



Санкт-Петербургский
государственный
университет

Сортировки

Сортировка Шелла, линейные сортировки

Никита Гребень

Факультет прикладной математики - процессов управления



Санкт-Петербургский
государственный
университет

- ❶ Случайные числа и время
- ❷ Сортировка Шелла
- ❸ Сортировка подсчетом
- ❹ Задачи



Санкт-Петербургский
государственный
университет

1 Случайные числа и время

- `std::mt19937`
- `std::chrono`

2 Сортировка Шелла

3 Сортировка подсчетом

4 Задачи

std::mt19937

Санкт-Петербургский
государственный
университет

- ▶ Очень эффективный генератор псевдослучайных чисел.
- ▶ Определяется в заголовочном файле `< random >`.
- ▶ Генерирует 32-битные псевдослучайные числа, используя хорошо известный и популярный алгоритм под названием Mersenne Twister.
- ▶ Функция `rand()` может использоваться в небольшом диапазоне, до 32767. Неэффективна для генерации действительно случайных чисел.

std::mt19937

Санкт-Петербургский
государственный
университет

```
#include <random>
using namespace std;
...
mt19937 mt(time(nullptr));
for(int i = 0; i < n; ++i)
    a[i] = (mt() % 100) + 100;
```

std::chrono

Санкт-Петербургский
государственный
университет

- ▶ Библиотека для работы со временем.
- ▶ Рассматривается три основных концепта
 - ✓ **Duration** - измеряет промежутки времени, например: одну минуту, два часа или десять миллисекунд.
 - ✓ **Timepoint** - ссылка на определенный момент времени, например, на чей-то день рождения, когда следующая пара и т.д.
 - ✓ **Clocks** - связывает момент во времени с реальным физическим временем.

std::chrono

Санкт-Петербургский
государственный
университет

```
#include <chrono>
using namespace chrono;
...
auto start = steady_clock::now();
sort(a.begin(), a.end());
auto end = steady_clock::now();
duration<double> elapsed_seconds = end - start;
cout << "elapsed time: " << elapsed_seconds.count() << "s\n";
```

Обзор



Санкт-Петербургский
государственный
университет


1 Случайные числа и время

2 Сортировка Шелла

- Алгоритм
- Пример
- Задание
- Реализация
- Анализ

3 Сортировка подсчетом

4 Задачи



Сортировка Шелла

Санкт-Петербургский
государственный
университет

- ▶ Сортировка Шелла - обобщенная версия алгоритма сортировки вставками.
- ▶ Сначала сортируем элементы, которые находятся далеко друг от друга, последовательно уменьшаем интервал между элементами, подлежащими сортировке.
- ▶ Расстояние между элементами уменьшается в зависимости от используемой последовательности.
- ▶ Последовательность, предложенная Дональдом Шеллом: $\lfloor \frac{N}{2} \rfloor, \lfloor \frac{N}{4} \rfloor, \dots, 1$.
- ▶ Алгоритмическая сложность варьируются и зависит от выбранной последовательности: $\Theta(N^2)$, $\Theta(N^{\frac{3}{2}})$, $\Theta(N \log^2 N)$, $O(N^{\frac{4}{3}})$.

Алгоритм



Санкт-Петербургский
государственный
университет

Рассматриваем смещения на каждом шагу (выберем последовательность Шелла, для наглядности): $h_k = N/2$, $h_{k-1} = h_k/2$, \dots , $h_0 = 1$. На каждом проходе сортируем элементы, которые находятся на расстоянии h_k друг от друга. На последнем шагу h_0 запускается обычная сортировка вставками. Т.е. выполняется следующая последовательность действий:

1. Разбиваем массив на списки, элементы внутри которых отстают друг от друга на h_i , кол-во таких списков равно h_i .
2. Внутри каждого такого списка элементы сортируются сортировкой вставками
3. Соединяем списки обратно в массив, переходим к шагу $i = i - 1$.

