Conception Avancée de Base de Données

Hash Concept

Traduction en cours

Alphabet Letters (26)

Alphabet

26 slots

Letter ASCII CODE as array index

CAR(A) = 65
CAR(B) = 66

INDEX (X) = CAR(X) - 64

A

J

Z

Alphabet Letters (26)

26 slots

Letter ASCII CODE as array index →

CAR(A) = 65
CAR(B) = 66

INDEX (X) = CAR(X) - 64

1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I
10	J
11	K
12	L
13	M
14	N
15	O
16	P
17	Q
18	R
19	S
20	T
21	U
22	V
23	W
24	X
25	Y
26	Z

Alphabet Letters (26)

26 slots

Letter ASCII CODE as array index →

1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I
10	J
11	K
12	L
13	M
14	N
15	O
16	P
17	Q
18	R
19	S
20	T
21	U
22	V
23	W
24	X
25	Y
26	Z

CAR(A) = 65
CAR(B) = 66

INDEX (X) = CAR(X) – 64

INDEX (K) = CAR (K) – 64 = 75 – 64 = 11

Alphabet Letters (26)

10 VALUES = WASTE 16 SLOTS

Letter ASCII CODE as array index →

1	A
2	B
3	
4	D
5	E
6	
7	
8	
9	
10	
11	K
12	L
13	
14	
15	O
16	
17	Q
18	
19	
20	T
21	
22	V
23	
24	
25	
26	Z

26 slots

CAR(A) = 65
CAR(B) = 66

INDEX (X) = CAR(X) – 64

INDEX (K) = CAR (K) – 64 = 75 – 64 = 11

Alphabet Letters (26)

10 VALUES = WASTE 16 SLOTS

Letter ASCII CODE as array index →

1	A
2	B
3	
4	D
5	E
6	
7	
8	
9	
10	
11	K
12	L
13	
14	
15	O
16	
17	Q
18	
19	
20	T
21	
22	V
23	
24	
25	
26	Z

CAR(A) = 65
CAR(B) = 66

INDEX (X) = CAR(X) – 64

INDEX (K) = CAR (K) – 64 = 75 – 64 = 11

Alphabet Letters (26)

10 VALUES = WASTE 16 SLOTS

Letter ASCII CODE as array index →

CAR(A) = 65
CAR(B) = 66

INDEX (X) = CAR(X) – 64

INDEX (K) = CAR (K) – 64 = 75 – 64 = 11

1	A
2	B
3	
4	D
5	E
6	
7	
8	
9	
10	
11	K
12	L
13	
14	
15	O
16	
17	Q
18	
19	
20	T
21	
22	V
23	
24	
25	
26	Z

Employees File

1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I
10	J
11	K
12	L
13	M
14	N
15	O
16	P
17	Q
18	R
19	S
20	T
21	U
22	V
23	W
24	X
25	Y
26	Z

26 slots

1	A
2	B
3	E
4	K
5	L
6	O
7	Q
8	T
9	V
10	Z

10 slots

Employees File

1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I
10	J
11	K
12	L
13	M
14	N
15	O
16	P
17	Q
18	R
19	S
20	T
21	U
22	V
23	W
24	X
25	Y
26	Z

1	A
2	B
3	E
4	K
5	L
6	O
7	Q
8	T
9	V
10	Z

INDEX (K) = HASH (K) = 4

Employees File

1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I
10	J
11	K
12	L
13	M
14	N
15	O
16	P
17	Q
18	R
19	S
20	T
21	U
22	V
23	W
24	X
25	Y
26	Z

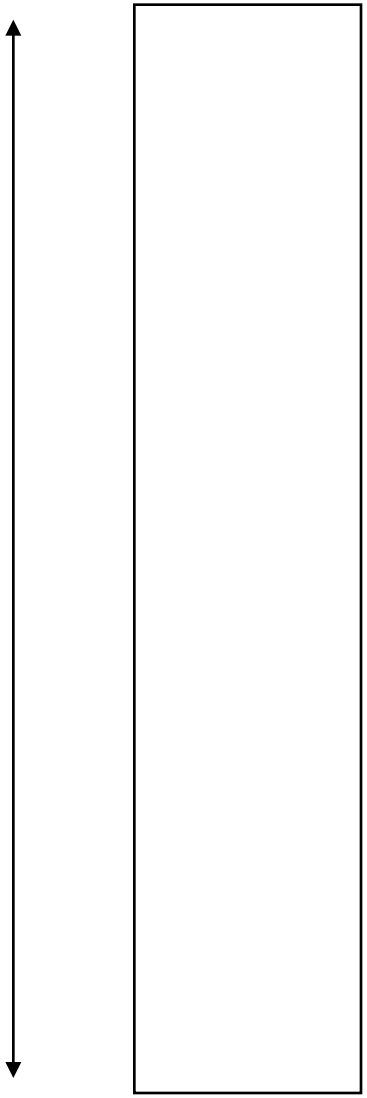
1	A
2	B
3	E
4	K
5	L
6	O
7	Q
8	T
9	V
10	Z

INDEX (K) = HASH (K) = 4

26 slots ⟶ 10 slots

Employees File

SSN : Social Security Number



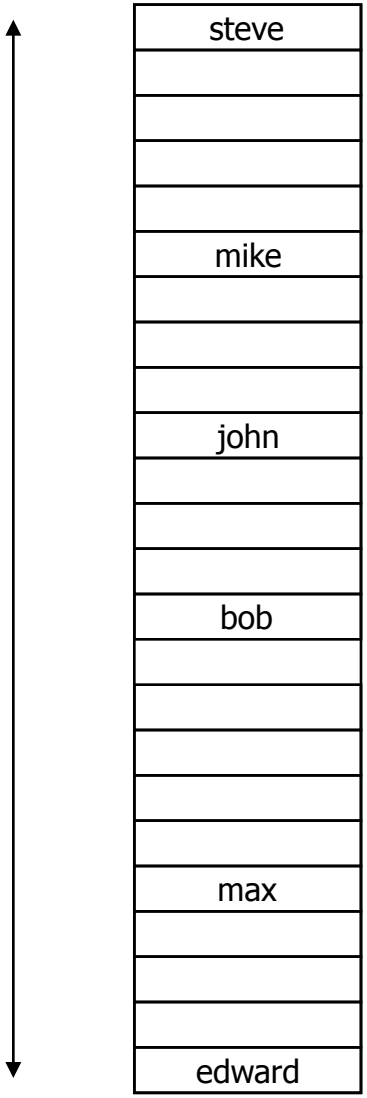
Social Security Number = address

Employees File

SSN : Social Security Number

John SSN : 125675798988090

Social Security Number = address



999999999999999

125675798988090

000000000000000

Employees File

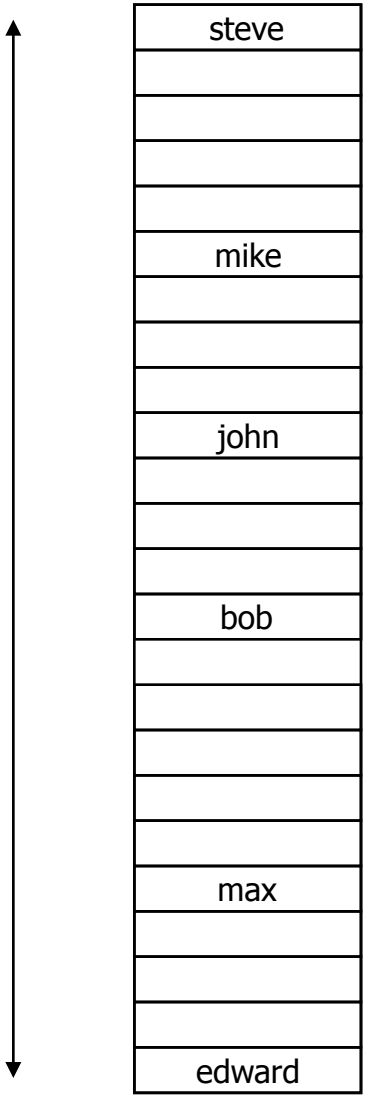
SSN : Social Security Number

John SSN : 125675798988090

Social Security Number = address

Requires Memory size : 999999999999999

= 1E+15 = 1 *petaoctet* (Po)



999999999999999

125675798988090

000000000000000

Employees File

SSN : Social Security Number

N element Array

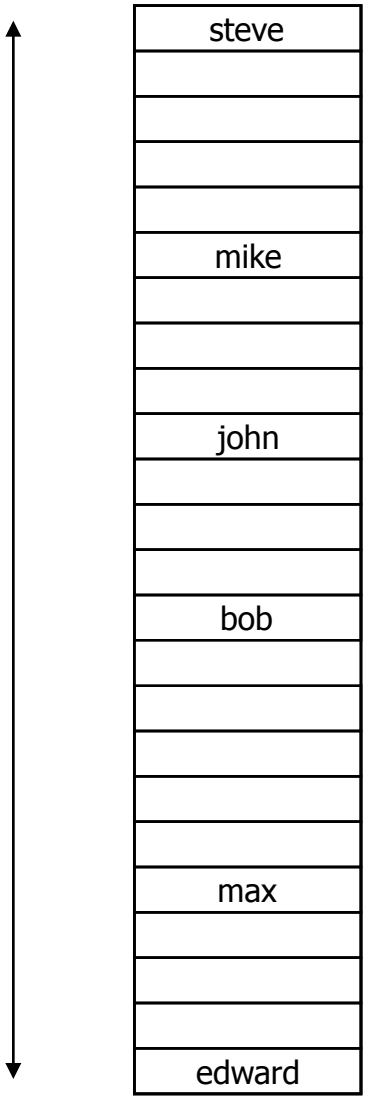
SSN

125675798988090

99

00

100 employees



999999999999999

125675798988090

000000000000000

Employees File

SSN : Social Security Number

N element Array

SSN

125675798988090

steve	99
bob	
john	
mike	
edward	00

steve	999999999999999
mike	
john	125675798988090
bob	
max	
edward	000000000000000

