Time Spent: 1 hour

Collaborators and Resources:

Problem 1

We're basically keeping track of the change in temperature from the previous day to today. As a result, the most extreme temperature change for any given period (of unspecified length) is going to be calculated as the maximal contiguous (a sequence of consecutive numbers) subsequence of an array A. That is, the maximum sum of any contiguous subsequence. The way I thought about this problem was to use the divide and conquer method in order to split the array into two and then keep track of the composite subarrays . . .

1. Define a function maxSeq as follows:
2. Given an array A = [1, ..., n] of integers representing temperature change:
3. If n = 1, the maximal contiguous subsequence is just A[n] so return A[n]. Otherwise,
   (a) Let L = [the first n/2 elements in A]
   (b) Set div. = the index of the rightmost value in L
   (c) Let R = [the last n/2 elements in A]
   (d) sumL = maxSeq(L)
   (e) sumR = maxSeq(R)
   (f) Return max(sumL, sumR)

Claim 1. The algorithm runs in O(n) time according to the master theorem and also returns the maximum sum of all contiguous subsequences of the array given.

Proof. According to the Master Theorem, we know that since f(n) is in $\theta(n^{log_b a})$; that is, there are two recursive calls (b = 2) and two operations (defining L and R) within each iteration of the function maxSeq. Basically, a = 2. So the master theorem dictates that we have the first case (i) in which T = $\theta(n^{log_b a}) = \theta(n^1)$ which clearly has better run-time than O(n^2). In addition, we can conjecture that the algorithm returns the maximum sum of all contiguous subsequences of the given array because the recursion will go all the way down to one element (where n = 1) and then build up the maximum sum for both the left and the right side of the divider (which is constructed each time). As we can see from the pseudo-code, the algorithm is a recursive one which runs in θ(n) time and also should fulfill its intended function. □