Data Structures and Algorithms

Trong-Hop Do

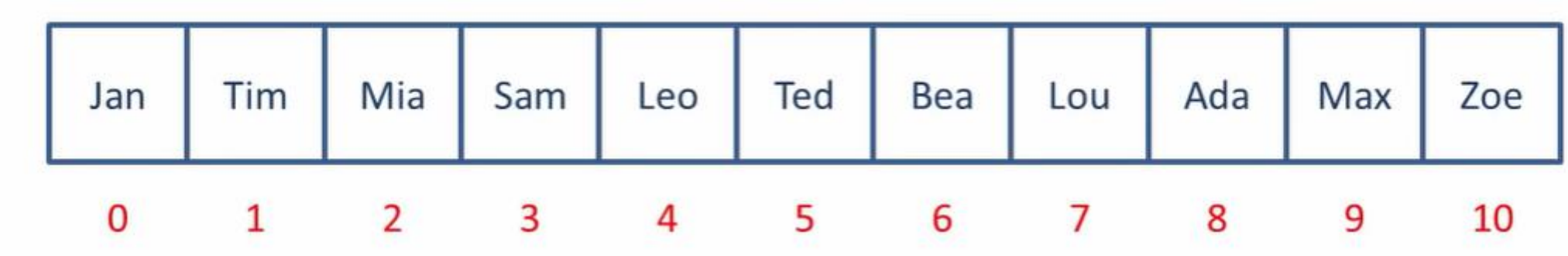
University of Information Technology, HCM city

Hash table

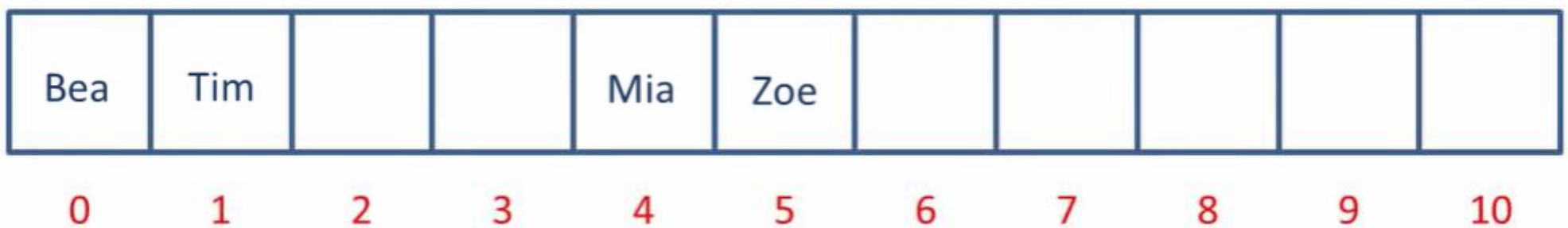
Problem

Find Ada

Ada = 8



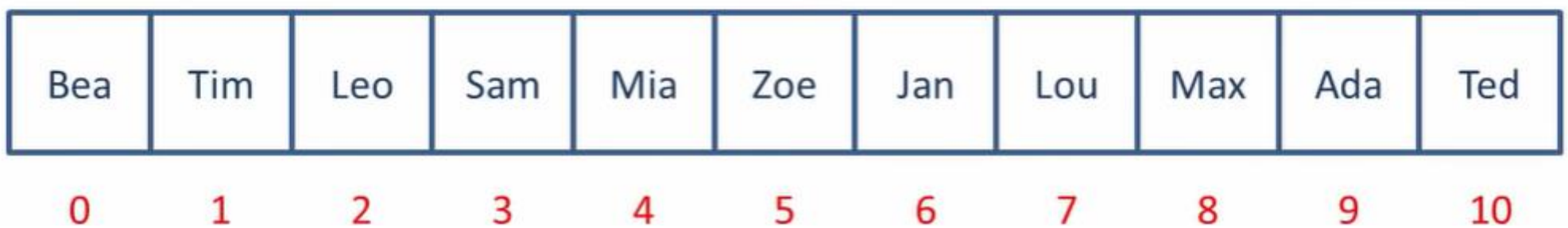
Mia	M	77	i	105	a	97	279	4
Tim	T	84	i	105	m	109	298	1
Bea	B	66	e	101	a	97	264	0
Zoe	Z	90	o	111	e	101	302	5



