Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**«Сортировка Шелла OpenMP»**

**Выполнил**:

студент группы 381708-2

Фамилия И.О.Утшо М.М.

**Проверил**:

Валентин Волокитин

Нижний Новгород

2020

**Содержание**

[Постановка задачи 3](#_Toc26446451)

[Метод решения 4](#_Toc26446452)

[Схема распараллеливания 5](#_Toc26446453)

[Описание программной реализации 6](#_Toc26446454)

[Подтверждение корректности 7](#_Toc26446455)

[Результаты экспериментов 8](#_Toc26446456)

[Заключение 9](#_Toc26446457)

[Приложение 10](#_Toc26446458)

# Постановка задачи

одной из наиболее важных проблем в информатике является процесс сортировки, по этой причине было разработано множество алгоритмов сортировки, таких как Quicksort, Merge sort, Bubble sort, Insert sort и Selection sort...и т.д. [1]. В информатике алгоритм сортировки-это алгоритм, который упорядочивает компоненты списка в определенном порядке. Алгоритмы сортировки преподаются в некоторых областях, таких как Информатика и математика. Существует множество алгоритмов сортировки, применяемых в области компьютерных наук. Они различаются по своей производительности, функциональности, использованию ресурсов и приложениям [2]. Кроме того, были исследованы алгоритмы параллельной сортировки для улучшения работы сортировки. Было создано множество алгоритмов параллельной сортировки, таких как сортировка столбцов [3], параллельная сортировка по радиусу [4], битоническая сортировка [5], выборочная сортировка [6] и параллельная сортировка слиянием [7]. Параллельные сортировки обычно требуют замены фиксированного количества данных между элементами процесса слияния и обработки. Алгоритм быстрой сортировки Хоара - это одна из наиболее интенсивно изучаемых проблем в области информатики. Он использует стратегию "разделяй и Побеждай", сводя задачу сортировки к нескольким более простым задачам сортировки и решая каждую из них [8]. хорошей производительности на практике алгоритм Quicksort считается одним из самых популярных алгоритмов сортировки. Фундаментальный алгоритм может быть кратко идентифицирован следующим образом [8, 9 и 10] одно значение выбирается из входных данных, которые обычно называются pivot value, этот pivot используется для разбиения входного набора данных на два подмножества, одно из которых содержит входные данные меньшего размера по сравнению с pivot value и другим значением. другое содержит входные данные, превышающие значение pivot. На каждом отдельном шаге эти разделенные наборы данных подразделяются, выбирая PIV точек поворота из каждого набора. Рекурсивная операция больше не происходит, когда нет никакого подразделения, которое возможно. Он использует технику " разделяй и Побеждай”, сводя задачу сортировки к более простым задачам сортировки и решая каждую из них [11]

# Метод решения

#include "pch.h"

#include <iostream>

#include "malloc.h"

#include "omp.h"

#include <time.h>

#include <ctime>

#include <cstdlib>

using namespace std;

//a[] is sort , n is size

void ShellSort(int a[], int n)

{

int gap;

int j;

int i;

int element;

for (gap = n / 2; gap > 0; gap /= 2)

{

#pragma omp parallel for shared( a, gap,n) private(i) default(none)

for (i = gap; i < n; i++)

{

#pragma omp parallel

element = a[i];

for (j = i; j >= gap; j -= gap)

{

#pragma omp critical

if (element < a[j - gap])

a[j] = a[j - gap];

}

a[j] = element;

}

}

}

void mergeSort(int a[], int t[], int n)

{

#pragma omp parallel sections

if (n > 1) {

#pragma omp section

for (int i = 0; i < n / 2; i++) {

t[i] = a[i];

}

#pragma omp section

mergeSort(a, t, n / 2);

#pragma omp section

for (int i = n / 2; i < n; i++) {

t[i] = a[i];

}

mergeSort(a, t, n / 2);

ShellSort(a, n);

}

}

void print(int a[], int size)

{

for (int i = 0; i < size; i++)

cout << a[i] << "";

cout << endl;

}

int main()

{

int n = 2000;

int a[2000];

int i;

srand((time(NULL)));

#pragma omp parallel for num\_threads(i)

for (i = 0; i < n; i++)

{

a[i] = (rand() % 100000) + 1;

}

double startTime = clock();

cout << "enter element:" << endl;

cin >> n;

cout << "enter elements the one that you put earlier:" << endl;

for (int i = 0; i < n; i++)

{

cin >> a[i];

}

cout << "array seq before sorting: ";

print(a, n);

ShellSort(a, n);

cout << "array seq after sorting: ";

print(a, n);

cout << "enter the size of the array" << endl;

cin >> n;

for (int i = 0; i < n; i++) {

cout << "enter the element of merge" << i << endl;

cin >> a[i];

}

mergeSort(a, 0, n - 1);

cout << "\tSorted Array Elements with shell" << endl;

for (int i = 0; i < n; i++) {

cout << a[i] << "\t";

}

ShellSort(a, n);

double endTime = clock();

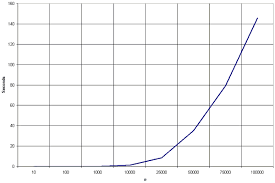
int totalTime = endTime - startTime; // The average time to run this

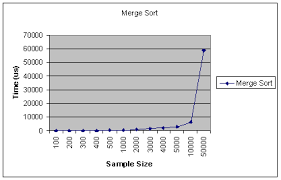
cout << "This is the time it took to run.\n" << endl;

cout << totalTime / 2000 << n << endl;

