HAI CON TRỎ - TWO POINTERS

Phan Văn Việt - Chuyên Vĩnh Phúc

Ngày 4 tháng 6 năm 2023

Tập hợp các bài toán sử dụng kỹ thuật hai con trỏ.

1 Trò chơi [C381A]

Tí và Tèo đang chơi một trò chơi. Quy tắc của trò chơi rất đơn giản. Hai người chơi có n lá bài được xếp thành một hàng trên bàn. Mỗi lá bài ghi một số nguyên, tất cả các số trên các lá bài đều khác nhau. Hai người chơi lần lượt thực hiện ượt đi, Tí đi trước. Trong lượt của mình, mỗi người chơi có thể lấy một lá bài: hoặc là lá bài ở vị trí ngoài cùng bên trái, hoặc là lá bài ở vị trí ngoài cùng bên phải. Trò chơi kết thúc khi không còn lá bài nào trên bàn. Người chơi có tổng số lớn nhất trên các lá bài của mình vào cuối trò chơi sẽ chiến thắng.

Tí và Tèo luôn chọn lá bài có số có số lớn nhất có thể trong lượt chơi của mình.

Yêu cầu: Hãy in ra số điểm tương ứng của Tí và Tèo khi trò chơi kết thúc.

Dữ liệu

- Dòng 1: ghi số nguyên n ($1 \leq N \leq 1000$) - số lượng lá bài trên bàn;
- Dòng 2: ghi n số nguyên $a_1, a_2, \cdots, a_n$ ($1 \leq a_i \leq 1000, \forall i = 1 \to n$) ứng với các số ghi trên các lá bài. Các số này là đôi một phân biệt.

Kết quả

- Ghi trên một dòng gồm hai số nguyên ứng với điểm của Tí và Tèo sau khi kết thúc trò chơi.

Ví dụ

Input	Output
4 4 1 2 10	12 5

Giải thích ví dụ

- Hai lượt chơi, Tí chọn tương ứng hai lá bài có số là 10 và 2.

2 Chọn cọc [DCPC19D1P2]

Có n chiếc cọc, chiếc thứ i có chiều cao là h_i. Bạn cần chọn một số cọc trong số đó sao cho chênh lệch giữa cọc cao nhất và cọc thấp nhất trong các cọc được chọn không lớn hơn k.

Yêu cầu: Hãy cho biết có thể chọn tối đa là bao nhiêu cọc thoả mãn điều kiện.

Dữ liệu

- Dòng 1: ghi hai số nguyên n, k ($1 \leq N \leq 2 \times 10^5$) - số lượng cọc và chênh lệch chiều cao tối đa cho phép;
- Dòng 2: ghi n số nguyên $h_1, h_2, \cdots, h_n$ ($1 \leq h_i \leq 10^9, \forall i = 1 \to n$) ứng với chiều cao của những chiếc cọc.

Kết quả

- Ghi một số nguyên duy nhất là số lượng cọc tối đa chọn được.

Ví dụ

Input	Output
5 3 25 9 1 6 8	3

Giải thích ví dụ

- Chiều cao các cọc được chọn là $9, 6, 8$.

Subtasks

- **Subtask 1:** 10% số điểm có $1 \leq n \leq 20$;
- **Subtask 2:** 15% số điểm có $1 \leq n \leq 2 \times 10^3$;
- **Subtask 3:** 75% số điểm không có ràng buộc bổ sung.

3 Xoá kí tự [C1203D1]

Cho hai xâu s, t chỉ gồm các chữ cái tiếng Anh viết thường. Dữ liệu vào đảm bảo t là xâu con của s. Tức là có thể nhận được t bằng cách xoá bỏ đi không hoặc một số kí tự (không nhất thiết phải liền kề) của s.

Ví dụ, các xâu "hung", "hg", "hn" đều là xâu con của "hung".

Yêu cầu: Hãy tìm cách xoá đi đúng một đoạn các phần tử liền nhau của s sao cho sau khi xoá t vẫn là xâu con của s và đoạn bị xoá là dài nhất có thể. In ra độ dài đoạn con được chọn để xoá.

Dữ liệu

- Dòng 1: ghi xâu s ($1 \leq |s| \leq 200$);
- Dòng 2: ghi xâu t ($1 \leq |t| \leq 200$).

Kết quả

- Ghi một số nguyên duy nhất là độ dài của xâu được chọn để xoá.

Ví dụ

Input	Output
bbaba bb	3

Giải thích ví dụ

- Có thể xoá xâu các kí tự được gạch chân "bb<u>aba</u>".

4 Max f_a [C660C]

Cho mảng a gồm n phần tử 0 và 1. Gọi f_a là độ dài đoạn con dài nhất các phần tử liên tiếp toàn giá trị 1 của a.

Yêu cầu: Bằng cách thay đổi nhiều nhất k số 0 thành số 1, hãy làm cho f_a lớn nhất có thể. In ra dãy sau biến đổi.

Dữ liệu

- Dòng 1: ghi hai số nguyên n, k $(1 \leq n \leq 3 \times 10^5, 0 \leq k \leq n)$ - số lượng phần tử của dãy;
- Dòng 2: ghi n số nguyên $a_1, a_2, \cdots, a_n$ $(0 \leq a_i \leq 1, \forall i = 1 \to n)$ - giá trị các phần tử của dãy.