Mia	M	77	i	105	a	97	279	4
Tim	T	84	i	105	m	109	298	1
Bea	B	66	e	101	a	97	264	0
Zoe	Z	90	o	111	e	101	302	5
Jan	J	74	a	97	n	110	281	6
Ada	A	65	d	100	a	97	262	9
Leo	L	76	e	101	o	111	288	2
Sam	S	83	a	97	m	109	289	3
Lou	L	76	o	111	u	117	304	7
Max	M	77	a	97	x	120	294	8
Ted	T	84	e	101	d	100	285	10

Bea	Tim	Leo	Sam	Mia	Zoe	Jan	Lou	Max	Ada	Ted
0	1	2	3	4	5	6	7	8	9	10

Index number = *sum ASCII codes* Mod *size of array*

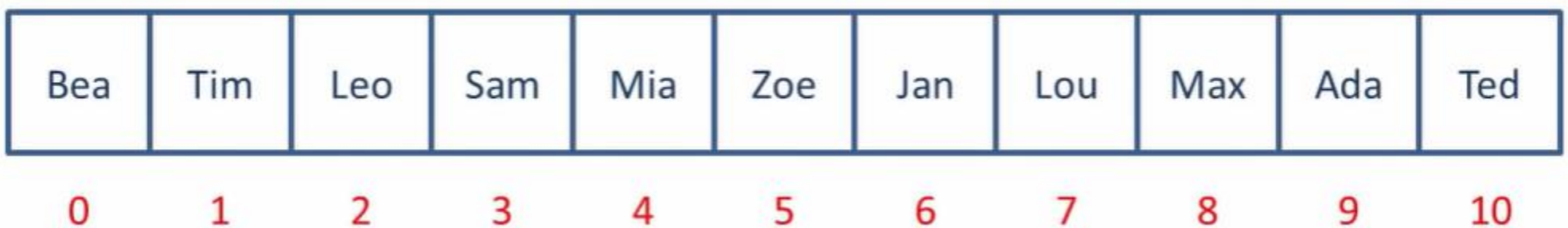


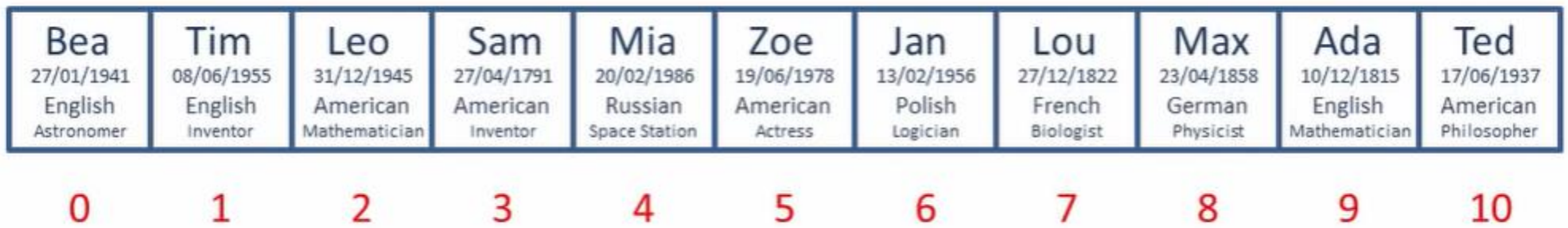
Find Ada

Ada = (65 + 100 + 97) = 262

262 Mod 11 = 9

myData = Array(9)




Bea
27/01/1941
English
Astronomer
Tim
08/06/1955
English
Inventor
Leo
31/12/1945
American
Mathematician
Sam
27/04/1791
American
Inventor
Mia
20/02/1986
Russian
Space Station
Zoe
19/06/1978
American
Actress
Jan
13/02/1956
Polish
Logician
Lou
27/12/1822
French
Biologist
Max
23/04/1858
German
Physicist
Ada
10/12/1815
English
Mathematician
Ted
17/06/1937
American
Philosopher
0
1
2
3
4
5
6
7
8
9
10

Hashing algorithm

- Calculation applied to a key to transform it into an address
- For numeric keys, divide the key by the number of available addresses, *n*, and take the remainder

address = *key* Mod *n*

- For alphanumeric keys, divide the sum of ASCII codes in a key by the number of available addresses, *n*, and take the remainder
- Folding method divides key into equal parts then adds the parts together
  - The telephone number 01452 8345654, becomes 01 + 45 + 28 + 34 + 56 + 54 = 218
  - Depending on size of table, may then divide by some constant and take remainder

