Homework 9
EXTRA CREDIT 2
Due Sunday April 26, 5 PM EST

Extra credit is worth one extra percent of your overall class grade. TAs will not answer Piazza questions or hours queries on any extra credit topics.

To submit your solutions, please make a directory ~/course/cs0160(extraCredit2 containing your solution PDF and then run the script cs0160_handin extraCredit2.

1 Written Problems

Problem 9.1
Adding an Edge

Given a graph $G$ of $V$ vertices and $E$ edges AND a minimum spanning tree $T$ of graph $G$, we would like to add a new edge $e$ with weight $w_e$ to $G$, forming a new graph $G'$. Describe an algorithm which constructs the minimum spanning tree of $G'$ in $O(V)$ time.

Solution:

Given that we already have a minimum spanning tree, $T$, we can add $e$ to $T$ and know that all but one edge of this newly constructed tree, $T'$, will be included in our final minimum spanning tree. However, simply adding $e$ to $T$ necessarily creates a cycle by the definition of a tree, so in order to properly construct a final minimum spanning tree we need to remove an edge from $T'$. Because we know that there are exactly $V$ edges in this new Graph (V-1 edges in a Tree plus the one added edge), then if we were to remove the highest weighted edge from the cycle we just created (which could only be composed of up to $V$ edges) we could again have a tree, everything would be connected, and we would have removed the largest weighing edge from the only portion of the graph that we modified. What this means is that by iterating over all edges in the cycle, which we have shown to be $O(V)$, we can remove the highest weighing edges without changing the portion of the MST that would never have been affected by this added edge, resulting in a new MST with the added edge included (if it is not the heaviest edge in the cycle) or an otherwise unmodified MST (if it is).
Problem 9.2

Rotated Array

Given a sorted array of n integers that has been rotated an unknown number of times, give an O(logn) algorithm that finds an element in the array. Here, rotating an array once means shifting its elements one position to the left, such that the element that was originally first in the array becomes last. You may assume that the array was originally sorted in increasing order and there are no duplicates.

Solution:

Use a modified version of binary search!

function rotated(array, key):
    low = 0
    high = size of array -1
    while low <= high:
        mid = low + ((high - low) / 2)
        if array[mid] == key:
            return mid
        if array[low] <= array[mid]:
            if array[low] <= key and key < array[mid]:
                high = mid -1
            else:
                low = mid +1
        else: //upper half is sorted
            if array[mid] < key and key <= array[high]:
                low = mid +1
            else:
                high = mid -1
    return -1

Problem 9.3

Rotated Array Part II

Now, it is given that in the array described above, the smallest integer has a value of 1, and the array contains only numerically consecutive integers. Write pseudocode to find how many times the original array was rotated.

Solution:

Iterate through the array until you find 1. The index of 1 is the number of times that the array has been rotated:
function rotate2(array):
    for i from 0 to end of array:
        if array[i] == 1:
            return i
    return -1