DSE 2256 DESIGN & ANALYSIS OF ALGORITHMS

Lecture 31

Space and Time Trade-Offs

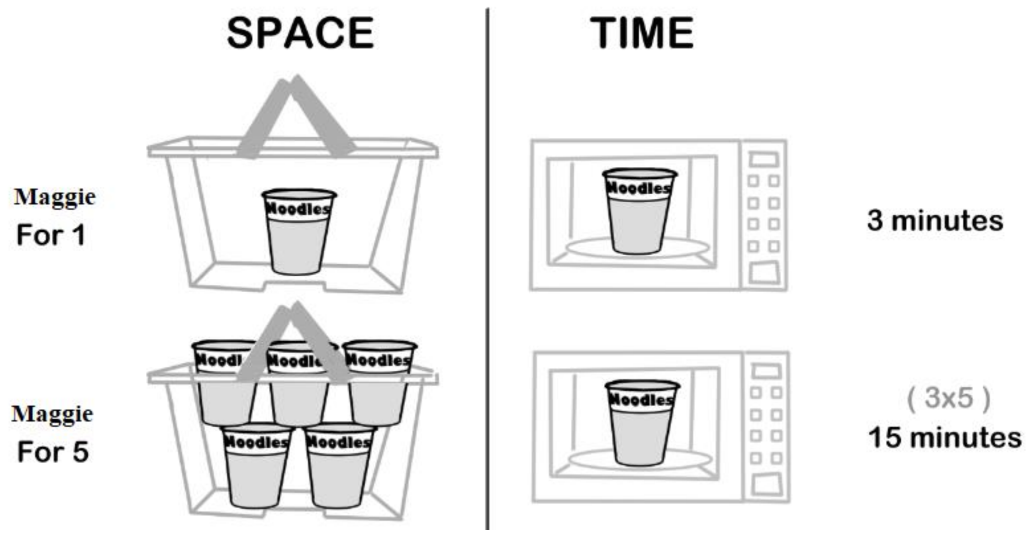
Prestructuring:

Hashing

Instructors:

Dr. Savitha G,
Assistant Professor, DSCA, MIT, Manipal

Dr. Abhilash K. Pai,
Assistant Professor, DSCA, MIT, Manipal



HASHING

Dictionary:

- Dynamic-set data structure for storing items indexed using *keys*.
- Supports operations Insert, Search, and Delete.

001	002	003	004
Alex	Bob	Rose	Sofia

Applications:

- Symbol table of a compiler.
- Memory-management tables in operating systems.
- Large-scale distributed systems.

Hashing:

- Effective way of implementing dictionaries (via Hash Tables).
- Works by prestructuring the input.

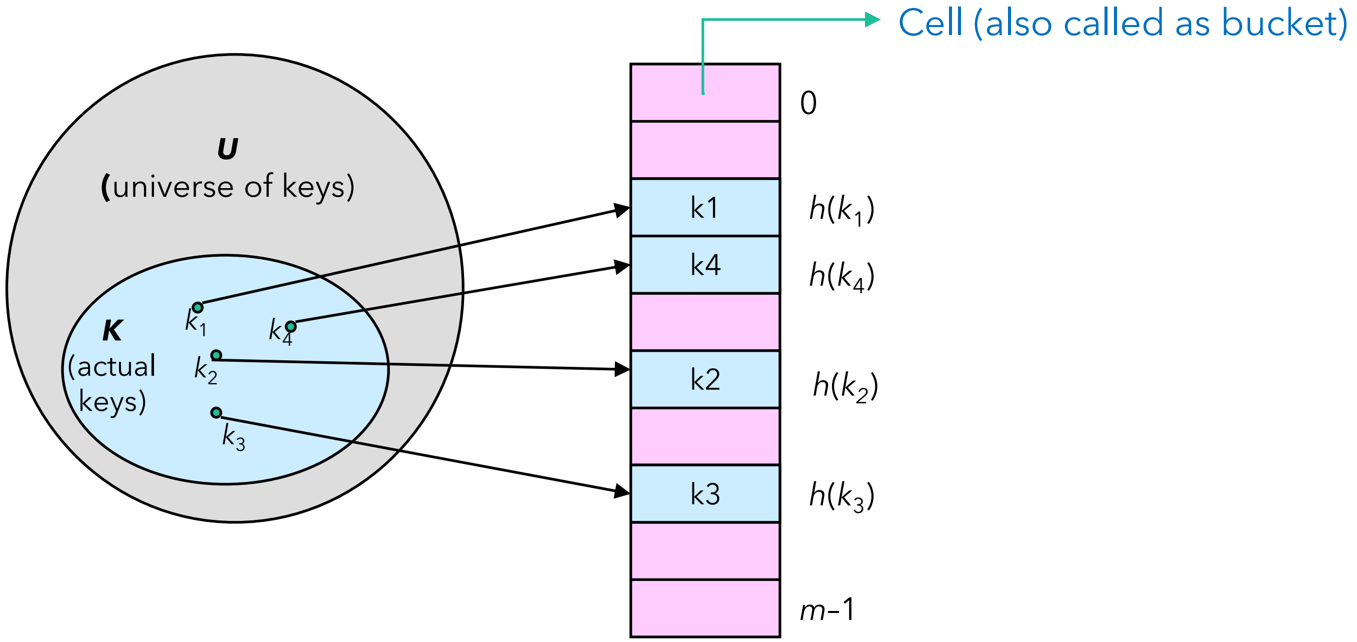
To **search** for a key inside an array:
- **O(n)** time using Linear search
- **O(logn)** time using Binary search

