Algorithm Comparator

A time based comparator application for different algorithms

Raj Pastagiya (22M2111)

Chaitanya Shravan Borkar (22M0810)

September 29, 2022

1 Background

We have seen many algorithms in our study in Computer Science and we have also theoretically compared them based on time complexity and space complexity. But as a Computer Science student we may also want to know how exactly those comparisons work on a real life computer system. While theoretical understanding may give approximate answers to our concerns in real life we may or may not use the same theoretical outcomes as there are many factors involved especially in calculating execution time of the algorithm on a certain data set.

2 The Algorithm Comparator

2.1 Problem Statement

Given practical implementation using programming language of the algorithm we want to compare the execution time on different type of data sets.

2.2 Tools and Technologies

- ***GitHub*** *for software management and version control*
- ***Eclipse IDE*** *as project development environment*
- ***Java*** *with* **Spring Framework** *for creating back-end APIs and supporting back-end logic*
- ***HTML, CSS*** *for supporting front-end logic of the application*
- ***Latex*** *for documenting the project*

2.3 Application Level Overview

Given a set of algorithms as selection options on front-end, user can choose 2 algorithms to compare with each other. The Application then will run these algorithms on a diverse type of data sets and note the time taken to execute certain process, sort the data set if the algorithm is sorting based. The required data then is given to front-end where front-end then will represent these data in a graphical form to show how the trend goes with different type of data sets. Below are different aspects of the application that defines the implementation of the application.

- ***Logical Algorithm Pools:***
  - *The predefined algorithms will come with the application which will be divided in some logical pools and the comparison can only be done between the algorithms in a specific pool.*
  - *Below are the predefined pools and algorithms that come with the application;*

1. ***Sorting Algorithms***
    1. *(a) Selection Sort*
    2. *(b) Merge Sort*
    3. *(c) Bucket Sort*
2. ***Searching Algorithms***
    1. *(a) Linear Search*
    2. *(b) Binary Search*
3. ***Knap-sack Algorithms***
    1. *(a) Brute Force Approach*
    2. *(b) Greedy Approach*
    3. *(c) Dynamic Approach*

- ***Data-Set Pools:***
    - *User will also be given choice of data-sets, which could be one of the following;*
        1. *Random unique numbers of various taste: Positive, Negative, Whole Numbers*
        2. *Random repeated numbers of various taste: Positive, Negative, Whole Numbers*
        3. *Sorted unique numbers of various taste: Positive, Negative, Whole Numbers*
        4. *Sorted repeated numbers of various taste: Positive, Negative, Whole Numbers*
- ***User's Perspective:***
    - ***User View:*** *User will be presented with different types of algorithms as options predefined under a logical comparison pool.*
    - ***Algorithm and Data set Selection:*** *User can select 2 algorithms to compare to within a single pool. User can also select data set and then submit it as a query.*
    - ***Graphical View:*** *Once submitted the query the user will be presented with a graph showing live comparison of time over different data sets.*
- ***Backend Implementations:***
    - ***Storage:*** *Backend will store the implementation of algorithms specified in above algorithm pools and some predefined data-sets (excluding random data-sets) in different files in an organized manner. The implementation of these algorithms will be done using with the reference of implementation done on well known open-source websites like GeeksforGeeks*
    - ***Request processing:*** *Given a specific query containing selected algorithms and data-set backend will use multi-threading approach to calculate execution time.*
    - ***Data collection to create graph:*** *For live graph creation either frontend will poll to an open API at certain period of time or one can use websocket approach to send live data containing data-set size and time taken for execution for specific algorithm to frontend.*
    - ***Caching:*** *Backend will also use caching mechanisms by using unique request id for each requests so it doesn't have to repeat same process for same type requests.*

2.4 Future Implementation(s)

A user can upload an algorithm written in some language listed on the application and for future comparisons user can then select uploaded algorithm with already available algorithms to see the execution trend on data sets.