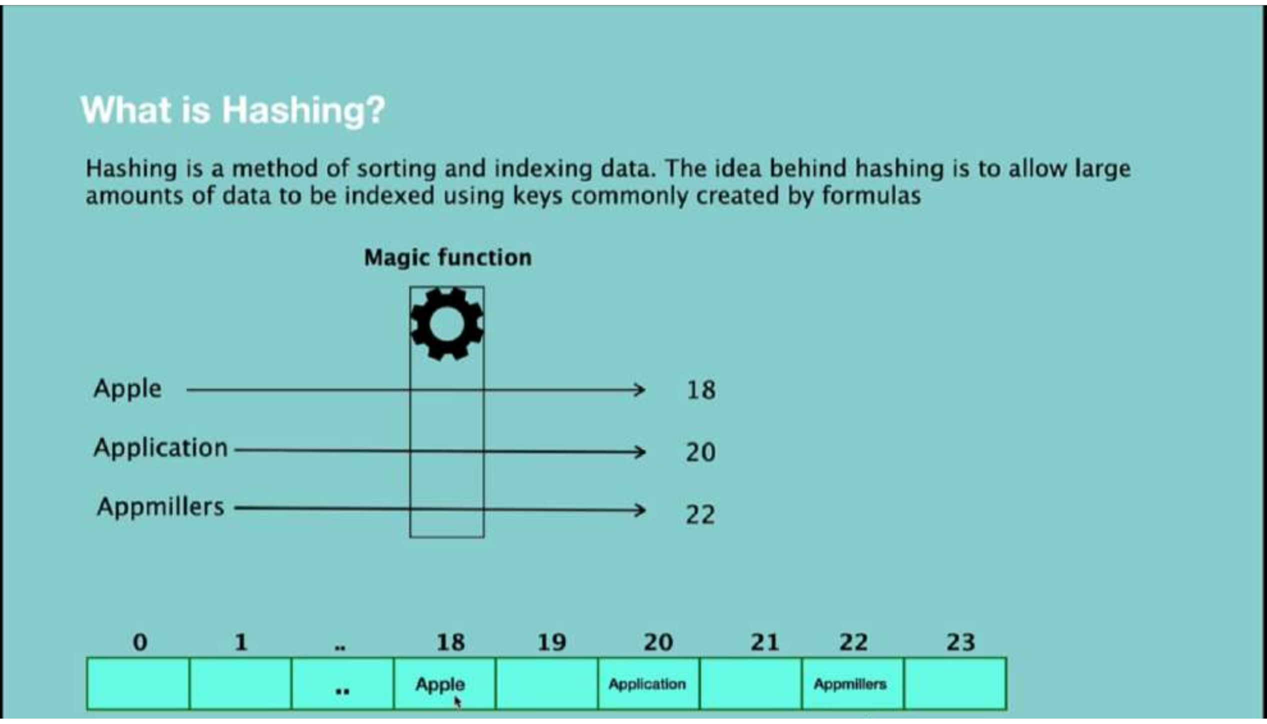
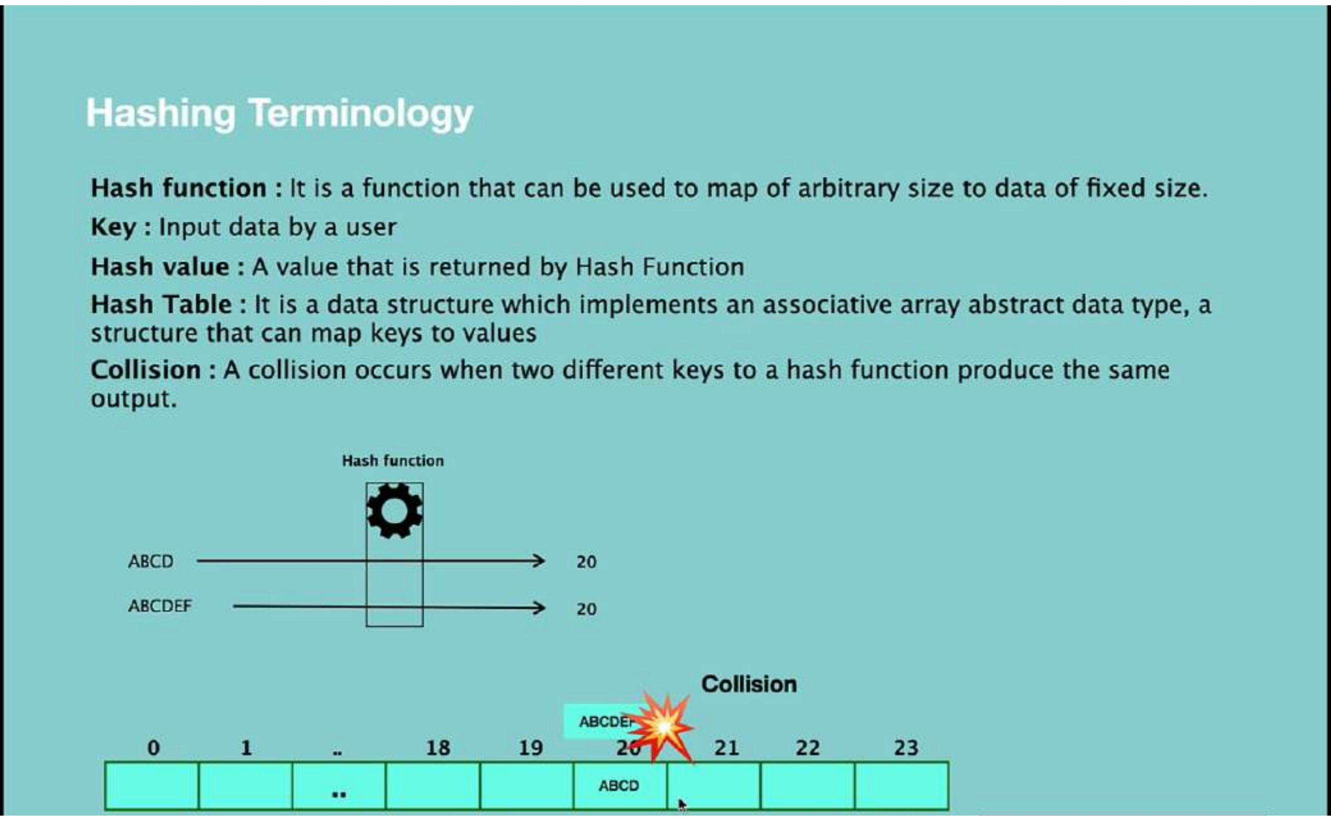