Using Hashing searching for a key can be done in constant time i.e., **O(1)**

Hash tables and hash functions

- The idea of hashing is to map keys (K) of a given dictionary of size n into a table of size m, called the **hash table**, by using a predefined function (h), called the **hash function**,

h : K -> location (cell) in the hash table



Hash Table

Hash tables and hash functions : Example

Example of a Hash function : h(k) = k mod m → hash function by division method (popularly used)

For the keys: **10, 12, 15, 16, 18, 19**

If **m=10**, then the **hash table will be of size 0 to m-1**

- The key **12** is hashed to the hash address **2**
- The key **19** is hashed to the hash address **9**

Hash Table	Hash addresses
10	0
	1
12	2
	3
	4
15	5
16	6
	7
18	8
19	9

Hash Table

Generally, a hash function should:

- be easy to compute
- distribute keys about evenly throughout the hash table

Collisions

If **$h(K_1) = h(K_2)$**, there is a **collision** (i.e., two keys hash to the same location in the hash table)

- Good hash functions result in fewer collisions, but some collisions should be expected

- **Two principal hashing schemes** handle collisions differently:

1. **Open hashing**
   - Each cell is a header of linked list of all keys hashed to it
2. **Closed hashing**
   - One key per cell
   - In case of collision, it finds another cell by

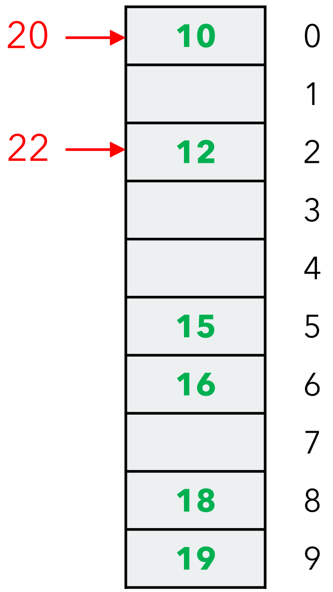
     2. a) Linear probing: use next linearly free cell

     2. b) Quadratic Probing: use next quadratically free cell

     2. c) Double hashing: use second hash function to compute increment

For, h(k) = k mod 10

Keys: **10, 12, 15, 16, 18, 19, 20, 22**



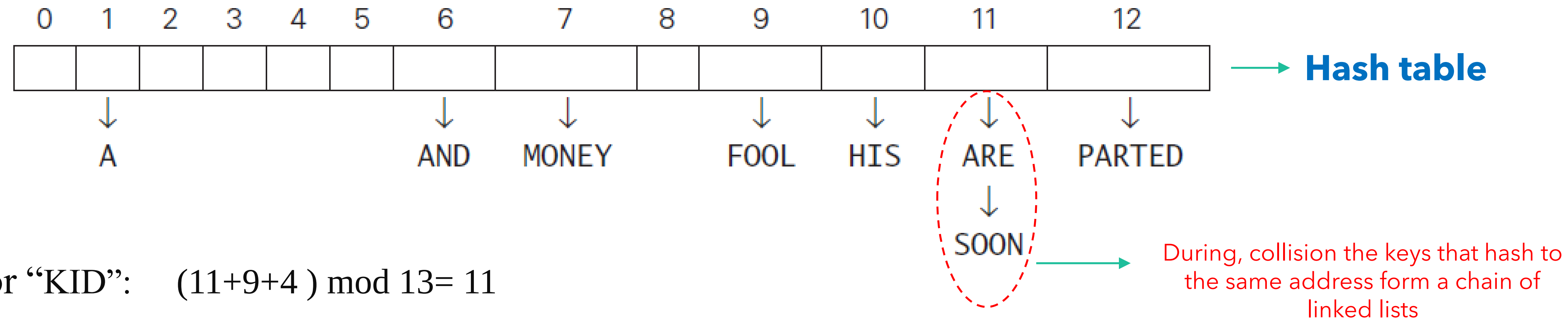
Open hashing (Separate chaining)

