Problem Set 2

All parts are due on February 22, 2018 at 11PM. Please write your solutions in the LaTeX and Python templates provided. Aim for concise solutions; convoluted and obtuse descriptions might receive low marks, even when they are correct. Solutions should be submitted on the course website, and any code should be submitted for automated checking on `py.mit.edu/6.006`.

Problem 2-1. [30 points] **Counting Inversions**

Given array `A` containing n integers, an **inversion** is a pair of indices (i, j) with $i < j$ for which `A[i] > A[j]`, i.e., a larger number `A[i]` appears *before* a smaller one `A[j]` in the array. The total number of inversions is a measure of how far an array is from being sorted.

(a) [3 points] List all inversions in the array below (assume zero-based indexing).

```
[4,5,3,1,6,3,6,8,4]
```

(b) [5 points] Give an $O(n^2)$ algorithm to count the number of inversions in an array.

(c) [8 points] Given array `A` whose left and right halves, `A[:n/2]` and `A[n/2:]`, are each sorted, give a linear time algorithm to count the number of inversions in `A`.

(d) [8 points] Give an $O(n \log n)$ algorithm to count the number of inversions in an array.

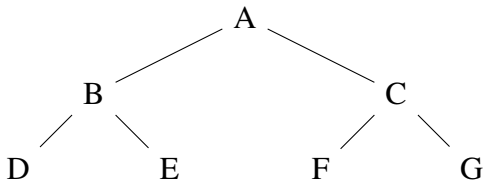
(e) [6 points] Prove that your $O(n \log n)$ algorithm is correct, i.e., returns the correct number of inversions.

Problem 2-2. [30 points] **Heap Practice**

(a) [10 points] For each array below, draw it as a binary tree and state whether it is a max heap, a min heap, or neither. If an array is neither, turn it into a max heap by repeatedly swapping adjacent nodes of the tree (i.e. run `build_max_heap`). You should communicate your swaps by drawing a sequence of trees, with each tree depicting one swap.

1. `[11, 2, 9, 1, 0, 3, 8]`
2. `[1, 3, 2, 4, 5, 7, 6]`
3. `[2, 4, 6, 5, 9, 7, 8]`
4. `[5, 6, 7, 1, 8, 2, 9]`
5. `[10, 3, 9, 1, 2, 7, 4]`

(b) [12 points] Consider the following binary tree on seven nodes labeled A – G.



Assume each tree node contains a unique key that satisfies the max heap property. For each of the following, list the node(s) that could contain the key described.

1. The heap's largest key
2. The heap's second largest key
3. The heap's third largest key
4. The heap's smallest key

(c) [8 points] Given integer array `A` satisfying the max heap property, index `i`, and value `v`, write pseudocode[1] for a function `change_value(A, i, v)` which replaces the value at `A[i]` with `v`, and then swaps at most $O(\log n)$ pairs of elements to ensure the array maintains the max heap property after the change. The only operations you are allowed to perform on the array are comparisons and swaps between pairs of the integers, i.e., `A[i] < A[j]` and `A[i], A[j] = A[j], A[i]`. You are **not allowed** to store array elements outside of the array.

[1] Psuedocode may be lines of English sentences, Python code, or a combination of the two.

Problem 2-3. [40 points] **Cow Party**

Bessie the cow wants to throw a party! She has F cow friends, and each cow friend C_i lives in a plane pasture, X_i meters east and Y_i meters north of Bessie's house. Bessie would like to invite all her friends to the party, but her house only has room to accommodate her and at most $G < F$ cow guests. Because Bessie is a pragmatic cow, she decides to invite the G cow friends that live closest to her so they don't need to travel far (you may assume that her distance to each cow friend is unique). Let's help Bessie compile her guest list!

(a) [5 points] Given an array of cow friend locations, describe an $O(F \log F)$ time algorithm to return Bessie's guest list.

(b) [10 points] Bessie thinks your $O(F \log F)$ time algorithm is too slow. Describe an $O(F + G \log F)$ time algorithm to return Bessie's guest list. Your algorithm should also be **in-place**, i.e., use no more than $O(1)$ extra space beyond the input array.

(c) [10 points] Bessie asks you to demonstrate your $O(F + G \log F)$ time algorithm for her. In doing so, she realizes that your algorithm involves reading **and writing** to the cow friend location list that she gave you. She is angry, and tells you that she does not want you writing on her list. She says that your algorithm may use $O(G)$ extra space, so long as you only read from her list. Describe a $O(F \log G)$ time algorithm to return Bessie's guest list using at most $O(G)$ extra space, provided only read access to the cow friend location array.

(d) [15 points] Write a Python function `close_cow_friends` that implements your $O(F \log G)$ time algorithm. You can download a code template containing some test cases and useful functions presented in recitation from the website. Submit your code online at `py.mit.edu/6.006`.

```
def close_cow_friends(locations, g):
    '''
    Return g locations closest to origin in increasing order.
    Input:  locations | generator of location tuples (x, y)
            g         | number locations to return
    '''
    ##################
    # YOUR CODE HERE #
    ##################
    return []
```