一中電研 37th 第二學期社內賽 題目解析

nevikw39

June 25, 2020

Abstract

1 遞迴縮寫 (acronym)

關鍵 string

1.1 題目

所謂「遞迴縮寫」是指一組字串的縮寫恰好參照到他本身，例如 `GNU Not Unix` 的縮寫是 `GNU` …

在資訊社群中，電神們有個慣例是傾向於使用「遞迴縮寫」的命名法來表達他們的幽默。

本題的任務是，請你檢查一組字串是否有可能是關於首字的「遞迴縮寫」。

1.1.1 輸入

一些以空白隔開的字串，數量及長度皆小於 25。

1.1.2 輸出

如果輸入的字串們可以存在遞迴縮寫，請輸出 `o´ó` 否則 `QQ`。

1.2 解析

1.3 參考程式碼

1.3.1 C++

```cpp
#include <bits/stdc++.h>
using namespace std;
inline char mytolower(const char &x) // convert a
    character to lower case by bitwise operation
{
    return x | 1 << 5;
}
int main()
{
    ios::sync_with_stdio(false);
```

```
    cin.tie(nullptr);
    string s;
    cin >> s;
    for (int i = 1, l = s.length(); i < l; i++)
    {
        string t;
        cin >> t;
        if (mytolower(t[0]) != mytolower(s[i]))
        {
            cout << "QQ\n";
            return 0;
        }
    }
    getline(cin, s);
    cout << (s.empty() ? "o'_'o\n" : "QQ\n");
    return 0;
}
```

1.3.2 Python

```
import sys


def main():
    lst = sys.stdin.buffer.readline().split()
    print("o'_'o" if len(lst) == len(lst[0]) and all(
        [e[0] | 1 << 5 == lst[0][i] | 1 << 5 for i, e
           in enumerate(lst)]) else "QQ")


if __name__ == "__main__":
    main()
```

2 旋轉矩陣 (rotate)

關鍵 GCD, LCM

2.1 題目

電電寫數學考卷遇到這麼一題：令一矩陣 A 表一平面上的線性變換 $\begin{pmatrix} \frac{-\sqrt{3}}{2} & \frac{-1}{2} \\ \frac{1}{2} & \frac{-\sqrt{3}}{2} \end{pmatrix}$，試問任一點 $P(x, y)$ 最少須經過多少次此變換後方回到原本的位置。

聰明的電電馬上想到 $A = \begin{pmatrix} \cos 150° & -\sin 150° \\ \sin 150° & \cos 150° \end{pmatrix} = R_{150°}$ 即一個旋轉 150 度的變換，那麼只要旋轉成「一周角」的倍數就是旋轉一圈相當於沒有變換。因此，你的任務是幫電電計算最少旋轉幾次後回到原本的位置。

不過，電電國的角度單位有很多種，「一周角」並不總是 360 度。

2.1.1 輸入

兩個整數 a, b 皆 $< 2^{31}$，分別代表旋轉矩陣的度數及「一周角」的度數。

2.1.2 輸出

請輸出最少旋轉幾次後回到原本的位置。

2.2 解析

本題怕太水所以故意把題目敘述打很複雜，但是真的是考數學得到的靈感 XDD

很顯然本題所求為 $\frac{lcm(a,b)}{a} = \frac{\frac{a*b}{gcd(a,b)}}{a} = \frac{b}{gcd(a,b)}$。

2.3 參考程式碼

2.3.1 C++

```cpp
#include <bits/extc++.h>
using namespace std;
using namespace __gnu_pbds;
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int32_t a, b;
    cin >> a >> b;
    cout << b / __gcd(a, b) << '\n';
    return 0;
}
```

2.3.2 Python

```python
import sys
import math


def main():
    a, b = map(int, sys.stdin.buffer.readline().split
        ())
    sys.stdout.buffer.write(str(b // math.gcd(a, b)).
        encode())


if __name__ == "__main__":
    main()
```

3 割草終戰 (recall)

關鍵 條件

3.1 題目

由於 I4S○ 菸粉綠蛆韓黑網軍蟑螂及國家機器動得非常厲害，我們先總統 蔣公在世、人類的救星、世界的偉人、自由的燈塔、民族的長城、宇宙的征服者，百年一見的政治鬼才 — 高譚市長韓總雞，遭遇地球文明史上最大的危機：罷免案。

