STL and Big Oh Cheat Sheet

When describing the Big-O of each operation (e.g. insert) on a container (e.g., a vector) below, we assume that the container holds n items when the operation is performed.

Name: list
Purpose: Linked list
Usage: list<int> x; x.push_back(5);
Inserting an item (top, middle*, or bottom): $O(1)$
Deleting an item (top, middle*, or bottom): $O(1)$
Accessing an item (top or bottom): $O(1)$
Accessing an item (middle): $O(n)$
Finding an item: $O(n)$
*But to get to the middle, you may have to first iterate through X items, at cost $O(x)$

Name: vector
Purpose: A resizable array
Usage: vector<int> v; v.push_back(42);
Inserting an item (top, or middle): $O(n)$
Inserting an item (bottom): $O(1)$
Deleting an item (top, or middle): $O(n)$
Deleting an item (bottom): $O(1)$
Accessing an item (top, middle, or bottom): $O(1)$
Finding an item: $O(n)$

Name: set
Purpose: Maintains a set of unique items
Usage: set<string> s; s.insert("Ack!");
Inserting a new item: $O(\log_2 n)$
Finding an item: $O(\log_2 n)$
Deleting an item: $O(\log_2 n)$

Name: map
Purpose: Maps one item to another
Usage: map<int,string> m; m[10] = "Bill";
Inserting a new item: $O(\log_2 n)$
Finding an item: $O(\log_2 n)$
Deleting an item: $O(\log_2 n)$

Name: queue and stack
Purpose: Classic stack/queue
Usage: queue<long> q; q.push(5);
Inserting a new item: $O(1)$
Popping an item: $O(1)$
Examining the top: $O(1)$

If instead of holding n items, a container holds p items, then just replace "n" with "p" when you do your analysis.