Hashing*

Arnamoy Bhattacharyya

*Some slides are taken from Danny Heap and Bogdan Simion, also youtube channel Kevin Drumm

Agenda

- Talk about hashing
- Course Evaluation

Lists and linear search

Data Structure	search()	insert()	delete()
List	O(n)	O(n)	O(n)

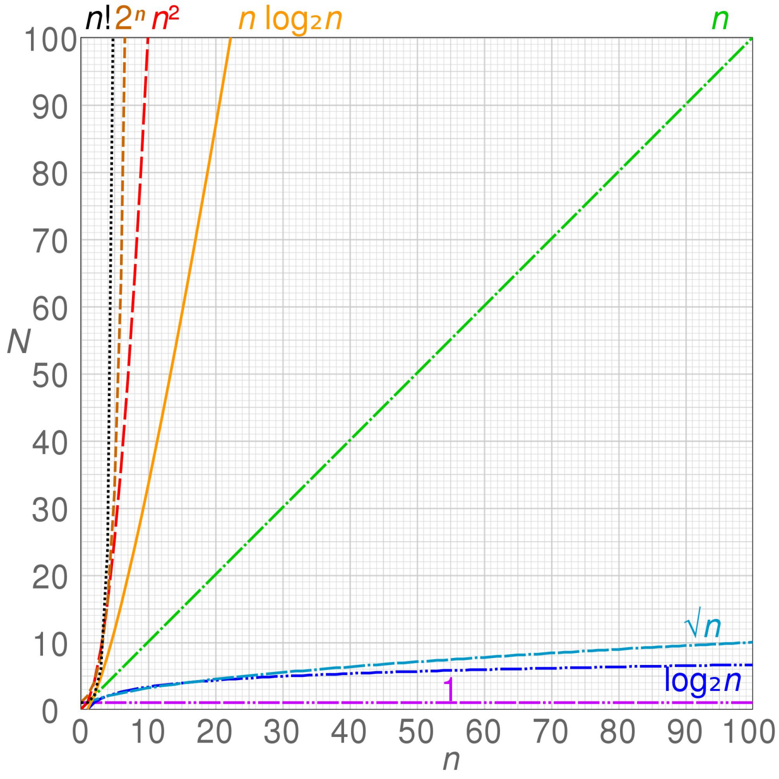
Ordered Lists and Binary search

Data Structure	search()	insert()	delete()
List	O(n)	O(n)	O(n)
Ordered List	O(lg n)	O(lg n)	O(lg n)

Demo

Onto Pycharm

Complexity Comparisons



How dictionary offers Constant time search?

By a technique called hashing

Example: hashing

What's the time complexity of:

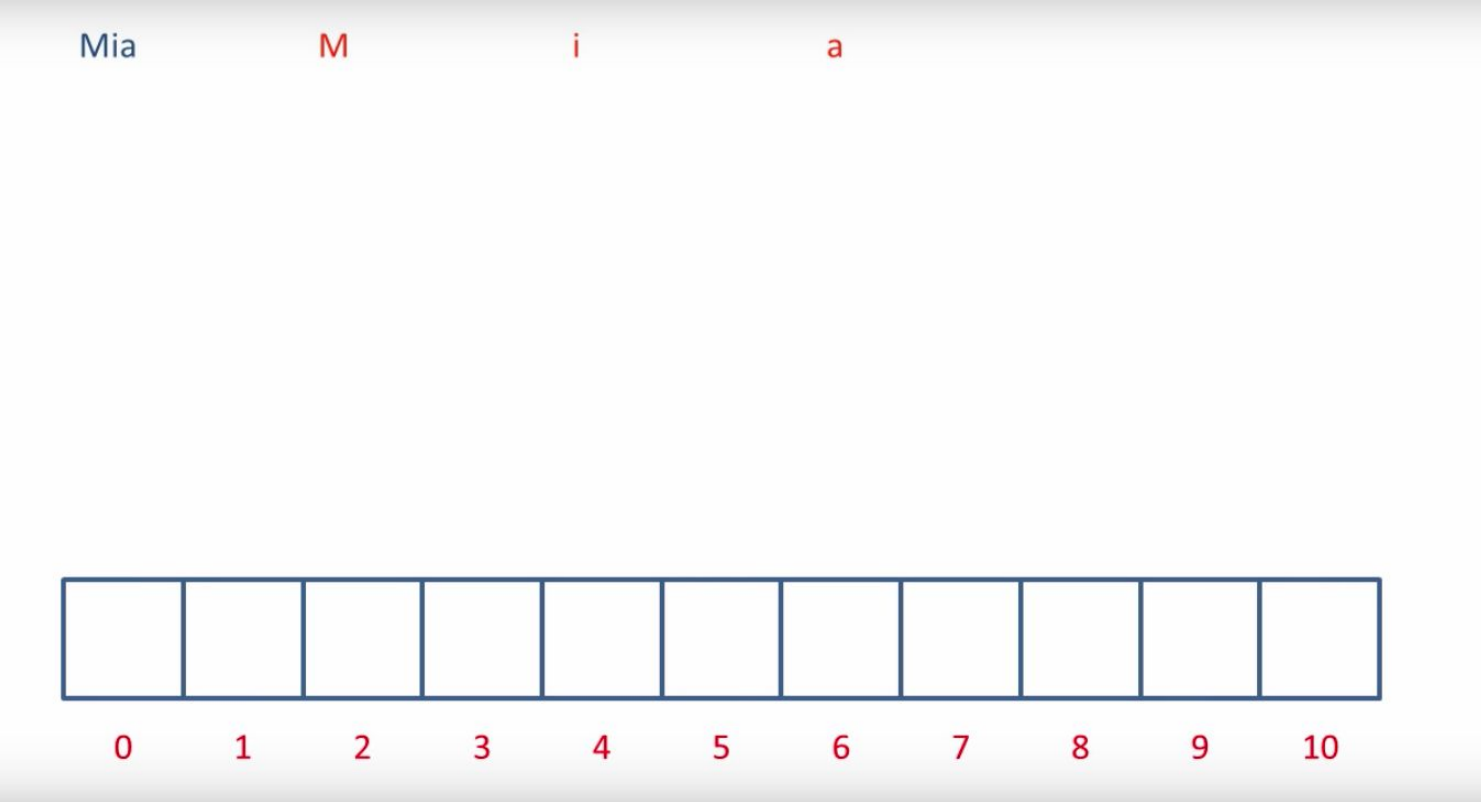
1) Finding the string 'Ada' in this list?
2) What if you knew the index of 'Ada'?

