Chapter 3
枚舉
Enumerate I

台南女中資訊研究社 38th C++ 進階班課程

TNGS IRC 38th C++ Advanced Course

講師：陳俊安 Colten

什麼是枚舉

- 暴力
- 對，就是暴力
- 而且你可以當作是基本上不用動腦的方式之一

枚舉這個技巧

- 非常難掌握
- 雖然枚舉的概念很簡單
- 但往往我們在寫題目的時候，有時可以枚舉的東西自己完全不會發現
- 掌握枚舉這一個技巧的好處
  - 可以用最無腦的方式拿到 AC，超棒的對吧

枚舉暖身：APCS 2020 10 月 第一題 人力分配

有一個公司有 n 個員工，還有兩個工廠。如果工廠一與工廠二分別有 X_1 與 X_2 個員工，兩個工廠的收益 Y_1, Y_2 分別會是

$$Y_1 = A_1 \times X_1^2 + B_1 \times X_1 + C_1$$

$$Y_2 = A_2 \times X_2^2 + B_2 \times X_2 + C_2$$

請你考慮所有分配員工的方式，找出收益最大的組合，輸出最大收益。

注意，每個員工皆需分配到其中一個工廠。

枚舉暖身：APCS 2020 10 月 第一題 人力分配

- 可以發現 n 的範圍很小，且每一個人一定要分配到其中一個工廠
- 而所有的分配情況是：
- (0,n), (1,n-1), (2,n-2), (3,n-3), … , (n,0)
- 總共會有 n 種可能，因此我們就枚舉第一個工廠分配的人數
- 如果第一個工廠給了 i 個人，那麼第二個工廠就會有 n - i 個人
- 把所有情況的結果都算出來，取最好的答案
- 時間複雜度 O(n)

枚舉暖身：台南女中資研社 37th 期末練習賽 pE. 倒水問題

- 一開始有一杯 N 豪升的水，有兩種操作
  - 把這一杯水倒出 a 毫升
  - 把這一杯水倒出 b 毫升
  - 這兩個操作可以各使用 10 次
- 最後剩下的水不能小於 K 毫升
- 如果最後的水剩下 X，請設計一個程式找出 X - K 的最小可能
- 並輸出你兩個操作各使用了幾次
  - 如果有多種策略可以讓 X - K 最小，輸出第一種操作使用比較多的

枚舉暖身：台南女中資研社 37th 期末練習賽 pE. 倒水問題

- 範圍

測資範圍限制：

- $1 \leq K \leq N \leq 10^4$
- $1 \leq a, b \leq 500$

枚舉暖身：台南女中資研社 37th 期末練習賽 pE. 倒水問題

- 對，我們其實就枚舉這兩個操作要使用各幾次即可
- 只是在算答案的時候記得，如果有多組操作方式都可以讓答案最好
- 那麼要使用第一種操作使用比較多的 (題目要求)

```
12    int now_best = n - k , ans1 = 0, ans2 = 0; // 最壞的情況就是什麼操作都不用
13
14    for(int i=10;i>=0;i--)
15    {
16        for(int j=10;j>=0;j--)
17        {
18            int total = n - a * i - b * j;
19
20            if( total >= k && total - k < now_best ) // 因為我把第一種操作大 ~ 小了，因此如果結果一樣，不會是答案
21            {
22                now_best = total - k;
23                ans1 = i;
24                ans2 = j;
25            }
26        }
27    }
28
29    cout << ans1 << " " << ans2 << " " << now_best << "\n";
```

有限度的枚舉：台南女中資研社 38th 教學上機考 pA. 飲品調配

為了研發出這款飲品，店長在這方面下了非常多的苦工，他希望這個飲品只由 3 種材料來完成，且這三種材料在飲品當中的總數量只會剛好有 N 份。

現在已經確定好了飲品製作的規則，接下來店長想要找到一個完美的比例來讓這款飲品更好喝，每杯飲料被製作出來都會有一個好喝程度，已知如果我們在一杯飲料放入 a 份第一種材料、b 份第二種材料、c 份第三種材料 $(a+b+c=N,\ 0\le a,b,c\le N)$，那麼這杯飲料的好喝程度會是 $2022+|b-c|+ab+bc+c^2-|b^2-a^2|$。

