Section 04: Dictionaries and Heaps

1. Hash table insertion

For each problem, insert the given elements into the described hash table. Do not worry about resizing the internal array.

(a) Suppose we have a hash table that uses separate chaining and has an internal capacity of 12. Assume that each bucket is a linked list where new elements are added to the front of the list.

Insert the following elements in the EXACT order given using the hash function $h(x) = 4x$:

0, 4, 7, 1, 2, 3, 6, 11, 16

Now insert the above elements in the EXACT order given using the hash function $h(x) = 5x$.

(b) Suppose we have a hash table that uses linear probing and has an internal capacity of 13.

Insert the following elements in the EXACT order given using the hash function $h(x) = 3x$:

2, 4, 6, 7, 15, 13, 19

(c) Suppose we have a hash table that uses quadratic probing and has an internal capacity of 10.

Insert the following elements in the EXACT order given using the hash function $h(x) = x$:

0, 1, 2, 5, 15, 25, 35

2. Evaluating hash functions

Consider the following scenarios.

(a) Suppose we have a hash table with an initial capacity of 12. We resize the hash table by doubling the capacity. Suppose we insert integer keys into this table using the hash function $h(x) = 4x$.

Why is this choice of hash function and initial capacity suboptimal? How can we fix it?

(b) Suppose we have a hash table with an initial capacity of 8 using linear probing. We resize the hash table by doubling the capacity.

Suppose we insert the integer keys $2^{20}, 2 \cdot 2^{20}, 3 \cdot 2^{20}, 4 \cdot 2^{20}, \ldots, i \cdot 2^{20}, \ldots$ using the hash function $h(x) = x$.

Describe what the runtime of the dictionary operations will over time as you keep adding these keys to the table.

3. Heaps

Consider the following sequence of operations:

$$\mathsf{insert}(3), \mathsf{insert}(1), \mathsf{insert}(6), \mathsf{insert}(5), \mathsf{insert}(2), \mathsf{insert}(4).\mathsf{insert}(0), \mathsf{remove}(2), \mathsf{remove}(5)$$

After each operation, show the min-heap by writing down the associated array and its corresponding tree along with the percolation steps for each operation.

4. Design

Imagine a database containing information about all trains leaving the Washington Union station on Monday. Each train is assigned a departure time, a destination, and a unique 8-digit train ID number.

What data structures you would use to solve each of the following scenarios. Depending on scenario, you may need to either (a) use multiple data structures or (b) modify the implementation of some data structure.

Justify your choice.

(a) Suppose the schedule contains 200 trains with 52 destinations. Given a destination, the goal is to print all the trains assigned to that destination.

(b) In the question above, the data structures were designed to efficiently print all the trains assigned to a destination. Now, given a destination, the goal is to print all the trains assigned to that destination in ascending order of their departure times.

(c) A train station wants to create a digital kiosk. The kiosk should be able to efficiently and frequently complete look-ups by train ID number so visitors can purchase tickets or track the location of a train. The kiosk should also be able to list out all the train IDs in ascending order, for visitors who do not know their train ID.

Note that the database of trains is not updated often, so the removal and additions of new trains happen infrequently (aside from when first populating your chosen data structure with trains).