VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY

CÀI ĐẶT THUẬT TOÁN SONG SONG

CS112.P11.KHTN

Sinh Viên :
Nguyễn Văn Minh - 23520945
Đồng Quốc Thắng - 23521421

Giảng viên :
Nguyễn Thanh Sơn

Ngày 30 tháng 11 năm 2024

Mục lục

Chương 1

Bài tập 1

1.1 Đề bài

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

- **Input:** 1 số nguyên X.
- **Output:** X có phải là số nguyên tố hay không.

1.2 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 số nhỏ và lớn.
- Chạy thử và show thời gian thực hiện cả song song và tuần tự trên các test case sau: $X = 1000000000000091$, $X = 10000000000000099$, $X = 100000000000000049$.

1.3 Hướng giải quyết

Chia khoảng kiểm tra $[2, \sqrt{X}]$ thành các đoạn nhỏ và xử lý song song trên nhiều luồng (threads).

1.4 Source Code

```python
import math
from multiprocessing import Pool
# thiet lap ham kiem tra mot doan so song song
def check_prime_range(args):
    x, start, end = args
    for i in range(start, end):
        if x % i == 0:
            return False
    return True

# thiet lap ham kiem tra so nguyen to song song
def is_prime_parallel(x, num_threads=4):
    if x < 2:
        return False
    sqrt_x = int(math.sqrt(x)) + 1
    step = sqrt_x // num_threads
    ranges = [(x, i, min(i + step, sqrt_x)) for i in range(2, sqrt_x, step)]

    with Pool(num_threads) as pool:
        results = pool.map(check_prime_range, ranges)

    return all(results)

test_cases = [1000000000000091, 10000000000000099, 100000000000000049]
for x in test_cases:
    parallel_result = is_prime_parallel(x, num_threads=4)
    print(f"so nguyen to: {parallel_result}")
```

1.5 Kết quả

```
[3]  import math
     from multiprocessing import Pool
     # thiet lap ham kiem tra mot doan so song song
     def check_prime_range(args):
         x, start, end = args
         for i in range(start, end):
             if x % i == 0:
                 return False
         return True

     # thiet lap ham kiem tra so nguyen to song song
     def is_prime_parallel(x, num_threads=4):
         if x < 2:
             return False
         sqrt_x = int(math.sqrt(x)) + 1
         step = sqrt_x // num_threads
         ranges = [(x, i, min(i + step, sqrt_x)) for i in range(2, sqrt_x, step)]

         with Pool(num_threads) as pool:
             results = pool.map(check_prime_range, ranges)

         return all(results)

     test_cases = [1000000000000091, 10000000000000099, 100000000000000049]
     for x in test_cases:
         parallel_result = is_prime_parallel(x, num_threads=4)
         print(f"so nguyen to: {parallel_result}")
```

```
so nguyen to: True
so nguyen to: True
so nguyen to: True
```

Chương 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.

2.2 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.

2.3 Hướng giải quyết

Chia ma trận A thành các hàng (hoặc khối) và xử lý song song trên nhiều luồng (threads).

2.4 Source Code

```python
import numpy as np
from multiprocessing import Pool

# ham nhan mot hang ma trn song song
def multiply_row(args):
    A_row, B = args
    return np.dot(A_row, B)

# ham nhan ma tran song song
def matrix_multiply_parallel(A, B, num_threads=4):
    rows_A, cols_A = A.shape
    rows_B, cols_B = B.shape
    if cols_A != rows_B:
        raise ValueError("so cot cua ma tran A phai bang so hang ma tran B")

    with Pool(num_threads) as pool:
        result = pool.map(multiply_row, [(A[i, :], B) for i in range(rows_A)])
    return np.array(result)

# set ma tran random
np.random.seed(42)
A = np.random.rand(400, 400)
B = np.random.rand(400, 400)

res_result = matrix_multiply_parallel(A, B, num_threads=4)
```

2.5 Kết quả

```
[4] import numpy as np
    from multiprocessing import Pool

    # ham nhan mot hang ma trận song song
    def multiply_row(args):
        A_row, B = args
        return np.dot(A_row, B)

    # ham nhan ma tran song song
    def matrix_multiply_parallel(A, B, num_threads=4):
        rows_A, cols_A = A.shape
        rows_B, cols_B = B.shape
        if cols_A != rows_B:
            raise ValueError("so cot cua ma tran A phai bang so hang ma tran B")

        with Pool(num_threads) as pool:
            result = pool.map(multiply_row, [(A[i, :], B) for i in range(rows_A)])
        return np.array(result)

    # set ma tran random
    np.random.seed(42)
    A = np.random.rand(400, 400)
    B = np.random.rand(400, 400)

    res_result = matrix_multiply_parallel(A, B, num_threads=4)
```

```
[5] res_result
```

```
array([[100.20902654,  96.17151181,  94.47831996, ...,  97.92328132,
         97.33967215,  96.68615576],
       [101.49813672,  98.27677454,  98.06861919, ...,  99.83809727,
         98.8994251 ,  97.42392507],
       [102.00852174,  97.58375695,  99.88572007, ..., 100.68414461,
         97.33153957, 100.88025307],
       ...,
       [101.74485239, 104.4952841 , 104.65398653, ..., 102.60264914,
        102.01496831, 103.62484148],
       [108.398013  , 101.27455359, 104.83974253, ..., 105.26547529,
        103.42123126, 107.04145155],
       [ 94.68617527,  92.43103286,  96.25267928, ...,  95.63167356,
         89.55951656,  97.18110881]])
```