Kĩ thuật hai con trỏ

Kĩ thuật hai con trỏ

Tác giả: Phan Đình Khôi - Đại học Bách Khoa Đà Nẵng

Reviewer: Nguyễn Khánh

Lời mở đầu

Bài viết này sẽ giúp bạn tìm hiểu thêm về **kỹ thuật hai con trỏ**. Kỹ thuật này được sử dụng khá phổ biến, giúp chương trình tiết kiệm thời gian và không gian xử lý.

Bài toán 1

Cho hai dãy số nguyên đã được **sắp xếp không giảm** a và b lần lượt có n và m phần tử. Hãy ghép chúng thành dãy c được bố trí theo thứ tự **không giảm**.

Giới hạn: $n, m \leq 10^5$ và $0 \leq a_i, b_i \leq 10^9$.

Phân tích

Hãy cùng xem ví dụ sau đây.

Cho trước hai dãy số a và b được sắp xếp không giảm:

$a = [1, 3, 6, 8, 10]$

$b = [2, 6, 7, 12, 14, 15]$

Làm cách nào để có thể ghép chúng thành một dãy số c cũng được sắp xếp không giảm ?

Trước tiên, hãy cùng xác định phần tử đầu tiên của dãy c.

Vì dãy c được bố trí theo thứ tự không giảm, cho nên phần tử đầu tiên của dãy c phải là phần tử có giá trị nhỏ nhất trong cả hai dãy a và b.

Ta có thể so sánh hai phần tử nhỏ nhất của hai dãy a, b và đưa phần tử có giá trị nhỏ hơn vào vị trí đầu tiên của dãy c.

Dãy a và b đã được sắp xếp không giảm, vì thế hai phần tử nhỏ nhất ở đây chính là hai phần tử ở vị trí đầu tiên ở mỗi dãy ($a[1]$ và $b[1]$).

$a = [\overset{\downarrow}{\color{red}1}, 3, 6, 8, 10]$

$b = [\overset{\downarrow}{2}, 6, 7, 12, 14, 15]$

$c = [\color{red}1]$

Bây giờ, phần tử tiếp theo của dãy c sẽ là phần tử nhỏ nhất trong các phần tử chưa được đưa vào dãy c.

Dãy a và b đã được sắp xếp không giảm, vì thế sau khi đưa $a[1]$ vào dãy c, $a[2]$ là phần tử nhỏ nhất chưa được chọn ở dãy a và $b[1]$ là phần tử nhỏ nhất chưa được chọn ở dãy b.

So sánh $a[2]$ và $b[1]$, chọn phần tử có giá trị nhỏ hơn và đưa vào dãy c.

$$a = [1, \overset{\downarrow}{3}, 6, 8, 10]$$

$$b = [\overset{\downarrow}{\color{red}{2}}, 6, 7, 12, 14, 15]$$

$$c = [1, \color{red}{2}]$$

Sau khi đưa $b[1]$ vào dãy c, $b[2]$ trở thành phần tử nhỏ nhất chưa được chọn ở dãy b.

Vẫn như thế, phần tử tiếp theo của dãy c sẽ là phần tử nhỏ nhất trong các phần tử chưa được đưa vào dãy c.

So sánh $b[2]$ và $a[2]$, chọn phần tử có giá trị nhỏ hơn dãy và đưa vào dãy c.

$$a = [1, \overset{\downarrow}{\color{red}{3}}, 6, 8, 10]$$

$$b = [2, \overset{\downarrow}{6}, 7, 12, 14, 15]$$

$$c = [1, 2, \color{red}{3}]$$

Sau khi đưa $a[2]$ vào dãy c, $a[3]$ trở thành phần tử nhỏ nhất chưa được chọn ở dãy a.

Ta nhận thấy rằng

- Tại mọi thời điểm, phần tử tiếp theo được đưa vào dãy c sẽ là phần tử có giá trị nhỏ nhất trong các phần tử chưa được chọn.
  - Bằng cách so sánh phần tử nhỏ nhất chưa được chọn ở dãy a và phần tử nhỏ nhất chưa được chọn ở dãy b, phần tử nhỏ hơn sẽ được chọn vào dãy c.
- Ban đầu, lúc dãy c chưa có phần tử nào
  - $a[1]$ là phần tử nhỏ nhất chưa được chọn trong dãy a.
  - $b[1]$ là phần tử nhỏ nhất chưa được chọn trong dãy b.
- Khi đưa phần tử $a[i]$ vào dãy c thì phần tử nhỏ nhất chưa được chọn trong dãy a sẽ là $a[i+1]$.
- Khi đưa phần tử $b[j]$ vào dãy c thì phần tử nhỏ nhất chưa được chọn trong dãy b sẽ là $b[j+1]$.

Giải pháp

Dựa vào những phân tích ta có giải pháp sử dụng **hai con trỏ** như sau:

- Dãy a có con trỏ i, con trỏ này bắt đầu ở vị trí đầu dãy a.
  - Con trỏ i này được thể hiện như phần tử nhỏ nhất chưa được chọn trong dãy a.
- Dãy b có con trỏ j, con trỏ này bắt đầu ở vị trí đầu dãy b.
  - Con trỏ j này được thể hiện như phần tử nhỏ nhất chưa được chọn trong dãy b.
- Ta sẽ lặp lại công việc này, cho đến khi đưa hết các phần tử trong dãy a và b vào dãy c:
  - Khi các phần tử trong một dãy nào đó, dãy a hoặc dãy b, đều đã được đưa vào dãy c: đưa lần lượt các phần tử trong dãy còn lại vào dãy c.
  - Ngược lại:
    - So sánh hai phần tử ở hai con trỏ.
    - Đưa phần tử có giá trị nhỏ hơn vào dãy c, nếu hai phần tử có giá trị như nhau thì chọn một trong hai.

- Tăng vị trí con trỏ ở phần tử được đưa vào lên một đơn vị.

