ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Khoa Khoa học máy tính

Môn học Phân Tích Và Thiết Kế Thuật Toán

Bài tập Cài đặt thuật toán song song

Cao Lê Công Thành
MSSV: 23521437
Đặng Quang Vinh
MSSV: 23521786

Mục lục

1 Bài tập 1

Xây dựng thuật toán kiểm tra số nguyên tố song song.

Input:

Một số nguyên X.

Output:

Xác định X có phải là số nguyên tố hay không.

Yêu cầu:

- Kiểm tra kết quả với khi thực hiện tính toán tuần tự.
- So sánh thời gian thực hiện giữa phương pháp song song và tuần tự trên nhiều test case có cả số nhỏ và lớn.
- Chạy thử và show thời gian thực hiện cả song song và tuần tự trên các test case sau:
  - $X = 1000000000000091$
  - $X = 10000000000000099$
  - $X = 100000000000000049$

1.1 Giải pháp:

Thuật toán kiểm tra số nguyên tố tuần tự:

Kiểm tra một số X có phải là số nguyên tố hay không bằng cách chia X cho các số nguyên từ 2 đến $\sqrt{X}$. Nếu X chia hết cho bất kỳ số nào trong khoảng này, nó không phải là số nguyên tố.

Thuật toán kiểm tra số nguyên tố song song:

Phương pháp song song chia dải số từ 2 đến $\sqrt{X}$ thành các đoạn nhỏ, mỗi đoạn được xử lý trên một lõi CPU khác nhau. Kết quả của từng lõi sẽ được tổng hợp lại để quyết định xem X có phải là số nguyên tố hay không.

1.2 Phần Code:

```python
import psutil
import math
import multiprocessing
import time

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

def check_prime_range(n, start, end):
    for i in range(start, end):
        if n % i == 0:
            return False
    return True

def is_prime_parallel(n,
   num_processes=psutil.cpu_count(logical=False)):
    if n < 2:
        return False
    sqrt_n = int(math.sqrt(n)) + 1
    step = (sqrt_n // num_processes) + 1

    with multiprocessing.Pool(processes=num_processes)
       as pool:
        tasks = [(n, i, min(i + step, sqrt_n)) for i in
           range(2, sqrt_n, step)]
        results = pool.starmap(check_prime_range, tasks)
        return all(results)

if __name__ == '__main__':
    test_cases = [1000000000000091, 10000000000000099,
       100000000000000049]

    for X in test_cases:
        print(f"Kiem tra so: {X}")

        # Tuan Tu
```

```
38          start = time.time()
39          result_seq = is_prime_sequential(X)
40          end = time.time()
41          print(f"Tuan Tu: {X} {'la so nguyen to' if
                result_seq else 'khong phai la so nguyen
                to'}")
42          print(f" Thoi gian thuc hien tuan tu:  {end -
                start:.5f} giay")
43
44          #  song song
45          start = time.time()
46          result_par = is_prime_parallel(X)
47          end = time.time()
48          print(f" song song: {X} {'la so nguyen to' if
                result_par else 'khong phai la so nguyen
                to'}")
49          print(f"Thoi gian thuc hien song song: {end -
                start:.5f} giay")
50          print("="*50)
```

