**1. Sàng Số Nguyên Tố**

**Bài toán**: Liệt kê (đếm) các số nguyên tố trong khoảng từ 1 đến N ( N ≤ 107) .

Cách làm thông thường của bài toán này là sử dụng hàm kiểm tra số nguyên tố có độ phức tạp O(√N) sau đó duyệt vòng lặp từ 1 tới N và kiểm tra. Độ phức tạp của phương pháp này sẽ là O(N√N)

Sàng số nguyên tố Eratosthenes là một thuật toán hiệu quả hơn, thuật toán được phát minh bởi nhà toán học người Hy Lạp. Độ phức tạp của thuật toán sàng số nguyên tố là O(NloglogN)

Ý tưởng của thuật toán đó là sử dụng mảng đánh dấu để có thể kiểm tra nhanh một số có phải là số nguyên tố hay không sau khi sàng.

**Thuật toán :**

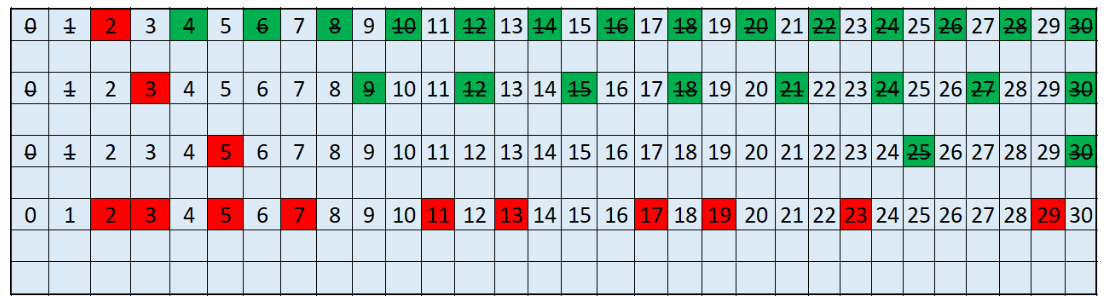
1. Coi tất cả các số nguyên từ 0 tới N đều là số nguyên tố
2. Sàng : Bước này sẽ loại bỏ các số nguyên không phải là số nguyên tố nếu nó là bội của một số nguyên tố khác

Sàng số nguyên tố cần 1 mảng đánh dấu đủ lớn, tối thiểu thì chỉ số của mảng đánh dấu này phải lớn hơn hoặc bằng N. Ví dụ nếu bạn sàng tới N = 1000 thì mảng đánh dấu ít nhất cần 1001 phần tử.

Vì thế bạn cũng sẽ thấy thuật toán này sẽ không thể sàng nếu N quá lớn vì bạn không thể khai báo mảng đánh dấu quá lớn, việc này sẽ làm tràn bộ nhớ cho phép (thông thường là 256MB trên các Online Judge).

Thuật toán này khá đơn giản, nó chỉ đơn thuần duyệt các số nguyên từ 2 tới √N và nếu số duyệt qua là số nguyên tố thì ta có thể loại bỏ các bội của số này

Ví dụ với N = 30 thì bạn cần duyệt từ 2 tới √30 = 5. Các số nguyên tố trong đoạn từ 2 tới 5 sẽ giúp bạn loại bỏ được các hợp số từ 2 tới 30.

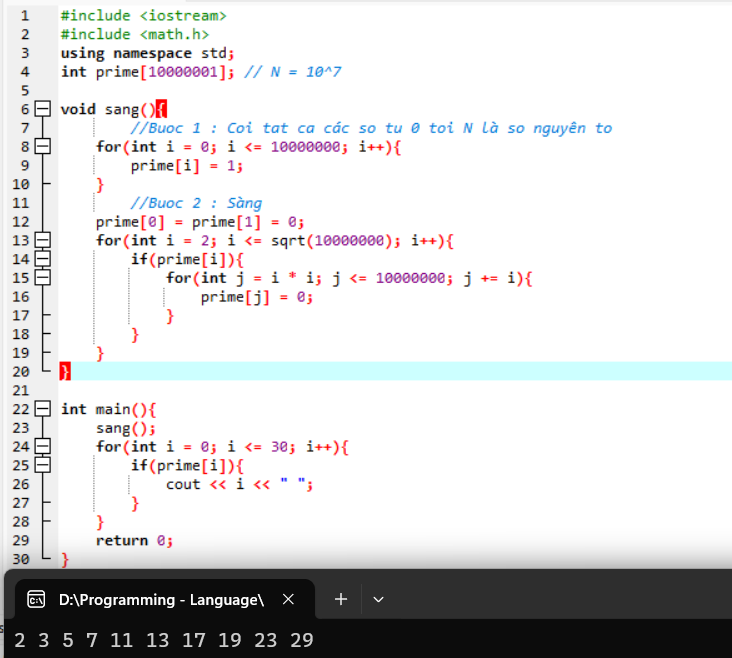


Số nguyên tố 2 thì loại bỏ các số : 4, 6, 8, 10, 12...30

Số nguyên tố 3 thì loại bỏ các số : 9, 12, 15, 18, 21,...30

Số nguyên tố 5 thì loại bỏ các số : 25, 30

**Ví dụ 1:**

****

**Giải thích mã nguồn :**

Trong mã nguồn trên có một vài điểm bạn cần chú ý :

