Analysis of Algorithms: Assignment 3
Due date: January 28 (Thursday)

Problem 1 (5 points)
A $d$-ary heap is like a binary heap, but instead of 2 children, nodes have $d$ children.

(a) How would you represent a $d$-ary heap in an array? What is the height of a $d$-ary heap of $n$ elements in terms of $n$ and $d$?

(b) Give an efficient implementation of HEAP-EXTRACT-MAX for a $d$-ary heap.

(c) Give an efficient implementation of a HEAP-INCREASE-KEY$(A, i, k)$ algorithm, which sets $A[i] \leftarrow \max(A[i], k)$ and updates the heap structure appropriately. Give its time complexity, in terms of $d$ and $n$, and briefly explain your answer.

Problem 2 (5 points)
Consider the following sorting algorithm:

\[
\text{STOOGE-SORT}(A, i, j) \\
\text{if } A[i] > A[j] \\
\hspace{1cm} \text{then exchange } A[i] \leftrightarrow A[j] \\
\text{if } i + 1 \geq j \\
\hspace{1cm} \text{then return} \\
\hspace{1.5cm} k \leftarrow \lfloor (j - i + 1)/3 \rfloor \\
\text{STOOGE-SORT}(A, i, j - k) \\
\text{STOOGE-SORT}(A, i + k, j) \\
\text{STOOGE-SORT}(A, i, j - k) \quad \triangleright \text{First two-thirds.} \\
\triangleright \text{Last two-thirds.} \quad \triangleright \text{First two-thirds again.}
\]

(a) Argue that STOOGE-SORT$(A, 1, n)$ correctly sorts the input array $A[1..n]$.

(b) Give the recurrence for the worst-case running time of STOOGE-SORT and a tight asymptotic ($\Theta$-notation) bound on the worst-case running time.

(c) Compare the worst-case running time of STOOGE-SORT with that of insertion sort, merge-sort, heap-sort, and quick-sort. Is it a good algorithm?

Problem 3 (bonus)
This problem is optional, and it does not affect your grade for the homework; however, if you solve it, then you get 2 bonus points toward your final grade for the course. You cannot submit this bonus problem after the deadline.

We consider an integer array $A[1..n]$ and define a segment sum from $p$ to $r$, where $1 \leq p \leq r \leq n$, as follows:

\[\text{sum}(p, r) = \sum_{p \leq i \leq r} A[i].\]

That is, it is the sum of all array elements in the segment $A[p..r]$. Note that the total number of distinct segments is $\frac{n(n+1)}{2}$. Write a linear-time (that is, $\Theta(n)$) algorithm that determines the maximum over all segment sums.