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

Delete: delete existing record

Lookup: check for a particular record

} using a "key"

(a "dictionary")

AMAZING GUARANTEE

all operations in $O(1)$ time!*

* ① properly implemented ② non-pathological data

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

Tim Roughgarden

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 $x+y=t$

Naive 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 t-x in A via binaray search

$\Theta(n \log n)$ →

$\Theta(n)^*$ time

Amazing: ① insert elements of A into hash table H ② for each x in A, Lookup t-x in H → $\Theta(n)^*$ time

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
- etc.