* Mảng prime[] có thể là mảng int, bool, char tùy bạn và tùy ngôn ngữ lập trình bạn sử dụng
* Kích cỡ mảng prime[] là bao nhiêu ? Như mình đã nói ở trên thì tối thiểu nó cần là N + 1 vì ta cần sử dụng tới chỉ số N của mảng này.
* Vì sao lại duyệt từ 2 tới √N ? Ví dụ N = a \* b thì bạn sẽ thấy nếu cả a và b đều lớn hơn √N thì điều này là không thể, chắc chắn sẽ có ít nhất a hoặc b ≤ √N. Vậy thì mình biết chắc chắn sẽ có 1 ước nguyên tố của N (nếu có) sẽ nằm trong đoạn từ 2 tới √N thì ta chỉ cần tìm số này thay vì quan tâm tới số còn lại. Ví dụ với N = 28 thì nhiều bạn thắc mắc là 28 sẽ có ước nguyên tố là 7 nằm ngoài khoảng [2, √N] nhưng rõ ràng thì nó cũng có 1 ước nguyên tố là 2 nằm trong đoạn này và 2 đã loại bỏ số 28 ra khỏi tập số nguyên tố. Vậy nên nếu N không phải là số nguyên tố thì chắc chắn nó sẽ có 1 ước nguyên tố thuộc đoạn [2, √N], ta sẽ dùng số này để loại bỏ N
* Tại sao bội của i là j lại bắt đầu từ i \* i mà không phải là 2 \* i ? Khi bạn đã biết i là số nguyên tố thì bạn sẽ nghĩ ngay bội số đầu tiên (khác i) là 2 \* i, điều này là chính xác bạn cũng có thể cho j chạy từ 2 \* i tuy nhiên để ý kỹ bạn sẽ thấy những bội số của i mà nhỏ hơn i \* i đã bị loại rồi. Ví dụ khi bạn xét i = 5 thì bội đầu tiên của i sẽ là 10 nhưng số này bị loại bởi 2, bội tiếp theo là 15, 20 cũng bị loại bởi 2, 3. Vậy rõ ràng là các bội của i mà nhỏ hơn i \* i sẽ bị loại bởi các số nguyên tố nhỏ hơn i nên bạn không cần loại nữa.
* Bước nhảy của vòng lặp khi xét bội của i sẽ là i, khi đó vòng lặp chỉ duyệt qua các bội của i mà thôi.

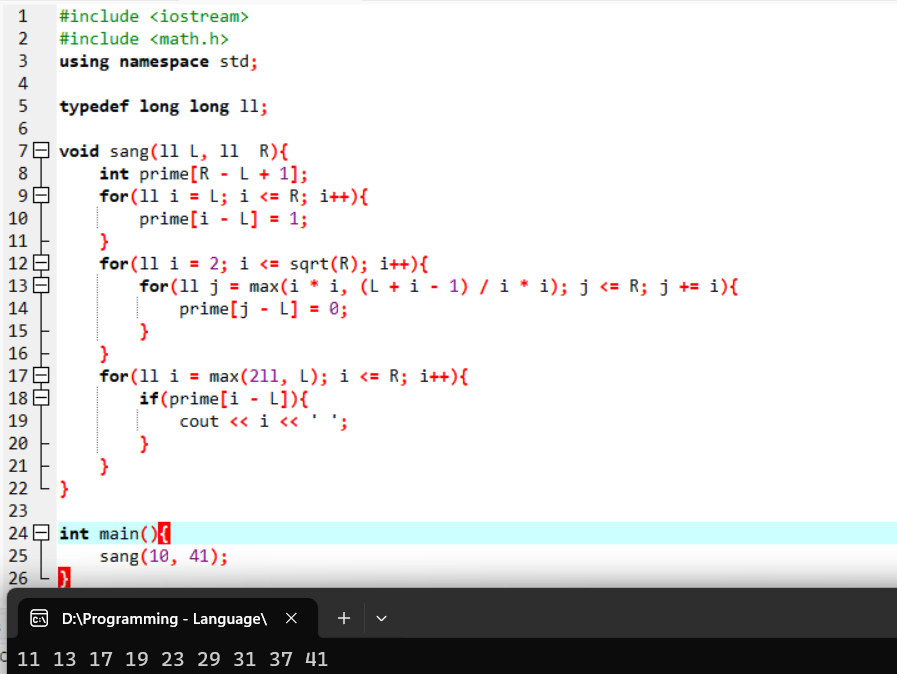
**2. Sàng Số Nguyên Tố Trên Đoạn**

Trong trường hợp bạn muốn sàng số nguyên tố trong đoạn giữa 2 số L, R thì yêu cầu bạn cũng cần xây dựng được mảng đánh dấu có cỡ R - L + 1.