Example: hashing

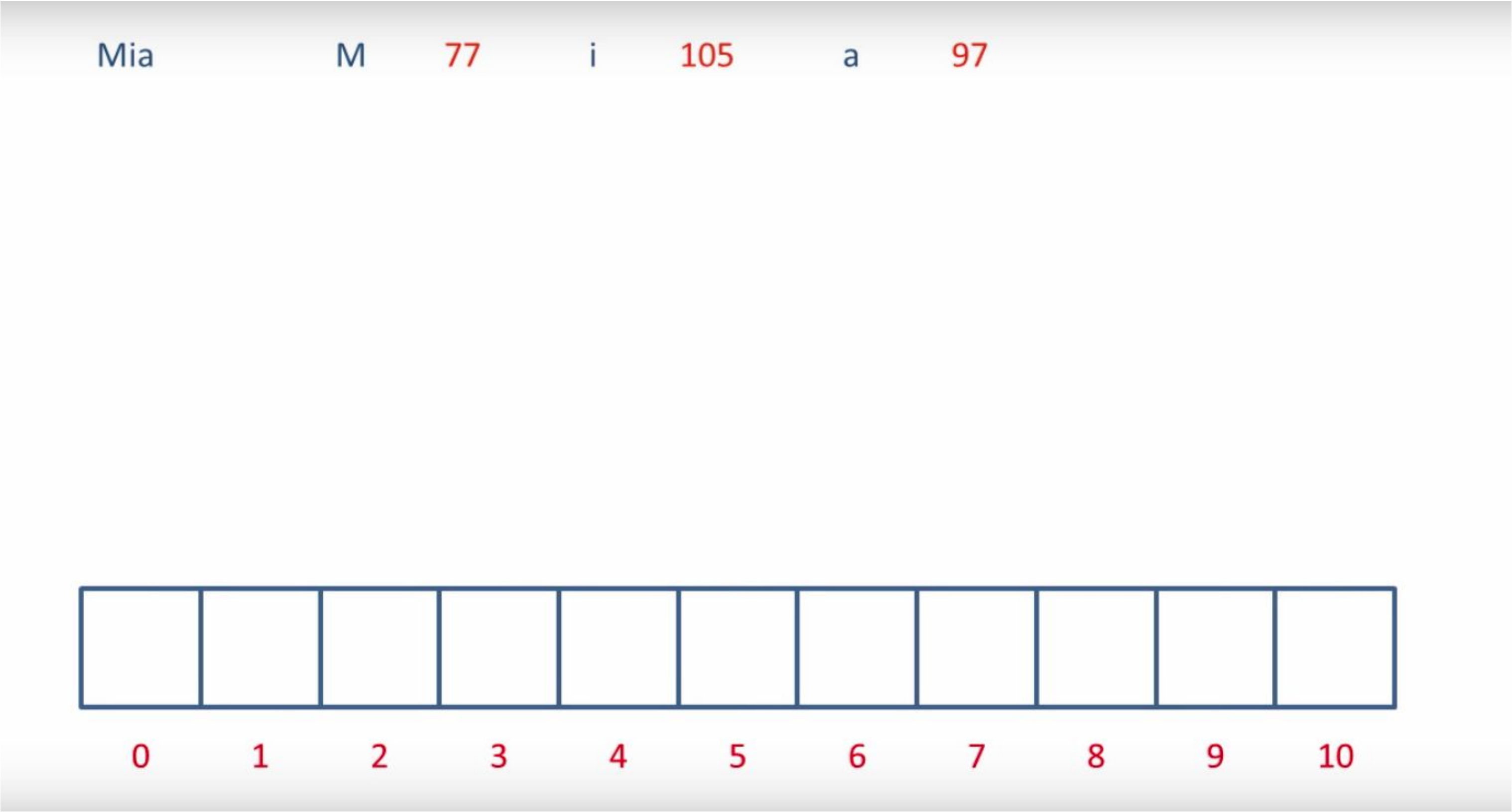
What's the time complexity of:

1) Finding the string 'Ada' in this list?
2) What if you knew the index of 'Ada'?
    a) But how do you know the index of 'Ada' in the list?

