Sorting and Searching

Instructors:
Md Nazrul Islam Mondal &
Rizoan Toufiq
Department of Computer Science & Engineering
Rajshahi University of Engineering &
Technology Rajshahi-6204

Outline

- Sorting
- Insertion Sort
- Selection Sort
- Merging
- Merge-Sort
- Radix Sort
- **Searching and Data Modification**
- **Hashing**

Searching and Data Modification

Searching and Data Modification

- Let S be a collection of data
  - Maintained in memory by a table using **some type of data structure**.
- Searching – Successful or Unsuccessful
- Data Modification – Insert, delete and Update(delete & Insert)

- Searching and Data modification depend on mainly on **the type of data structure** that is used.

Searching and Data Modification

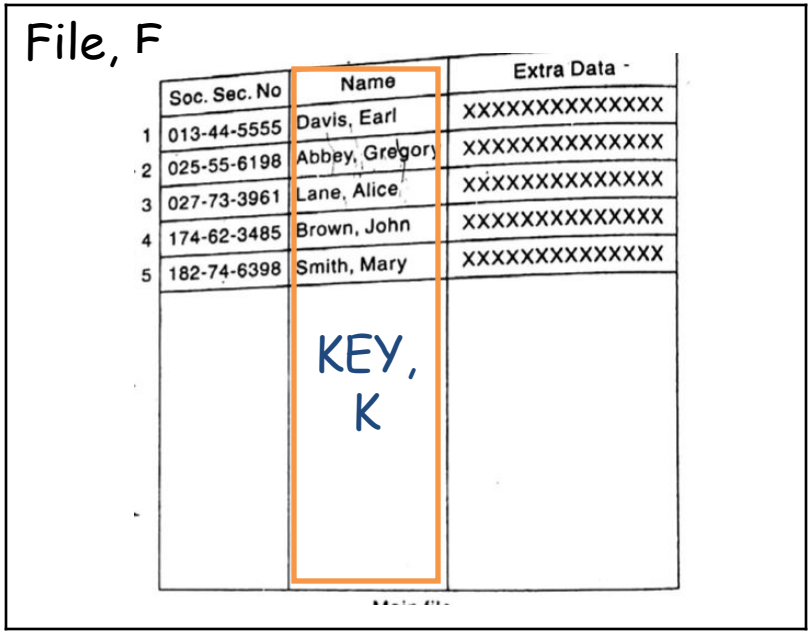
- Sorted Array:
  - Searching Complexity: **O(logn)** – Binary Search
  - Inserting and Deleting are very Slow i.e. **O(n)**
  - **Use**: a Great deal of searching but only very little data modification

- Linked List:
  - Searching Complexity: O(n) – Slow
  - Data Modification: Fast
  - Use: A great deal of data modification.
- Binary Search Tree:
  - Average searching Complexity: O(logn)
  - Data Modification: relatively Fast (since Linked List is used)
  - **Drawback**: Unbalanced tree

