Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет цифровых трансформаций

Дисциплина:
«Алгоритмы и структуры данных»

Практическая работа №5

«Сортировки»

Выполнила:
Абаянцева Е. Ю., студент группы J3110

Санкт-Петербург
2024 г.

1. Память сортировки Odd-Even: $O(n^2)$.
   Память сортировки Merge Sort: $O(n)$.
   Память сортировки Counting Sort: $O(k)$.

2. Сложность сортировки Odd-Even Sort: $O(n) = n^2$.
   Сложность сортировки Merge Sort: $O(n) = n \log n$.
   Сложность сортировки Counting Sort: $O(n) = n + k$.

3. Код сортировок:

   (a) Odd-Even Sort:

```
33  std::vector<int> Odd_Even_Sort( std::vector<int>& mas, int& coun ) {
34      int n = mas.size();
35      int isSorted = 0;
36      coun += sizeof(n) + sizeof(isSorted);
37
38      while (isSorted == 0) {
39          isSorted = 1;
40          for (int i = 1; i < n - 1; i += 2) {
41              if (mas[i] > mas[i + 1]) {
42                  std::swap(mas[i], mas[i + 1]);
43                  isSorted = 0;
44              }
45          }
46          for (int i = 0; i < n - 1; i += 2) {
47              if (mas[i] > mas[i + 1]) {
48                  std::swap(mas[i], mas[i + 1]);
49                  isSorted = 0;
50              }
51          }
52      }
53
54      // coun = sizeof(n) + sizeof(isSorted);
55      // std::cout << coun << "\n";
56
57      return mas;
58  }
```

Рис. 1: Odd-Even Sort

   (b) Merge Sort:

```
66      std::vector<int> lmas(mas.begin(), mas.begin() + mas.size() / 2);
67      std::vector<int> rmas(mas.begin() + mas.size() / 2, mas.end());
68      coun += sizeof(lmas) + sizeof(rmas);
69
70      lmas = Merge_Sort(lmas, coun);
71      rmas = Merge_Sort(rmas, coun);
72
73      int n = 0, m = 0, k = 0;
74      coun += sizeof(n) + sizeof(m) + sizeof(k);
75      std::vector<int> pob_mas(mas.size());
76      coun += sizeof(pob_mas);
77
78      while (n < lmas.size() && m < rmas.size()) {
79          if (lmas[n] <= rmas[m]) {
80              pob_mas[k] = lmas[n];
81              n++;
82          }
83          else {
84              pob_mas[k] = rmas[m];
85              m++;
86          }
87          k++;
88      }
89
90      while (n < lmas.size()) {
91          pob_mas[k] = lmas[n];
92          n++;
93          k++;
94      }
95
96      while (m < rmas.size()) {
97          pob_mas[k] = rmas[m];
98          m++;
99          k++;
100     }
```

Рис. 2: Merge Sort

(c) Counting Sort:

```cpp
std::vector<int> Counting_Sort( const std::vector<int>& mas3, int& coun ) {
    int maxElement = *std::max_element(mas3.begin(), mas3.end());
    int countArrayLength = maxElement + 1;
    coun += sizeof(maxElement) + sizeof(countArrayLength);

    std::vector<int> countArray(countArrayLength, 0);
    coun += sizeof(countArray);

    for (int el : mas3) {
        countArray[el]++;
    }

    for (int i = 1; i < countArrayLength; i++) {
        countArray[i] += countArray[i - 1];
    }

    std::vector<int> outputArray(mas3.size());
    int i = mas3.size() - 1;
    coun += sizeof(outputArray) + sizeof(i);
    while (i >= 0) {
        int currentEl = mas3[i];
        countArray[currentEl]--;
        int newPosition = countArray[currentEl];
        outputArray[newPosition] = currentEl;
        i--;
    }

    return outputArray;
}
```

Рис. 3: Counting Sort

4. Лучшие случаи для каждой сортировки:
   (a) Odd-Even Sort: когда при первой пробежке весь массив отсортируется. Сложность – $O(n)$.
   (b) Merge Sort: когда все элементы списка уже отсортированы. Сложность – $O(n \cdot \log n)$.
   (c) Counting Sort: когда все элементы массива равны. Сложность – $O(n)$.

   Худшие и средние случаи для каждой сортировки:
   (a) Odd-Even Sort: когда одной пробежки по массиву недостаточно. Сложность – $O(n^2)$.
   (b) Merge Sort: когда элементы не отсортированы по массиву. Сложность – $O(n \cdot \log n)$.
   (c) Counting Sort: когда элементы различны и большие расходы в значениях (тогда выделяется больший массив). Сложность – $O(n+k)$.

