HASHING FUNCTIONS

Hashing Function

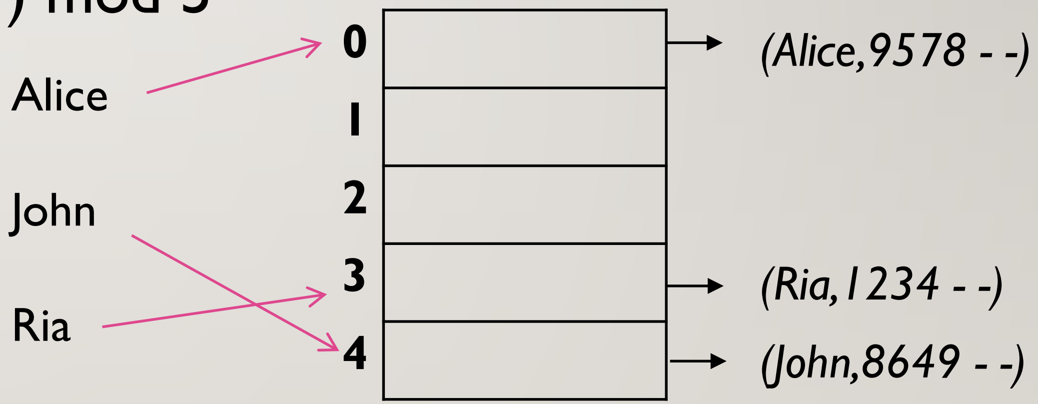
- Hashing function is a function which is applied on a key by which it produces an integer, which can be used as an address in hash table.
- A simple hashing function: $h(k) = k \bmod m$

Properties of Hashing Functions

- Easy to compute
- Uniform distribution
- Less collisions

Hash Table: Example

- **Example:** phone book with table size N = 5
- **hash function** h(w) = (length of the word w) mod 5



- **Problem:** collisions
- Where to store Joe (collides with Ria)

Collisions

- Collisions occur when different elements are mapped to the same cell.
- Keys k_1, k_2 with $h(k_1) = h(k_2)$ are said to collide

What should we do now?

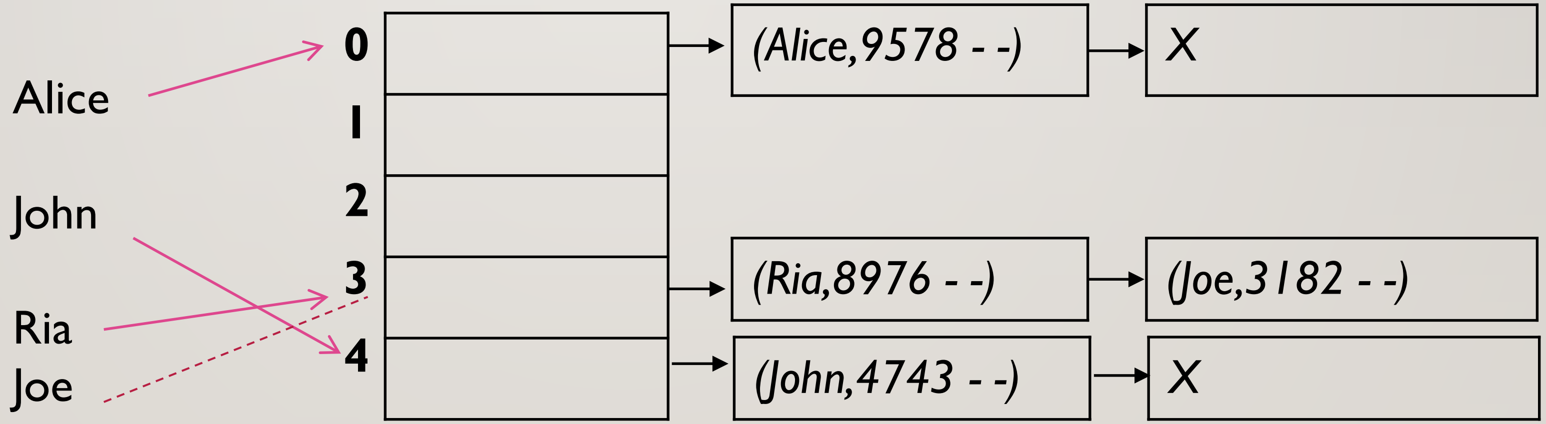
- Find a better hashing algorithm
- Use a bigger table
- Need a system to deal with collisions

Resolving Collisions

- Two different methods for collision resolution:
  - **Separate Chaining**: Use a dictionary data structure (such as a linked list) to store multiple items that hash to the same slot.
  - **Closed Hashing (or *Open Addressing*)**: search for empty slots using a second function and store item in first empty slot that is found.

