CSCI 2270
Data Structures and Algorithms
Hash Tables

Elizabeth White
elizabeth.white@colorado.edu
Office hours: ECCS 128
Wed 1-2pm
Fri 2-3pm

Administrivia

Last homework on graphs, due Saturday night

Find shortest-hop path between 2 cities

depth first or breadth first search

Lab this week: buffer overrun

Hash tables

Work on key, value pairs of data

Example: social security number + person's name

Imagine storing these pairs in an array of strings

Write the int, string pair into the array slot

What's wrong with this? Wastes space.

How could we do this with a smaller array?

Hash tables

How could we do this with a smaller array?

Use part of the key as the index

First 3 digits of SSN?

Now can store key_value pairs in smaller table

Problem: state school

Last 3 digits of SSN? Better hash function.

Modulus operator: O(1)

Still, collisions when 2 pairs get the same slot of the table array

Resolving collisions

Open addressing: put the colliding data in the next open slot

Problem: now we have to search more of the table

And doing this causes more collisions down the line

Collisions tend to cluster

Run time: one hash function plus checking for the next open slot

Resolving collisions

Double hashing:

If a collision happens, apply a second hash function

e.g., check the next slot n slots away, 2n slots away, 3n slots away…. where n is determined by another hash function

Second hash function should eventually hit every slot in the table (‘relatively prime’)

Reduces clustering, but doesn’t eliminate collisions

Run time: one hash function plus checking for the next open slot

Resolving collisions

Chained hashing: make an array of linked lists

First hash function finds the head_ptr of the correct list for this key

Then we insert the pair into the list at that index

Run time: one hash function plus time to insert into list

Performance depends on size

Hash table's load factor α determines its speed

α = number of items in table/capacity of table (0 to 1)

Low α: fast performance

High α: slows down

Notice that for chained hash tables, α can be > 1

Open addressing average time to add, remove, find: $\frac{\left(1+\frac{1}{(1-\alpha)}\right)}{2}$

Double hashing average time: $\frac{-\ln(1-\alpha)}{\alpha}$

Chained hashing average time: $1 + \frac{\alpha}{2}$