* Vector

1. vector<Type> vect;

will allocate the vect, i.e. the header info, on the stack, but the elements on the free store ("heap"). These elements are allocated in a contiguous memory. When the vector is full, reallocation happens.

1. vector<Type> \*vect = new vector<Type>;

allocates everything on the free store.

1. vector<Type\*> vect;

will allocate the vect on the stack and a bunch of pointers on the free store, but where these pointers (vector elements) will point, is determined by how you use them (you could point element 0 to the free store and element 1 to the stack).

* List
* This denotes doubly linked list.
* Elements can be added to the front, last, or even in between (before a particular element) in a list.
* Traversal is slow compared to vector but resizing, deletion and insertion are fast.
* For singly linked list, we have forward\_list.
* Sets
* Self-balancing binary search trees.
* Elements are always ordered.
* Elements cannot be repeated or modified. They can be removed and re-added with a modified value.
* Unordered sets
* Implemented via hash tables. Sets differ from maps as sets don’t have any value associated with their elements (or keys). They are mainly used for checking presence of an element.
* Unordered map
* bucket\_count and bucket\_size in unordered map tells us the total number of buckets and elements in a particular bucket respectively.
* Priority queue in C++
* Container used to store elements is vector.
* make\_heap, push\_heap, and pop\_heap operations are used to arrange elements according to priorities.
* A custom comparator can be passed as template argument.
* priority\_queue<node, vector<node>, Comparator> pq;

class CompareTime {

public:

bool operator()(node& a, node& b)

{

}

};

* deque

Double ended queues. Unlike vectors, their storage may not be continuously allocated, but they keep information about it internally so as to facilitate O(1) search through [] operator. [deques](http://www.cplusplus.com/deque) are not guaranteed to store all its elements in contiguous storage locations: accessing elements in a deque by offsetting a pointer to another element causes undefined behavior. Unlike vectors, elements in a deque can be inserted in front as well as back.

Can be implemented as vector of vectors. O(1) amortized analysis can be done for finding an element.



For interview purpose, it can be implemented by circular array or doubly linkedlist.