

Data Structures and Algorithm

Assignment #7-8

Name: Tuana Melisa Aksoi

No: 230104004903

1. Introduction

In this project it was implemented the graph coloring system with multiple sorting strategies.

1.1 Graph Implementation

- Developed a graph data structure using an adjacency matrix representation
- Created a specialized `AdjacencyVect` class for efficient vertex connection storage. And implemented the all methods for the **`AdjacencyVect`** class (Bonus).
- Implemented graph operations including edge addition, connection verification, and neighbor retrieval

1.2 Sorting Algorithms

Implemented three fundamental sorting algorithms:

1. Insertion Sort

- Insertion Sort builds the sorted array one element at a time by placing each new element in its correct position among previously sorted elements.

2. Selection Sort (Bonus)

- Selection Sort repeatedly finds the minimum element from the unsorted portion and places it at the beginning of the sorted portion.

3. Quick Sort

- The Quick Sort implementation in this project uses a randomized pivot selection strategy with hybrid optimization. The algorithm works by selecting a random pivot element from the array, partitioning the array around this pivot, and recursively sorting the resulting sub-arrays. For small partitions (10 elements or less), it switches to either Insertion Sort or Selection Sort the main reason for that by switching to these algorithms for small subarrays, the overall sorting process becomes faster and more efficient.

2. Environment

The program was tested on macOS with Java version 24.0.1, as well as within the provided Docker environment. In both cases, the program worked successfully without any bugs.

3. Complexity Analysis

3.1. Insertion Sort Complexity

- Best case $O(n)$ when array is nearly sorted
- Worst case $O(n^2)$ when array is reverse sorted

3.2. Selection Sort

- All cases $O(n^2)$

3.3. Quick Sort

- Average/Best case: $O(n \log n)$
- Worst case $O(n^2)$ (rare, with poor pivot choices)

3.4. MatrixGraph Class

- addEdge / setEdge: $O(1)$
- getEdge: $O(1)$
- getNeighbors: $O(1)$
- reset: $O(n)$
- size: $O(1)$

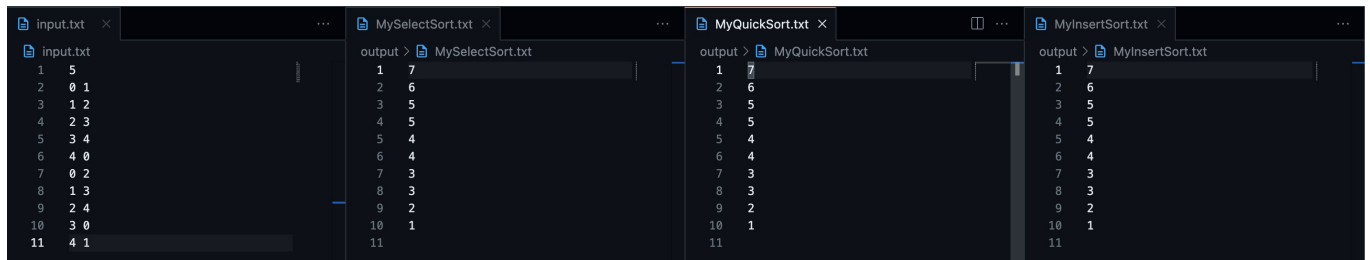
3.5. AdjacencyVect class

- add : $O(1)$
- addAll / removeAll / containsAll: $O(m)$
- clear / retainAll / toArray / remove / contains: $O(n)$
- size / isEmpty: $O(1)$
- iterator (hasNext/next): $O(n)$ worst case

4. Testing

I created the input.txt file and the output folder and runned the program using the given commands:

- make clear
- make collect
- make build
- make run ARGS="input.txt output/"

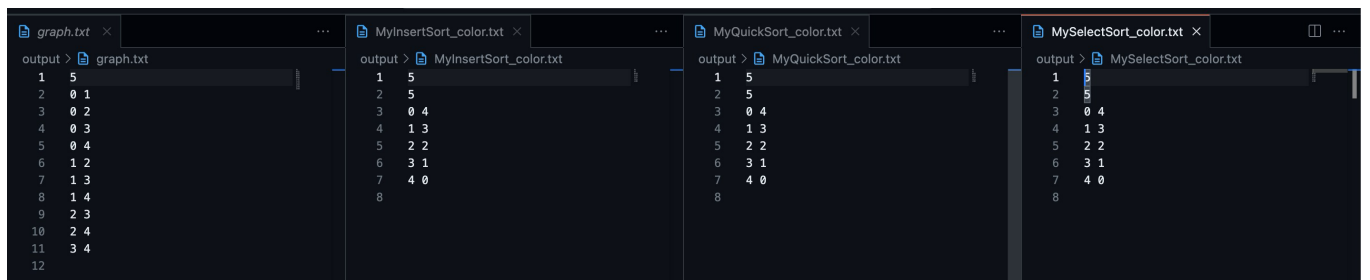


```
input.txt
1 5
2 0 1
3 1 2
4 2 3
5 3 4
6 4 0
7 0 2
8 1 3
9 2 4
10 3 0
11 4 1

MySelectSort.txt
1 7
2 6
3 5
4 5
5 4
6 4
7 3
8 3
9 2
10 1
11

MyQuickSort.txt
1 7
2 6
3 5
4 5
5 4
6 4
7 3
8 3
9 2
10 1
11

MyInsertSort.txt
1 7
2 6
3 5
4 5
5 4
6 4
7 3
8 3
9 2
10 1
11
```

