Data Structures notes

Abiyaz Chowdhury

November 6, 2017

1 Vectors (dynamic arrays) and linked lists

Both are linear containers that store objects to be retrieved in some well-defined manner. The fundamental difference is that vectors are contigious blocks of memory, whereas linked lists are non-contigious, with each node having both a data object as well as a pointer to the next node. In a circular linked list, the tail node points to the first node. A regular linked list has the tail node pointing to null, marking the end of the list. Typically, vectors and linked list are supplied as the memory address of their head node, so that the other nodes can be accessed. However, for doubly linked lists, in which nodes contain pointers to their predecessor as well as successor nodes, the tail node is also provided, so one can traverse the linked list from tail to head. The predecessor of the head node is usually defined to be null, though for a doubly circular linked list the predecessor of the head node is the tail node. Vectors usually have a fixed size, although when a vector becomes full, there is often the option of resizing it by doubling its capacity by allocating a new block of memory with double capacity and copying the elements of the original vector to the new memory block assigned to it.
TODO: Time complexities for vector/linked list.

2 B-trees

2.1 Definition

A B-tree is a type of self-balancing tree data structure that generally has a higher branching factor than regular binary search trees, and therefore has a lesser height. A non-leaf node in a B-tree contains a set of keys, and pointers to a set of child nodes. If for example a node has the keys $(1, 3)$ and the child nodes are (a_0, a_1, a_2), then all keys in the subtree a_0 must be less than 1, all keys in subtree a_1 must be between 1 and 3, and all keys in subtree a_2 must be greater than 3. The leaf nodes contain the actual data records (or pointers to the data records), whereas the non-leaf nodes point to nodes further down the tree. Knuth formally defines a B tree of order m as:

1. Every node has at most m children.

2. Every non-leaf node (except root) has at least $\lceil m/2 \rceil$ children.
3. The root has at least two children if it is not a leaf node.
4. A non-leaf node with k children contains $k - 1$ keys.
5. All leaves appear in the same level.

2.2 Height

Consider a B-tree of order m. A node can have at most $m - 1$ keys. Then the smallest possible value for the height of this tree occurs when all nodes are completely filled (i.e. have exactly m) children.