ID1020 2018-09-19 #8

Symbol tables

- Insert a value with specified key
- Given a key, search for the corresponding value.

Ex. DNS-lookup

Associative memory (CAM - Content Adressable Memory)

Symbol tables also known as maps, dictionaries, associative arrays.

API: put, get, contains, delete, keys

Conventions
- null not allowed
- return null if key is not present
- overwrite old value when using same key

Value type: Any generic type

Key type: several natural assumptions
- Assume keys are Comparable, use compareTo().
- Assume keys are any generic type, use equals() to test equality.
- Assume keys are any generic type, use equals() to test equality; use hashCode() to scramble key. (built-in to Java)

Best practices: Use immutable types for symbol table keys.

Immutable in Java: Integer, Double, String, java.io.File, ...
Mutable in Java: StringBuilder, java.net.URL, arrays, ...

Equality test
equals:
- Reflexive: x.equals(x) is true.
- Symmetric: x.equals(y) iff y.equals(x).
- Transitive: if x.equals(y) and y.equals(z), then x.equals(z).
- Non-null: x.equals(null) is false.

Default implementation: (x==y) (tests if x and y is same object

Custom implementations: Integer, Double, String, java.io.File, ...

User-defined implementations: Some care needed

- Reference equality
- Check against null
- Check type/cast
- Compare each significant field
  - For arrays, check each entry

Frequency counter — value is a counter for key

Sequential search in a linked list $O(N)$
Ordered array of key-value pairs
Rank helper function How many keys < key? (Binary search)
Good for many searches, bad for many inputs
Search $O(\log N)$ Insert $O(N)$

Ordered symbol table API:
min, max, floor, ceiling, rank, select, deleteMin, deleteMax, size(lo, hi)
keys(lo, hi)

Binary search trees

A binary search tree is a binary tree in symmetric order

Search: If less, go left; if greater, go right; if equal, search hit.
Insert: If less, go left; if greater, go right; if null, insert.

Quicksort-correspondance

Floor / Cieling

Rank - store size of subtree in each node

Inorder traversal
- Traverse left subtree
- Enqueue key
- Traverse right subtree

Deletion
- Lazy approach, set node value to null (tombstone)
- Delete minimum
  - Go left until finding a node with a null left link
  - Replace that node by its right link.
  - Update subtree counts
- Hibbard deletion
  - 0 children - Delete by setting parent link to null
  - 1 child - Delete by replacing parent link
  - 2 children - find successor and delete minimum in right subtree
  - put successor in place of node to delete
  - Not symmetric