SOLUTION 3

SỐ HẠNG: tìm ước nguyên tố nhỏ nhất

Với n = a+b, ước chung của a và b cũng là ước của n,

Nếu x là ước của n thì y = n/x cũng là ước của n,

Với ước y bao giờ cũng phân tích n thành tổng a và b, trong đó a và b đều chia hết cho y, ví dụ a = y, b = n – y

Ước nhỏ nhất của n là số nguyên tố nhỏ nhất trong phép phân tích n ra thừa số nguyên tố nếu n là hợp số và bằng n nếu n là nguyên tố.

Để giảm độ phức tạp thuật toán giải thuật cần phân biệt 2 trường hợp:

- N là số chẵn, khi đó x = 2
- N là số lẻ: cần kiểm tra x với x = 3, 5, 7, …$\sqrt{n}$

  Trường hợp n là số nguyên tố a=1, b=n-1

  Độ phức tạp là O($\sqrt{n}$)

SEQ

<u>Sub 1</u>:

Ta có thể sử dụng thuật toán QHĐ với dp(L, R) là số điểm lớn nhất có thể đạt đc với đoạn A[L] .... A[R]

dp(L,R) = max[dp(L,R), max[dp(L, x) , dp(x + 1, R)]] nếu A[L] + A[L + 1] +...+ A[x] = A[x + 1] + ... + A[R].

dp(L,R) = 0 nếu không có X thỏa mãn.

Đpt: O(n^3).

<u>Sub 2</u>:

Ta có thể cải tiến thuật toán trên thành O(n ^2) bằng cách tính trước X cho mỗi đoạn con từ l đến r.

<u>Sub 3</u>:

Gọi tổng tất cả các phần tử của dãy A là S.

Ta nhận thấy rằng do với mỗi bước dãy đều được chia làm hai phần bằng nhau nên sau khi chia lần thứ y thì dãy sẽ có tổng là S / (2 ^ y).

Ta sẽ QHĐ với dp(L, y) là số điểm lớn nhất có được với dãy có phần tử đầu là A[L] và đang chia lần thứ y.

Ta sẽ sử dụng chặt nhị phân để tìm vị trí R thỏa mãn tổng từ A[L] đến A[R] = S / (2 ^ y).

Nếu tìm thấy R thì :

dp(L,y) = max[dp(L,y), max[dp(L, y + 1), dp (r + 1, y + 1)] + 1].

Ngược lại dp(L,y) = 0.

INFINITY

Xử lí số lớn, tìm kiếm nhị phân

Mỗi số tự nhiên mới trong dãy sẽ tương ứng với một nhóm các phần tử của dãy:

1 → 1

2 → 1 2 1

3 → 1 2 3 2 1

4 → 1 2 3 4 3 2 1

…

Số m sẽ tương ứng với nhóm có 2*m-1 phần tử

Độ dài phần đầu của dãy, khi có m nhóm là 1 + 3 + 5 + … + (2*m-1) = m^2

Để xác định số ở vị trí n ta cần biết vị trí cần tìm thuộc nhóm nào

Gọi k là nhóm chứa vị trí n.

K sẽ là số tự nhiên nhỏ nhất thỏa mãn điều kiện $k^2 \geq n$

Với $n \leq 10^{10}$ bằng cách duyệt trực tiếp có thể tìm k với độ phức tạp $O(\sqrt{n})$

```
k=1;
while(k*k<n)++k;
t=k*k;
if(t==n)ans=1;
    else
    {
        t=n-(k-1)*(k-1);
        if(t<=k)ans=t; else ans=t+1-k;
    }
```

