AtCoder Beginner Contest 100 Editorial

by E869120, square1001

June 16th, 2018

For international readers: English editorial starts from page 6.

全体の講評

問題	正解者数	提出数	難易度*	First AC	最速実行	最短コード
A: Happy Birthday!	2321	3656	150	0:40	0ms	19 bytes
B: Ringo's favorite numbers	1806	6710	280	1:40	0ms	22 bytes
C: *3 or /2	1887	2433	270	1:02	1ms	42 bytes
D: Patisserie ABC	574	1636	450	5:48	1ms	138 bytes

※コンテスト中の結果です。また、難易度* はコンテスト中の得点で換算した予想難易度です。

今回は、ABC 100 回目なので全体の講評を書きました。
皆さんコンテストに参加いただきありがとうございました。

今回のコンテストは、ABC 100 回目と記念すべき回なので、誕生日などに多く食べることのできる「ケーキ」をテーマにした問題が多くありました。
例えば、A 問題。これは設定が誕生日でしたが、実は ABC100 回目記念だからケーキにしたという隠された意図がありました。あなたは読み取れましたでしょうか？

また、今回は制約に「100 の倍数」が多く出現しました。例えば、D 問題の制約は $N \leq 10000$ かつ $a_i \leq 10^{10}$ ですね。10 や 100 が多く含まれるように意図的に設定しました。ちなみに B 問題の「100 で割った回数」の 100 も ABC100 由来です。

という訳で、皆さん解説の最初のページをお読みいただきありがとうございました。解説本体もお楽しみくださいませ。

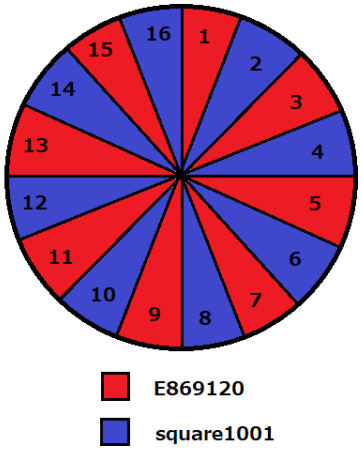
解説自体は、2 ページ目からとなります。

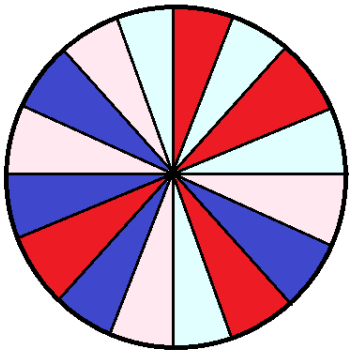
問題 A：Happy Birthday!

答えは、 $X \leq 8$ かつ $Y \leq 8$ のとき答えは “Yay!” となり、それ以外のとき答えは “:(“ となります。

まず、同じ人が取るケーキが隣り合ってはいけないので、16 ÷ 2 = 8 個を超えて置くことはできません。つまり、どちらかが 9 以上のとき食べたい数のケーキを食べることができないことが示せます。

また、$X \leq 8, Y \leq 8$ のとき、E869120 君が時計回りで 1, 3, 5, 7, 9, 11, 13, 15 番目の中から、square1001 君が時計回りで 2, 4, 6, 8, 10, 12, 14, 16 番目の中から選ぶと、食べたい数のケーキを、取り方の条件を満たしながら食べることができます。





[Example: A = 4, B = 4]

これが分かれば、if 文などを用いて条件分岐をするだけです。

サンプルコード： https://beta.atcoder.jp/contests/abc100/submissions/2663851

問題 B：Ringo’s Favorite Numbers

この問題には、たくさんの解法があります。そのうち代表的な 2 通りの解法を説明します。

方針 1：「全探索」

「整数 x が与えられる。そのとき、x は 100 で何回割り切れるか？」という問題を解くことを考えます。これは、x を 100 で割り切れないようになるまで割り続けることによって、解くことができます。

答えが K の場合、$x = 1,2,3,\ldots,K$ について先ほどの問題を解くことになります。先ほどの問題は十分高速に動作し、K の値は高々 1010000（$D = 2, N = 100$ のとき）なので、実行時間制限に余裕で間に合うプログラムを書くことができます。

サンプルコード： https://beta.atcoder.jp/contests/abc100/submissions/2669995

方針 2：「条件に当てはまる数を列挙」

100 でちょうど D 回割り切れるような数は、$a = 100^D$ として次のようになります：

$a, 2a, 3a, 4a, \ldots, 99a, 101a, 102a, 103a, \ldots, 199a, 201a, 202a, 203a, \ldots, 299a, 301a, \ldots$

