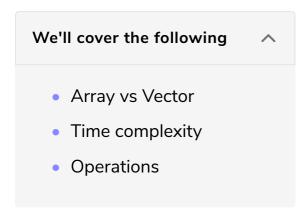
Vectors

In this lesson, we'll compare vectors with arrays and see how to use C++ STL Vector.



Array vs Vector

Many times, you need a structure like an array that is dynamic in size. Vectors are dynamic arrays. Vectors, similar to an array, have contiguous memory allocation so that random access is O(1) for vectors as well.

One way to implement vectors using an array is to copy the array to a new array of double the size every time the first array is full. As discussed in the complexity analysis chapter, the amortized time complexity is O(1) for inserting elements in such a case.

Time complexity

Time complexity for vector operations are the same as arrays:

- Inserting at end O(1)
- Inserting in between O(N)
- Deleting last node O(1)
- Deleting other nodes O(N)

Operations

Here are some operations on vectors that we will be using very frequently. Read this for a complete documentation on vectors.

Please make sure you understand important vector methods and how to deal with iterators. This course will not go over these C++ concepts.

```
vector<int> v; // new empty vector
vecotr<int> v(5); // new vector of size 5
vecotr<book boolean> v(5, true); // new vector of size 5 with all values initialize
d to true

v.size(); // get size
v[i]; // access ith element

v.push_back(x); // insert x at end of vector
v.pop_back(x); // delete last element

v.begin(); // get iterator to beginning
v.end(); // get iterator to end (theoretically, the element after the last element)

v.erase(v.begin() + 4); //delete 4th element

sort(v.begin(), v.end()) //sort vector
reverse(v.begin(), v.end()) // reverse the vector
```

In the next lesson, we'll see a solved problem about reversing a subarray of an array.