CSC 220 Data Structures

Maps and Hash Tables II – Collision Handling
Parkland College Spring 2016
20160316
Collision Handling

• inevitable hashing and compression will give collisions
• collision resolution rules different strategies depending upon underlying implementation
  • chicken and egg, what comes first, the strategy or the implementation
Separate Chaining

- bucket array of containers (Python lists)

- load factor, $\lambda$
  - expected size number of items/size of bucket array
  - if $< 1$, then essential map behaviors are $O(\text{ceil}(\lambda)) = O(1)$
  - called “O(1) expected” complexity
  - *not* the same as amortization
Open Addressing

• single-level bucket array
• handle collisions with additional math
• if collision, find empty spot
• $\lambda$ must be $\leq 1$
  • (otherwise must resize array)
Linear Probing

- if collision, take next available spot
- \( \text{Bucket}[\text{hash(key)} + i \mod N] \)
  - \( i \) starts at 0
- get, set, delete key have to make sure they found correct key \textit{and} value, i.e., they have to \textit{probe} \( O(n) \)

drawback: “clusters”
Other probes

- **Quadratic**
  - \( \text{Bucket}[\text{hash}(\text{key}) + i^{*}2 \mod N] \)
  - secondary clusters

- **Double hashing**
  - \( \text{Bucket}[\text{hash}(\text{key}) + i \times \text{hash2}(\text{key}) \mod N] \)
  - second hash often simpler

- **Pseudo-random**
  - \( \text{Bucket}[\text{hash}(\text{key}) + \text{rand}(i) \mod N] \)
  - what Python dict() uses
  - better at avoiding clusters
Hash Table Efficiencies

- Depends on load factor, $\lambda$, and implementation
- Separate chaining want $\lambda < 1$
  - otherwise non-trivial linear searches in chains
- Open addressing want $\lambda < \text{approx. 0.5}$
  - depends on probe
  - can resize, but then must recompress
- (see table in textbook/notes)
- “Expected” – “we have the correct load factor”
Implementations

- `hash_map_base.py`
  - abstract
  - keeps track of bucket array
  - map length
  - load factor
  - hashing
- `chain_hash_map.py`
- `probe_hash_map.py`