Code Festival 2018 予選 B 解説

2018 年 10 月 14 日

Writer: E869120, square1001

Problem A: Probability of Participation

まず一個気づくべきところとしては、出力の項を見ると「パーセント単位で出力しなさい」となっています。しかし、振るサイコロは 100 面体なので「1 つの目が出る確率」＝「1 パーセント」となります。ですので、以下の問題に帰着できるのです。

- 1 から 100 の中で N で割り切れない整数は何個あるか？

これは、逆に「N で割り切れる整数の個数」を考えると分かりやすいです。そのような個数は $100/N$ 個（小数点以下切り捨て）なので、「割り切れない整数の個数」は $100-(100/N)$ となります。あとはこれを出力するだけです。

C++ だと、「`cout << 100 - 100 / N << endl;`」のような文でできます。

ソースコード (C++)：

- https://beta.atcoder.jp/contests/code-festival-2018-qualb/submissions/3273506

Problem B: Tensai

顔の面白さが x の人を 1 回トレーニングすると、どのような状態でも写真の好感度はちょうど x 上がります。1 回だけトレーニングが行えるとすると、写真の好感度（の増分）を最大化したい場合、顔の面白さが最も大きい人をトレーニングすればよいです。

何回トレーニングをしても顔の面白さは変わらないので、好感度の増分はどのトレーニングでも変わらないことになります。したがって、顔の面白さが最も大きい人を X 回トレーニングしたときに、写真の好感度が最大になります。

このように、「最も得をする動きを繰り返して最適な答えを出す」ようなアルゴリズムを、「貪欲法」といいます。知っておくとよいでしょう。

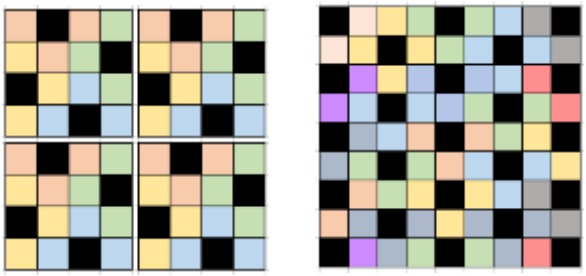
また、少し考えれば、この値が $a_1b_1 + a_2b_2 + a_3b_3 + \cdots + a_Nb_N + X \times \max(b_1, b_2, b_3, \ldots, b_N)$ であることが分かります。そうすると、時間計算量 $O(N)$ でこの問題を解くことができます。

ソースコード（C++）：

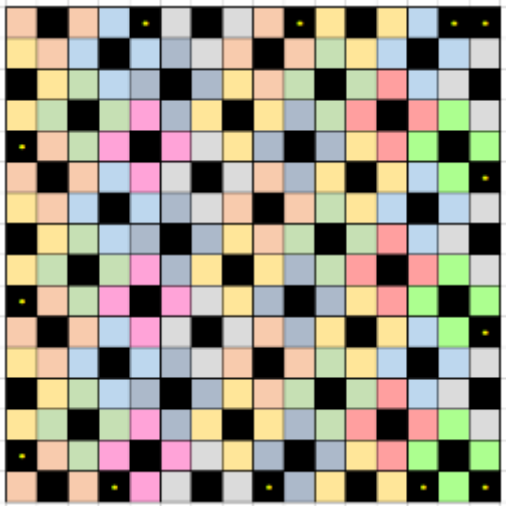
- https://beta.atcoder.jp/contests/code-festival-2018-qualb/submissions/3374090

Problem C: Special Cake for CODE FESTIVAL

まず、この問題には様々な解法があります。これは思いついた人も多いかもしれませんが、以下のような方法でスプレーをかける位置を決めると $H, W = 1{,}000$ のとき $250{,}000$ 個程度のケーキにかけることになります。そのため、正解できません。（下図において、黒い場所にスプレーをかけます。色は各スプレーの影響範囲です。）



ですので、より良いアイデアを考える必要があります。



例えば上図のようなアイデアの場合、**基本的には** $(x + 2y)\ mod\ 5 = 0$ のときのみ、上から x 番目かつ左から y 番目のピースにスプレーをかけることになります。

