Chapter 8: Data Structures

WMC CS Club

March 17, 2022

For the ACSL Intermediate Division, we focus on five data structures: stacks, queues, trees (particularly BST), and priority queues/heaps.

§1 Stacks and Queues

A stack is like placing a plate on a pile of plates, which is a last-in-first-out (LIFO) structure.

A queue is like waiting in a line, which is a first-in-first-out (FIFO) structure.

Both the stacks and the queues support two operations: `POP()` (remove an element; if no element left, returns `NIL`) and `PUSH(param)` (insert an element). Here is an example to show the difference between a stack and a queue.

Example 1.1

```
PUSH("A")
PUSH("M")
PUSH("E")
X = POP()
PUSH("R")
X = POP()
PUSH("I")
X = POP()
X = POP()
X = POP()
X = POP()
PUSH("C")
PUSH("A")
PUSH("N")
```

Solution 1.1 If these operations are applied to a stack, then the values of the pops are: E, R, I, M, A and `NIL`. After all, there are three items still on the stack: the N is at the top, and C is at the bottom. If, instead of using a stack we used a queue, then the values popped would be: A, M, E, R, I and `NIL`. There would be three items still on the queue: N at the top and C on the bottom.

§2 Trees and Binary Search Trees

Trees, in general, use the following terminology: the **root** is the top node in the tree; **children** are the nodes that are immediately below a parent node; **leaves** are the bottom-most nodes on every branch of the tree; and **siblings** are nodes that have the same immediate parent.

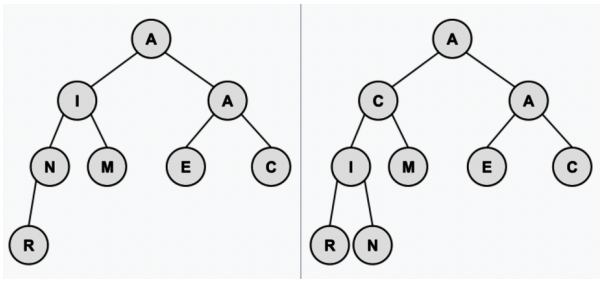
A **binary search tree (BST)** is composed of nodes having three parts: information (or a key), a pointer to a left child, and a pointer to a right child. It has the property that the key at every node is always **greater than or equal** to the key of its left child, and **less than** the key of its right child (that is, in ACSL, when there is a new node with duplicate key, always place it as the left child). The ACSL almost always give the input as a string, so we need to process each character one by one.

Example 2.1

Build a binary search tree for the word `SASKATOON` and determine how many nodes have only one child in the BST.

§3 Priority Queues and Heaps

A **priority queue** (implemented by **min-heap**) is quite similar to a binary search tree, but one can only delete the smallest item and retrieve the smallest item. A min-heap priority queue node has values/keys of both its children **greater than equal** to its own value/key. It is very common used in code to reduce the program run time from $O(n^2)$ to $O(n \log n)$.



The algorithm for insertion is not too difficult: put the new node at the bottom of the tree and then go up the tree, making exchanges with its parent, until the tree is valid. The heap at the left was building from the letters A, M, E, R, I, C, A, N (in that order); the heap at the right is after a C has been added.

§4 Practice Problems

Problems

1. Consider an initially empty stack. After the following operations are performed, what is the value of `Z`? `PUSH(3); PUSH(6); PUSH(8); Y = POP(); X = POP(); PUSH(X-Y); Z = POP();`
2. Create a min-heap for `PROGRAMMING`. What are the letters in the bottom-most row?
3. Create a binary search tree from the letters in the word `PROGRAM`. What is the internal path length (sum of the depths of all nodes)?
4. When creating a BST for `GREATEXPECTATIONS`, which node(s) have two children?

Answer Key

1. -2
2. `RORN`
3. 12
4. G, A, R, T