}

# Схема Шелла





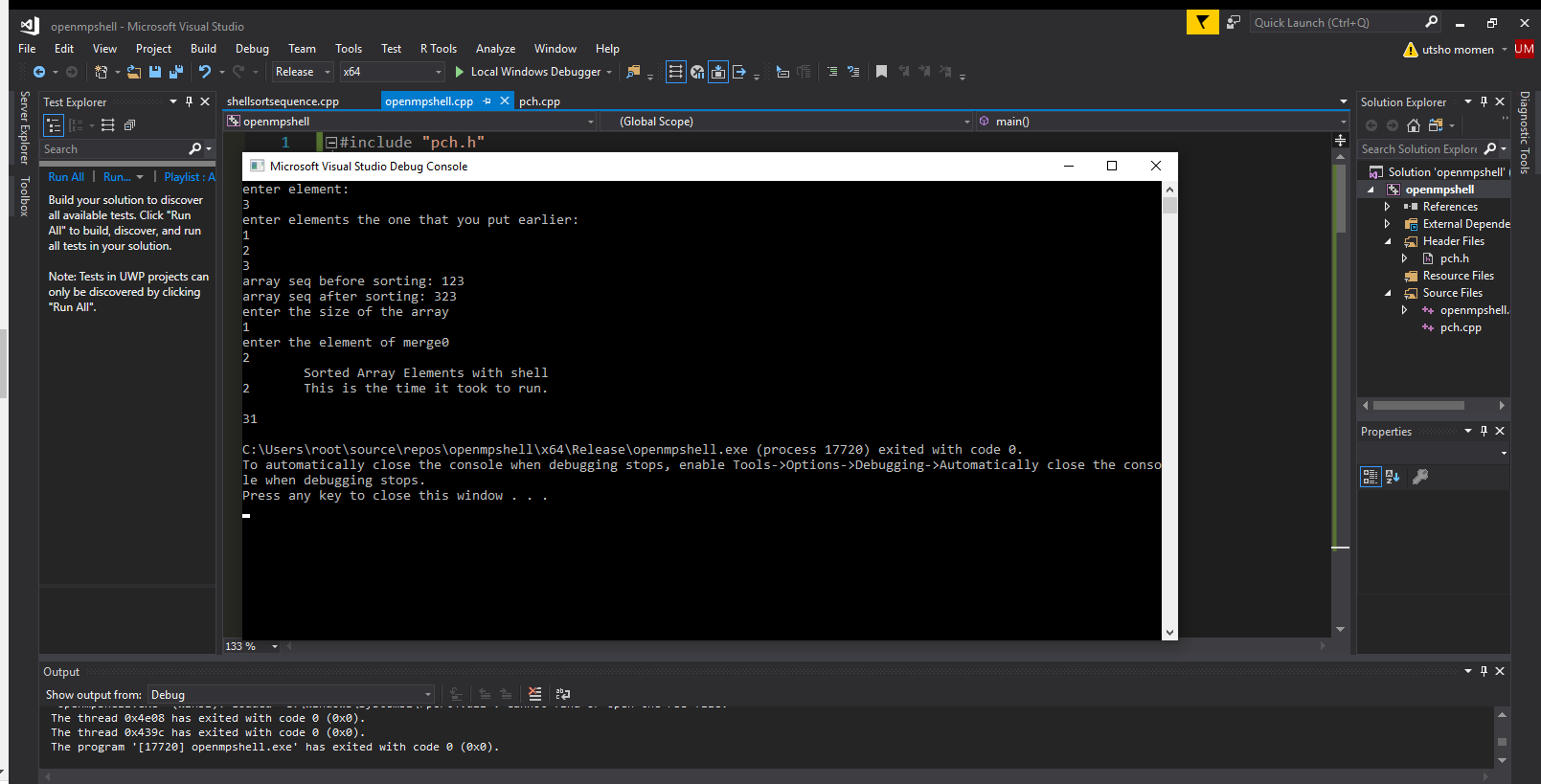
# Результаты экспериментов

Linux

//g++ -c openmpshell.cpp -o openmpshell.o -fopenmp

//g++ openmpshell.cpp openmpshell -fopenmp –lpthread

windows



# Заключение

Shell sort - это высокоэффективный алгоритм, который приходит к улучшению по сравнению с сортировкой вставки.

В то время как сортировка вставки работает путем увеличения ее элементов на 1, shell sort использует параметр “gap” для разделения массива на суб-массивы, элементы которых разделены “gap”. Затем мы можем отсортировать отдельный список с помощью сортировки вставки, чтобы получить полный отсортированный массив.

Сортировка оболочки выполняется быстрее, чем сортировка вставки, и занимает меньше времени для сортировки массива по сравнению с сортировкой вставки. В нашем предстоящем уроке мы рассмотрим все о технике сортировки кучи для сортировки структур данных.

# Приложение

1.A K Verma

2.A New Approach for Sorting List to Reduce Execution Time, volume 7

Posted: 2013-10-13

3.A K S Shukla

Review of Radix Sort & Proposed Modified Radix Sort for Heterogeneous Data Set in 4.Distributed Computing Environment, volume 2

Posted: 2012

5.J Hammad

A Comparative Study between Various Sorting Algorithms, volume 15

Posted: 2015-03

6.S K P Megha Jain

Bitonic Sorting Algorithm: A Review, volume 113

Posted: 2015-03