**Cho một danh sách các phân số, hãy tìm chỉ số của phân số lớn nhất (đếm từ 0)**  
  
Giả định rằng không có các phân số bằng nhau trong tập đầu vào

**Ví dụ:**

* Với numerators = [5, 2, 5] và denominators = [6, 3, 4], thì kết quả maxFraction(numerators, denominators) = 2.   
  5/4 là phân số lớn nhất, có chỉ số là 2.

**Đầu vào/Đầu ra:**

* **[Thời gian chạy] 0.5 giây**
* **[Đầu vào] array.integer numerators**

Mảng các số nguyên thể hiện tử số của các phân số.

*Điều kiện:*  
2 ≤ numerators.length ≤ 10  
1 ≤ numerators[i] ≤ 15.

* **[Đầu vào] array.integer denominators**

Mảng có cùng độ dài vs numerators thể hiện mẫu số của các phân số

*Điều kiện:*  
denominators.length = numerators.length  
1 ≤ denominators[i] ≤ 15.

* **[Đầu ra] integer**
  + chỉ số của phân số lớn nhất (đếm từ 0).

**Lý thuyết :**

* Phân số a/b lớn hơn phân số c/d ⇔ (ad - bc)/bd > 0. Nếu dữ kiện cho a, b, c, d > 0, thì để so sánh 2 phân số, ta chỉ cần so sánh ad - bc với 0 là được
* **Ví dụ :** 2/3 > 1/2, vì 2 \* 2 - 3 \* 1 > 0
* Các bạn có thể thắc mắc là tại sao không khai báo kiểu biến số thực rồi so sánh trực tiếp luôn
  + Thực tế là, máy tính không thể tính toán chính xác được những phân số không hữu hạn. Nó sẽ làm tròn giá trị phân số đến 1 lượng chữ số thập phân nào đấy. Và sẽ có rủi ro phần lẻ khi bạn tính toán, so sánh các phân số với nhau.
  + Chỉ cần làm tròn khác đi 1 chút, thì *2 phân số bằng nhau* cũng sẽ bị tính là khác biệt. Vậy nên, nếu dữ liệu không quá lớn, mình khuyến khích các bạn so sánh theo cách đưa ra ở trên.

**Hướng dẫn bài tập.**

Code mẫu:

Ngôn ngữ C++:

int maxFraction(std::vector<int> numerators, std::vector<int> denominators)

{

int cs = 0;

for (int i = 1; i < numerators.size(); i++){

if (numerators[cs] \* denominators[i] < numerators[i] \* denominators[cs]){

cs = i;

}

}

return cs;

}