Introduction

This week we will be reviewing symbol tables, their implementations, and applications.

Symbol Tables

Symbol tables are ways of storing key, value pairs. We can think of them like a dictionary where the words are the keys and the definitions are the values. Like dictionaries, symbol tables allow us to easily find a given key and then it’s corresponding value. Symbol tables also support easy additions of new keys with their values.

There are two important things to remember about symbol tables in Java. First that they are usually unordered. This means that there is no guarantee about the order items will be drawn from the map using an iterator and that the order may change after using an iterator. Second that keys are unique. When we add a key-value pair to a symbol table, if the key was already present, the value is updated to the new value. If it was not present, the key is added with the given value.

Java Map

In Java, we can implement a symbol table using the built-in Map library. We must specify the types for the keys and values. Maps support the following operations:

- **put(key, value)**: Adds a key-value pair to our Map.
- **get(key)**: Returns the value associated with the given key or Null if not present
- **contains(key)**: Returns a boolean corresponding to whether or not the given key is present in the Map.
- **iterator**: Returns an iterator over the keys in the Map

Implementation

A naive implementation of a symbol table is an array of keys where were iterate through to find the key were looking for. This implementation clearly has an \( O(n) \) runtime for the get operation above where \( n \) is the number of keys in the symbol table. We can instead implement a symbol table using binary search to achieve faster runtimes. To do this we much make sure that the keys are all comparable we can then build two parallel arrays of keys and values such that the key array is sorted. We can then search the array using binary search to find if the given key if it exists.

With a binary search implementation, the run time of **put** is still \( O(n) \) as the array must be shifted for each addition. The runtime of get and contains however are both now \( O(log(n)) \) since they are conducted using binary search of a sorted array.

Symbol Tables in Other Languages

In Java we implement symbol tables with the Map collection, a built-in library which nonetheless must be imported into our program. In other languages, there is special syntax specifically for symbol tables making them easy to instantiate and edit. Python for example uses "dictionaries" as its implementation of a symbol table. In Python, dictionaries are a primitive type like ints or lists (Pythons term for arrays) and thus can be created as easily as these types can.
Practice Problems

**Problem 1.** You are given a stream of $n$ characters from a known alphabet of $k$ characters. Determine which characters appear exactly 5 times in the stream. Try to use $O(n \log(k))$ time and $O(k)$ space.

**Problem 2.** Jesse is a history buff. He has memorized all the most important dates in the past $k$ years. He found there are exactly $n$ important dates. He wants his list of dates to be available online so that people can check if their birthdays were important dates. His website can’t run too slow though and needs you to build an algorithm to check if a date is important. Design an algorithm that does this in $O(\log(k))$.