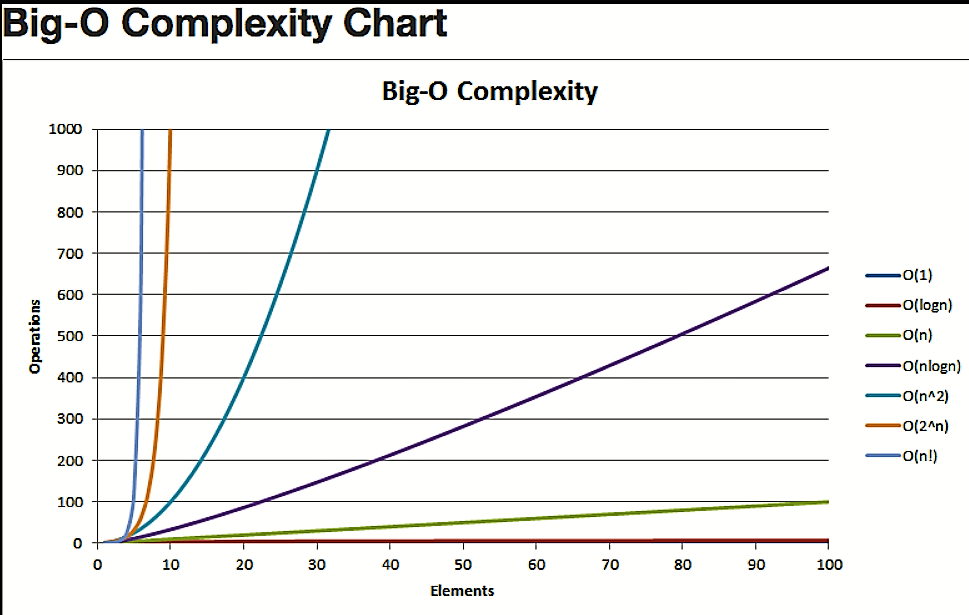
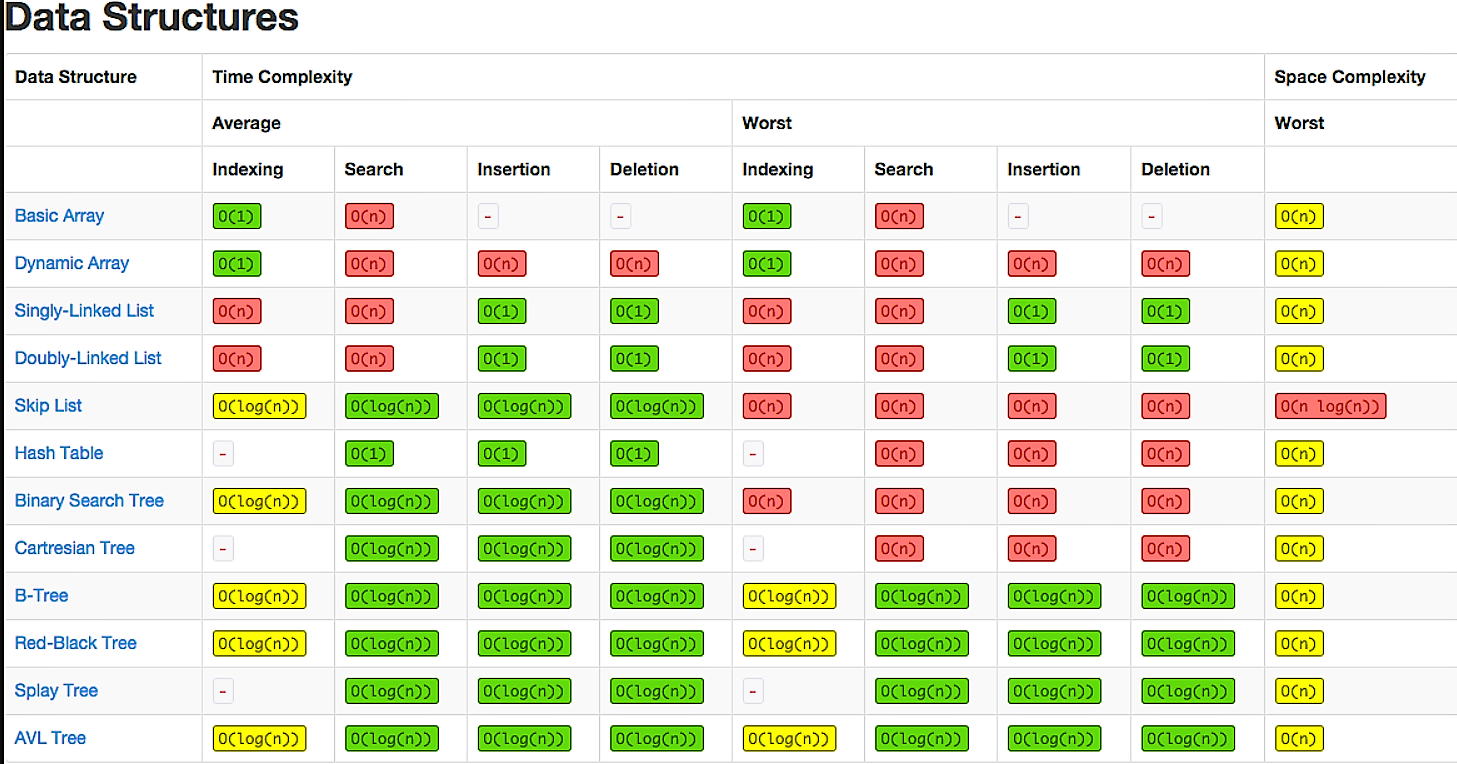
**\*\*\* STL - NOTE \*\*\***

