EECS 281 DATA STRUCTURES AND ALGORITHMS

Lecture 5: Dictionary ADT and Hashing
Recurrence Relations

Dictionary ADT

How do you use a dictionary?

Used where you need to do some sort of table lookup:
* search for a key in a table
* the key is usually associated with some data/value of interest

Also known as associative array

Why search for key instead of just searching for the data?

Dictionary ADT

Key space is usually more regular/structured than value space, so easier to search

Dictionary entry is a
`<const key_type, data_type>` pair
- for example, `<title, mp4_file>`
- `<"Avatar", avatar.mp4>`

Normally associate a given key with only a single value or a pointer to data

Dictionary is optimized to quickly add `<key, data>` pairs, retrieve data by key

Types of Dictionary

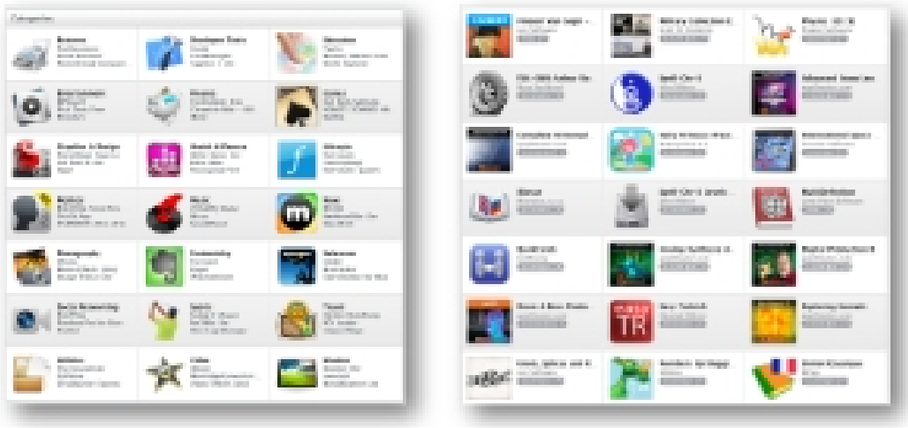
Whether items are grouped by some category such as by subject, by popularity, chronologically, etc.
* unordered
* ordered

Whether items are listed by a collating sequence of the key, e.g., numerical, alphabetical
* unsorted
* sorted

Adding entries into an ordered (sorted) list must retain the ordered (sorted) property of the list

The AppStore

What type of dictionary do we see at the AppStore?



Unsorted Dictionary Runtimes

Implementation	Search	Insert
Arrays	$O(?)$	$O(?)$
Linked Lists	$O(?)$	$O(?)$
Hashing (amortized)	$O(1)$	$O(1)$

Hashing

Access table items by their keys in relatively constant time regardless of their locations

Main idea: use arithmetic operations (hash function) to transform keys into table locations

- the same key is always hashed to the same location
- such that insert and search are both directed to the same location in $O(1)$ time

Hash table: an array of buckets, where each bucket contains items assigned by a hash function

Hashing Example

In a text editor, to speed up search, we build a hash table and hash each word into the table

Let hash table size (M) = 16

Let hash function ($h()$) = (sum all characters) mod 16

- by "sum all characters" we mean sum the ASCII (or UTF-8) representation of the character
- for example, h("He") = (72+101)%16 = 13

Let sample text be the following N=13 words:

"He was well educated and from his portrait a shrewd observer might divine"

Hashing Example

(sum all characters) mod 16

He → 13
was → 11
well → 4
educated → 15
and → 3
from → 4
his → 4
portrait → 5
a → 1
shrewd → 13
observer → 8
might → 9
divine → 15

$N = 13, M = 16$

0		
1	a	
2		
3	and	
4	well from his	■
5	portrait	
6		
7		
8	observer	
9	might	
10		
11	was	
12		
13	He shrewd	■
14		
15	educated devine	■

Collision and Collision Resolution

Collision occurs when the hash function maps two or more items—all having different search keys—into the same bucket

What to do when there is a collision?

Collision-resolution scheme:

- assigns distinct locations in the hash table to items involved in a collision

Separate Chaining

A collision resolution scheme that lets each bucket points to a linked list of elements

- insertion:
  - compute $k = h(key)$
  - prepend to k^{th} bucket in $O(1)$ time
    (but may need to check for duplicates)
- search:
  - compute $k = h(key)$
  - search in k^{th} container (e.g., check every element)

Separate Chaining