Khi đó để đánh dấu giá trị các số từ L tới R thì bạn đánh dấu các chỉ số từ 0 tới R - L

Ví dụ L = 10, R = 41 thì bạn sẽ đánh dấu các chỉ số tương ứng là 0 tới 31

Thuật toán sàng số nguyên tố trong đoạn tương tự như thuật toán sàng số nguyên tố từ 1 tới N

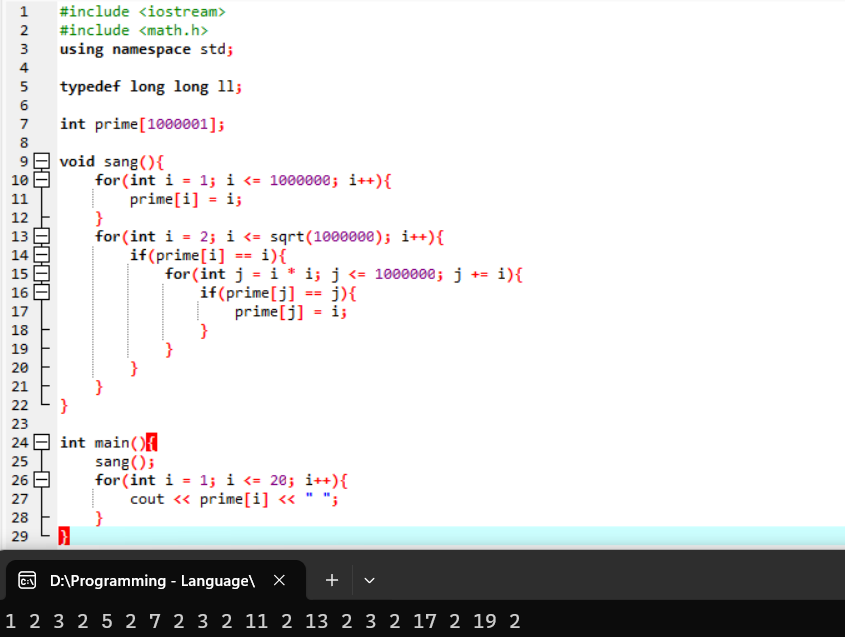


**Giải thích mã nguồn :**

* Vì sao không dùng mảng cỡ R + 1 ? Ở đây L và R có thể rất lớn (lên tới long long) và bạn chỉ sàng được khi L và R của bạn không cách nhau quá xa nên không thể dùng mảng cỡ R mà cần dùng mảng cỡ R - L + 1. Ví dụ L = 1000000000 và R = 1000800000 thì bạn chỉ cần dùng mảng cỡ R - L + 1 = 800001 thay vì cỡ 1000800000 ( điều này là không thể)
* Vì sao lại duyệt bội đầu tiên của i là max(i \* i, (L + i - 1) / i \* i), giá trị (L + i - 1) / i \* i là gì ? (L + i - 1) / i \* i là bội nhỏ nhất của L ≥ i \* i, công thức này dựa trên phép chia nguyên trong lập trình, bạn có thể nháp ra để hiểu rõ hơn. Khi duyệt bội của i ta không nên duyệt từ i \* i ví dụ i = 2 và L = 1000000000 thì thay vì phải duyệt từ 4 bạn nên duyệt từ 1000000000, thông thường bạn sẽ duyệt từ i \* i nhưng trong trường hợp này có thể tìm max của i \* i với số nhỏ nhất ≥ L mà chia hết cho i để giảm số vòng lặp không cần thiết
* Vì sao khi duyệt để in kết quả không duyệt từ L mà duyệt từ max(2, L) ? Vì khi duyệt bội của các số nguyên tố thì bạn duyệt từ i = 2 nên không thể xét được các số 0, 1 không phải là số nguyên tố, ví dụ L = 0, 1 thì bạn sẽ in ra 0, 1 là số nguyên tố. Để tránh việc này thì bạn sẽ duyệt từ max(2, L)

**3. Một Số Ứng Dụng Của Sàng Nguyên Tố**

**Bài toán 1 :**Tìm ước nguyên tố nhỏ nhất của các số từ 1 tới N, coi ước nguyên tố nhỏ nhất của 1 là chính nó



**Bài toán 2** : Phân tích thừa số nguyên tố kết hợp sàng số nguyên tố, phương pháp này hiệu quả khi bạn cần phân tích nhiều số nguyên tố trong đoạn cho trước. Còn nếu chỉ phần tích 1 số thì bạn làm theo phương pháp thông thường.