しかし、下図でいう「黄色い点がある黒く塗られたピース」は、$(x + 2y)\ mod\ 5 = 0$ を満たすマスのみにスプレーをかけると**食べることができてしまいます**。ですのでこのようなピースについては別途スプレーをかけなければなりません。さて、このようなピースは何個あるでしょうか。図を見てもらえれば分かる通り、端にあるピースのおよそ 5 分の 1 だけがあてはまるので、$H, W = 1\,000$ のとき $(1\,000 \times 4 - 4) \div 5 \fallingdotseq 800$ 個程度となります。よって、おおよその回数は $200\,000 + 800 = 200\,800$ 個。制限回数である $201\,800$ 個以内のケーキにスプレーをかけることになるので、正解となります。

ソースコード（C++）：

- https://beta.atcoder.jp/contests/code-festival-2018-qualb/submissions/3274338

Problem D : Sushi Restaurant

◆ 空腹度と寿司の数が決まっていた場合の「不適合度」

まず、空腹度 $a_1, a_2, a_3, \dots, a_N$ の人がおり、寿司の数が $b_1, b_2, b_3, \dots, b_N$ の皿があるときの「不適合度」は、次のように計算することができます。

> $a'_1, a'_2, a'_3, \dots, a'_N$ を $a_1, a_2, a_3, \dots, a_N$ を昇順ソートしたものとし，$b'_1, b'_2, b'_3, \dots, b'_N$ を $b_1, b_2, b_3, \dots, b_N$ を昇順ソートしたものとすると、「不適合度」は $|a'_1 - b'_1| + |a'_2 - b'_2| + |a'_3 - b'_3| + \cdots + |a'_N - b'_N|$ となる。

上の計算がなぜ成立するかは、次のようにして証明できます。「i 番目に寿司の数が少ない (同じ場合は適当に順序付ける) 皿を取るの人が、空腹度が p_i 番目に少ない人である」とき、 $i < j$ かつ $p_i > p_j$ であるとき、この 2 つの皿を交換すると不適合度が上がることはない、ということで、最終的に $p_1 = 1, p_2 = 2, p_3 = 3, \dots, p_N = N$ の状態になるまで交換し続ければ不適合度が上がることはないので、$p_1 = 1, p_2 = 2, p_3 = 3, \dots, p_N = N$ のときに不適合度が最小になることが分かります。つまり、上の計算が成立します。

◆ 確率計算：ステップ 1 – 方針を立てよう

上の計算より、「i 番目に寿司の数が少ない皿」は、「空腹度が i 番目に少ない人」にしか関係しないことが分かりました。そこで、「空腹度が i 番目に少ない人の空腹度は、どのような確率分布になっているだろうか？」ということについて考えます。

問題文から、空腹度は $x_1, x_2, x_3, \dots, x_M$ のいずれかであることは明らかです。そこで、「空腹度が i 番目に少ない人」の空腹度が x_j である確率を t_j とします (t_j の計算は後でやります)。そのとき、i 番目の皿の寿司の数が c であるときのこの人の不満度の期待値は、$t_1|x_1 - c| + t_2|x_2 - c| + t_3|x_3 - c| + \cdots + t_M|x_M - c|$ となります。実は、c を X 軸に、不満度を Y 軸にとったグラフは折れ線状になり、$c = x_1, x_2, x_3, \dots, x_M$ のいずれかで最小値を取ります。この最小値の計算は、普通にやっても計算量 $O(M^2)$ でき、少し工夫すれば計算量 $O(M)$ で求めることができます。これを N 人の人に対して求めるので、全体で計算量 $O(NM)$ で計算することができます。

さて、残りは $t_1, t_2, t_3, \dots, t_M$ の値 (確率分布) を求めるだけです。

◆ 確率計算：ステップ 2 - $O(N^2 \times M)$ の解法

$t_{i,j}$ を、「空腹度が i 番目に小さい (同じ場合は、ID を適当に割り振ってこれが小さい方とする) 人の空腹度が x_j である確率」とします。

これを言い換えると、ある $p \leq i < q$ を満たす整数 (p, q) があり、空腹度が $(p-1)$ 番目までに小さい人の空腹度が x_j 未満であり、空腹度が p 番目から $(q-1)$ 番目までに小さい人の空腹度が x_j であり、空腹度が q 番目以降に小さい人の空腹度が x_j より大きい、ということです。