依據《公職人員選舉罷免法》第 90 條：

> 罷免案投票結果，有效同意票數多於不同意票數，且同意票數達原選舉區選舉人總數四分之一以上，即為通過。有效罷免票數中，不同意票數多於同意票數或同意票數不足前項規定數額者，均為否決。

因為高譚市民非常非常多，有可能超過五百萬的五百萬倍，所以想請你幫忙寫個程式判斷罷免通過與否。

3.1.1 輸入

三整數 n, a, b 分別代表全體選舉人總數，有效同意、不同意票數。

$$0 \le n < 2^{64}, 4 \times a < n, a + b \le n$$

3.1.2 輸出

首先請就罷免結果，輸出 `!!666` 或 `QQ` 。

接著請輸出「沉默不出來投票的韓粉」之人數。

3.2 解析

本題雖然就是個水題，但是有不少陷阱。

首先，由於 n 的範圍 2^{64} ，因此必須使用 `uint64_t`。其次，同意票數需大於選舉人總數之四分之一，但須注意 C++ 的除法總是向零取整，假若寫成 `a >= n / 4` 則會存在誤差。

3.3 參考程式碼

3.3.1 C++

```cpp
#include <stdio.h>
#include <stdint.h>   // uint64_t
#include <inttypes.h> // SCNu64
int main()
{
    uint64_t n, a, b;
                                          // unsigned
        long long is far less portable.
```

```
    scanf("%" SCNu64 "%" SCNu64 "%" SCNu64, &n, &a, &b
        ); // you may use %164u or %llu on different os
        , so it's better to use macro.
    puts(a << 2 >= n && a > b ? "!!666" : "QQ~~");
    printf("%" PRIu64, n - a - b);
    return 0;
}
```

3.3.2 Python

```
import sys


def main():
    n, a, b = map(int, sys.stdin.buffer.readline().
        decode().split())
    sys.stdout.buffer.write(
        (("!!666\n%d" if a >= n / 4 and a > b else "QQ
            ~~\n%d") % (n - a - b)).encode())


if __name__ == "__main__":
    main()
```

4 病毒擴散 (pow)

關鍵 迴圈、（快速冪）

4.1 題目

很久很久以前，在遙遠的銀河系…*(A long time ago in a galaxy far far away...)*

多久以前呢？？應該是於「雅汶戰役」前 2087 年 *(2087 BBY)* 吧，在杳無人煙的帝國祕密基地、岩漿星球穆斯塔法 *(Mustafar)* 發現惹一種冠狀病毒，*GHO* 銀河衛生組織將之暫時命名為 **87-nCoV**。

已知這種奇怪的病毒會不斷的自行分裂，現在給你此病毒每次分裂後的個數 r，求在 t 個週期後病毒數量會變為原來的幾倍？？

4.1.1 輸入

輸入僅有一行，兩個整數 r, t 以空白分隔，分別代表每一病毒每次分裂後的個數及經過多少週期。

4.1.2 輸出

輸出最後病毒數量變為原來的幾倍。因為答案或許很大，所以請對 $10^9 + 7$ 取模。

4.2 解析

本題分為兩個小題。

70% $r < 1000000000, t < 10$ 在 `long` 內可以迴圈輕鬆解決。

30% $r < 1000000000, t < 1000000000$ `C++` 須自行實做快速冪，`Python` 若善用內建函式則秒殺。

4.3 參考程式碼

4.3.1 C

```c
#include <stdio.h>
#include <stdint.h>
#include <inttypes.h>
const int M = 1e9 + 7;
uint64_t mulmod(uint64_t x, uint64_t n)
{
    uint64_t y = 0;
    while (x)
    {
        if (x & 1)
            y = (y + n) % M;
        x >>= 1;
        n = (n << 1) % M;
    }
    return y;
}
uint64_t binexp(uint64_t x, uint64_t n)
{
    uint64_t y = 1;
    while (n)
    {
        if (n & 1)
            y = mulmod(y, x); // equal to y = (y * x)
                % M
        x = mulmod(x, x);     // similarly
        n >>= 1;
    }
    return y;
}
int main()
{
    uint64_t r, t;
    scanf("%" SCNu64 "%" SCNu64, &r, &t);
    printf("%" PRIu64 "\n", binexp(r, t));
    return 0;
}
```

