Algorithm Design and Analysis

Tutorial 2, 2015

Instruction

- You may work in a group of no more than 3 people
- At the end of the period, each group should show the lecturer answers to the questions below to get the mark
- The total mark for this exercise is 30 marks. This exercise is worth 5% of your final grade

Question 1

Implement binary heap data structures as described in class. You need develop an `ArrayHeap` class, which is used to store elements of a generic type `E` that extends the `Comparable` interface. You data structure must adopt the array representation of a complete binary tree taught in class, and support the following methods:

- `void add(E, v)`: Add a new value `v` of type `E` to the heap, if `v` is not already contained in the heap. If `v` is already contained in the heap, then do not add it again.

  Make sure that you implement the `percolateUp` method as discussed in class.

- `E remove()`: Return and remove the element with the smallest value from the heap.

  Make sure that you implement the `pushDown` method as discussed in class.

- `E getMin()`: Return the element with the smallest value in the heap.
- `String toString()`: Return a string representation of the contents of the array.
- `int size()`: Return the current number of elements in the heap.
- `boolean isEmpty()`: Return true if and only if the heap is empty.

Note: You need to create a mechanism to resize the array as you put in more elements to the data structure, as the number of elements added may exceed the bounds of the array. To do that you should use the following algorithm: whenever the current array is full, and an `add` method is called, you should create a new array with double the size, and copy all existing elements from the original array into the new array.
(9 Marks)

Question 2

Use Karatsuba's algorithm to multiply the two binary integers 10011011 and 10111010. Write down your all your steps on a piece of paper.
(5 Marks)

Question 3

Consider the following algorithms:

- Algorithm A solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.
- Algorithm B solves problems by dividing them into 2 subprograms of half the size, recursively solving each subproblem, and then combining the solution in quadratic time.
- Algorithm C solves problems of size n by recursively solving a subproblem of size $n - 1$ with an additional $\log n$ time operation.

What are the running time of each of these algorithm? Show working.
(6 Marks)

Question 4

Implement a Java class `Sort.java` containing implementations of both merge sort and quick sort:

(a) Implement merge sort as described in the lecture slides. You should create a method called `mergeSort(int[] data)`

(b) Implement quick sort as described in the lecture slides. You should create a method called `quickSort(int[] data)`

(c) Measure and compare the running times of both algorithm when running on a number of randomly generated integer arrays of length $n = 1024$. Write down the running times.

(10 Marks)