What is Hashing?

Hashing is a method of sorting and indexing data. The idea behind hashing is to allow large amounts of data to be indexed using keys commonly created by formulas



Hashing Terminology

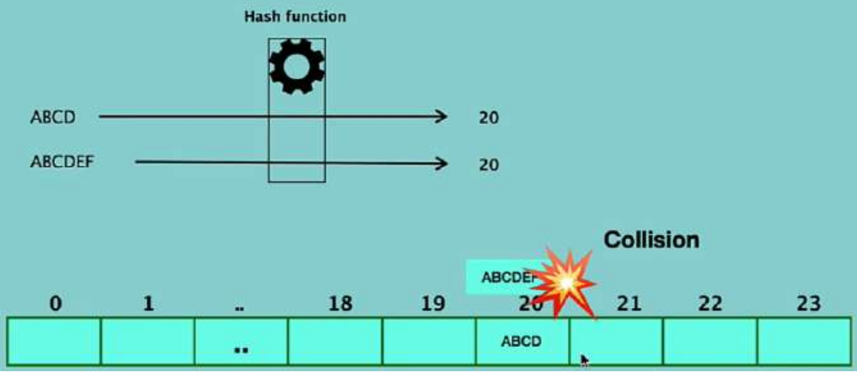
Hash function : It is a function that can be used to map of arbitrary size to data of fixed size.

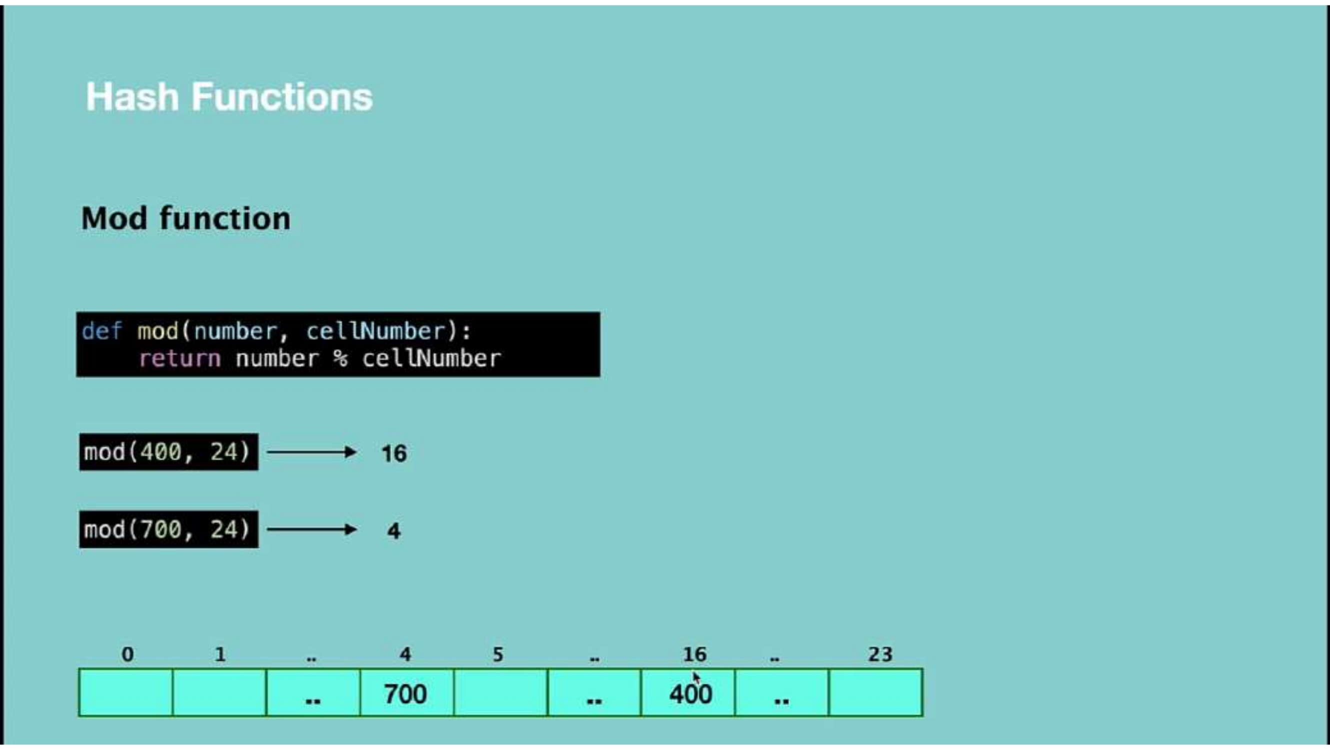
Key : Input data by a user

Hash value : A value that is returned by Hash Function

Hash Table : It is a data structure which implements an associative array abstract data type, a structure that can map keys to values

Collision : A collision occurs when two different keys to a hash function produce the same output.