4.3.2 Python

```python
import sys


def main():
    r, t = map(int, sys.stdin.buffer.readline().split
        ())
    sys.stdout.buffer.write(str(pow(r, t, 10**9 + 7)).
        encode())


if __name__ == "__main__":
    main()
```

5 二次曲線美化 (conic)

關鍵 迴圈、條件

5.1 題目

電電最近學到二次曲線，在電腦上打惹許多二次曲線的一般式 ($ax^2 \pm bxy \pm cy^2 \pm dx \pm ey \pm f; a, b, c, d, e, f \in \mathbb{N}$)。可是寫的時候因為很趕很隨意，導致式子擠在一起很不方便閱讀，因此想請你寫個程式幫他美化重新排版。

排版的規則如下：

1. 各項必須依照降冪及字典順序出現
2. 指數應當出現在 *caret* 字元 '^' 後方
3. 常數項就只有常數
4. 唯具有非零係數的項可以出現，除非每項係數皆為 0，則常數項可以出現
5. 空格僅在二元運算子 + —加— 和 − —減— 的兩側出現
6. 若領導係數為正則毋需性質符號，反之應輸出一負號
7. 負項被視為減去正項，首項例外
8. 係數 ± 1 只有在常數項出現

5.1.1 輸入

輸入僅有一行，六個整數 a, b, c, d, e, f 以空白分隔，分別代表該二次曲線之各項。

5.1.2 輸出

輸出符合規則的二次多項式。

5.2 解析

本題是個比較複雜的條件判斷水題，沒有任何技巧，就是需要耐心而已。

5.3 參考程式碼

5.3.1 C

```c
#include <bits/stdc++.h>
using namespace std;
int inline myabs(int x) // absolute value function
    based on bitwise operation, which is a little bit
    efficient
{
    return (x ^ (x >> 31)) - (x >> 31);
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    const string plus[] = {"", " + "}, minus[] = {"-",
        " - "}, symbols[] = {"x^2", "xy", "y^2", "x",
       "y"};
    array<int, 6> a;
    for (int &i : a)
        cin >> i;
    bool flag = false;
    for (int i = 0; i < 5; i++)
        if (a[i])
        {
            cout << (a[i] > 0 ? plus[flag] : minus[
               flag]) << (myabs(a[i]) > 1 ? to_string(
               myabs(a[i])) : "") << symbols[i];
            flag = true;
        }
    if (a[5])
    {
        if (flag)
            cout << (a[5] > 0 ? " + " : " - ") << abs(
               a[5]);
        else
            cout << a[5];
    }
    else if (!flag)
        cout << '0';
    return 0;
}
```

5.3.2 Python

```python
import sys


def main():
    a = list(map(int, sys.stdin.buffer.readline().
        decode().split()))
    plus = ["", " + "]
    minus = ["-", " - "]
    symbols = ["x^2", "xy", "y^2", "x", "y"]
    flag = False
    for i in range(5):
        if a[i]:
            sys.stdout.buffer.write(((plus[flag] if a[
                i] > 0 else minus[flag]) +
                                     f"{abs(a[i]) if
                                         abs(a[i]) > 1
                                         else ''}" +
                                         symbols[i]).
                                         encode())
            flag = True
    if a[5]:
        if flag:
            sys.stdout.buffer.write(
                ((" + %d" if a[5] > 0 else " - %d") %
                    abs(a[5])).encode())
        else:
            sys.stdout.buffer.write(f"{a[5]}".encode()
                )
    elif not flag:
        sys.stdout.buffer.write('0'.encode())


if __name__ == "__main__":
    main()
```

6 數字朗讀 (num)

關鍵 迴圈、遞迴

6.1 題目

電電欲令電腦朗讀出數字，他現在已經有英文轉語音 *(TTS)* 的 *API*

在英文中與中文不同，差兩次以上的位數並不需要「零」或 “end” （比賽結束後你可以請 Google 小姐念給你聽）。此外，正式的寫法如支票，並不會用 “a hundred” 等等而應該是 “one hundred”。