Kết quả

- Dòng 1: ghi số nguyên m là giá trị lớn nhất của f_a sau biến đổi.
- Dòng 2: ghi n số nguyên là dãy sau khi biến đổi. Nếu có nhiều phương án biến đổi, chỉ cần in ra một phương án đúng.

Ví dụ

Input	Output
7 1 1 0 0 1 1 0 1	4 1 0 0 1 1 1 1

5 Đếm đoạn con [THICC17P5]

Cho dãy gồm n số nguyên $a_1, a_2, \cdots, a_n (1 \leq a_i \leq 10^6, \forall i = 1 \to n)$.

Yêu cầu: Hãy đếm xem trong dãy có bao nhiêu dãy con (các phần tử liên tiếp) có ít nhất k phần tử phân biệt.

Dữ liệu

- Dòng 1: ghi hai số nguyên n, k $(1 \leq k \leq n \leq 10^5)$;
- Dòng 2: ghi n số nguyên $a_1, a_2, \cdots, a_n$ $(1 \leq a_i \leq 10^6, \forall i = 1 \to n)$.

Kết quả

- Ghi một số nguyên duy nhất là số mảng đếm được.

Ví dụ

Input	Output
5 3 1 2 3 4 5	6

Giải thích ví dụ

- Các đoạn con đếm được là: $(1, 2, 3)$, $(2, 3, 4)$, $(3, 4, 5)$, $(1, 2, 3, 4)$, $(2, 3, 4, 5)$, $(1, 2, 3, 4, 5)$

Subtasks

- **Subtask 1:** 40% số điểm có $1 \leq n \leq 100$;
- **Subtask 2:** 60% số điểm còn lại không có ràng buộc bổ sung.

6 Đoạn ngắn nhất [D18C2P4]

Cho dãy gồm n số nguyên $a_1, a_2, \cdots, a_n (0 \leq a_i \leq 5 \times 10^9, \forall i = 1 \to n)$.

Yêu cầu: Hãy tìm đoạn con các phần tử liên tiếp ngắn nhất sao cho tổng các phần tử của đoạn con này không nhỏ hơn m.

Dữ liệu

- Dòng 1: ghi hai số nguyên n, m $(1 \leq n \leq 5 \times 10^5, 1 \leq m \leq 10^{18})$;
- Dòng 2: ghi n số nguyên $a_1, a_2, \cdots, a_n (0 \leq a_i \leq 5 \times 10^9, \forall i = 1 \to n)$.

Kết quả

- Ghi một số nguyên duy nhất là độ dài đoạn con tìm được. Nếu không tìm được đoạn con nào thoả mãn, ghi '-1'

Ví dụ

Input	**Output**
5 6 1 3 2 5 0	2
5 30 1 2 3 4 5	-1

Subtasks

- **Subtask 1:** 10% số điểm có $1 \leq n \leq 800$;
- **Subtask 2:** 30% số điểm có $1 \leq n \leq 3000$;
- **Subtask 3:** 60% số điểm còn lại không có ràng buộc bổ sung.

7 Tổng ba số [TSUM]

Cho dãy gồm n phần tử $a_1, a_2, \cdots, a_n$ $(1 \leq a_i \leq 10^9, \forall i = 1 \to n)$.
Yêu cầu: Hãy chọn ra ba phần tử phân biệt trong dãy sao cho tổng của chúng bằng x.

Dữ liệu

- Dòng 1: ghi hai số nguyên n, x $(1 \leq N \leq 5000, 1 \leq x \leq 10^9)$ - số lượng lá bài trên bàn;
- Dòng 2: ghi n số nguyên $a_1, a_2, \cdots, a_n$ $(1 \leq a_i \leq 10^9, \forall i = 1 \to n)$.

Kết quả

- Ghi trên một dòng gồm ba số nguyên ứng với chỉ số của ba phần tử được chọn. Nếu có nhiều bộ ba thoả mãn, bạn chỉ cần in một bộ bất kỳ trong đó. Nếu không có bộ ba nào thoả mãn, in ra `IMPOSSIBLE`.

Ví dụ

Input	Output
4 8 2 7 5 1	1 3 4

Giải thích ví dụ

- $a[1] + a[3] + a[4] = 2 + 5 + 1 = 8$

8 Lây nhiễm virus [CORONA]

Trên trục toạ độ Ox, có n người, người thứ i đứng ở toạ độ x_i.

Trong số n người này, có đúng một người đã bị nhiễm virus CORONA nhưng chưa xác định được chính xác ai là người nhiễm virus. Virus sẽ lây từ người bị nhiễm sang người chưa bị nhiễm nếu khoảng cách giữa hai người này nhỏ hơn hoặc bằng 2 trong thời gian cực nhanh mà ta có thể coi như là bằng 0.
Yêu cầu: Hãy xác định xem sau khi virus lây lan sang tất cả những người có thể, trong tình huống tốt nhất, có ít nhất bao nhiêu người bị nhiễm bệnh và trong tình huống xấu nhất, có nhiều nhất bao nhiêu người bị nhiễm bệnh (tuỳ thuộc vào vị trí của người nhiễm bệnh ban đầu).

