## C++ lower\_bound and upper\_bound for vectors

* If there are duplicate values inside a vector, C++ lower\_bound will find the 1st occurrence of such element. C++ upper\_bound will do the exact opposite, find the last occurrence of such element
* Note, the upper bound is always at an index one higher than it should be (since it’s finding the upper bound). This is assuming the vector is zero-indexed, otherwise, it is 2 higher and lower bound is 1 higher
  + If the element doesn’t exist in the set, then lower bound will do a rightBinarySearch, find the index of the element that is slightly bigger
  + Upper bound will do a leftBinarySearch + 1, it will find the index of the element that is slightly smaller and basically be one too far on the right, so make sure you – 1 the upper bound result



## Vectors

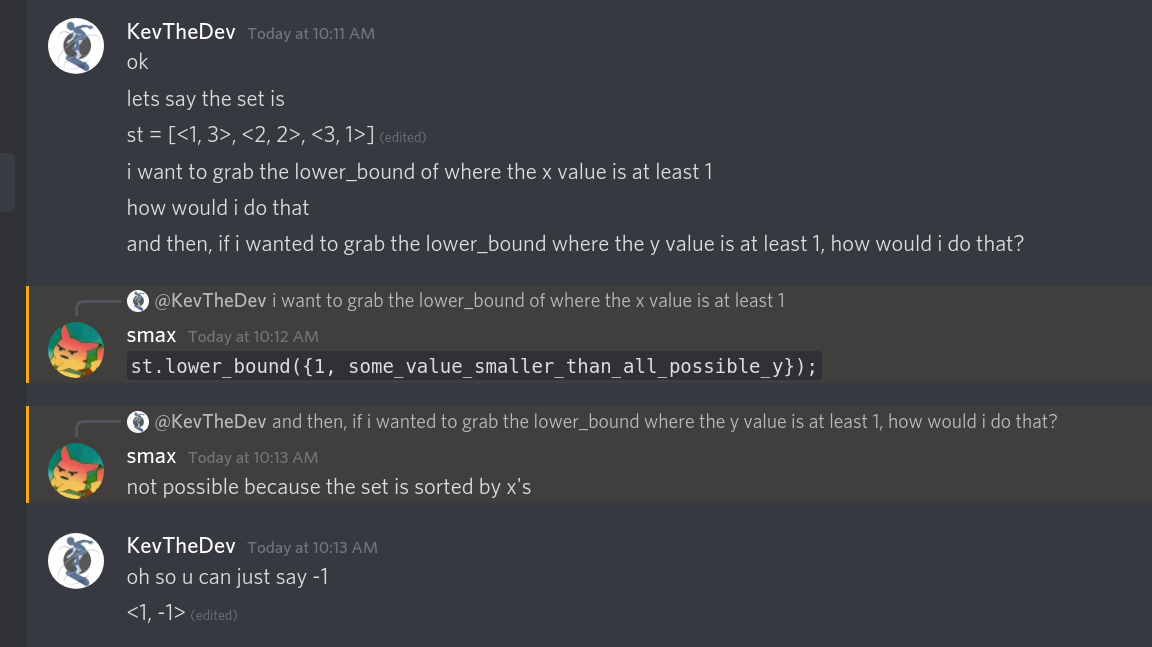
* Vectors have a vector.pop\_back() that deletes the last element in O(1)

## Deque

* A stack and a queue together

auto ptr = &a; ← Stick in an & in front of variable name for pointer, or if in front of a parameter in a function stick it in there as a reference param

\*ptr == a; ← Stick a \* in front of a pointer to dereference the pointer into the original value.



Typical priority queue construction: prioritiy\_queue<ll> pq;

priority\_queue<T, vector<T>, cmp> pq;

For example, if you want to do a priority queue of pair<ll, ll> but pop out the smallest pair.f instead of the greatest pair.f, create the priority queue like this:

priority\_queue<pair<ll, ll>, vector<pair<ll, ll>>, greater<pair<ll, ll>>> pq