COMP352

Tutorial 8

Maps, Dictionaries, Hashs -- Outline

- Maps
- Dictionaries
- Ordered Maps
- Binary Search
- Hash Tables
- Hash Maps

Map ADT

- A Map is an ADT that allows us to store values based on unique keys.
- These entries are typically called key-value pairs.
- The keys are required to be unique so that each key only points to one value.
- This is similar to how an array works except that instead of integer indexes, we use a more generic key as the "index"

Map ADT -- Methods

- size(), isEmpty()
- get(k)
  - return the value associated with key k, if none, return null
- put(k, v)
  - store v as the new value for key k. return the old value or null if none.
- remove(k)
  - remove the entry with key k if one exists. return the old value or null if none.
- keys(), values()
  - return an iterable collection of all the keys or values stored in the map.
- entries()
  - return an iterable collection of all the entries stores in the map as key-value pairs.

Dictionary ADT

- The Dictionary ADT functions very similarly to the Map ADT.
- Dictionaries unlike maps allow more than one entry to have the same key.

Dictionary ADT -- Methods

- size(), isEmpty()
- get(k)
  - return the first value associated with key k, if none, return null
- getAll(k)
  - return an iterable collection of all the values associated with key k.
- put(k, v)
  - add an entry with key k and value v.
- remove(e)
  - remove the key-value entry e.
- entries()
  - return an iterable collection of all the entries stores in the dictionary as key-value pairs.

MAP ADT -- ORDERED MAP

- An ordered map stores key-value pairs in an ordered search table.
- An ordered search table is an implementation of a Map using an ordered ArrayList.
- Binary Search is a classic algorithm to locate an entry in the table. It runs in O(log n) time.

Hash Tables

- A hash table is an efficient means to store a map.
- A Hash table consists of two components:
  - Bucket Array:
    - An array of a generally fixed size where each entry is can be thought of as a "bucket“ (list) that contains a set of key-value pairs.
  - Hash Function:
    - A function that maps the generic key type to an integer to use as an index for the Bucket Array. The result of the hash function is called a hash value.

Hash Functions -- Collisions

- One issue with hash tables is how well the Hash Function behaves. That is to say, how well the keys map to integers.
- When two keys share the same hash value (result of the hash function), we get a collision.
- A good hash function minimizes collisions under most conditions.
- The way a hash table implementation handles collisions has an impact on the running time complexity of functions relying on the hash table.

Collision Handling

- **Separate Chaining**: each Bucket A[i] stores a small map (list)
- **Linear Probing**:

A[i+1 mode N] i=h(k) if A[i+1]occupied

try A[i+2] mode N

- **Quadratic Probing**: A[i+f(j) mode N] f(j)=j^2 j=0,1,2,3,..
- **Double Hashing**:

A[i+f(j) mode N] f(j)=j.h'(k) j=1,2,3

PROBLEM R-9.7, R-9.8, R-9.9, R-9.10

- Assume an 11 entry hash table
- Use the hash function h(i) = 2i + 5 mod 11
- Insert the keys: 12, 44, 13, 88, 23, 94, 11, 39, 20, 16, 5
- Draw the contents of the hash table given that for collisions:
  - Chaining is used
  - Linear Probing is used
  - Quadratic Probing is used
  - Double Hashing is used with h'(k) = 7 - (k mod 7)

Problem Solving

- Consider a hash table of size 7 with hash function ($h(k) = k \bmod 7$). Draw the table that results after inserting, in the given order, the following values: 19,26,13,48,17 for each of the three scenarios below :

1) when collisions are handled by separate chaining

2) when collisions are handled by linear probing

3) when collisions are handled by double hashing using a second hash function:

$h'(k) = 5 - (k \bmod 5)$

PROBLEM SOLVING

- Let H be a hash table where collisions are handled by Linear Probing and where re-hashing is used each time the load factor (number of item in the table divided by the size of the table) exceeds ½. We assume that the initial size of H is 2 and that re-hashing doubles the size of the table. After inserting 10 items with different keys, what is the size of the hash table H?

Problem Solving

- Assume an M entry hash table which needs to store N keys.
- Use the hash function h(i) = i mod M
- What is the worst-case search time?
- Would you use this for time critical applications?

Problem Solving

- Consider an initially empty hash table of size M and hash function h(x) = x mod M. In the worst case, what is the time complexity to insert n keys into the table if separate chaining is used to resolve collisions (without re-hashing) ? Suppose each entry (bucket) of the table stores an unordered linked list. When adding a new element to unordered linked list, such as element is inserted at the beginning of the list.
- What is the answer if the linked list are ordered ?
- What is the answer if the collisions are resolved using linear probing , and n <= M/2

Problem Solving

- Assume a 2D array A with a size of n x n.
- This array only contains 1s and 0s.
- All the 1s are before all the 0s in each row
- Describe an algorithm to count all the 1s in A that runs in O(n log n) and not O(n^2)

example A with n = 4 might look like:

1 1 1 0
0 0 0 0
1 0 0 0
1 1 1 1