Definitions

List: A list is either null or a pair whose tail is a list.

Tree: A tree of a certain data type is a list whose elements are of that data type or trees of that data type. (no tree of null or tree of pairs)

Binary tree: A binary tree is the empty tree, or it has an entry, a left branch or left subtree, a right branch or right subtree.

BST: A BST is a binary tree where all entries in the left subtree are smaller than the entry, and all entries in the right subtree are larger than the entry.

Arrays: An array is a data structure that stores a sequence of data elements. Arrays can be read and written in constant time. *Exception: Assigning to an array element A[i], where index I >= array_length(A), takes Θ(i – array_length(A)) time.*

Environment model

3 things to check for when creating a frame for a function call

1. Extend frame from where function is created. (unless the function comes from the global env)
2. Are there function params? If so, extend frame for params. If not, don't.
3. Are there **new** name declarations in the function body? If so, extend another frame. If not, don't.

If function has no params and no new name declarations in the body, no new frame should be extended.

An environment is a sequence of frames. Each frame contains bindings of values to names.

For a === b, we say "is a identical to b"

true, false, null, undefined — each is identical to itself and nothing else

Numbers — two numbers are identical iff they have the same representation in the double-precision floating-point representation

Strings — two strings are identical if they have the same characters in the same order

Functions — functions are made by function expressions, and their creation bestows an identity upon them

Pairs — pairs are made by the pair function, and their creation bestows an identity upon them

Primitive values (e.g. numbers, strings, Boolean values, null) in bindings are drawn inside frames

Compound structures (e.g. pairs, function objects) are drawn outside frames

Evaluating if statements can be a function application as well

Sanity Check

When you think you're done drawing the environment model, check these 3 things:

1. Do all your function objects have a right arrow pointing at a frame? If not, something's wrong
2. Does every name in every frame have a value assigned? If not, something's wrong.
3. Is every frame pointing at exactly one other frame? If not, something's wrong.

Passing these 3 checks does not guarantee your drawing is correct, but any failure here means your drawing is definitely wrong.

Arrays

An array is a data structure that stores a sequence of data elements. Arrays can be read and written in constant time, *Exception*: Assigning to an array element A[i], where index i ≥ array_length(A), takes Θ(i – array_length(A)) time.

All components in the header for a for-loop are non-optional.

Break terminates the loop and **continue** terminates the current iteration but continues the loop after that.

Searching & Sorting Algorithms

Linear search: O(n) time

Binary search: O(log n) time, requires the input to be sorted, which probably takes another O(n log n) time (Merge sort).

Selection sort: Theta(n^2)

Lists: Find the smallest element x and remove it from the list. Sort the remaining list and put the x in front.

Arrays: Build the sorted array from left to right. For each remaining unsorted portion to the right of position i, find the smallest element and swap it into position i.

Insertion sort: Theta(n^2)

Lists: Sort the tail of the given list using wishful thinking. Insert the head into the right place.

Arrays: Move a pointer i from left to right. The array to the left of i is sorted already. Swap the value at i with its neighbor to the left, until the neighbor is smaller.

Merge sort: Theta(n log n)

Lists: Split the list in half, sort each half using wishful thinking. Merge the sorted lists together.

Arrays: Sort the halves. Merge the halves (using temporary arrays).

Quicksort: Theta(n log n)

Partition the list about a pivot. Append left and right of the pivot. Left portion should have numbers smaller than the pivot, vice versa for right portion.

Other definitions

A **list** of a certain data type is null or a pair whose head is of that data type and whose tail is a list of that data type.

A **tree** of a certain data type is a list whose elements are of that data type, or trees of that data type.

A **binary tree** is the empty tree, or it has an entry, a left subtree, or a right subtree.

A **binary search tree** (BST) is a binary tree where all entries in the left subtree are smaller than the entry, and all entries in the right subtree are bigger than the entry.

Common Recurrence Relations

T(n) = T(n-1) + O(1) = O(n)
T(n) = T(n/2) + O(1) = O(log n)
T(n) = T(n-1) + O(log n) = O(n log n)
T(n) = T(n-1) + O(n) = O(n^2)
Generally, T(n) = T(n-1) + O(n^k) = O(n^k+1)
T(n) = 2 T(n/2) + O(n) = O(n log n) → Merge sort
T(n) = T(n/2) + O(n) = O(n)
T(n) = 2 T(n/2) + O(1) = O(n)
T(n) = 2 T(n-1) + O(1) = O(2^n)

The space complexity of a program refers to the additional space required by the execution of the program. You need time to create space. But taking time, does not necessarily mean you create space. Theta(g(n)) time → O(g(n)) space. Theta(g(n)) space → Omega(g(n)) time