Module 4

**

Hashing

- An algorithm that uses a function (hash function) to map large datasets (variable length), called keys, to smaller datasets of fixed length
- Hash Table (hash map)
- A data structure that uses a hash function to efficiently map keys to values
- Very efficient search and retrieval
- Used in many computer software
- Associative arrays
- Database
- Indexing
- Caches
- Sets

Hash Table Example

- Phone numbers
- 66752378
- 68744483
- Hash function
- h(x) = x % 997;
- x = phone number
- We must store the key values

Hash Functions and Hash Values

- Assume a hash table of size N
- Keys: used to identify the data
- Hash function: used to compute a hash value
- Hash value (hash code)
- Computed from the key using hash function to get a number in the range 0~N~1
- Used as the index (address) of the table entry for the data
- Regarded as the “home address” of a key
- Goal: address are different and spread evenly
- When two keys have the same hash value – collision

Collision

- A hash function may map different keys to the same slot
- Many to one mapping and not one to one
- E.g. 66754372 hashes to the same location of 66753278
- Two keys have the same hash value

Good Hash Function

- O(1)
- Scatters keys evenly
- Less collisions
- Less slots

Bad Hash Functions

- Digit selection
- E.g. choose the 4th and 8th digits of a phone number
-

Perfect Hash Functions

- One-to-one mapping between keys and hash values
- Hence, no collision occurs
- Possible only if all keys are known
- Applications
- Compiler and interpreter search for reserved keywords; shell interpreter searches for built-in commands
- Example
- GU gperf is a freely available perfect hash function, written in C++. It automatically constructs perfect functions from a user supplied list of keywords

Uniform Hash Functions

- Distributes keys uniformly in the hash table
- If keys are uniformly distributed in [0, X), map them to a hash table of size m (m < X) using hash function
-

Division Method

- Map into a hash table of m slots
- Use the modulo operator to map and integer to a value between 0 and m-1
- (n % m) = remainder of n divided by m, where n and m are positive integers
- hash(k) = k % m
- One of the most popular methods

A good choice of m

Multiplication Method

Examples

Collision resolution techniques

- Open hashing
- Closed addressing
- Separate chaining
- Use a linked list to store collided keys
- Always insert at the beginning (or back) of the list
- Find the address in the hash table via hash function
- Search across the linked list one at at time
- Performance
- insert(key,data)
- O(1)
- find(key)
- O(1+alpha) on average
- delete(key)
- O(1+alpha) on average
- alpha=number of keys/capacity
- If alpha <= constant, complexity of all three operations: O(1)
- Address given to a key is fixed
- This is closed addressing
- Types of questions
- Give some random numbers and a hash function
- Which index will have longest chain
- What does the hash table look like
- Which indexes will have chains
- Which indexes will not have chains
- Closed hashing

- Open addressing
- Keys are open to be readdressed
- Keys are closed to be put into that hash address if full
- Linear probing
- Inline poke
- hash(k) = k mod 7, (ie, table-size (m) = 7)
- Worry about size of hash table
- Note: 7 is prime
- Collision
- Scan forward for next empty slot
- Wrap around after reaching last result
- Example
- hash(k) = k mod 7
- Given m = 7
- Draw hash table with indexes 0 to 6
- hash(25) = 25 mod 7 = 4
- Put 25 in the 4th index
- hash(15) = 15 mod 7 = 1
- Put 15 in the 1st index
- *,15,,,25,,*_
- hash(1) = 1 mod 7 = 1
- Collision
- Look for next empty slot
- Put 1 in the 2nd index
- hash(35) = 35 mod 7 = 0
- Insert 35 at 0th index
- hash(50) = 50 mod 7 = 1
- Collision
- Look for next empty slot
- Look for next empty slot
- Insert 50 at 3rd index
- Search 50
- If numbers inserted in a certain sequence using this hash function using linear probing, if i am searching for 50, how many probes will I get
- 50 mod 7 = 1
- Should be at index 1, its not at index one
- Keep searching till you reach the index
- Which is index 3
- Found after 3 probes

- If we find an empty slot (p-slot)?, we stop probing
- The empty slot does get probed
- Problems
- Primary clustering
- Cluster: a collection of consecutive occupied slots
- Primary cluster: cluster covering home address of a key
- Linear probing can create large primary clusters that increase the running time of operations
- Quadratic probing
- Probe sequence
- (hash(key) + k^2) % m
- K is the number of collisions
- Example
- Problems
- Double hashing
- Reduce secondary clustering, by using a second hash function to generate different probe sequences for different keys
- Number of collisions is not dependent on which index the collision is at
- Example
- Problems
- Solution

Rehashing

**