CMSC420 Glossary

asymptotic complexity analysis

for data structures, it's the assessment of the rate of change of either the execution time or the storage requirements, usually as the problem size grows without bound. since time and space are not expected to decrease as the number of elements increases, we expect the functions involved in the analysis to be positive increasing (or at worst monotonic non-decreasing). that allows us to use the weaker version of the definitions of the bounds given in this glossary (as in NO limit notation required).

non-data-structures specific:
investigation of the rate of change of some measure associated with a program or algorithm with respect to a distinguished parameter, when that parameter is allowed to grow without bound. (don't waste your time memorizing this version...just understand what it means).

AVL tree

BST with the additional rule that the heights of the subtrees of any node in the tree differ by at most one. Insertion follows BST rules, and then applies either a single or a double rotation to restore AVL tree property as needed. Deletion follows BST rules–finding replacement node and reestablishing the height difference rule can require more than a constant number of rotations; so, lazy delete is often used. Goal is search in $\mathcal{O}(logn)$. Visit Dave Mount's notes for more details.

binary heap

a complete (or left complete) binary tree where the value in the root of every subtree is no greater than the values in its child nodes (for a min heap); or the value in the root of every subtree greater than or equal to the values in its child nodes (for a max heap)

Typically stored in a static array in C or C++ to exploit known (fast) access function and make rebuilding the heap fast.

access function: node at index i has its left child at $2i + 1$ and its right child at $2i + 2$.

used as the basis for a priority queue.

BST disease

this is a DHD (Darth Hugue detail). It refers to that quality of a BST where the insertion order is such that all non-leaf nodes have at most one non-empty child, and search is in $\mathcal{O}(n)$, just like for a nasty linked list.

Commentary:
Why is this not 'linked list disease'?
Because linked lists can't improve their perfomance by reordering the nodes. unsuccessful search is in $\mathcal{O}(n)$ whether the elements are sorted or not. Thus, $\mathcal{O}(n)$ is the way it is.

In the average case, indeed, a BST has search in $\mathcal{O}(logn)$...it's merely the size of the constants forcing one to require larger and larger values of n, the number of elements to begin to observe the log-like behavior. However, by manipulating the order of data insertion, or the rules regarding relationships between parent and child nodes, the depth of the structure can be manipulated to guarantee that search is in $\mathcal{O}(logn)$.

Thus, if a k-ary tree node had at most one non-empty child node, then $k - 1$ link spots would be wasted and search would be in $\mathcal{O}(logn)$...thus all behave like a BST. The disease is not fatal, but can be rectified by adding a rule (such as limiting the height differences among children of a node) that forces the max depth to be proportional to $logn$, the number of element in the strucure.

BST Property

a binary tree in which each node contains a key satisfies the BST (binary search tree) property if and only if for every node in the structure, the values of the keys in the node's left subtree are less than the node's key, and the values of the keys in the node's right subtree are "greater than or equal to" the node's key.

Notes:
a) "greater than or equal to" is in quotes because most authors don't include 'equal' anywhere in this definition because the definition is usually made because keys are assumed to be unique. however, there are situations, such as in the k-d tree (which will appear probably thursday), where we need to know which way 'equal' goes.

b) the choice of which side gets the 'equal' (left or right) is technically a convention....as in, the resulting binary search tree will be unique as long as everyone puts 'equal' on the right. so, for our class, we will have 'the key in the current node less than or equal to the keys in the right subtree of the node' to make sure that there is no ambiguity in the structures you consult.

c) the generalization of the BST is the K-ary Search Tree (KST) property, which we will use in discussing the point quadtree (the 2-dimensional analog of a BST).

complete k-ary tree

a complete k-ary tree is full with all leaves on the same level. if you add one more node you have to add another level.

alternate characterizations of complete k-ary trees:
a) a k-ary tree in which all nodes have k-non empty children except for the last level, where all nodes have k empty children.

b) a full k-ary tree where all the leaves are at the same level

c) a full k-ary tree where every level is maximal, and no other nodes can be added without increasing the number of levels in the tree

Commentary:
note that these are used for binary heaps (d-ary heaps) to ensure that they can be stored in static arrays without requiring a symbol for an 'empty' node. typically, storage is reserved for a complete k-ary tree, and then the structure may cycle through left complete/complete/left complete/complete as information is removed from the binary heap.

d heap

similar to a binary heap but it is a complete (or left complete) d-ary tree where either:
a) max heap: the key in the root of each subtree is greater than or equal to the keys in its child nodes;

or

b) min heap; the key in the root of each subtree is less than or equal to the keys in its child node.