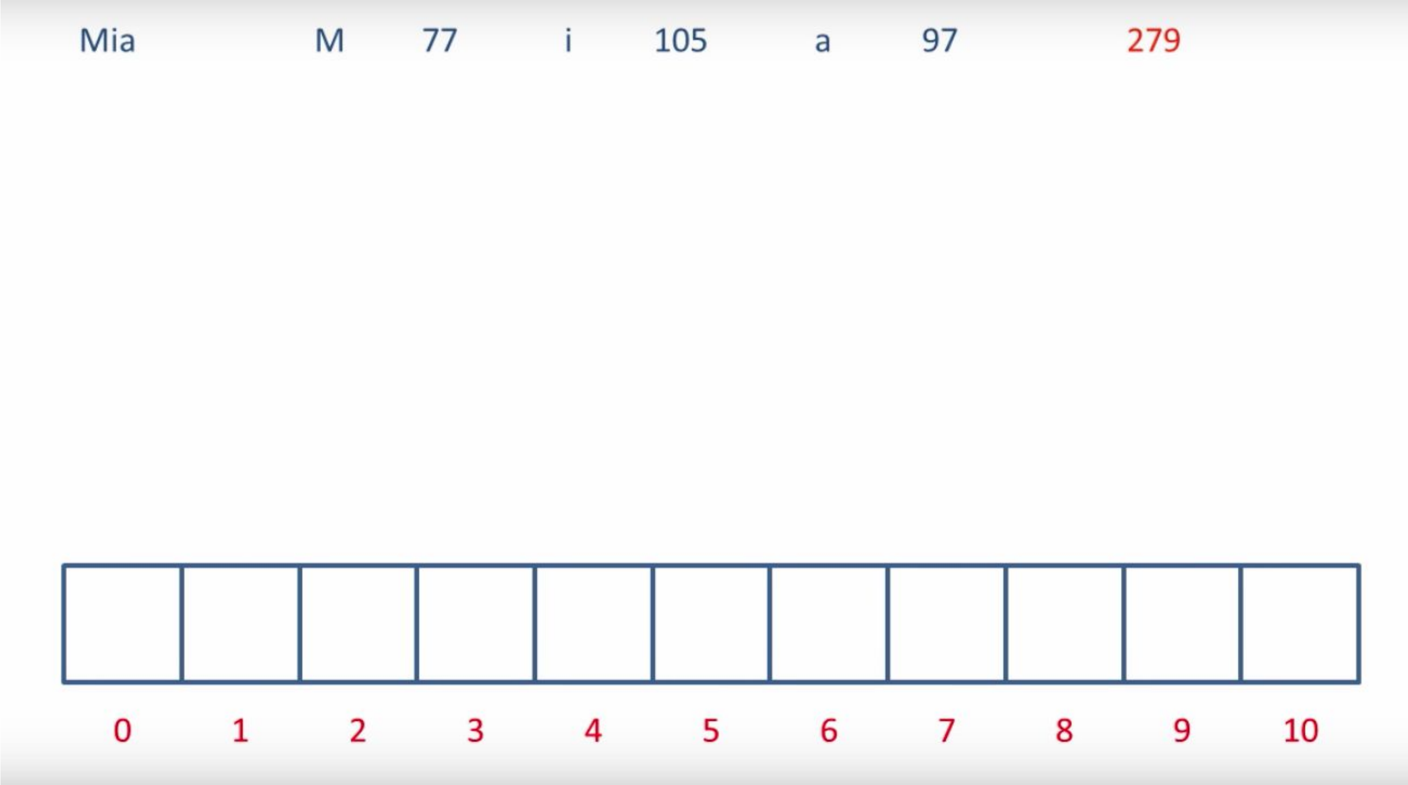
Populating the List



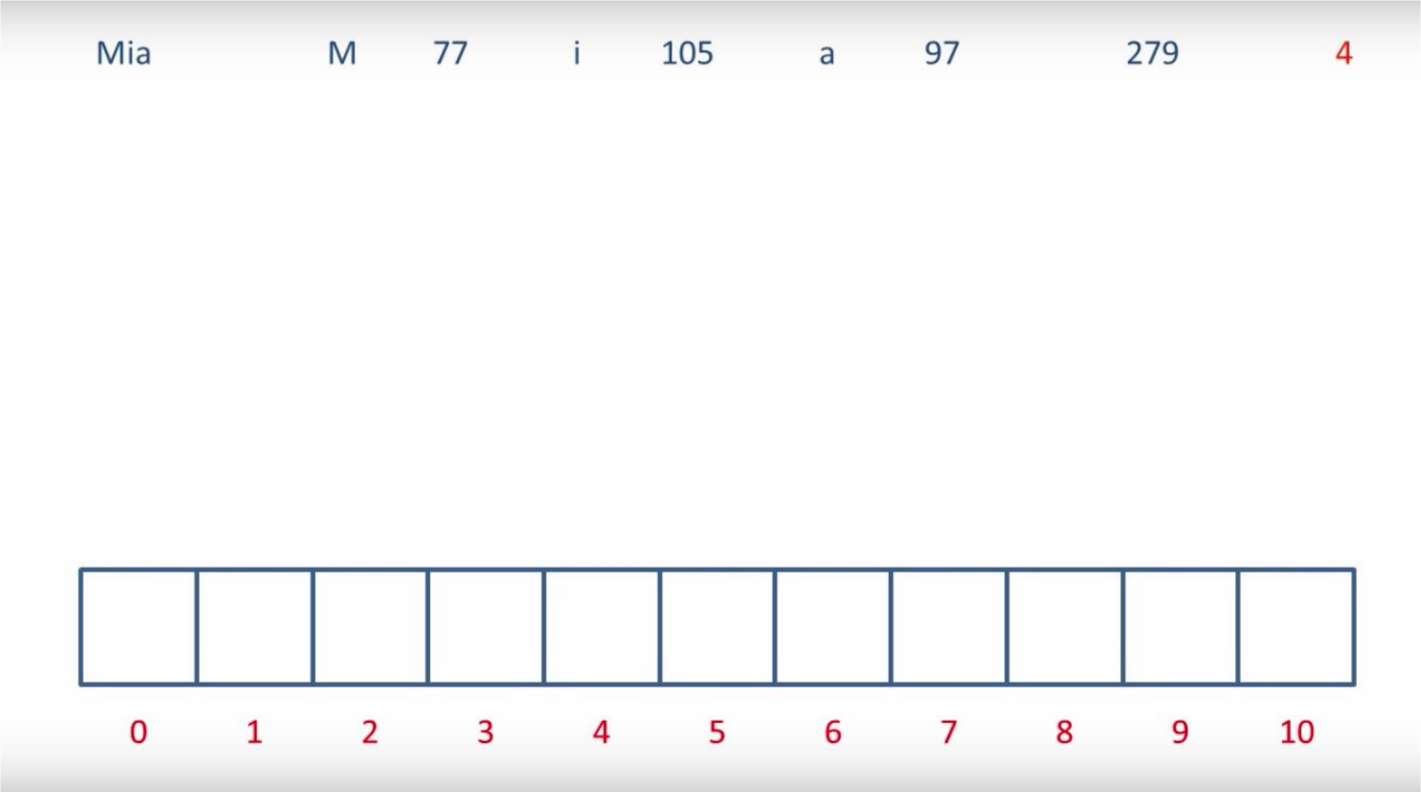
Populating the List



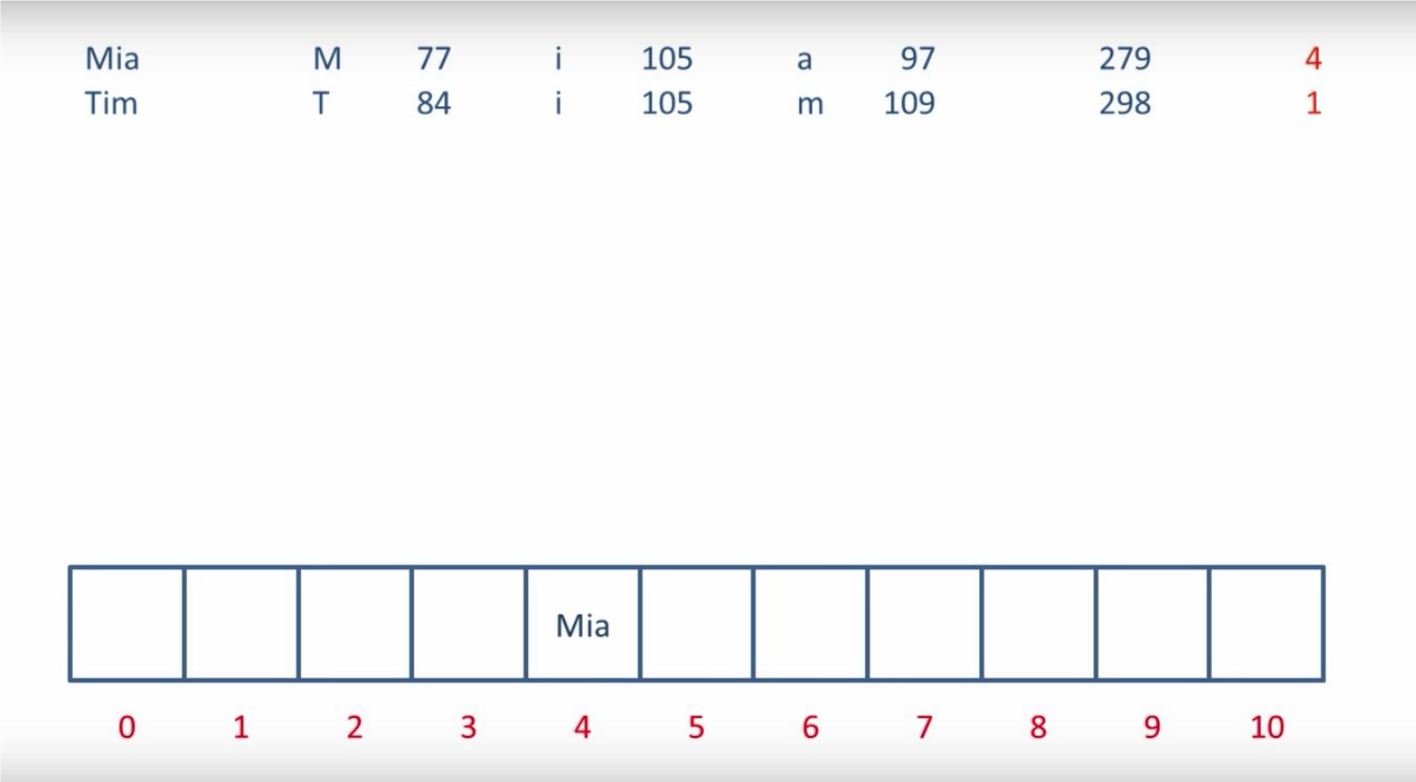
Populating the List



Populating the List

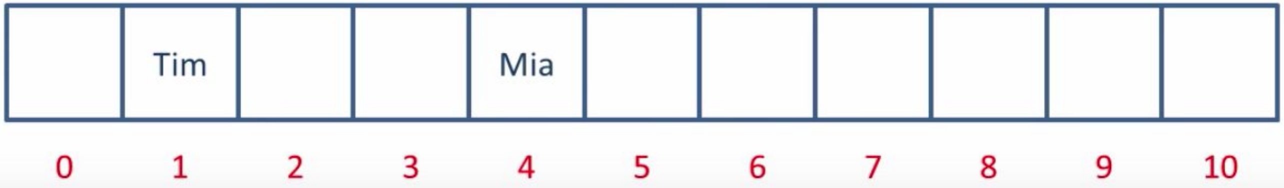


Populating the List



Populating the List