Dữ liệu

- Dòng 1: ghi số nguyên T ($1 \leq T \leq 2000$) - số lượng truy vấn cần trả lời;
- Tiếp theo là T truy vấn, mỗi truy vấn gồm:
  - Dòng đầu tiên của truy vấn ghi số nguyên n ($2 \leq n \leq 8$);
  - Dòng thứ hai của truy vấn ghi n số nguyên $x_1, x_2, \cdots, x_n$, ($x_1 < x_2 \cdots < x_n, 0 \leq x_i \leq 10, \forall i = 1 \to n$).

Kết quả

- Ghi trên T dòng, mỗi dòng gồm hai số nguyên ứng với số người bị nhiễm trong trường hợp tốt nhất và số người bị nhiễm trong trường hợp xấu nhất của truy vấn tương ứng.

Ví dụ

Input	Output
3 2 3 6 3 1 3 5 5 1 2 5 6 7	1 1 3 3 2 3

Giải thích ví dụ

- Truy vấn 1: Khoảng cách giữa hai người là 3 nên sẽ không có thêm người nào bị lây bệnh;
- Truy vấn 2: Khoảng cách giữa hai người liền kề bất kỳ là 2 nên trong trường hợp nào thì cuối cùng tất cả đều sẽ bị lây bệnh;
- Truy vấn 3:
  - Một trong những trường hợp khiến kết quả ít người nhiễm bệnh nhất là người nhiễm bệnh ban đầu ở vị trí 1. Khi đó, sẽ chỉ có người ở vị trí 2 bị lây bệnh;
  - Một trong những trường hợp khiến nhiều người bị lây bệnh nhất là người nhiễm bệnh ban đầu ở vị trí 5, khi đó, những người ở các vị trí 6 và 7 sẽ bị lây bệnh.

Subtasks

- **Subtask 1:** 10% số điểm có $n \leq 3$;
- **Subtask 2:** 90% số điểm không có ràng buộc bổ sung.

9 Xâu mạnh [SSTRING]

Một xâu được gọi là **xâu mạnh** nếu trong xâu đó có chứa ít nhất k kí tự '*' liên tiếp.

Yêu cầu: Bạn cần trả lời T truy vấn, mỗi truy vấn cho một xâu, yêu cầu cho biết xâu đó có phải là **xâu mạnh** hay không.

Dữ liệu

- Dòng 1: ghi số nguyên T ($1 \leq T \leq 10$) - số lượng truy vấn cần trả lời;
- Tiếp theo là T truy vấn, mỗi truy vấn gồm:
  - Dòng đầu tiên của truy vấn hai ghi số nguyên n, k ($1 \leq k \leq n \leq 10^6$);
  - Dòng thứ hai của truy vấn ghi xâu S gồm n kí tự chữ cái tiếng Anh thường hoặc kí tự '*'. Tổng của tất cả các n trong T truy vấn không quá 10^6.

Kết quả

- Ghi trên T dòng, mỗi dòng với câu trả lời cho của truy vấn tương ứng. Ghi `YES` nếu xâu đã cho là xâu mạnh, ngược lại ghi `NO`.

Ví dụ

Input	Output
3 5 2 *a*b* 5 2 *a**b 5 1 abcde	NO YES NO

Giải thích ví dụ

- Truy vấn 1: Trong xâu, độ đoạn dài nhất toàn kí tự '*' là $1 < 2$ nên xâu không phải xâu mạnh;
- Truy vấn 2: Trong xâu, độ đoạn dài nhất toàn kí tự '*' là $2 \geq 2$ nên xâu không phải xâu mạnh;
- Truy vấn 3: Trong xâu, không có kí tự '*' nên xâu không phải xâu mạnh;

Subtasks

- **Subtask 1:** 30% số điểm có tổng của các n trong tất cả các truy vấn không quá 10^4;
- **Subtask 2:** 70% số điểm không có ràng buộc bổ sung.

10 Tạo dãy đối xứng [MAKEPAL]

Cho dãy gồm n phần tử $a_1, a_2, \cdots, a_n$ ($1 \leq a_i \leq 10^5, \forall i = 1 \to n$). Và thao tác gồm các bước như sau:

1. Chọn một phần tử a_i ($a_i \geq 1, \forall i1 \to n$) trong dãy đã cho;
2. Chia phần tử đó thành hai phần tử mới X, Y sao cho $X + Y = A_i$ và $X, Y \geq 1$

Lưu ý rằng độ dài của dãy sẽ tăng lên 1 sau khi thực hiện một lần thao tác.

Ví dụ: Với dãy $[4, 6, 7, 2]$ thực hiện thao tác tách phần tử thứ nhất thành hai số $4 = 1+3$ ta có dãy $[1, 3, 6, 7, 2]$.

Yêu cầu: Bạn cần trả lời T truy vấn, mỗi truy vấn cho một dãy và yêu cầu cho biết cần thực hiện ít nhất bao nhiêu thao tác để dãy trở thành dãy đối xứng. Dãy đối xứng là dãy mà đọc từ trái sang cũng giống như đọc từ phải sang.

