Week 09 Tutorial

Hash Tables

Hashing

- Hashtable: a data structure used to store key-value pairs
- Hash function: takes some input (key) and computes it into an index, which is where the value is inserted into the array
- Hash functions should give the same output for the same input
- Can store and access elements from hash tables in O(1) time in average

Example:
hash function: h(x) = x % 10

Hash collisions

- Occurs when the hash function returns the same index for multiple inputs

Methods to solve this

- separate chaining: keep a linked list in each index
- linear probing: on collision, keep probing to the right until an empty slot is found and insert the item into it
- double hashing: on collision use a second hash function to determine the increment of probing and probe right until empty slot found

Hash tables

Advantages

- O(1) time complexity for insertion and searching

Disadvantages

- If a large hash table is being used but there are only a few items being added, this results in a waste of space. We also have to resize the table when we fill up all the slots.

Balanced Search Trees

Advantages

- We allocate memory as we insert nodes so no memory goes to waste
- BSTs are ordered so it's efficient to get the smallest or largest values

Disadvantages

- O(log n) time complexity for insertion and sorting
  - (BUT this is still quite fast)

7. Calls to functions (e.g. some recursive functions) are sometimes very expensive to compute. One often-employed trick when working with such functions is **memoisation**: the first time the function is called with some input, we *cache* this result (i.e. store it somewhere for use later on), and then for subsequent calls to the function with the same input, we consult the cache rather than actually evaluating the call.

   a. Design an ADT for a memoiser that supports memoisation of an expensive single-argument function.
   b. One potential problem with memoisation is that the cache's memory usage can grow quite large when the function is called for lots of inputs. This is often solved by limiting the size of the cache and having a policy which can be used to pick entries to remove from the cache to make more room. What are some possible policies you could use?
   c. Can you think of how you would actually implement some of these policies?