Mia	M	77	i	105	a	97	279	4
Tim	T	84	i	105	m	109	298	1



Populating the List

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

Populating the List

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

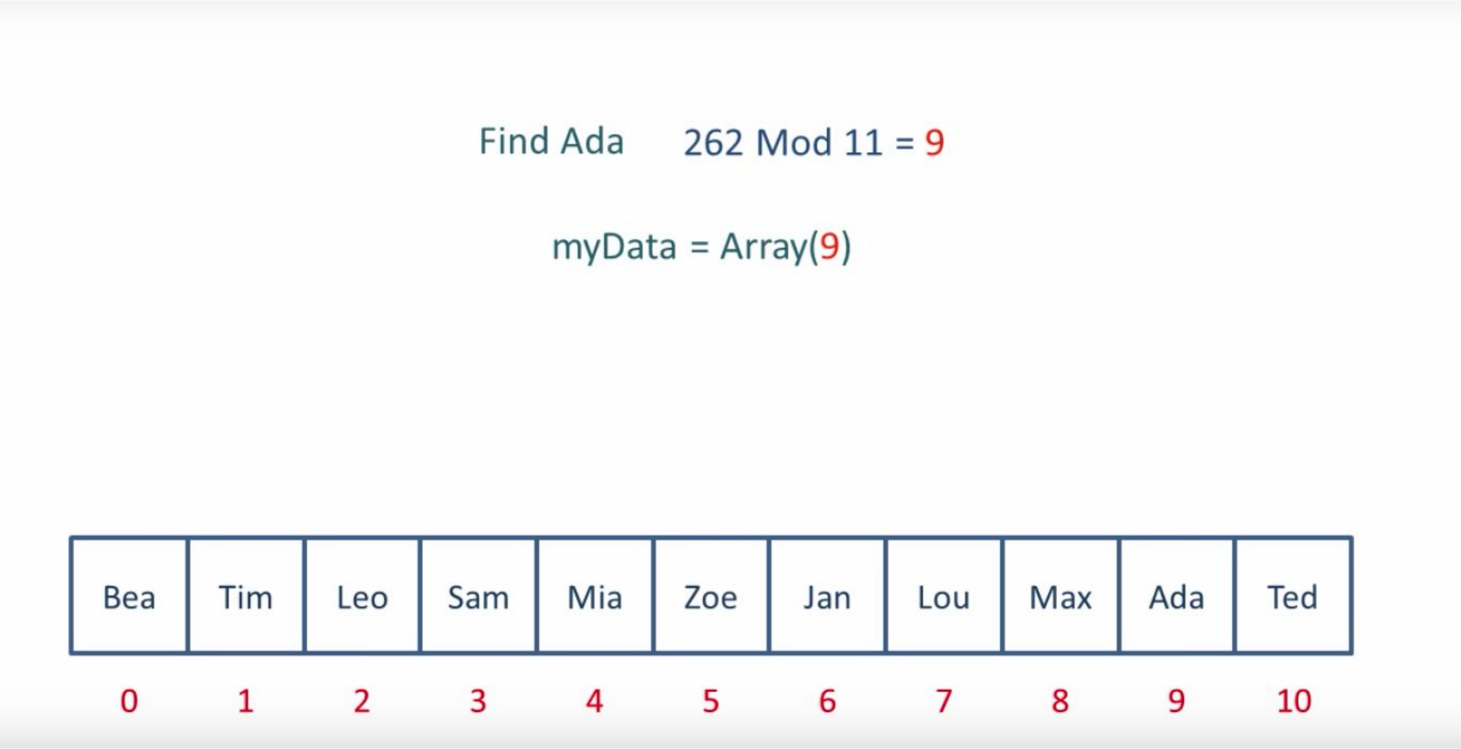
Bea	Tim	Leo	Sam	Mia	Zoe	Jan	Lou	Max	Ada	Ted
0	1	2	3	4	5	6	7	8	9	10