Dữ liệu

- Dòng 1: ghi số nguyên T ($1 \leq T \leq 10^5$) - số lượng truy vấn cần trả lời;
- Tiếp theo là T truy vấn, mỗi truy vấn gồm:
  - Dòng đầu tiên của truy vấn hai ghi số nguyên n ($1 \leq n \leq 10^5$) là số lượng phần tử của dãy;
  - Dòng thứ hai của dãy gồm n số nguyên $a_1, a_2, \cdots, a_n$ ($1 \leq a_i \leq 10^5, \forall i = 1 \to n$). Tổng của tất cả các n trong T truy vấn không quá 2×10^5. Dữ liệu đảm bảo bạn sẽ không nhận được phán quyết `TLE` với thuật toán có độ phức tạp $O(T \times N)$ với N là Tổng của tất cả các n trong T truy vấn.

Kết quả

- Ghi trên T dòng, mỗi dòng gồm một số nguyên duy nhất là câu trả lời cho truy vấn tương ứng.

Ví dụ

Input	Output
3 4 3 7 6 4 5 1 4 5 4 1 5 1 2 3 4 5	2 0 4

Giải thích ví dụ

- Truy vấn 1:
  - Thao tác 1: chọn phần tử thứ 2 và tách nó thành $7 = 1 + 6$, dãy trở thành $[3, 1, 6, 6, 4]$;
  - Thao tác 1: chọn phần tử thứ 5 của dãy hiện tại và tách nó thành $4 = 1 + 3$, dãy trở thành $[3, 1, 6, 6, 3, 1]$.
- Truy vấn 2: dãy đã đối xứng sẵn rồi nên không cần thực hiện thao tác nào nữa.

Hướng dẫn trò chơi [C1381A]

Giải thuật

- Bài tập này đơn giản chỉ là thực hiện theo yêu cầu của bài toán.
- Việc cài đặt bài toán này chính là thực hiện ý tưởng của kĩ thuật hai con trỏ đơn giản.

Mã triển khai C++

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e3 + 1;
int n, a[N], v[2]; //v[0] la tong diem cua Ti, v[1] la tong diem cua Teo

int main()
{
  #define taskname "C381A"

  if (fopen (taskname".inp", "r"))
  {
    freopen (taskname".inp", "r", stdin);
    freopen (taskname".out", "w", stdout);
  }

  cin.tie (0)->sync_with_stdio (0);
  cin >> n;

  for (int i = 0; i < n; i++) cin >> a[i];

  int i = 0, j = n - 1, turn = 0;

  /**
  i: con tro thu nhat, xac dinh chi so ngoai cung ben trai
  j: con tro thu hai, xac dinh chi so ngoai cung ben phai
  turn = 0 la luot cua Ti, turn = 1 la luot cua Teo
  Vong lap while dich chuyen hai con tro tu hai dau cho den khi duyet het mang
  */
  while (i <= j)
  {
    if (a[i] <= a[j]) v[turn] += a[j--];    //Lay phan lon hon, tang diem cho nguoi dang
   den luot

    else v[turn] += a[i++];

    turn = 1 - turn;    //Doi luot choi cho nguoi con lai
  }

  cout << v[0] << " " << v[1];
  return 0;
}

```

Độ phức tạp tính toán

1. Khởi tạo các biến và mảng:
   - Khai báo mảng `a` có kích thước N và mảng `v` có 2 phần tử (O(1)).
   - Khai báo biến `n` (O(1)).
2. Đọc dữ liệu vào:

- Đọc giá trị `n` (O(1)).
- Duyệt qua mảng `a` và đọc các giá trị (O(n)).

3. Thiết lập con trỏ và biến đếm:
   - Khởi tạo biến `i = 0`, `j = n - 1`, và `turn = 0` (O(1)).
4. Vòng lặp chính:
   - Trong khi `i <= j`, thực hiện các bước sau:
     - Nếu `a[i] <= a[j]`, thì tăng tổng điểm của người đang đến lượt (`v[turn]`) bằng giá trị `a[j]` và giảm `j` đi 1 (O(1)).
     - Ngược lại, tăng tổng điểm của người đang đến lượt (`v[turn]`) bằng giá trị `a[i]` và tăng `i` lên 1 (O(1)).
     - Đổi lượt chơi bằng cách gán `turn = 1 - turn` (O(1)).
5. In kết quả:
   - In giá trị tổng điểm của Ti và Teo (O(1)).

Tổng thời gian thực hiện của thuật toán là O(n), với n là kích thước của mảng `a`.

Hướng dẫn - chọn cọc [DCPC19D1P2]

Subtask 1

- Sử dụng quay lui để duyệt mọi tập con của mảng và chọn ra tập con kích thước lớn nhất thoả mãn điều kiện. Thông thường, các thí sinh sẽ ít khi làm riêng subtasks này mà sẽ gộp nó cùng với subtask 2 do việc triển khai mã quay lui thường không quá dễ đối với các thí sinh cộng với việc subtasks này chỉ được 10% điểm.
- Độ phức tạp $\Theta(n^2 \times n)$ với $\theta(2^n)$ cho việc duyệt các tập và $\Theta(n)$ cho việc kiểm tra xem tập có thoả mãn điều kiện hay không.

Subtask 2

- Nhận thấy rằng sẽ luôn có lợi thế nếu chọn các cọc có chiều cao lệch nhau càng ít càng tốt. Do đó, nếu bạn đã sắp xếp lại theo chiều cao của các cọc thì câu hỏi lúc này trở thành: tìm đoạn con dài nhất gồm các phần tử liên tiếp mà chênh lệch không quá k;
- Duyệt mọi cặp chỉ số l, r bằng hai vòng lặp *for* để tìm ra đoạn dài nhất, lúc này chênh lệch lớn nhất các phần tử trong đoạn có thể tính bằng $a_r - a_l$;
- Độ phức tạp $\Theta(n^2)$.

Subtask 3

- Để có thể lấy điểm của subtask này, bạn cần tối ưu hoá thuật toán duyệt trâu của subtasks 2 bằng tìm kiếm nhị phân hoặc kĩ thuật hai con trỏ.
- Dưới đây là mã triển khai sử dụng hai con trỏ với độ phức tạp là $\Theta(n \log n)$ cho việc sắp xếp các phần tử.

Mã triển khai C++

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 2e5 + 5;
int n, k;
int h[N];
int lo = 0, ans = 0;


int main()
{
  #define taskname "cpc19c1p2"

  if (fopen (taskname".inp", "r"))
  {
    freopen (taskname".inp", "r", stdin);
    freopen (taskname".out", "w", stdout);
  }

  cin.tie (0)->sync_with_stdio (0);
  cin >> n >> k;

  for (int i = 0; i < n; i++) cin >> h[i];

  sort (h, h + n);

  /**
  Duy tri hai bien (con tro), lo la chi so dau mut ben trai doan dang xet
```

```
  hi la chi so dau mut ben phai doan dang xet;
  Thu moi vi tri co the cua hi, voi moi vi tri cua hi, di chuyen low xich dan ve phia
  hi cho den khi chenh lech cua doan [lo, hi] nho hon hoac bang k.
  */

  for (int hi = 1; hi < n; hi++)
  {
    while (h[hi] - h[lo] > k) lo++;

    if (ans < hi - lo) ans = hi - lo; // Cap nhat lai ans cho moi doan [lo,hi] tim duoc
  }

  cout << ans + 1 << '\n';
  return 0;
}
```