Collision

Mia	M	77	i	105	a	97	279	4
Tim	T	84	i	105	m	109	298	1
Bea	B	66	e	101	a	97	264	0
Zoe	Z	90	o	111	e	101	302	5
Sue	S	83	u	117	e	101	301	4
Len	L	76	e	101	n	110	287	1
Moe	M	77	o	111	e	101	289	3
Lou	L	76	o	111	u	117	304	7
Rae	R	82	a	97	e	101	280	5
Max	M	77	a	97	x	120	294	8
Tod	T	84	o	111	d	100	295	9

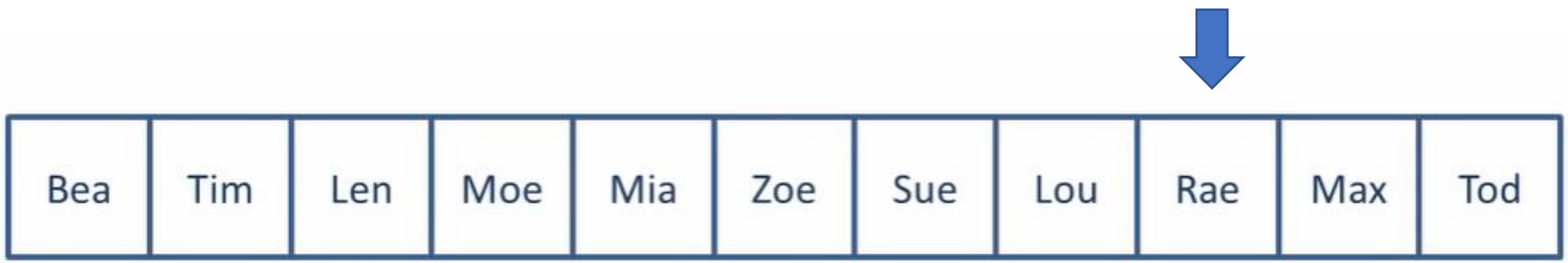
Bea	Tim	Len	Moe	Mia	Zoe	Sue	Lou	Rae	Max	Tod
0	1	2	3	4	5	6	7	8	9	10

Find Rae

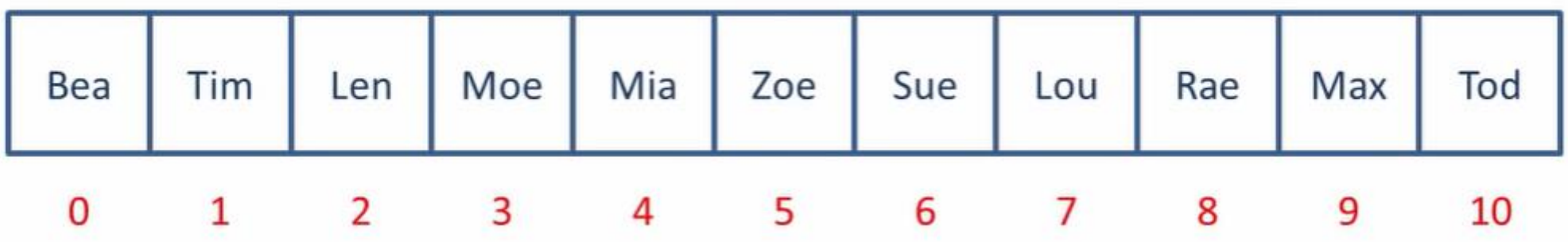
Rae = (82 + 97 + 101) = 280

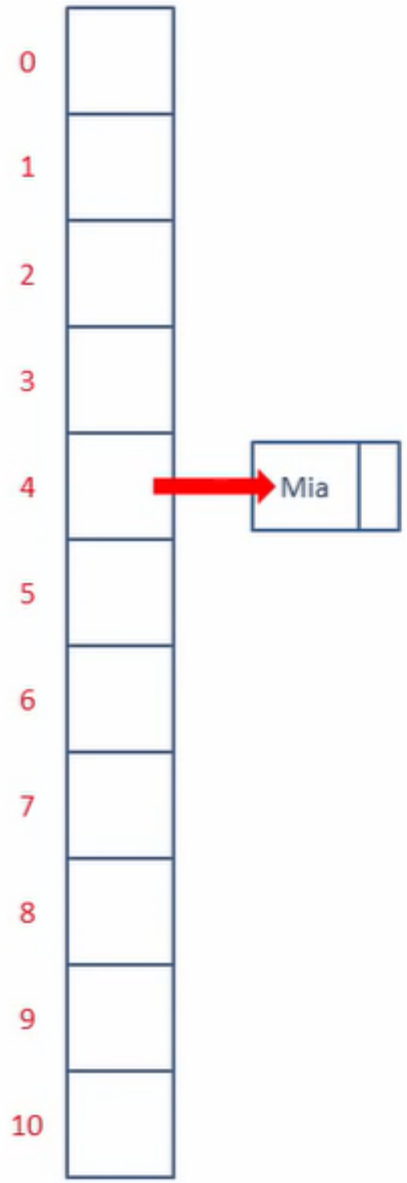
280 Mod 11 = 5

myData = Array(5)

