ABC 114 解説

出題・解説: @evima0

2018 年 12 月 2 日

A: 753

正解までの道のりを細かく分割すると次の通りです。

1. 標準入力から整数 X を受け取る。
2. X から何らかの方法で答えとなる文字列 `YES` または `NO` を得る。
3. 得た文字列を標準出力に出す。

手順 1, 3 については practice contest 問題 A の言語別サンプルコードが参考になります。手順 2 については、if 文、“or” に相当する論理演算子と比較演算子を使うのが最も素直ですが、他の手も考えられます。C++ による素直な実装例と、Python3 によるやや気取った実装例を示します。

```cpp
#include <iostream>
using namespace std;
int main(){
    int X;
    cin >> X;
    string ans;
    if(X == 7 || X == 5 || X == 3){
        ans = "YES";
    }else{
        ans = "NO";
    }
    cout << ans << endl;
}
```

```python
print(['NO', 'YES'][[7, 5, 3].count(int(input()))])
```

B: 754

全体的な方針としては、ルンルンの行動の選択肢はそれほど多くないので、すべての選択肢を実際に試して最も 753 に近かった結果を採用すればよいです。以下、この方針を実装することについて書きます。

文字列 S の長さを N とし、S の最初の文字を 0 文字目、最後の文字を $N-1$ 文字目と呼ぶことにします。例えば $N=6$ のとき、ルンルンの行動の選択肢は「S の $0,1,2$ 文字目を取り出す」「$1,2,3$ 文字目を取り出す」「$2,3,4$ 文字目」「$3,4,5$ 文字目」の 4 通りです。一般的には、整数 $i=0,1,2,...,N-3$ のそれぞれに対して「S の $i,i+1,i+2$ 文字目を取り出す」という抜き出し方が存在します。

あとは、「S の $i,i+1,i+2$ 文字目を取り出し」て得られる 3 桁の整数をプログラム上でどのように求めるかが主な問題です。言語によって事情が異なり、Python などでは気軽に文字列から整数への変換を行えますが、C++ などではやや手間です。ASCII コード上で数字 '0', '1', ..., '9' が順番に並んでいるという性質を使って「自前」で変換する手も検討に値します。

そのほかに絶対値や「二つ（以上）の数のうち小さい方」などの部品も必要になりますが、言語の標準ライブラリを使うとスムーズです。以下の C++, Python3 による実装例を参考にしてください。

```cpp
#include <algorithm>    // min
#include <cmath>        // abs
#include <iostream>
using namespace std;
int main(){
    string S;
    cin >> S;
    int N = S.size();
    int ans = 999;
    for(int i = 0; i <= N - 3; ++i){
        int num = (c[i] - '0') * 100 + (c[i+1] - '0') * 10 + c[i+2] - '0';
        ans = min(ans, abs(num - 753));
    }
    cout << ans << endl;
}
```

```python
S = input()
print(min(abs(int(S[i:i+3]) - 753) for i in range(len(S) - 2)))
```

C: 755

N が小さければ、1 から N までの数をすべてチェックして七五三数を直接数えることができます。しかし与えられる N は最大で 999999999 (10 億 - 1) と大きく、2 秒という時間制限でこの方針は厳しいです。[*1]

入出力例 3 ($N = 999999999$, 出力: 26484) でひっそり示されているように、七五三数はそれほど多くありません。ですから、「$357, 375, 573, \ldots$」と七五三数の方を列挙して N 以下にある個数を数えるべきです。

そのためには、まず「$7, 5, 3$ のみを含む数」(333 や 57577 なども該当。**準七五三数**と呼ぶことにします) をすべて列挙し、そのうち「$7, 5, 3$ をすべて含む数」(七五三数) のみをカウントすると単純です。N 以下の準七五三数もそれほど多くない[*2]ので時間切れの心配はありません。

あとは、N 以下の準七五三数をどのように列挙するかが問題です。最も素直な手は再帰関数でしょう[*3]。「空っぽのもの」から始めて、「持っているものの後ろに $7, 5, 3$ を付け足して新たに 3 つのものを得る」ことを繰り返します。