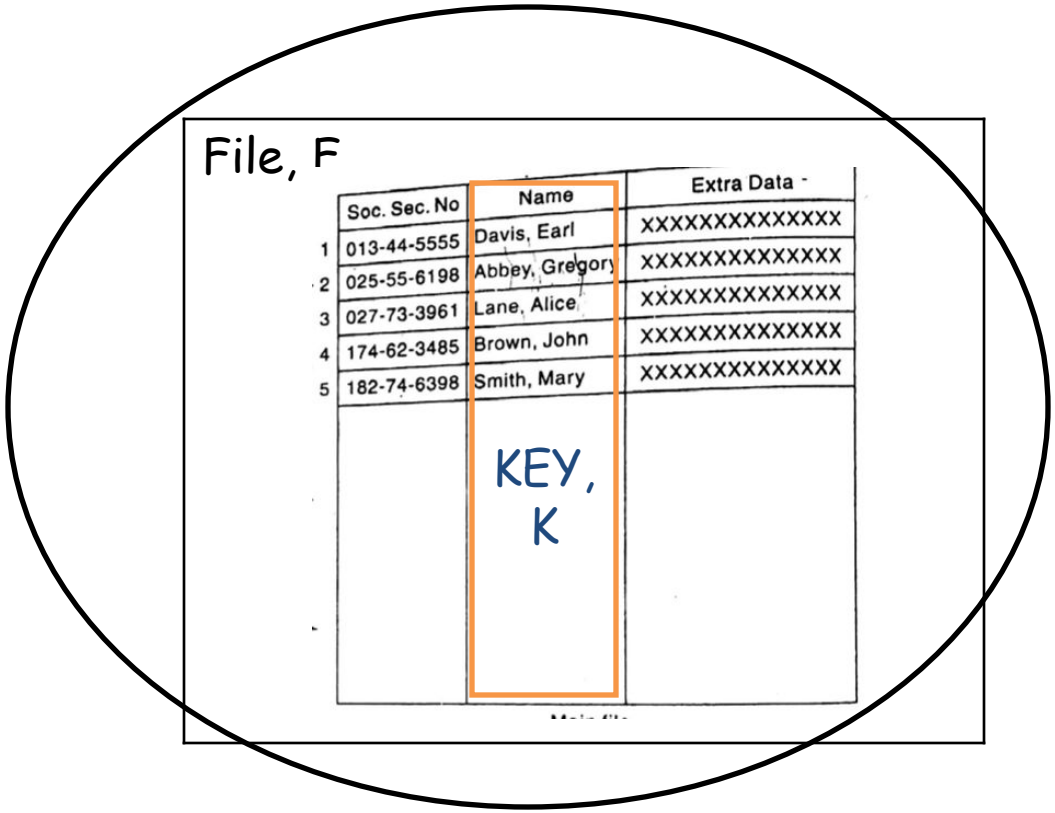
Hashing

Hashing

- Searching time depends on the number **n** of elements in a collection S of data.
- Hashing or Hash Addressing – independent of the number of n



Hashing



- Memory Size m
- Maintain By Table T
- L set of memory addresses of the location in T

- Map K to L

Hashing (Example)

- In a Company
  - 68 employees
  - Assign 4-digit employee number as primary key (K)
  - Key indicates the memory location (L)
  - Searching: No compare, O(1)
  - Tradeoff space for time is not worth the expense.

Hashing

- Idea: Use key to determine the address of a record
- Drawback : waste of memory
- So we cant use the key directly as the address of a record
- We need modification, We can write

$$H: K \rightarrow L$$

- The function H is called a hash function or hashing function.

- **Drawback**: H may not yield distinct values; it is possible that two different keys k_1 and k_2 will yield the same hash address. This situation is called **collision.**
- Some method must be used to resolve it.

Hash Function

- Two Principal criteria considered to select a hash function –
  1. H should be very **easy and quick to compute**.
  2. H should **uniformly distribute** the hash address throughout the set L so that there are **a minimum number of collisions**.

- One technique is to "chop"a key k into pieces and combine the pieces in some way to form the hash address H(k).

Hash Function (Division Method)

- Let total number of key is n
- m is a prime number and m>n
- The hash function H:K□L, H(k) = k (mod m) or H(k)= k(mod m)+1
- H:K□L, H(k)=k (mod m) ,
  - when the range of hash address is [1,m]
- H:K□, H(k)=k (mod m),
  - When the range of hash address is [0, m-1]

Hash Function (Division Method)

- **Example**: Number of Employees - 68

L:	00	01	02	…	99

Map the following Key

3205	7148	2345					

- **Division Method**:
  - Since address starts from 00, we will use H:K□L, H(k) = k (mod m)
  - We chose m less than 99 and must be prime, i.e. m = 97
  - H(3205) = 3205 (mod 97) = 4 , **Store data of 3205 at 04.**
  - H(7148) = 7148 (mod 97) = 67
  - H(2345) = 2345 (mod 97) = 17

Hash Function (Midsquare Method)

- The Key, k is squared.
- Then the hash function H is defined by **H(k) = l,**
  - where l is obtained by deleting digits from both ends of k^2.