Populating the List

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

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

Search

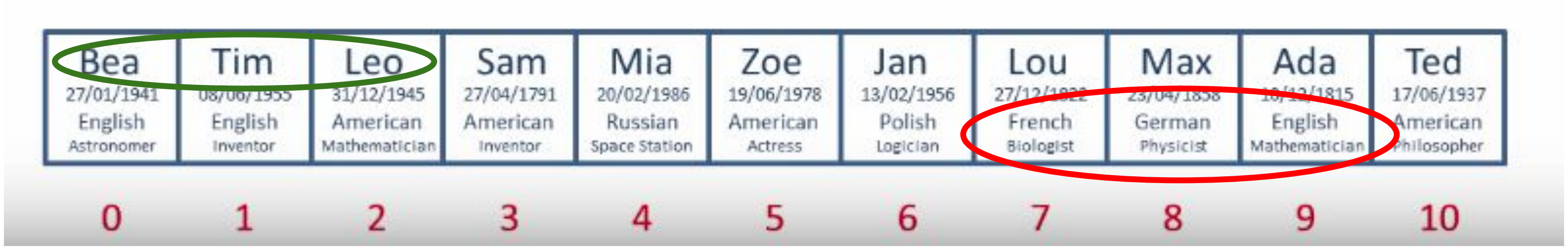

Find Ada
262 Mod 11 = 9
myData = Array(9)
Bea
Tim
Leo
Sam
Mia
Zoe
Jan
Lou
Max
Ada
Ted
0
1
2
3
4
5
6
7
8
9
10