Độ phức tạp tính toán

1. Khởi tạo biến và mảng:
   - Độ phức tạp: $O(1)$
2. Đọc dữ liệu vào:
   - Độ phức tạp: $O(n)$
3. Sắp xếp mảng:
   - Độ phức tạp: $O(n \log n)$
4. Vòng lặp chính:
   - Độ phức tạp: $O(n)$
5. Vòng lặp trong vòng lặp:
   - Độ phức tạp: $O(n)$

Độ phức tạp tổng thể của chương trình là $O(n \log n)$.

Hướng dẫn xoá kí tự [C1203D1]

- Thử xoá tất cả các đoạn $[l, r]$ có thể với hai vòng lặp;
- Với mỗi đoạn thử xoá, duyệt đoạn trước và sau đoạn thử xoá để tìm xem có thấy *sb* trong *sa* hay không (hàm ok(l,r)). Nếu có thì cập nhật **Max** với độ dài đoạn đang thử xoá.
- Hàm ok(l,r) được thực hiện như sau:
  - Sử dụng phương pháp hai con trỏ. Con trỏ thứ nhất là i dùng để duyệt mọi chỉ số trong *sa*, con trỏ thứ hai là x dùng để lưu chỉ số đang xét của *sb*;
  - Do đã xoá đoạn l, r nên dòng thứ 14 dùng để bỏ qua các phần tử đã bị xoá;
  - Nếu kí tự đang xét của sa khớp với kí tự đang xét của *sb* thì tăng x lên 1 (xét kí tự kế tiếp của *sb*);
  - Kết luận ngay *sb* là con của *sa* sau khi xoá nếu đã khớp đủ tất cả các kí tự có trong *sb*.
  - nếu không khớp đủ kí tự có trong *sb* thì kết luận *sb* không là con của *sa* sau khi xoá.

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 1;
string sa, sb; // tuong ung voi s va t trong de bai
int na, nb, ans = 0;

// ham ok(l,r) tra ve true neu sau khi xoa doan [l,r] sb van la con cua sa
bool ok (int l, int r)
{
  int x = 0;

  for (int i = 0; i <= na; i++)
  {
    if (i >= l && i <= r) continue;

    if (sb[x] == sa[i]) x++;

    if (x == nb)  return true;
  }

  return false;
}
int main()
{
  #define taskname "C1203D1"

  if (fopen (taskname".inp", "r"))
  {
    freopen (taskname".inp", "r", stdin);
    freopen (taskname".out", "w", stdout);
  }

  cin.tie (0)->sync_with_stdio (0);
  cin >> sa >> sb;
  na = sa.size(); // do dai cua xau sa
  nb = sb.size(); // do dai cua xau sb

  \\Duyet moi doan [l,r] co the xoa
  for (int l = 0; l < na; ++l)
  for (int r = l; r < na; ++r)
  {
    if (ok (l, r)) ans = max (ans, r - l + 1);
  }

```