Separate Chaining

- Each cell of the hash table points to a linked list of elements that are mapped to this cell.
- Simple, but requires additional memory outside of the table



Closed Hashing or Open Addressing

- Open addressing does not introduce a new structure.
- If a collision occurs then we look for availability in the next spot generated by an algorithm.
- There are many implementations of open addressing, using different strategies for where to probe next:

1. Linear Probing
2. Quadratic Probing
3. Double Hashing

Contd..

- Given an item X, try cells $h_0(X), h_1(X), h_2(X), \ldots, h_i(X)$
  - $h_i(X) = (Hash(X) + F(i)) \bmod \textit{TableSize}$
  - $F(0) = 0$
- F is the *collision resolution* function. Some possibilities:
  - Linear: $F(i) = i$
  - Quadratic: $F(i) = i^2$
  - Double Hashing: $F(i) = i*Hash_2(X)$

Linear Probing Example

Index	insert(14) 14%7 = 0	insert(8) 8%7 = 1	insert(21) 21%7 =0	insert(2) 2%7 = 2
0	14	14	14	14
1		8	8	8
2			21	21
3				2
4				
5				
6				

Quadratic Probing Example

	insert(14) $14\%7 = 0$	insert(8) $8\%7 = 1$	insert(21) $21\%7 = 0$	insert(2) $2\%7 = 2$
0	14	14	14	14
1		8	8	8
2				2
3				
4			21	21
5				
6				

Double Hashing

- *Double hashing can be done using :*

  (hash1(key) + i * hash2(key)) % TABLE_SIZE

- First hash function is typically

hash1(key) = key % TABLE_SIZE

- A popular second hash function is :

hash2(key) = PRIME – (key % PRIME)

where PRIME is a prime smaller than the TABLE_SIZE.

Double Hashing Example

	insert(19)	insert(27)	insert(36)	insert(10)	
	19%13 = 6	27%13 = 1	36%13 = 10	10%13 = 10	
0					
1		27	27	27	Collision 2
2					
3					
4					
5				10	
6	19	19	19	19	
7					
8					
9					
10			36	36	Collision 1
11					
12					

Let Hash2(key)=7-(key % 7)

Hash1(10)=10%13=10 (Collision1)
Hash 2(10)=7-(10%7)=4

(Hash1(10)+1*Hash2(10))%13=1(Collision 2)
(Hash1(10)+2*Hash2(10))%13=5

PRACTICE QUESTIONS ON GROWTH OF FUNCTIONS

Q1. Give a big-O notation to estimate the sum of the first n positive integers.

Q2. Give a big-O estimate for the factorial function.

Q3. Give a big-O estimate for the following function:

$$f(n) = 3n \log (n!) + (n^2 + 3) \log n$$

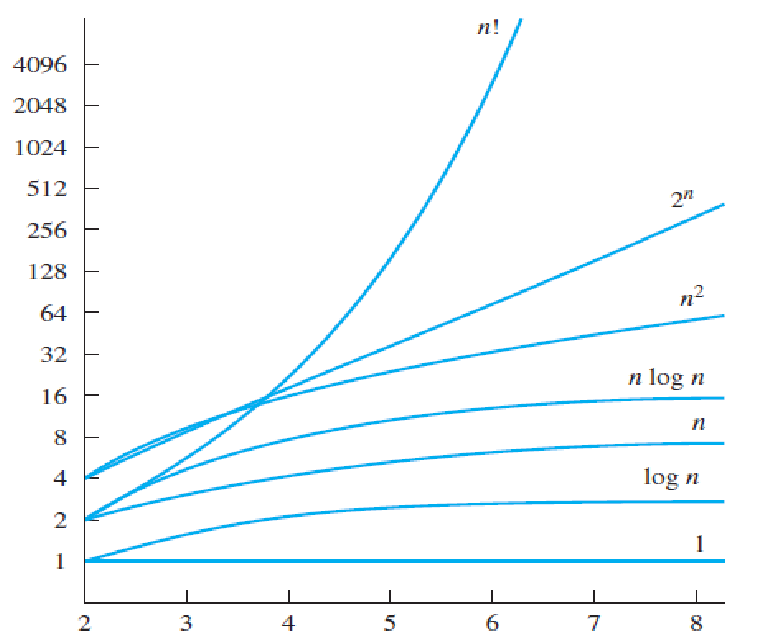
Q4. Give a big-O estimate for the following function:

$$f(x) = (x + 1) \log (x^2 + 1) + 3x^2$$

As mentioned before, big-*O notation is used to estimate the number of operations needed to* solve a problem using a specified procedure or algorithm. The functions used in these estimates often include the following:

1, $\log n$, n, $n \log n$, n^2, 2^n, $n!$

Using calculus it can be shown that each function in the list is smaller than the succeeding function, in the sense that the ratio of a function and the succeeding function tends to zero as *n grows without bound. Figure displays the graphs of these functions, using a scale for* the values of the functions that doubles for each successive marking on the graph. That is, the vertical scale in this graph is logarithmic.



A Display of the Growth of Functions Commonly Used in Big-*O Estimates.*

Thank You