ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ МОЛДОВЫ

ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАТИКИ
СПЕЦИАЛЬНОСТЬ: ПРИКЛАДНАЯ ИНФОРМАТИКА

Студент: Фомин Егор

ИНДИВИДУАЛЬНАЯ РАБОТА №2

По курсу: Алгоритмы и Структуры Данных
На тему: Методы сортировки

Цикл 1 – лицензия

Кишинев, 2023

Оглавление

Введение

В современном информационном обществе обработка и управление данными являются ключевыми аспектами, определяющими эффективность программных систем. Одним из фундаментальных этапов работы с данными является их сортировка. Методы сортировки представляют собой важный инструмент в области алгоритмов и структур данных, играя ключевую роль в решении разнообразных задач, начиная от оптимизации поиска до управления базами данных.

Целью данной индивидуальной работы является исследование различных методов сортировки, их алгоритмов, анализ эффективности и областей применения. В ходе работы будут рассмотрены такие важные аспекты, как оценка сложности алгоритмов, а также сравнение методов сортировки по их производительности.

Задание

Требуется реализовать программу на любом удобном вам языке программирования, в которой реализованы методы сортировки массивов данных. Чем больше методов, тем выше оценка. Для каждого метода сортировки проанализировать теоретическую и практическую сложности.

В конечной программе необходимо включить комментарии, объясняющие ваш подход и использованные алгоритмы. Программу следует прикрепить к заданию на moodle и продемонстрировать преподавателю на лабораторном занятии.

Для каждого алгоритма вывести следующую информацию:

- теоретическая оценка сложности
- количество сравнений
- количество перестановок
- время выполнения алгоритма

Методы сортировки:

1. Сортировка пузырьком (Bubble Sort):
2. Быстрая сортировка (Quick Sort)
3. Сортировка вставками (Insertion Sort):
4. Сортировка слиянием (Merge Sort)
5. Сортировка выбором (Selection Sort):

Решение

Выбранные методы сортировки:

- Сортировка пузырьком
- Сортировка вставки
- Сортировка выбора

Выбранный язык программирования: ***JavaScript***

Ссылка для клонирования репозитория:

https://github.com/Tcyiu/ASD-Individual-2.git

```
// ! Custom class
class MyClass {
  constructor(id, name, age, salary, isActive) {
      this.id = id;
      this.name = name;
      this.age = age;
      this.salary = salary;
      this.isActive = isActive;
  }
}

// ! Random number between args generator
const getRandomNumber = (arg1, arg2) => {
  if (arg1 > arg2) {
      [arg1, arg2] = [arg2, arg1];
  }
  return Math.floor(Math.random() * (arg2 - arg1 + 1)) + arg1;
}

// ! Unique id generator
const getUniqueId = () => {
  return Math.random().toString().substr(2, 3);
}

// ! Array with 50 instances of MyClass
const objectsArray = [];
```

```
for (let i = 0; i < 50; i++) {
    const obj = new MyClass(getUniqueId(), `Name ${i}`, getRandomNumber(18, 65),
getRandomNumber(500, 5000), i % 2 === 0);
    objectsArray.push(obj);
}
```

Пузырьковая сортировка

```
const bubbleSort = (arr) => {
  const temp = [...arr];
  const n = temp.length;
  let comparisons = 0;
  let swaps = 0;

  for (let i = 0; i < n - 1; i++) {
      for (let j = 0; j < n - i - 1; j++) {
          comparisons++;
          if (temp[j].id > temp[j + 1].id) {
              swaps++;
              // ! Swap elements
              [temp[j], temp[j + 1]] = [temp[j + 1], temp[j]];
          }
      }
  }

  return { comparisons, swaps };
}

// ? Theoretical complexity estimation: O(n^2)
// ? Number of comparisons: In the worst and average case - O(n^2), in the best case - O(n)
// ? Number of swaps: In the worst and average case - O(n^2), in the best case - 0
// ? Execution time: Compared to other algorithms, bubble sort is inefficient. The execution
time is high, especially on large datasets.

console.group('Bubble sort:')
console.time('Time')
console.log('Result:', bubbleSort(objectsArray));
console.timeEnd('Time')
console.groupEnd();
```

Сортировка вставки

```
const insertionSort = (arr) => {
  const temp = [...arr];
  const n = temp.length;
  let comparisons = 0;
  let swaps = 0;

  for (let i = 1; i < n; i++) {
    const key = temp[i];
    let j = i - 1;

    comparisons++;
```

