Hash Tables

Bryce Boe

2013/08/20

CS24, Summer 2013 C

Outline

- Lab 8 Solution
- Hash Tables

Hash Tables

- ADT that has expected running time of O(1) for all operations
- How can we do this?
  - Trade space for running time

Hash Function

- Converts a key into an index that fits inside of the hash-table array
- *Ideally* should be uniformly distributed

Collisions

- What should we do when collisions occur?
  - Linear Probing
    - new_hash = (old_hash + c) % max_size
    - Can result in clustering, but better **locality of reference**
  - Quadratic Probing
    - new_hash = (old_hash + $c*i^2$) % max_size
    - May not examine all buckets
  - Buckets and chaining
    - Store a linked list at each location to store collisions in

Deletions and probing

- How can we handle deletions of items when performing probing?
  - Mark the “empty” space as deleted
    - Frees up the space and does not break previous entries
    - Becomes inefficient overtime
      - Acceptable due to re-sizing the hash table