COMP 3711H – Fall 2016
Tutorial 4

1. **Using Black-box median algorithms (modified from CLRS)**
For this problem, you assume that you are given a black-box ($O(n)$ time algorithm for finding the median ($\lceil n/2 \rceil$nd) item in a size n array. This means that you can call the algorithm and use its result but can't peer inside of it.

   (a) Show how *Quicksort* can be modified to run in $O(n \log n)$ *worst case* time.

   (b) Give a simple linear-time algorithm that solves the selection problem for an arbitrary order statistic. That is, given k, your algorithm should find the k smallest item.

   (c) For n distinct elements $x_1, x_2, \ldots, x_n$ with associated positive weights $w_1, w_2, \ldots, w_n$ such that $\sum_{i=1}^{n} w_i = 1$, the **weighted (lower) medium** is the element x_k satisfying

$$\sum_{x_i < x_k} w_i < \frac{1}{2} \quad \text{and} \quad \sum_{x_i > x_k} w_i \leq \frac{1}{2}.$$

   If the x_i are sorted, then it is easy to solve this problem in $O(n)$ time by just summing up the weights from left to right and walking through the sums until k is found. Show that if the items are *not* sorted you can still solve the problem in linear time using the black box median finding algorithm.

2. **Polynomial Evaluation** The input to this problem is a set of $n+1$ coefficients $a_0, a_1, \ldots, a_n$. Define $A(x) = \sum_{i=0}^{n} a_i x_i$

   (a) Given value x, how can you evaluate $A(x)$ using $O(n)$ multiplications and $O(n)$ additions?
   Can you evaluate $A(x)$ using at most n multiplications and n additions?

   (b) Now suppose that $A(x)$ has at most k non-zero terms. How can you evaluate $A(x)$ using $O(k \log n)$ operations.
   Hint. How can you evaluate x^n using $O(\log n)$ operations.

3. **Interpolating Polynomials** The values $A(x_0), A(x_1), \ldots, A(x_n)$, define a unique degree n polynomial having those values. In class, we saw the Langragian interpolation formula for finding the coefficients $a_0, a_1, \ldots, a_n$ of $A(x)$. This worked by first setting

$$I_i(x) = \prod_{0 \leq j \leq n,\, j \neq i} \frac{x - x_j}{x_i - x_j}$$

   and then defining

$$A(x) = \sum_i A(x_i) I_i(x).$$

   Show how to use the formula to evaluate the coefficients of $A(x)$ in $O(n^2)$ time.

   Hint How long does it take to divide a degree n polynomial by a degree one polynomial? You can use this procedure as a subroutine.

4. **More Median of Medians** For this problem you can assume the following fact: $\alpha, \beta \geq 0$, N is a non-negative integer and c, D constants (possibly negative). For $n > N$, if

$$T(n) \leq T(\alpha n + c) + T(\beta n + d) + \Theta(n)$$

then

$$T(n) = \begin{cases} O(n) & \text{if } \alpha + \beta < 1 \\ O(n \log n) & \text{if } \alpha + \beta = 1 \\ \Omega(n \log n) & \text{if } \alpha + \beta > 1 \end{cases}.$$

Recall that our determinsitic selection algorithm yielded the recurrence

$$T(n) = T(n/5) + T(7n/10 + 6) + \gamma n$$

for some constant γ. The formula above implies $T(n) = O(n)$.

Our algorithm (i) splits the items into sets of 5 elements, (ii) found the median of each set and then (iii) found x, the median of those medians. It then ran *partition* with x as a pivot and recursed on the appropriate subset. From the definition of x we were able to prove that the subarrays created by partition both had at most $7n/10 + 6$ elements, leading to the recurrence relation and hence $O(n)$ running time.

Now suppose that instead of splitting the items into sets of size 5, we split them into sets of size 3 and then ran the algorithm the same way. Would we still get an $O(n)$ time algorithm?

What about if we split into sets of size 7?

5. **Extra Problem: Finding defective coins**
You have just been hired as the quality-control engineer for a company that makes coins. The coins must all have identical weight. You are given a set of n coins and are told that *at most one* (possibly none) of the n coins is lighter than the others. Your task is to develop an efficient test procedure to determine which of the n coins is defective or report that none is defective. To do this test you have a scale. For each measurement you place some of the coins on the left side of the scale and some of the coins on the right side. The scale indicates either that (1) the left side is heavier, (2) the right side is heavier or (3) both subsets have the same weight. It does not indicate how much heavier or lighter.

   (A) Design an algorithm for solving this problem that works in $\log_2 n + c$ time, for some constant c (try to make c as small as possible).
   Hint: try a divide and conquer approach.

   (B) Design an algorithm for solving this problem that works in $\log_3 n + c$ time, for some constant c (try to make c as small as possible).

   (C) The problem now changes in that the defective coin, if it exists, may be lighter or *heavier* than the other coins. Modify the algorithm from part (A) or (B) to work in this case. How fast is your algorithm?