5. Асимптотика всех сортировок:

(a) Odd Even Sort:

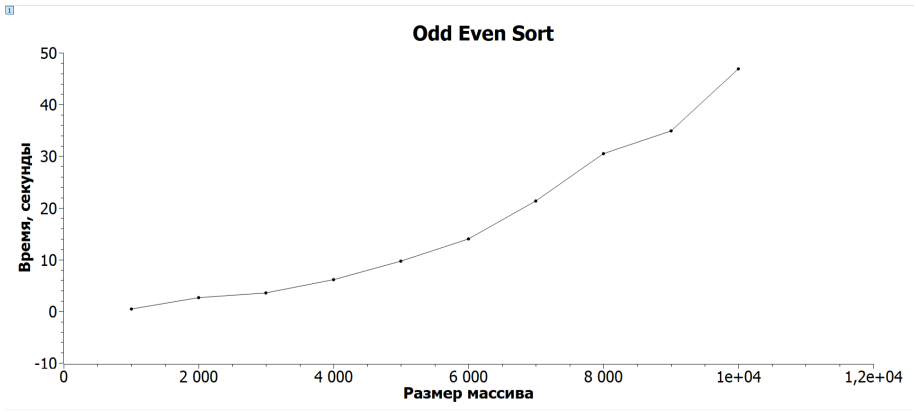

Рис. 4: Odd Even Sort

(b) Merge Sort:

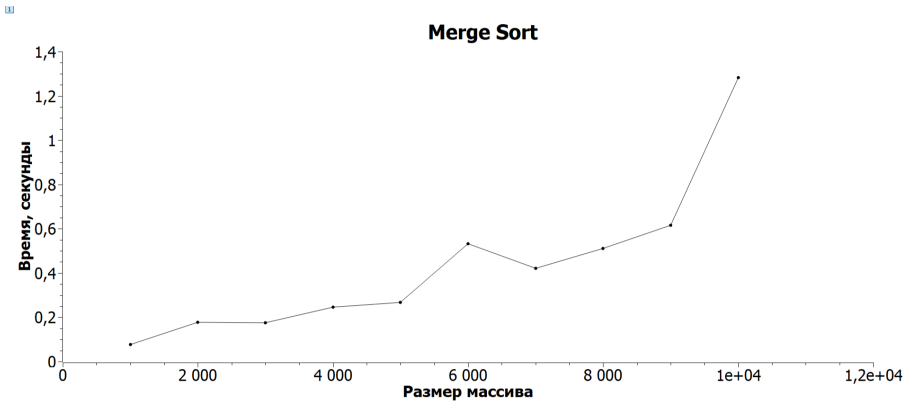

Рис. 5: Merge Sort

(c) Counting Sort:

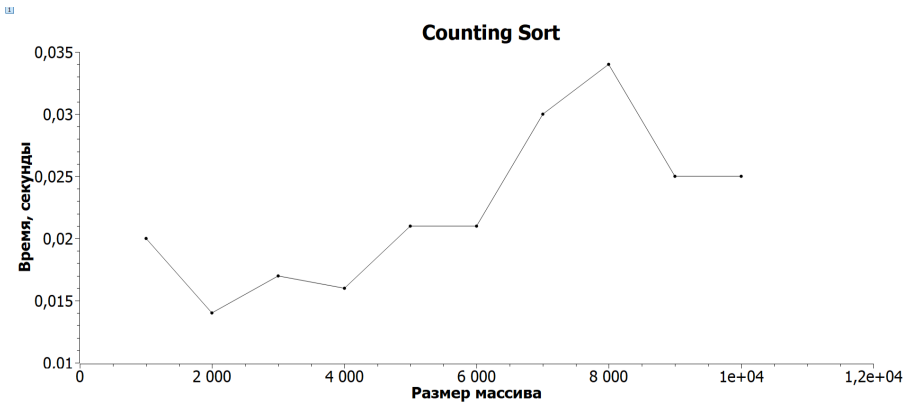

Рис. 6: тест 3

(d) Все сортировки:

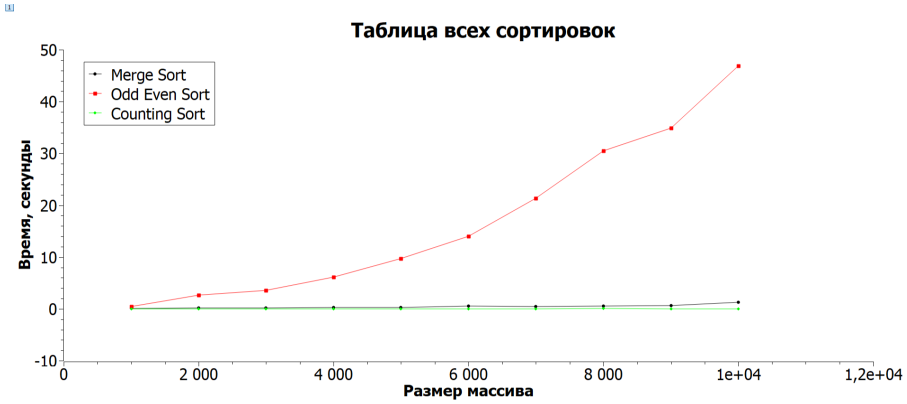

Рис. 7: все сортировки

6. Вывод: когда у нас полностью неотсортирован массив натуральных чисел и имеются различные расходы в них, то лучше использовать Counting Sort. Если всё более-менее рассортировано, то лучше использовать Odd-Even Sort или Merge Sort. Сортировку Counting Sort лучше использовать остальных, когда мы работаем с массивом натуральных чисел. Odd-Even Sort – если нет.