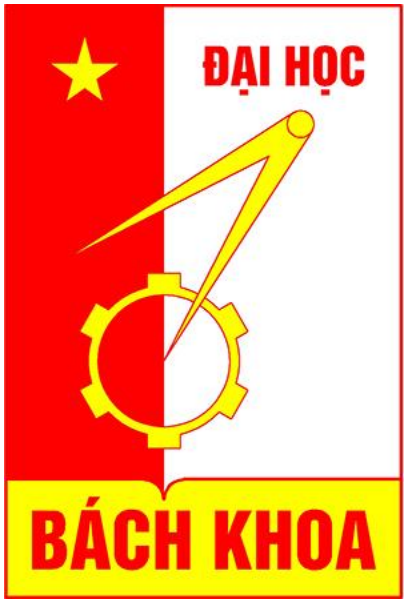
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC



BÁO CÁO GIỮA KÌ

MÔN TÍNH TOÁN SONG SONG

Giảng viên hướng dẫn: TS. Đoàn Duy Trung

Sinh viên thực hiện: Nguyễn Văn Triển

MSSV: 20195934

Lớp: Toán tin 02 – K64

Hà Nội, 11/2021

Mục lục

Phần 1. Bài tập chung

1. Bài toán ma trận nhân vector

Số luồng threads = 4				
STT	Kích thước ma trận	Thời gian thực hiện tuần tự	Thời gian thực hiện song song	Hiệu suất
Test 01	10	0.000001	0.000392	0.000637755
Test 02	100	0.000048	0.000503	0.023856859
Test 03	1000	0.004875	0.002927	0.416381961
Test 04	2000	0.019439	0.009524	0.510263545
Test 05	3000	0.043113	0.022682	0.475189578
Test 06	4000	0.075484	0.038439	0.490933687
Test 07	5000	0.117483	0.059689	0.492063027
Test 08	6000	0.169303	0.085541	0.494800739
Test 09	7000	0.0231865	0.116031	0.049957554
Test 10	8000	0.299606	0.158232	0.473365059
Test 11	9000	0.375391	0.200683	0.467641753
Test 12	10000	0.464938	0.233592	0.497596236

2. Bài toán tính toán số Fibonacci

2.1 Code tuần tự

```
#include <iostream>
#include <math.h>
#include <inttypes.h>
#include <windows.h>

using namespace std;

// Function that converts numbers form LongInt type to double type
double LiToDouble(LARGE_INTEGER x) {
    double result =
        ((double)x.HighPart) * 4.294967296E9 +
(double)((x).LowPart);
    return result;
}
// Function that gets the timestamp in seconds
double GetTime() {
    LARGE_INTEGER lpFrequency, lpPerfomanceCount;
    QueryPerformanceFrequency(&lpFrequency);
    QueryPerformanceCounter(&lpPerfomanceCount);
    return LiToDouble(lpPerfomanceCount) / LiToDouble(lpFrequency);
```

```
}

uint64_t fib(unsigned m) { // Direct Calculation, correct for abs(m)
<= 92
    double sqrt5r = 1.0 / sqrt(5.0);
    double golden = (1.0 + sqrt(5.0)) / 2.0;
    return rint(pow(golden, m) * sqrt5r);
}

void fibo(uint64_t n, uint64_t *f) {
    for (int i = 1; i <= n; i++)
        f[i] = fib(i);
}
int main()
{
    uint64_t f[100], n;
    double Start, Finish, Duration;

    cout << "n = ";
    cin >> n;
    Start = GetTime();
    fibo(n, f);
    Finish = GetTime();
    Duration = Finish - Start;
    //print result
    cout << "result: ";
    for (int i = 1; i <= n; i++)
        cout << f[i] << " ";
    cout << endl;
    cout << "time of execution: " << Duration << endl;
    return 0;
}
```