Employees File

SSN : Social Security Number

N element Array

SSN

125675798988090

steve	99
bob	
john	
mike	
edward	00

steve	999999999999999
mike	
john	125675798988090
bob	
max	
edward	000000000000000

Employees File

SSN : Social Security Number

N element Array

addresses

steve
bob
john
mike
edward

value

99

hash

00

steve
mike
john
bob
max
edward

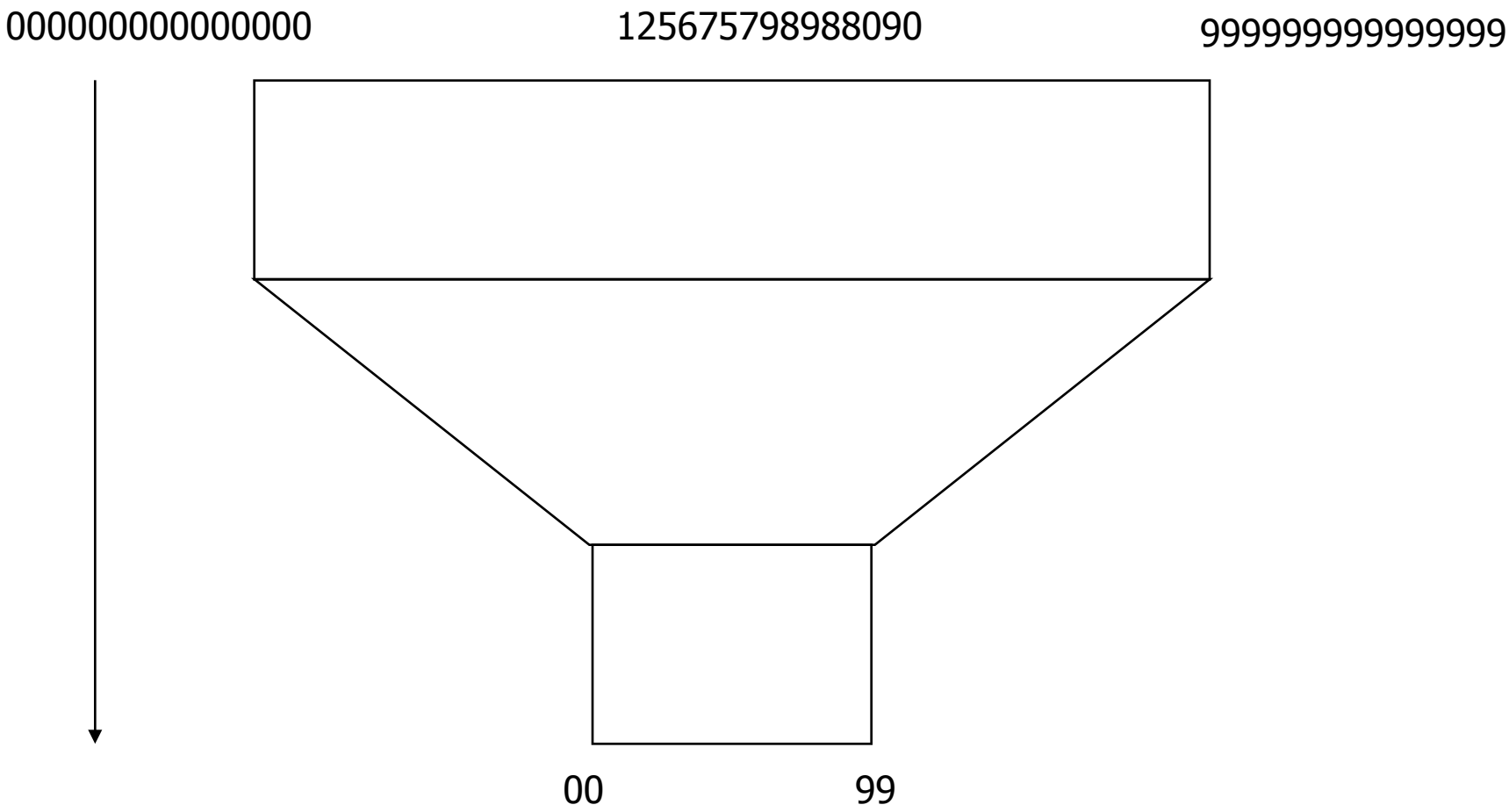
999999999999999

125675798988090

KEY

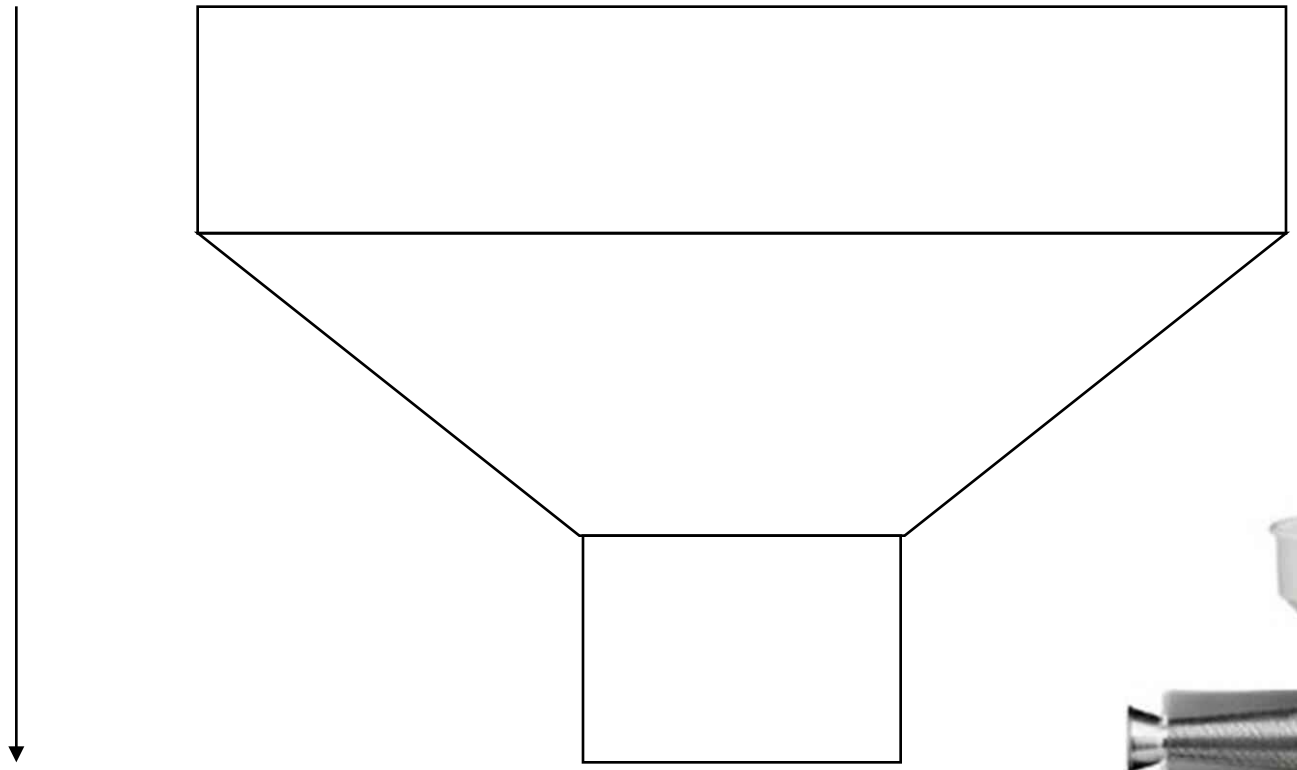
000000000000000

Hashing



Hashing

000000000000000 125675798988090 999999999999999

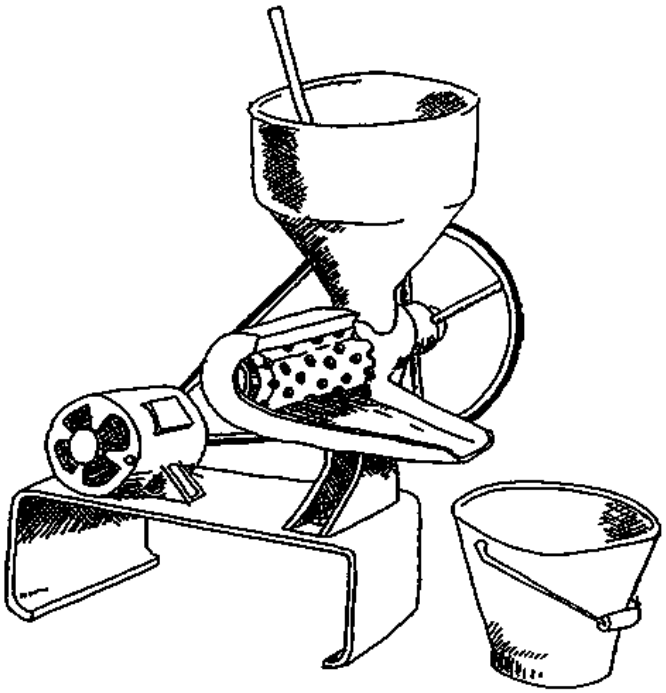


Hashing

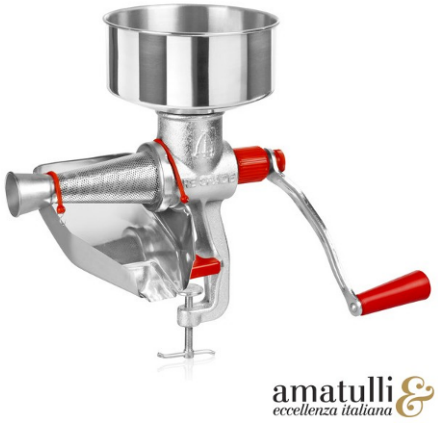
000000000000000 125675798988090 999999999999999