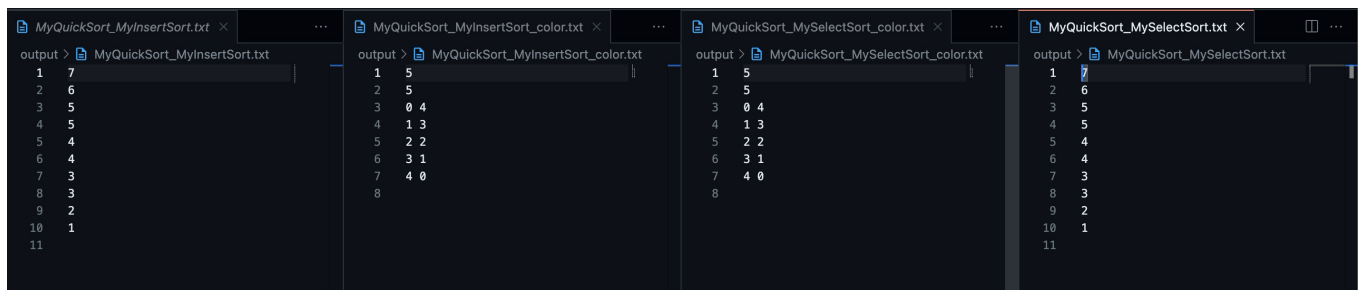


```
graph.txt
1 5
2 0 1
3 0 2
4 0 3
5 0 4
6 1 2
7 1 3
8 1 4
9 2 3
10 2 4
11 3 4
12

MyInsertSort_color.txt
1 5
2 5
3 0 4
4 1 3
5 2 2
6 3 1
7 4 0
8

MyQuickSort_color.txt
1 5
2 5
3 0 4
4 1 3
5 2 2
6 3 1
7 4 0
8

MySelectSort_color.txt
1 5
2 5
3 0 4
4 1 3
5 2 2
6 3 1
7 4 0
8
```



```
MyQuickSort_MyInsertSort.txt
1 7
2 6
3 5
4 5
5 4
6 4
7 3
8 3
9 2
10 1
11

MyQuickSort_MyInsertSort_color.txt
1 5
2 5
3 0 4
4 1 3
5 2 2
6 3 1
7 4 0
8

MyQuickSort_MySelectSort_color.txt
1 5
2 5
3 0 4
4 1 3
5 2 2
6 3 1
7 4 0
8

MyQuickSort_MySelectSort.txt
1 7
2 6
3 5
4 5
5 4
6 4
7 3
8 3
9 2
10 1
11
```

Also as the bonus I did the unit testing in the docker environment and there is the results

```
gtu@0ee968515c22:~/submission$ make clean
rm -rf build sources.txt doc
gtu@0ee968515c22:~/submission$ make collect
find src -name "*.java" > sources.txt
gtu@0ee968515c22:~/submission$ make build
find src -name "*.java" > sources.txt
javac -d build @sources.txt
gtu@0ee968515c22:~/submission$ make run ARGS="input.txt output/"
java -cp build Main.MyTests input.txt output/
Sorter tests passed
MatrixGraph tests passed
AdjacencyVect tests passed
All tests passed successfully!
gtu@0ee968515c22:~/submission$
```

As the result the sortings, martixGraph and the adjacencyVect class was runned successfully in the docker environment.

5. AI usage

- AI was used to help understand and explain algorithmic concepts, such as sorting algorithms and graphs.
- Provide some input and output testing.
- Helped writing the MyTest.java

6. Challenges

- Implementing the hybrid Quick Sort (switching to Insertion or Selection Sort for small partitions)

7. Conclusion

- This project helped to understand how to combine di^erent sorting algorithms. I learned how algorithm choices a^ect performance and how to make my code work well in di^erent environments.
- The all methods from the AdjacenyVect was implemented, and the selection sort was implemented as a bonuses.
- The program was runned successfully.