すなわち、$m \times 100^D$（m は 100 の倍数ではない）と表される数だけが 100 でちょうど D 回割り切れます。これをもとに解法を考えてみましょう。

$q =$（$N - 1$ を 99 で割った商）とします。そのとき、この問題の答えは $(100q + 1) \times 100^D$ 以上 $(100q + 99) \times 100^D$ 以下の値になります。また、$100q \times 100^D$ 以下に「100 でちょうど D 回割り切れるような数」は $99q$ 個あるので、先ほど答えを限定した範囲の中で $N - 99q$ 番目になります。したがって、答えは $(100q + N - 99q) \times 100^D$ になります。

より簡略化すると、答えは $(N - q) \times 100^D = \left(N - \left\lfloor \frac{N-1}{99} \right\rfloor\right) \times 100^D$ になります。

方針 2+α：「ズルい解法：N≤100 までを利用して解く」

「方針 2」の考察で、$a = 100^D$ として、100 でちょうど D 回割り切れるような数の最初 100 個は $a, 2a, 3a, 4a, \ldots, 97a, 98a, 99a, 101a$ になります。

したがって、$N \leq 100$ を仮定すると、$N \leq 99$ のとき $Na = N \times 100^D$ が答えとなり、$N = 100$ のとき $101a = 101 \times 100^D$ が答えになります。

サンプルコード： https://beta.atcoder.jp/contests/abc100/submissions/2663837

問題 C：*3 or /2

$N \leq 10000, a_i \leq 10^9$ なので，操作をたくさん行ったときに、最終的な値が 90000 桁近くなるケースがあります。これは、仮に最適な方法が分かっていたとしても、シミュレーションをするのには桁数が大きすぎるので、何か別の方法を考えなければなりません。

まず、「3 倍する」操作を何回かやっておくことで、「2 で割る」操作の回数が増えたり減ったりするでしょうか？答えは No です。なぜなら、奇数を掛けて 2 で割れる回数が増えることはなく、整数を掛けて 2 で割れる回数が減ることがないからです。

したがって、「3 倍する」操作はその後の操作回数に何ら影響を及ぼしません。すなわち、「3 倍する」は「何もしない」という条件に置き換えることができます。よって、各操作は「N 個すべての数に対して、その数を「2 で割る」または「そのままにする」のどちらかを選ぶが、全部そのままにすることはできない」というように置き換えられます。

次に、各操作について考えてみましょう。「2 で割る」ことによって残り操作回数は減りますが、「そのままにする」は残り操作回数が減らないので、各操作で「2 で割る」個数を最小化、すなわち「2 で割る」個数を 1 個にすることを考えます。したがって、1 回の操作で「N 個の数の中から 1 つ選び、これを 2 で割る」に置き換えることができます。

あとはウィニングランです。答えは (a_1 が 2 で割れる回数) + (a_2 が 2 で割れる回数) + … + (a_N が 2 で割れる回数) となります。x が 2 で割れる回数は単純にシミュレーションしても計算量 $O(log\, x)$ で求めることができるので、全体の計算量は $O(N\, log(max\, a_i))$ になります。

おまけ：計算量 $O(N\, log\, log\, (max\, a_i))$ で求める方法もあります。x が 2 で割れる回数を二分法で求めることによって、この計算量を達成することができます。

サンプルコード： https://beta.atcoder.jp/contests/abc100/submissions/2670458

問題 D：Patisserie ABC

まず、以下の簡単な問題を考えてみましょう。

> N 個のケーキがあり、綺麗さ・おいしさ・人気度が決まっている。
> M 個のケーキを選ぶ。
> （綺麗さの合計） ＋ （おいしさの合計） ＋ （人気度の合計） の最大値を求めよ。

この問題は簡単に解くことができます。ケーキ i の綺麗さを x_i、おいしさを y_i、人気度を z_i とすると、求める値が $(x_i + y_i + z_i)$ の合計になるので、$(x_i + y_i + z_i)$ の値の大きい順に M 個取るという解法が通用します。

しかし、本来の問題は、（綺麗さの合計の**絶対値**）＋（おいしさの合計の**絶対値**）＋（人気度の合計の**絶対値**）を最大化する問題でした。
すなわち、いくつかの要素の合計を負にする、という手も存在します。