darth hugue details (DHD)	a phrase that will be used to imply a more complex set of concepts, even if no one else uses it. For example “BST disease” is what i use to refer to any structure that has the same property as the bst of having the shape of the structure determined by the order of element insertion, and therefore the worst case is an expensive linked list, where the nodes can have k spaces reserved for expansion or child nodes, but only one is ever non empty for the non-leaf nodes. I don’t call it linked list disease, becausee manipulating the shape of a linked list is impossible, and linked list disease is fatal, while a k-ary tree’s shape can be manipulated by element insertion order, and can thus can be accomodated by adding rules to how information is inserted into the structure to the basic BST (KST) property that determine the ’natural’ or inherent information orderings that are provided by the mere navigation structure.
depth of a node in a k-ary tree	the depth the root is zero; the depth of a node is one plus the depth of its parent. Note that this is also the level label. Alternatively, the depth of a node in a k-ary tree is the number of edges in the path from the node to the root.
Dynamic data set	refers to the fact that either the number of elements, the values of the keys stored therein, or both are expected to vary throughout the program/application life cycle.
extended k-ary tree	this is a k-ary tree, where the internal nodes are different than the external or leaf nodes. internal nodes have space for k children, and must have at least one nonempty child. leaf nodes have space only for keys (data) and reserve no space for children. why? because they truly have no children. Alternatively, an extended k-ary tree is a k-ary tree where the extenders correspond to the empty child pointers, but can also point to another object. By construction, all extended k-ary trees are full. this makes keeping track of how much overhead is needed to support the data structure easy to compute, and does In an extended k-ary tree, the internal nodes all have at least one non empty child. However, a 2nd kind of node, called an extender, is used to indicate the presence of a child pointer...or potential child pointer. Commentary: What’s the difference? All the internal nodes contain space for pointers to k children, as well as some type of information to guide the choice–sometimes this information is stored in the internal node explicitly; however, most often, a predefined rule is used to determine the correct searh path; so no space or minimal space can be devoted to supporting the branching. Internal nodes are never converted to leaf nodes. They may be deleted, but will never contain a real key. The external nodes will always be external nodes, but can devote all their storage to keys.

external node	used to refer to leaf node of many structures, most commonly, k-ary trees, where an external node has k empty children. Also used for k-ary tries, where leaf nodes have no children. Commentary: What's the difference between "empty chidren" and "no children"? Empty children means that the external node is allocated with space for expansion; that is, given the way the structure is managed, external or leaf nodes become internal nodes when their (empty) children are instantiated. The purpose of the definition is not to annoy but to capture the function or management of the nodes with no 'out arcs'. No children means that no space must be allocated for kids because the leaf is terminal–it will never become an internal node; this is what happens in tries. Commentary 2: The goal of all these definitions is to capture the manner in which the notions of finite graphs and their representations are instantiated on a computer–not merely to annoy you.
forest of general trees	is one or more general trees.
full k-ary tree	a) a k-ary tree in which all internal nodes have k non-empty children. b) a finite set of elements that is either empty or consists of a root and k empty children, or a root and k non-empty disjoint subsets, each of which is a full k-ary tree. (this combines both definitions–k-ary tree and full, and is not appropriate for just a 'full' structure).
full k-ary tree theorem	in a full k-ary tree, the number of external nodes (E) satisfies: $E = (k-1)I + 1$, where I is the number of internal nodes.
full k-ary tree theorem corollary	any k-ary tree with n nodes has $P = (k-1)n + 1$ empty child pointers.
general tree	a general tree is a finite set of elements that consists of a root and zero or more disjoint subsets, each of which is a general tree. Note: general trees can't be empty. Why not?.

Graph, G(V,E)	a graph consists of a finite set of vertices, V, and a set of edges, E, where $G(V, E)$ edge $< vi, vj >$ connects vi to vj, for $i \neq j$. Data structures might be referred to as a study of finite graphs and the associated algorithms–their properties, their applicability, and when NOT to use them ;-) Graph commentary: in data structures, all graphs are finite–meaning that the vertex set has some (finite) cardinality (number of elements), with $\|V\| = n$. While our calculations will assume that n approaches infinity–especially to assess scalability of graph-specific algorithms in terms of time and space–we always treat n as a finite integer that can be permitted to grow to be arbitrarily large. But, it remains finite. As an example, even though we have a finite number of elements in our set, if the set is so big that we can't fit all the keys in 'main memory' and do actual work, then we have to use what are called 'external sorting methods.' Otherwise, we use 'internal sorting methods'.
Graph, sparse	a graph is sparse if the number of edges for n vertices is in $\mathcal{O}(n)$.
height of a k-ary tree	the height of a k-ary tree is often defined as the number of nodes on the unique path between the root and a node whose level has the largest label. Check Dave Mount's notes to see what he uses.
Huffman coding tree	trie used to encode and decode symbols to produce a variable length encoding in which the frequency of a given symbol is roughly inversely proportional to the length of the code for a given symbol. Symbols occur at the leaves, and the method used to construct the tree guarantees that weighted path length associated with the binary tree is minimized (where the weight of a symbol corresponds to its frequency in training data or in a file to be compressed).
internal node	typically associated with trees, but valid for a graph, refers to any node that is not an external (or leaf) node. For a k-ary tree, any node that has at least one non-empty child. internal nodes of k-ary trees and k-ary tries have to be allocated with space for child nodes. however, k-ary tree internal nodes start life as leaves of the k-ary tree. k-ary trie internal nodes never change their role. (just a note to remind you that when we traverse structures we exploit the connectivity of the structure (list, tree, or graph) as well as the relationships among the values stored therein).
k-ary tree	a finite set of elements that is either empty or consists of a root and k disjoint subsets, each of which is a k-ary tree.