具体的な実装について書きます。Python などの整数と文字列の相互変換が簡単な言語では、数を主に文字列として扱うと実装しやすいでしょう。C++ などで文字列を経由するべきかは悩むところです。経由せず整数を直接取り扱う場合、整数を「10 倍して 5 を足す」(例: $1234 \times 10 + 5 = 12345$. 後ろに 5 が付いた！) などの操作が役立ちます。以下の Python3 による実装例を参考にしてください。

```python
N = int(input())

def dfs(s):  # 文字列 s で始まる七五三数の個数
    if int(s) > N:
        return 0
    ret = 1 if all(s.count(c) > 0 for c in '753') else 0  # s 自体が七五三数なら +1
    for c in '753':
        ret += dfs(s + c)
    return ret

print(dfs('0'))  # 本当は dfs('') と書きたいが 4 行目でのエラーを防ぐため仕方なく
```

[*1] 実はこの方針で C++ などの言語で無理やり間に合わせることも不可能ではありませんが、かえって難しいです。
[*2] 9 桁の準七五三数は $3^9 = 19683$ 個、1〜9 桁の準七五三数をすべて合わせても 29523 個
[*3] 他の手段としては、3 進数を小さい方から列挙して $0 \to 3, 1 \to 5, 2 \to 7$ と変換する手もあります。

D: 756

$N = 100$ のとき $N! = 93326215443944152681...$ (158 桁) です。これを直接取り扱っても不毛で、効率的な枠組みで考える必要があります。

準備体操として、入出力例 2 の説明にある 32400 が確かに約数を 75 個持つことを確認しましょう。32400 を素因数分解 (リンク先は Wikipedia の同名の記事) すると $32400 = 2^4 3^4 5^2$ となります。よって $2^a 3^b 5^c (0 \leq a \leq 4, 0 \leq b \leq 4, 0 \leq c \leq 2)$ という形で表せる数はすべて 32400 の約数で、他にはありません。a の値は $0, 1, 2, 3, 4$ の 5 通り、b の値も 5 通り、c の値は 3 通りで、約数は全部で $5 \times 5 \times 3 = 75$ 個です。

同じように考えると、3 つの異なる素数 p, q, r を使って $p^4 q^4 r^2$ と表せる数はすべて七五数です。ほかに、$p^{14} q^4, p^{24} q^2, p^{74}$ という「パターン」の数も約数を $15 \times 5 = 25 \times 3 = 75$ 個持ちます。$75 = 3 \times 5 \times 5$ を分割することを考えると以上の 4 パターンがすべてで、これらをそれぞれ数えればよいです。

最も複雑な $p^4 q^4 r^2$ のパターンを考えます。まず $1, 2, \ldots, N$ をそれぞれ素因数分解して合算することで $N!$ を素因数分解します[*4]。素数 s に対して「$N!$ の素因数分解での s の指数」[*5]を $e(s)$ と書くと、$e(p) \geq 4, e(q) \geq 4, e(r) \geq 2$ が成立すれば $p^4 q^4 r^2$ は $N!$ の約数です。このような p, q, r の組合せの数を三重ループで数える[*6]か、もう少し考えて数式で求めることもできます。Python3 での後者の例を示します。[*7]

```python
N = int(input())
e = [0] * (N+1)
for i in range(2, N+1):
    cur = i
    for j in range(2, i+1):
        while cur % j == 0:
            e[j] += 1
            cur //= j

def num(m):  # e の要素のうち m-1 以上のものの個数
    return len(list(filter(lambda x: x >= m-1, e)))

print(num(75) + num(25) * (num(3) - 1) + num(15) * (num(5) - 1)
    + num(5) * (num(5) - 1) * (num(3) - 2) // 2)
```

[*4] N 自体は 100 以下と小さいので、愚直に割り算を繰り返せば十分です。エラトステネスの篩などの効率的なアルゴリズムは不要

[*5] 例えば $N = 4$ のとき $N! = 24 = 2^3 3$ なので $e(2) = 3, e(3) = 1$

[*6] ただし、例えば $2^4 3^4 5^2$ と $3^4 2^4 5^2$ が同じ数であることに注意

[*7] 余談: ここでは 75 という具体的な値に注目して解きましたが、「約数を x 個持つ数の個数」を x の小さい順に求めていく解法もあります (動的計画法と呼ばれる手法です)。