CSC115 Lecture 20

Hashing

Hashing Introduction

Need efficient ways to search for Data

- Linear Search – O(n)
- Binary Search – O(log n)

If implemented perfectly

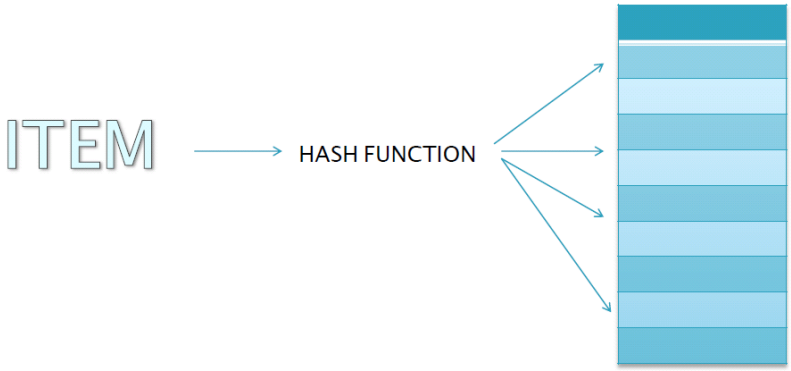
- Hash Functions – O(1)

Realistically – O(n/k)

Important for time critical situations

- 911 calls
- Instruction sets

HASHING



Some Hashing Ideas

Example: Store a complete English language Dictionary

- Every word would have its own cell (in an array or list or table), but what is the relationship between words and index number
- Given a word like morphosis, how do we find its index number?

Small Example: Converting Words to Numbers

Assuming words are made up of 26 letters (ignore capitals for now), 27 if we include a blank

One possibility:

- a is 1
- b is 2
- etc

Then Add the Digits

Convert all letters to numbers then add the digits.

Example:

- cats, c = 3, a=1, t=20, s=19
- 3+1+20+21 = 43
- So, store cats at location 43 in an array

Bounds of Hash

Assume we are restricted to 10 letters words

- Location 0 in array is a blank word
- Location 270 is the word zzzzzzzzzz

Unfortunately there are 50,000 words in the Dictionary, so in this model

- Each location stores 185 words (50000/270)

Another Hashing Idea

Have each character in a word be represented by a power of 27

- Similar to Base 2 (Binary) or Base 10 or Base 16

Then

- cats = $3*27^3+1*27^2+20*27^1+19*27^0$
  = 60337
- So cats would go to location 60337 in Array

And zzzzzzzzzz= 7 trillion

- Hash function is too big!!

A Better Idea

- Need a way to compress the huge range of numbers we obtained into a range that matches the size of a reasonable array
- English language is 50,000 words, lets allocate double this so 100,000.
- Trying to squeeze 0 -7 trillion Down to 0 – 100,000

Idea: Use the modulo operator

A Smaller Example

Squeeze the numbers from 1 to 199
into range 0 – 9

- We have a largeNumber = 1 to 199
- We want a smallNumber = 0 to 9
- The *smallRange* is 10

Thus:
smallNumber = largeNumber % smallRange

Back to our Dictionary (Huge) Example

- We have a hugeNumber = 0 to 7 trillion
- We want a arrayIndex = 0 to 100,000
- The *arraySize* = 100000

Thus,
arrayIndex = hugeNumber % arraySize

Collisions

The price paid for squeezing a large range into a small range??
– collisions can occur

- Collision: Two values that to HASH to the same Array location

Does this make the scheme no good?

- No! We can deal with collisions in a few ways

Open Addressing

During an attempt to insert a new item . . .

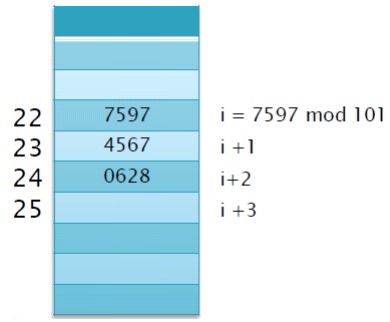
- Hash table location is already occupied

Solution: PROBE for an empty location

- 3 methods for doing this (next slides)

Linear Probing

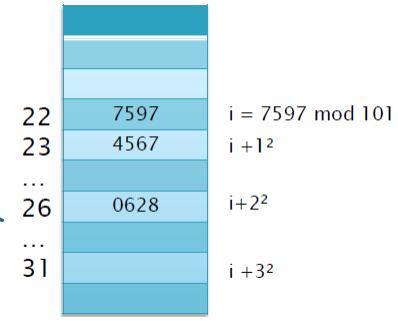
- Find the original hash location
- search the array sequentially for an available location
- Insert element



Quadratic Probing

- Find the original hash location
- search the array quadratically for an available location
- Insert element

Actually: We will not use or test Quadratic Probing this term… Its just included for completeness!



External Chaining

Avoids Collisions all together:

- The hash table is an array of linked lists