Problem Set 1

All parts are due Thursday, February 20 at 11:59PM. Please download the .zip archive for this problem set, and refer to the `README.txt` file for instructions on preparing your solutions. Remember, your goal is to communicate. Full credit will be given only to a correct solution which is described clearly. Convoluted and obtuse descriptions might receive low marks, even when they are correct. Also, aim for concise solutions, as it will save you time spent on write-ups, and also help you conceptualize the key idea of the problem.

Your Name: Eunice Wu

Collaborators: Rebekah Cha, Jordan Powell

Part A

Problem 1-1.

(a) $f_1(n) < f_2(n) < f_3(n) < f_5(n) < f_4(n)$

(b) $f_4(n) < f_5(n) < f_3(n) < f_1(n) < f_2(n)$

(c) $f_1(n) < f_2(n) < f_3(n) < f_5(n) \sim f_4(n)$

Problem 1-2.

(a) $\theta(x)$

(b) $\theta(x \log(x))$

(c) $\theta(y \log(x))$

(d) $\theta(\log x \log y)$

(e) $\theta(x + y)$

Problem 1-3.

(a) The recurrence can be represented by a binary tree and therefore has a height O(log(number of columns)). Supposing that we have an array with dimensions n rows x m columns. Each step can be represented by

$$T(n, m) = \theta(n) + 2T(n, m/2)$$

where $f(n)$, the time it takes to divide/combine and execute each step, is $\theta(n)$ because to find the peak, each step involves finding the maximum of each column which runs through all n rows of the current midpoint column and therefore is O(n).

(b) *Assume*: Recursive call returns peak for subproblem (given by problem)
Base Case: Number of Columns (m) = 1
The algorithm will terminate when passed the base case because each step is strictly smaller than the previous step. At the base case case, it is also guaranteed to return the correct answer because, assuming that the method peakProblem.getMaximum() is correct, finding the maximum is the only step needed for the base case Peak Problem. In the case where there is no dividing line, the "getBetterNeighbor" function just returns the previous "bestLoc," so at m = 1, the algorithm is correct.
Inductive Step: Let us assume that the algorithm is correct for an m-column array and all subarrays. We can add another column so that we are now at m+1 columns. There will be two cases upon finding the midpoint column's max:

case 1: A neighbor is the better peak
If this is the case, then we will move on to apply the algorithm on that neighbor's corresponding $\frac{(m+1)}{2}$ column array which, since $m >= 1$, is a sub array of an m-column array, and therefore would return the right answer with algorithm 3.

case 2: The previous stored bestLoc is returned If this is the case,

Problem 1-4.

(a)

Problem 1-5.

Problem 1-6.

Part B

Submit your implemented python script.