$dp[4][2][89]$ か $dp[4][3][891]$ のどちらかです。

計算回数は、$30000 \times 4 \times 1000 \times 2$ 回程度必要ですが、計算に時間がかかる操作が比較的少ないため間に合います。メモリも $30000 \times 4 \times 1000$ 個程度の配列を必要としますが、bool 型配列で良いので、余裕を持って間に合います。

★ サンプルコード

- C++: https://atcoder.jp/contests/sumitrust2019/submissions/8717655

おまけ: $100 \times N$ 回程度の計算でできる解法を考えてみましょう。また、$3 \times N$ 回程度の計算でできる解法も考えてみましょう。

E: Colorful Hats 2

この問題では、単純な掛け算のみで解く簡単な方法があるので、これを紹介します。

★ 本質的な考察

ここでは、人 $1, 2, 3, ..., i$ のうち赤色の帽子を被っている人数を a_i、青色の帽子を被っている人数を b_i、緑色の帽子を被っている人数を c_i とします。また、x_i を $[a_i, b_i, c_i]$ の中で 1 番目に大きい値、y_i を $[a_i, b_i, c_i]$ の中で 2 番目に大きい値、z_i を $[a_i, b_i, c_i]$ の中で 3 番目に大きい値とします。

そのとき、「人 i の前にいる、人 i と同じ色の帽子を被っている人数 A_i」が分かれば、x_i, y_i, z_i の値は特定できるのです。

★ 具体例

例えば、$N = 6$ で、$A_i = 0, 1, 0, 0, 1, 2$ の場合、x_i, y_i, z_i の値は以下の表のようになります。

i	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$
x_i	0	1	2	2	2	2	3
y_i	0	0	0	1	1	2	2
z_i	0	0	0	0	1	1	1

例えば、$i = 4$ のときを考えましょう。人 3 までの時点で、2 $(= x_i)$ 人が被っている色の帽子、1 $(= y_i)$ 人が被っている色の帽子、0 $(= z_i)$ 人が被っている色の帽子があります。$A_4 = 0$ より、人 4 はその時点で 0 人が被っている色の帽子を被ることが確定します。そして、人 4 までの時点で多い順に $[2, 1, 1]$ 人が被っている帽子の色があると分かります。

★ 掛け算で答えを求める

最後に、「人 $1, 2, ..., i-1$ の帽子の色が決まったとき、人 i の帽子の被り方は何通りあるか」を考えましょう。（これを T_i とします。）x_i, y_i, z_i の値が分かっているので、各人ごとに独立に考えることができ、答えは $T_1 \times T_2 \times T_3 \times ... \times T_N$ となります。

また、T_i の値は $[x_{i-1}, y_{i-1}, z_{i-1}]$ の中でいくつ A_i と等しいものがあるか、になります。計算量は $O(N)$ です。

★ 上の例について答えを求めてみよう！

上の例において、5 人目までの時点での帽子の色を、多くの人が被っている順に α, β, γ とします。（それぞれ x_5, y_5, z_5 人が被っています）そのとき人 6 は、$x_5 = A_6, y_5 = A_6$ を同時に満たすため、α 色と β 色を被る可能性があり、$T_6 = 2$ となります。同様に他の T_i についても求めていくと、上の例の答えは $3 \times 1 \times 2 \times 1 \times 2 \times 2 = 24$ と分かります。

★ サンプルコード

- C++: https://atcoder.jp/contests/sumitrust2019/submissions/8683667
- Python3: https://atcoder.jp/contests/sumitrust2019/submissions/8697277

F: Interval Running

この問題は一見複雑な場合分けが必要に見えるかもしれません。ただ、これはあることに気づくことで、一気に単純になります。この問題を解くのに二種類の方針があるので、それぞれ説明していきます。

★ 解法 1 「相対速度を考える」

2 人の間の距離の差を考えてみましょう。時刻 t での高橋君のスタートからの距離を $f(t)$、青木君のスタートからの距離を $g(t)$ として、$f(t) - g(t)$ のグラフを考えてみましょう。これは、下図のようになります。