****



**Vector std::vector :**

**Use for:**

* Simple storage
* Adding but not deleting
* Serialization
* Quick lookups by index
* Easy conversion to C-style arrays
* Efficient traversal (contiguous CPU caching)

**Do not use for :**

* Insertion/deletion in the middle of the list
* Dynamically changing storage
* Non-integer indexing

Syntax:

int data, i;

    vector<int> v;// if we know size of vector vector<int> v(n,0);

    unsigned int s = v.size();

    v.insert(v.begin(), data);

    v.insert(v.begin() + i, data);//insert at i position from begin

    v.push\_back(data);

    int head = v.front();

    head = v[0];

reverse(v.begin(), v.end());

    int value = v.at(i);

    value = v[i];

    int tail = v.back();

    tail = v[v.size() - 1];

    for (auto it = v.begin();it != v.end();it++)

        cout << \*it << " ";

    for (auto& x : v)

        cout << x << " ";

// Remove head, index, tail

    v.erase(v.begin()); // head

    v.erase(v.begin() + index); // index

    v.pop\_back();

v.clear();

**Pair:**

pair<int, string> p;

    p = make\_pair(5, "jagdish");//-->O(1);

    p = { 23,"jagdish" };

pair<int, string> arr[3];

    arr[0] = make\_pair(1, "jagdish kumar");

    arr[1] = make\_pair(2, "chaturanan");

    arr[2] = make\_pair(3, "shail patel");

    for (auto& x : arr)

        cout << x.first << " " << x.second << endl;

swap(arr[0], arr[4]);

reverse(arr, arr + 3);

sort(arr, arr + 3);

**Map:**

**Use for**

* Key-value pairs
* Constant lookups by key
* Searching if key/value exists
* Store data in sorted form
* Removing duplicates
* std::map

Ordered map

Insert --- O(log(n))

Access by Key --- O(log(n))

Remove by Key --- O(log(n))

Find/Remove Value --- O(log(n))

* std::unordered\_map

Hash tableInsert --- O(1)

Access by Key --- O(1)

Remove by Key --- O(1)

map<int, int> m;

unsigned int size = m.size();

m.insert({ 3\*3, 9 });

for (int i = 11;i <= 19;i++)

        m[i] = i \* i;

bool exists = (m.find(12) != m.end());

auto v = m.find(12);//it return iterator

    if (v == m.end())

        cout << "value not found" << endl;

    else cout << v->first << " " << v->second<< endl;

// Count the number of elements with a certain key

    cout << m.count(1) << endl;

m.erase("key");

m.clear();

**SET :**

Use for

* Removing duplicates
* Sets are often implemented with binary search trees
* Ordered dynamic storageInsert --- O(log(n))

Remove --- O(log(n))

Find --- O(log(n))

set<string> s;

    s.insert("jagdish kumar");

unsigned int size = s.size();

s.erase(20);

auto it = s.find("jagu");//O(log(n))

for (auto &x : s)

        cout << x << endl;

s.clear();

bool exists = (s.find(20) != s.end());

unsigned int count = s.count(20);

**Stack :**

* First-In Last-Out operations
* Reversal of elements

stack<int> s;

    s.push(12);

while (!s.empty()) {

        cout << s.top() << endl;

        s.pop();

    }

unsigned int size = s.size();

**Queue :**

Use for

* First-In First-Out operations
* Ex: Simple online ordering system (first come first served)
* Ex: Semaphore queue handling
* Ex: CPU scheduling (FCFS)

queue<int> q;

    q.push(5);

while (!q.empty()) {

        cout << q.front() << endl;

        q.pop();

    }

int head = q.front(); // head

    int tail = q.back();

unsigned int size = q.size();