00 99

Hashing




amatulli &
eccellenza italiana
amatulli &
eccellenza italiana


amatulli &
eccellenza italiana

Employees File

SSN : Social Security Number

N element Array

Address space

addresses

john
bob
mike
edward

99

00

value

hash

steve
mike
john
bob
max
edward

999999999999999

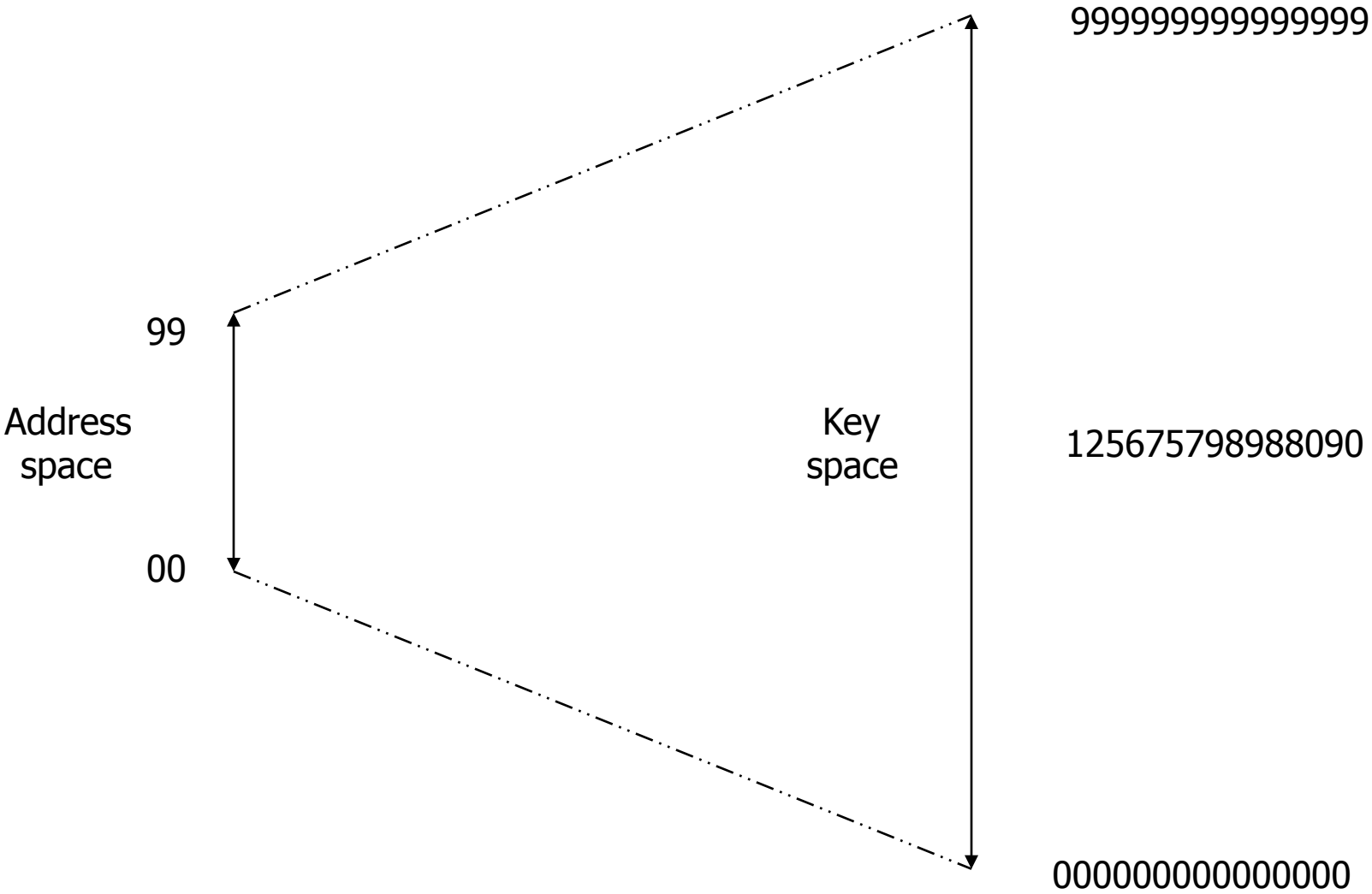
125675798988090

KEY

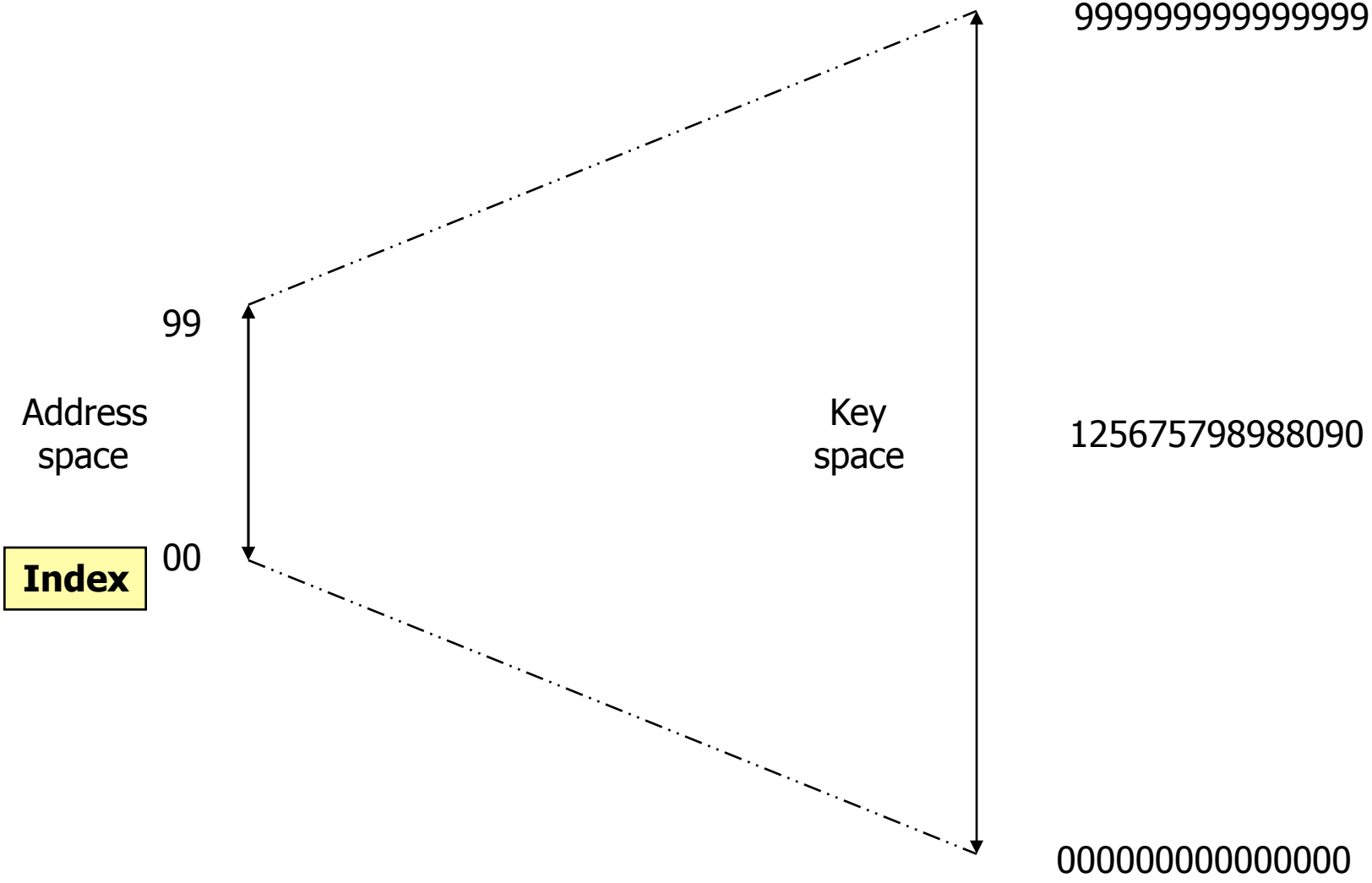
000000000000000

Key space

Key space to Address space mapping

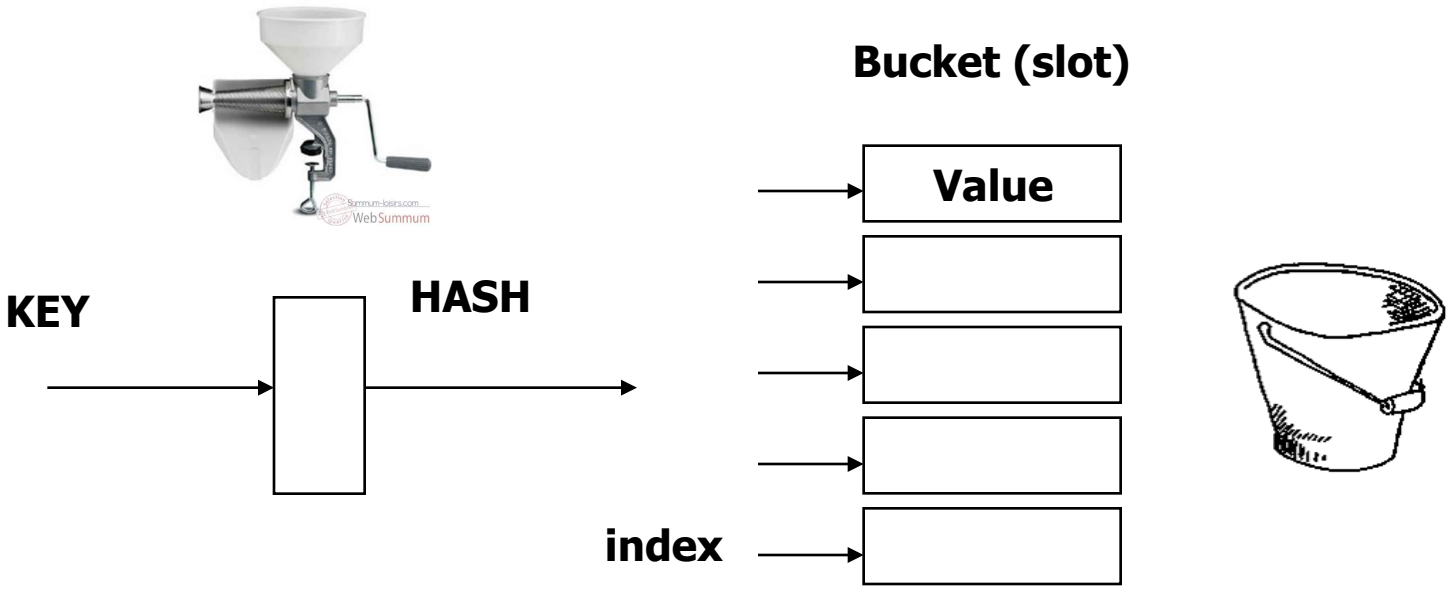


Key space to Address space mapping

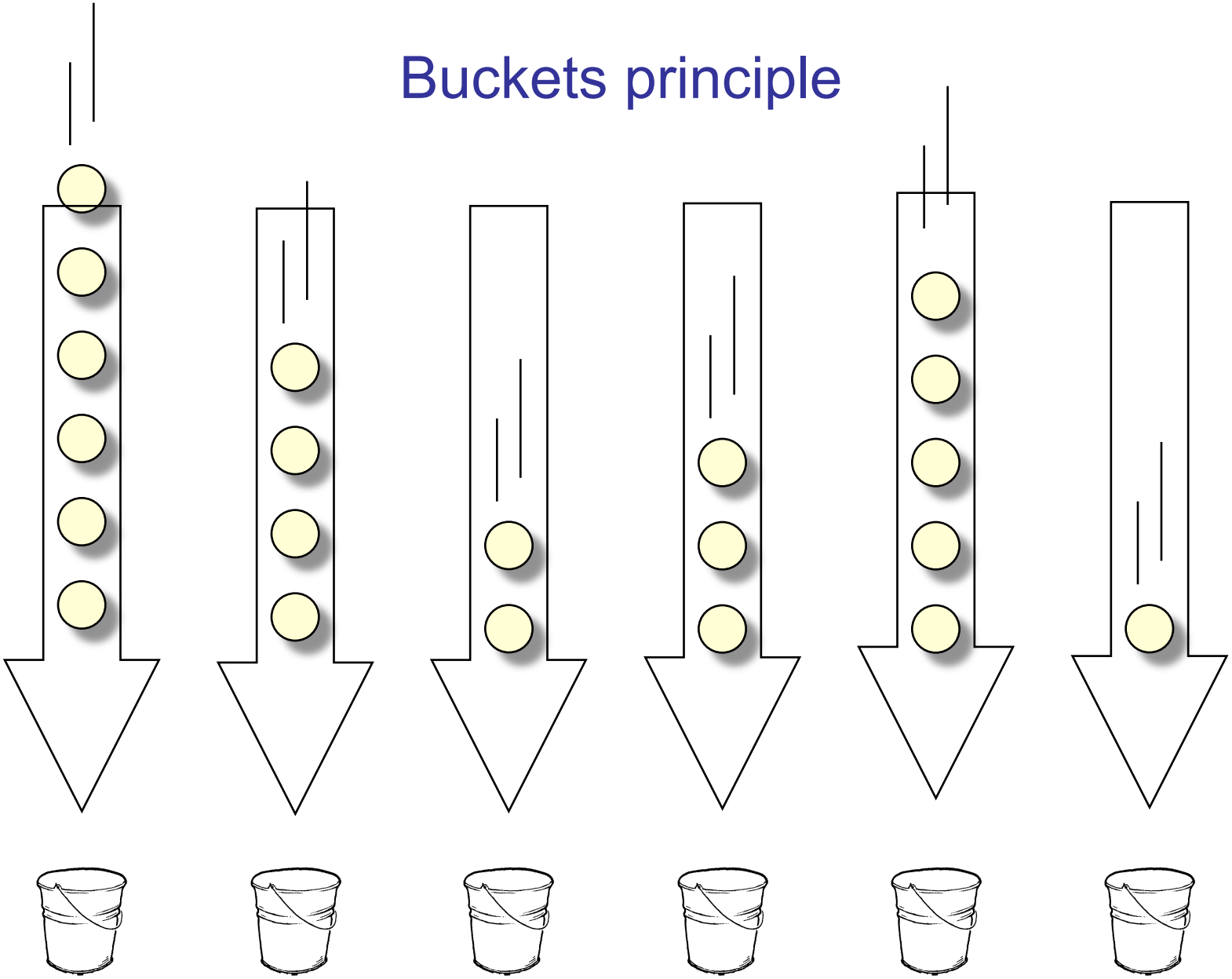


Hash Function

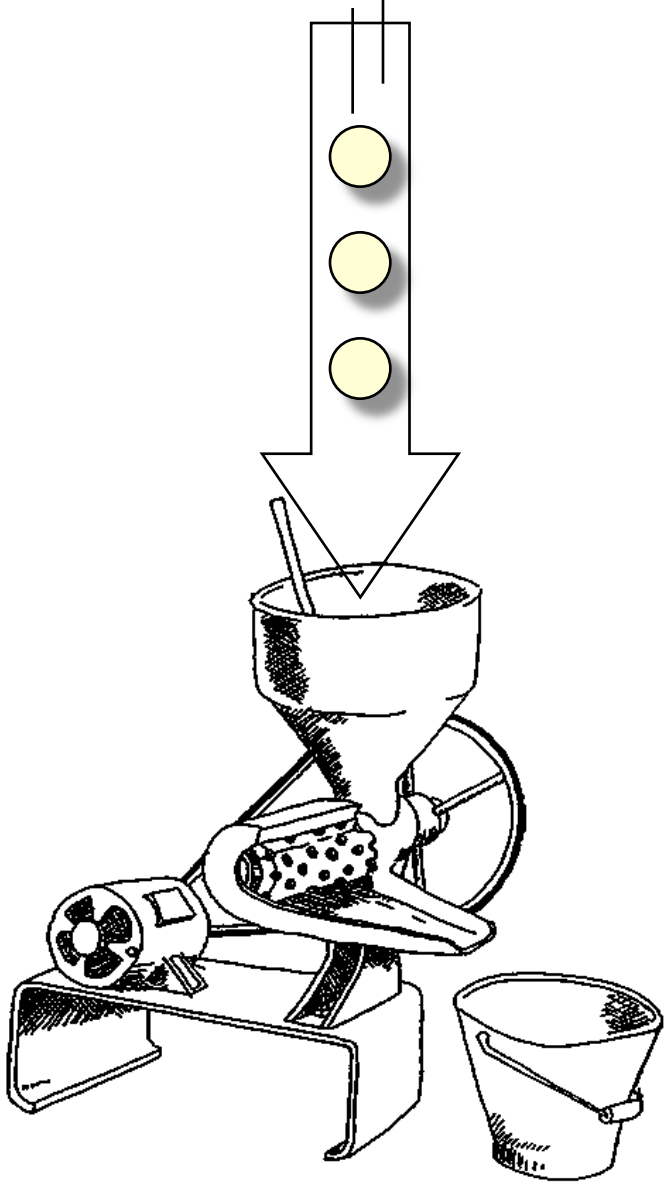
- The hash function is used to transform the key into the index (the hash) of an array element (the slot or bucket) where the corresponding value is to be stored and sought.



Buckets principle



Buckets principle



Hash table

- hash table is an array-based data structure.

index →

	Key	Value
0	129007	Fred
1	926647	Steve
2		
3	975378	Richard
4		
5	269908	Robert

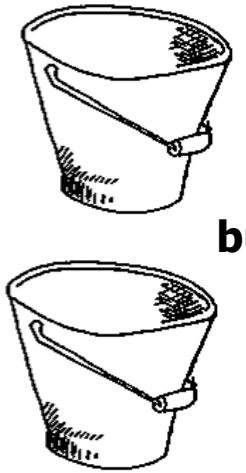
SSN

Hash table

- hash function is used to convert the key into the index of an array element, where associated value is to be seek.

Key			index	Value
129007	→ h(129007)	→	0	Fred
926647	→ h(129007)	→	1	Steve
			2	
975378	→ h(975378)	→	3	Richard
			4	
269908	→ h(269908)	→	5	Robert

