Australian National University

Divide-and-Conquer and Dynamic Programming

Assignment

Tasks for this week

- Implement Merge sort (**MergeSort** class, **mergeSort** and **merge** methods)
- Implement Karatsuba multiplication (**MultiplicationAlgorithm** class and **KMultiply** method)
- Implement Edit Distance computation (**EditDistance** class and **minDistance** method)

- The code structure/skeleton is available on Wattle

- Submission Guidelines
  - The last slide contains information about the submission
  - Read it carefully to avoid losing marks!

Task 1 – Merge Sort (1 mark)

- Implement the following methods in the `MergeSort.java`:

1. Method **mergeSort()**

   This method uses divide-and-conquer to sort an array. It must use merge() method

2. Method **merge()**

   This method merges two sorted subarrays

Reference: Lecture 5 Algorithms Part I, slide 20

How to test your code?

Use the `MergeSortTest.java` file to check whether your implementation passes the test cases or not. It may be used as an indicator that your code is working correctly. Please be aware that we will use additional test cases to verify and assess your code.

What is tracker class?

The `tracker.java` is used for marking purpose to track divide-and-conquer calls. Do not modify any code associated with the tracker. Otherwise, you will be penalized for violation.

Task 2 – Karatsuba Multiplication (1 mark)

- Implement the following method in the `MultiplicationAlgorithm.java`:

1. Method KMultiply()

   This method uses Karatsuba multiplication to compute the product x times y.

   x and y are two n-digit input numbers

Reference: Lecture 5 Algorithms Part I, slide 49

How to test your code?

Use the `KMultiplyTest.java` file to check whether your implementation passes the test cases or not. It may be used as an indicator that your code is working correctly. Please be aware that we will use additional test cases to verify and assess your code.

What is tracker class?

The `tracker.java` is used for marking purpose to track divide-and-conquer calls. Do not modify any code associated with the tracker. Otherwise, you will be penalized for violation.

Task 3 – Edit Distance (1 mark)

- Implement the following method in the **`EditDistance.java`**:

1. Method **minDistance()**

   This method computes the minimal total cost of a sequence of character edits between two strings.

   The costs of character edits are defined in EditCost enum.

   Do not modify the character edit costs. Otherwise, your answers will not be marked correctly.

Reference: Lecture 7 Algorithms Part III, slides 10-14

How to test your code?

Use the **`EditDistanceTest.java`** file to check whether your implementation passes the test cases or not. It may be used as an indicator that your code is working correctly. Please be aware that we will use additional test cases to verify and assess your code.

Submission Guidelines

- **Assignment deadline: see the deadline on Wattle (always!)**
- Submission mode: via Wattle (Assignment)

Submission format (IMPORTANT):

- Upload **only** your final version of:
  `MergeSort.java` (for task 1), `MultiplicationAlgorithm.java` (for task 2) and `EditDistance.java` (for task 3) to Wattle
- Each test case must **run for at most 1000ms**, otherwise it will fail (zero marks).
- **Do not** change the file names
- **Do not** upload any other files (only the specified files are needed)
- **Do not** upload a folder (your submission should be only **three java files**).
- The answers will be marked by an automated marker.
  - **Do not** change the structure of the source code including class name, package structure, etc.
  - **You are only allowed to edit the designated code segment indicated in the comments.**
- **Do not** import packages outside of the standard java SE package. The list of available packages can be found here:
  https://docs.oracle.com/en/java/javase/12/docs/api/index.html
- Any violation of the submission format will result in zero marks