- Keys are stored **in linked lists <u>outside</u> a hash table** whose elements serve as the lists headers.

- <u>Example:</u> The keys are: A, FOOL, AND, HIS, MONEY, ARE, SOON, PARTED

  $h(K)$ = sum of K's letters' positions in the alphabet MOD 13

keys	A	FOOL	AND	HIS	MONEY	ARE	SOON	PARTED
hash addresses	1	9	6	10	7	11	11	12



Search for "KID": (11+9+4) mod 13= 11

Open hashing (Separate chaining)

- If hash function distributes keys uniformly, average length of linked list will be **$\alpha = n/m$.**

  This ratio is called ***load factor.***

  - n =no. of keys stored in table , m= no. of slots in table.
  - α = Average keys per slot or load factor =n/m

- For ideal hash functions, the average numbers of probes in successful, *S*, and unsuccessful searches, *U*:

$$S = 1+\alpha/2, \quad U = \alpha$$

- Load α is typically kept small (ideally, about 1)

- Open hashing still works if $n > m$

For searching, insertion, and deletion operations, the time efficiency is :
$O(1+\alpha)$

Closed Hashing (Open Addressing)

- All the keys are hashed in the table without the use of linked list.

- To resolve collision three techniques are used:

  - Linear Probing

  - Quadratic Probing

  - Double hashing

Linear Probing

Checks the cell (probing) following the one where collision occurs.

- If that cell is empty, the new key inserted there.
- If it is occupied, its immediate successor cell is checked.

Size of the hash table

$$h'(x) = (h(x) + f(i)) \bmod m$$

Where f(i) = i

i= 0,1,2 …

Example:

keys	A	FOOL	AND	HIS	MONEY	ARE	SOON	PARTED
hash addresses	1	9	6	10	7	11	11	12

0	1	2	3	4	5	6	7	8	9	10	11	12
	A											
	A								FOOL			
	A					AND			FOOL			
	A					AND			FOOL	HIS		
	A					AND	MONEY		FOOL	HIS		
	A					AND	MONEY		FOOL	HIS	ARE	
	A					AND	MONEY		FOOL	HIS	ARE	SOON
PARTED	A					AND	MONEY		FOOL	HIS	ARE	SOON

Note:

If the end of the hash table is reached, the search is wrapped to the beginning of the array (like a circular array)

Linear Probing suffers from **Clustering**

Quadratic Probing and Double Hashing

Quadratic Probing:

If a collision occurs use a quadratic function to do probing.

$$h'(x) = (h(x) + f(i)) \bmod m$$

Where $f(i) = i^2$

$i = 0,1,2 \ldots$

Double hashing:

If a collision occurs use another hash function to do probing.

$$h'(x) = (h_1(x) + i*h_2(x)) \bmod m$$

$h_1(x)$ is the initial hash function

$h_2(x)$ is the secondary hash function

$i = 0,1,2 \ldots$

For instance:

$h_1(x) = k \bmod 11$

$h_2(x) = 7 - (k \bmod 7)$

Exercise

Q. Given Keys = 3, 2, 9, 6, 11, 13, 7, 12

Given, $h_1(k) = 2k+3$ and $m = 11$

Use division method (mod operation) to store/insert these keys values in a hash table.
In case of collisions resolve them using :

(a) Linear Probing

Note: Use $h'(k) = (h_1(k) + i) \bmod m$

(b) Quadratic Probing

Note: Use $h'(k) = (h_1(k) + i^2) \bmod m$

(c) Double Hashing

Note: Use $h'(k) = (h_1(k) + i*h_2(k)) \bmod m$
Where, $h_1(k)$ is same as defined in the Q.
Use, $h_2(k) = 3k+1$

Thank you!

Any queries?