EECS 2502
Week 6 - Tutorial

Exercise 9.1

Write all possible arrays that store a binary max-heap with keys: 1, 2, 3, 4, 5.

Exercise 9.2

- Consider a max-priority queue Q implemented using a binary max-heap. We would like to design an **ExtractSecondLargest(Q)** operation, which returns the second largest key in Q and deletes it from Q.
- The worst-case running time of this operation must be in O(logn). We assume that all keys in Q are distinct integers.

Exercise 9.3

Consider an array of size 8 that stores a binary max-heap. The indices of the array start from 0 and end at 7.

Write ALL possible indices that could have the **third-largest element** of the array. Assume all elements are distinct.

Exercise 9.4

Consider an open addressing hash table with **m** buckets and consider the following sequence of n keys: 1, 3, 5, 7, ..., 2n-1 (assume n < m).

Specify a hash function **h(k)** and a type of **probing** so that inserting the **n-th** key (2n-1) needs to visit **n** buckets in the hash table.

Exercise 9.5

Consider this “super-efficient” implementation of a max-priority queue using an **unsorted doubly-linked list**, together with an extra variable ***max*** that stores a pointer to the maximum element in the linked list. The claimed worst-case running times of the operations would be the following:

- **Insert**: just insert at the head of the linked list in O(1) time.
- **ExtractMax**: just delete the node pointed to by max in O(1) time (because the list is doubly linked).
- **Max**: just return the priority of the node pointed to by max in O(1) time.

Does it work?

Exercise 9.6

Given two binary max-heaps A and B (each stored in an array), design an **efficient** algorithm that **merges** A and B into a new binary max-heap C that consists of all elements of A and B.