1.3 Kết quả:

```
Kiểm tra số: 1000000000000091
Kết quả tuần tự: 1000000000000091 là số nguyên tố
Thời gian thực hiện (tuần tự): 1.63029 giây
Kết quả song song: 1000000000000091 là số nguyên tố
Thời gian thực hiện (song song): 0.67187 giây
==================================================
Kiểm tra số: 10000000000000099
Kết quả tuần tự: 10000000000000099 là số nguyên tố
Thời gian thực hiện (tuần tự): 5.10629 giây
Kết quả song song: 10000000000000099 là số nguyên tố
Thời gian thực hiện (song song): 1.91233 giây
==================================================
Kiểm tra số: 100000000000000049
Kết quả tuần tự: 100000000000000049 là số nguyên tố
Thời gian thực hiện (tuần tự): 19.77827 giây
Kết quả song song: 100000000000000049 là số nguyên tố
Thời gian thực hiện (song song): 5.76709 giây
==================================================
Press any key to continue . . .
```

```
Kiểm tra số: 11
Kết quả tuần tự: 11 là số nguyên tố
Thời gian thực hiện (tuần tự): 0.00000 giây
Kết quả song song: 11 là số nguyên tố
Thời gian thực hiện (song song): 0.14929 giây
================================================
Kiểm tra số: 15
Kết quả tuần tự: 15 không phải số nguyên tố
Thời gian thực hiện (tuần tự): 0.00000 giây
Kết quả song song: 15 không phải số nguyên tố
Thời gian thực hiện (song song): 0.11309 giây
================================================
Kiểm tra số: 111
Kết quả tuần tự: 111 không phải số nguyên tố
Thời gian thực hiện (tuần tự): 0.00000 giây
Kết quả song song: 111 không phải số nguyên tố
Thời gian thực hiện (song song): 0.11444 giây
================================================
Press any key to continue . . .
```

1.4 Nhận xét kết quả:

Ta thấy thuật toán song song chạy nhanh hơn thuật toán tuần tự đối với các số lớn. Tuy nhiên đối với những số nhỏ thì thuật toán tuần tự lại hiệu quả hơn.

2 Bài tập 2

2.1 Đề bài

Xây dựng thuật toán nhân ma trận song song.

- Input: 2 ma trận A, B.
- Output: Kết quả nhân 2 ma trận A, B.

Yêu cầu:

- Kiểm tra kết quả với khi thực hiện tính toán tuần tự.
- So sánh thời gian thực hiện giữa song song và tuần tự trên nhiều test case có cả khi ma trận kích thước nhỏ và lớn.
- Sinh ngẫu nhiên ma trận A, B kích thước 400×400. Show ra thời gian thực hiện cả song song và tuần tự trên các test case trên.

- Không sử dụng numpy.dot() hay các hàm nhân ma trận code sẵn.

2.2 Hướng giải quyết

- **Nhân ma trận tuần tự:** Triển khai nhân hai ma trận bằng cách duyệt từng hàng của ma trận A và từng cột của ma trận B để tính từng phần tử của ma trận kết quả C.
- **Nhân ma trận song song:** Sử dụng đa luồng (multi-threading) hoặc đa tiến trình (multi-processing) để tính toán các phần tử của ma trận C song song. Chia công việc theo hàng, mỗi luồng hoặc tiến trình xử lý một nhóm hàng của ma trận A.