SSN



buckets

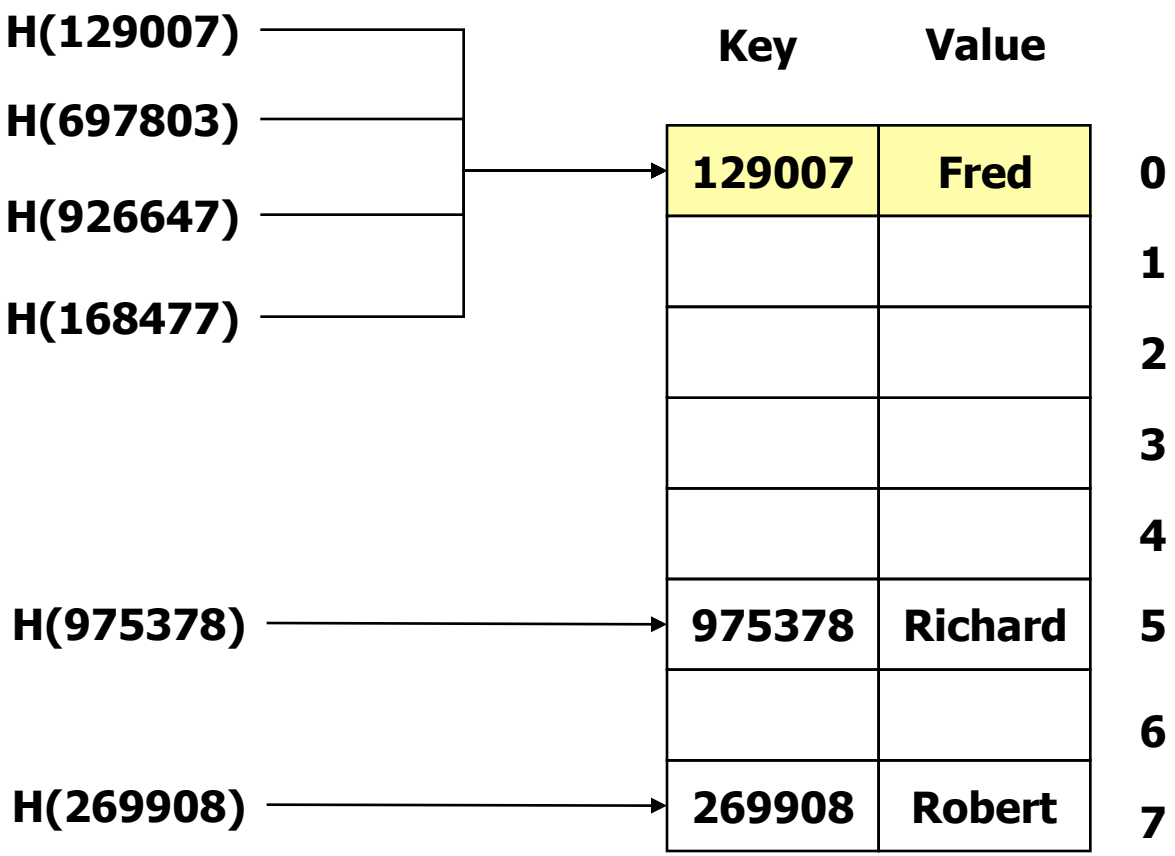
Collision

- If the hash function returns a slot that is already occupied there is a collision

	Key	Value	
H(129007)	129007	Fred	0
H(926647)			1
H(975378)	975378	Richard	2
			3
H(269908)	269908	Robert	4

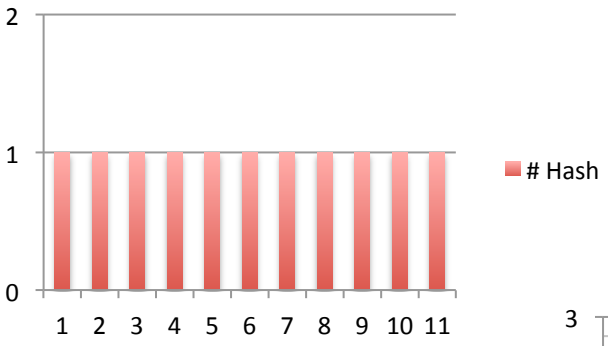
hash clustering

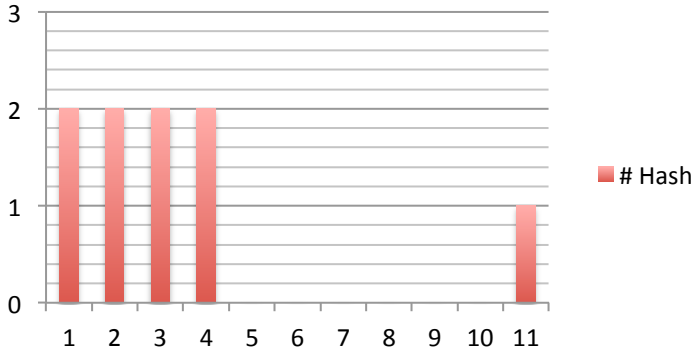
- When the distribution of keys into buckets is not random, we say that the hash table exhibits clustering.

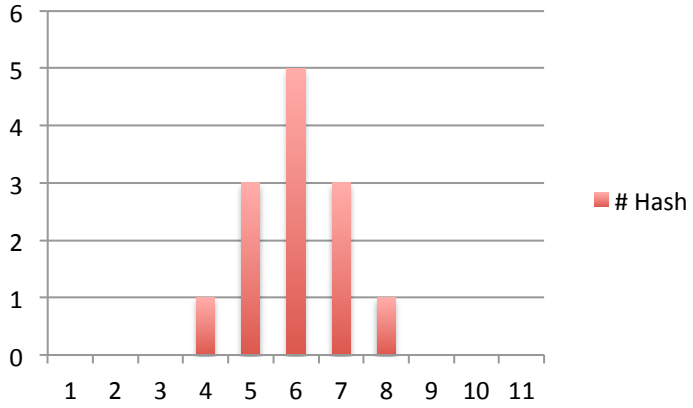


Hash function

- Good Hash function provides uniform distribution of hash values.
- Poor hash function will cause collisions and hash cluster.

