CSC 225 SPRING 2016
ALGORITHMS AND DATA STRUCTURES I
ASSIGNMENT 2
UNIVERSITY OF VICTORIA

1. Solve the following recurrence equation to get a closed-formula for $T(n)$. Assume the n is a power of three.

$$\begin{aligned} T(n) &= 1 \text{ if } n = 1 \\ &= 2T\left(\frac{n}{3}\right) + n \text{ if } n \geq 2 \end{aligned}$$

2. Solve the following recurrence equation to get a closed-formula for $T(n)$. Assume the n is a power of two.

$$\begin{aligned} T(n) &= 1 \text{ if } n = 1 \\ &= T(n-1) + \log n \text{ if } n \geq 2 \end{aligned}$$

3. Show how Quick-Sort algorithm works on the following input sequence S using the quick-sort tree.

   Use the pivot rule that picks the element in the "middle": For an array $A[0, 1, \ldots, n-1]$ of size n, it uses the element in $A[n/2]$ as pivot if n is even and the element in $A[(n-1)/2]$ as pivot if n is odd [5 Marks].

$$S = [85\ 24\ 63\ 45\ 17\ 31\ 96\ 50]$$

4. In any array A, an *inversion* is a pair of entries that are out of order in A. That is, an inversion is a pair (i, j) such that $i < j$ and $A[i] > A[j]$. Develop an algorithm for computing the number of inversions in a given array by modifying Merge-Sort. The running time of your algorithm should be $O(n \log n)$.

5. Consider an implementation of a stack using an extendible array. That is, instead of giving up with a "StackFullException" when the stack becomes full, we replace the current array S of size N with a larger one of size $f(N)$ and continue processing the push operations. Suppose that we are given two possible choices to increase the size of the array: (1) $f(N) = N + c$ (for convenience, we start with an initial array of size 0) (2) $f(N) = 2N$ (we start with an initial array of size 1). Compare the two strategies and decide which one is better.

   To analyse the two choices, assume the following cost model: A "regular" push operation costs one unit of time. A "special" push operation, when the current stack is full, costs $f(N) + N + 1$ units of time. That is, we assume a cost of $f(N)$ units to create the new array, N units of time to copy the N elements and one unit of time to copy the new element.