店長想要知道在 3 種材料總數量為 N 的情況下，飲料的好喝程度的最大值是多少，請你設計一個程式幫忙計算出來吧！

別忘記了！這 3 種材料的分配方式都必須符合 $a+b+c=N,\ 0\le a,b,c\le N$ 且 a,b,c 都必須是非負整數。

測資範圍限制

- $1\le N\le 5000$

有限度的枚舉：台南女中資研社 38th 教學上機考 pA. 飲品調配

- 如果我們枚舉三種材料個要使用幾次
- 時間複雜度為 O(N^3)
- N^3 = 5000 * 5000 * 5000 > 10^9
- 肯定吃一個大大的 TLE

有限度的枚舉：台南女中資研社 38th 教學上機考 pA. 飲品調配

- 題目有說 a + b + c = N, 這意味著你只要知道 a,b 的數量，就可以知道 c 的數量
- 因此我們不需要枚舉 a,b,c，只要枚舉 a,b 就可以了
- 如此一來時間複雜度就變成了 O(N^2)

```
 7 signed main(void)
 8 {
 9     int n;
10     cin >> n;
11
12     for(int a=0;a<n;a++)
13     {
14         for(int b=0;b<n;b++)
15         {
16             int c = n - ( a + b );
17             if( c < 0 ) break;
18
19             // do something
20         }
21     }
22
23     return 0;
24 }
```

有限度的枚舉：Codeforces Round #702 pC. Sum of Cubes

C. Sum of Cubes

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a positive integer x. Check whether the number x is representable as the sum of the cubes of two positive integers.

Formally, you need to check if there are two integers a and b ($1 \le a, b$) such that $a^3 + b^3 = x$.

For example, if $x = 35$, then the numbers $a = 2$ and $b = 3$ are suitable ($2^3 + 3^3 = 8 + 27 = 35$). If $x = 4$, then no pair of numbers a and b is suitable.

Input

The first line contains one integer t ($1 \le t \le 100$) — the number of test cases. Then t test cases follow.

Each test case contains one integer x ($1 \le x \le 10^{12}$).

Please note, that the input for some test cases won't fit into 32-bit integer type, so you should use at least 64-bit integer type in your programming language.

Output

For each test case, output on a separate line:

- "YES" if x is representable as the sum of the cubes of two positive integers.
- "NO" otherwise.

You can output "YES" and "NO" in any case (for example, the strings yEs, yes, Yes and YES will be recognized as positive).

有限度的枚舉：Codeforces Round #702 pC. Sum of Cubes

- 如果要使 a^3 + b^3 = x，那麼如果 a^3 = i，那 b^3 = x - i
- 因此我們可以去枚舉 a^3 是多少，並看看 x - a^3 是否屬於 完全立方數
- 看看是否為完全立方數可以善用 cbrt () 函數，開立方根

經典問題：列出 x 的所有因數（x <= 10^9）

- 在還沒有學過之前，相信大家都直接枚舉 1 ~ x 所有數字
- 但現在會 TLE 了，我們該怎麼處理？
- 這其實也是有限度的枚舉哦！

經典問題：列出 x 的所有因數（x <= 10^9）

- 如果 a 是 x 的因數，那麼表示 x / a 也是 x 的因數
- 因此我們在枚舉比較小的數字的時候，其實就也可以把大的因數計算出來
  - 2 是 10 的因數，我們可以順便抓出 10 / 2 = 5 這一個因數

經典問題：列出 x 的所有因數（x <= 10^9）

- 假設 a * b = x，且保證 a <= b
- 我們枚舉 a，那麼我們需要枚舉到什麼程度？
  - 根號 x
  - 如果 a > 根號 x，a 在之前一定已經被我們計算出來了
  - 根號 x * 根號 x = x
- 如此一來時間複雜度就會是 O(根號 x)