Hash Functions

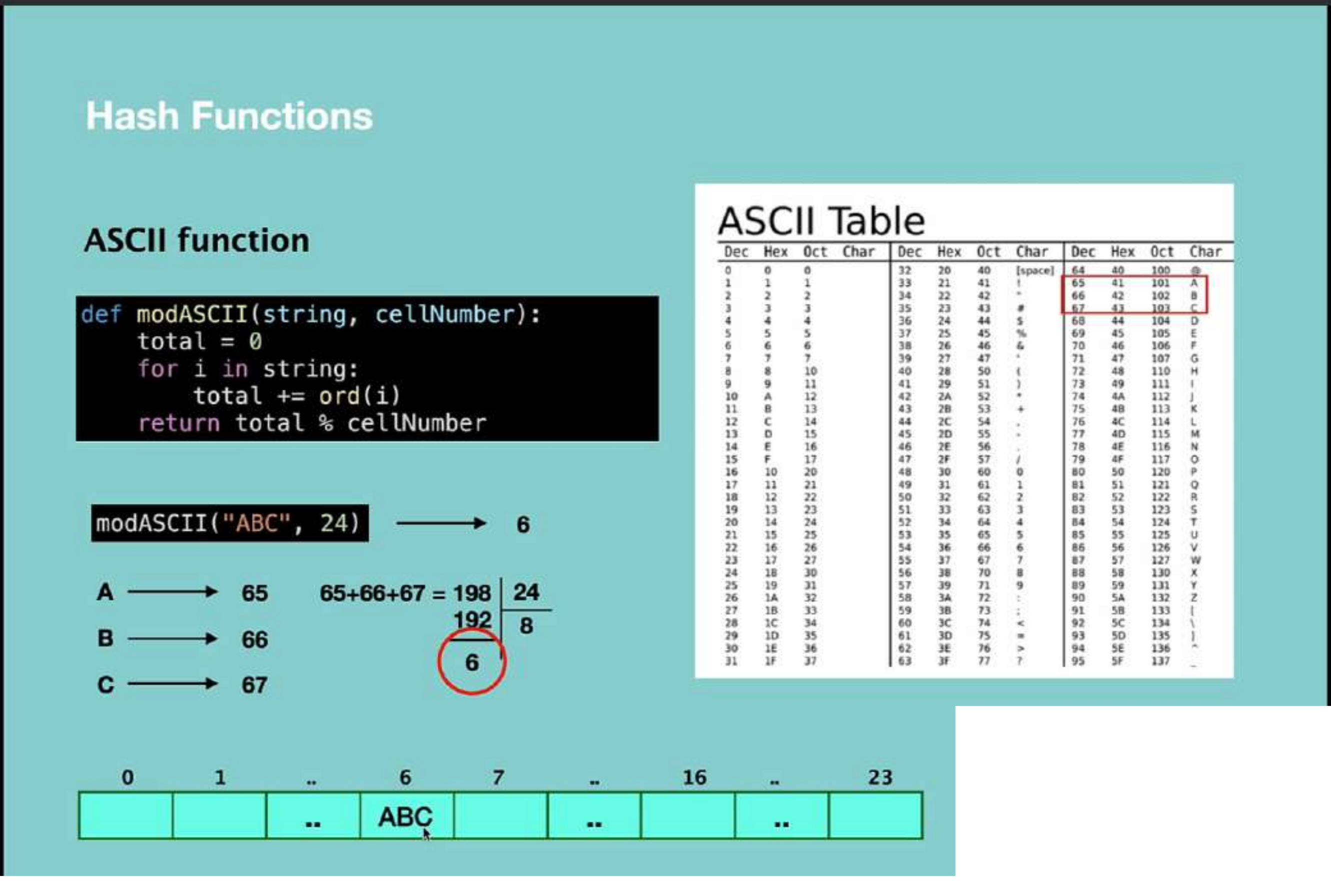
Mod function

```python
def mod(number, cellNumber):
    return number % cellNumber
```

`mod(400, 24)` $\longrightarrow$ 16

`mod(700, 24)` $\longrightarrow$ 4

0	1	..	4	5	..	16	..	23
		..	700		..	400	..	



Hash Functions

ASCII function

```python
def modASCII(string, cellNumber):
    total = 0
    for i in string:
        total += ord(i)
    return total % cellNumber
```

```
modASCII("ABC", 24)
```

$\longrightarrow$ 6

A $\longrightarrow$ 65

B $\longrightarrow$ 66

C $\longrightarrow$ 67

65+66+67 = 198 | 24

192 | 8

$\boxed{6}$

0	1	..	6	7	..	16	..	23
		..	ABC		..		..	