Key, value pairs

Key Value pairs

Key:Value Pairs

Keys



Values

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

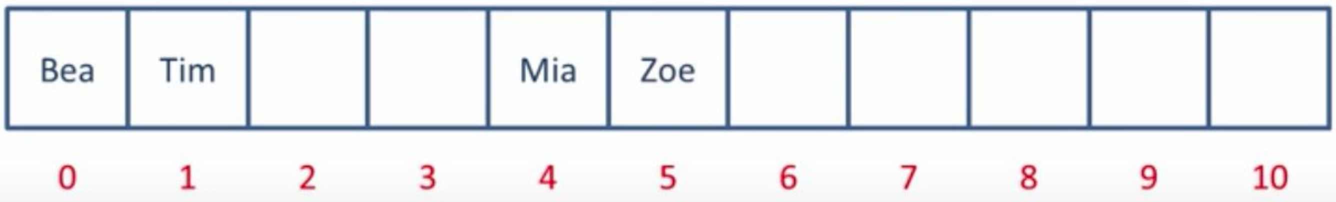
$$\text{address} = key \text{ Mod } n$$

- 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

Collisions

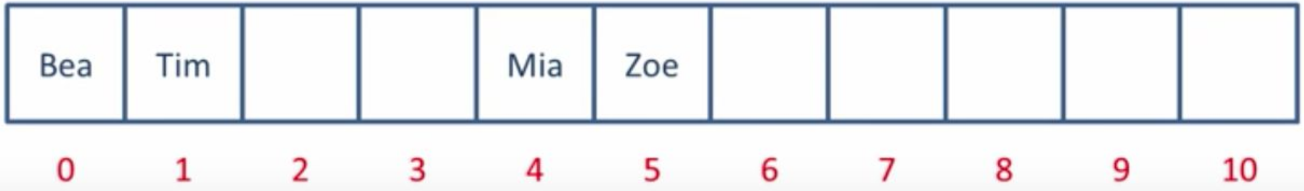
Collisions

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



Collisions

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



Collisions: Linear Probing

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

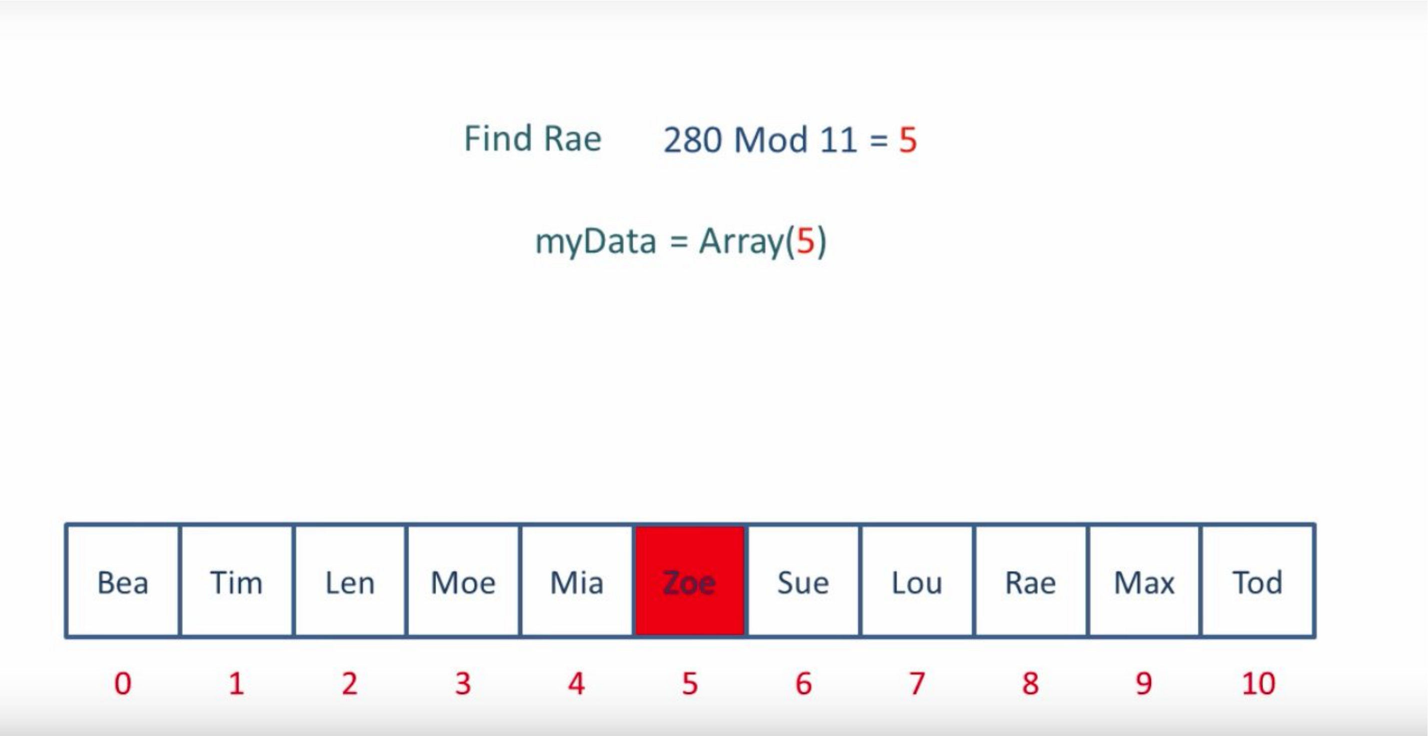
Bea	Tim			Mia	Zoe	Sue				
0	1	2	3	4	5	6	7	8	9	10