2.3 Code và kiểm thử

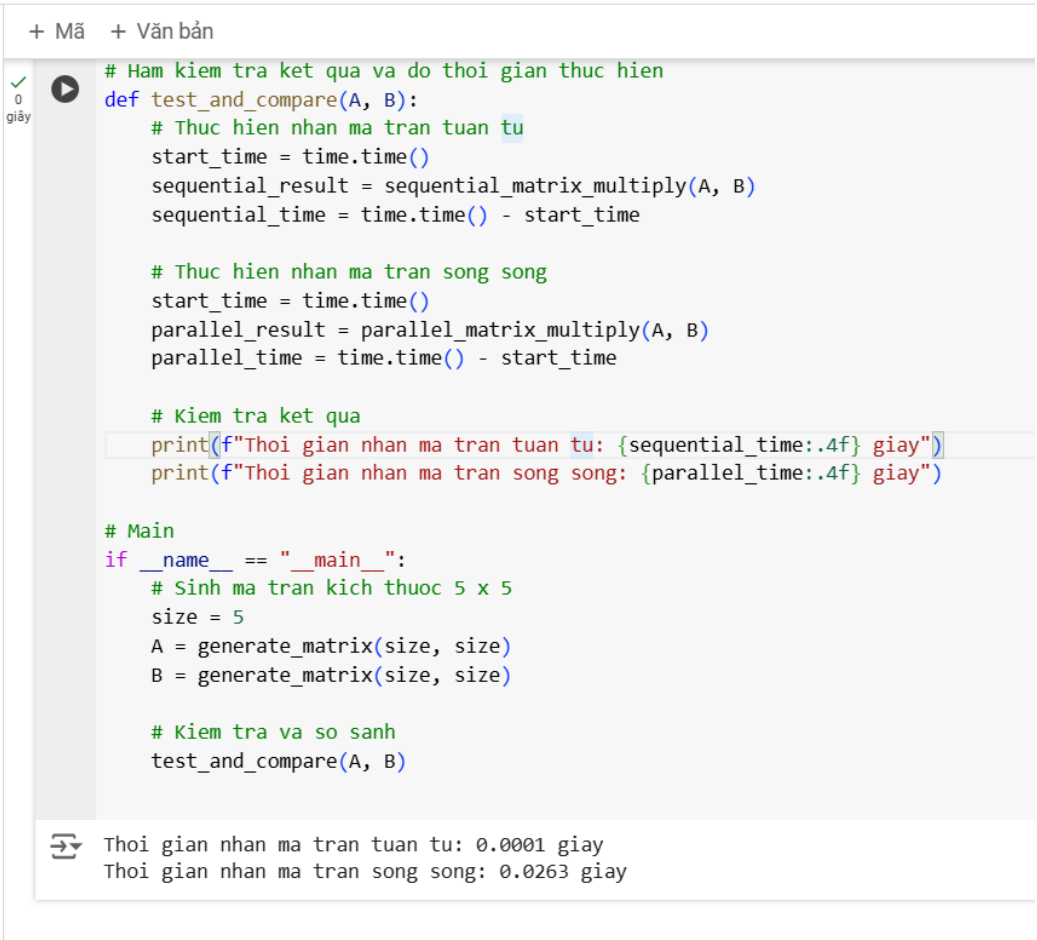
Code:

```
import random
import time
from multiprocessing import Pool

# Ham tao ma tran ngau nhien
def generate_matrix(rows, cols, value_range=(1, 10)):
    return [[random.randint(*value_range) for _ in
       range(cols)] for _ in range(rows)]

# Ham nhan ma tran tuan tu
def sequential_matrix_multiply(A, B):
    rows_A, cols_A = len(A), len(A[0])
    rows_B, cols_B = len(B), len(B[0])

    # Kiem tra dieu kien kich thuoc hop le
    if cols_A != rows_B:
        raise ValueError("So cot cua ma tran A phai
           bang so hang cua ma tran B")

    # Khoi tao ma tran ket qua
    C = [[0 for _ in range(cols_B)] for _ in
       range(rows_A)]

    # Nhan ma tran tuan tu
    for i in range(rows_A):
        for j in range(cols_B):
```

```
            for k in range(cols_A):
                C[i][j] += A[i][k] * B[k][j]
    return C

# Ham tinh mot hang cua ma tran ket qua
def compute_row(row_index, A, B, cols_B):
    return [
        sum(A[row_index][k] * B[k][j] for k in
            range(len(A[0]))) for j in range(cols_B)
    ]

# Ham nhan ma tran song song
def parallel_matrix_multiply(A, B):
    rows_A, cols_A = len(A), len(A[0])
    rows_B, cols_B = len(B), len(B[0])

    if cols_A != rows_B:
        raise ValueError("So cot cua ma tran A phai
            bang so hang cua ma tran B")

    # Su dung Pool de tinh song song
    with Pool() as pool:
        result = pool.starmap(
            compute_row,
            [(i, A, B, cols_B) for i in range(rows_A)]
        )
    return result

# Ham kiem tra ket qua va do thoi gian thuc hien
def test_and_compare(A, B):
    # Thuc hien nhan ma tran tuan tu
    start_time = time.time()
    sequential_result = sequential_matrix_multiply(A, B)
    sequential_time = time.time() - start_time

    # Thuc hien nhan ma tran song song
    start_time = time.time()
    parallel_result = parallel_matrix_multiply(A, B)
    parallel_time = time.time() - start_time

    # Kiem tra ket qua
    print(f"Thoi gian nhan ma tran tuan tu:
        {sequential_time:.4f} giay")
```

```
    print(f"Thoi gian nhan ma tran song song:
        {parallel_time:.4f} giay")

# Main
if __name__ == "__main__":
    # Sinh ma tran kich thuoc 400 x 400
    size = 400
    A = generate_matrix(size, size)
    B = generate_matrix(size, size)

    # Kiem tra va so sanh
    test_and_compare(A, B)
```

