

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:

Ананьевский М. С.

« \_\_\_\_ » \_\_\_\_\_ 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 меньше -  $O(N^{3/2})$ .

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  $O(N^{7/6})$ . Стоит остановиться на значении  $d_{i-1}$ , если  $3 \cdot d_i > N$ . Есть один нюанс: так как промежуток должен уменьшаться, 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.

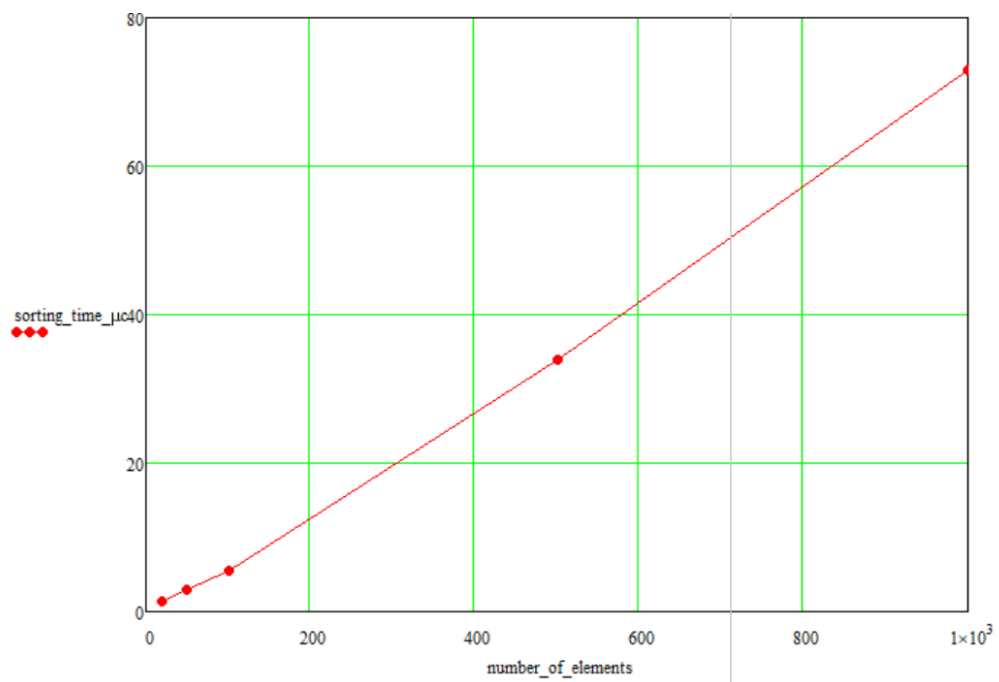


Рисунок 1 – График зависимости времени выполнения сортировки массива, от его размера

## 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

```
1  #include <iostream>
2
3  using namespace std;
4
5  void shell_sort(int array[], int size_of_array) //сортировка Шелла
6  {
7      // дистанция выбрана согласно последовательности Пратта
8      int distance[7] = { 9, 8, 6, 4, 3, 2, 1 };
9      int iteration = 0;
10     int position = 0;
11     // в зависимости от номера итерации у нас берется разная дистанция
12     while (iteration < 7)
13     {
14         for (position = 0; position < size_of_array - distance[iteration]; position++)
15         {
16             // сравниваем элементы в паре, при необходимости - меняем местами
17             if (array[position] > array[position + distance[iteration]])
18             {
19                 int temporary_variable = array[position];
20                 array[position] = array[position + distance[iteration]];
21                 array[position + distance[iteration]] = temporary_variable;
22             }
23         }
24         iteration ++;
25     }
26 }
27
28 //главная функция
29 void main()
30 {
31     setlocale(LC_ALL, "Rus");
32
33     // задаем исходный массив
34     int array[20] = { 1, 21, 3, 69, 85, 4, 45, 2, 9, 5, 110, 250, 700, 100, 1105, 52, 99, 10, 9400, 7 };
35     cout << " Исходный массив: 1, 21, 3, 69, 85, 4, 45, 2, 9, 5, 110, 250, 700, 100, 1105, 52, 99, 10, 9400, 7 " << endl;
36
37     // вызываем функцию сортировки
38     // необходимо, чтобы массив состоял из 20 элементов
39     shell_sort(array, 20);
40
41     // выводим результат
42     cout << " Отсортированный массив: " << endl;
43     for (int number_of_elements = 0; number_of_elements < 20; number_of_elements++)
44     {
45         cout << array[number_of_elements] << ", ";
46     }
```