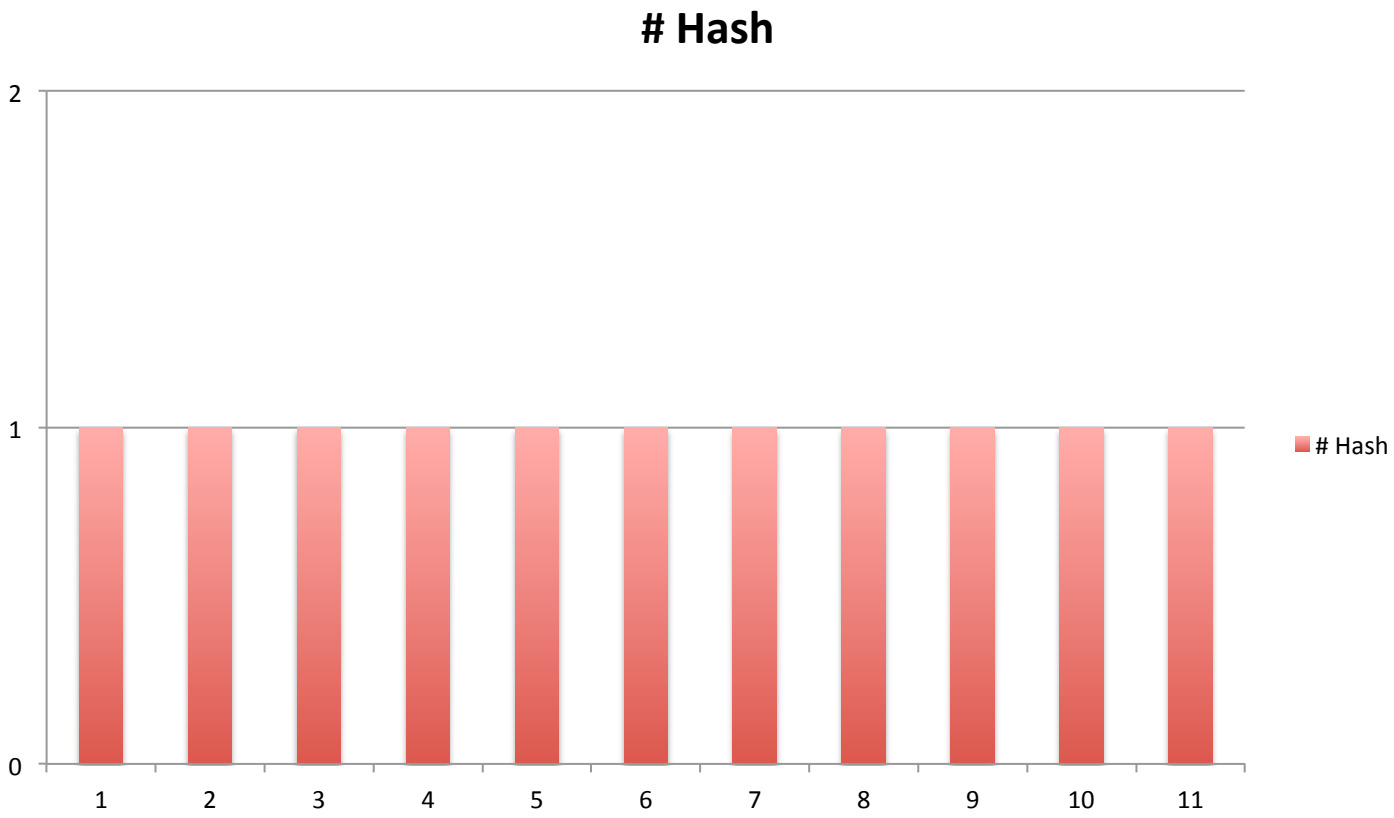






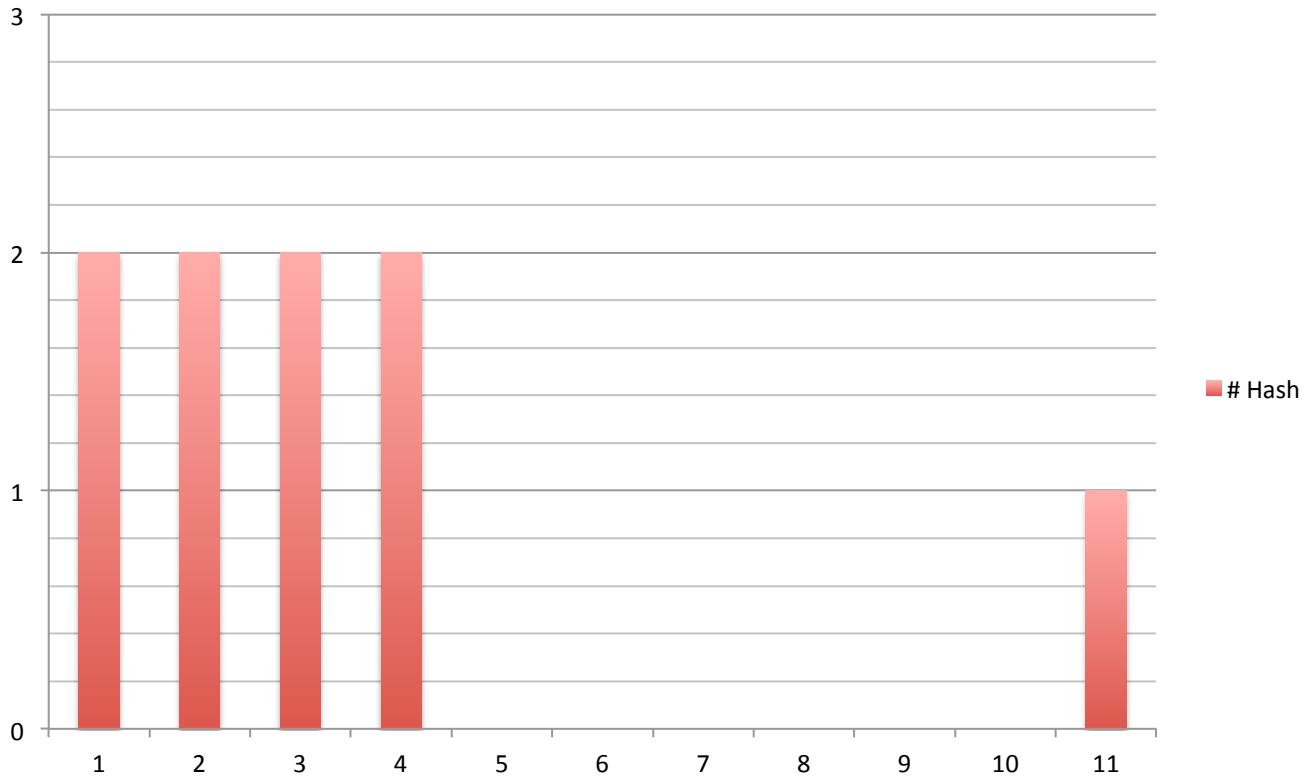
Hash Distribution

Value	# Hash
0	1
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1



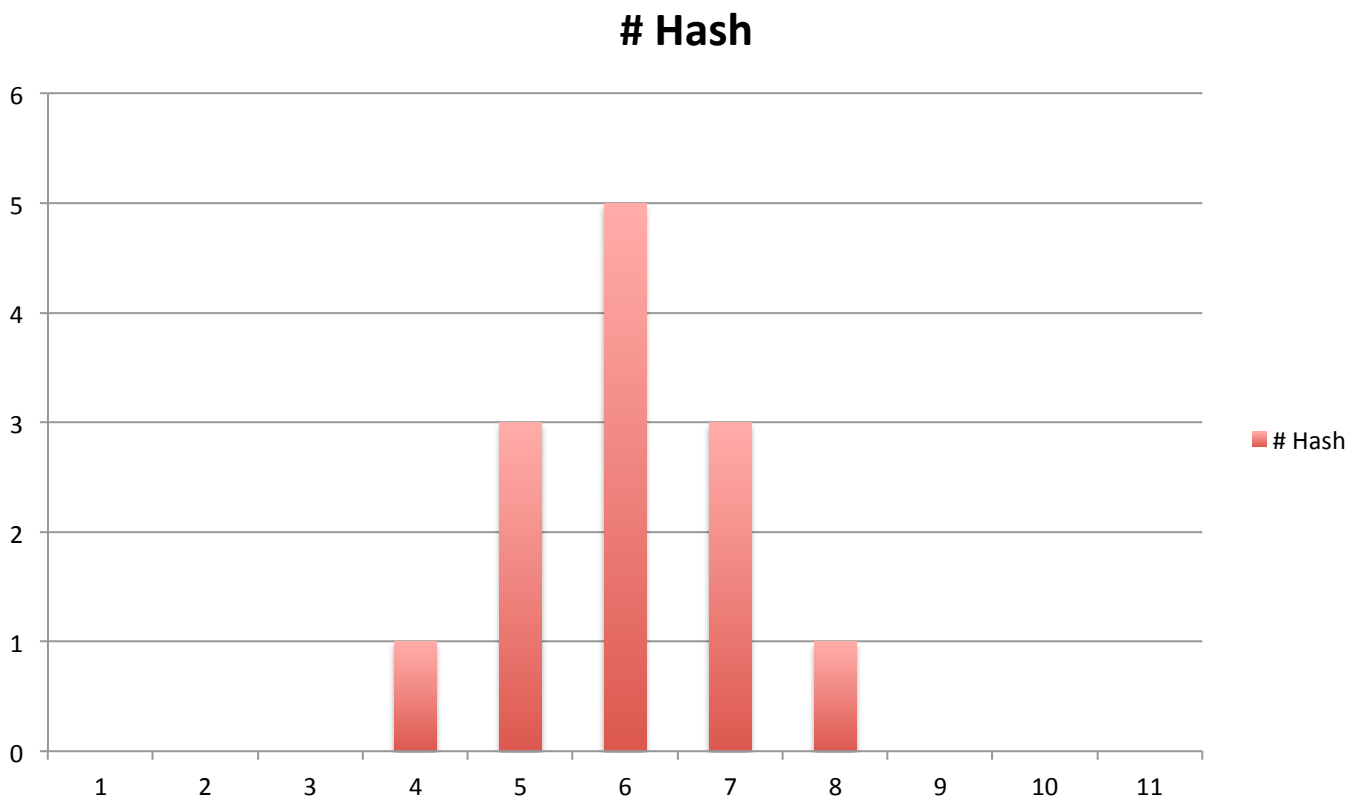
Hash Distribution

Value	# Hash
0	2
1	2
2	2
3	2
4	0
5	0
6	0
7	0
8	0
9	0
10	1



Hash Distribution

Value	# Hash
0	0
1	0
2	0
3	1
4	3
5	5
6	3
7	1
8	0
9	0
10	0



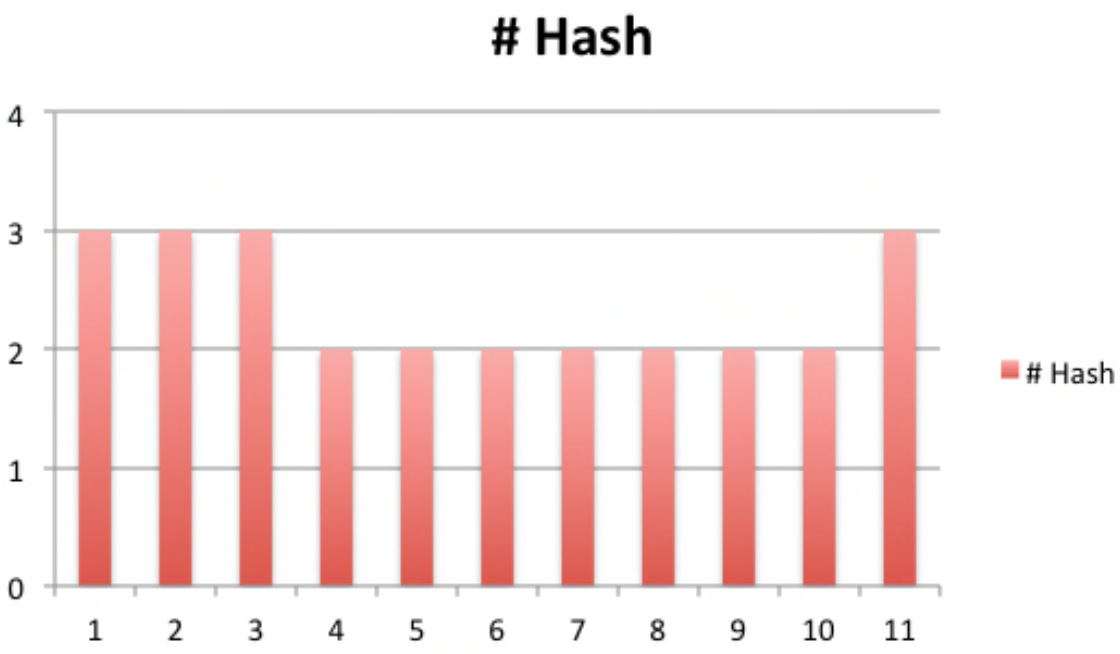
Hash function toy implementation : Modulo N

R	CAR(R)	mod (11)
A	65	10
B	66	0
C	67	1
D	68	2
E	69	3
F	70	4
G	71	5
H	72	6
I	73	7
J	74	8
K	75	9
L	76	10
M	77	0
N	78	1
O	79	2
P	80	3
Q	81	4
R	82	5
S	83	6
T	84	7
U	85	8
V	86	9
W	87	10
X	88	0
Y	89	1
Z	90	2