Kiểm thử 2 ma trận 5 x 5:



```
# Ham kiem tra ket qua va do thoi gian thuc hien
def test_and_compare(A, B):
    # Thuc hien nhan ma tran tuan tu
    start_time = time.time()
    sequential_result = sequential_matrix_multiply(A, B)
    sequential_time = time.time() - start_time

    # Thuc hien nhan ma tran song song
    start_time = time.time()
    parallel_result = parallel_matrix_multiply(A, B)
    parallel_time = time.time() - start_time

    # Kiem tra ket qua
    print(f"Thoi gian nhan ma tran tuan tu: {sequential_time:.4f} giay")
    print(f"Thoi gian nhan ma tran song song: {parallel_time:.4f} giay")

# Main
if __name__ == "__main__":
    # Sinh ma tran kich thuoc 5 x 5
    size = 5
    A = generate_matrix(size, size)
    B = generate_matrix(size, size)

    # Kiem tra va so sanh
    test_and_compare(A, B)
```

```
Thoi gian nhan ma tran tuan tu: 0.0001 giay
Thoi gian nhan ma tran song song: 0.0263 giay
```

Kiểm thử 2 ma trận 20 x 20:

+ Mã + Văn bản

```
    return result

# Ham kiem tra ket qua va do thoi gian thuc hien
def test_and_compare(A, B):
    # Thuc hien nhan ma tran tuan tu
    start_time = time.time()
    sequential_result = sequential_matrix_multiply(A, B)
    sequential_time = time.time() - start_time

    # Thuc hien nhan ma tran song song
    start_time = time.time()
    parallel_result = parallel_matrix_multiply(A, B)
    parallel_time = time.time() - start_time

    # Kiem tra ket qua
    print(f"Thoi gian nhan ma tran tuan tu: {sequential_time:.4f} giay")
    print(f"Thoi gian nhan ma tran song song: {parallel_time:.4f} giay")

# Main
if __name__ == "__main__":
    # Sinh ma tran kich thuoc 20 x 20
    size = 20
    A = generate_matrix(size, size)
    B = generate_matrix(size, size)

    # Kiem tra va so sanh
    test_and_compare(A, B)
```

```
Thoi gian nhan ma tran tuan tu: 0.0059 giay
Thoi gian nhan ma tran song song: 0.0613 giay
```

Kiểm thử 2 ma trận 400 x 400:

```
def test_and_compare(A, B):
    # Thuc hien nhan ma tran tuan tu
    start_time = time.time()
    sequential_result = sequential_matrix_multiply(A, B)
    sequential_time = time.time() - start_time

    # Thuc hien nhan ma tran song song
    start_time = time.time()
    parallel_result = parallel_matrix_multiply(A, B)
    parallel_time = time.time() - start_time

    # Kiem tra ket qua
    print(f"Thoi gian nhan ma tran tuan tu: {sequential_time:.4f} giay")
    print(f"Thoi gian nhan ma tran song song: {parallel_time:.4f} giay")

# Main
if __name__ == "__main__":
    # Sinh ma tran kich thuoc 400 x 400
    size = 400
    A = generate_matrix(size, size)
    B = generate_matrix(size, size)

    # Kiem tra va so sanh
    test_and_compare(A, B)
```

```
Thoi gian nhan ma tran tuan tu: 15.5994 giay
Thoi gian nhan ma tran song song: 9.7463 giay
```

2.4 Nhận xét kết quả

Ta thấy thuật toán song song chạy nhanh hơn thuật toán tuần tự đối với các testcase có kích thước lớn. Đối với những tescase có kích thước nhỏ thì thuật toán tuần tự lại hiệu quả hơn