Assignment 2
Algorithms and Data Structures 1 (1DL210)
2021

Rishi Sudhan Venkata Subramanian

October 7, 2021

Instructions

- Unless you have asked for and received special permission, solutions must be prepared in groups. If you got special permission for the first assignment, you also have special permission for this assignment.
- Make sure that your programs run on the UNIX/LINUX system. Any (sub)solutions that don't run on the UNIX/LINUX system **will** receive 0 points. In particular, you should verify the output of your implementation against the output of the UNIX/LINUX sorting command line `sort -n`.
- When you have completed the assignment, you should have four files: three sorting programs and one Pdf document explaining the differences between them. Make a `.zip` file containing these four files.
- Submit the resulting `.zip` file to Studium. Only **one** of you from each group needs to do this.

Implementing Sorting Algorithms

For this assignment, you are going to implement three different sorting algorithms based on their textbook descriptions that you can find in the lecture slides.

- Each algorithm is to be implemented in separate files.
- The file sort.py contains skeleton code to run your algorithm. Call your sorting algorithm inside respective functions of sort.py. For example, bubblesort is implemented and called at line 32 inside `run_bubblesort()` function.
- The program should read a file called `nums.txt`, which can be assumed to contain integers separated by newlines.
- `rangen.py`(or `rangen_py3x.py` for python 3x) takes an integer argument n and outputs a file `nums.txt` containing n rows with random numbers in the range $\{0, \ldots, n\}$. Example of usage (on python **3.x**): `python rangen_py3x.py 100`
- Output of each algorithm is stored in a file. For example, for insertion sort the file is `insertionsorted.txt`.

- Additionally, you should verify that your result matches the output of the `sort -n` UNIX/LINUX system command. Running`test.sh` will do for you.
- If you are not implementing your algorithm in python or not using the python skeleton code, make sure that your program behaves similarly. In this case, you probably need to re-implement yourself a `run()` function to run your algorithms.
- Your code should produce output files with the names `insertionsorted.txt,` `quicksorted.txt,` and `heapsorted.txt`. Each of these files should contain output sequence as integers separated by newlines.

Note that the arrays in the lecture and in the textbook are indexed from 1, whereas in most programming languages they are indexed from 0.

Insertion Sort (3p)

Implement INSERTION-SORT from the second lecture.

Quicksort (3p)

Start by implementing PARTITION. Then use it to implement QUICKSORT from the fourth lecture.

Heapsort (3p)

Implement the functions MAX-HEAPIFY, BUILD-MAX-HEAP, and HEAPSORT from the fifth lecture.

Comparison (1p)

What are the differences between the three algorithms? For each algorithm, mention a situation where it has an advantage over the other two.
Insertion sort - The main advantage of the insertion sort is its simplicity. It also exhibits a good performance when dealing with a small list.
Quick sort- The quick sort is regarded as the best sorting algorithm. This is because of its significant advantage in terms of efficiency because it is able to deal well with a huge list of items. Because it sorts in place,no additional storage is required as well.
Heap sort- Heap Sort in Data Structure is used when the smallest or highest value is needed instantly. Optimized performance, efficiency, and accuracy are a few of the best qualities of this algorithm. The algorithm is also highly consistent with very low memory usage.