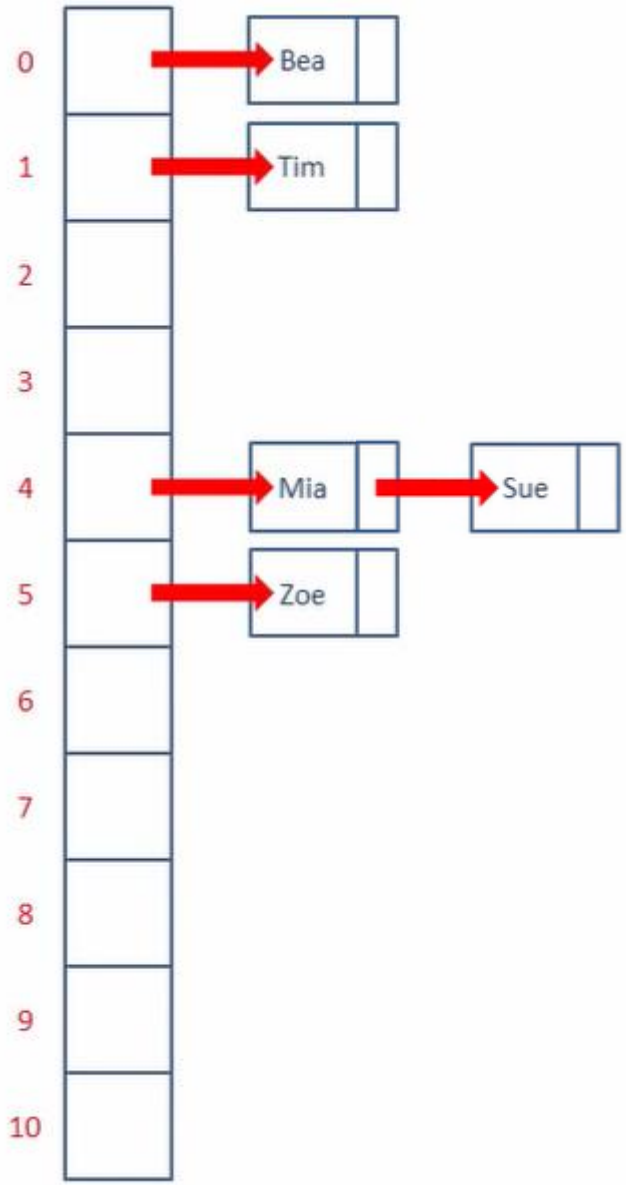


$$\text{Load Factor} = \frac{\text{Total number of items stored}}{\text{Size of the array}}$$