Пример



Санкт-Петербургский
государственный
университет



Рис. 1: Изначальный массив

Пример



Санкт-Петербургский
государственный
университет

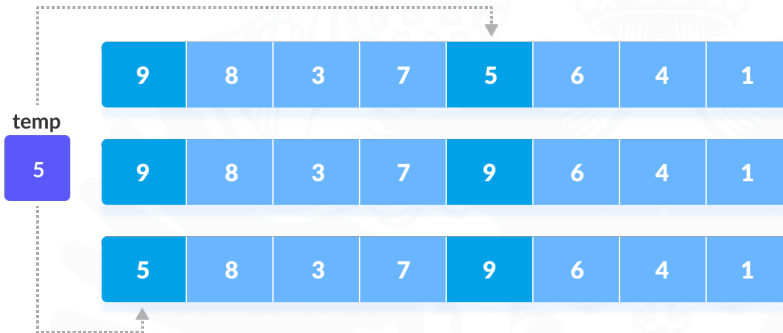


Рис. 2: Сравнение элементов массива на интервале $N/2$

Пример



Санкт-Петербургский
государственный
университет

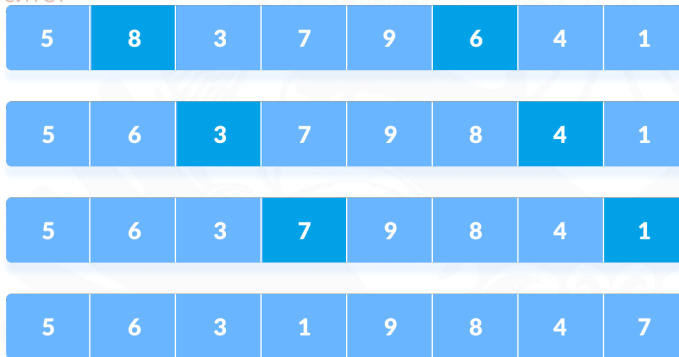


Рис. 3: Перестраивание элементов массива на интервале $N/2$

Пример



Санкт-Петербургский
государственный
университет

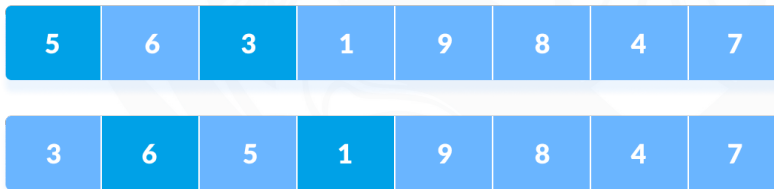


Рис. 4: Перестраивание элементов массива на интервале $N/4$

Пример



Санкт-Петербургский
государственный
университет

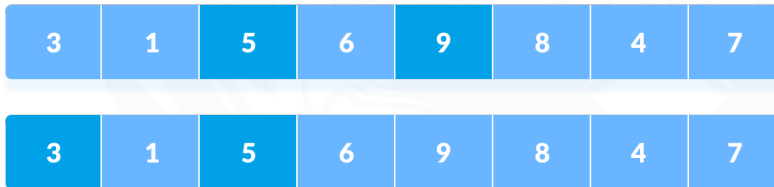


Рис. 5: Сравниваются все элементы в массиве, расположенные на текущем интервале

Пример



Санкт-Петербургский
государственный
университет

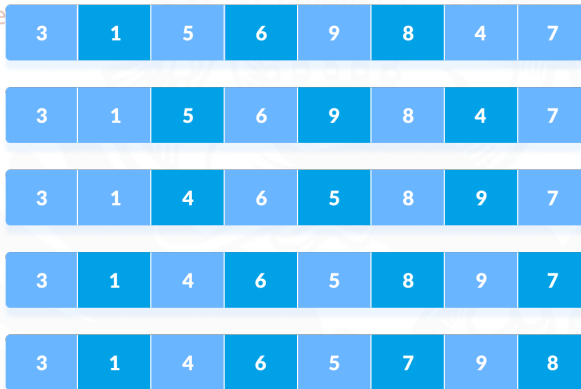


Рис. 6: Перестраивание элементов массива на интервале $N/4$

Пример



Санкт-Петербургский
государственный
университет

3	1	4	6	5	7	9	8
1	3	4	6	5	7	9	8
1	3	4	6	5	7	9	8
1	3	4	6	5	7	9	8
1	3	4	5	6	7	9	8

Рис. 7: Сортировка массива на
последней итерации

1	3	4	5	6	7	9	8
1	3	4	5	6	7	9	8
1	3	4	5	6	7	9	8
1	3	4	5	6	7	8	9

Рис. 8: Сортировка массива на
последней итерации

Задание



Санкт-Петербургский
государственный
университет

- ▶ Реализовать сортировку Шелла по последовательности интервалов:

$$h_k = \frac{3^k - 1}{2}.$$

- ▶ h_k должно быть не более $\left\lceil \frac{N}{3} \right\rceil$.