このような確率は、$pl = p_1 + p_2 + \cdots + p_{j-1}, pr = p_{j+1} + p_{j+2} + \cdots + p_M$ として、$(pl)^{p-1} \times (pr)^{M-q+1} \times (p_j)^{q-p} \times \frac{M!}{(p-1)!(M-q+1)!(q-p)!}$ で計算することができます。

この掛け算で求められる確率が「支配する」範囲は、$t_{p,j}, t_{p+1,j}, t_{p+2,j}, \ldots, t_{q-1,j}$ です。つまり、(p, q, j) を全探索して、それぞれが「支配する」範囲に対して掛け算で求められる確率を足していけば、すべての $t_{i,j}$ の値が求まります。

pl, pr, p_j の累乗や、$M!$ までの階乗の値を前もって計算しておけば、(p, q, j) に対する確率はそれぞれ定数時間で求めることができます。また、範囲に対して足す操作は「いもす法」とよく呼ばれている、累積和を応用したテクニック（知らない人は調べてみると分かると思います）を用いて定数時間でできます。したがって、全体の計算量は、(p, q, j) の組み合わせ数と同じオーダーの $O(N^2 \times M)$ となります。

◆ 確率計算：ステップ 3 - $O(NM)$ の解法

制約を見てみると、$N \leq 2000, M \leq 2000$ となっています。これは、ステップ 2 の解法が実行時間制限に間に合わないということを意味します。そこで、次のような考察をすると、(p, q, i) を全探索する必要がなくなってきます。

まず、$u_{i,j}$ を「空腹度が i 番目に小さい人の空腹度が x_j 以上である確率」とします。まず、空腹度が i 番目に小さい人の空腹度が x_j 以上であることを言い換えれば、ある p $(p \leq i)$ について、空腹度が $(p-1)$ 番目までに小さい人の空腹度が x_j 未満、p 番目以降に小さい人の空腹度が x_j 以上、ということになります。この確率は、$pl = x_1 + x_2 + \cdots + x_{j-1}, pr = x_j + x_{j+1} + \cdots + x_M$（$pr$ の定義が先ほどと違うことに注意）として、$(pl)^{p-1} \times (pr)^{M-p+1} \times \frac{M!}{(p-1)!(M-p+1)!}$ が求める確率になります（ステップ 2 と同じように計算できます）。また、この (p, j) の組が支配する $u_{i,j}$ の範囲は $u_{p,j}, u_{p+1,j}, u_{p+2,j}, \ldots, u_{N,j}$ なので、ステップ 2 と同じように「いもす法」を使って高速に計算することができます。

少し考えてみると、実は $t_{i,j} = u_{i,j} - u_{i,j+1}$ であることが分かります（定義に当てはめてみればわかると思います）。なので、全部の $u_{i,j}$ が求まれば、$t_{i,j}$ も全部求まることになります！さらに、ステップ 2 では (p, q, j) 全部に対して計算していたのに対し、今回の方法だと (p, j) 全部に対してしか計算しなくてもよいので、この組の個数が $O(NM)$ 個であることから、全体の計算量 $O(NM)$ でこの問題を解くことができました！

実装上の注意ですが、double 型は 10^{308} までの精度しか持たないので、対数（log）を使って確率を計算して、最後に exp して復元すると、オーバーフローを気にすることなく確率を計算することができます。

ソースコード（C++）

◆ https://beta.atcoder.jp/contests/code-festival-2018-qualb/submissions/3280159

E. Game of +-

◆ ステップ 1： 問題の言い換え

M を $1,2,3,4,5,6,\ldots,N$ の最小公倍数として、「G として作れる最小の数」は $1/M$ になります。

まず、「$1/a\ (1 \le a \le N)$ を増やす / 減らす」という操作について考えますが、分数で考えるよりも整数で考えた方がやりやすいので、代わりに各操作を「$M/a\ (1 \le a \le N)$ を増やす / 減らす」にし、最終的な G の値を 1 にすることを考えます。