2.2 Code song song

```
#include <iostream>
#include <math.h>
#include <inttypes.h>
#include <omp.h>
#include <Windows.h>

using namespace std;

// Function that converts numbers form LongInt type to double type
double LiToDouble(LARGE_INTEGER x) {
    double result =
        ((double)x.HighPart) * 4.294967296E9 +
(double)((x).LowPart);
    return result;
}
// Function that gets the timestamp in seconds
double GetTime() {
    LARGE_INTEGER lpFrequency, lpPerfomanceCount;
```

```
    QueryPerformanceFrequency(&lpFrequency);
    QueryPerformanceCounter(&lpPerfomanceCount);
    return LiToDouble(lpPerfomanceCount) / LiToDouble(lpFrequency);
}

uint64_t fib(unsigned m) { // Direct Calculation, correct for abs(m)
<= 92
    double sqrt5r = 1.0 / sqrt(5.0);
    double golden = (1.0 + sqrt(5.0)) / 2.0;
    return rint(pow(golden, m) * sqrt5r);
}

void fib_sequence_parallel(uint64_t n, uint64_t* f) {
#pragma omp parallel
    {
        size_t thread = omp_get_thread_num();
        size_t threads = omp_get_num_threads();
        int start = (thread + 0) * n / threads;
        int end = (thread + 1) * n / threads;
        for (int i = start; i < end; i++) {
            f[i] = fib(i);
        }
    }
}
// max 92
int main(void) {
    uint64_t n;
    uint64_t f[1000];
    double Start, Finish, Duration;

    cout << "n = ";
    cin >> n;
    //program
    Start = GetTime();
    fib_sequence_parallel(n+1, f);
    Finish = GetTime();
    Duration = Finish - Start;
    //end
    for (int i = 1; i <= n; i++)
        cout << f[i] << " ";
    cout << "\n time of execution: " << Duration;

    return 0;
}
```

2.3 Kết quả

Số luồng threads = 4				
STT	Kích thước (n số fibonacci)	Thời gian thực hiện tuần tự	Thời gian thực hiện song song	Hiệu suất
Test 01	5	0.0000461	0.0000557	0.206912029
Test 02	10	0.0000583	0.0000582	0.250429553
Test 03	20	0.0000709	0.0000733	0.241814461
Test 04	30	0.0000861	0.0000909	0.23679868
Test 05	40	0.0001074	0.0001026	0.261695906
Test 06	50	0.0001325	0.0001256	0.263734076
Test 07	60	0.0001515	0.0001397	0.271116679
Test 08	70	0.0001614	0.0001561	0.258488149
Test 09	80	0.0001689	0.000166	0.25436747
Test 10	90	0.00018	0.000172	0.261627907

3. Bài toán tìm số nguyên tố từ 1 tới n

3.1 Code tuần tự

```
#include <iostream>
#include<stdio.h>
#include<omp.h>
#include <Windows.h>

using namespace std;
// Function that converts numbers form LongInt type to double type
double LiToDouble(LARGE_INTEGER x) {
    double result =
        ((double)x.HighPart) * 4.294967296E9 +
(double)((x).LowPart);
    return result;
}
// Function that gets the timestamp in seconds
double GetTime() {
    LARGE_INTEGER lpFrequency, lpPerfomanceCount;
    QueryPerformanceFrequency(&lpFrequency);
    QueryPerformanceCounter(&lpPerfomanceCount);
    return LiToDouble(lpPerfomanceCount) / LiToDouble(lpFrequency);
}
void PrimeNumber(int n, int Prime[]) {
    int i, j;
    for (i = 2;i * i <= n;i++)
    {
        for (j = i * i;j <= n;j = j + i)
        {
```

```
            if (Prime[j] == 1)
                Prime[j] = 0;
        }
    }
}
int main()
{
    int i, n;
    double Start, Finish, Duration;

    cout << "n = ";
    cin >> n;
    int* Prime = new int[n];
    for (i = 2;i <= n;i++)
        Prime[i] = 1;
    Start = GetTime();
    //program
    PrimeNumber(n, Prime);
    Finish = GetTime();
    Duration = Finish - Start;
    for (i = 2; i <= n; i++)
        if (Prime[i] == 1)
            cout << i << " ";
    cout << "\ntime of execution: " << Duration;
    return 0;
}
```