Collisions: Linear Probing

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

Bea	Tim	Len		Mia	Zoe	Sue				
0	1	2	3	4	5	6	7	8	9	10

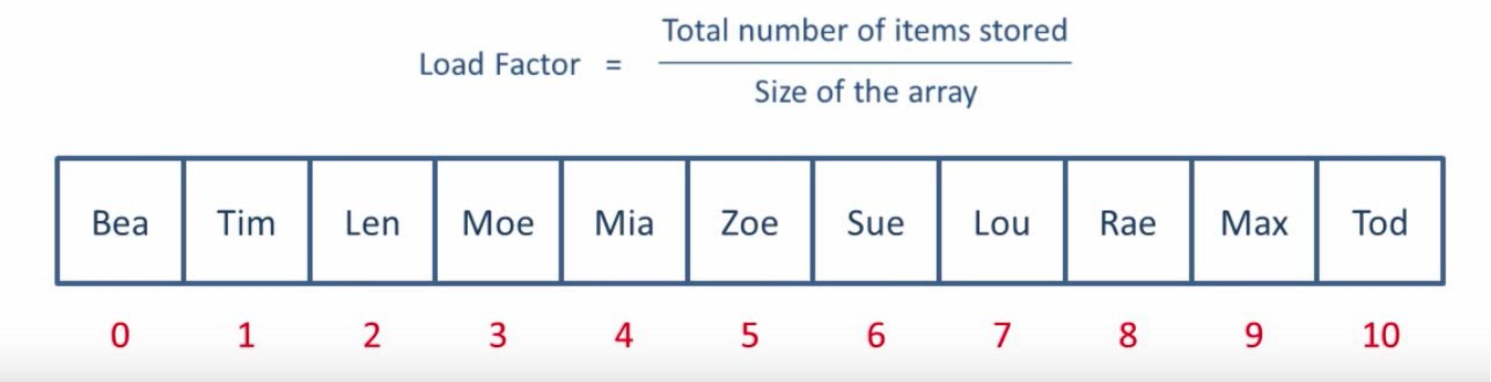
Collisions: Linear Probing



Collisions

More data → more probability for collision

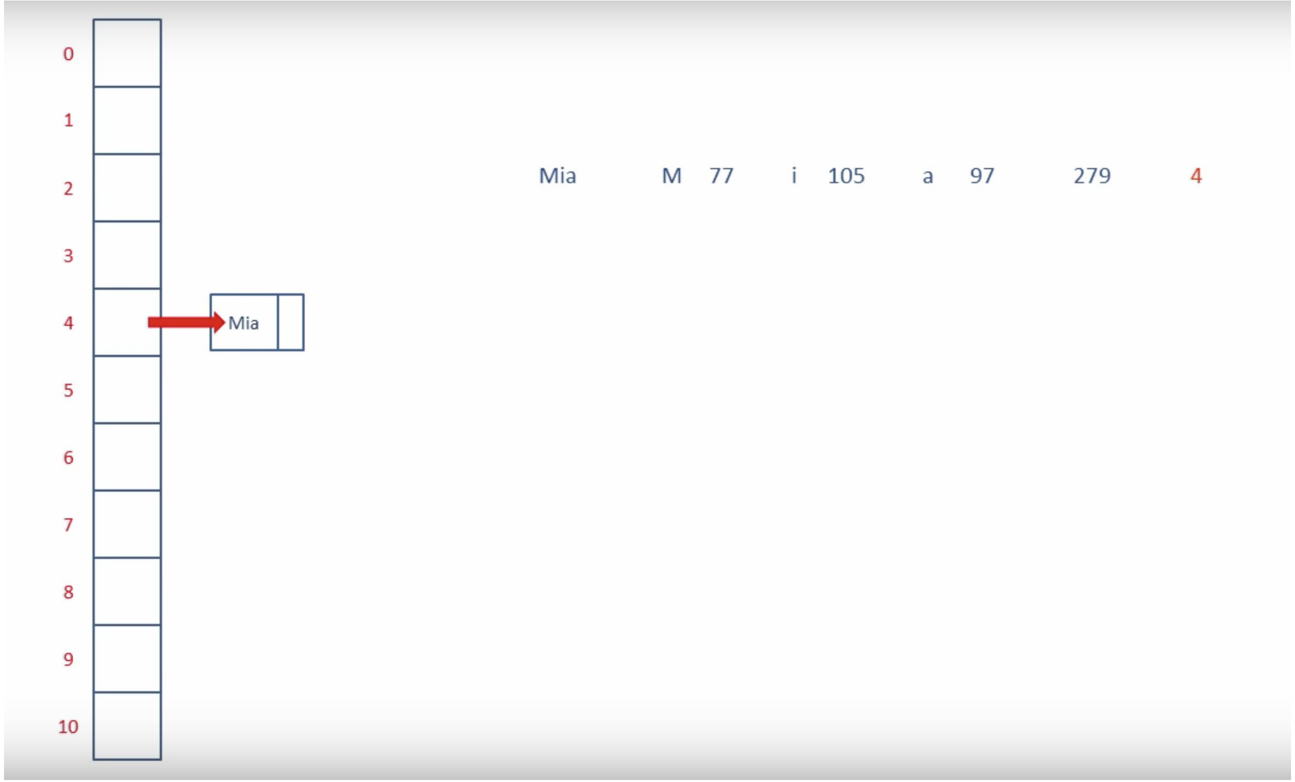
Make the hash table big enough as compared to the number of keys



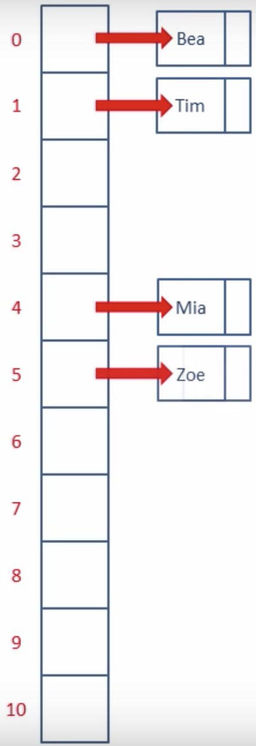
Resize the array based on certain threshold of *Load Factor*

Collisions: Chaining

Collisions: Chaining

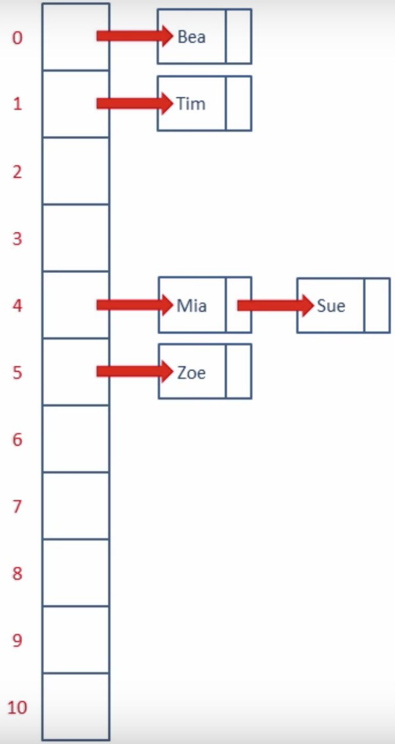


Collisions: Chaining



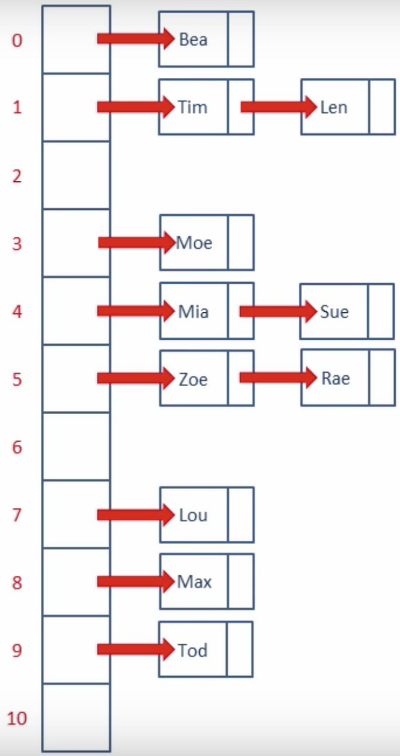
Mia	M	77	i	105	a	97	279	4
Tim	T	84	i	105	m	109	298	1
Bea	B	66	e	101	a	97	264	0
Zoe	Z	90	o	111	e	101	302	5