```
45      cout << ans << endl;
46      return 0;
47  }
```

Hướng dẫn - Max f_a [C660C]

Giải thuật cụ thể

- Khởi tạo biến $ans = 0$ để lưu trữ độ dài dãy con dài nhất thỏa mãn yêu cầu bài toán và biến L, R để lưu vị trí bắt đầu và kết thúc của dãy con đó.
- Tính mảng tổng tiền tố pre[], trong đó pre[i] là tổng số phần tử 0 từ đầu đến vị trí i trong dãy a.
- Duyệt qua dãy a:
  - Khởi tạo left = 1 và right = 1.
  - Duyệt từ left = 1 đến n:
  - Tăng biến right cho đến khi tổng số phần tử 0 trong đoạn từ left đến right vượt quá k hoặc right vượt quá kích thước mảng.
  - So sánh độ dài của dãy con hiện tại (right - left) với độ dài dãy con tối đa đã tìm thấy (ans).
  - Nếu độ dài hiện tại lớn hơn ans, cập nhật giá trị ans = right - left, L = left, R = right - 1.
- In ra giá trị ans.
- Duyệt qua mảng a:
- Nếu vị trí i nằm trong khoảng [L, R], in ra giá trị 1.
- Ngược lại, in ra giá trị a[i].

Mã triển khai C++

```
#include <bits/stdc++.h>
using namespace std;
const int N = 3e5 + 2;
int n, k;
int a[N], pre[N];
int ans = 0, L, R;


int main()
{
  #define taskname "C660C"

  if (fopen (taskname".inp", "r"))
  {
    freopen (taskname".inp", "r", stdin);
    freopen (taskname".out", "w", stdout);
  }

  cin.tie (0)->sync_with_stdio (0);
  cin >> n >> k;

  for (int i = 1; i <= n; i++)
  {
    cin >> a[i];
    pre[i] = pre[i - 1] + (1 - a[i]);
  }

  for (int left = 1, right = 1; left <= n; ++left)
  {
    while (pre[right] - pre[left - 1] <= k && right <= n)
    ++right;

    if (ans < right - left)
```

```
    {
      ans = right - left;
      L = left;
      R = right - 1;
    }
  }

  cout << ans << '\n';

  for (int i = 1; i <= n; ++i)
  {
    if (i >= L && i <= R) cout << 1 << ' ';

    else cout << a[i] << ' ';
  }

  return 0;
}
```

Tính toán độ phức tạp

Dưới đây là phân tích độ phức tạp của từng phần của thuật toán trên:

1. Khởi tạo mảng và biến ($O(1)$):
   - Khai báo mảng `a` và `pre` có kích thước N ($O(1)$).
   - Khai báo biến `n`, `k`, `ans`, `L`, `R` ($O(1)$).
2. Đọc dữ liệu vào ($O(n)$):
   - Đọc giá trị `n`, `k` ($O(1)$).
   - Duyệt qua mảng `a` và tính mảng `pre` ($O(n)$).
3. Tìm đoạn con dài nhất ($O(n)$):
   - Duyệt qua mảng `a` ($O(n)$).
   - Trong mỗi lần duyệt, có một vòng lặp `while` để duyệt `right` nhưng tổng số lần duyệt `right` trong các lần duyệt tối đa là n lần ($O(n)$).
   - Cập nhật giá trị `ans`, `L`, `R` ($O(1)$).
4. In kết quả ($O(n)$):
   - In giá trị `ans` ($O(1)$).
   - Duyệt qua mảng `a` và in kết quả ($O(n)$).

Tổng thời gian thực hiện của thuật toán là ($O(n)$), trong đó n là kích thước của mảng `a`. Do đó, độ phức tạp của thuật toán được xác định là ($O(n)$).

Hướng dẫn - Đoạn ngắn nhất [D18C2P4]

Subtask 1 + 2

- Duyệt tất cả các đoạn con, kiểm tra tổng đoạn con bằng mảng tổng tiền tố;
- Độ phức tạp $O(n^2)$

Subtask 3

- Nhận xét: Bởi vì giá trị các phần tử là không âm nên nếu đoạn $[l, r]$ có tổng không nhỏ hơn m thì đoạn $[l, r+1]$ cũng sẽ có tổng không nhỏ hơn m.
- Từ nhận xét trên, ta có thể sử dụng hai con trỏ như sau:
  - Ban đầu $l = r = 1$;
  - Lặp lại thao tác sau cho đến khi l, r không còn thuộc dãy nữa:
    - Dịch r sang phải cho đến khi tổng đoạn lớn hơn hoặc bằng m. Lúc này đoạn bắt đầu tại l sẽ kết thúc tại r. Đem so sánh để lấy min các đoạn thoả mãn;
    - Dịch l sang phải một vị trí.
- Độ phức tạp $O(n)$ do tổng tất cả các lần dịch chuyển của r là n.

Mã triển khai C++

```cpp
#include <bits/stdc++.h>
using namespace std;

const int N = 5e5 + 1;
int n, ans;
long long m, a[N], pre[N];

int main()
{
  #define taskname "boss"

  if (fopen (taskname".inp", "r"))
  {
    freopen (taskname".inp", "r", stdin);
    freopen (taskname".out", "w", stdout);
  }

  cin.tie (0)->sync_with_stdio (0);
  cin >> n >> m;

  for (int i = 0; i < n; ++i)
  {
    cin >> a[i];
    pre[i + 1] = pre[i] + a[i];
  }

  if (pre[n] < m)
  {
    cout << -1 << endl;
    return 0;
  }

  ans = n;

  for (int l = 1, r = 1; l <= n; ++l)
  {
```

```
    while (pre[r] - pre[l - 1] < m && r <= n) ++r;

    if (pre[r] - pre[l - 1] >= m) ans = min (ans, r - l + 1);
  }

  cout << ans << '\n';
  return 0;
}

```

Một cách triển khai khác

```
#include <bits/stdc++.h>
using namespace std;

