std::set, multimap, multiset

std::set is somewhat equal to map<T, void>

multiset is somewhat equal to map<T, int>

In multimap, instead of operator[], you use lower_bound, equal_range, etc.

HashTable, unordered_map

We define a function Hash : A -> B (B == int), to encode an element of hashTable. We won't hash to be:

- Easily and effectively computable
- Want the resulting indecies to be equally (in terms of probability) distributed on our array

Finding good hash functions

Ideal Hashing

When we know all the elements to come, we can calculate the ideal hash functions to do everything in pure O(1)

Polynomial Hashing

We hash string, by looking at it as a polynomial:

s = a_0 a_1 ... a_n -> a_0 + a_1 * p + a_2 * p^2 + ... a_n * p^n

Operations

- find - O(1) (on average, considering all possible values)

Note: std::map ++ of iterator works in O(1) (not amortized, but kinda)

- insert
- erase
- remove
- rehash (reserves the amount of memory, needed to insert the needed number of elements without rehashing (might reserve a bit more))
- operator[]
- at
- reserve
- load_factor
- max_load_factor

When we insert, we can encounter collisions. Here is how we deal we them:

Open addressing

If a[Hash(x)] has an element we try the element after it, and go on until we find an empty spot.

The strategies of finding the spot:

- Linear probing (Hash(x) + i * b)
- Quadratic probing (Hash(x) + i * b + i^2 * a)

Chained HashTable

Each spot is a list of elements with the same hash. The length of list is, on average, small, so we are OK.

Implementation

First try std::vector of std::forward_lists, but the iteration of hash_table is slow

Second try We hold everything in one linked list, and in a slot, we hold the pointer to the first element in the block of elements in list

The node struct know looks like:

```
struct Node {
    pair<const Key, Value> kv;
    size_t hash;
};
```

How do we erase? For that we need to hold the pointer to the last element of the previous bucket in the slot (for when we need to erase the first element of the bucket).

When rehashing, we "splice" the element we need to move from the current list

Exception safety

Here are our template args:

template <Key, Value, Hash, Equal, Allocator>

The problems are with: Hash, Equal, and allocator classes, though Hash and equal are easy to handle.