Để hiểu rõ hơn, ta hãy cùng xem qua ví dụ sau đây:

$a = [1, 3, 6, 8, 10], b = [2, 6, 7, 12, 14, 15]$

- Đặt $i = 1$ và $j = 1$.

$$a = [\overset{\overset{i}{\downarrow}}{1}, 3, 6, 8, 10]$$
$$b = [\underset{\underset{j}{\uparrow}}{2}, 6, 7, 12, 14, 15]$$
$$c = []$$

- Vì $a[i] < b[j]$ nên ta đưa $a[i]$ vào mảng c và tăng vị trí i lên một.

$$a = [1, \overset{\overset{i}{\downarrow}}{3}, 6, 8, 10]$$
$$b = [\underset{\underset{j}{\uparrow}}{2}, 6, 7, 12, 14, 15]$$
$$c = [1]$$

- Vì $b[j] < a[i]$ nên ta đưa $b[j]$ vào mảng c và tăng vị trí j lên một.

$$a = [1, \overset{\overset{i}{\downarrow}}{3}, 6, 8, 10]$$
$$b = [2, \underset{\underset{j}{\uparrow}}{6}, 7, 12, 14, 15]$$
$$c = [1, 2]$$

- Vì $a[i] < b[j]$ nên ta đưa $a[i]$ vào mảng c và tăng vị trí i lên một.

$$a = [1, 3, \overset{\overset{i}{\downarrow}}{6}, 8, 10]$$
$$b = [2, \underset{\underset{j}{\uparrow}}{6}, 7, 12, 14, 15]$$
$$c = [1, 2, 3]$$

- Vì $a[i] = b[j]$ nên ta có thể đưa bất kỳ một trong hai phần tử. Ở đây ta đưa phần tử $a[i]$ vào c và tăng vị trí i lên một.

$$a = [1, 3, 6, \overset{\overset{i}{\downarrow}}{8}, 10]$$
$$b = [2, \underset{\underset{j}{\uparrow}}{6}, 7, 12, 14, 15]$$
$$c = [1, 2, 3, 6]$$

- Vì $b[j] < a[i]$ nên ta đưa $b[j]$ vào mảng c và tăng vị trí j lên một.

$$a = [1, 3, 6, \overset{\overset{i}{\downarrow}}{8}, 10]$$
$$b = [2, 6, \underset{\underset{j}{\uparrow}}{7}, 12, 14, 15]$$
$$c = [1, 2, 3, 6, 6]$$

- Vì $b[j] < a[i]$ nên ta đưa $b[j]$ vào mảng c và tăng vị trí j lên một.

$$a = [1, 3, 6, \overset{\overset{i}{\downarrow}}{8}, 10]$$
$$b = [2, 6, 7, \underset{\underset{j}{\uparrow}}{12}, 14, 15]$$
$$c = [1, 2, 3, 6, 6, 7]$$

- Vì $a[i] < b[j]$ nên ta đưa $a[i]$ vào mảng c và tăng vị trí i lên một.

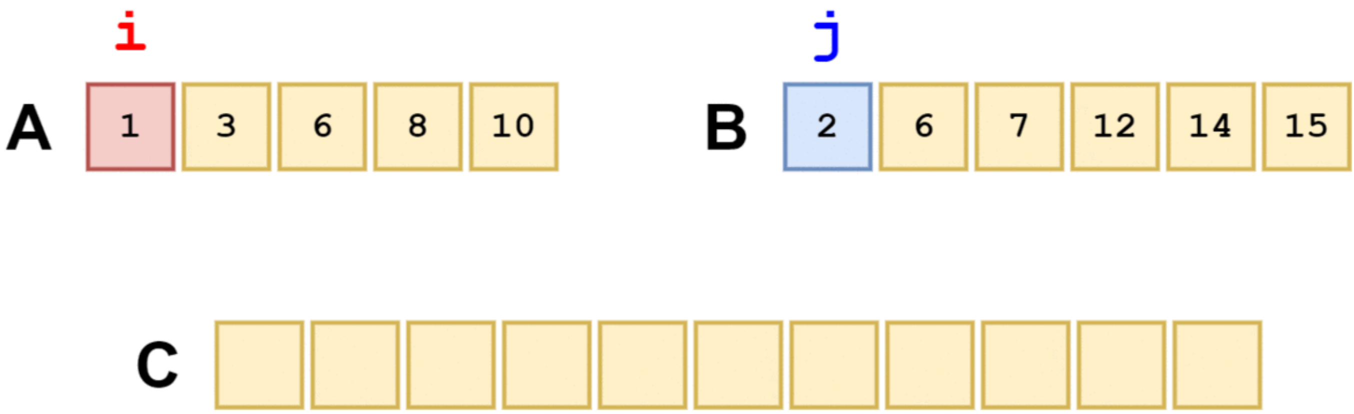
$$a = [1, 3, 6, 8, \overset{\overset{i}{\downarrow}}{10}]$$
$$b = [2, 6, 7, \underset{\underset{j}{\uparrow}}{12}, 14, 15]$$
$$c = [1, 2, 3, 6, 6, 7, 8]$$

- Vì $a[i] < b[j]$ nên ta đưa $a[i]$ vào mảng c và tăng vị trí i lên một.

$$a = [1, 3, 6, 8, 10]\overset{i}{\downarrow}$$
$$b = [2, 6, 7, \underset{\overset{\uparrow}{j}}{12}, 14, 15]$$
$$c = [1, 2, 3, 6, 6, 7, 8, 10]$$

- Vì tất cả các phần tử trong dãy a đều đã được đưa vào dãy c nên từ đưa lần lượt các phần tử chưa được chọn trong dãy b vào trong dãy c

$$c = [1, 2, 3, 6, 6, 7, 8, 10, 12, 14, 15]$$



Cài đặt

```
int i = 1, j = 1;
vector<int> c;
while (i <= n || j <= m){
    if (j == m + 1 || (i <= n && a[i] <= b[j]))
        c.push_back(a[i++]);
    else
        c.push_back(b[j++]);
}
for (auto it: c)
    cout << it << " ";
```

Độ phức tạp

Vị trí con trỏ i luôn tăng và tăng quá không quá n lần, vị trí con trỏ j cũng luôn tăng và tăng không quá m lần.

Vì thế độ phức tạp của giải pháp là $O(n + m)$.

Luyện tập

- VNOJ - NKSGAME
- CODEFORCES - 1251C
- CODEFORCES - 1036D

Bài toán 2

Cho một mảng số nguyên a có n phần tử, mảng này đã được **sắp xếp tăng dần**. Hãy tìm hai vị trí **khác nhau bất kỳ** sao cho tổng của hai phần tử ở hai vị trí đó có giá trị là x.

Giới hạn: $2 \leq n \leq 10^6$ và $0 \leq a_i, x \leq 10^9$

Phân tích

Hãy cùng xem ví dụ sau đây.

Cho trước mảng số a được sắp xếp tăng dần và $x = 16$:

$a = [2, 5, 6, 8, 10, 12, 15]$

Làm cách nào để có thể tìm hai vị trí khác nhau mà tổng hai phần tử ở hai vị trí đó có tổng là x ?

Trước tiên, ta có một chút nhận xét sau:

- $a[1] < a[2] < a[3] < a[4] < a[5] < a[6] < a[7]$ vì dãy a tăng dần.
- $a[1] + a[7] = 17 > X \Rightarrow X < a[1] + a[7] < a[2] + a[7] < a[3] + a[7] < a[4] + a[7] < a[5] + a[7] < a[6] + a[7]$.

Có thể thấy, tổng của $a[7]$ với các phần tử khác trong dãy đều lớn hơn X. Vì thế ta không quan tâm đến $a[7]$ nữa.