例えば $N = 7$ の場合、$M = lcm(1,2,3,4,5,6,7) = 420$ となるので、以下の数だけ G を増減させることで 1 を作ることを考えればよいです。これ以降では、まず $G \equiv 1\ (mod\ M)$ となる整数 G を作ることを考えていきます。

a	1	2	3	4	5	6	7
対応する数	420	210	140	105	84	70	60

ここからは、最終的に $G = 1$ となるように操作することではなく、$G \equiv 1\ (mod\ M)$ となるように操作することを考えます。なぜなら、残りの分は $M/1$ を足す・引くことによって $G = 1$ にできるからです。

◆ ステップ 2： mod について考える

$G \equiv 1\ (mod\ M)$ という条件は、「M を素因数分解したときに、全ての項で割ったあまりが 1 となる」と言い換えられます。例えば $M = 420$ のとき、素因数分解すると $2^2 \times 3 \times 5 \times 7$、すなわち $4 \times 3 \times 5 \times 7$ となります。ですので、p を $4,3,5,7$ で割ったあまりが全て 1 であることと $p \equiv 1\ (mod\ 420)$ を満たすことは同値です。

そこで皆さん考えてみましょう。各操作「$M/a\ (1 \le a \le N)$ を増やす / 減らす」ときに $G\ mod\ p^c$ の値はどれだけ変わるのでしょうか。

$M = p_1^{e_1} \times p_2^{e_2} \times p_3^{e_3} \times \ldots \times p_s^{e_s}$ と素因数分解されるとします。整数 a として $p_i^{e_i}$ を選んで操作すると、実は $G\ mod\ p_i^{e_i}$ の値だけ変わり、すべての $k\ (k \neq i)$ に対して $p_k^{e_k}$ の値は変わらないのです。

なぜなら、$a = \frac{M}{p_i^{e_i}} = p_1^{e_1} \times p_2^{e_2} \times p_3^{e_3} \times \ldots \times p_{i-1}^{e_{i-1}} \times p_{i+1}^{e_{i+1}} \times \ldots \times p_s^{e_s}$ となるから、足す数（あるいは引く数）を $p_k^{e_k}$ で割った余りは 0 になるからです。例えば、$N = 7$ の場合は以下の表のようになります。

a	足す数	mod 4	mod 3	mod 5	mod 7
4	105	**1**	0	0	0
3	140	0	**2**	0	0
5	84	0	0	**4**	0
7	60	0	0	0	**4**

◆ ステップ 3： 最終段階 - 操作の方法 ・ 操作回数の解析

残りはもう簡単です。「G に $M/p_i^{e_i}$ を足すこと」をすると $mod\ p^c$ の値はいくつ増えるか（以後、この値を D とする)、といった値を求めた後は、「$M/p_i^{e_i}$ で何回足せば / 引けばいいか」というのを求めるだけです。これは以下のような方法を使うことによって操作回数を少なくできます。

- $DA \equiv 1\ (mod\ p_i^{e_i})$ を満たす最小の正整数を A とする
  - $2A < p_i^{e_i}$ のとき、A 回「足す」
  - $2A \geq p_i^{e_i}$ のとき、$p^{e_i} - A$ 回「引く」

操作回数は、以下のように解析することができます。

- $N = 100$ の場合、$lcm(1, 2, \dots, 100) = 2^6 \times 3^4 \times 5^2 \times 7^2 \times 11 \times 13 \times 17 \times 19 \times \dots \times 89 \times 97$ となり、$2^6, 3^4, 5^2, 7^2, 11, \dots, 89, 97$ を全部足した値は 1262 となります。
- 各素因数の項 $p_i^{e_i}$ につき、期待値は $p_i^{e_i}/4$ 程度となります。なぜなら、A の値に明確な規則性はなく、$[0, 1, 2, 3, 4,\ 5, \dots, p_i^{e_i}/2, p_i^{e_i}/2, \dots, 5, 4, 3, 2, 1]$ の平均は約 $p_i^{e_i}/4$ となるからです。
- ですので、回数の期待値は $1262 \div 4 = 315.5$ 回程度となります。この方法でやって G/M の値が正の中で最小にならない（$1.000\dots, 2.000\dots, -0.999\dots$ といった値となる）場合に $+1\ /\ -1$ を足さなければならないので、320 回程度と考えられます。

実際の値は、最悪ケースの $N = 79$ のとき、求めた期待値よりも若干少ない 298 回となりますので、余裕をもって制限の 320 回に間に合います。少し方法が違っており数回間に合わない、という場合でも、例えば $mod\ 64$ に関して $1/64$ だけでなく $1/32$、$1/16$、$1/8$ などをフル活用することによって合計 30 回程度減らすことも可能です。

