Tutorial 5

Introduction

In this tutorial, students are asked to implement the sorting algorithms including Quick sort, Heap sort, Counting sort, Radix sort. Student may refer to the example section to see the implementation code of 4 mentioned sorting algorithms. Class SortComparison2 implements all sorting algorithms together (4 in week 4 plus Quick sort and Heap sort) and demonstrates the running time of each algorithm.

In the exercise sections, students will try to use sorting algorithm to solve several related problems.

Examples

1. Example 01 – Quick sort.

Student should create a class that implements the Quick sort algorithm, with the following functions (please refer to class ***QuickSort*** in the tutorial source code project):

- Ask user to input n, array A using the keyboard
- Sort the array A using Quick sort algorithm
- Show the result

2. Example 02 – Heap sort

Student should create a class that implements the Insertion sort algorithm, with the following functions (please refer to class ***HeapSort*** in the tutorial source code project):

- Ask user to input n, array A using the keyboard
- Sort the array A using Heap sort algorithm
- Show the result

3. Example 03 – Counting sort

Student should create a class that implements the Counting sort algorithm, with the following functions (please refer to class ***CountingSort*** in the tutorial source code project):

- Ask user to input n, array A using the keyboard (each element of A is in the range [1..k])
- Sort the array A using Counting sort algorithm
- Show the result

4. Example 04 – Radix sort

Student should create a class that implements the Radix sort algorithm, with the following functions (please refer to class ***RadixSort*** in the tutorial source code project):

- Ask user to input n, array A using the keyboard (each element of A has maximum k digits)

- Sort the array A using Radix sort algorithm
- Show the result

5. Example 05 – Sorting algorithm comparison

Student should improve the class from last week to implement Selection sort, Insertion sort, Bubble sort, Merge sort, Quick sort, Heap sort algorithm, with the following functions (please refer to class ***SortComparison2*** in the tutorial source code project):

- Randomly generated an array of the size N, copy the generated array to a new array
- Sort the new array using 6 sorting algorithms. Every time, before doing sorting, you need to redo the copying from the source array to make sure that you don't sort an already sorted array.
- Measure and show the running time of each sorting algorithm.

Exercises

1. Exercise 1

Given a list of N students including student's name and student's mark (an integer in the range 0…10). We want to find M students who has the highest marks. Please design an O(n) algorithm to solve this problem and write a Java program to implement your algorithm. Your program should:

- Ask user to input n, list of student including name and mark.
- Ask user to input an integer m
- Show the name of m students who has the highest mark.

2. Exercise 2

Given a list of N strings, all of them have the same length M. We assume that all characters are ASCII code, residing in 256 positions of the character set (from code 0 to code 255). We want to sort this list of strings in the ascending order. Please write a Java program to solve this problem. Your program should:

- Ask user to input n, list of n strings, all of them have the same length m.
- Sort the list of strings in the ascending order.
- Show the result.

3. Exercise 3

Please try to design an algorithm with the time complexity O(n) to solve the problem in Exercise 2.