k-d tree (of order p)

a k-d tree is a BST used to organize $p-$dimensional keys; a binary search tree property is supported, except that the portion of the key that is used to determine branch decisions at each level rotates through the p coordinate positions. The term ‘key discriminator’ is often used for the process that extracts the appropriate coordinate value from the $p-$dimensional coordinate.

k-way search tree

a k-ary tree in which every node can hold keys (or valid data or value– your choice of term), and a k-way decision is made according to the k-ary (binary) search tree property.

Commentary:
If one implements this structure with external nodes and internal nodes having the same content–as in space for k child pointers, and the key(s) to be used in determining which child to explore—every leaf node will waste the space for k child pointers. since a p+1 level k-way search tree will have a maximum of k^p external nodes, that’s k empty kids each, giving k^{p+1} empty child pointer slots.

This motivated the development of a different computational model, where the internal nodes served only as holders for guide values, and the keys are all in the “leaves” which correspond to the child pointers that are actually used.

k-way search trie

an extended k-ary tree in which the keys are all stored in the leaves (external nodes or extenders) and the internal nodes are used to facilitate search to the appropriate leaf. The values (guides associated with internal nodes, and keys associated with the leaf nodes) must satisfy the k-way search tree property, as well.

left complete k-ary tree

a left complete k-ary tree is complete, except for the last level, where nodes are filled in from left to right. After the last node there are no more non-empty nodes to the right.

Commentary:
Note that these are used for binary heaps (d-ary heaps) to ensure that they can be stored sequentially in static arrays without requiring a symbol for an ‘empty’ node. Typically, storage is reserved for a complete k-ary tree, and then the structure may cycle through left complete/complete/left complete/complete as information is removed from the heap.

level of a k-ary tree

the level of the root is zero; the level of a node is one plus the level of the parent. Note that by agreement, the empty k-ary tree has level numbered -1.

Warning: a k-ary tree with only a root has ONE level, which is labelled ‘0’. A k-ary tree with a two nodes, a root and one child, has TWO levels, which are labelled ‘0’ for the root, and ‘1’ for the child of the root. Be careful.

list

a finite set of elements or finite connected graph $G(V, E)$, where each element can be involved in edges with at most two other nodes.

look! look! That means that the number of edges is proportional to the number of vertices, as in $|E| = c|V|$ for some constant $c > 0$. So, the number of edges grows linearly with respect to the number of vertices.

Typically, in data structures, a list is considered to be a connected graph, where the edges model some relationship between either the nodes as stored (merely the shape or syntax or underlying structure) or the information (semantics or relationships among keys) they hold.

node types in tries and trees more darth hugue details (dhd)

for a graph, an internal node has at least one outgoing arc that is non- empty. note too, that our trees have the assumption of being directed graphs, because a k-ary tree node has exactly one parent, except for the root. so, a non-root internal node of a k-ary tree can be thought of as an element of a directed graph having at most one incoming arc or parent and at least one out going arc, or non-empty child.

These annoying comments are supposed to help some of you think. Why?
Because if all the nodes in my structure maintain the same type (internal, external) throughout their life time, then a natural way to write code is to exploit the commonality associated with treating each node as being a node in the structure. this makes the common case fast, since all internal nodes of trees or tries serve as guides to the leaves (using the values in internal nodes).

then, once each node has done the ‘stuff we do because we are nodes’, the “stuff we do because we are ‘type x’ ” nodes can be separated from the general case, and treated accordingly. This should make sense becaue the nodes have two roles: contain data, and facilitate navigation through the data set.

Thus, any language or process accessing the graph should remember to exploit both sets of attributes.

example? the BST can support six different output patterns from one arrangement of nodes...(inorder, pre order, post order, reverse in order, reverse pre order, reverse post order) A linked list can support two–forwards and backwards; however the cost of the two is paid in space–either staticallly or dynamically.

so code that can exploit the connections –the logical or physical path through the node set (captured in whether the node is a root, or a non-root internal or external node.

Ordered List if we can distinguish the first, 2nd, 3rd, and so on element of the list, the positioning in the list imparts an ORDER on its elements that is independent of the keys. This is in contrast to a sorted list, where the ordering of elements of the list is based on the relationships among the keys present in the list.