- ▶ При каком минимальном N , приблизительно, расчет на вашем устройстве занимает более двух секунд?

Реализация

```
1 void shell_sort(vector<int>& a) {  
2     int h = 1;  
3     int n = a.size();  
4     while (3 * h + 1 <= (n + 2) / 3)  
5         h = 3 * h + 1;  
6     for(; h > 0; h /= 3) {  
7         for(int i = h; i < n; ++i) {  
8             int temp = a[i];  
9             int j = i;  
10            for(; j >= h && a[j - h] > temp; j -= h)  
11                a[j] = a[j - h];  
12            a[j] = temp;  
13        }  
14    }  
15 }
```

Анализ



Санкт-Петербургский
государственный
университет

Временная сложность

- ▶ Производительность в наихудшем случае
 - ✓ Худшая оценка $\Theta(N^2)$ - последовательность Шелла.
 - ✓ Наилучшая оценка $\Theta(N \log^2 N)$ - последовательность гладких чисел $2^p 3^q$.
- ▶ Производительность в среднем
 - ✓ Строго зависит от выбранной последовательности.
- ▶ Производительность в лучшем случае
 - ✓ Лучшая возможная оценка $\Theta(N \log N)$ - достигается у многих последовательностей.
 - ✓ Наихудшая оценка $\Theta(N \log^2 N)$

Обзор



Санкт-Петербургский
государственный
университет

- ① Случайные числа и время
- ② Сортировка Шелла
- ③ Сортировка подсчетом
 - Алгоритм
 - Реализация
- ④ Задачи

Сортировка подсчетом



Санкт-Петербургский
государственный
университет

- ▶ Линейное время работы $\Theta(N + K)$
- ▶ Используется, когда элементы массива (либо ключи более сложной структуры данных) принимают небольшие по модулю значения
- ▶ Является устойчивой

Алгоритм



Санкт-Петербургский
государственный
университет

Полагаем, что длина массива N и все элементы в нём меньше K :

1. Инициализируем некоторый массив C длины K нулями.
2. Пройдемся по нашему массиву и запишем в $C[i]$ количество чисел, равных i .
3. Проходимся по массиву C и записываем в исходный массив элемент i ровно $C[i]$ раз.

Реализация



Санкт-Петербургский
государственный
университет

```
1 void counting_sort(vector<int>& A, int K) {  
2     vector<int> C(K, 0);  
3     for(int x: A)  
4         C[x]++;  
5     int ind = 0;  
6     for(int x = 0; x < K; ++x)  
7         for (int j = 0; j < C[x]; ++j)  
8             A[ind++] = x;  
9 }
```


Обзор



Санкт-Петербургский
государственный
университет

- ① Случайные числа и время
- ② Сортировка Шелла
- ③ Сортировка подсчетом
- ④ Задачи
 - Сортировка Шелла (оптимизация Пратта)
 - Запросы на отрезке

Сортировка Шелла (оптимизация Пратта)



Санкт-Петербургский
государственный
университет

- ▶ Реализуйте сортировку Шелла на последовательности интервалов 3-гладких чисел (числа вида $2^p 3^q$, $p, q \geq 0$). [Последовательность 3-гладких чисел](#) в OEIS.
- ▶ Сравните время работы вашей сортировки со временем работы `std::sort` из `<algorithm>` при $N = 10^3, 10^4, 10^5, 10^6$.
- ▶ Массив инициализировать случайными 32-битными числами с помощью `std::mt19937`.

Запросы на отрезке



Санкт-Петербургский
государственный
университет

- ▶ Вам дан массив из N элементов, где $0 \leq a[i] \leq K - 1$.
- ▶ Необходимо ответить на Q запросов, сколько чисел из вашего массива принадлежат отрезку $[l \dots r]$, каждый запрос за время $O(1)$.
- ▶ Алгоритм должен выполняться за время $O(N + K + Q)$.