CS280 – HashMaps
February 29, 2016

http://azrael.digipen.edu/~mmead/www/Courses/CS280/Hashing-1.html

- We can't do better than O(log(N)) using comparisons
- Hashing lets us have O(K) access
    - HUGE!!

Time vs. Space Tradeoff

- The basic idea is to associate a nuber with the data
    - Lets you do constant array lookup
- If universal set has large range, that is how many elements must be in the array
- Any number of inputs – gives you one output
    - Definition of a hash function
- Hashing is jumbling the data
- Perfect hashing means there are no collisions
    - In practice, this is usually not possible
- Want to distribute data uniformly across the array
- *return 0; //we will be given this for a test case in the driver*

Collision Resolution by Probing

- Probing becomes inefficient but typically there are few collisions / probes requred
- Open-addressing: Data stored directly in the array
- Closed-addressing: Array stores pointers to data
- Want to use prime numbers for hashing

- Using modulus (circular array) we wrap around when probing "past the end"
- Don't worry about collisions since they are so unlikely, tons of other things could go wrong before a collision causes a performance problem
- Biggest problem on hash assignment is keeping track of probes
- *probes++; //have this in one place only, just on one helper function. Otherwise it will be hard to find where you missed an increment*
- Have a lookup helper function
- Insert and lookup in the same direction (front to back)
- When load factor gets above a certain factor, we grow the table to avoid collisions
- More space between elements == less probes
- Rule of thumb, don't let the table get above 2/3 full
- Implementing a has table is easier than a full functioning advanced tree structure