Hash Function (Midsquare Method)

- **Example**: Number of Employees - 68

L:	00	01	02	…	99

Map the following Key

3205	7148	2345					

- **Midsquare Method**:
  - 4^{th} and 5^{th} digits are chosen

K	3205	7148	2345
K^2	102 72 025	510 93 904	54 99 025
H(k) or l	72	93	99

Hash Function (Folding Method)

- The Key, k is partitioned into a number of parts, k_1, k_2, … k_r
- Each part has the same number of digits (except the last)
- Then the parts are added together, ignoring the last carry.

$$H(k) = k_1 + k_2 + \ldots + k_r$$

- Another Process:
  - Sometimes, for extra "milling", the even-numbered parts are k_2, k_4, … each reversed before the addition.

Hash Function (Folding Method)

- **Example**: Number of Employees - 68

L:	00	01	02	…	99

Map the following Key

3205	7148	2345					

- Folding **Method**:
  - H(3205) = 32+05 = 37
  - H(7148) = 71+48 = 19 (ignoring last carry 1)
  - H(2345) = 23+45 = 68

  - H(3205) → 32 + 05 → 32+50(reversed) → 82
  - H(7148) → 71+48 → 71+84 → 55
  - H(2345) → 23 +45 → 23 + 54 → 77

Collision Resolution

- Let new record R with key k
- H(k) = l, suppose the memory location l is already occupied.
- This situation is called **collision**.

- **Load Factor**: λ = n/m (small value is better)
  - n = the number of keys in K
  - m = the number of hash address in L
- Suppose we have 24 student and have a table with space for 365 records. H:K□L, H(k) = student birthday
- **Load Factor** = λ = 24/365 = 7%

Collision Resolution

- The efficiency of a hash function with a collision resolution-
  - The average number of probes needed to find the location of the record with a given key k.
- **Two quantities** –

  S(λ) = average number of probes for a successful search

  U(λ) = average number of probes for an unsuccessful search

Collision Resolution
(Open Addressing: Linear Probing and Modifications)

- Suppose that a new record R with key k is to be added to the memory table T.
- But H(k) =h is already filled.
- Resolve the collision:
  - assign R to the first available location following T[h].

Collision Resolution
(Open Addressing: Linear Probing and Modifications)

- Assume table T with m location is cicular, so that T[1] comes after T[m].
- Search for the record R in the table T by linearly searching locations T[h], T[h+1], T[h+2]… until finding R (Successful) or meeting an empty location (Unsuccessful search).

Collision Resolution
(Open Addressing: Linear Probing and Modifications)

- S(λ) = average number of probes for a successful search
- λ=load factor

$$S(\lambda) = \frac{1}{2}\left(1 + \frac{1}{1-\lambda}\right)$$

- U(λ) = average number of probes for an unsuccessful search

$$U(\lambda) = \frac{1}{2}\left(1 + \frac{1}{(1-\lambda)^2}\right)$$

Collision Resolution
(Open Addressing: Linear Probing and Modifications)

- Suppose the table T has 11 memory location and the file F consists of 8 records with the following address:

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Open Addressing Procedure (linear Probing):**

Table T:	_	_	_	_	_	_	_	_	_	_	_
Address:	1	2	3	4	5	6	7	8	9	10	11

Collision Resolution
(Open Addressing: Linear Probing and Modifications)

- Suppose the table T has 11 memory location and the file F consists of 8 records with the following address:

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Open Addressing Procedure:**

Table T:	_	_	_	A	_	_	_	_	_	_	_
Address:	1	2	3	4	5	6	7	8	9	10	11

Collision Resolution
(Open Addressing: Linear Probing and Modifications)

- Suppose the table T has 11 memory location and the file F consists of 8 records with the following address:

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Open Addressing Procedure:**

Table T:	_	_	_	A	_	_	_	B	_	_	_
Address:	1	2	3	4	5	6	7	8	9	10	11

Collision Resolution
(Open Addressing: Linear Probing and Modifications)

- Suppose the table T has 11 memory location and the file F consists of 8 records with the following address:

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Open Addressing Procedure:**

Table T:	_	C	_	A	_	_	_	B	_	_	_
Address:	1	2	3	4	5	6	7	8	9	10	11

Collision Resolution
(Open Addressing: Linear Probing and Modifications)