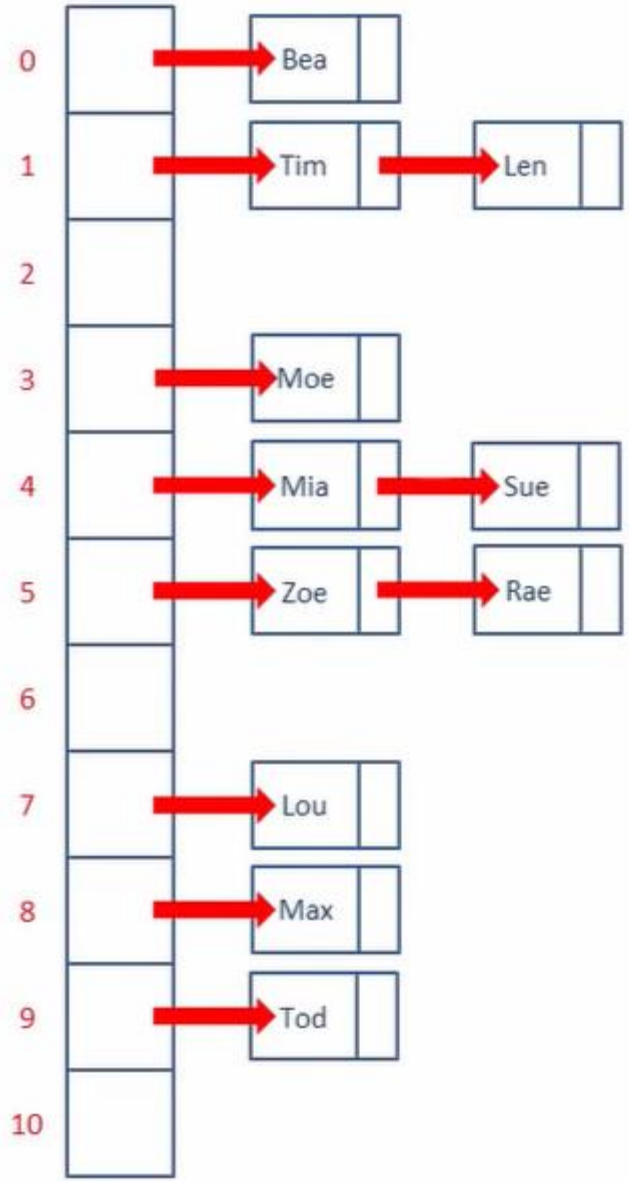




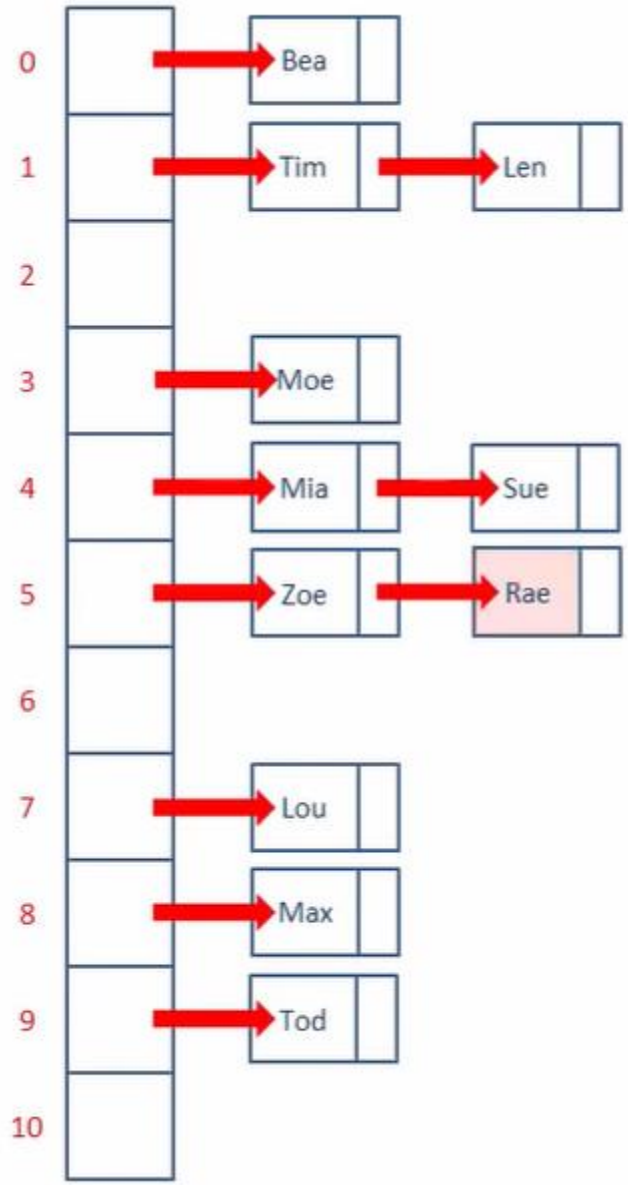
Mia M 77 i 105 a 97 279 4



Mia	M	77	i	105	a	97	279	4
Tim	T	84	i	105	m	109	298	1
Bea	B	66	e	101	a	97	264	0
Zoe	Z	90	o	111	e	101	302	5
Sue	S	83	u	117	e	101	301	4



Mia	M	77	i	105	a	97	279	4
Tim	T	84	i	105	m	109	298	1
Bea	B	66	e	101	a	97	264	0
Zoe	Z	90	o	111	e	101	302	5
Sue	S	83	u	117	e	101	301	4
Len	L	76	e	101	n	110	287	1
Moe	M	77	o	111	e	101	289	3
Lou	L	76	o	111	u	117	304	7
Rae	R	82	a	97	e	101	280	5
Max	M	77	a	97	x	120	294	8
Tod	T	84	o	111	d	100	295	9