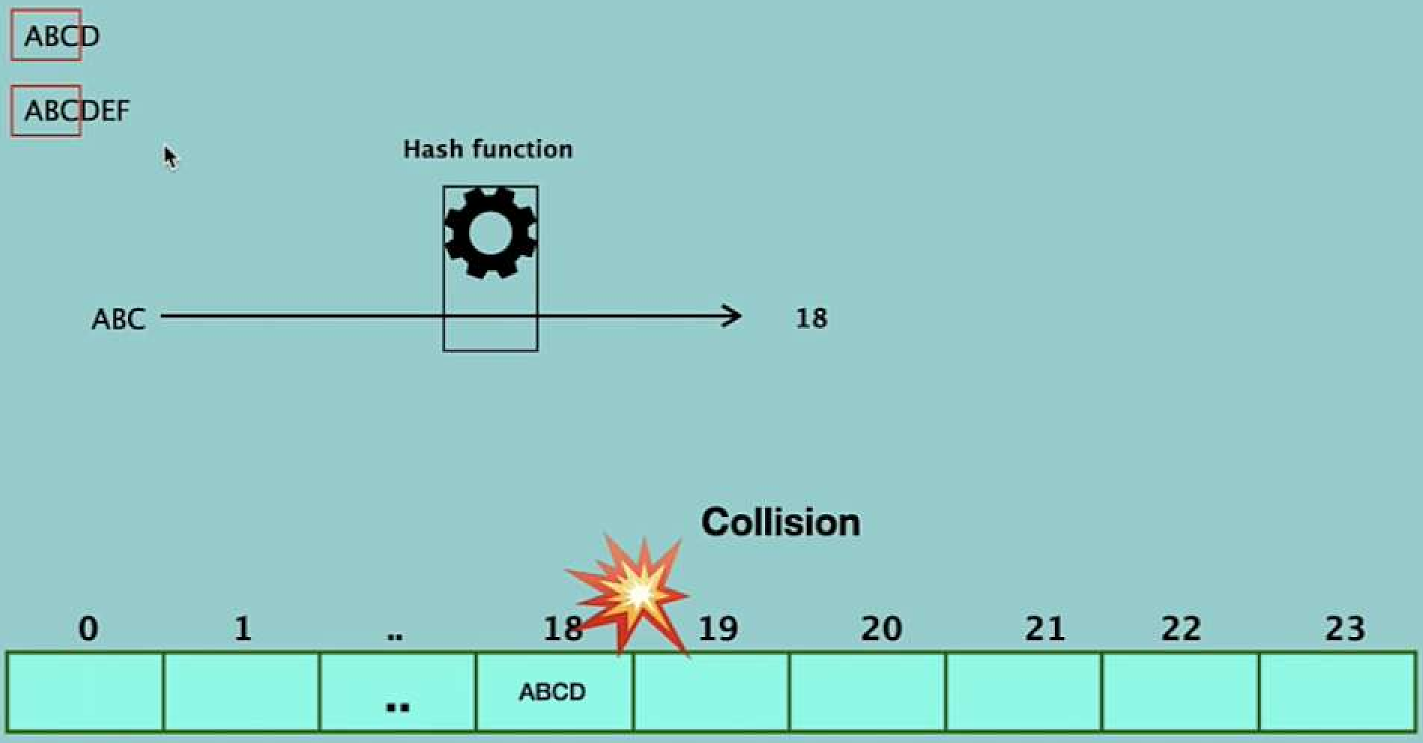
ASCII Table

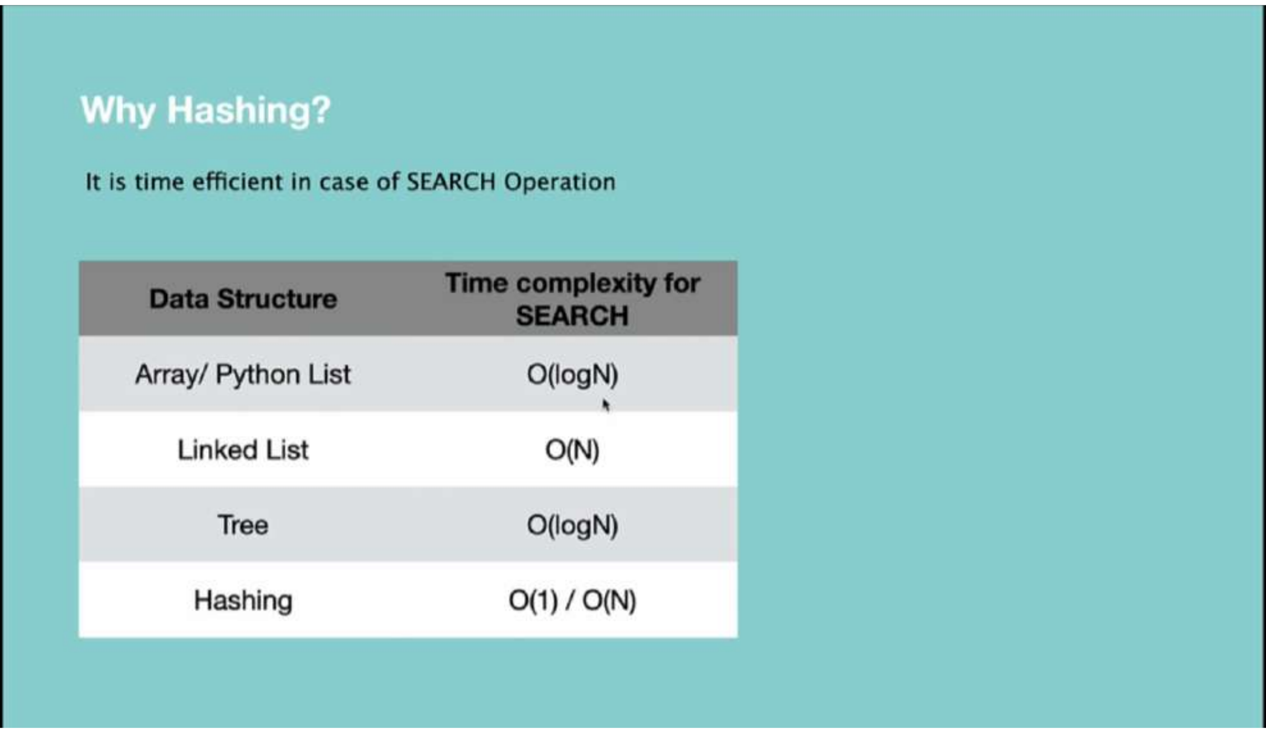
Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@
1	1	1		33	21	41	!	65	41	101	A
2	2	2		34	22	42	"	66	42	102	B
3	3	3		35	23	43	#	67	43	103	C
4	4	4		36	24	44	$	68	44	104	D
5	5	5		37	25	45	%	69	45	105	E
6	6	6		38	26	46	&	70	46	106	F
7	7	7		39	27	47	'	71	47	107	G
8	8	10		40	28	50	(	72	48	110	H
9	9	11		41	29	51	)	73	49	111	I
10	A	12		42	2A	52	*	74	4A	112	J
11	B	13		43	2B	53	+	75	4B	113	K
12	C	14		44	2C	54	,	76	4C	114	L
13	D	15		45	2D	55	-	77	4D	115	M
14	E	16		46	2E	56	.	78	4E	116	N
15	F	17		47	2F	57	/	79	4F	117	O
16	10	20		48	30	60	0	80	50	120	P
17	11	21		49	31	61	1	81	51	121	Q
18	12	22		50	32	62	2	82	52	122	R
19	13	23		51	33	63	3	83	53	123	S
20	14	24		52	34	64	4	84	54	124	T
21	15	25		53	35	65	5	85	55	125	U
22	16	26		54	36	66	6	86	56	126	V
23	17	27		55	37	67	7	87	57	127	W
24	18	30		56	38	70	8	88	58	130	X
25	19	31		57	39	71	9	89	59	131	Y
26	1A	32		58	3A	72	:	90	5A	132	Z
27	1B	33		59	3B	73	;	91	5B	133	[
28	1C	34		60	3C	74	<	92	5C	134	\
29	1D	35		61	3D	75	=	93	5D	135	]
30	1E	36		62	3E	76	>	94	5E	136	^
31	1F	37		63	3F	77	?	95	5F	137	_

Hash Functions

Properties of good Hash function

- It distributes hash values uniformly across hash tables
- It has to use all the input data



