Introduction to Algorithms: 6.006
Massachusetts Institute of Technology — February 15, 2019
Instructors: Jason Ku, Vinod Vaikuntanathan, and Virginia Williams — Problem Set 2

Problem Set 2

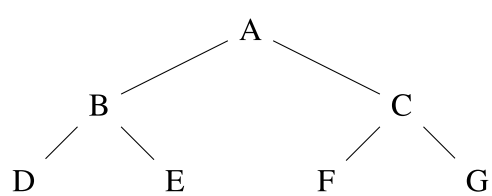
All parts are due on February 22, 2019 at 6PM. Please write your solutions in the LaTeX and Python templates provided. Aim for concise solutions; convoluted and obtuse descriptions might receive low marks, even when they are correct. Solutions should be submitted on the course website, and any code should be submitted for automated checking on `alg.mit.edu`.

Problem 2-1. [20 points] **Heap Practice**

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

1. `[0, 10, 5, 23, 12, 8, 240]`
2. `[17, 7, 16, 5, 6, 2]`
3. `[7, 12, 7, 12, 14, 18, 10]`
4. `[8, 5, 10, 7, 1, 2, 12]`

(b) [10 points] Consider the following binary tree on seven nodes labeled A – G.



Each node stores a key from the multiset $X = \{6, 5, 3, 2, 6, 1, 4\}$ so as to satisfy the **max-heap property** (there are two 6s in X, so exactly two nodes will store a 6). For each node in {A, B, C, D, E, F, G}, list the key values that could be stored there.

Problem 2-2. Shifty Array [10 points]

Describe a data structure that maintains a sequence of items and supports the following operations, each in $O(1)$ time. For each operation, specify whether the running time achieved by your data structure is worst-case or amortized.

- `at(i)`: return the ith item in the sequence
- `insert(x)`: add x as the first item in the sequence
- `shift()`: move the first item in the sequence to the back of the sequence

Problem 2-3. [35 points] **Consulting**

(a) [10 points] **Picky Produce:** High-end grocery store chain Fole Whoods has hired you to purchase watermelons from a wholesale distributor that sells watermelons at one price regardless of size. Fole Whoods on the other hand sells watermelons to customers by weight, at a price of d dollars per ounce, so heavier watermelons sell for a higher price. However, due to restrictive shopping safety regulations, Fole Whoods is not allowed to sell watermelons that weigh more than x ounces, even though their distributor sells them. Each week, you are given a list of p pairs (n_i, o_i) representing the ID number and integer weight in ounces of each watermelon available for purchase. Describe an $O(p \log w)$-time algorithm to determine the IDs of the w most profit-maximizing watermelons available on any given week.

(b) [10 points] **Sorting Students:** 0.660 is a popular algorithms course at a very organized university. Each semester, the course splits its students (sometimes unevenly) into recitations, with each student is in exactly one TA's recitation. At the end of the term, each TA assigns an integer grade to each of their students, and submits to the instructors a list of (name, grade) pairs, one for each student in their recitation. The instructors must then combine grades from all the recitations into one sorted list in order to assign letter grade cutoffs. As the TAs are very organized, each TA submits their list of students to the instructors ordered increasing by grade. Given the sorted list from each of the r TAs, describe an $O(s \log r)$ time algorithm to construct a list containing the names of all s students, sorted by their grade in the course.

(c) [15 points] **Unstable Stable:** Dirgah is a gamekeeper of fantastic beasts at the wizarding school of Wogharts. He has a steady flow of new animals arriving from around the world, which he houses in various pens in his stable; space is limited, so he often houses many animals in the same pen. Each animal is associated with a positive integer representing its *pedagogical value* to the school, whereas the *value of a pen* is the average value of animals housed in that pen. Whenever a new animal arrives, Dirgah houses it in any pen with lowest value. Students often come to Dirgah to request a pet. Even though Dirgah is not supposed to give out animals to students, he has a soft heart. If a lowest valued pen contains any animals, he will give the student whatever animal from that pen has lowest value. Describe a database to help Dirgah keep track of the animals in his pens supporting the following operations; p is the number of pens in the stable and n is the number of stabled animals at the time of the operation.

- `initialize(P)`: given a list `P` of p lists containing the values of animals in each pen, make a database to keep track of them in $O(k + p)$ time, where k is the number of animals in `P`.
- `add_animal(v)`: house an animal with value `v` into the lowest value pen in $O(\log n + \log p)$ time.
- `remove_pet()`: Remove a lowest value animal from the lowest valued pen in $O(\log n + \log p)$ time (if one exists).

Problem 2-4. [45 points] **Bear Bowl Buying**

Mama Bear is very disappointed: Silvribraids broke into her house, ate her porridge, and even stole her favorite bowl! So Mama Bear is off to the store to buy a new one. But Mama Bear can't have just any bowl; she needs one that is not too big and not too small, but just right. As Mama Bear walks around the store, she will look at each bowl and enter its positive integer size into a database on her phone. At any point during her shopping, she would like to know the **best** bowls she's seen so far: specifically, k bowls that are neither the $\lceil \frac{n-k}{2} \rceil$ largest nor the $\lfloor \frac{n-k}{2} \rfloor$ smallest of the n bowls she has seen so far (or all n bowls when $n \leq k$). In this problem, you will design a database to help Mama Bear shop, supporting the following two operations:

- `record_bowl(s)`: add a bowl of size s to the database
- `best_bowls()`: return the sizes of the k best bowls in the database

(a) [5 points] Mama Bear can't find her heaps anywhere! Describe in a data structure that **does not use heaps**, supporting `record_bowl(s)` in $O(n)$ time and `best_bowls()` in $O(k)$ time.

(b) [15 points] Luckily, Mama Bear has found some heaps in the cupboard! Describe a data structure that **uses heaps**, supporting `record_bowl(s)` in $O(k + \log n)$ time and `best_bowls()` in $O(k)$ time.

(c) [25 points] Implement your `record_bowl(s)` and `best_bowls()` methods from part (b) in a Python class `MamaBearDB`. You can download a code template containing some test cases from the website. You may adapt any code presented in lecture or recitation, but for this problem, **you may NOT import external packages** and **you may NOT use Python's built-in sort functionality** (the code checker will remove `List.sort` and `sorted` from Python prior to running your code). Submit your code online at `alg.mit.edu`.

```python
class MamaBearDB:
    def __init__(self, k):
        "Initialize database"
        self.k = k

    def record_bowl(self, s):
        "Add bowl with size s"
        ##################
        # YOUR CODE HERE #
        ##################
        pass

    def best_bowls(self):
        "Return the self.k best bowls"
        ##################
        # YOUR CODE HERE #
        ##################
        return []
```