$a = [2, 5, 6, 8, 10, 12, \color{red}{15}]$

- $a[1] + a[6] = 14 < X \Rightarrow a[1] + a[2] < a[1] + a[3] < a[1] + a[4] < a[1] + a[5] < a[1] + a[6] < X$.

Có thể thấy, tổng của $a[1]$ với các phần tử khác trong các phần tử ta quan tâm đều nhỏ hơn x. Vì thế ta không quan tâm đến $a[1]$ nữa.

$a = [\color{red}{2}, 5, 6, 8, 10, 12, \color{red}{15}]$

- $a[2] + a[6] = 17 > X \Rightarrow X < a[2] + a[6] < a[3] + a[6] < a[4] + a[6] < a[5] + a[6]$.

Có thể thấy, tổng của $a[6]$ với các phần tử khác trong các phần tử ta quan tâm đều lớn hơn x. Vì thế ta không quan tâm đến $a[6]$ nữa.

$a = [\color{red}{2}, 5, 6, 8, 10, \color{red}{12}, \color{red}{15}]$

Như vậy, tại một thời điểm bất kỳ, những phần tử chúng ta cần quan tâm đến sẽ là các phần tử trong đoạn $[i, j]$ nào đó.

Ta có một số nhận xét sau:

- Nếu $i = j$, trong mảng A không tồn tại hai vị trí khác nhau mà tổng hai phần tử ở đó có giá trị là X.
- Ngược lại:
  - Nếu $a[i] + a[j] = X$, ta đã tìm được hai vị trí cần tìm (i và j).
  - Nếu $a[i] + a[j] < X$, không quan tâm đến $a[i]$ nữa và các phần tử chúng ta cần quan tâm đó là các phần tử trong đoạn $[i + 1, j]$.
  - Nếu $a[i] + a[j] > X$, không quan tâm đến $a[j]$ nữa và các phần tử chúng ta cần quan tâm đó là các phần tử trong đoạn $[i, j - 1]$.

Giải pháp

Từ những phân tích vừa rồi ta có giải pháp sử dụng hai con trỏ như sau:

- Một con trỏ (i) được đặt ở đầu mảng A, con trỏ còn lại (j) được đặt ở cuối mảng A.
- Nếu tổng của hai phần tử ở hai vị trí con trỏ
  - Nhỏ hơn X: tăng vị trí con trỏ i lên một đơn vị.
  - Lớn hơn X: giảm vị trí con trỏ j đi một đơn vị.
- Tiếp tục di chuyển cho đến khi hai con trỏ gặp nhau.

- Khi con trỏ chưa gặp nhau mà tổng ở hai vị trí con trỏ có giá trị là X thì ta đã tìm được hai vị trí cần tìm (i và j), kết thúc chương trình.

Để hiểu rõ hơn, ta hãy cùng xem qua một số ví dụ sau đây:

Ví dụ 1: $a = [2, 5, 6, 8, 10, 12, 15]$ và $x = 16$.

- Đặt $i = 1$ và $j = N$.

  $$a = [\overset{\overset{i}{\downarrow}}{\color{red}2}, 5, 6, 8, 10, 12, \underset{\underset{j}{\uparrow}}{\color{blue}15}]$$

- Vì $a[i] + a[j] = 2 + 15 = 17 > x$ nên giảm vị trí j đi một đơn vị.

  $$a = [\overset{\overset{i}{\downarrow}}{\color{red}2}, 5, 6, 8, 10, \underset{\underset{j}{\uparrow}}{\color{blue}12}, 15]$$

- Vì $a[i] + a[j] = 2 + 12 = 14 < x$ nên tăng vị trí i lên một đơn vị.

  $$a = [2, \overset{\overset{i}{\downarrow}}{\color{red}5}, 6, 8, 10, \underset{\underset{j}{\uparrow}}{\color{blue}12}, 15]$$

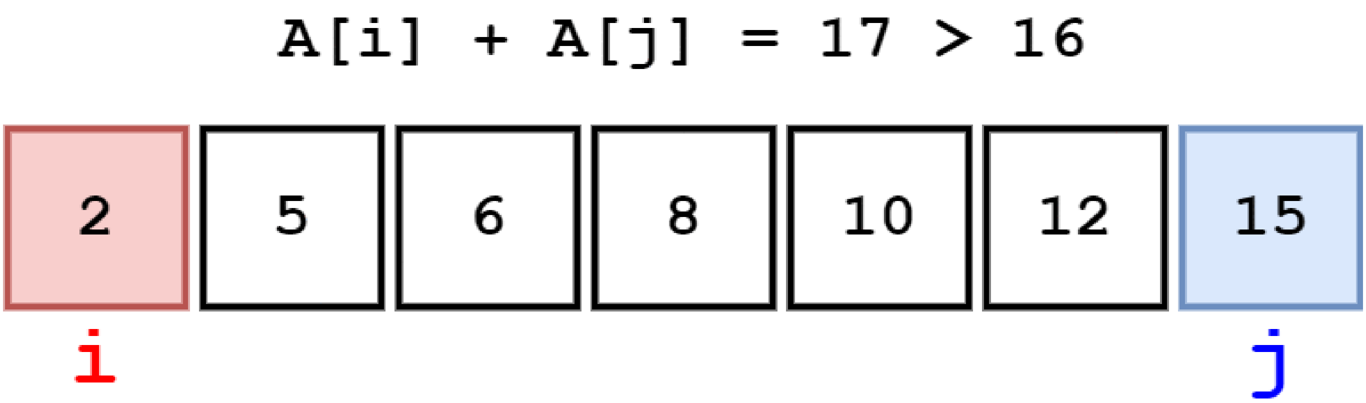
- Vì $a[i] + a[j] = 5 + 12 = 17 > x$ nên giảm vị trí j đi một đơn vị.

  $$a = [2, \overset{\overset{i}{\downarrow}}{\color{red}5}, 6, 8, \underset{\underset{j}{\uparrow}}{\color{blue}10}, 12, 15]$$

- Vì $a[i] + a[j] = 5 + 10 < x$ nên tăng vị trí i lên một đơn vị.

  $$a = [2, 5, \overset{\overset{i}{\downarrow}}{\color{red}6}, 8, \underset{\underset{j}{\uparrow}}{\color{blue}10}, 12, 15]$$

- Vì $a[i] + a[j] = 6 + 10 = x$ nên hai vị trí cần tìm là hai vị trí i và j.



Ví dụ 2: $a = [2, 3, 7, 8, 10, 12, 15]$ và $x = 16$.