【サンプルコード (C++)】

- https://beta.atcoder.jp/contests/code-festival-2018-qualb/submissions/3274375

【おまけ】

$N = 100$ の場合の想定解答は、以下のようになります。（※収まるように出力形式を変えています）

```
(+, 11) (+, 11) (+, 11) (+, 11) (+, 25) (+, 25) (+, 25) (+, 25) (+, 25) (+, 25) (+, 25) (+, 25) (+, 31)
(+, 31) (+, 31) (+, 37) (+, 37) (+, 37) (+, 37) (+, 37) (+, 37) (+, 37) (+, 37) (+, 37) (+, 37) (+, 37)
(+, 37) (+, 37) (+, 37) (+, 37) (+, 37) (+, 47) (+, 47) (+, 47) (+, 47) (+, 47) (+, 47) (+, 47) (+, 47)
(+, 47) (+, 47) (+, 47) (+, 47) (+, 47) (+, 47) (+, 47) (+, 47) (+, 49) (+, 49) (+, 49) (+, 49) (+, 64)
(+, 64) (+, 64) (+, 64) (+, 64) (+, 64) (+, 64) (+, 64) (+, 64) (+, 64) (+, 64) (+, 73) (+, 73) (+, 73)
(+, 73) (+, 73) (+, 73) (+, 73) (+, 73) (+, 73) (+, 73) (+, 73) (+, 73) (+, 73) (+, 73) (+, 73) (+, 73)
(+, 73) (+, 73) (+, 73) (+, 79) (+, 79) (+, 79) (+, 79) (+, 81) (+, 81) (+, 81) (+, 81) (+, 81) (+, 81)
(+, 81) (+, 81) (+, 81) (+, 81) (+, 81) (+, 81) (+, 81) (+, 81) (+, 81) (+, 81) (+, 81) (+, 81) (+, 81)
(+, 81) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89)
(+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89)
(+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 89)
(+, 89) (+, 89) (+, 89) (+, 89) (+, 89) (+, 97) (+, 97) (+, 97) (+, 97) (+, 97) (+, 97) (+, 97) (+, 97)
(+, 97) (+, 97) (+, 97) (+, 97) (+, 97) (+, 97) (-, 13) (-, 13) (-, 17) (-, 17) (-, 17) (-, 19) (-, 19)
(-, 19) (-, 19) (-, 19) (-, 19) (-, 19) (-, 19) (-, 23) (-, 23) (-, 23) (-, 23) (-, 23) (-, 23) (-, 23)
(-, 29) (-, 29) (-, 29) (-, 29) (-, 29) (-, 29) (-, 29) (-, 29) (-, 41) (-, 41) (-, 41) (-, 41) (-, 41)
(-, 41) (-, 41) (-, 41) (-, 43) (-, 43) (-, 43) (-, 43) (-, 43) (-, 43) (-, 53) (-, 53) (-, 53) (-, 53)
(-, 53) (-, 53) (-, 53) (-, 53) (-, 53) (-, 53) (-, 53) (-, 53) (-, 53) (-, 53) (-, 53) (-, 53) (-, 53)
(-, 53) (-, 53) (-, 53) (-, 53) (-, 53) (-, 53) (-, 59) (-, 59) (-, 59) (-, 59) (-, 59) (-, 59) (-, 59)
(-, 59) (-, 59) (-, 59) (-, 59) (-, 59) (-, 59) (-, 59) (-, 59) (-, 59) (-, 59) (-, 59) (-, 59) (-, 59)
(-, 59) (-, 59) (-, 59) (-, 61) (-, 61) (-, 61) (-, 61) (-, 61) (-, 61) (-, 67) (-, 67) (-, 67) (-, 67)
(-, 67) (-, 67) (-, 67) (-, 67) (-, 67) (-, 71) (-, 71) (-, 71) (-, 71) (-, 71) (-, 71) (-, 71) (-, 71)
(-, 71) (-, 71) (-, 71) (-, 71) (-, 71) (-, 83) (-, 83) (-, 83) (-, 83) (-, 83) (-, 83) (-, 83)
```