***Tìm chữ số khác không cuối cùng của n!(giai thừa)***

**Ví dụ:**

* Với n = 5, kết quả lastDigitDiffZero(n) = 2.  
  5! = 1 · 2 · 3 · 4 · 5 = 120.
* Với n = 6, kết quả lastDigitDiffZero(n) = 2.  
  6! =1 · 2 · 3 · 4 · 5 · 6 = 720.
* Với n = 10, kết quả lastDigitDiffZero(n) = 8.  
  10! = 3628800.

**Đầu vào/Đầu ra**

* **[Thời gian chạy] 0.5 giây**
* **[input] integer n**  
  *Điều kiện:*  
  1 ≤ n ≤ 10^6.
* **[output] integer**Chữ số cuối cùng khác 0 của n!

**Gợi ý:**

* n có thể rất lớn, nên ko thể tính được n! theo cách thông thường
* Hãy chỉ lưu trữ vài chữ số khác 0 cuối cùng của n là được

**Lý thuyết :**

* Một số X như thế nào thì chia hết cho 10 ? Đó là khi X phân tích ra thừa số nguyên tố, sẽ luôn chứa 2 thừa số là 2 và 5 (Vì 10 = 2 \* 5, mà 2 và 5 nguyên tố cùng nhau)
* Một số X*có k chữ số 0 tận cùng*, tức là X*chia hết cho 10k, và không chia hết cho 10k+1.*Khi đó, bậc lũy thừa của 2 và 5, giá trị nhỏ hơn sẽ phải đúng bằng k (10k= 2k \* 5k)
* Vậy ta cần xác định số chữ số 0 của số X, thì chỉ cần xác định bậc lũy thừa của 2 và 5 mà X có, rồi lấy ra số nhỏ hơn
* **Ví dụ** **:**X = 27 \* 3 \* 55 ⇒ X có 5 *chữ số 0 tận cùng*. Điều đó đúng, vì giá trị X = 1200000
* Code minh họa:

int NumOf0s(int X) {

int num2 = 0, num5 = 0;

// tinh bac luy thua cua 2

while (X % 2 == 0) {

num2++;

X /= 2;

}

// tinh bac luy thua cua 5

while (X % 5 == 0) {

num5++;

X /= 5;

}

return min(num2, num5);

}

* Chúng ta đã tìm hiểu được cách đếm số chữ số 0 tận cùng của 1 số. Vậy muốn tìm chữ số tận cùng khác 0 đầu tiên của 1 số, thì làm như thế nào ?
* Đơn giản lắm, chúng ta chỉ cần xác định được số chữ số 0 tận cùng của X. Nếu X chia hết cho 10k, thì bạn chỉ việc chia X cho 10k là sẽ tìm ra thôi :)
* Các bạn sẽ thắc mắc làm sao xác định được *số chữ số 0 tận cùng* của X nếu như X quá lớn (ví dụ : X = n!, X = nk với n, k đều là giá trị lớn ...)
  + Thực ra thì, những số X này đều được cấu thành từ những giá trị đủ để ta tính toán. Nên thay vì gộp chung chúng lại rồi tìm 1 lần, các bạn có thể tìm bậc lũy thừa của 2 và 5 cho từng số, rồi cộng tổng lại
  + Tính đúng đắn của nó, kế thừa từ định lý toán cơ bản : khi nhân 2 lũy thừa cùng cơ số, được 1 lũy thừa mới có bậc bằng tổng bậc 2 lũy thừa kia
  + Code minh họa :

int NumOf0s(vector a, int n) {

int sum2 = 0, sum5 = 0;

for (int i = 0; i < n; ++i) {

int num2 = 0, num5 = 0;

while (a[i] % 2 == 0) {

num2++;

a[i] /= 2;

}

while (a[i] % 5 == 0) {

num5++;

a[i] /= 5;

}

sum2 += num2, sum5 += num5;

}

return min(sum2, sum5);

}

Code mẫu:

Ngôn ngữ C++:

int lastDigitDiffZero(int n)

{

long long res = 1;

for (int i=2;i<=n;i++)

{

res \*=i;

while ( res%10 == 0 ) res /= 10;

res = res % 100;

}

while ( res%10 == 0 ) res /= 10;

return res% 10;

}