Data Structures with C++ : CS189

Lecture 12-1: Hash

Recap

- We used the idea of graphs to solve some really cool real world problems
  - HasLoop - OS Deadlock deadlock detection
  - DJ - Pathfinding
  - Max Flow - Measure internet bandwidth
  - Primm - Optimize set of connections
- Graphs are a technology, not an STL container

Recap: Sets

- If you try to search a List, that is O(n) because your only choice is to just travel through all of them
- A Set (with its BST) gave us O(log n) because it kept the data sorted internally and each decision cut off half the tree
  - And a Map is just a Set with K and V instead of T
- Can we get faster?

Hash

- We know that grabbing a specific item from a Vector is O(1) if you already know which one you want
    - Array indexes
- Hash technology means to process a piece of data in to a set array index using one of many different algorithms
- An unordered_map is the STL name
    - Hash is the technology

The Hashing Function

- Inside an unordered map, there is a function that takes a T and returns an int
- This function is a functor belonging to T because how to hash something is different for each kind of data
- The hash-or has two goals
    - Produce the same integer output for any T input
        - Otherwise what you put in you can't get out
    - Try to produce as many different integer outputs as it can
        - If the hash function returns "7" for every T, it actually produces a List for O(n)

Result

- The algorithm is:
  - Give your T to the hashor
  - It is processed in a unique way and gives back an int
  - Use that int as the index into our internal array
    - Using "% size" to not go out of bounds
  - If there is something already in that spot we have two options
    - Find a different spot using a consistent algorithm
    - Store a list of T in the array position (referred to as a bucket usually)
  - The pseudo-randomness of the hashor is why we are called "unordered" while Set and Map are ordered

Creating a Hashor

- Remember the goal of the hashor is
  - Get the same result for the same input every time
  - Try get as many different results as possible over all potential data
- Division
  - Take a number from the object that never changes, and divide it by the table size
    - Making the table size a prime number helps spread out the answers
    - This is actually always the last step of any hash because it keeps the hash result from going out of bounds.

Folding

```
// A good one I've seen for
strings:

int X = 0;
string S = "Ohai.";
for( int i = 0; i < S.length(); i++)
        X = asciiOf(S[i]) + 37 * X;
return 0;
```

- What if the only unchangable number in the object is from 1-5. The hashor won't be able to spread them out well.
- Take any other unchangable part of the object and shuffle it in some way that makes the results as random as possible
  - For a string, take four characters (int not byte because of ascii codes) and add or xor them with the next four letters until you've done all letters
  - For an int, break the digits into 3 digit blocks and sum them. Or sum their reverses. Or reverse every other block. Or square each, mod them by table size, sum them all, add every other number together, multiply by a large prime number, add 7, remove all the 6's, and boy howdy can you

Unchangable

- The reason I keep stressing this is imagine the unique data I pick for a Student is their birthday.
    - 8232020 % 19 = 4
    - Birthday never changes, your record is always at bucket 4
- But what if I use Age
    - (39 * 7) add last to every digit % 19 = 17
    - (40 * 7) add last to every digit % 19 = 14
- When a student has a birthday, you wouldn't be able to find their record again.

Shifting / Bit Manipulation

- Start with the first letter, or the first byte of a number
- Flip the second and fifth bit
- Shift it two bits left
  - Shifting left is actually the same as * 4
- XOR it with the next byte, and repeat from the flip step
- Mangled and convoluted is good. We just need consistent and diverse.
- Perfection would be every possible input comes out with a unique number
  - After the % table size of course

End