Virtual Labs for Merge Sort - Requirements Specification

Venkatesh Choppella, Ojas Mohril

March 16, 2021

Contents

1 Introduction

1.1 Purpose of the Document

This is a Requirements Specification document for a *virtual lab* on the topic of "Merge Sort". This document describes the objectives of the lab and all the experiments to be included in the lab. This document is intended to direct the design and implementation of the target lab.

1.2 Lab Summary

Name Merge Sort

Domain Computer Science and Engineering

Objective The objective this lab is to demonstrate the process of sorting an array of values using the merge sort algorithm. After performing the experiments in this lab, the learner is expected to -

- Have a clear understanding of the formulation of the sorting problem.
- Identify merge-sort as one of many sorting algorithms.
- Understand the basic operations involved in applying the merge sort algorithm. The user should be able to understand :
  - the steps involved in a merge operation.
  - how the split and merge operations are used to sort an array.

Scope of the Sorting As the basic concept of comparison remains the same irrespective of the type items; for the experiments in this lab, the sorting problem is limited to searching for numbers (Integers).

Size of the list 5 (min) to 10 (max).

- Limiting the size of the list allows us to keep the interface manageable.

Range of Values Non-negative integers in the range of 0 (min) to 100 (max).

Target Audience Undergraduate students in second or third year of a B. Tech / B. E. degree in Computer Science and Engineering (or similar disciplines).

Number of Experiments 5

2 Interface Requirements

The user interface should have the following elements:

1. **Navigation** Links to all the experiments.
2. **Experiment Details** **Objective** Learning objectives of the experiment.

   Experiment Setup Description of the experiment interface and how to interact with the various elements.

   Instructions Steps-by-step guide for the user to perform the experiment.
3. **Interactive Experiment** The actual setup that presents the users with various interactive elements and allows them to perform the experiment and gives useful feedback at each step.

2.1 Experiment Interface

The experiment interface should provide the following:

2.1.1 Elements

Unsorted Array List of items that acts as the search space.

Sorting Indicator A visual indicator that tells the user that the sorting is complete.

Recursion Graph Experiments involving demonstration of merge-sort algorithm need to provide an interactive Directed Acyclic Graph that allows the user to perform the recursion step-by-step.

Control buttons Buttons that allow the user to perform required operations.

Prompt The system should give a feedback message to the user after each interaction. This feedback indicates the result of the latest action and hints towards the next possible actions.

3 Experiments

3.1 Shuffle

3.1.1 Objective

The objective of this experiment is to introduce the concept of "shuffling" two arrays to make a larger array.

3.1.2 Learning Outcomes

The user should be able to learn the following after performing this experiment:

1. **The basic operation of shuffling** The user should be able to identify that in order to shuffle two arrays the items in the array need to comparable and that repeated application of shuffling items from two arrays, leads to a larger array containing items from both sub-arrays.
2. **Preconditions for shuffling** There is no precondition to shuffling. Any two arrays can be shuffled together to form a new array.
3. **Properties of resulting array** The number of items in the resulting array is equal to the sum of items in both the sub-arrays.

3.2 Merge Unsorted Lists

3.2.1 Objective

The objective of this experiment is to demonstrate the "merge" operation. Performing this experiment, the user learn step-by-step how to merge two lists.

3.2.2 Learning Outcomes

The user should be able to learn the following after performing this experiment:

1. **Merging is a special kind of shuffling operation** The user should be able to identify the fact that merging is a constrained form of shuffling and that the same result can be achieved by using the shuffling operation if the correct strategy is followed.
2. **Properties of resulting array** The user should be able to learn that the resulting array is not guaranteed to be sorted.

3.3 Merge Sorted Lists

3.3.1 Objective

The objective of this experiment is to step-by-step demonstrate the merging two sorted arrays and to show that the resulting array is always sorted.

3.3.2 Learning Outcomes

The user should be able to learn the following after performing this experiment:

- 1. **Merging two sorted arrays results in a larger sorted array.**

3.4 Recursive Merge-sort

3.4.1 Objective

The objective of this experiment is to demonstrate the recursive merge-sort algorithm.

3.4.2 Learning Outcomes

The user should be able to learn the following after performing this experiment:

1. **Basic recursive sub-structure** The user should be able to identify the recursive sub-structure and the base-case of the merge-sort algorithm. The two basic operations "split" and "merge" should be clearly demonstrated.

3.5 Recursive Arbitrary Merge-sort

3.5.1 Objective

The objective of this experiment is to demonstrate that the merge-sort recursive sub-structure can be solved in several ways by arbitrarily picking the nodes to split and merge.

3.5.2 Learning Outcomes

The user should be able to learn the following after performing this experiment:

1. **Any order of split-and-merge leads to sorting** The user should be able to try different combinations of split and merge operations and observe that the end result is always a sorted version of the original array.

2. **Merge-sort Algorithm automates the process** The merge-sort algorithm automates the process of sorting by deciding the order of split and merge operations.