Collisions: Chaining



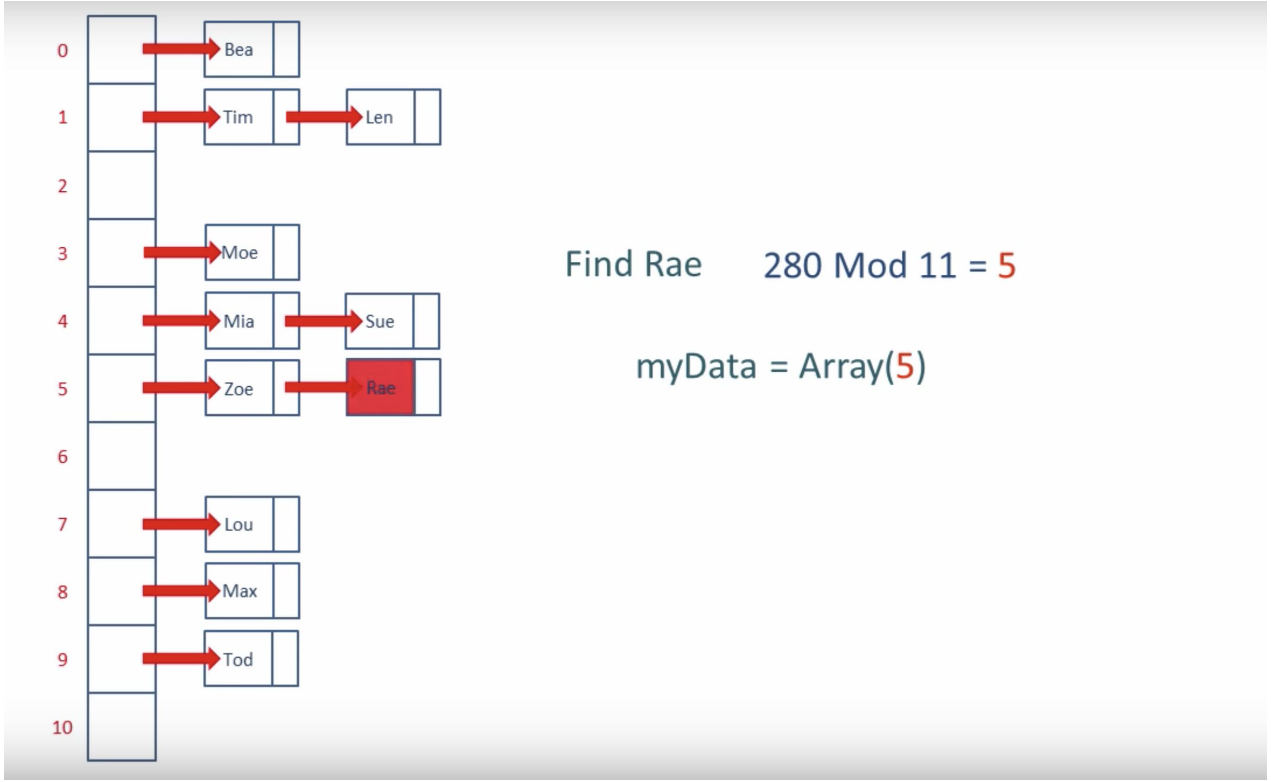
Mia	M	77	i	105	a	97	279	4
Tim	T	84	i	105	m	109	298	1
Bea	B	66	e	101	a	97	264	0
Zoe	Z	90	o	111	e	101	302	5
Sue	S	83	u	117	e	101	301	4

Collisions: Chaining



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

Collisions: Chaining



Collison Resolution

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

How probable are collisions?

Highly Probable!

Birthday paradox:

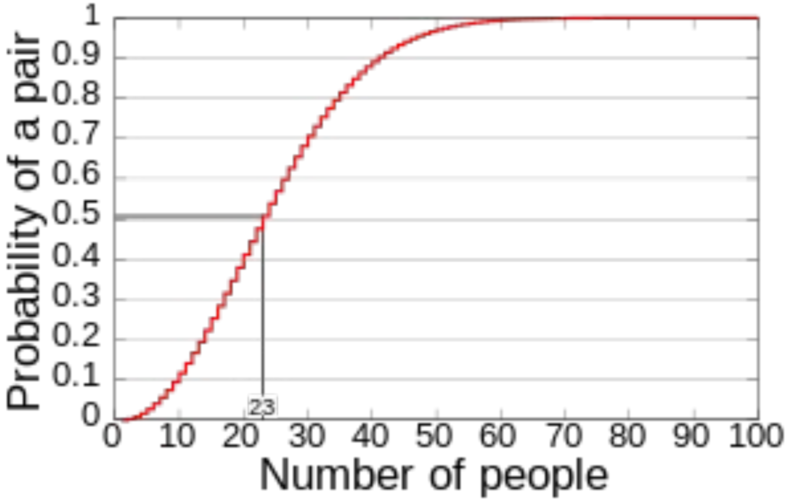
Given a room of ‘n’ people, at least how many people you require to find among them any two people with matching birthdays with 50% probability?

the mathematics is a bit counter-intuitive... the probability of a non-collision for 23 birthdays is:

$$p = \frac{366}{366} \times \frac{365}{366} \times \cdots \times \frac{344}{366} \approx 0.493$$

How probable are collisions?

Birthday Paradox



For 32-bit hash values: only 77,000 elements are required for significant risk (50%) of collision

Objectives of Hash Function

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

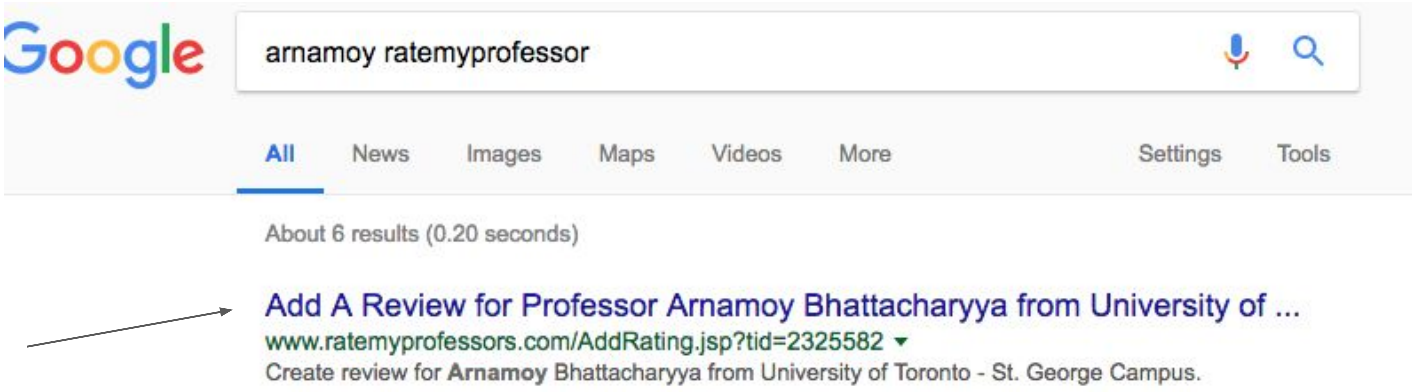
Next Lecture

Implementing dictionary using Chaining

Course Evaluations

Check your e-mail for a link to your evaluations, or log-in to www.portal.utoronto.ca and click the *Course Evals* tab!

Google Arnamoy ratemyprofessor



Developing our own Python Dictionary

Onto pycharm