Tutorial 7

1 Sequence

- A sequence is a container of elements, indexed by a set of contiguous non-negative integers
- Common operations:
  - insert (element, index)
  - append (element)
  - at (index)
  - remove (index)
- Implementating a sequence

	insert	append	at	remove
Array	$O(n)$	$O(1)$	$O(1)$	$O(n)$
C++ style vector	$O(n)$	$O(1)$(Amortized)	$O(1)$	$O(n)$
Linked list	$O(n)$	$O(1)$	$O(n)$	$O(n)$

2 Dictionary ADT

- Common operations:
  - add (key, value)
  - overwrite (key, value)
  - lookup (key)
  - remove (key)
  - maybe print
- Implementing a dictionary

	add	overwrite	lookup	remove
Sorted vector	$O(n)$	$O(\log n)$	$O(\log n)$	$O(n)$
Unsorted vector	$O(1)$	$O(n)$	$O(n)$	$O(n)$
Sorted linked list	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Unsorted linked list	$O(1)$	$O(n)$	$O(n)$	$O(n)$
Balanced BST	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$

- C++ map

- The C++ map maintains the elements as sorted, and is usually implemented using a red-black tree
  - A red-black tree is a kind of BST that does some self-balancing on insert/delete
  - They guarantee worst case of $O(\log n)$ for insert/delete/lookup

3 Hash Table

What if we want to do insert / delete / lookup in constant time, i.e., $O(1)$

- A hash table is simply a vector of k slots.
- hash function take a key value (e.g., student number) and calculates a valid bucket index
  hash: $key \longrightarrow \{0, 1, 2, \ldots, k-1\}$
- Types of hash table
  1. Closed hashing with linear probing
     - If there's a collision, keep adding one to the index until we find an empty (or zombie) slot
     - Set corresponding node to be zombie node when deleting. (lazy deletion)
  2. Open hashing with chaining
     - Each slot stores a pointer that points to a linked list
     - Insert the new node at front of the corresponding slot

4 Exercises

1. Implement this subset of the sequence ADT covered in class using linked list and vector. Assume the sequence stores strings:
   insert(assume no insertion at the end)
   append
   at
2. We are going to use a Hash Table to store the contents of a phone book. Each Node represents a person in the phone book with a name and phone number. Write a hash function that can be used to add a new contact to the Hash Table.
3. Consider the following hash function $h(k) = k \mod 7$. Insert the following entries into a hash table of size 7 using chaining: 14, 10, 20, 13, 0, 21.
4. Consider the following hash function $h(k) = k \mod 7$. Insert the following entries into a hash table of size 7 using linear probing: 14, 10, 20, 13, 0, 21. Afterwards, show the resulting hash table after deleting 14. Moreover, outline the result of searching for 13.