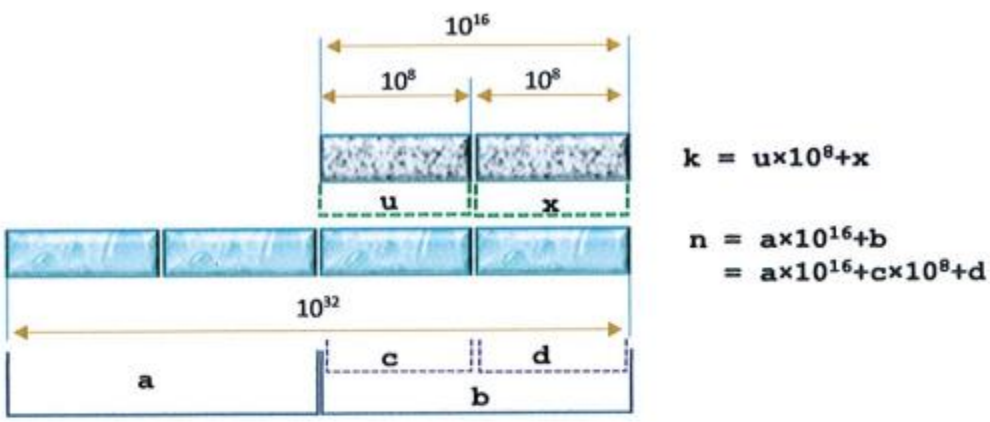
Với $n>10^{10}$ cần tổ chức tìm kiếm nhị phân

Theo điều kiện đầu bài: $n \leq 10^{32} \rightarrow k \leq 10^{16}$

Có thể trực tiếp xác định k bằng phương pháp tìm kiếm nhị phân. Nhưng việc này đòi hỏi phải tổ chức nhân và so sánh số lớn – một điều khá phức tạp và hiệu quả chương trình giảm đi một cách đáng kể.

Có một cách hiệu quả hơn

Để tính k^2 ta chỉ cần tổ chức biểu diễn k dưới dạng số lớn, trong trường hợp này – hiểu quả nhất là theo cơ số 10^8



Dễ dàng tính được u: $u = \lfloor \sqrt{a} \rfloor$

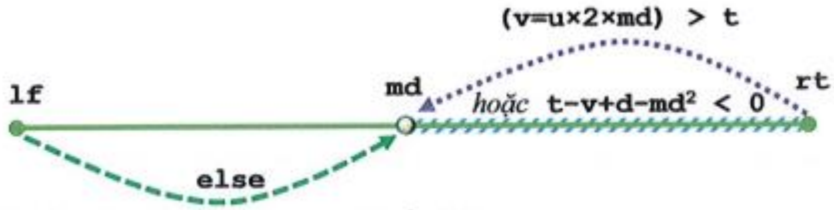
Xác định v: n = (a,b)

$$n - u^2 = (a-u^2, b) \leq 2*u*x + x^2$$

Lưu ý vế phải là một số trong phạm vi biểu diễn của kiểu dữ liệu int64_t

Đặt $t = (a-u^2)*10^8 + c$

Tiêu chuẩn tìm kiếm nhị phân:



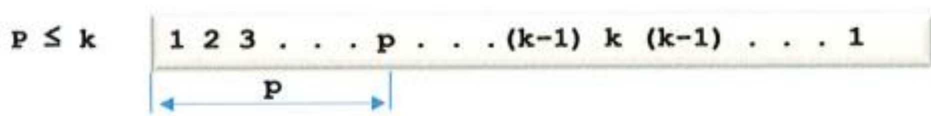
Kết quả nhóm tìm được sẽ là $k = u*10^8 + lf$

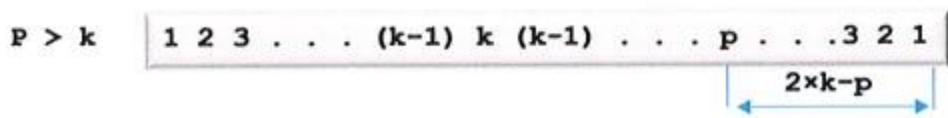
Cần tăng kết quả tìm kiếm từ nhị phân lên 1 nếu n không phải là số chính phương