ですので、綺麗さ・おいしさ・人気度のそれぞれの要素について、「正の方向に最大化する」「負の方向に最大化する」のどちらを選ぶか、ということを考えると分かりやすいです。要素は 3 つあるので、$2^3 = 8$ 通り全探索します。
例えば、綺麗さ・おいしさを正の方向に最大化、人気度を負の方向に最大化したい場合、$x_i + y_i - z_i$ の大きい順にソートして、上から M 個のケーキを選ぶと良いです。

答えは全探索した 8 通りの中で最も良い解となります。全探索はビット演算などを用いると簡単になります。計算量は $O(N \log N) * 8 = O(N \log N)$ となります。

また、この問題は実は $O(N)$ で解くことができます。長さ N の数列をソートして M 番目に大きい値 A を最悪計算量 $O(N)$ で求めるアルゴリズムは このリンク に書かれており、値 A が求まった後は求めた値より大きい値の合計を計算すれば良いので、この問題の最悪計算量は $O(N)$ となります。

サンプルコード： https://beta.atcoder.jp/contests/abc100/submissions/2670489

Overall Analysis / Review

Problem	Corrects	Submissions	Difficulty*	First AC	Fastest	Shortest
A: Happy Birthday!	2321	3656	150	0:40	0ms	19 bytes
B: Ringo's favorite numbers	1806	6710	280	1:40	0ms	22 bytes
C: *3 or /2	1887	2433	270	1:02	1ms	42 bytes
D: Patisserie ABC	574	1636	450	5:48	1ms	138 bytes

※Expected difficulty is a score in this problem's difficulty. It's only a guess from a formula.

Thank you for everyone to participating the contest or seeing this editorial! This contest is exactly 100th ABC, so we wrote a new thing, "analysis / review".

Since this contest is an anniversary, some problems' theme was "cake", which is commonly eaten on birthdays (at least in Japan).

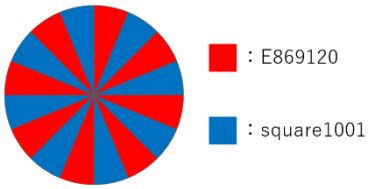
For example, problem A. This was about E869120 and square1001's 16th birthday, but there was a hidden intent to celebrate this ABC 100 anniversary. Or did you make a guess correctly?

And also, in this contest, many "multiple of 100" were appeared. For example, the constraints of problem D was $N \leq 10000 = 100^2$ and $a_i \leq 10^{10} = 100^5$. And the problem B's "the number of times that can be divided by 100" is derived from the ABC "100". Also, there are many 10's because square root of 100 is exactly 10.

Anyway, thanks for reading the first page of English editorial. Please enjoy the main body of this editorial :)

The main body of this editorial starts from the next page.

Problem A : Happy Birthday!



First, consider the case of $X \leq 8$ and $Y \leq 8$ (both wants no more than 8 pieces of cakes). In this case, if you only allow E869120 taking X pieces in red area and square1001 taking Y pieces in blue area, it will satisfy the condition (no two adjacent cakes are for same person). Therefore, the answer for this case is “Yay!”.

Second, if one person wants to take more than 8 cakes, it will definitely not satisfy the condition, so the answer for this case is “:(“.

If you realize this, remaining is only some implementation. The implementation of this problem is as easy as a piece of cake. You can use if-else statement to branch two conditions.

Sample Code (C++) : https://beta.atcoder.jp/contests/abc100/submissions/2663851

Problem B : Ringo's Favorite Numbers

There are many possible solutions in this problem. This time, we will explain two representative solutions.

Approach 1 : Search Everything

Consider this sub-problem: "You are given an integer x. How many times can x be divided by 100?". This can be solved by simulation and it works very fast.

If the answer is K, we should solve this sub-problem for $x = 1,2,3,\ldots,K$. Since K is no more than 1010000 (when $D = 2, N = 100$), you can write a program which is much faster than execution time limit.

Sample Code (C++) : https://beta.atcoder.jp/contests/abc100/submissions/2669995

Approach 2 : "Pencil and Paper" Approach

Actually, you don't have to do brute-forcing (searching all possible answer). Actually, this problem can be solved by pencil and paper (if D, N is fixed).

Let $a = 100^D$. The number which can be divided by 100 exactly D times is following:

$$a, 2a, 3a, 4a, \ldots, 99a, 101a, 102a, 103a, \ldots, 199a, 201a, 202a, 203a, \ldots, 299a, 301a, \ldots$$

This means, only $m \times 100^D$ (m is not a multiple of 100) can be divided by 100 exactly D times. Let's think about the solution based on this property.

