Design and Analysis of Algorithms I

Data Structures

Hash Tables and Applications

Hash Table: Supported Operations

Purpose : maintain a (possibly evolving) set of stuff. (transactions, people + associated data, IP addresses, etc.)

Insert : add new record

Using a "key"

unlike most data structures and algos, hash tables don't always work. For pathological input data, hash tables are sure to fail.

Delete : delete existing record

AMAZING GUARANTEE

All operations in O(1) time ! *

Lookup : check for a particular record
(a "dictionary")

* 1. properly implemented   2. non-pathological data

Even though python dicts are hash tables, don’t mix hash tables with literal
meaning of english dicts
In english dicts, elements are ordered and binary search can be applied.
There is no such order here

hash table is not suited if you want to find
the min, max element. for that heap/search tree is better

Application: De-Duplication

Given : a “stream” of objects.

Linear scan through a huge file

Or, objects arriving in real time

Goal : remove duplicates (i.e., keep track of unique objects)

-e.g., report unique visitors to web site

- avoid duplicates in search results

Solution : when new object x arrives

- lookup x in hash table H
- if not found, Insert x into H

Application: The 2-SUM Problem

Input : unsorted array A of n integers. Target sum t.

Goal : determine whether or not there are two numbers x,y in A with

$x + y = t$

Naïve Solution : $\theta(n^2)$ time via exhaustive search

Better : 1.) sort A ($\theta(n \log n)$ time) 2.) for each x in A, look for $\xleftarrow{\theta(n \log n)}$ t-x in A via binary search

$\theta(n)$ *time*

Amazing : 1.) insert elements of A into hash table H 2.) for each x in A, Lookup t-x $\xleftarrow{}$ $\theta(n)$ *time*

Tim Roughgarden

Further Immediate Applications

- Historical application : symbol tables in compilers

- Blocking network traffic

- Search algorithms (e.g., game tree exploration)
    - Use hash table to avoid exploring any configuration (e.g., arrangement of chess pieces) more than once

in graph search there is a basic step of - "if node is explored", this is a lookup problem which can be made efficient via hash table

- etc.