- Suppose the table T has 11 memory location and the file F consists of 8 records with the following address:

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Open Addressing Procedure:**

Table T:	_	C	_	A	_	_	_	B	_	_	D
Address:	1	2	3	4	5	6	7	8	9	10	11

Collision Resolution
(Open Addressing: Linear Probing and Modifications)

- Suppose the table T has 11 memory location and the file F consists of 8 records with the following address:

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Open Addressing Procedure:**

Table T:	_	C	_	A	E	_	_	B	_	_	D
Address:	1	2	3	4	5	6	7	8	9	10	11

- **Collision Occurred , Store next address**

Collision Resolution
(Open Addressing: Linear Probing and Modifications)

- Suppose the table T has 11 memory location and the file F consists of 8 records with the following address:

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Open Addressing Procedure:**

Table T:	X	C	_	A	E	_	_	B	_	_	D
Address:	1	2	3	4	5	6	7	8	9	10	11

- **Collision Occurred , Store next address**

Collision Resolution
(Open Addressing: Linear Probing and Modifications)

- Suppose the table T has 11 memory location and the file F consists of 8 records with the following address:

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Open Addressing Procedure:**

Table T:	X	C	_	A	E	Y	_	B	_	_	D
Address:	1	2	3	4	5	6	7	8	9	10	11

- **Collision Occurred , Store next address**

Collision Resolution
(Open Addressing: Linear Probing and Modifications)

- Suppose the table T has 11 memory location and the file F consists of 8 records with the following address:

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Open Addressing Procedure:**

Table T:	X	C	Z	A	E	Y	_	B	_	_	D
Address:	1	2	3	4	5	6	7	8	9	10	11

- **Collision Occurred , Store next address**

Collision Resolution
(Open Addressing: Linear Probing and Modifications)

- **Open Addressing Procedure:**

Table T:	X	C	Z	A	E	Y	_	B	_	_	D
Address:	1	2	3	4	5	6	7	8	9	10	11

- The average number S of probes for a successful search follows:

$$S = (1+1+1+1+2+2+2+3)/8 = 13/8 \approx 1.6$$

- The average number of U of probes for an unsuccessful search follows:

$$U = (7+6+5+4+3+2+1+2+1+1+8)/11 = 40/11 \approx 3.6$$

Collision Resolution
(Open Addressing: Quadratic Probing)

- Disadvantage of linear probing: tend to cluster, that is, appear next to one another, the load factor is greater than 50%

- Quadratic Probing:
  - Instead of searching the location with addresses h, h+1, h+2,… , we linearly search the location with addresses h, h+1, h+4, h+9, h+16,…, $h+i^2$, …

Collision Resolution
(Quadratic Probing)

- Suppose the table T has 11 memory location and the file F consists of 8 records with the following address:

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Open Addressing Procedure (Quadratic Probing):**

Table T:	_	_	_	_	_	_	_	_	_	_	_
Address:	1	2	3	4	5	6	7	8	9	10	11

Collision Resolution
(Quadratic Probing)

- Suppose the table T has 11 memory location and the file F consists of 8 records with the following address:

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Open Addressing Procedure (Quadratic Probing):**

Table T:	_	_	_	A	_	_	_	_	_	_	_
Address:	1	2	3	4	5	6	7	8	9	10	11

Collision Resolution
(Quadratic Probing)

- Suppose the table T has 11 memory location and the file F consists of 8 records with the following address:

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Open Addressing Procedure (Quadratic Probing):**

Table T:	_	C	_	A	_	_	_	B	_	_	D
Address:	1	2	3	4	5	6	7	8	9	10	11

Collision Resolution
(Quadratic Probing)

- Suppose the table T has 11 memory location and the file F consists of 8 records with the following address:

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Open Addressing Procedure (Quadratic Probing):**

Table T:	_	C	_	A	E	_	_	B	_	_	D
Address:	1	2	3	4	5	6	7	8	9	10	11

- **Collision Occurred : h+1**

Collision Resolution
(Quadratic Probing)

- Suppose the table T has 11 memory location and the file F consists of 8 records with the following address:

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Open Addressing Procedure (Quadratic Probing):**

Table T:	X	C	_	A	E	_	_	B	_	_	D
Address:	1	2	3	4	5	6	7	8	9	10	11

- **Collision Occurred : h+1**

Collision Resolution
(Quadratic Probing)

- Suppose the table T has 11 memory location and the file F consists of 8 records with the following address:

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Open Addressing Procedure (Quadratic Probing):**

Table T:	X	C	_	A	E	Y	_	B	_	_	D
Address:	1	2	3	4	5	6	7	8	9	10	11

- **Collision Occurred : h+1**

Collision Resolution
(Quadratic Probing)

- Suppose the table T has 11 memory location and the file F consists of 8 records with the following address:

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Open Addressing Procedure (Quadratic Probing):**

Table T:	X	C	_	A	E	Y	_	B	Z	_	D
Address:	1	2	3	4	5	6	7	8	9	10	11

- **Collision Occurred : h+8**

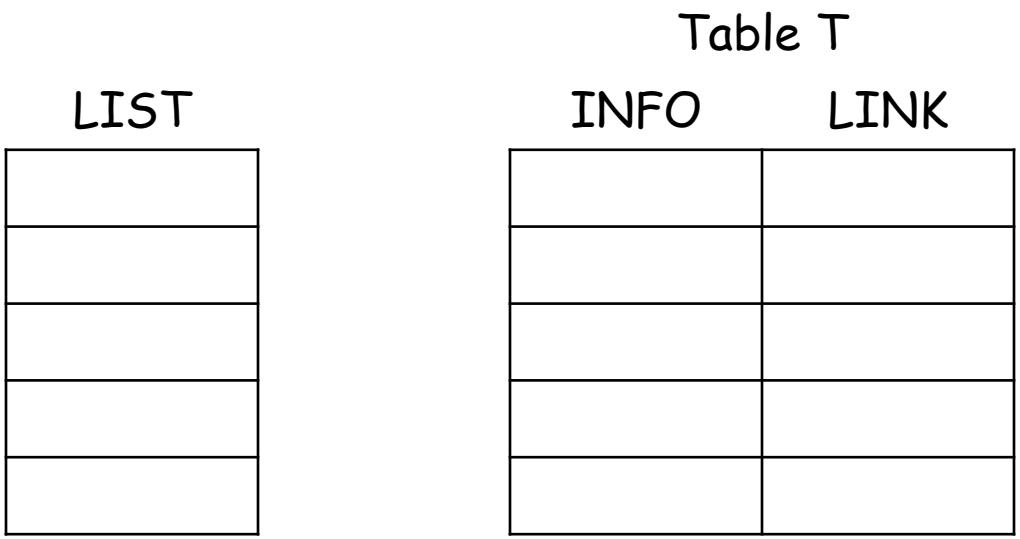
Collision Resolution
(Open Addressing: Double Hashing)

- Here a second has function H' is used for resolving a collision, as follows.
- Suppose a record R with key k has the hash address H(k) = h and H'(k) = h' ≠ m.
- Then we linearly search the location with addresses

  h, h+h', h+2h', h+3h', … …

Collision Resolution
(Chaining)

- **Maintain two table:**
  - Table T as before with additional field LINK which is used so that all records in T with the same hash address h may be liked together to form a linked list.
  - Hash address table LIST which contains pointer to the linked list in T.



Collision Resolution
(Chaining)

- Suppose a new record R with key k is added to the file F.
- Place R in the first available location in the table T.
- Add R to the linked list with pointer LIST[H(k)].
- If the linked list of records are not sorted, then R is simply inserted at the beginning of its linked list.