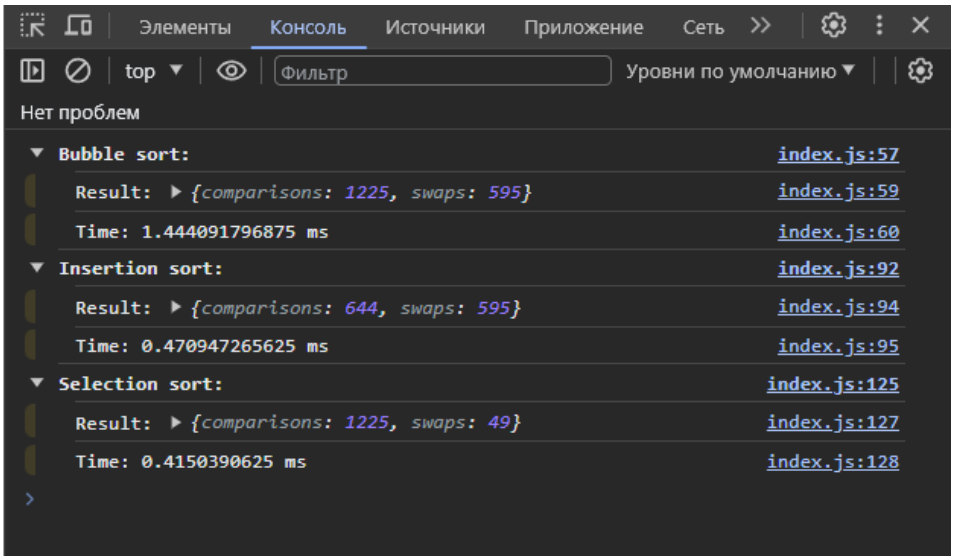
```
    while (j >= 0 && temp[j].id > key.id) {
        comparisons++;
        swaps++;
        temp[j + 1] = temp[j];
        j = j - 1;
    }

    temp[j + 1] = key;
  }

  return { comparisons, swaps };
}

// ? Theoretical complexity estimation: O(n^2) in the worst case, O(n) in the best case
// ? Number of comparisons: In the worst and average case - O(n^2), in the best case - O(n)
// ? Number of swaps: In the worst and average case - O(n^2), in the best case - 0
// ? Execution time: Efficient for small arrays.

console.group('Insertion sort:')
console.time('Time')
console.log('Result:', insertionSort(objectsArray));
console.timeEnd('Time')
console.groupEnd();
```

Сортировка выбора

```
function selectionSort(arr) {
  const temp = [...arr];
  const n = temp.length;
  let comparisons = 0;
  let swaps = 0;

  for (let i = 0; i < n - 1; i++) {
    let minIndex = i;

    for (let j = i + 1; j < n; j++) {
        comparisons++;
        if (temp[j].id < temp[minIndex].id) {
            minIndex = j;
        }
    }

    swaps++;
    [temp[i], temp[minIndex]] = [temp[minIndex], temp[i]];
  }
  return { comparisons, swaps };
}

// ? Theoretical complexity estimation: O(n^2)
// ? Number of comparisons: Always O(n^2)
// ? Number of swaps: Always O(n)
// ? Execution time: Inefficient for large datasets.
```

```
console.group('Selection sort:')
console.time('Time')
console.log('Result:', selectionSort(objectsArray));
console.timeEnd('Time')
console.groupEnd();
```

Результат:



Вывод

В ходе изучения методов сортировки были подробно рассмотрены и проанализированы основные алгоритмы, такие как пузырьковая сортировка, сортировка вставки и сортировка выбора. Сравнительный анализ эффективности позволил выделить преимущества и недостатки каждого метода в различных сценариях.

Библиография

- Knuth, D. E. (1998). "The Art of Computer Programming, Volume 3: Sorting and Searching." Addison-Wesley.
- Sedgewick, R. (2011). "Algorithms, Part I." Princeton University Press.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). "Introduction to Algorithms." The MIT Press.
- Laakmann McDowell, G. (2015). "Cracking the Coding Interview: 150 Programming Questions and Solutions." CareerCup.

- Bentley, J. (1987). "Programming Pearls." Communications of the ACM, 30(6), 484-495.