

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN  
KHOA KHOA HỌC MÁY TÍNH

---

**PHƯƠNG PHÁP THIẾT KẾ THUẬT TOÁN PHÂN TÁN  
BÀI TẬP NHÓM 4**

---

*Sinh viên:*

NGUYỄN HUY PHƯỚC - 23521234  
PHAN NHẬT TÂN - 23521405

*Giảng viên:*

Nguyễn Thanh Sơn



**UIT**

TP. HỒ CHÍ MINH, năm 2024

# Mục lục

<b>1</b>	<b>Bài 1: Kiểm tra nguyên tố</b>	<b>2</b>
1.1	Đề bài . . . . .	2
1.2	Ý tưởng: . . . . .	2
1.3	Cài đặt . . . . .	3
<b>2</b>	<b>Bài 2: Nhân ma trận</b>	<b>4</b>
2.1	Đề bài: . . . . .	4
2.2	Ý tưởng: . . . . .	4
2.3	Cài đặt: . . . . .	5

# 1 Bài 1: Kiểm tra nguyên tố

## 1.1 Đề bài

Cho số nguyên dương  $X$  hãy kiểm tra  $X$  phải số nguyên tố hay không.

## 1.2 Ý tưởng:

- Mục tiêu: Xác định số nguyên  $X$  có phải là số nguyên tố hay không.
- Ý tưởng:
  1. Nếu  $X \leq 1$ , trả về **False**.
  2. Lặp qua tất cả các số nguyên  $i$  từ 2 đến  $\lfloor \sqrt{X} \rfloor$ .
  3. Nếu tồn tại  $i$  sao cho  $X \bmod i = 0$ , thì  $X$  không phải là số nguyên tố.
  4. Nếu không tìm được ước số nào, trả về **True**.
- Độ phức tạp:  $O(\sqrt{X})$ .

Song song

- Mục tiêu: Tối ưu hóa quá trình kiểm tra số nguyên tố bằng cách sử dụng đa luồng hoặc đa tiến trình.
- Ý tưởng:
  1. Nếu  $X \leq 1$ , trả về **False**.
  2. Chia khoảng kiểm tra  $[2, \lfloor \sqrt{X} \rfloor]$  thành  $T$  đoạn:  
$$\text{Mỗi đoạn } [\text{start}_i, \text{end}_i) \text{ với } i = 1, 2, \dots, T.$$
  3. Khởi tạo  $T$  tiến trình song song. Mỗi tiến trình kiểm tra liệu  $X$  có chia hết cho bất kỳ số nào trong đoạn của nó.
  4. Nếu bất kỳ tiến trình nào tìm thấy  $i$  sao cho  $X \bmod i = 0$ , dừng tất cả các tiến trình và trả về **False**.
  5. Nếu không có tiến trình nào tìm thấy ước số, trả về **True**.
- Độ phức tạp:
  - Lý thuyết:  $O\left(\frac{\sqrt{X}}{T}\right)$  với  $T$  là số luồng.
  - Thực tế phụ thuộc vào số lỗi CPU và overhead của việc quản lý luồng.

## 1.3 Cài đặt

```
import math
import time
from multiprocessing import Pool, cpu_count

def is_prime_sequential(x):
    if x <= 1:
        return False
    for i in range(2, int(math.sqrt(x)) + 1):
        if x % i == 0:
            return False
    return True

def check_range(args):
    x, start, end = args
    for i in range(start, end):
        if x % i == 0:
            return False
    return True

def is_prime_parallel(x):
    if x <= 1:
        return False

    # Chia công việc kiểm tra
    num_cores = cpu_count() # Số lượng CPU
    sqrt_x = int(math.sqrt(x)) + 1
    step = (sqrt_x // num_cores) + 1
    ranges = [(x, i, min(i + step, sqrt_x)) for i in range(2, sqrt_x, step)]

    with Pool(num_cores) as pool:
        results = pool.map(check_range, ranges)

    return all(results)

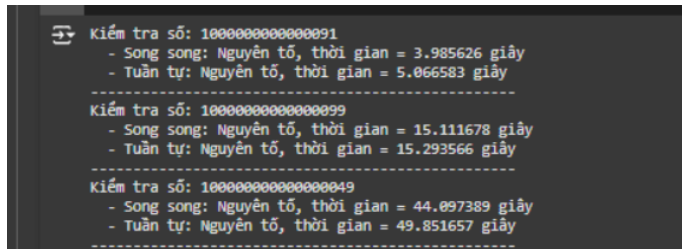
def compare_execution_time(test_cases):
    for x in test_cases:
        print(f"Kiểm tra số: {x}")

        start = time.time()
        result_seq = is_prime_sequential(x)
        time_seq = time.time() - start
        print(f" - Tuan tu: {'Song song' if result_seq else 'Khong nguyen to'}, thời gian = {time_seq:.6f} giây")

        start = time.time()
        result_par = is_prime_parallel(x)
        time_par = time.time() - start
        print(f" - Song song: {'Nguyen to' if result_par else 'Khong nguyen to'}, thời gian = {time_par:.6f} giây")
        print("-" * 50)
```

```
test_cases = [
    1000000000000000091, # Nguyen to
    1000000000000000099, # Khong nguyen to
    1000000000000000049 # Nguyen to
]

compare_execution_time(test_cases)
```



```
Kiểm tra số: 1000000000000000091
- Song song: Nguyên tố, thời gian = 3.985626 giây
- Tuần tự: Nguyên tố, thời gian = 5.066583 giây
-----
Kiểm tra số: 1000000000000000099
- Song song: Nguyên tố, thời gian = 15.111678 giây
- Tuần tự: Nguyên tố, thời gian = 15.293566 giây
-----
Kiểm tra số: 1000000000000000049
- Song song: Nguyên tố, thời gian = 44.097389 giây
- Tuần tự: Nguyên tố, thời gian = 49.851657 giây
-----
```

