Hashing (continued)

Double Hashing

Several keys hash to same index → **clustering** around that location ***…why bad???***

One strategy is to use **2nd hash function** to try different spots when there's a collision

h1(key) determines which index to try first
h2(key) determines how far away from index to try next

Ex: h1(key) = 200 and data[200] is already taken
h2(key) = 8 so next try data[208], data[216], data[224], etc.

Warning!

- **Make sure you still wrap around the array**
- **Don't come back to original index without having tried other available positions**

Make CAPACITY be a prime number

h1(key) = key % CAPACITY

h2(key) = 1 + (key % (CAPACITY – 2))

Chained Hashing

- Each slot in hash table can hold **more than one entry**
- That is, each slot is **pointer to a linked list** (not a key value)

Average Search Time for Hashing

Load factor f = # entries / CAPACITY (can be > 1 for open-address hashing)

Open Addressing with Linear Probing
avg. # table elements examined = 0.5 * (1 + (1 / (1 – f)))

Open Addressing with Double Hashing
avg. # table elements examined = – ln(1 – f) / f

Chained Hashing
avg. # table elements examined = 1 + (f / 2)