Find Rae 280 Mod 11 = 5

myData = Array(5)

Collison Resolution

- Open addressing
  - Linear probing
  - Plus 3 rehash
  - Quadratic probing (*failed attempts*)^2
  - Double hashing
- Closed addressing

Open addressing

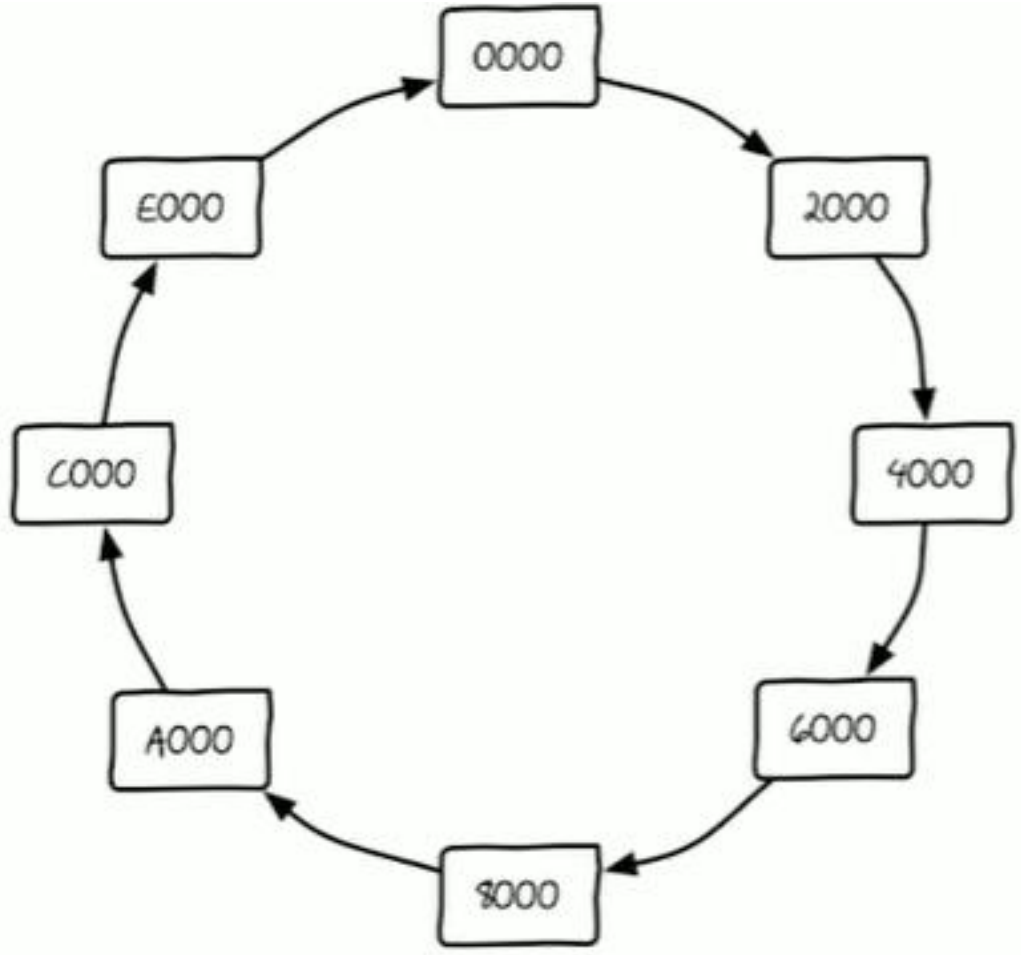
- All elements are stored in the hash table itself. So at any point, the size of the table must be greater than or equal to the total number of keys.
- Operations:
  - Insert(k): Keep probing until an empty slot is found. Once an empty slot is found, insert k.
  - Search(k): Keep probing until slot’s key doesn’t become equal to k or an empty slot is reached.
  - Delete(k): Delete operation is interesting. If we simply delete a key, then the search may fail. So slots of deleted keys are marked specially as “deleted”. The insert can insert an item in a deleted slot, but the search doesn’t stop at a deleted slot.

Objectives of Hash Function

- Minimize collisions
- Uniform distribution of hash values
- Easy to calculate
- Resolve any collisions

Applications: consistent hasing Cassandra

Partition key	Murmur3 hash value
jim	-2245462676723223822
carol	7723358927203680754
johnny	-6723372854036780875
suzy	1168604627387940318



Applications: topic partitioning Kafka