26 => 11

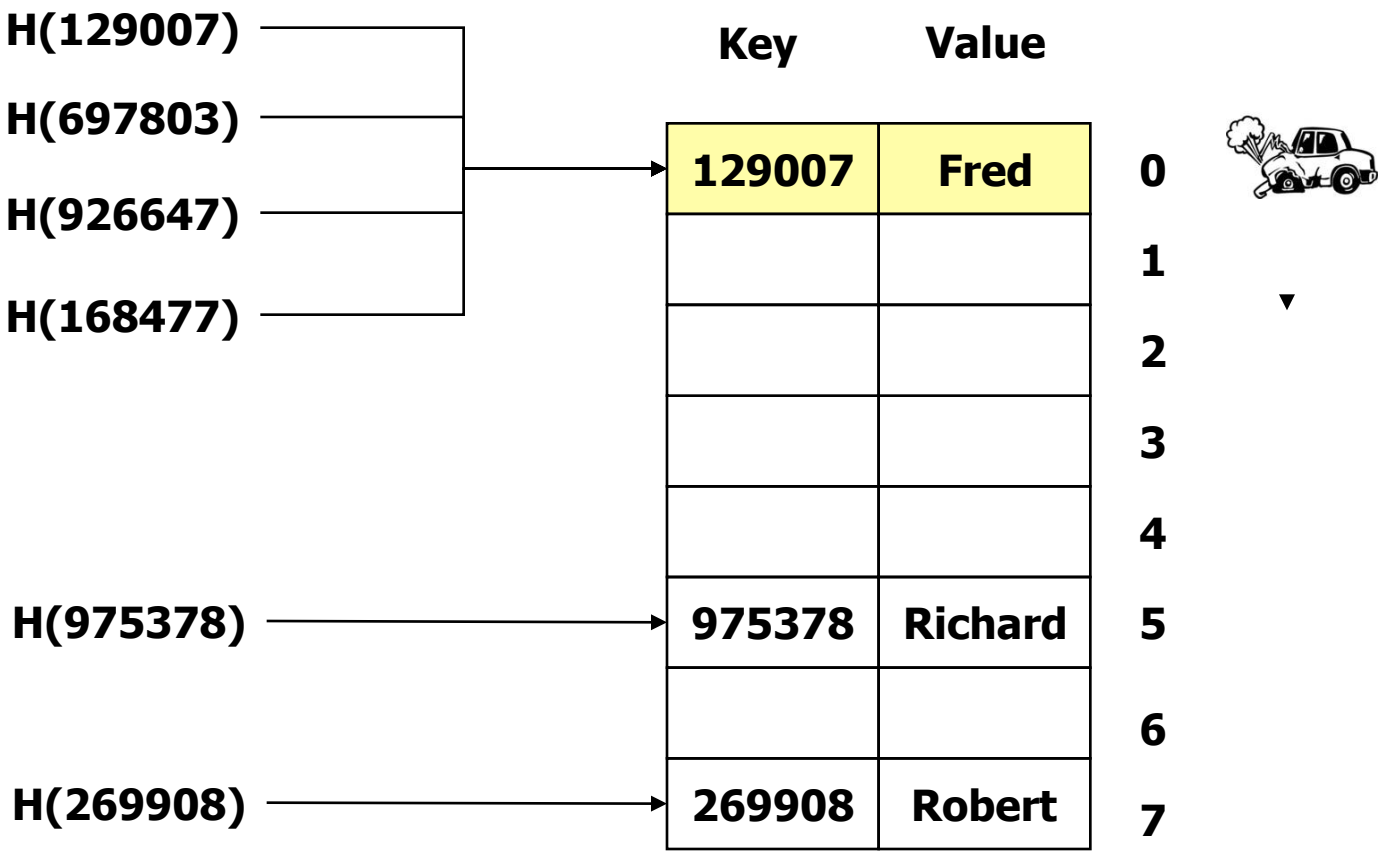
Modulo N Hash Distribution

R	CAR(R)	mod (11)
A	65	10
B	66	0
C	67	1
D	68	2
E	69	3
F	70	4
G	71	5
H	72	6
I	73	7
J	74	8
K	75	9
L	76	10
M	77	0
N	78	1
O	79	2
P	80	3
Q	81	4
R	82	5
S	83	6
T	84	7
U	85	8
V	86	9
W	87	10
X	88	0
Y	89	1
Z	90	2



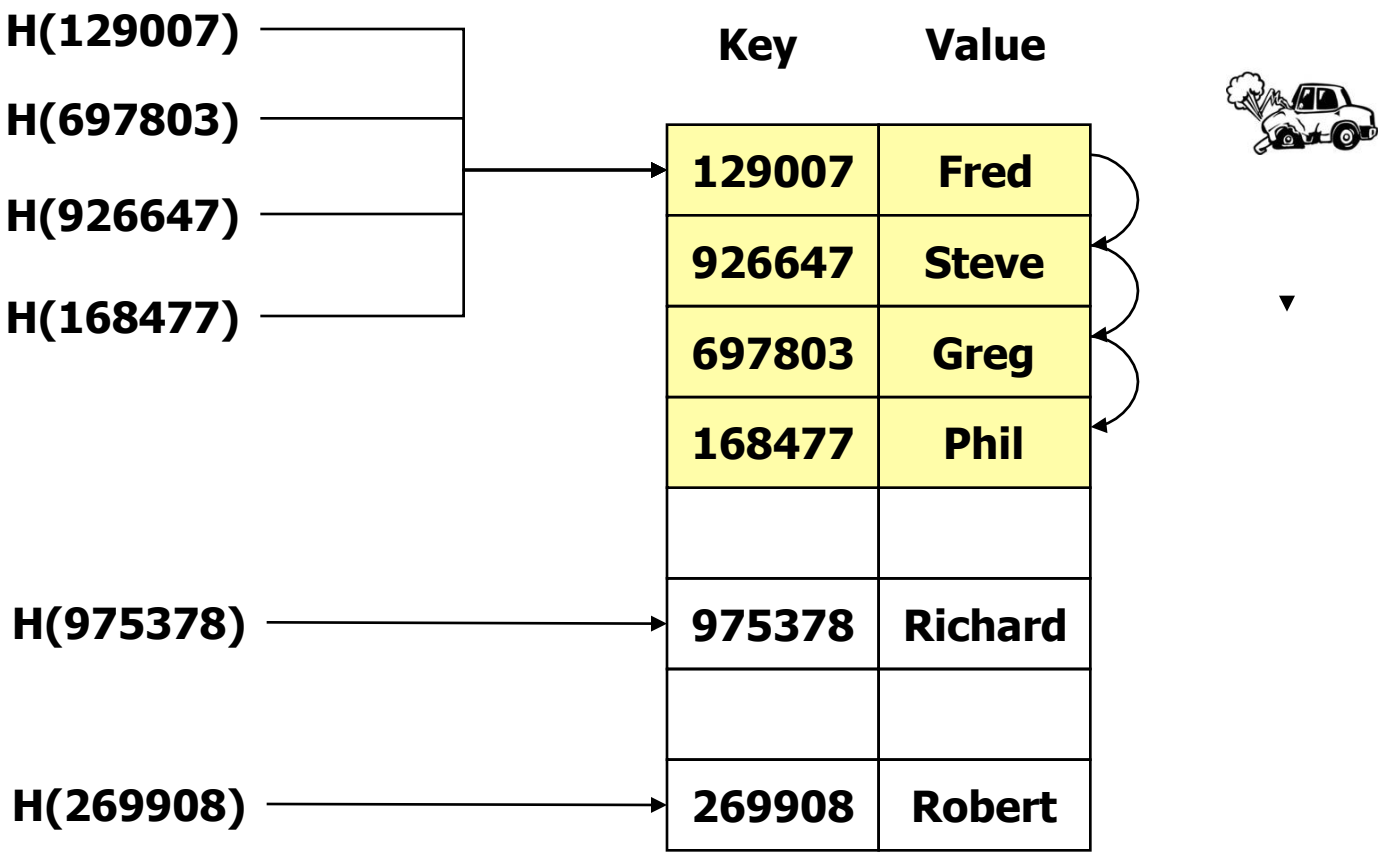
Collision handling strategies

- Closed addressing (open hashing).
- Open addressing (closed hashing).



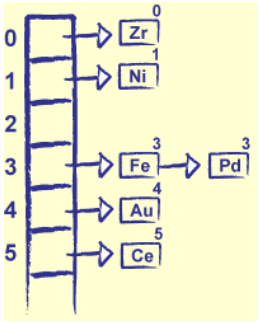
Open addressing (closed hashing).

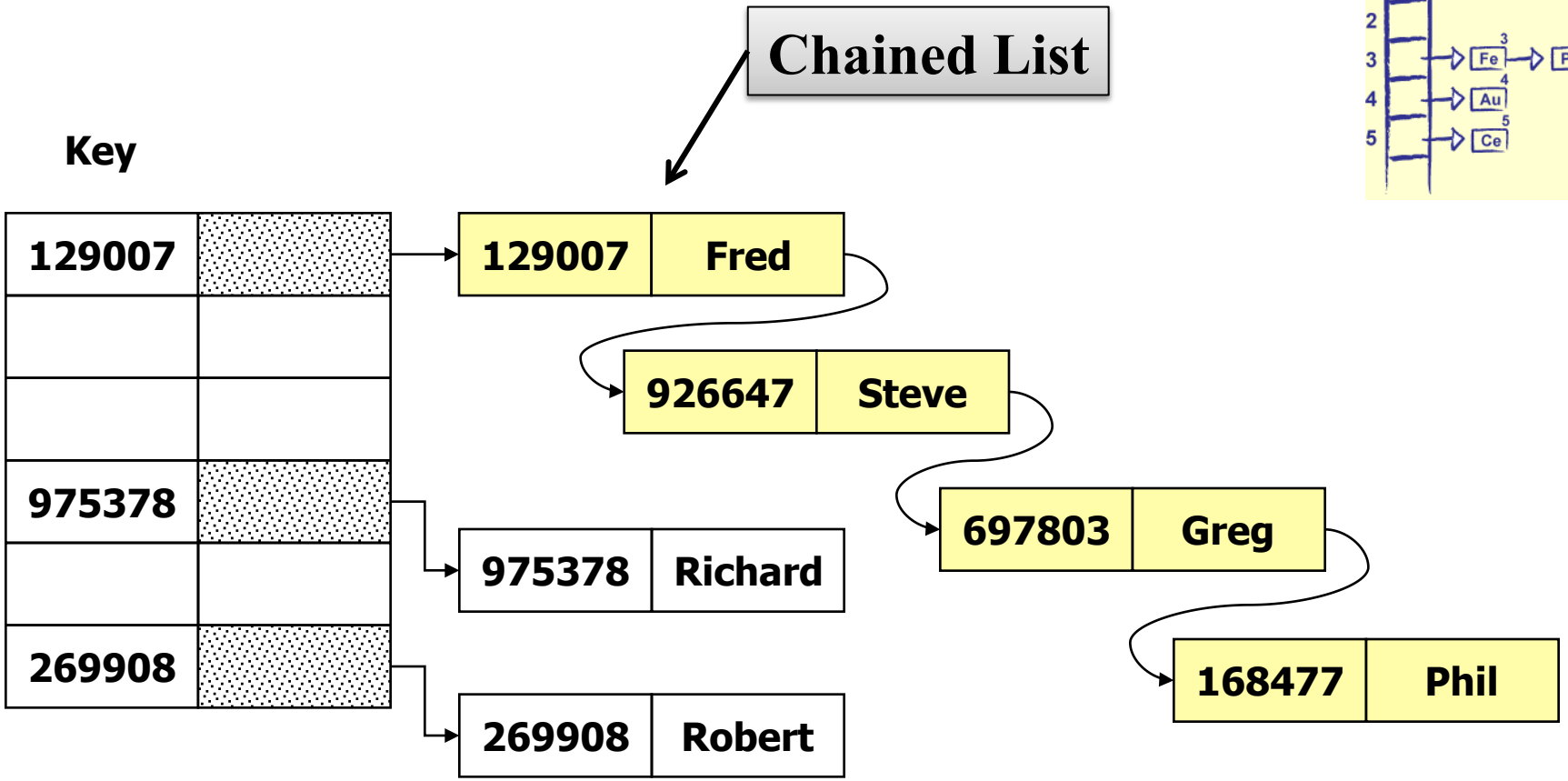
- When there is a collision, "Probe" the array to find an empty slot after the occupied slot.