遞迴枚舉：Gunjyo 與骰子 II

Output: standard output

本題為 Gunjyo 與骰子 II 的簡單版本，簡單與困難之間的差異在於輸入的方式，請詳細閱讀題目敘述。

Gunjyo 學姊身為大家的學姊所以她有 n 個骰子，每個骰子都長的一樣，點數只會有三種可能，分別為 a_1, a_2, a_3。

接下來 Gunjyo 會一次骰這 n 個骰子，她想要知道所有的骰子總和能湊出 $\leq x$ 的組合會有幾種。

特別注意的是，不同顆骰子被視為不同的組合。

Input

只有一組資料。

第一行輸入一個四個整數 n, a_1, a_2, a_3, x。

測資範圍限制

- $1 \leq n \leq 18$
- $1 \leq a_1, a_2, a_3 \leq 1000$
- $1 \leq x \leq 18000$

Output

輸出一個整數，表示答案。

遞迴枚舉：Gunjyo 與骰子 II

- 你應該不會想要寫 18 層迴圈對吧
- 所以這個時候高級版的迴圈 遞迴 就派上用場了
- 我們利用遞迴把每一個骰子的狀態都枚舉出來

```
16
17 void solve(int idx,int sum,int x) // 目前在枚舉第 idx + 1 個骰子 , 總和為 sum , 題目要求 ≤ x 的數量
18 {
19     if( idx == n )
20     {
21         if( sum ≤ x ) ans++;
22
23         return;
24     }
25
26     for(int i=0;i<3;i++) solve(idx+1,sum+a[i],x); // 枚舉骰出來的點數
27 }
```

```
33     int x;
34     cin >> n >> a[0] >> a[1] >> a[2] >> x;
35
36     solve(0,0,x);
37
38     cout << ans << "\n";
```

枚舉關鍵點：奇數偶數全排列（Hard）

PROBLEMS SUBMIT CODE MY SUBMISSIONS STATUS STANDINGS ADM. EDIT CUSTOM INVOCATION

H2. 奇數偶數全排列 (Odd and even permutation)【Hard Version】

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Easy 與 Hard 的差別在於分數的計算方式，因此請都詳細閱讀兩種版本的題目

Colten 很喜歡只有 3 個數字的序列，除此之外，他還會對這種序列再細分，並幫這些序列打分數。

- 如果這個序列的奇數剛好有 2 個則分數加 1 分。
- 如果這個序列的偶數剛好有 2 個則分數加 2 分。
- 如果這個序列的奇偶互不相臨則分數加 5 分。
- 如果這個序列第 i $(1 \leq i \leq 2)$ 個位置是奇數，那麼如果第 $i+1$ 個位置也是奇數，則加 5 分。
- 如果這個序列第 i $(1 \leq i \leq 2)$ 個位置是偶數，那麼如果第 $i+1$ 個位置也是偶數，則加 5 分。
- 如果這個序列第 1 個位置的數字加上第 3 個位置的數字大於第 2 個位置的數字的話則加 5 分。
- 如果這個序列的任意兩項相加，都會大於序列中的所有數字的話，則分數加 7 分。

為了得到分數越高的序列，Koying 決定幫 Colten 改變序列的順序，因此他可以對序列做數次操作 (也可以完全不做任何操作)。

Koying 每次操作可以任意選擇兩個位置，並將這兩個位置的數字交換位置。

現在 Colten 想知道 Koying 排出的序列最高的分數可能是多少。

因此請你設計一個程式，告訴 Colten 答案吧！

Input

只有一筆資料。

輸入只有一行，依序輸入 3 個正整數，依序代表序列原本由左到右排列的三個數字。

保證序列裡的數字不超過 1000。

NHDK Ten Point Round Group

Public

Manager

Ten Point Round #12 (Div. 3)

Finished

Judgem. status: **100%** (0+1779=1779)

Contest Manager

→ About Time Scaling

This contest uses time limits scaling policy (depending on a programming language). The system automatically adjusts time limits by the following multipliers for some languages. Despite scaling (adjustment), the time limit cannot be more than 30 seconds. Read the details by the link.

→ Virtual participation

Virtual contest is a way to take part in past contest, as close as possible to participation on time. It is supported only ICPC mode for virtual contests. If you've seen these problems, a virtual contest is

例題 7：奇數偶數全排列（Hard）

- 關鍵點在於數字的排列方式，而不是我們要怎麼使用最佳策略
- 所以我們就只要把所有排列組合枚舉一次就可以了
- 把計算分數的東西寫成一個 Function 會比較乾淨一點

全排列枚舉

- 把所有排列的情況都枚舉出來，選出最好的答案
- 更直白的來說的話，就是每一種排列都試試看
- 其實剛剛介紹到的 奇數偶數全排列（Hard）也算是一種全排列枚舉

使用遞迴的方式把所有排列都列出來

```cpp
void solve(int n)
{
    if( n == 4 )
    {
        for(int i=1;i≤4;i++) cout ≪ ans[i] ≪ " ";
        cout ≪ "\n";
        return;
    }

    for(int i=1;i≤4;i++)
    {
        if( used[i] == 0 ) // 當前這組還沒有用過 i 這個數字
        {
            used[i] = 1; // 第 ( n + 1 ) 個數字我們用 i 試試看，把 i 標記成我們用過了
            ans[n+1] = i;
            solve(n+1);
            used[i] = 0; // 當前這一個位置要試試看擺其他數字了，所以現在 i 變成我們還沒用過了
        }
    }
}
signed main()
{
    solve(0);

    return 0 ;
}
```