Санкт-Петербургский политехнический университет Петра Великого

Институт машиностроения, материалов и транспорта

Курсовая работа

Дисциплина: Объектно-ориентированное программирование

Тема: Сортировка Шелла

Выполнил студент группы 3331506/80401: Бондаренко И. А.

Преподаватель: Кузнецова Е. М.

«___ »_____ 2021 г.

Санкт-Петербург

2021

1.1 Предназначение сортировки Шелла

Сортировка Шелла нужна для упорядочивания элементов в массиве.

1.2 Описание работы алгоритма сортировки Шелла

Алгоритм сортировки Шелла заключается в сравнении значений, стоящих друг от друга на промежутке d.

Изначально этот интервал, как правило, берется равным N/2, где N - число элементов в массиве. После этого сопоставляются элементы, отстоящие друг от друга на d позиций, и в случае необходимости - меняются местами. Когда все элементы массива сравнятся между собой, интервал d уменьшается на определенное значение и снова повторяется проход по массиву. Это происходит до тех пор, пока промежуток d не станет равным 1. При d = 1 происходит последний проход по массиву.

1.3 Описание скорость работы этого алгоритма в зависимости от количества входных данных, т.е. в терминах O(n)

При равных размерах массива, скорость работы алгоритма зависит от того, как изменяется наш выбранный промежуток d. В конце сортировки у нас всегда d = 1.

Первоначально Шелл решил после каждого прохода делить d на 2. При таком подходе скорость алгоритма была $O(N^2)$.

Американский инженер Хиббард предложил брать значения

$$2^i - 1 \leq N, i \in \mathbb{N}.$$

У такой последовательности сложность будет поменьше - $O\left(N^{3/2}\right)$.

Одну из лучших последовательностей предложил американский ученым в области информатика – Седжвик:

$$d_i = 9 \cdot 2^i - 9 \cdot 2^{i/2} + 1 \text{ если } i \text{ четное}$$

$$d_i = 8 \cdot 2^i - 6 \cdot 2^{i/2} + 1 \text{ если } i \text{ нечетное}$$

При такой последовательности средняя сложность равна $O\left(N^{7/6}\right)$. Стоит остановиться на значении d_{i-1}, если $3 \cdot d_i > N$. Есть один нюанс: так как промежуток должен уменьшатся, то следует посчитать массив приращений перед запуском сортировки, а после использовать их уже в обратном порядке.

Математик Пратт Предложил последовательность, которая будет обеспечивать минимальную сложность. Значения будут:

$$2^i \cdot 3^j \leq N/2, i, j \in \mathbb{N}.$$

Они обеспечивают сложность алгоритма $O(N \cdot (\log N)^2)$.

Ниже, на рисунке 1, представлен график зависимости времени выполнения сортировки массива, от его размера при использовании последовательности Пратта.

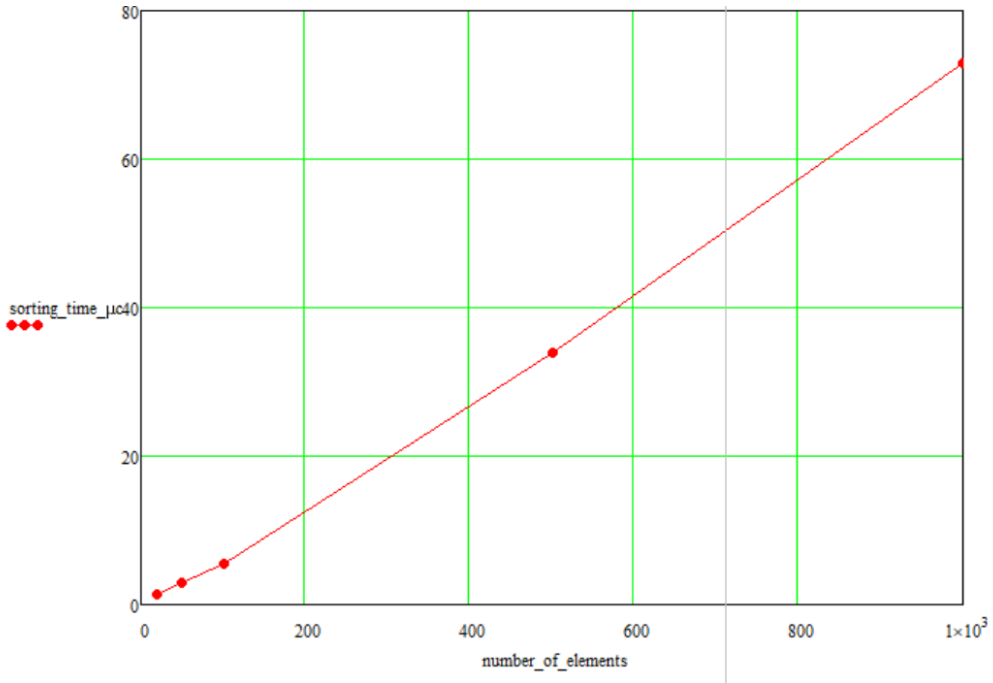

Рисунок 1 – График зависимости времени выполнения сортировки массива, от его размера

Список литературы

1. Седжвик, Р. Алгоритмы на C++ / Р. Седжвик – Вильямс, 2011. – 1056 с.

2. Хайнеман, Д. Алгоритмы. Справочник. С примерами на C, C++, Java и Python /Д. Хайнеман, Г. Поллис, С. Селков. – Вильямс, 2017. – 434 с.

Приложение

```cpp
#include <iostream>

using namespace std;

void shell_sort(int array[], const int size_of_array) //сортировка Шелла
{
    // дистанция выбрана согласно последовательности Пратта
    int distance[33] = { 1, 2, 3, 4, 6, 8, 9, 12, 16, 18, 24, 27, 32,
                         36, 48, 54, 64, 72, 81, 96, 108, 128, 144, 162,
                         192, 216, 243, 256, 288, 324, 384, 432, 486 };
    int iteration = 6;
    int position = 0;
    // в зависимости от номера итерации у нас берется разная дистанция
    while (iteration >= 0)
    {
        for (position = 0; position < size_of_array - distance[iteration]; position++)
        {
            // сравниваем элементы в паре, при необходимости - меняем местами
            if (array[position] > array[position + distance[iteration]])
            {
                std::swap(array[position], array[position + distance[iteration]]);
            }
        }
        iteration--;
    }
}

//главная функция
void main()
{
    setlocale(LC_ALL, "Rus");

    // задаем исходный массив
    int array[20] = { 1, 21, 3, 69, 85, 4, 45, 2, 9, 5, 110, 250, 700, 100, 1105 , 52, 99, 10, 9400, 7 };
    cout << " Исходный массив:  1, 21, 3, 69, 85, 4, 45, 2, 9, 5, 110, 250, 700, 100, 1105 , 52, 99, 10, 9400, 7 " << endl;

    // вызываем функцйию сортировки
    // необходимо, чтобы массив состоял из 20 элементов
    shell_sort(array, 20);

    // выводим результат
    cout << " Отсортированный массив:  " << endl;
    for (int number_of_elements = 0; number_of_elements < 20; number_of_elements++)
    {
        cout << array[number_of_elements] << ", ";
    }
}

```