Problem Session 3

Problem 3-1. Hash It Out

Insert integer keys `A = [67, 13, 49, 24, 40, 33, 58]` in order into a hash table of size 9 using the hash function $h(k) = (11k + 4) \bmod 9$. Collisions should be resolved via chaining, where collisions are stored at the end of a chain. Draw a picture of the hash table after all keys have been inserted.

Problem 3-2. Hash Sequence

Hash tables are not only useful for implementing Set operations; they can be used to implement Sequences as well! (Recall the Set and Sequence interfaces were defined in Lecture and Recitation 02.) Given a hash table, describe how to use it as a black-box[1] (using only its Set interface operations) to implement the Sequence interface such that:

- `build(A)` runs in expected $O(n)$ time,
- `get_at` and `set_at` each run in expected $O(1)$ time,
- `insert_at` and `delete_at` each run in expected $O(n)$ time, and
- the four dynamic first/last operations each run in amortized expected $O(1)$ time.

Problem 3-3. Critter sort

Ashley Getem collects and trains Pocket Critters to fight other Pocket Critters in battle. She has collected n Critters in total, and she keeps track of a variety of statistics for each Critter C_i. Describe **efficient**[2] algorithms to sort Ashley's Critters based on each of the following keys:

(a) Species ID: an integer x_i between $-n$ and n (negative IDs are grumpy)

(b) Name: a unique string s_i containing at most $10\lceil \lg n \rceil$ English letters

(c) Number of fights fought: a positive integer f_i under n^2

(d) Win fraction: ratio w_i/f_i where $w_i \leq f_i$ is the number of fights won

[1] By black-box, we mean you should not modify the inner workings of the data structure or algorithm.
[2] By "efficient", we mean that faster correct algorithms will receive more points than slower ones.

Problem 3-4. College Construction

MIT has employed Gank Frehry to build a new wing of the Stata Center to house the new College of Computing. MIT wants the new building be as tall as possible, but Cambridge zoning laws limit all new buildings to be no higher than positive integer height h. As an innovative architect, Frehry has decided to build the new wing by stacking two giant aluminum cubes on top of each other, into which rooms will be carved. However, Frehry's supplier of aluminum cubes can only make cubes with a limited set of positive integer side lengths $S = \{s_0, \dots, s_{n-1}\}$. Help Frehry purchase cubes for the new building.

(a) Assuming the input S fits within $\Theta(n)$ machine words, describe an **expected** $O(n)$-time algorithm to determine whether there exist a pair of side lengths in S that exactly sum to h.

(b) Unfortunately for Frehry, there is no pair of side lengths in S that sum exactly to h. Assuming that $h = 600n^6$, describe a **worst-case** $O(n)$-time algorithm to return a pair of side lengths in S whose sum is closest to h without going over.

Problem 3-5. Po-*k*-er Hands

Meff Ja is a card shark who enjoys playing card games. He has found an unusual deck of cards, where each of the n cards in the deck is marked with a lowercase letter from the 26-character English alphabet. We represent a deck of cards as a sequence of letters, where the first letter corresponds to the top of the deck. Meff wants to play a game of Po-k-er with you. To begin the game, he deals you a Po-k-er hand of k cards in the following way:

1. The deck D starts in a pile face down in a known order.
2. Meff **cuts** the deck uniformly at random at some location $i \in \{0, \dots, n-1\}$, i.e., move the top i cards in order to the bottom of the deck.
3. Meff then deals you the top k cards from the top of the cut deck.
4. You **sort** your k cards alphabetically, resulting in your Po-k-er **hand**.

Let $P(D, i, k)$ be the Po-k-er hand resulting from cutting a deck D at location i. Then cutting deck $D =$ `'abcdbc'` at location 2 would result in the deck `'cdbcab'`, which would then yield the Po-4-er hand $P(D, 2, 4) =$ `'bccd'`. From a given starting deck, many hands are possible depending on where the deck is cut. Meff wants to know the **most likely** Po-k-er hand for a given deck. Given that the most likely Po-k-er hand is not necessarily unique, Meff always prefers the lexicographically smallest hand.

(a) Describe a data structure that can be built in $O(n)$ time from a deck D of n cards and integer k, after which it can support `same(i, j)`: a constant-time operation which returns True if $P(D, i, k) = P(D, j, k)$ and False otherwise.

(b) Given a deck of n cards, describe an $O(n)$-time algorithm to find the most likely Po-k-er hand, breaking ties lexicographically. State whether your algorithm's running time is worst-case, amortized, and/or expected.