DATA STRUCTURE

At the heart of virtually every computer program are its algorithms and its data structures. It is hard to separate these two items, for data structures are meaningless without algorithms to create and manipulate them, and algorithms are usually trivial unless there are data structures on which to operate. This category concentrates on four of the most basic structures: stacks, queues, binary search trees, and priority queries. Questions will cover these data structures and implicit algorithms, not on implementation language details.

STACK

A *stack* is usually used to save information that will need to be processed later. Items are processed in a "last-in, first-out" (LIFO) order. A stack supports two operations: PUSH and POP. A command of the form "PUSH(A)" puts the key A at the top of the stack; the command "POP (X)" removes the top item from the stack and stores its value into variable X. If the stack was empty (because nothing had ever been pushed on it, or if all elements has been popped off of it), then X is given the special value of NIL. An analogy to this is a stack of books on a desk: a new book is placed on the top of the stack (pushed) and a book is removed from the top also (popped). Some textbooks call this data structure a "push-down stack" or a "LIFO stack".

Consider the following sequence of 14 operations:

PUSH(A), PUSH(M), PUSH(E), POP(X), PUSH(R), POP(X), PUSH(I), POP(X), POP(X), POP(X), POP(X), PUSH(C), PUSH(A), PUSH(N)

If these operations are applied to a stack, then the values of the pops are: E, R, I, M, A and NIL. After all of the operations, there are three items still on the stack: the N is at the top (it will be the next to be popped, if nothing else is pushed before the pop command), and C is at the bottom.

QUEUE

A *queue* is usually used to process items in the order in which requests are generated; a new item is not processed until all items currently on the queue are processed. This is also known as "first-in, first-out" (FIFO) order. Queues operate just like stacks, except items are removed from the bottom instead of the top. A good physical analogy of this is the way a train conductor or newspaper boy uses a coin machine to give change: new coins are added to the tops of the piles, and change is given from the bottom of each. Some textbooks refer to this data structure as a "FIFO stack".

Consider the following sequence of 14 operations:

PUSH(A), PUSH(M), PUSH(E), POP(X), PUSH(R), POP(X), PUSH(I), POP(X), POP(X), POP(X), POP(X), PUSH(C), PUSH(A), PUSH(N)

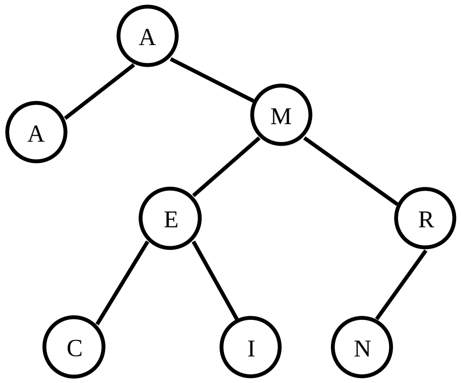
If, instead of using a stack we used a queue, then the values popped would be: A, M, E, R, I and NIL. There would be three items still on the queue: N at the top and C on the bottom. Since items are removed from the bottom of a queue, C would be the next item to be popped regardless of any additional pushes.

BINARY SEARCH TREE

A *binary search tree* is used when one is storing a set of items and needs to be able to efficiently process the operations of insertion, deletion and query (i.e. find out if a particular item is part of the set and if not, which item in the set is close to the item in question).

A binary search tree 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. The following tree is built from the keys A, M, E, R, I, C, A, N in that order:

The *root* of the resulting tree is the node containing the key A; note that duplicate keys are inserted into the tree as if they were less than their equal key.



"depth"

The tree has a depth (sometimes called height) of 3 because the deepest node is 3 nodes below the root.

"leaf node"

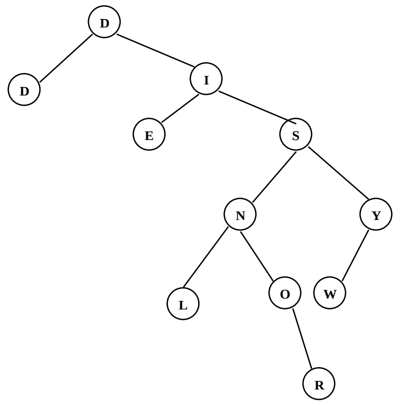
Nodes with no children are called *leaf* nodes; there are four of them in the tree: A, C, I and N.

"comparisons"

3 *comparisons* were needed: against the root A, the M and the R.

Create a binary search tree using the string DISNEYWORLD. How many nodes have just one child?

The nodes Y and O have one child.



Given the following commands on an initially empty queue, what is next item that would be POPPED?

PUSH(S); PUSH(U); PUSH(N); POP(X); PUSH(S); PUSH(H); PUSH(I); POP(X); PUSH(N);
PUSH(E); PUSH(S); PUSH(T); POP(X); PUSH(A); PUSH(T); PUSH(E); POP(X); POP(X);
POP(X); POP(X); POP(X)

~~SUNSHINE~~STATE

A queue is FIFO. There are 8 POP's so the next letter to be POP'ed is S.

Given that the command SWITCH changes a stack to a queue or vice versa and REVERSE switches the order of a given stack or queue, what is the next item to be POPPED from an initially empty queue?

PUSH(S); PUSH(U); PUSH(N); SWITCH; PUSH(S); PUSH(H); PUSH(I); REVERSE; PUSH(N); PUSH(E); SWITCH; PUSH(S); PUSH(T); PUSH(A); PUSH(T); PUSH(E); POP(X); POP(X); SWITCH; POP(X); POP(X); REVERSE; POP(X); POP(X)

Queue: SUN
Stack: SUNSHI
IHSNUSNE
Queue: IHSNUSNESTATE
SNUSNESTATE
Stack: SNUSNESTA
ATSENSU

List the nodes at depth 3 of the binary search tree of the following string?

NORTHHOLLYWOODACSL