**BÁO CÁO TECHNICAL REVIEW – PAIR PROGRAMING**

**Chủ đề: Tối ưu thuật toán tìm số nguyên tố trong Java**

**Nhóm thực hiện**

* **Sinh viên 1 (Driver):** 29211153547 **-** Đỗ Văn Hiếu
* **Sinh viên 2(Navigator):** 29211154867 - Nguyễn Trọng Khôi
* **Thời gian thực hiện:** 22/2/2025

1. **Mô tả giải pháp ban đầu**

Giải pháp ban đầu sử dụng một hàm isprime(n) để kiểm tra số nguyên tố và một hàm printPrime(n) để in tất cả các số nguyên tố nhỏ hơn n. Cách kiểm tra số nguyên tố thực hiện bằng vòng lặp từ 2 đến sqrt(n), kiểm tra từng số một.

* **Quy trình hoạt động**:

1. Nhập số n từ bàn phím.
2. Duyệt từng số từ 2 đến n-1 và kiểm tra xem số đó có phải số nguyên tố không.
3. Nếu là số nguyên tố, in ra màn hình và đếm tổng số lượng.
4. Đo thời gian thực thi để đánh giá hiệu suất thuật toán.
5. **Lỗi và điểm chưa tối ưu được phát hiện**
6. Hàm kiểm tra số nguyên tố chưa hiệu quả:

* Kiểm tra từng số riêng lẻ làm tốn thời gian.
* Không tối ưu hóa việc bỏ qua số chẵn (trừ 2).

1. Vòng lặp kiểm tra tất cả các số:

* Lặp qua từng số từ 2 đến n-1, tốn nhiều thời gian.
* Không lưu trữ kết quả kiểm tra số nguyên tố nên bị lặp lại nhiều lần.

1. Độ phức tạp cao (O(n√n))

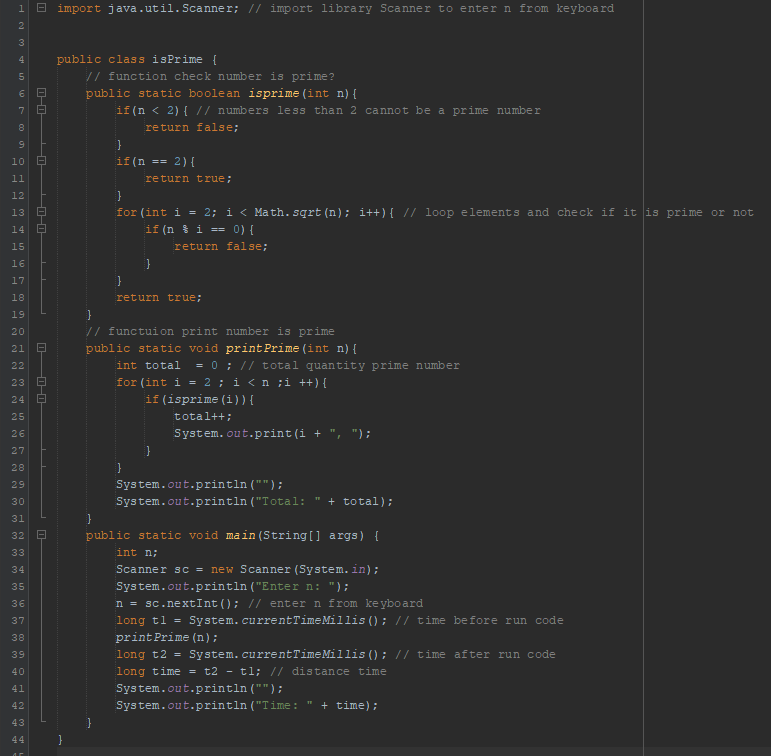
* Khi n lớn (vd: 10^6), thuật toán mất nhiều thời gian.

1. **Giải pháp tối ưu và kết quả**

Giải pháp 1: **Sàng Eratosthenes**

* Cách thực hiện:
* Dùng một mảng isPrime[] để đánh dấu số nguyên tố.
* Bắt đầu từ 2, loại bỏ tất cả bội số của nó.
* Tiếp tục với các số nguyên tố tiếp theo, đến khi kiểm tra hết.
* Cuối cùng, duyệt qua mảng để lấy danh sách số nguyên tố.
* Hiệu quả:
* Độ phức tạp giảm xuống O(n log log n).
* Tăng tốc đáng kể so với kiểm tra từng số riêng lẻ.
* Lưu trữ kết quả trong mảng giúp truy vấn nhanh hơn.
* Kết quả:
* Trước tối ưu: Khi n = 10^6, thời gian chạy mất vài giây.
* Sau tối ưu: Khi n = 10^6, thời gian giảm xuống dưới 100ms.

1. **Kết luận**

* Giải pháp ban đầu sử dụng vòng lặp để kiểm tra từng số nguyên tố riêng lẻ, tuy hoạt động đúng nhưng chậm.
* Sau khi cải tiến bằng Sàng Eratosthenes, thuật toán chạy nhanh hơn đáng kể, phù hợp với các giá trị n lớn.
* Bài học rút ra:
* Luôn kiểm tra lỗi logic trước khi tối ưu.
* Tìm thuật toán có độ phức tạp thấp hơn thay vì tối ưu từng dòng code.
* Sử dụng bộ nhớ hiệu quả giúp giảm thời gian xử lý đáng kể.
* ****

Navigator

Driver