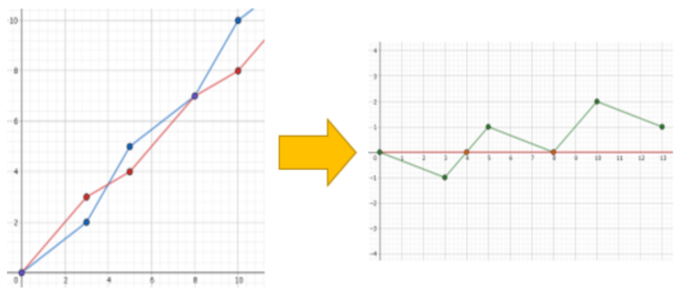


図. 高橋君と青木君の距離の差のグラフ

よく見てみましょう。これは、実に単純な形になっています！

- 最初の T_1 分間で、$(A_1 - B_1) \times T_1$ メートル分増える
- 次の T_2 分間で、$(A_2 - B_2) \times T_2$ メートル分増える
- これが交互に無限回繰り返される

さて、高橋君と青木君が出会うタイミングは、2 人の距離の差が 0 になったとき、つまり x 軸との交点になります。そこで、$P = (A_1 - B_1) \times T_1$、$Q = (A_2 - B_2) \times T_2$ として、次のような数直線上の点 (最初は「0 の位置」にいる) の動きを考えてみましょう。

- 点が距離 P だけ移動する (x の位置から $x + P$ の位置に動かされる)
- 点が距離 Q だけ移動する (x の位置から $x + Q$ の位置に動かされる)
- これらを 1 サイクルとし、それが交互に無限回繰り返される

そのときに「0 の位置」を何回通るかが、この問題の答えとなります。 ここで、$P < 0$ の場合 (最初負方向に移動する場合) の答えを、3 通りに場合分けして考えてみましょう。($P > 0$ の場合は、P, Q を -1 倍すると同じ答えが得られます。)

$P + Q < 0$ の場合

答えは 0 になります。なぜなら、1 サイクルが終わったとき正方向に行くので、二度と 0 の位置に来ることがないからです。

$P + Q = 0$ の場合

答えは無限になります。なぜなら、1 サイクルが終わったら 0 の位置に戻ってきて、これが無限回繰り返さ

れるからです。

$P+Q>0$ の場合

k サイクル目が終わると、点は $k\times(P+Q)$ の位置にあります。その次のサイクルで 0 の位置を何回通ることになるでしょうか？

- $k\times(P+Q)+P<0$ の場合：2 回 (0 の位置より奥で跳ね返るので)
- $k\times(P+Q)+P=0$ の場合：1 回 (0 の位置で跳ね返るので)
- $k\times(P+Q)+P>0$ の場合：0 回 (0 の位置でより手前で跳ね返るので)

1 サイクル目では 1 回だけ 0 の位置を通ります。それを考えると、$\frac{-P}{P+Q}$ の商を S、あまりを T として、0 の位置を通る回数は次のようになります。

- $T\neq0$ の場合、$S\times2+1$ 回 ($1+2+2+2+\cdots+2+0+0+\cdots$ 回)
- $T=0$ の場合、$S\times2$ 回 ($1+2+2+2+\cdots+2+1+0+0+\cdots$ 回)

これで、単純な場合分けと簡単な計算で、この問題の答えを求めることができました！

＜実装例 (Python 3)＞ https://atcoder.jp/contests/sumitrust2019/submissions/8717451

★ 解法 2 「性質が変わるところを二分探索」

高橋君・青木君の T_1+T_2 秒の動作を 1 サイクルとします。

このとき、1 サイクルで高橋君・青木君が出会う回数は、最初から順に $1,2,2,2,2,2,\cdots,2,1,0,0,0,\cdots$ のようになっている場合もあれば、$1,1,1,1,1,1,\cdots$ のようになっている場合もあります。しかしどんな場合でも、「数が変わる場所」は高々 3 ヶ所しかありません。ですのでこの “境界部分” を、二分探索などを用いて上手く求めると、入力された値の対数時間で答えを求めることができます。

有限回しか高橋君と青木君が出会わない場合でも、時刻 10^{20} くらいで出会うことがあり、出会う場所の座標が 10^{30} 程度になることがあります。これは 64 ビット整数型におさまりませんが、`Java`・`C#`・`Python` などのプログラミング言語では多倍長整数演算を扱うことで解くことができます。また、C++ でも実行環境によっては「`__int128`」という型で 128 ビット整数型を扱うことによって解くことができます。