- Đặt $i = 1$ và $j = N$.

  $$a = [\overset{\overset{i}{\downarrow}}{\color{red}2}, 3, 7, 8, 10, 12, \underset{\underset{j}{\uparrow}}{\color{blue}15}]$$

- Vì $a[i] + a[j] = 5 + 12 = 17 > x$ nên giảm vị trí j đi một đơn vị.

  $$a = [\overset{\overset{i}{\downarrow}}{\color{red}2}, 3, 7, 8, 10, \underset{\underset{j}{\uparrow}}{\color{blue}12}, 15]$$

- Vì $a[i] + a[j] = 2 + 12 = 14 < x$ nên tăng vị trí i lên một đơn vị.

  $$a = [2, \overset{\overset{i}{\downarrow}}{\color{red}3}, 7, 8, 10, \underset{\underset{j}{\uparrow}}{\color{blue}12}, 15]$$

- Vì $a[i] + a[j] = 3 + 12 = 15 < x$ tăng vị trí i lên một đơn vị.

$$a = [2, 3, \overset{i}{\overset{\downarrow}{7}}, 8, 10, \underset{j}{\underset{\uparrow}{12}}, 15]$$

- Vì $a[i] + a[j] = 7 + 12 = 19 > x$ giảm vị trí j đi một đơn vị.

$$a = [2, 3, \overset{i}{\overset{\downarrow}{7}}, 8, \underset{j}{\underset{\uparrow}{10}}, 12, 15]$$

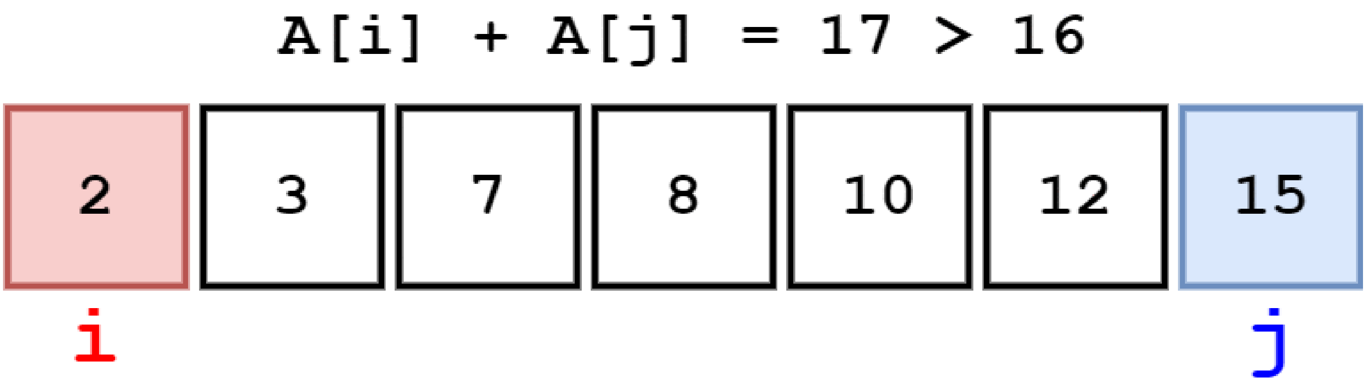
- Vì $a[i] + a[j] = 7 + 10 = 17 > x$ giảm vị trí j đi một đơn vị.

$$a = [2, 3, \overset{i}{\overset{\downarrow}{7}}, \underset{j}{\underset{\uparrow}{8}}, 10, 12, 15]$$

- Vì $a[i] + a[j] = 7 + 8 = 15 < x$ tăng vị trí i lên một đơn vị.

$$a = [2, 3, 7, \underset{j}{\underset{\uparrow}{\overset{i}{\overset{\downarrow}{8}}}}, 10, 12, 15]$$

- Vì $i = j$ nên không tìm được hai vị trí cần tìm.



Cài đặt

```
int i = 1, j = N;
while (i < j) {
    if (a[i] + a[j] == x) {
        cout << i << " " << j;
        return 0;
    }
    if (a[i] + a[j] < x)
        i += 1;
    else
        j -= 1;
}
cout << "No solution";
```

Độ phức tạp

Vị trí con trỏ i luôn tăng, vị trí con trỏ j thì luôn giảm.

Hơn nữa, sự thay đổi vị trí hai con trỏ này sẽ dừng lại khi tổng hai phần tử ở hai vị trí con trỏ có tổng là X hay khi vị trí i bằng vị trí j.

Vì thế, việc thay đổi vị trí hai con trỏ sẽ không quá n lần, độ phức tạp của giải pháp là $O(n)$.

Luyện tập

- LQDOJ - FINDPAIR
- LQDOJ - CNTPAIR02
- VNOJ - NDCCARD
- VNOJ - TWOSUM

Bài toán 3

Cho dãy số **nguyên dương** a có n phần tử. Hãy tìm độ dài đoạn con dài nhất trong dãy sao cho tổng các phần tử trong đoạn này không quá s.

Dữ liệu đảm bảo các phần tử trong dãy a đều có giá trị không quá s.

Giới hạn: $1 \leq n \leq 10^6, 1 \leq a_i \leq 10^9$ và $s \leq 10^{18}$.

Phân tích

Để dễ dàng phân tích, ta tạm gọi

- $sum(l, r)$ là tổng các phần tử trong đoạn $[l, r]$.
- Một đoạn con $[l, r]$ là đoạn con "tốt" nếu $sum(l, r) \leq s$

Qua đây, bài toán của chúng ta sẽ là tìm độ dài đoạn con "tốt" dài nhất.

Vì dãy a là một dãy số nguyên dương cho nên

- $sum(1, r) > sum(2, r) > ... > sum(r - 1, r) > sum(r, r)$.
- Nếu đoạn con $[l, r]$ là đoạn con "tốt" thì với mọi $x \geq l$, đoạn $[x, r]$ là đoạn con "tốt".
- Nếu đoạn con $[l, r]$ không là đoạn con "tốt" thì với mọi $x \leq l$, đoạn $[x, r]$ không là đoạn con "tốt".

Với r là một vị trí bất kỳ, nếu như l là vị trí nhỏ nhất sao cho đoạn $[l, r]$ là một đoạn "tốt" thì

- mọi $x \geq l$ thì đoạn con $[x, r]$ là một đoạn "tốt".
- mọi $x < l$ thì đoạn con $[x, r]$ không là một đoạn "tốt".
- đoạn con $[l, r]$ là một đoạn con "tốt" dài nhất trong các đoạn con "tốt" có vị trí kết thúc tại r.

Từ đó, với mỗi r từ 1 đến n, nếu ta xác định được vị trí l, ta có thể biết được độ dài của đoạn con "tốt" dài nhất của dãy a.