Ordered Tree a tree in which the relative position of subtrees matters. Typically associated with k-ary trees, but can also be applied to general trees.

Oriented Tree a tree in which relative order of subtrees is unimportant–only the parent/child and sibling/sibling relationships matter. Typically associated with general trees or rooted trees.

perfect k-ary tree DEPRECATED! See the definition for COMPLETE instead.

a) a k-ary tree in which all nodes have k-non empty children except for the last level, where all nodes have k empty children.

b) a full k-ary tree where all the leaves are at the same level

c) a full k-ary tree where every level is maximal, and no other nodes can be added without increasing the number of levels in the tree.

Planar Graph a graph in which edges intersect only at vertices.

PM Quadtree (extended) 4-ary tree and 4-way search trie used to represent polygonal maps, with predefined partitioning of potential key-space, all quadrants closed on all 4 sides, and all keys in the leaves. PM3, PM2, PM1 determined by rules for legal black (key-holding) nodes. Data consists of edges that intersect only at vertices, with each black node containing at most one vertex.

PM3: additionally, a black node can contain any number of q-edges, where a q-edge is a portion of a segment in a quadtree restricted to that partition.

PM2: all q-edges in a partition must intersect in a common vertex, either within the quadrant, or outside of it.

PM1: at most one q-edge per black node, unless the black node contains a vertex, in which case only q-edges that include that vertex can be in the black node.

point quadtree (k-ary tree) a point quadtree or point octree or point k-ary tree is a structure used to support search for 2D, 3D, and larger tuples. So, a point quadtree would be used for (x, y) coordinates; point octree for (x, y, z) coordinate data. These are also k-way search trees, and inherit lovely BST disease, when they satisfy the KST (k-ary search tree property) and have keys in all the nodes. The trigger word here is *Point*.

point region quadtree this is a trie–the key word is *Region*. It's an extended 4-ary tree, and a 4-way search trie, where the internal nodes correspond to predefined partitions (typically powers of two, with smallest partition being 1×1, perhaps corresponding to a pixel) and serve as guides to the keys that are stored in leaves. internal nodes are called gray nodes; external nodes are called black nodes. note that at most one key can occupy a given PR quadtree leaf. furthermore, traditionally, PR quadtree quadrants are closed on the left and bottom. Furthermore, it must be maximal, meaning that no excess partitions remain after a delete–any unneeded gray nodes must collapse to correspond to the partitioning appropriate for the vertices that remain after the deletion.

priority queue a list from which elements are removed according to a predefined rank.

q-edge refers to that portion of a line segment that is clipped or restricted by a partition of a PM quadtree.

skiplist (also randomized skip list)

this is a probabilistic structure based on a linked list, but with several levels of links to support navigation that is technically $\mathcal{O}(n)$, but the expected value is big-oh(log n). number of links per element is determined at insertion time based on a random number generator. The average number of links per node is 2, and the storage requirements are about that for an AVL tree. Complexity of programs is about that of linked list insertion and deletion–merely a few more links.

Sorted list	a list where the relative positions of keys are determined by applying a rule that compares keys pairwise, and outputs them according to some predefined ranking. So, perhaps a better way to think of it is as a list where the position of the element or node in the list corresponds to the rank of the key when sorted. This is in contrast to an Ordered list, where no such rule is applied.
Sparse Graph	a graph where the number of edges is proportional to a constant times the number of vertices.
Sparse Matrix	a 2-dimensional array where the number of non-zero entries is proportional to the row or column size, not the number of entries in the matrix. For a sparse $n \times n$ matrix, that means that at most some (positive) constant cn of the n^2 total entries are non-zero. Note that the adjacency matrices associated with sparse graphs are also sparse.
static data set	the number of keys and their values are known at run time, and remain unchanged. Note: One can have a fixed-size data set, but allow key modification. Similarly, one can have fixed-key values, but allow the number of times they are instantiated to vary and therefore the number of elements to change at runtime. Both of these cases have to be handled using methods which support dynamic behavior for the appropriate parameter (number of elements or values of keys).
Tree	1) (Samet) a branching structure between nodes. 2) a general term for a finite, connected non-empty graph $G(V, E)$, where each node or vertex, except for one distinguished vertex, called the 'root', has a single parent vertex. The root has no parent vertex. look! look! The size of the edge set satisfies $\|E\| = c\|V\|$ for some constant $c > 0$, meaning that the number of edges grows linearly with the addition of new vertices. Note: In previous semesters, the unadorned word 'tree' was never used in a technical or formal sense in this data structures class. The main reason it was here was so I could write the paragraph above that says "look! look!"–this has now changed. That is, the term 'tree' is now eligible for use in a valid definition on the 2nd exam and on the final.