3.2 Code song song

```
#include <iostream>
#include <omp.h>
#include <windows.h>

using namespace std;
// Function that converts numbers form LongInt type to double type
double LiToDouble(LARGE_INTEGER x) {
    double result =
        ((double)x.HighPart) * 4.294967296E9 +
(double)((x).LowPart);
    return result;
}
// Function that gets the timestamp in seconds
double GetTime() {
    LARGE_INTEGER lpFrequency, lpPerfomanceCount;
    QueryPerformanceFrequency(&lpFrequency);
    QueryPerformanceCounter(&lpPerfomanceCount);
    return LiToDouble(lpPerfomanceCount) / LiToDouble(lpFrequency);
}
void PrimeNumberParallel(int n, int Prime[]) {
    int i, j;
#pragma omp parallel
    {
        size_t thread = omp_get_thread_num();
        size_t threads = omp_get_num_threads();
```

```
        int start = (thread + 0) * n / threads;
        int end = (thread + 1) * n / threads;
        if ((start == 0) or (start == 1)) start = 2;
        for (i = start; i <= end; i++) {
            for (j = 2; j * i <= n; j++)
                Prime[i * j] = 0;
        }
    }
}
int main()
{
    int i, n;
    double Start, Finish, Duration;

    cout << "n = ";
    cin >> n;
    int* Prime = new int[n];
    for (i = 2; i <= n; i++)
        Prime[i] = 1;
    Start = GetTime();
    //program
    PrimeNumberParallel(n, Prime);
    Finish = GetTime();
    Duration = Finish - Start;

    cout << "result: ";
    for (i = 2; i <= n; i++)
        if (Prime[i] == 1)
            cout << i << " ";
    cout << "\ntime of execution: " << Duration;
    return 0;
}
```

3.3 Kết quả

Số luồng threads = 4				
STT	Kích thước ma trận	Thời gian thực hiện tuần tự	Thời gian thực hiện song song	Hiệu suất
Test 01	10	0.0000012	0.0004856	0.000617792
Test 02	100	0.0000021	0.0004567	0.001149551
Test 03	1000	0.0000124	0.0004193	0.007393275
Test 04	2000	0.0000211	0.0004881	0.010807212
Test 05	3000	0.000033	0.0005018	0.016440813
Test 06	4000	0.0000443	0.0004413	0.025096306
Test 07	5000	0.0000567	0.0004399	0.032223233
Test 08	6000	0.0000686	0.000431	0.039791183
Test 09	7000	0.0000822	0.0004723	0.043510481
Test 10	8000	0.0000937	0.0004417	0.053033733
Test 11	9000	0.0001078	0.0004477	0.06019656
Test 12	10000	0.0001241	0.0004744	0.065398398

Phần 2. Bài tập riêng

Bài 1. Viết chương trình tìm giá trị lớn nhất (nhỏ nhất) trong mảng một chiều

1. Code tuần tự

```
#include <iostream>
#include <omp.h>
#include <fstream>
#include <windows.h>

using namespace std;

// Function that converts numbers form LongInt type to double type
double LiToDouble(LARGE_INTEGER x) {
    double result =
        ((double)x.HighPart) * 4.294967296E9 +
(double)((x).LowPart);
    return result;
}
// Function that gets the timestamp in seconds
double GetTime() {
    LARGE_INTEGER lpFrequency, lpPerfomanceCount;
    QueryPerformanceFrequency(&lpFrequency);
    QueryPerformanceCounter(&lpPerfomanceCount);
    return LiToDouble(lpPerfomanceCount) / LiToDouble(lpFrequency);
}
void FindMax(int n, int a[], int b[]) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (a[i] >= a[j]) b[i]++;
        }
    }
}
void FindMin(int n, int a[], int c[]) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (a[i] <= a[j]) c[i]++;
        }
    }
}
int main()
{
    int flag = 0, n, i;
    fstream file;
    double Start, Finish, Duration;
    file.open("../../../array.txt", ios::in); // random array in
file "array.txt"
    if (file.fail()) {
        cout << "fail to open file \"array\"";
        flag = 1;
    }
    if (flag != 1) {
```

```
            file >> n;
            int* a = new int[n];
            int* b = new int[n] {0}; // find max
            int* c = new int[n] {0}; //find min
            for (i = 0; i < n; i++)
                file >> a[i];
            Start = GetTime();
            //program
            FindMax(n, a, b);
            FindMin(n, a, c);
            Finish = GetTime();
            Duration = Finish - Start;
            for (int i = 0; i < n; i++)
                if (b[i] == n) {
                    cout << "Max: " << a[i] << endl;
                    break;
                }
            for (int i = 0; i < n; i++)
                if (c[i] == n) {
                    cout << "Min: " << a[i] << endl;
                    break;
                }
            cout << "time of execution: " << Duration;
        }
        file.close();
        return 0;
    }
```