Vị trí n trong dãy tương ứng với vị trí p trong nhóm k

$P = t-v+d-(lf-1)^2$

Có hai trường hợp



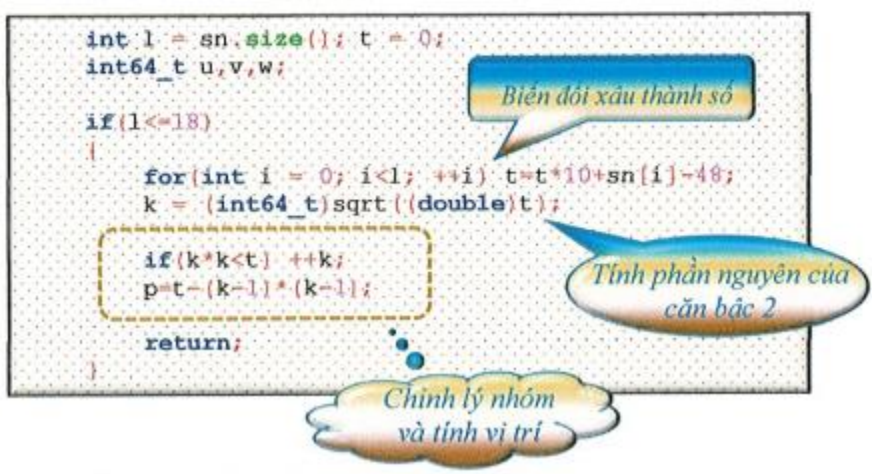


Tổ chức dữ liệu

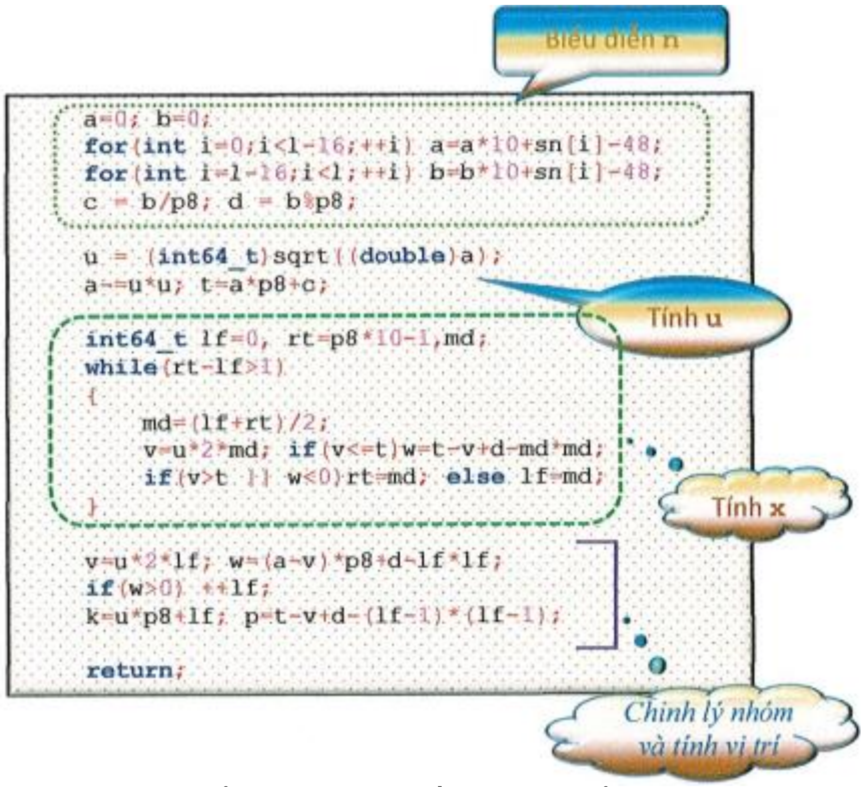
- Lưu n dưới dạng xâu
- Hệ thống các biến đơn a, b, c, d – biểu diễn lại n phục vụ các phép tính số lớn

Xử lí

- Với n<= 10^{18} : trực tiếp tìm kiếm nhị phân xác định nhóm chứa vị trí n



- Với trường hợp cần làm việc với số lớn



```
a=0; b=0;
for(int i=0;i<l-16;++i) a=a*10+sn[i]-48;
for(int i=l-16;i<l;++i) b=b*10+sn[i]-48;
c = b/p8; d = b%p8;

u = (int64_t)sqrt((double)a);
a-=u*u; t=a*p8+c;

int64_t lf=0, rt=p8*10-1,md;
while(rt-lf>1)
{
    md=(lf+rt)/2;
    v=u*2*md; if(v<=t)w=t-v+d-md*md;
    if(v>t || w<0)rt=md; else lf=md;
}

v=u*2*lf; w=(a-v)*p8+d-lf*lf;
if(w>0) ++lf;
k=u*p8+lf; p=t-v+d-(lf-1)*(lf-1);

return;
```


Nhận xét: việc triển khai hai sơ đồ xử lí chỉ để giảm thời gian xử lí trong trường hợp cụ thể