Let q the quotient of $N - 1$ divided by 99. The answer of this problem will be between $(100q + 1) \times 100^D$ and $(100q + 99) \times 100^D$. Also, there are $99q$ numbers at most $100q \times 100^D$ and can be divided by 100 exactly D times. This means the answer will be ranked $N - 99q$ from smallest value which we confined the range of the answer a while ago. Therefore, the answer will be $(100q + N - 99q) \times 100^D$.

Simplifying this, the answer will be $(N - q) \times 100^D = \left(N - \left\lfloor \frac{N-1}{99} \right\rfloor\right) \times 100^D$.

Approach 2+α : "Crafty solution" using N≤100

Let $a = 100^D$. Seeing Approach 2, the first 100 numbers which can be divided by 100 exactly D times are $a, 2a, 3a, 4a, \ldots, 97a, 98a, 99a, 101a$. Therefore, assuming $N \leq 100$, the answer will be $Na = N \times 100^D$ if $N \leq 99$ otherwise $101a = 101 \times 100^D$.

Sample Code (C++) : https://beta.atcoder.jp/contests/abc100/submissions/2663837

Problem C: *3 or /2

Let's see the characteristics of "*3" or "/2". You can "multiply a_i by 3" unlimited time, but you cannot "divide a_i by 2", because you cannot choose this operation if a_i is an odd number.

So, the optimal solution seems like as follows:

- For each operation, divide exactly 1 number which is an even number, and multiple other numbers.

Anyway, how many times will you divide?

Let $f(x)$ be "the number of times that x can be divided by 2". For example, $f(8)$ = 3, $f(244)$ = 2 and $f(100)$ = 2.

If you choose "multiple a_i by 3", $f(a_i)$ should always be unchanged. Otherwise (if you choose "divide a_i by 2"), $f(a_i)$ should always decrease by 1.

Finally, if all $f(a_i)$ became zero (This means that all number is odd), you cannot do operation anymore.

So, the maximum number of operation is $f(a_1) + f(a_2) + \cdots + f(a_N)$. Since you can determine $f(x)$ with $O(log\, x)$ complexity, the complexity of this algorithm is $O(N\, log\, max\, a_i)$.

Sample Code: https://beta.atcoder.jp/contests/abc100/submissions/2670458

Bonus: You can solve the problem with complexity $O(N \log\log \max a_i)$. Because, if you use binary-search by the value of $f(x)$, you can determine $f(x)$ with complexity $O(\log\log x)$.

Problem D : Patisserie ABC

First, let's think about this easy sub-problem:

> *There are N cakes, which the value of beauty, tastiness, and popularity.*
> *You will choose M cakes.*
> Find the maximum possible value of (sum of beauty) + (sum of tastiness) + (sum of popularity).

This sub-problem can be solved easily. If cake i's beauty is x_i, tastiness is y_i, and popularity is z_i, the answer will be sum of $(x_k + y_k + z_k)$ when k is the chosen cakes. Therefore, choosing M cakes from higher $(x_i + y_i + z_i)$ is the optimal choice.

But the original problem is to maximize (absolute value of sum of beauty) + (absolute value of sum of tastiness) + (absolute value of sum of popularity). So, there's also a way to make some parameters' sum to negative.

Thus, you can understand easily, if you think about "maximizing force toward positive direction" or "maximizing force toward negative direction" for each parameter: beauty, tastiness, and popularity. Since there are three parameters, you can brute-force $2^3 = 8$ ways.

For example, if you want sum of beauty and sum of tastiness to maximize force toward positive, and want sum of popularity to maximize force toward negative, choosing M cakes from higher $(x_i + y_i - z_i)$ is the optimal choice.

The answer will be the maximum value among $2^3 = 8$ ways. It is possible to use recursion for brute-forcing, but using "brute-force with bit operations" will make easier implementation. Supposing that you sort a length-n array in $O(n \log n)$, the time complexity will be $O(N \log N) \times 8 = O(N \log N)$.

Sample Code (C++) : https://beta.atcoder.jp/contests/abc100/submissions/2670489

Bonus : This problem can be solved in linear time. First, you can calculate m-th biggest value in linear time. The algorithm called "Median of medians" enables calculating median in linear time, and then binary-searching the array, the m-th biggest value can be calculated in linear time. The sum of number from 1st to m-th biggest value will be calculated by, simply, "summing up the value larger than m-th biggest value", plus we need some adjustment.