2. Code song song

```
#include <iostream>
#include <omp.h>
#include <fstream>
#include <windows.h>

using namespace std;

// Function that converts numbers form LongInt type to double type
double LiToDouble(LARGE_INTEGER x) {
    double result =
        ((double)x.HighPart) * 4.294967296E9 +
(double)((x).LowPart);
    return result;
}
// Function that gets the timestamp in seconds
double GetTime() {
    LARGE_INTEGER lpFrequency, lpPerfomanceCount;
    QueryPerformanceFrequency(&lpFrequency);
    QueryPerformanceCounter(&lpPerfomanceCount);
    return LiToDouble(lpPerfomanceCount) / LiToDouble(lpFrequency);
}
```

```
void FindMax_Parallel(int n, int a[], int b[]) {
#pragma omp parallel shared(b)
    {
        #pragma omp for
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (a[i] >= a[j]) b[i]++;
            }
        }
    }
}
void FindMin_Parallel(int n, int a[], int c[]) {
#pragma omp parallel shared(c)
    {
        #pragma omp for
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (a[i] <= a[j]) c[i]++;
            }
        }
    }
}
int main()
{
    int flag = 0, n, i;
    fstream file;
    double Start, Finish, Duration;
    file.open("../../../array.txt", ios::in); // random array in
file "array.txt"
    if (file.fail()) {
        cout << "fail to open file \"array\"";
        flag = 1;
    }
    if (flag != 1) {
        file >> n;
        int* a = new int[n];
        int* b = new int[n] {0}; // find max
        int* c = new int[n] {0}; //find min
        for (i = 0; i < n; i++)
            file >> a[i];
        Start = GetTime();
        //program
        FindMax_Parallel(n, a, b);
        FindMin_Parallel(n, a, c);
        Finish = GetTime();
        Duration = Finish - Start;
        for (int i = 0; i < n; i++)
            if (b[i] == n) {
                cout << "Max: " << a[i] << endl;
                break;
            }
        for (int i = 0; i < n; i++)
            if (c[i] == n) {
                cout << "Min: " << a[i] << endl;
```

```
                break;
            }
        cout << "time of execution: " << Duration;
    }
    file.close();
    return 0;
}
```

3. Kết quả

Số luồng threads = 4				
STT	Kích thước mảng	Thời gian thực hiện tuần tự	Thời gian thực hiện song song	Hiệu suất
Test 01	10	0.0000066	0.0033302	0.000495466
Test 02	100	0.0001904	0.0031119	0.015296121
Test 03	1000	0.0166313	0.012444	0.33412287
Test 04	2000	0.0663144	0.0426465	0.38874468
Test 05	3000	0.138326	0.0797991	0.433357018
Test 06	4000	0.238736	0.120053	0.497147093
Test 07	5000	0.402135	0.188491	0.533361009
Test 08	6000	0.593504	0.233975	0.634153221
Test 09	7000	0.874892	0.314325	0.695849837
Test 10	8000	1.21917	0.464043	0.656819519
Test 11	9000	1.48849	0.53002	0.702091431
Test 12	10000	1.80565	0.663665	0.680181266