

ABC 168 解説

sheyasutaka

2020 年 5 月 17 日

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

A: ∴ (Therefore)

1 の位ごとに場合分けをしても解けますが、より簡潔に書けないでしょうか？

C/C++ や C#, Java などには switch 文というものがあり、これを使うことで問題文通りの記述を直感的に書けます (C での実装例を以下に示します).

余談ですが, Ruby では case 文がこれにあたり, もっと高機能です.

Listing 1 C での実装例

```
1 #include <stdio.h>
2
3 int main(void) {
4     int n;
5     scanf("%d", &n);
6
7     switch (n % 10) {
8         case 2:
9         case 4:
10        case 5:
11        case 7:
12        case 9:
13            puts("hon");
14            break;
15        case 0:
16        case 1:
17        case 6:
18        case 8:
19            puts("pon");
20            break;
21        case 3:
```

```
22         puts("bon");
23         break;
24     }
25
26     return 0;
27 }
```

B: ... (Triples Dots)

長さが K 以下かどうか判定し，上回ってれば問題文通りの文字列操作を行ってから出力すればいいです．

PHP には `truncate` というそのものずばりな機能があり，これを使うのも手です．

Listing 2 C++ での実装例

```
1 #include <iostream>
2 using std::cin;
3 using std::cout;
4
5 #include <string>
6 using std::string;
7
8 int main(void) {
9     int k;
10    string s;
11
12    cin >> k;
13    cin >> s;
14
15    if (s.size() > k) {
16        s = s.substr(0, k) + "...";
17    }
18
19    cout << s << '\n';
20
21    return 0;
22 }
```

Listing 3 Python3 での実装例

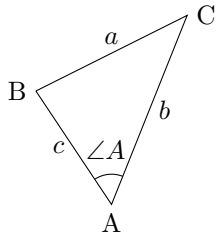
```
1 k = int(input())
2 s = input()
3
4 if len(s) > k:
5     s = s[:k] + "..."
6
7 print(s)
```

C: : (Colon)

2 端点の座標を求めてからその距離を計算するのは面倒です。これを使ってみましょう:

余弦定理

三角形 ABC について、 $a^2 = b^2 + c^2 - 2bc \cos \angle A$ が成り立つ。ここで、 a, b, c は $|BC|, |CA|, |AB|$ のことを指す。



時針・分針の長さを b, c , 時針・分針の間の角度を $\angle A$ として上の式に代入すれば、求める長さ (の 2 乗) が求まります。角度については、「時針が $H + \frac{M}{60}$ 時間で動いた角度」から「分針が M 分で動いた角度」を引けばよいです (C++ 標準の \cos 関数の場合、 mod をとる必要はありません)。

Listing 4 C での実装例

```
1 #include <stdio.h>
2 #include <math.h>
3
4 #define PI 3.14159265358979323846264338327950L
5
6 int main(void) {
7     int a, b, h, m;
8     scanf("%d%d%d%d", &a, &b, &h, &m);
9
10    long double rad = PI * 2 * (
11        (long double)h / 12.0 + ((long double)m / 60.0) / 12.0 - (long
12            double)m / 60.0);
13
14    long double rsq = (long double)(a * a + b * b) - (long double)(2 * a *
15        b) * cosl(rad);
16
17    printf("%20.20Lf\n", sqrtl(rsq));
18 }
```

D: .. (Double Dots)

ある部屋から部屋 1 に移動するために廊下を通る最小の回数のことを、その部屋の「深さ」と呼ぶことにします。特に、部屋 1 の深さだけが 0 です。

すると、以下の重要な事実が成り立ちます。

— < ! > < ! > < ! > < ! > < ! > < ! > —

深さ $d + 1$ の部屋には、深さ d の部屋が少なくとも 1 つ繋がっている。

証明は簡単です (そうでない部屋があると、直感的に嫌な気分になります)。このことから、深さ $d + 1$ の部屋の道しるべが深さ d の部屋を指すようにすれば、常に目標を達成できることがわかります。

実装の際には、幅優先探索で各部屋の深さを求めてから、辺を 1 つずつ見て行って随時更新すれば楽です。

E: • (Bullet)

$(A_i, B_i) = (0, 0)$ のイワシは他のどの個体とも仲が悪いので、「 $(A_i, B_i) = (0, 0)$ なイワシをどれか 1 匹だけ選ぶ」「 $(A_i, B_i) = (0, 0)$ なイワシをどれも選ばない」のいずれかです。勿論、前者の選び方はそのようなイワシの個数に等しいので、以下では $(A_i, B_i) = (0, 0)$ なイワシを除外して考えます。

イワシの「傾き」を $\frac{A_i}{B_i}$ として、これを既約分数で表すことを考えます。具体的には、次のように決めます。

- $B_i = 0$ のとき、傾きは $1/0$ 。
- $B_i > 0$ のとき、傾きは $\frac{A_i}{\gcd(A_i, B_i)} / \frac{B_i}{\gcd(A_i, B_i)}$ 。
- $B_i < 0$ のとき、傾きは $-\frac{A_i}{\gcd(A_i, B_i)} / -\frac{B_i}{\gcd(A_i, B_i)}$ 。

ちょっとした場合分けから、傾きと仲の悪さの関係について次のことが言えます。

- 傾き $1/0$ のイワシと仲が悪いのは、傾き $0/1$ のイワシ全てのみ。逆もまた然り。
- 傾き a/b ($a, b \neq 0$) のイワシと仲が悪いのは、傾き $-b/a$ (を a の符号で通分したもの) のイワシ全てのみ。逆もまた然り。

たとえば以下の傾きが、互いに ”仲の悪いつがい” です。

$$\begin{aligned} 1/0 &\longleftrightarrow 0/1 \\ 5/3 &\longleftrightarrow -3/5 \\ -2/1 &\longleftrightarrow 1/2 \end{aligned}$$

したがって、”仲の悪いつがい” になる傾きのペア (高々 N 通り) 全てについて、連想配列などを使って各傾きになるイワシの個数が求まれば、その後は基礎的な数え上げの範疇です。オーバーフローには気を付けてください。

A_i, B_i の制約が大きいため、傾きを有理数の代わりに実数で表現しようとする (long double でも) 精度が足りないおそれがあります。

F: . (Single Dot)

x 座標・ y 座標それぞれを重複を除いてソートし、十分なサイズの 2 次元グリッド上に各線分を刻み込んでから BFS すれば、 $\mathcal{O}(NM)$ 時間となって十分間に合います。