Đây là kết quả thực hiện

## 2 Bài 2: Nhân ma trận

### 2.1 Đề bài:

Bài toán yêu cầu thực hiện nhân hai ma trận  $A$  kích thước  $m \times n$  và  $B$  kích thước  $n \times p$  để tạo ra ma trận kết quả  $C$  có kích thước  $m \times p$ . Công thức nhân ma trận là:

$$C[i][j] = \sum_{k=1}^n A[i][k] \cdot B[k][j]$$

Trong đó,  $A[i][k]$  là phần tử của ma trận  $A$  tại dòng  $i$  và cột  $k$ , và  $B[k][j]$  là phần tử của ma trận  $B$  tại dòng  $k$  và cột  $j$ .

### 2.2 Ý tưởng:

Tuần tự:

- Duyệt qua tất cả các phần tử của ma trận kết quả  $C$ , tức là các cặp chỉ số  $(i, j)$ .
- Với mỗi cặp  $(i, j)$ , tính tổng tích các phần tử  $A[i][k] \cdot B[k][j]$  từ  $k = 1$  đến  $n$ .
- Độ phức tạp thời gian của thuật toán tuần tự là  $O(m \cdot n \cdot p)$ .

Song Song:

- Chia công việc tính toán các phần tử  $C[i][j]$  giữa nhiều tiến trình song song:
  1. Chia theo hàng: Mỗi tiến trình tính toán một số hàng của ma trận  $C$ .
  2. Chia theo cột: Mỗi tiến trình tính toán một số cột của ma trận  $C$ .
- Sử dụng các thư viện đa tiến trình như `multiprocessing` trong Python để thực hiện song song các phép toán.
- Độ phức tạp lý thuyết:  $O\left(\frac{m \cdot n \cdot p}{T}\right)$ , với  $T$  là số tiến trình được sử dụng.

## 2.3 Cài đặt:

```
import random
import time
import multiprocessing
import numpy as np

# Hàm nhân ma trận tuần tự
def matrix_multiply_sequential(A, B):
    m, n = len(A), len(A[0])
    n2, p = len(B), len(B[0])

    if n != n2:
        raise ValueError("Số cột của ma trận A phải bằng số dòng của ma trận B")

    # Khởi tạo ma trận kết quả
    C = [[0 for _ in range(p)] for _ in range(m)]

    for i in range(m):
        for j in range(p):
            for k in range(n):
                C[i][j] += A[i][k] * B[k][j]

    return C

# Hàm nhân ma trận song song
def multiply_chunk(A_chunk, B, result, row_start, row_end, p):
    m = len(A_chunk)
    n = len(A_chunk[0])

    for i in range(m):
        for j in range(p):
            for k in range(n):
                result[row_start + i][j] += A_chunk[i][k] * B[k][j]

# Hàm chia công việc cho các tiến trình
def matrix_multiply_parallel(A, B):
    m, n = len(A), len(A[0])
    n2, p = len(B), len(B[0])

    if n != n2:
        raise ValueError("Số cột của ma trận A phải bằng số dòng của ma trận B")

    # Khởi tạo ma trận kết quả
    C = [[0 for _ in range(p)] for _ in range(m)]

    # Chia công việc cho các tiến trình
    num_processes = multiprocessing.cpu_count()
    chunk_size = m // num_processes
    processes = []

    for i in range(num_processes):
        row_start = i * chunk_size
        row_end = (i + 1) * chunk_size if i != num_processes - 1 else m
        A_chunk = A[row_start:row_end]
        p = multiprocessing.Process(target=multiply_chunk, args=(A_chunk, B, C, row_start, row_end))
        processes.append(p)
```

```

        p.start()

    for p in processes:
        p.join()

    return C

# Hàm tạo ma trận ngẫu nhiên
def generate_random_matrix(rows, cols):
    return [[random.randint(0, 10) for _ in range(cols)] for _ in range(rows)]

# Chạy thử nghiệm
def run_test():
    size = 400
    A = generate_random_matrix(size, size)
    B = generate_random_matrix(size, size)

    # Kiểm tra nhân ma trận tuần tự
    start_time = time.time()
    C_sequential = matrix_multiply_sequential(A, B)
    sequential_time = time.time() - start_time

    # Kiểm tra nhân ma trận song song
    start_time = time.time()
    C_parallel = matrix_multiply_parallel(A, B)
    parallel_time = time.time() - start_time

    # In ra thời gian
    print(f"Thời gian thực hiện (tuần tự): {sequential_time:.4f} giây")
    print(f"Thời gian thực hiện (song song): {parallel_time:.4f} giây")

if __name__ == "__main__":
    run_test()

```

```

TypeError: Process object cannot be interpreted
Thời gian thực hiện (tuần tự): 16.2922 giây
Thời gian thực hiện (song song): 8.2623 giây

```

Đây là kết quả thực hiện