Collision Resolution
(Chaining)

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Chaining:**

	LIST
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

	TABLE T	
	INFO	LINK
1		2
2		3
3		4
4		5
5		6
6		7
7		8
8		9
9		10
10		11

AVAIL 1

NEW AVAIL
AVAIL LINK[AVAIL]
INFO[NEW] ITEM

LINK[NEW] LIST[H(k)]
LIST[H(k)] NEW

Collision Resolution
(Chaining)

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Chaining:**

	LIST
1	
2	
3	
4	1
5	
6	
7	
8	
9	
10	

	TABLE T	
	INFO	LINK
1	A	0
2		3
3		4
4		5
5		6
6		7
7		8
8		9
9		10
10		11

AVAIL□2

NEW□ AVAIL
AVAIL□ LINK[AVAIL]
INFO[NEW]□ITEM

LINK[NEW]□ LIST[H(k)]
LIST[H(k)]□ NEW

Collision Resolution
(Chaining)

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Chaining:**

	LIST
1	
2	
3	
4	1
5	
6	
7	
8	2
9	
10	

	TABLE T	
	INFO	LINK
1	A	0
2	B	0
3		4
4		5
5		6
6		7
7		8
8		9
9		10
10		11

AVAIL□3

NEW□ AVAIL
AVAIL□ LINK[AVAIL]
INFO[NEW]□ITEM

LINK[NEW]□ LIST[H(k)]
LIST[H(k)]□ NEW

Collision Resolution
(Chaining)

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Chaining:**

	LIST
1	
2	3
3	
4	1
5	
6	
7	
8	2
9	
10	

	TABLE T	
	INFO	LINK
1	A	0
2	B	0
3	C	0
4		5
5		6
6		7
7		8
8		9
9		10
10		11

AVAIL□4

NEW□ AVAIL
AVAIL□ LINK[AVAIL]
INFO[NEW]□ITEM

LINK[NEW]□ LIST[H(k)]
LIST[H(k)]□ NEW

Collision Resolution
(Chaining)

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Chaining:**

	LIST
1	
2	3
3	
4	1
5	
6	
7	
8	2
9	
10	
11	4

	TABLE T	
	INFO	LINK
1	A	0
2	B	0
3	C	0
4	D	0
5		6
6		7
7		8
8		9
9		10
10		11
11		0

AVAIL□5

NEW□ AVAIL
AVAIL□ LINK[AVAIL]
INFO[NEW]□ITEM

LINK[NEW]□ LIST[H(k)]
LIST[H(k)]□ NEW

Collision Resolution
(Chaining)

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Chaining:**

	LIST
1	
2	3
3	
4	5
5	
6	
7	
8	2
9	
10	
11	4

	TABLE T	
	INFO	LINK
1	A	0
2	B	0
3	C	0
4	D	0
5	E	1
6		7
7		8
8		9
9		10
10		11
11		0

AVAIL□6

NEW□ AVAIL
AVAIL□ LINK[AVAIL]
INFO[NEW]□ITEM

LINK[NEW]□ LIST[H(k)]
LIST[H(k)]□ NEW

Collision Resolution
(Chaining)

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Chaining:**

	LIST
1	
2	3
3	
4	5
5	
6	
7	
8	2
9	
10	
11	6

	TABLE T	
	INFO	LINK
1	A	0
2	B	0
3	C	0
4	D	0
5	E	1
6	X	4
7		8
8		9
9		10
10		11
11		0

AVAIL□7

NEW□ AVAIL
AVAIL□ LINK[AVAIL]
INFO[NEW]□ITEM

LINK[NEW]□ LIST[H(k)]
LIST[H(k)]□ NEW

Collision Resolution
(Chaining)