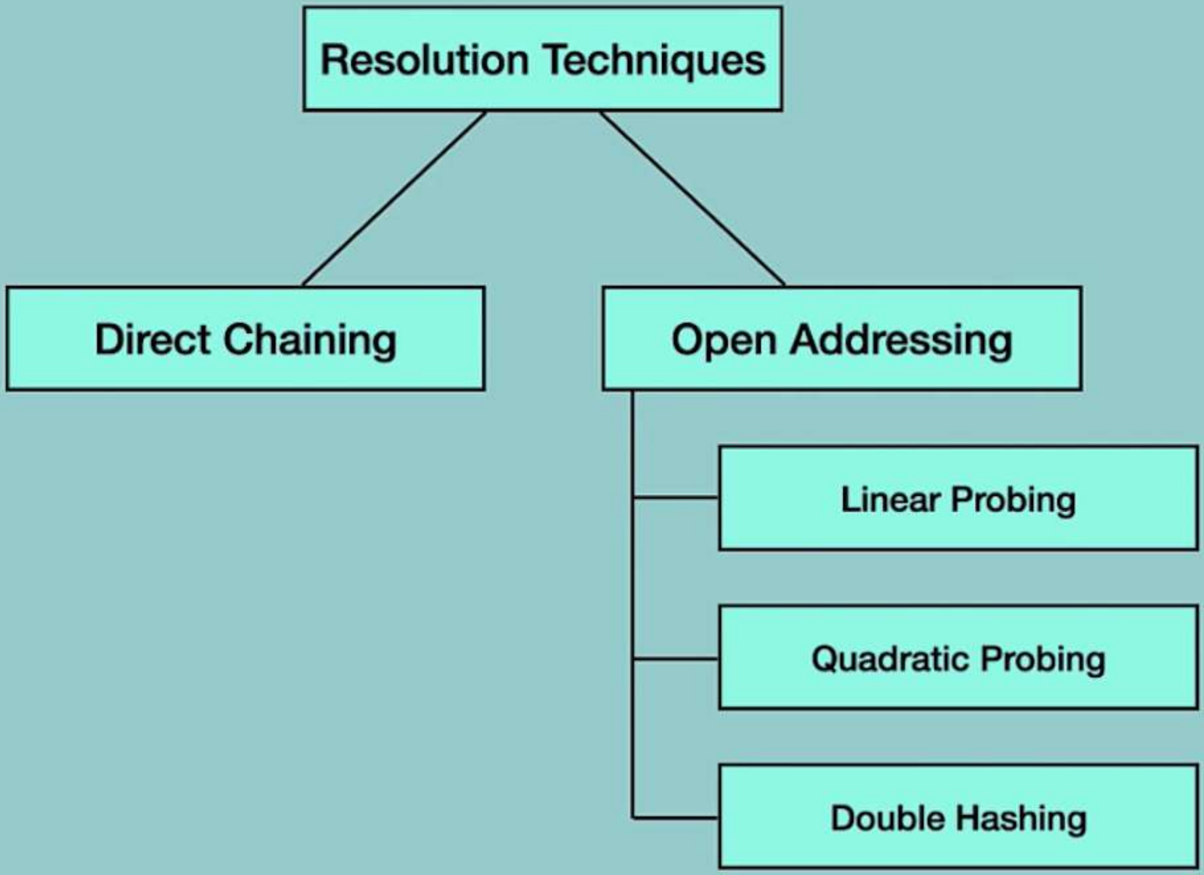


Why Hashing?

It is time efficient in case of SEARCH Operation

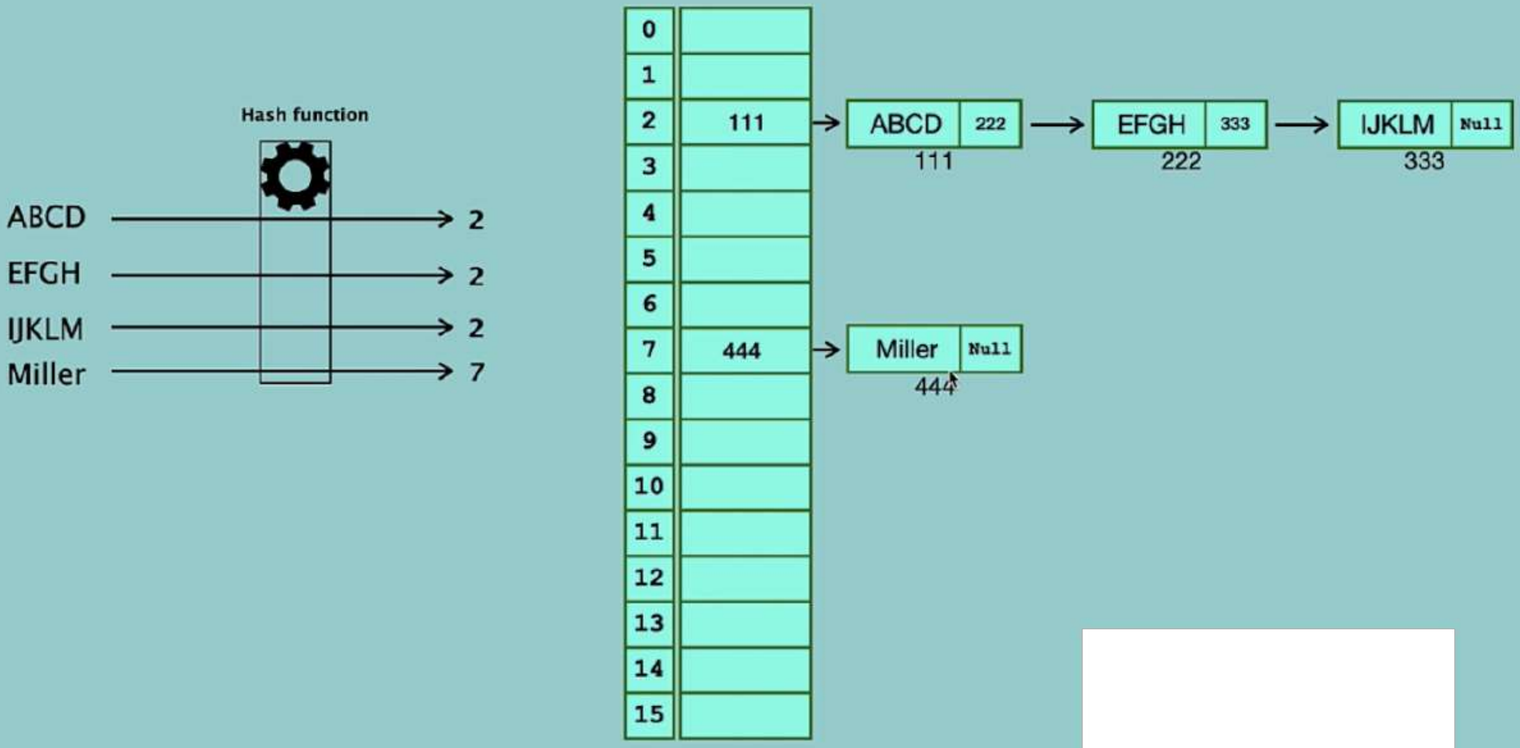
Data Structure	Time complexity for SEARCH
Array/ Python List	O(logN)
Linked List	O(N)
Tree	O(logN)
Hashing	O(1) / O(N)

Collision Resolution Techniques



Collision Resolution Techniques

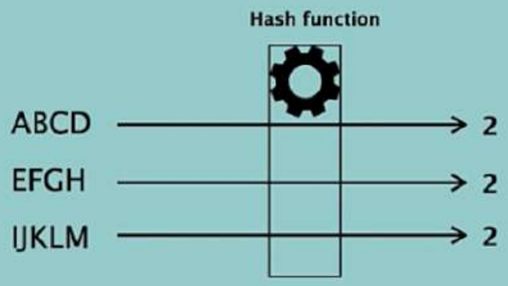
Direct Chaining : Implements the buckets as linked list. Colliding elements are stored in this lists



Collision Resolution Techniques

Open Addressing: Colliding elements are stored in other vacant buckets. During storage and lookup these are found through so called probing.

