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 std thread»**

**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

Std thread in shell sort

Implementing thread in shell sort with merge

Поток может быть создан несколькими способами:

Использование указателя функции

Использование функтора

Использование лямбда-функции

Функция, вычисляющая сумму всех элементов между beginIndex и endIndex в векторе v

Теперь предположим, что вы хотите разбить вектор на две секции и вычислить общую сумму каждой секции в отдельном потоке t1 и t2,t3

мы будем создавать потоки с помощью указателей функций

# Метод решения

#include "pch.h"

#include <iostream>

#include "malloc.h"

#include "omp.h"

#include <ctime>

#include <cstdlib>

#include <thread>

#include <csignal>

class ShellSort {

int a[1000], n;

public:

void source();

void print();

void sortmain();

};

/\*

void shellsort::mergesort(int \*a, int t[], int n)

{

if (n > 1) {

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

t[i] = a[i];

}

mergesort(a, t, n / 2);

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

t[i] = a[i];

}

mergesort(a, t, n / 2);

}

}

\*/

void ShellSort::source()

{

std::cout << "type element : ";

std::cin >> n;

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

std::cin >> a[i];

}

void ShellSort::print() {

std::cout << "\n\_print data\_\n";

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

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

}

//t(temp) == element

void ShellSort::sortmain() {

int i, j, t, gap;

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

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

t = a[i];

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

if (t < a[j - gap])

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

else

break;

}

a[j] = t;

}

}

}

int main() {

std::cout << "\t0\_\_\_0\t" ;

ShellSort t;

t.source();

t.print();

t.sortmain();

ShellSort s;

std::thread t1(&ShellSort::source, &s);

std::thread t2(&ShellSort::sortmain, &s);

std::thread t3(&ShellSort::print, &s);

t1.join();

t2.join();

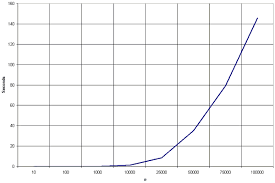
t3.join();

system("pause");

return 0;

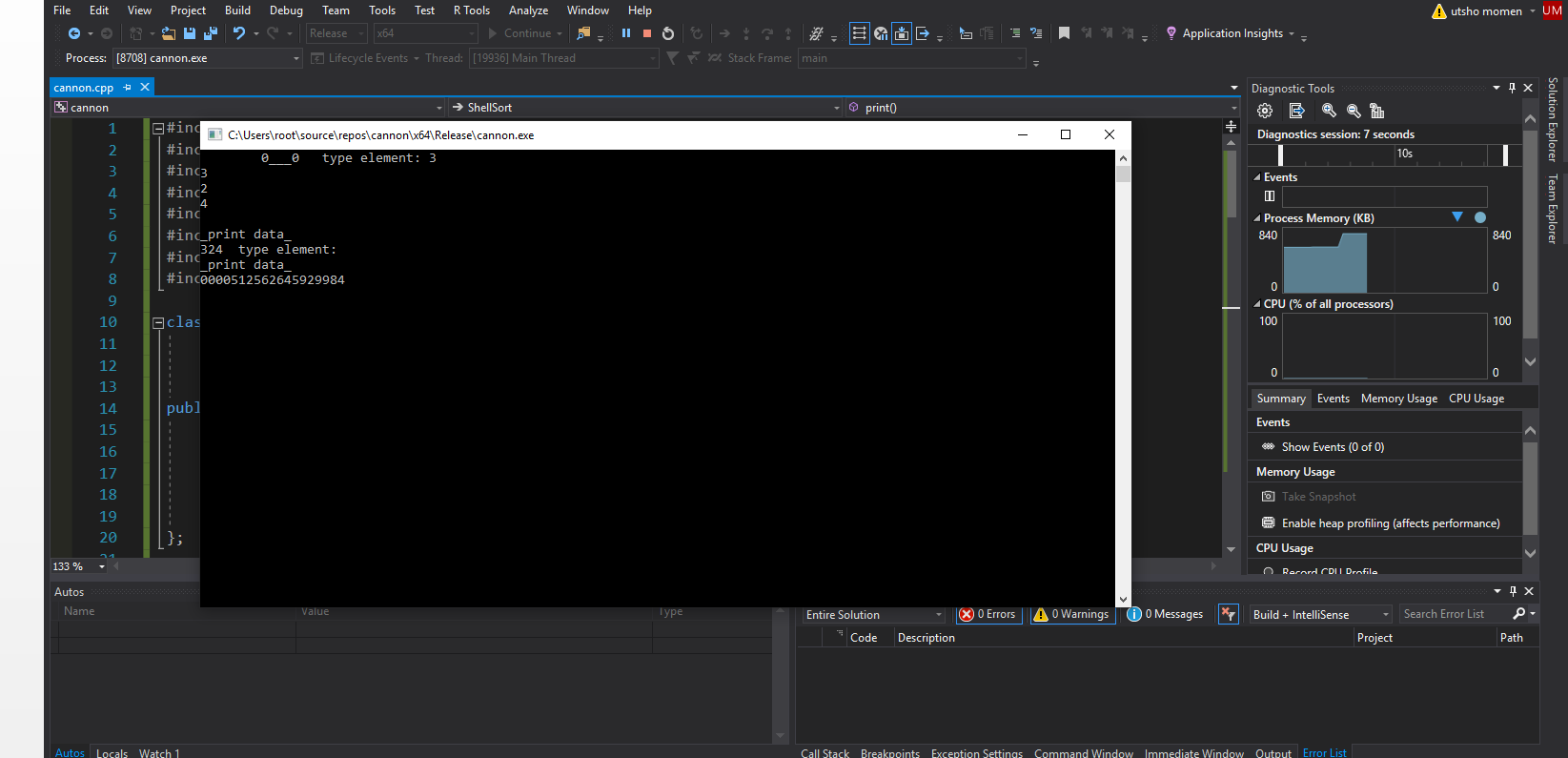
}

# Схема Шелла



# Результаты экспериментов

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