Hãy cùng nhận xét vị trí của l với mỗi r từ 1 đến n qua ví dụ sau đây:

Cho trước dãy $a = [2, 6, 5, 3, 6, 8, 9]$ và $s = 20$

- $r = 1 \rightarrow l = 1$
  - $$a = [\overset{l,r}{\overset{\downarrow}{2}}, 6, 5, 3, 6, 8, 9]$$
  - $sum(l, r) = 2$
- $r = 2 \rightarrow l = 1$
  - $$a = [\overset{l}{\overset{\downarrow}{2}}, \overset{r}{\overset{\downarrow}{6}}, 5, 3, 6, 8, 9]$$
  - $sum(l, r) = 8$
- $r = 3 \rightarrow l = 1$

  - $a = [\overset{\overset{l}{\downarrow}}{2}, 6, \overset{\overset{r}{\downarrow}}{5}, 3, 6, 8, 9]$
  - $sum(l, r) = 13$
- $r = 4 \to l = 1$
  - $a = [\overset{\overset{l}{\downarrow}}{2}, 6, 5, \overset{\overset{r}{\downarrow}}{3}, 6, 8, 9]$
  - $sum(l, r) = 16$
- $r = 5 \to l = 2$
  - $a = [2, \overset{\overset{l}{\downarrow}}{6}, 5, 3, \overset{\overset{r}{\downarrow}}{6}, 8, 9]$
  - $sum(l, r) = 20$
- $r = 6 \to l = 4$
  - $a = [2, 6, 5, \overset{\overset{l}{\downarrow}}{3}, 6, \overset{\overset{r}{\downarrow}}{8}, 9]$
  - $sum(l, r) = 17$
- $r = 7 \to l = 6$
  - $a = [2, 6, 5, 3, 6, \overset{\overset{l}{\downarrow}}{8}, \overset{\overset{r}{\downarrow}}{9}]$
  - $sum(l, r) = 17$

r	l	**Độ dài đoạn con**
1	1	1
2	1	2
3	1	3
4	1	4
5	2	4
6	4	3
7	6	2

Độ dài của đoạn con "tốt" dài nhất của dãy là giá trị lớn nhất của độ dài các đoạn con "tốt" dài nhất với vị trí kết thúc từ 1 đến n.

Ở đây, độ dài đoạn con "tốt" dài nhất của dãy là 4.

Qua ví dụ vừa rồi, ta thấy rằng, vị trí l đối với các giá trị r từ 1 đến n có giá trị không giảm.

Thật vậy, với mọi $x < l$ thì $sum(x, r) > s \Rightarrow sum(x, r + 1) > s$, vì thế giá trị l đối với $r + 1$ phải không quá giá trị l đối với r.

Hơn nữa vì các phần tử trong dãy a đều có giá trị không quá s cho nên luôn tồn tại vị trí $l \le r$ sao cho đoạn $[l, r]$ là một đoạn "tốt".

Giải pháp

Với những phân tích như trên, ta có giải quyết bài toán với phương pháp hai con trỏ như sau:

- Hai con trỏ l và r sẽ đặt ở vị trí 1.
  - Hai con trỏ này được thể hiện như hai vị trí l, r như ở trên phần phân tích.
- Di chuyển lần lượt con trỏ r từ 1 đến n.
  - Sau mỗi lần di chuyển con trỏ r, nếu
    - $sum(l, r) \leq s$: giữ nguyên vị trí con trỏ l.
    - $sum(l, r) > s$: tăng dần vị trí con trỏ l cho đến khi $sum(l, r) \leq s$.
  - Hiện tại với vị trí con trỏ l và r, ta biết đoạn "tốt" dài nhất với vị trí kết thúc tại r là đoạn $[l, r]$.
- Độ dài đoạn con "tốt" dài nhất chính là giá trị độ dài lớn nhất của các đoạn "tốt" dài nhất với vị trí kết thúc tại r, với mỗi r từ 1 đến n.

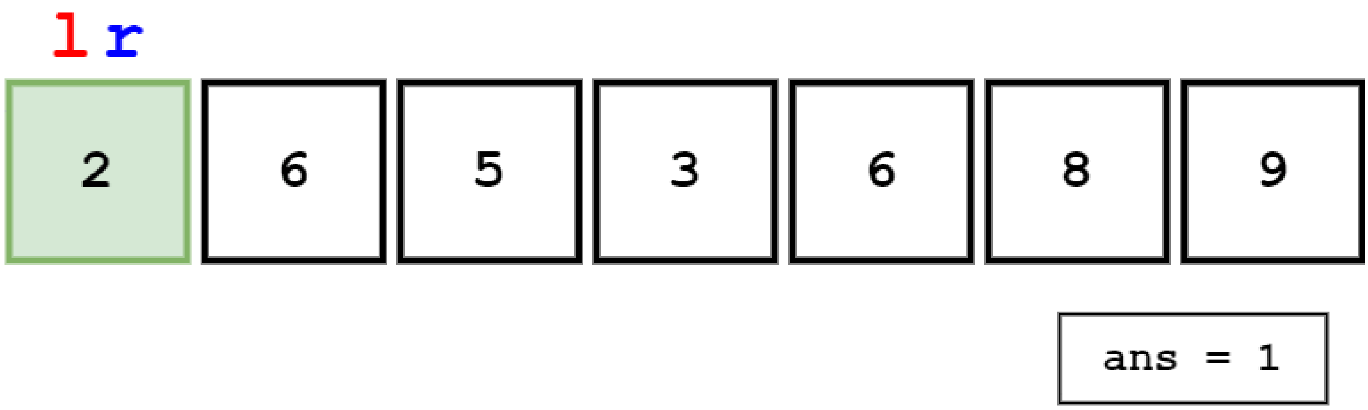
Để hiểu rõ hơn, ta hãy cùng xem qua một số ví dụ sau đây:

$a = [2, 6, 5, 3, 6, 8, 9]$ và $s = 20$

- Sử dụng biến ans để lưu lại giá trị lớn nhất của độ dài đoạn "tốt" có vị trí kết thúc tại r, với r từ 1 đến n.
- Đặt $l = 1$ và $r = 1$
  - $a = [\overset{\overset{l,r}{\downarrow}}{2}, 6, 5, 3, 6, 8, 9]$
  - vì $a[1] \leq s$ nên đoạn $[1, 1]$ là một đoạn "tốt".
  - $ans = max(ans, r - l + 1)$
- Tăng vị trí r lên 1 đơn vị
  - $a = [\overset{\overset{l}{\downarrow}}{2}, \overset{\overset{r}{\downarrow}}{6}, 5, 3, 6, 8, 9]$
  - vì $sum(l, r) = 8 \leq s$ nên đoạn $[l, r]$ là một đoạn tốt.
  - $ans = max(ans, r - l + 1)$
- Tăng vị trí r lên 1 đơn vị
  - $a = [\overset{\overset{l}{\downarrow}}{2}, 6, \overset{\overset{r}{\downarrow}}{5}, 3, 6, 8, 9]$
  - vì $sum(l, r) = 13 \leq s$ nên đoạn $[l, r]$ là một đoạn tốt.
  - $ans = max(ans, r - l + 1)$
- Tăng vị trí r lên 1 đơn vị
  - $a = [\overset{\overset{l}{\downarrow}}{2}, 6, 5, \overset{\overset{r}{\downarrow}}{3}, 6, 8, 9]$
  - vì $sum(l, r) = 16 \leq s$ nên đoạn $[l, r]$ là một đoạn tốt.
  - $ans = max(ans, r - l + 1)$
- Tăng vị trí r lên 1 đơn vị
  - $a = [\overset{\overset{l}{\downarrow}}{2}, 6, 5, 3, \overset{\overset{r}{\downarrow}}{6}, 8, 9]$
  - vì $sum(l, r) = 22 > s$ nên tăng vị trí l.
- Tăng vị trí l lên 1 đơn vị
  - $a = [2, \overset{\overset{l}{\downarrow}}{6}, 5, 3, \overset{\overset{r}{\downarrow}}{6}, 8, 9]$