Linear probing : It places new key into closest following empty cell

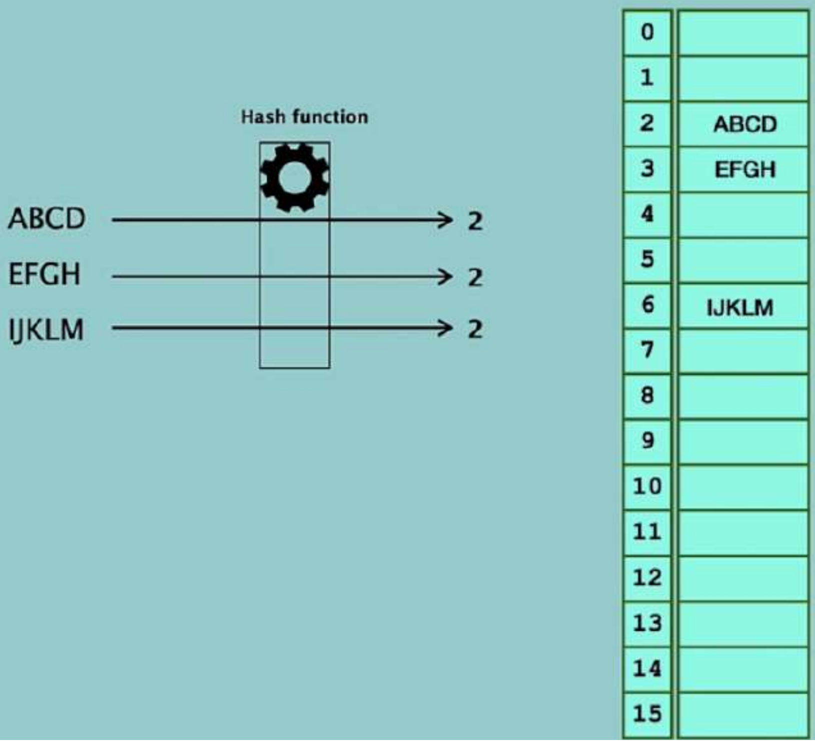


0	
1	
2	ABCD
3	EFGH
4	IJKLM
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

Collision Resolution Techniques

Open Addressing: Colliding elements are stored in other vacant buckets. During storage and lookup these are found through so called probing.

Quadratic probing : Adding arbitrary quadratic polynomial to the index until an empty cell is found