const int N = 5e5 + 1;
int n, ans;
long long m, a[N], pre[N];

int main()
{
  #define taskname "boss"

  if (fopen (taskname".inp", "r"))
  {
    freopen (taskname".inp", "r", stdin);
    freopen (taskname".out", "w", stdout);
  }

  cin.tie (0)->sync_with_stdio (0);
  cin >> n >> m;

  for (int i = 0; i < n; ++i)
  {
    cin >> a[i];
    pre[i + 1] = pre[i] + a[i];
  }

  if (pre[n] < m)
  {
    cout << -1 << endl;
    return 0;
  }

  ans = n;

  for (int i = n; i >= ans; --i)
  {
    while (pre[i] - pre[i - ans + 1] >= m) ans--;
  }

  cout << ans << '\n';
  return 0;
}

```

Hướng dẫn giải - Tổng ba số [TSUM]

- Gọi ba số cần tìm là a_i, a_j, b_k;
- Duyệt mọi a_i trong dãy, với mỗi a_i được chọn, sử dụng hai con trỏ để tìm hai số còn lại sao cho có tổng bằng $x - a_i$.
- Độ phức tạp tổng thể $O(n)$.

Mã triển khai C++

```cpp
#include <bits/stdc++.h>
const int N = 5e3 + 5;
using namespace std;
int n, x;
pair<int, int> a[N];
int main()
{
  #define taskname "TSUM"

  if (fopen (taskname".inp", "r"))
  {
    freopen (taskname".inp", "r", stdin);
    freopen (taskname".out", "w", stdout);
  }

  ios_base::sync_with_stdio (0);
  cin.tie (0);
  cin >> n >> x;

  for (int i = 1; i <= n; ++i)
  {
    cin >> a[i].first;
    a[i].second = i;
  }

  sort (a + 1, a + n + 1);
  for (int i = 1; i <= n; ++i)
  {
    for (int l = 1, r = n; l <= r;)
    {
      int ai = a[i].first, al = a[l].first, ar = a[r].first;
      int sum = x - ai;

      if (l < i && i < r && sum == al + ar)
      {
        cout << a[l].second << ' ' << a[i].second << ' ' << a[r].second << '\n';
        return 0;
      }

      else al + ar < sum ? ++l : --r;
    }
  }

  cout << "IMPOSSIBLE" << '\n';
  return 0;
}

```

Hướng dẫn giải - Lây nhiễm virus [CORONA]

Hướng dẫn nhanh

- gọi i là một **điểm ngắt** nếu khoảng cách giữa x_i và x_{i+1} lớn hơn 2. Các điểm ngắt này sẽ chia hàng người thành nhiều đoạn, ta cần tìm độ dài đoạn ngắn nhất và độ dài đoạn nhiều nhất.

Hướng dẫn chi tiết

- Nhận xét: virus sẽ ngừng lây lan theo một hướng nếu khoảng cách hai người liền kề lớn hơn 2;
- Giả sử người thứ i là người đầu tiên nhiễm bệnh, virus sẽ lây lan sang bên phải cho tới khi gặp người thứ r ($r \geq i$) đầu tiên mà $x_{r+1} - x_r > 2$. Để dễ dang cài đặt vòng lặp, ta có thể coi $x_{n+1} = \infty$. Mã giả sau đây mô tả cách tìm r:

Algorithm 1: Tìm r

$r \leftarrow i$;
while $(r < n)$ *and* $(x[r+1] - x[r] \leq 2)$ **do**
  r = r + 1
end

- Tương tự, virus sẽ ngừng lây lan sang trái cho tới khi gặp người thứ l ($l \leq i$) đầu tiên sao với $x_l - x_{l-1} > 2$, coi $x_0 = -\infty$. Mã giả sau đây mô tả cách tìm l:

Algorithm 2: Tìm l

$r \leftarrow i$;
while $(l > 1)$ *and* $(x[l] - x[l-1] \leq 2)$ **do**
  l = l - 1
end

- Do giới hạn kích thước đầu vào rất nhỏ nên ở vấn đề này, bạn có thể cài trâu cũng đủ để AC. Tuy nhiên, có thể sử dụng kỹ thuật hai con trỏ để tối ưu hoá hơn, hướng tới một thuật toán có độ phức tạp $O(n)$

Algorithm 3: Two pointers

$l \leftarrow 1$
$bestCase \leftarrow n$
$worstCase \leftarrow 1$
while $l \leq N$ **do**
  $r \leftarrow findRight(l)$
  $len \leftarrow r - l + 1$
  $bestCase \leftarrow min(bestCase, len)$
  $worstCase \leftarrow max(worstCase, len)$
  $l \leftarrow r + 1$
end

- Mã triển khai dưới đây dựa trên ý tưởng của giải thuật ở trên.

Mã triển khai tham khảo C++

