CIS 575. Introduction to Algorithm Analysis
Assignment #6, Spring 2019
Due Thursday, March 21, 11:59pm

You may if you so prefer work in groups of two in which case each name should be listed on your answer but only one of you should submit.

Problem (40p). *Overview*: We shall explore various (in-place) algorithms for sorting an array of integers. In your favorite programming language, implement: the *insertion sort* algorithm, and the *heapsort* algorithm. Your programs must also output *the number of times two elements are swapped* which we shall use as the measure of running time.

1. (6p) Write a program that implements the Insertion Sort algorithm.
2. (3p) Write a program that for a given n generates a *Pseudo-Random* array of size n, in that index i contains $(13 \cdot i)$ mod n. For example, if $n = 10$ and you have chosen a language where the first index in an array is 0 (in some languages it is 1), you should return an array with the elements $0, 3, 6, 9, 2, 5, 8, 1, 4, 7$.
3. (3p) Write a program that for a given n generates an *Almost-Ordered* array of size n, in that index i contains i, except if i mod $13 = 12$ in which case it contains $(i + 13)$ mod n. For example, if $n = 40$ you should return an array with the elements $\ldots 10, 11, 25, 13, 14, \ldots 23, 24, 38, 26, 27, \ldots 36, 37, 11, 39$.
4. (6p) Run your Insertion Sort algorithm on 6 test sets: Pseudo-Random input and Almost-Ordered input, each of size 100, of size 1,000, and of size 10,000. Report the running times (the number of swaps).
5. (6p) Write a program that implements the HeapSort algorithm. Recall that the first step is to convert the array into a heap, as done by the pseudo-code

   for i $\leftarrow n$ **downto** 1
   SIFT(i)

   where SIFT is defined on the lecture slides (and corresponds to what *Cormen* calls MAX-HEAPIFY). *For this question*, feel free to look online for inspiration!
6. (6p) Run your HeapSort algorithm on the 6 test sets from Question 4, and report the running times (that is, number of swaps).
7. (10p) Repeat Question 6 for a version of HeapSort that uses an alternative approach (with PERCOLATE defined on the lecture slides) to convert the input array into a heap:

   for i $\leftarrow 1$ **to** n
   PERCOLATE(i)

To summarize, you should submit the code you have written (you do not need to tell how to run it), and tables depicting the results of your various experiments. You are *not* asked to write about how your results correspond to the asymptotic running times we have discovered by theoretical analysis earlier in this course, but you are very much encouraged to think about it!