$1^2, 2^2, 3^2, 4^2..$

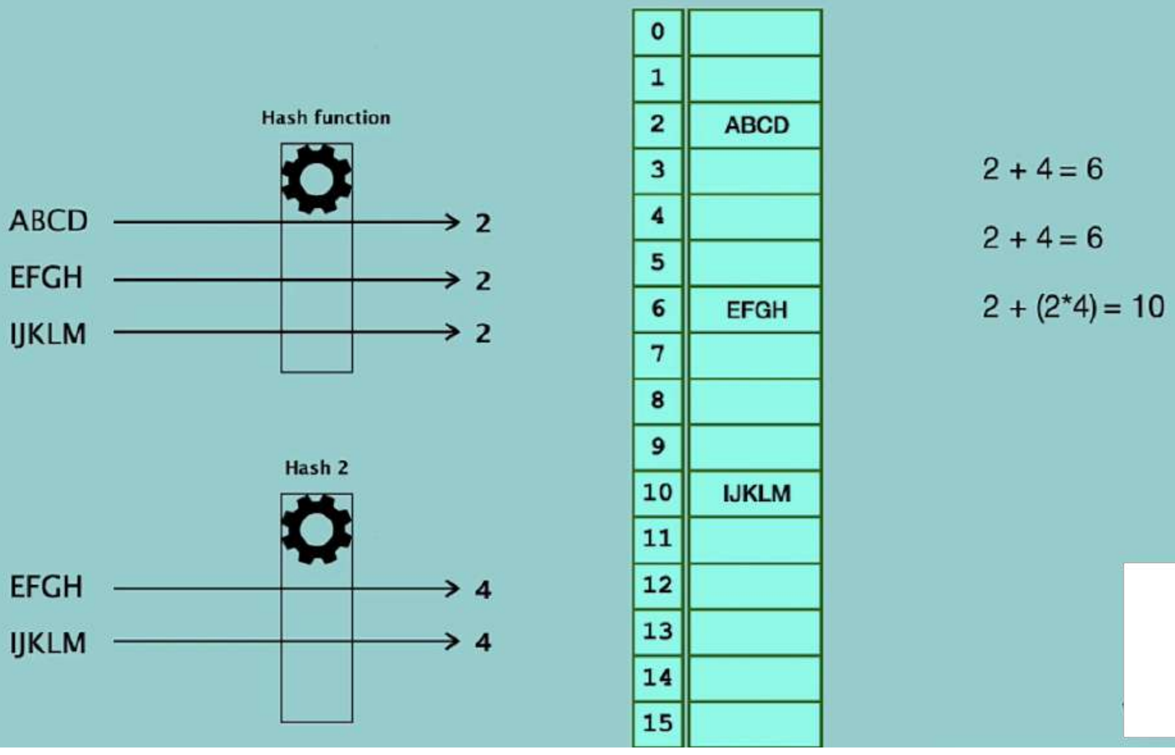
$2 + 1^2 = 3$

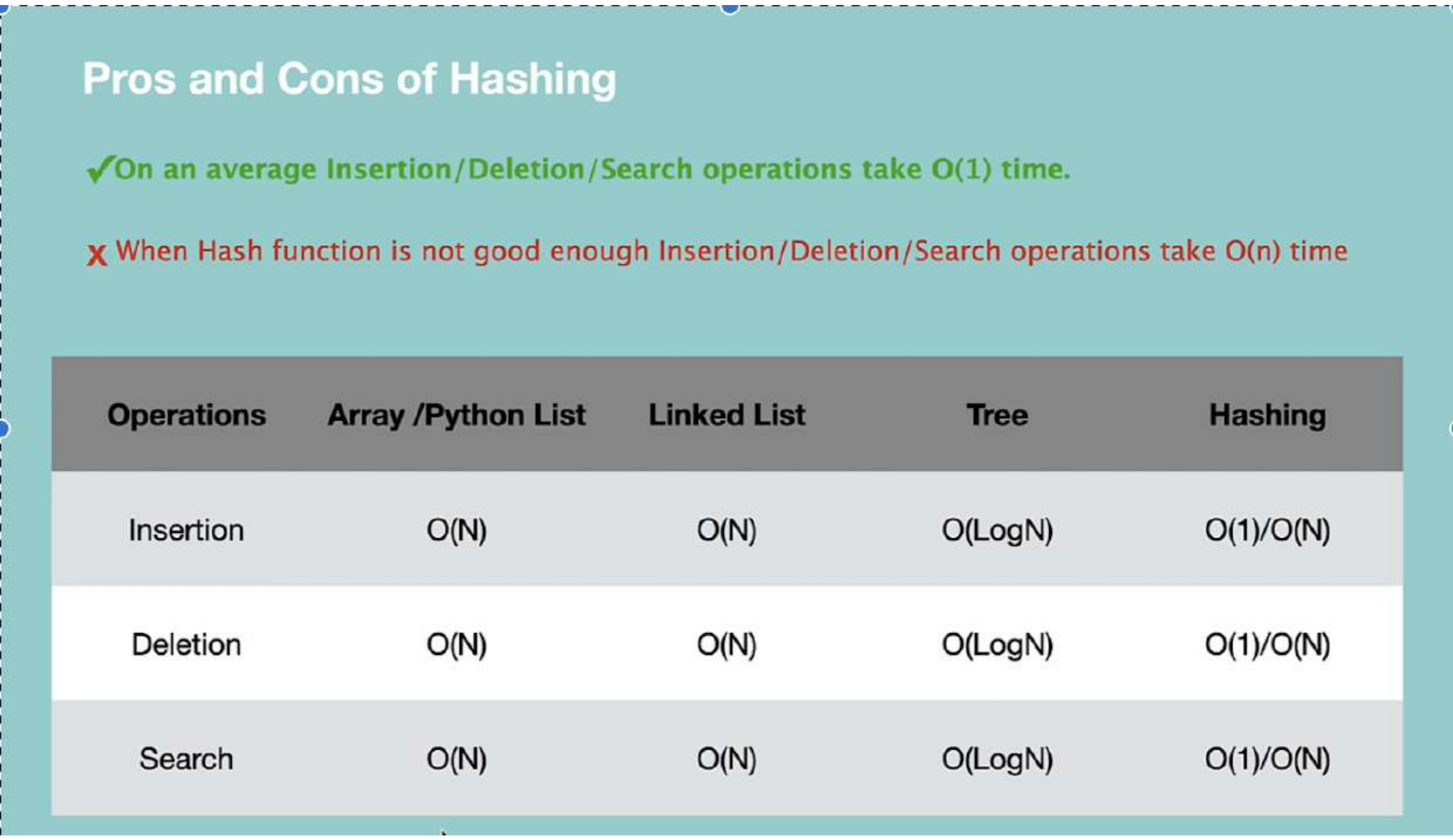
$2 + 2^2 = 6$

Collision Resolution Techniques

Open Addressing: Colliding elements are stored in other vacant buckets. During storage and lookup these are found through so called probing.

Double Hashing : Interval between probes is computed by another hash function





Pros and Cons of Hashing

✓On an average Insertion/Deletion/Search operations take O(1) time.

x When Hash function is not good enough Insertion/Deletion/Search operations take O(n) time

Operations	Array /Python List	Linked List	Tree	Hashing
Insertion	O(N)	O(N)	O(LogN)	O(1)/O(N)
Deletion	O(N)	O(N)	O(LogN)	O(1)/O(N)
Search	O(N)	O(N)	O(LogN)	O(1)/O(N)