6.1.1 輸入

僅有一整數 $n < 2^{31}$。

6.1.2 輸出

n 在英文的念法。

6.2 解析

6.3 參考程式碼

6.3.1 C++

```
#include <bits/stdc++.h>
using namespace std;
const string num_to_words(int x)
{
    static const string digits[] = {"", "one", "two",
        "three", "four", "five", "six", "seven", "eight
        ", "nine", "ten", "eleven", "twelve", "thirteen
        ", "fourteen", "fifteen", "sixteen", "seventeen
        ", "eighteen", "nineteen"},
                        tys[] = {"", "", "twenty", "
                            thirty", "forty", "fifty",
                            "sixty", "seventy", "eighty
                            ", "ninety"};
    if (x < 0)
        return "minus " + num_to_words(-x);
    if (x == 0)
        return "zero";
    if (x < 20)
        return digits[x];
    if (x < 100)
        return tys[x / 10] + ((x % 10) ? '-' +
            num_to_words(x % 10) : "");
    if (x < 1000)
        return digits[x / 100] + " hundred" + ((x %
            100) ? ' ' + num_to_words(x % 100) : "");
    if (x < 1000000)
        return num_to_words(x / 1000) + " thousand" +
            ((x % 1000) ? ' ' + num_to_words(x % 1000)
            : "");
    if (x < 1000000000)
        return num_to_words(x / 1000000) + " million"
            + ((x % 1000000) ? ' ' + num_to_words(x %
            1000000) : "");
    if (x < 1000000000000)
        return num_to_words(x / 1000000000) + "
            billion" + ((x % 1000000000) ? ' ' +
            num_to_words(x % 1000000000) : "");
```

```
    return "error";
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int n;
    cin >> n;
    cout << num_to_words(n) << '\n';
    return 0;
}
```

6.3.2 Python

```
import sys


def num_to_words(x: int):
    digits = ["", "one", "two", "three", "four", "five
        ", "six", "seven", "eight", "nine", "ten", "
        eleven",
              "twelve", "thirteen", "fourteen", "
                 fifteen", "sixteen", "seventeen", "
                 eighteen", "nineteen"]
    tys = ["", "", "twenty", "thirty", "forty",
           "fifty", "sixty", "seventy", "eighty", "
              ninety"]
    if x < 20:
        return digits[x]
    if x < 100:
        return tys[x // 10] + (('-' + digits[x % 10])
            if x % 10 else '')
    if x < 1000:
        return digits[x // 100] + " hundred" + (' ' +
            num_to_words(x % 100)) if x % 100 else ''


def main():
    n = int(sys.stdin.buffer.readline())
    if not n:
        sys.stdout.buffer.write("zero".encode())
    if n < 0:
        sys.stdout.buffer.write("minus".encode())
        n = -n
    if n >= 1000000000:
        sys.stdout.buffer.write(
            (num_to_words(n // 1000000000) + " billion
               ").encode())
        n %= 1000000000
```

```
        if n:
            sys.stdout.buffer.write(' '.encode())
    if n >= 1000000:
        sys.stdout.buffer.write(
            (num_to_words(n // 1000000) + " million").
                encode())
        n %= 1000000
        if n:
            sys.stdout.buffer.write(' '.encode())
    if n >= 1000:
        sys.stdout.buffer.write(
            (num_to_words(n // 1000) + " thousand").
                encode())
        n %= 1000
        if n:
            sys.stdout.buffer.write(' '.encode())
    sys.stdout.buffer.write(num_to_words(n).encode())


if __name__ == "__main__":
    main()
```

7

關鍵

7.1 題目

7.1.1 輸入

7.1.2 輸出

7.2 解析

7.3 參考程式碼

7.3.1 C++

7.3.2 Python

8

關鍵

8.1 題目

8.1.1 輸入

8.1.2 輸出

8.2 解析

8.3 參考程式碼

8.3.1 C++

8.3.2 Python

9

關鍵

9.1 題目

9.1.1 輸入

9.1.2 輸出

9.2 解析

9.3 參考程式碼

9.3.1 C++

9.3.2 Python

10

關鍵

10.1 題目

10.1.1 輸入

10.1.2 輸出

10.2 解析

10.3 參考程式碼

10.3.1 C++

10.3.2 Python