So sánh hai thuật toán và cách cài đặt code

Sử dụng vòng lặp để kiểm tra từ 2 đến căn bậc hai - sqrt() của số xem có tồn tại một ước nào đó không.

```
#include <stdio.h>
#include <math.h>

// Determine the primality of a given number (use sqrt() function in math.h)
bool isPrime(int num)
{
    if (num <= 1)
        return false;

    for (int i = 2; i <= sqrt(num); i++)
        if (num % i == 0)
            return false;
    return true;
}

int main()
{
    int number;
    printf("Nhap mot so bat ky: ");
    scanf("%d", &number);

    if (isPrime(number))
        printf("%d la so nguyen to!\n", number);
    else
        printf("%d khong phai la so nguyen to!\n", number);

    return 0;
}
```

Nhận xét: *Đoạn code này sử dụng một vòng lặp for và tính căn bậc hai của số để kiểm tra số nguyên tố. Điều này làm cho mã trở nên dễ đọc và hiểu, vì nó phản ánh một cách rõ ràng cách kiểm tra số nguyên tố bằng cách xem xét tất cả các ước tiềm năng của số. Tuy nhiên, nó sử dụng một số thư viện (math.h) để tính căn bậc hai, điều này có thể được xem xét là không cần thiết trong trường hợp này.*

Sử dụng một cách đơn giản hơn để kiểm tra số nguyên tố bằng cách kiểm tra từ 2 đến $\dfrac{n}{2}$ xem có tồn tại ước nào đó không.

```
#include <stdio.h>

// Determine the primality of a given number (non-use sqrt() function in math.h)
bool isPrime(int num) {
    if (num <= 1)
```

```
        return false;

    for (int i = 2; i <= num / 2; i++)
        if (num % i == 0)
            return false;

    return true;
}

int main() {
    int number;

    printf("Nhap mot so bat ky: ");
    scanf("%d", &number);

    if (isPrime(number))
        printf("%d la so nguyen to!\n", number);
    else
        printf("%d khong phai la so nguyen to!\n", number);

    return 0;
}
```

Nhận xét: *Đoạn code này sử dụng một vòng lặp for đơn giản để kiểm tra số nguyên tố bằng cách xem xét tất cả các số từ 2 đến $\dfrac{n}{2}$. Điều này làm cho mã trở nên dễ đọc với một vòng lặp đơn giản hơn, nhưng nó có thể không hiệu quả bằng cách sử dụng căn bậc hai. Mã này dễ hiểu hơn nhưng có thể không tối ưu về hiệu suất.*