```cpp
#include <bits/stdc++.h>

using namespace std;

int main()
{
  #define taskname "CORONA"
```

```
8
9   if (fopen (taskname".inp", "r"))
10  {
11    freopen (taskname".inp", "r", stdin);
12    freopen (taskname".out", "w", stdout);
13  }
14
15  ios_base::sync_with_stdio (0);
16  cin.tie (0);
17  int t;
18  cin >> t;
19
20  while (t--)
21  {
22    int n;
23    cin >> n;
24    vector <int> v (n);
25
26    for (auto &i : v)
27    cin >> i;
28
29    int temp, mini = INT_MAX, maxi = INT_MIN, count = 1;
30
31    for (int i = 0; i < n - 1; ++i)
32    {
33      temp = abs (v[i + 1] - v[i]);
34
35      if (temp <= 2)
36      ++count;
37
38      else
39      {
40        mini = min (mini, count);
41        maxi = max (maxi, count);
42        count = 1;
43      }
44    }
45
46    mini = min (mini, count);
47    maxi = max (maxi, count);
48    cout << mini << ' ' << maxi << '\n';
49  }
50
51  return 0;
52 }
```

Hướng dẫn giải - Xâu mạnh [SSTRING]

Tóm tắt vấn đề

- Cho một xâu, kiểm tra xem trong xâu đó có tồn tại một đoạn con gồm ít nhất k dấu '*' liên tiếp hay không.

Subtask 1

- Có thể duyệt trâu để kiểm tra tất cả $n - k + 1$ xâu độ dài k của xâu đã cho;
- Độ phức tạp thuật toán $O(T \times n \times (n - k))$.

Subtask 2

- Ta có thể cải thiện tốc độ của thuật toán subtask 1 bằng kỹ thuật hai con trỏ để giảm độ phức tạp thuật toán cho mỗi truy vấn xuống còn $O(n)$;
- Độ phức tạp $O(T \times n)$.

Mã triển khai tham khảo C++

```cpp
#include <bits/stdc++.h>

using namespace std;
const int N = 1e6 + 1;
int main()
{
  #define taskname "SSTRING"

  if (fopen (taskname".inp", "r"))
  {
    freopen (taskname".inp", "r", stdin);
    freopen (taskname".out", "w", stdout);
  }

  ios_base::sync_with_stdio (0);
  cin.tie (0);
  int t;
  cin >> t;
  int tn = 0;

  while (t--)
  {
    int n, k;
    cin >> n >> k;
    tn += n;
    string s;
    cin >> s;
    int maxv = 0, cnt = 0;

    for (char c : s)
    {
      if (c == '*') cnt++;

      else
      {
        maxv = max (maxv, cnt);
        cnt = 0;
      }
    }

```

```
41      maxv = max (maxv, cnt);
42      string ans = maxv >= k ? "YES\n" : "NO\n";
43      cout << ans;
44    }
45
46    return 0;
47  }
```

Hướng dẫn giải - Tạo dãy đối xứng [MAKEPAL]

Tóm tắt vấn đề

- Cho một mảng A , trong một thao tác, bạn có thể chọn A_i bất kỳ rồi chia nó thành X và Y sao cho $X + Y = A_i$.
- Tìm số thao tác tối thiểu để biến A thành một dãy đối xứng.

Giải thuật

- Xét hai phần tử đầu và cuối của dãy a_1 và a_n.
  - Nếu $a_1 = a_n$ có thể bỏ qua hai phần tử này và số phần tử cần xử lý còn lại là $n - 2$;
  - Nếu $a_1 < a_n$ ta cần thực hiện ít nhất một thao tác để làm cho hai đầu của dãy bằng nhau. Do vậy, ta sẽ tách a_n thành $(a_n - a_1) + a_1$. Lúc này hai đầu của dãy đã bằng nhau và ta còn $n - 1$ phần tử cần xử lý;
  - Thực hiện tương tự với trường hợp $a_1 > a_n$.
- Trong cả ba trường hợp, mỗi lần kích thước dãy giảm ít nhất là 1. Nếu cài đặt bằng các kiểu dữ liệu như mảng tĩnh, vector, ... thì độ phức tạp có thể lên đến $O(n^2)$ vì việc xoá phần tử ở đầu có độ phức tạp là $O(n)$. Tuy nhiên, có hai cách để xử lý việc này:
  - Cách 1: đơn giản chỉ cần sử dụng cấu trúc dữ liệu `deque`. Kiểu dữ liệu này cho phép chèn hoặc xoá ở cả hai đầu trong $O(1)$;
  - Cách 2: sử dụng hai con trỏ (cũng tương đương với việc cài đặt mô phỏng một `deque` trên mảng):
    - Ban đầu, $L = 1, R = n$
    - Tại mỗi bước lặp, so sánh a_L với a_R;
    - Việc xoá phần tử có thể thực hiện bằng cách tăng L hoặc giảm R;

Mã triển khai C++

```cpp
#include <bits/stdc++.h>

using namespace std;

int main()
{
  #define taskname "boss"

  if (fopen (taskname".inp", "r"))
  {
    freopen (taskname".inp", "r", stdin);
    freopen (taskname".out", "w", stdout);
  }

  ios_base::sync_with_stdio (0);
  cin.tie (0);
  int test;
  cin >> test;

  while (test--)
  {
    int n;
    cin >> n;
    int a[n];

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

```
    {
      cin >> a[i];
    }

    int l = 0, r = n - 1, k = 0;

    while (l < r)
    {
      if (a[l] == a[r])
      {
        l++, r--;
      }

      else
      if (a[l] > a[r])
      {
        a[l] = a[l] - a[r];
        r--;
        k += 1;
      }

      else
      {
        a[r] = a[r] - a[l];
        l++;
        k += 1;
      }
    }

    cout << k << endl;
  }

  return 0;
}
```