- vì $sum(l, r) = 20 \le s$ nên đoạn $[l, r]$ là một đoạn tốt.
- $ans = max(ans, r - l + 1)$

- Tăng vị trí r lên 1 đơn vị
  - $a = [2, \overset{\overset{l}{\downarrow}}{6}, 5, 3, 6, \overset{\overset{r}{\downarrow}}{8}, 9]$
  - vì $sum(l, r) = 28 > s$ nên tăng vị trí l.
- Tăng vị trí l lên 1 đơn vị
  - $a = [2, 6, \overset{\overset{l}{\downarrow}}{5}, 3, 6, \overset{\overset{r}{\downarrow}}{8}, 9]$
  - vì $sum(l, r) = 22 > s$ nên tăng vị trí l.
- Tăng vị trí l lên 1 đơn vị
  - $a = [2, 6, 5, \overset{\overset{l}{\downarrow}}{3}, 6, \overset{\overset{r}{\downarrow}}{8}, 9]$
  - vì $sum(l, r) = 17 \le s$ nên đoạn $[l, r]$ là một đoạn tốt.
  - $ans = max(ans, r - l + 1)$
- Tăng vị trí r lên 1 đơn vị
  - $a = [2, 6, 5, \overset{\overset{l}{\downarrow}}{3}, 6, 8, \overset{\overset{r}{\downarrow}}{9}]$
  - vì $sum(l, r) = 26 > s$ nên tăng vị trí l.
- Tăng vị trí l lên 1 đơn vị
  - $a = [2, 6, 5, 3, \overset{\overset{l}{\downarrow}}{6}, 8, \overset{\overset{r}{\downarrow}}{9}]$
  - vì $sum(l, r) = 23 > s$ nên tăng vị trí l.
- Tăng vị trí l lên 1 đơn vị
  - $a = [2, 6, 5, 3, 6, \overset{\overset{l}{\downarrow}}{8}, \overset{\overset{r}{\downarrow}}{9}]$
  - vì $sum(l, r) = 17 \le s$ nên đoạn $[l, r]$ là một đoạn tốt.
  - $ans = max(ans, r - l + 1)$



Cài đặt

Để có thể tính được tổng các phần tử từ l đến r trong khi l và r đang di động, ta sẽ sử dụng biến sum để lưu lại tổng của đoạn $[l, r]$ hiện tại.

Sau khi di chuyển r sang phải, biến sum sẽ cộng thêm giá trị $a[r]$.

Trước khi di chuyển l sang phải, biến sum sẽ trừ đi giá trị $a[l]$.

```
int ans = 0, sum = 0;
for (int l = 1, r = 1; r <= n; r++) {
    sum += a[r];
    while (sum > s) {
      sum -= a[l];
      l++;
    }
    ans = max(ans, r - l + 1);
}
cout << ans;
```

Độ phức tạp

Vị trí con trỏ r luôn tăng, vị trí con trỏ l luôn tăng và luôn tăng không giá trị r.

Hơn nữa, mỗi vị trí l và r tăng không quá n lần.

Vì thế độ phức tạp của giải pháp là $O(n)$.

Luyện tập

- VNOJ - SOPENP
- VNOJ - PRODUCT
- VNOJ - KRECT
- VNOJ - VMQUABEO

Bài toán 4

Bạn được cho một dãy số nguyên như sau:

- $x_0 = 1$
- $x_{i+1} = (a \cdot x_i + x_i \ div \ b) \ mod \ c$.

Tìm n nhỏ nhất sao cho tồn tại $m < n$ và $x_m = x_n$. Dữ liệu đảm bảo n không quá $2 \cdot 10^7$.

Giới hạn: $1 \leq a \leq 10^4$ và $1 \leq b, c \leq 10^{14}$.

Phân tích

Để dễ dàng phân tích ta định nghĩa hàm f như sau:

$$f(x) = (a \cdot x + x \ div \ b) \ mod \ c$$

Dãy số của chúng ta sẽ có dạng

$$x_0 = 1, x_1 = f(x_0), x_2 = f(x_1), \ldots, x_i = f(x_{i-1}), \ldots$$

Với phép chia lấy dư cho c thì mọi $i > 0$, giá trị của x_i sẽ có giá trị nằm trong khoảng $[0, c-1]$.

Vì thế, dãy số với vô hạn phần tử này sẽ tồn tại $x_m = x_n$ với $m < n$. (theo nguyên lý Dirichlet)

Có thể thấy, khi dãy tồn tại $x_m = x_n$, dãy sẽ xuất hiện chu kỳ. Cụ thể như sau:

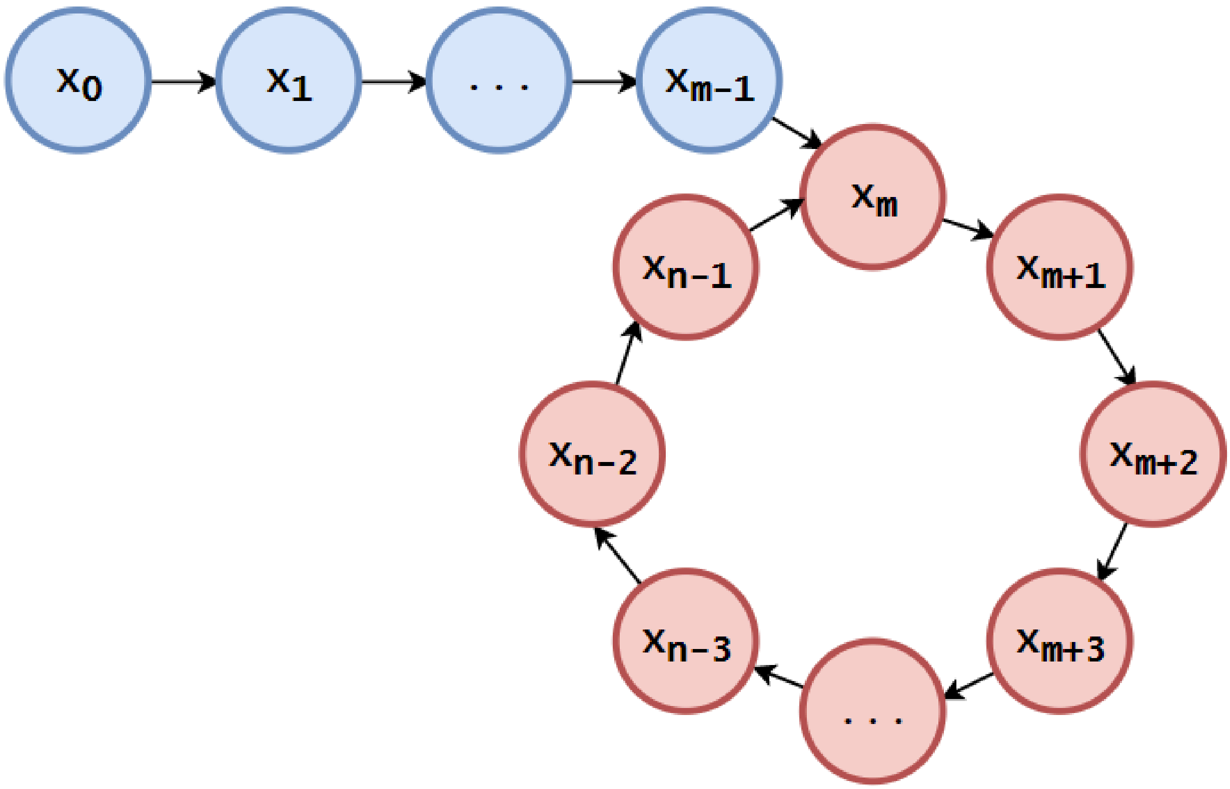
Gọi n là giá trị nhỏ nhất thỏa mãn tồn tại $m < n$ sao cho $x_m = x_n$.

$$x_0, x_1, x_2, \ldots, \underset{\uparrow}{x_m}, x_{m+1}, \ldots, x_{n-1}, \underset{\uparrow}{x_n}, \ldots$$

Khi đó, dãy sẽ có chu kỳ lặp lại các phần tử từ x_m đến x_{n-1}

$$\underbrace{x_0, x_1, \ldots, x_{m-1}}_{}, \underbrace{x_m, x_{m+1}, \ldots, x_{n-1}}_{}, \underbrace{x_m, x_{m+1}, \ldots, x_{n-1}}_{}, \underbrace{x_m, x_{m+1}, \ldots, x_{n-1}}_{}, \ldots$$

Dãy số có thể biểu diễn như hình sau đây:



Bài toán có thể giải quyết nếu chúng ta phần tử bắt đầu chu kỳ (x_μ) và độ dài của chu kỳ λ.

Cụ thể, xem ví dụ sau đây:

$$a = 8, b = 2, c = 31$$

Ta có dãy số

$$\underbrace{1, 8, 6, 20, 15}_{}, \underbrace{3, 25, 26, 4}_{}, \underbrace{3, 25, 26, 4}_{}, \underbrace{3, 25, 26, 4}_{}, \ldots$$

Giá trị n cần tìm của bài toán là $n = 9$.

Ta có thể tính được giá trị này bằng cách xác định

- phần tử bắt đầu chu kỳ x_μ.
- độ dài chu kỳ λ.

Ở đây, phần tử bắt đầu chu kỳ là x_5 và độ dài chu kỳ là 4.

Giá trị $n = \mu + \lambda = 5 + 4 = 9$.

Giải pháp

Để xác định giá trị μ và λ, ta sử dụng thuật toán Floyd's tortoise and hare

Rùa và Thỏ

Khởi tạo hai con trỏ, $toroise$ (rùa) và $hare$ (thỏ).

Tại mỗi thời điểm, ta tịnh tiến hai con trỏ này như sau:

- Tortoise (rùa): tịnh tiến một "bước"
  - Nếu hiện tại con trỏ $tortoise$ đang là x, nó sẽ được tịnh tiến đến $f(x)$.
  - $x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \rightarrow \ldots$
  - Vì dãy số của chúng ta có chu kỳ nên ta có công thức tính giá trị của con trỏ $tortoise$ sau t lần tịnh tiến:
    - $t < \mu$: x_t
    - $t \geq \mu$: $x_{\mu+(t-\mu) \ mod \ \lambda}$
- Hare (thỏ): tịnh tiến hai "bước"
  - Nếu hiện tại con trỏ $hare$ đang là x, nó sẽ được tịnh tiến đến $f(f(x))$.
  - $x_0 \rightarrow x_2 \rightarrow x_4 \rightarrow x_6 \rightarrow x_8 \rightarrow \ldots$
  - Vì dãy số của chúng ta có chu kỳ nên ta có công thức tính giá trị của con trỏ $hare$ sau t lần tịnh tiến:
    - $2t < \mu$: x_{2t}
    - $2t \geq \mu$: $x_{\mu+(2t-\mu) \ mod \ \lambda}$

Ngoài lúc ban đầu, hai con trỏ $tortoise$ và $hare$ sẽ luôn gặp nhau tại thời điểm nào đó. Thật vậy:

- $2t < \mu$:
  - Sau t lần tịnh tiến, $tortoise$ = x_t và $hare$ = x_{2t}.
  - Tuy nhiên, $\mu + \lambda$ mới bắt đầu lại chu kỳ cho nên các phần tử từ x_0 đến $x_{\mu+\lambda-1}$ phải đôi một khác nhau.
  - Vì thế $x_t \neq x_{2t}$, $tortoise$ và $hare$ chưa gặp nhau lúc này.
- $2t \geq \mu$ và $t < \mu$
  - Sau t lần tịnh tiến, $tortoise$ = x_t và $hare$ = $x_{\mu+(2t-\mu) \ mod \ \lambda}$.
  - Tuy nhiên, $\mu + \lambda$ mới bắt đầu lại chu kỳ cho nên các phần tử từ x_0 đến $x_{\mu+\lambda-1}$ phải đôi một khác nhau.
  - Vì thế $x_t \neq x_{\mu+(2t-\mu) \ mod \ \lambda}$, $tortoise$ và $hare$ chưa gặp nhau lúc này.
- $t \geq \mu$
  - Sau t lần tịnh tiến, $tortoise$ = $x_{\mu+(t-\mu) \ mod \ \lambda}$ và $hare$ = $x_{\mu+(2t-\mu) \ mod \ \lambda}$.
  - Giả sử $tortoise$ và $hare$ gặp nhau thì $\mu + (t-\mu) \ mod \ \lambda = \mu + (2t-\mu) \ mod \ \lambda \Leftrightarrow t \ mod \ \lambda = 0$.
  - Vậy, $tortoise$ và $hare$ sẽ gặp nhau sau t lần tịnh tiến, trong đó t là số nguyên có giá trị lớn hơn hoặc bằng μ và chia hết cho λ.
  - Trừ lúc khởi tạo, hai con trỏ $tortoise$ và $hare$ sẽ gặp nhau khi giá trị của cả hai con trỏ là $x_{\mu+(\lambda-\mu \ mod \ \lambda) \ mod \ \lambda}$.

