*GCPD* (Greatest Common Prime Divisor) được định nghĩa là số nguyên tố lớn nhất là ước của các số nguyên dương cho trước. Nhiệm vụ của bạn là tìm *GCPD* của hai số nguyên a và b.

**Ví dụ**

* Với a = 12 và b = 18, đầu ra là greatestCommonPrimeDivisor(a, b) = 3;
* Với a = 12 và b = 13, đầu ra là greatestCommonPrimeDivisor(a, b) = -1.

**Đầu vào/Đầu ra**

* **[giới hạn thời gian chạy] 0.5 giây**
* **[đầu vào] integer a**

*Điều kiện tiền đề:*  
2 ≤ a ≤ 150.

* **[đầu vào] integer b**

*Điều kiện tiền đề:*  
2 ≤ b ≤ 150.

* **[đầu ra] integer**
  + số*GCPD* của a và b, hoặc -1 nếu nó không tồn tại.

**Lý thuyết**

***Sàng nguyên tố Eratosthenes***

* Tư tưởng của phương pháp trên là ta sẽ tìm mọi số nguyên tố có giá trị bé hơn hoặc bằng n. Từ đó có thể kết luật được số n có phải là một số nguyên tố hay không ?
* Thuật toán được miêu tả như sau:   
  1. Lập một danh sách các số liên tiếp từ 2 đến n
  2. Bước đầu tiên ta đặt số a = 2
  3. Lần lượt đánh dấu các số a\*a, a\*(a+1), a\*(a+2), ... có trong danh sách mình đã tạo trước. Nếu như a\*a > n thì ta kết thúc thuật toán.
  4. Tìm số đầu tiên lớn hơn a chưa được đánh dấu trong danh sách. Nếu không tìm thấy thì kết thúc thuật toán, ngược lại thì gán a bằng số đó và lặp lại *bước 3*
* Ví dụ minh họa:
  1. Một danh sách các số từ 2 đến 20:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 |

* 1. ta gán a = 2 và đánh dấu các số 2\*2, 2\*3, 2\*4, ..., 2\*10. Ta được bảng:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 |

* 1. Tiếp tục tìm số lớn hơn 2 mà chưa được đánh dấu trong danh sách và gán a bằng số đó ⇒ a = 3. Tiếp tục đánh dấu các số 3\*3, 3\*4, ..., 3\*6. Ta được bảng:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 |

* 1. Tiếp tục tìm số lớn hơn 3 mà chưa được đánh dấu trong danh sách và gán a bằng số đó ⇒ a = 5. Tiếp tục đánh dấu các số chia hết cho 5 mà lớn hơn 5\*5 = 25. Không tồn tại số đó trong danh sách. Ta dừng lại quá trình làm của mình ở đây. Cuối cùng bảng ta thu được sẽ là:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 |

* 1. Nhận thấy các số trong danh sách không được đánh dấu là số nguyên tố
  2. bool mark[1002];
  3. bool isPrime(int n)
  4. {
  5. if (n <= 1) return false;
  6. int a = 2;
  7. while (true) {
  8. if (a\*a > n) break;
  9. int p = a\*a;
  10. while (p <= n) {
  11. mark[p] = true;
  12. p += a;
  13. }
  14. ++a;
  15. while (a <= n && mark[a]) ++a;
  16. if (a > n) break;
  17. }
  18. return (mark[n]) ? false : true;
  19. }
* *Độ phức tạp: O( nlog(n) )*

**Hướng dẫn bài tập.**

Code mẫu:

Ngôn ngữ C++:

bool arr[1000001];

void snt(int n){

for (int i = 2; i <= n; i++)

arr[i] = 1;

arr[0] = arr[1] = 0;

for (int i = 2; i <= sqrt(n); i++)

if (arr[i])

for (int j = 2 \* i; j <= n; j += i)

arr[j] = 0;

}

int greatestCommonPrimeDivisor(int a, int b)

{

snt(min(a, b));

int d = 0;

for (int i = min(a, b); i >= 2; i--)

if (arr[i] && a % i == 0 && b % i == 0)

return i;

return -1;

}