Problem Set 1

All parts are due on February 15, 2019 at 6PM. 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 `alg.mit.edu`.

Problem 1-1. [20 points] **Asymptotic behavior of functions**

For each of the following sets of five functions, order them so that if f_a appears before f_b in your sequence, then $f_a = O(f_b)$. If $f_a = O(f_b)$ and $f_b = O(f_a)$ (meaning f_a and f_b could appear in either order), indicate this by enclosing f_a and f_b in a set with curly braces. For example, if the functions are:

$$f_1 = n, \qquad f_2 = \sqrt{n}, \qquad f_3 = n + \sqrt{n},$$

the correct answers are $(f_2, \{f_1, f_3\})$ or $(f_2, \{f_3, f_1\})$.

Note: Recall that a^{b^c} means $a^{(b^c)}$, not $(a^b)^c$, and that $\log$ means $\log_2$ unless a different base is specified explicitly. Stirling's approximation may help for part **d)**.

a)	**b)**	**c)**	**d)**
$f_1 = 6006^n$	$f_1 = n^{3 \log n}$	$f_1 = 3^{2^n}$	$f_1 = 2^n$
$f_2 = (\log n)^{6006}$	$f_2 = (\log \log n)^3$	$f_2 = 2^{2^{n+1}}$	$f_2 = \binom{n}{2}$
$f_3 = 6006n$	$f_3 = \log((\log n)^3)$	$f_3 = 2^{2n}$	$f_3 = n^2$
$f_4 = n^{6006}$	$f_4 = \log(3^{n^3})$	$f_4 = 8^n$	$f_4 = \binom{n}{n/2}$
$f_5 = n \log(n^{6006})$	$f_5 = (\log n)^{\log(n^3)}$	$f_5 = 2^{n^3}$	$f_5 = 2(n!)$

Problem 1-2. [10 points] ***k*-Way Merge Sort**

Merge sort is a divide-and-conquer algorithm based on the idea that merging two sorted subarrays into one large sorted array is possible in linear time. Recall that merge sort divides unsorted input into two equally-sized subarrays, recursively sorts those subarrays, and then merges them.

An excited computer scientist named Talan Uring is wondering whether he can modify the merge sort algorithm to sort even faster! In particular, he wants to k-way merge sort. A k-way merge sort divides the input into k equally-sized subarrays. These k subarrays are then recursively sorted and merged into a new array. To merge them, maintain a pointer for each subarray. Each pointer starts at the beginning of its subarray. The algorithm then determines the minimum of the elements pointed to by the k pointers and copies it to the next position in the new array. The pointer to the minimum element is then moved forward by one. This procedure continues until all the pointers are at the end of their subarrays. Note that a 2-way merge sort corresponds to standard merge sort.

(a) [3 points] Suppose Talan sets k to 5. Write a recurrence for the 5-way merge sort algorithm.

(b) [4 points] Solve the recurrence from (a) using the Master Theorem. How does the running time of this approach compare to standard merge sort?

(c) [3 points] Talan now considers setting k to a **non-constant** value, specifically $k = \sqrt{n}$ where an array of size n is divided into $\sqrt{n}$ subarrays to merge. Write a recurrence for $\sqrt{n}$-way merge sort. You do **not** have to solve this recurrence.

Problem 1-3. [20 points] **Solving recurrences**

Derive solutions to the following recurrences in two ways: draw a recursion tree **and** apply the Master Theorem. Assume $T(1) = \Theta(1)$.

(a) [5 points] $T(n) = 2\,T(\frac{n}{2}) + \Theta(2^n)$

(b) [5 points] $T(n) = 3\,T(\frac{n}{9}) + O(\sqrt{n})$

(c) [5 points] $T(n) = 2\,T(\frac{n}{3}) + \log_7 n + 6$

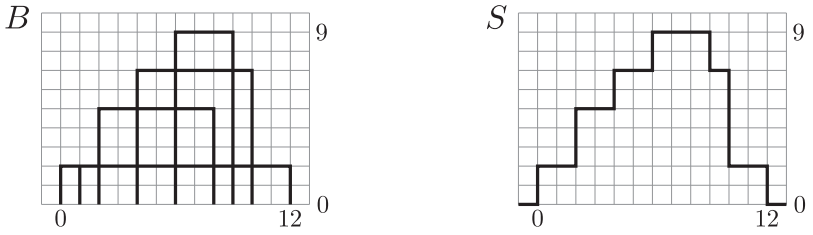
(d) [5 points] $T(n) = \sqrt{n}\,T(\sqrt{n}) + O(n)$ Assume $T(2) = \Theta(1)$ and only draw a recursion tree.

Problem 1-4. [50 points] **Cityscapes**

Judy Ruliani is the mayor of Yew Nork, a small town located on the Husdon Bay. She has pledged to transform the town into a major metropolitan city by facilitating the construction of skyscrapers, which will remake the city skyline as seen from across the bay. Given a set of construction requests for new buildings, she would like to know what the future skyline of her city will be.

Each proposed new **building** is represented by a triple (x_L, h, x_R), denoting what it will look like from across the bay: a rectangular projection having horizontal extent from $x_L \in \mathbb{N}$ to $x_R \in \mathbb{N}$ with $x_L < x_R$, and vertical extent from height 0 to $h \in \mathbb{N}^+$. The **skyline** of a set of buildings is an ordered sequence of pairs $((x_0, h_0), (x_1, h_1), \ldots, (x_{k-1}, h_{k-1}), (x_k, h_k))$, such that for all $i \in \{0, \ldots, k-1\}$: $x_i < x_{i+1}$, $h_i \neq h_{i+1}$, and for every $x \in [x_i, x_{i+1})$: there exists no input building (x_L, h, x_R) such that $x_L \leq x < x_R$ and $h > h_i$, and if $h_i > 0$, there exists an input building (x_L, h, x_R) such that $x_L \leq x < x_R$ and $h = h_i$. Note, we require $h_k = 0$ and $h_0 \neq 0$.

For example, the skyline of buildings $B = \{(4, 7, 10), (2, 5, 8), (6, 9, 9), (0, 2, 1), (1, 2, 12)\}$ would be $S = ((0, 2), (2, 5), (4, 7), (6, 9), (9, 7), (10, 2), (12, 0))$.



(a) [5 points] Given n buildings, a naïve incremental algorithm can construct a skyline by computing the skyline of the first k buildings recursively, and then add building $k+1$ to the skyline. Describe an $O(k)$ time algorithm to add one building to a skyline which was constructed from k buildings. (For any algorithm you describe in this class, you should **argue that it is correct**, and **argue its running time**.)

(b) [10 points] Consider two sets of buildings B_1 and B_2 where $n = |B_1| + |B_2|$, with S_1 the skyline of B_1 and S_2 the skyline of B_2. Describe an $O(n)$ time algorithm to compute the skyline of $B_1 \cup B_2$ from S_1 and S_2.

(c) [10 points] Describe an $O(n \log n)$ time algorithm to find the skyline of n buildings.

(d) [25 points] Write a Python function `build_skyline` that implements your algorithm. You can download a code template containing some test cases from the website. Submit your code online at `alg.mit.edu`.

```
def build_skyline(B):
    '''
    Input:  B | Tuple of building triples (xL, h, xR)
    Output: S | List of pairs corresponding to the skyline of B
    '''
    ##################
    # YOUR CODE HERE #
    ##################
    return []
```