Cách cài đặt để $tortoise$ và $hare$ gặp nhau:

```
int tortoise = 1, hare = 1;
while (true) {
    tortoise = f(tortoise);
    hare = f(f(hare));
    if (tortoise == hare)
```

```
        break;
    }
```

Tìm μ

Khởi tạo một con trỏ mới $p = x_0$, con trỏ này được tịnh tiến giống như con trỏ $tortoise$.

Tịnh tiến cùng lúc hai con trỏ p và $tortoise$ và dừng lại cho đến khi chúng gặp nhau.

Số lần tịnh tiến ở đây chính là μ.

Chứng minh:

- Trong những lần tịnh tiến từ 0 đến $\mu - 1$, con trỏ p nhận giá trị từ x_0 đến $x_{\mu-1}$ (các giá trị không có trong chu kỳ) . Còn con trỏ $tortoise$, vì đã nằm ở chu kỳ, nên giá trị của $tortoise$ sẽ nhận giá trị của các phần tử có trong chu kỳ. Vì thế $tortoise$ và p chưa gặp nhau.
- Hai con trỏ p và $tortoise$ gặp nhau tại lần tịnh tiến thứ μ:
  - Con trỏ p có giá trị x_μ.
  - Lúc chưa tịnh tiến p, con trỏ $tortoise$ có giá trị $x_{\mu+(t-\mu) \ mod \ \lambda}$ (đã nêu ở mục Rùa và Thỏ). Vì đã ở trong chu kỳ cho nên, sau khi tịnh tiến μ lần con trỏ $tortoise$ sẽ có giá trị là $x_{\mu+(t) \ mod \ \lambda}$. Mà t là số nguyên dương chia hết cho λ, cho nên con trỏ $tortoise$ có giá trị là x_μ.

Cách cài đặt tìm μ:

```
int mu = 0, p = 1;
while (p != tortoise) {
    p = f(p);
    tortoise = f(tortoise);
    mu++;
}
```

Tìm λ

Bây giờ cả hai con trỏ $tortoise$ và p đang có giá trị là x_μ.

Chúng ta giữ nguyên giá trị $tortoise$, và tịnh tiến p cho đến khi p có giá trị x_μ lại.

Vì p đã ở trong chu kỳ cho nên, sau khi tinh tiến λ lần, p sẽ lại có giá trị là x_μ.

```
int lambda = 0;
while (true) {
    lambda++;
    p = f(p);
    if (tortoise == p)
      break;
}
```

Để hiểu rõ hơn, ta hãy cùng xem qua một số ví dụ sau đây:

$a = 2, b = 2, c = 32$

Ta có dãy số

$$\underbrace{1, 2, 5, 12}, \underbrace{30, 11, 27, 3, 7, 17, 10, 25}, \underbrace{30, 11, 27, 3, 7, 17, 10, 25}, \underbrace{30, 11, 27, 3, 7, 17, 10, 25}, \ldots$$

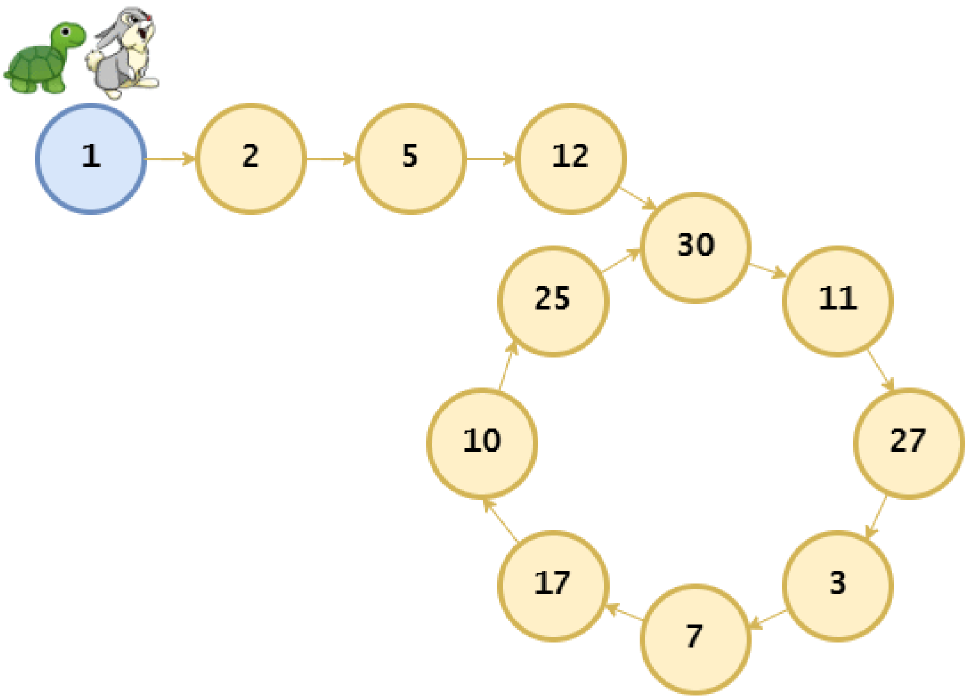
Giá trị n cần tìm của bài toán là $n = 12$.

Ta có thể tính được giá trị này bằng cách xác định

- phần tử bắt đầu chu kỳ x_μ.
- độ dài chu kỳ λ.

Ở đây, phần tử bắt đầu chu kỳ là x_4 và độ dài chu kỳ là 8.

Giá trị $n = \mu + \lambda = 4 + 8 = 12$.



Độ phức tạp

Khi khởi tạo hai con trỏ $tortoise$ và $hare$ ở x_0, hai con trỏ sẽ gặp lại nhau lần đầu tiên sau t bước.

Cụ thể t là số nguyên nhỏ nhất sao cho t có giá trị lớn hơn hoặc bằng μ và chia chết cho λ.

Vì thế việc xác định được vị trí $tortoise$ và $hare$ gặp nhau sẽ mất không quá $\mu + \lambda$ bước. Hơn nữa, việc xác định μ mất μ bước, xác định λ mất λ bước.

Kết luận: độ phức tạp của bài toán là $O(\mu + \lambda)$. (trong đó $\mu + \lambda \leq 2 \cdot 10^7$)

Luyện tập

- LODOJ - TORHAR
- CODEFORCES - Sequence analysis
- CODEFORCES - Pseudo-Random Number Generator
- CODEFORCES - Cooperative Game