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Chaining:**

	LIST
1	
2	3
3	
4	5
5	7
6	
7	
8	2
9	
10	
11	6

	TABLE T	
	INFO	LINK
1	A	0
2	B	0
3	C	0
4	D	0
5	E	1
6	X	4
7	Y	0
8		9
9		10
10		11
11		0

AVAIL□8

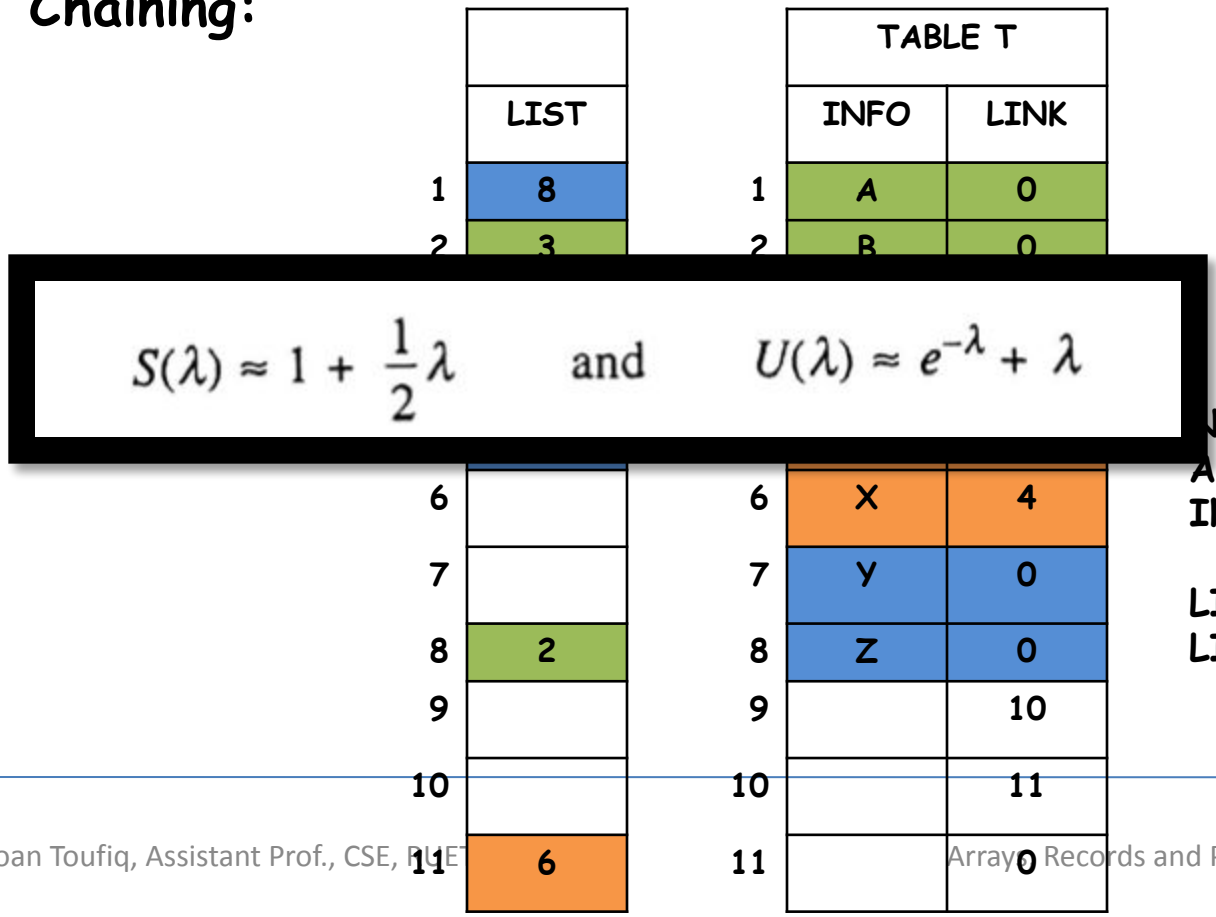
NEW□ AVAIL
AVAIL□ LINK[AVAIL]
INFO[NEW]□ITEM

LINK[NEW]□ LIST[H(k)]
LIST[H(k)]□ NEW

Collision Resolution
(Chaining)

Record:	A	B	C	D	E	X	Y	Z
H(k):	4	8	2	11	4	11	5	1

- **Chaining:**



$$S(\lambda) \approx 1 + \frac{1}{2}\lambda \quad \text{and} \quad U(\lambda) \approx e^{-\lambda} + \lambda$$

AVAIL 9

NEW AVAIL
AVAIL LINK[AVAIL]
INFO[NEW] ITEM

LINK[NEW] LIST[H(k)]
LIST[H(k)] NEW

Collision Resolution
(Chaining)

- **Disadvantage**:
  - Need 3m memory cell (Table- INFO, LINK, LIST).
  - It may be more useful to use open addressing with a table with 3m locations, which has the load factor $\lambda \leq 1/3$, than to use chaining to resolve collition.

Any Query?