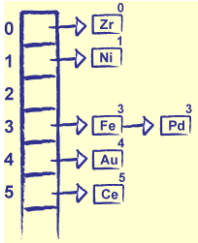
Closed addressing (open hashing).

- Each slot of the hash table contains a link to another data structure.



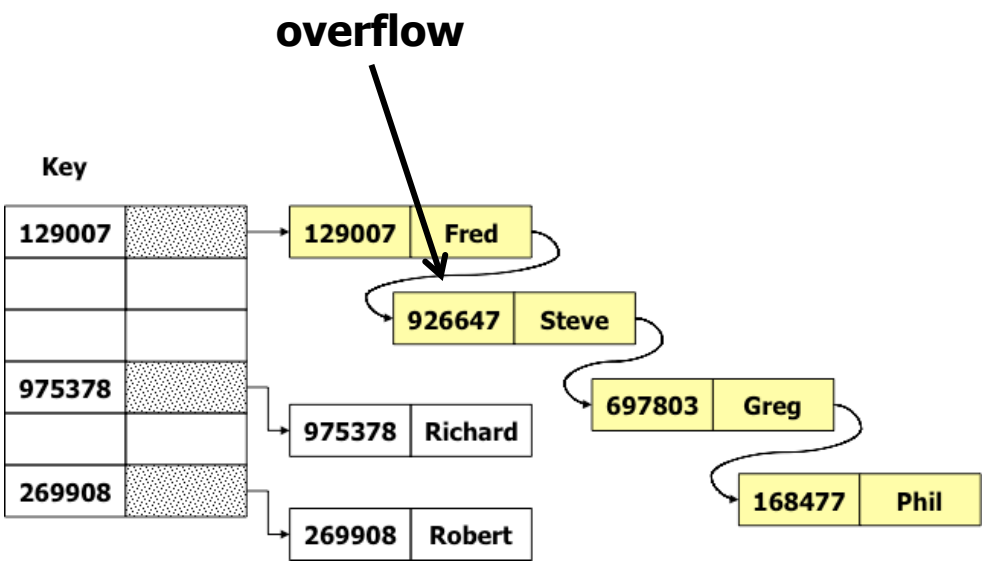


Closed addressing (open hashing).



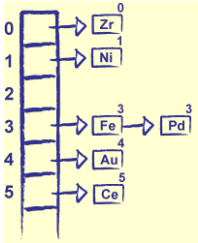
- Each slot of the hash table contains a link to another data structure.

Overflow Table

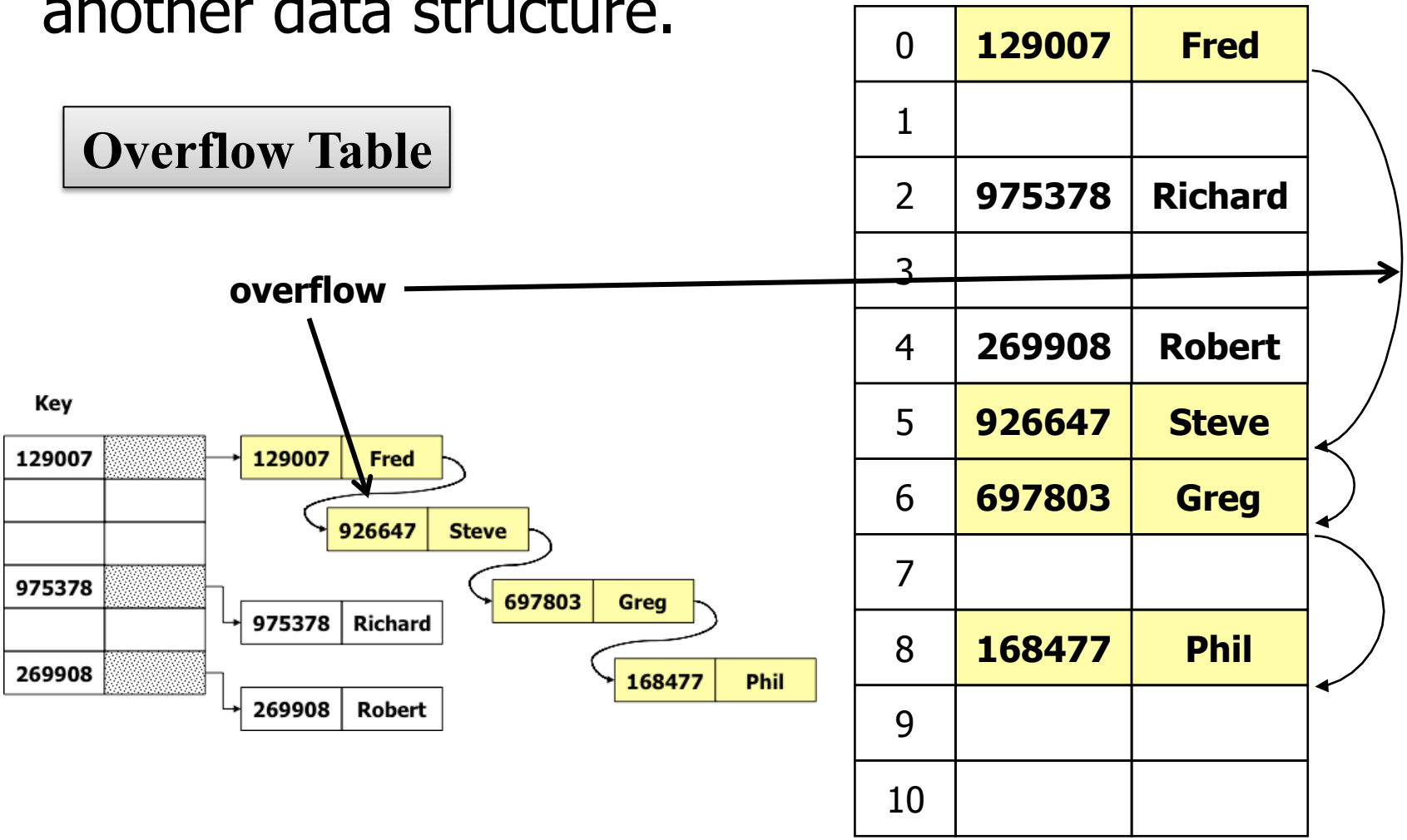


0	**129007**	**Fred**
1		
2	**975378**	**Richard**
3		
4	**269908**	**Robert**
5	**926647**	**Steve**
6	**697803**	**Greg**
7		
8	**168477**	**Phil**
9		
10		

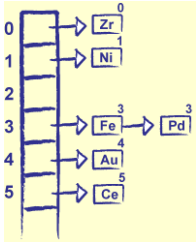
Closed addressing (open hashing).



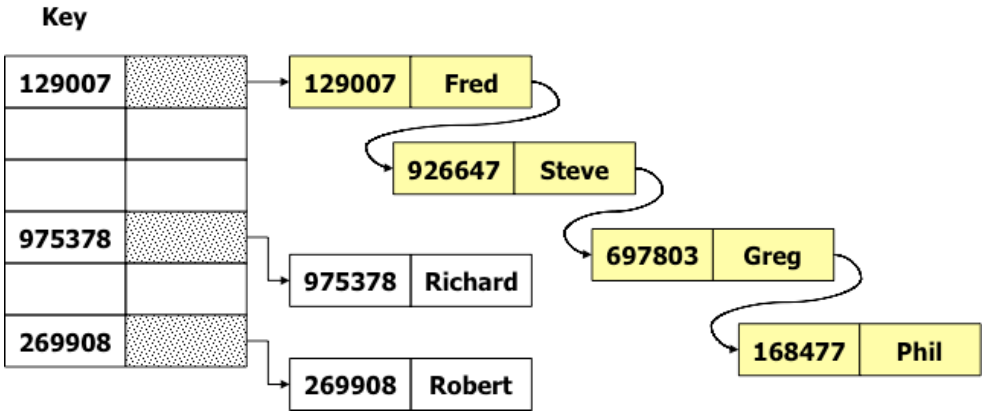
- Each slot of the hash table contains a link to another data structure.



Closed addressing (open hashing).

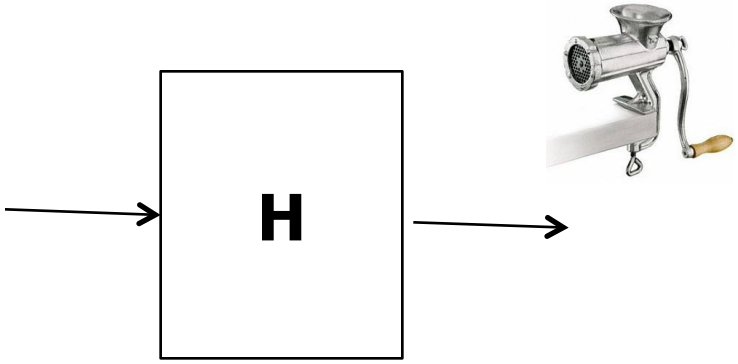


- Each slot of the hash table contains a link to another data structure.

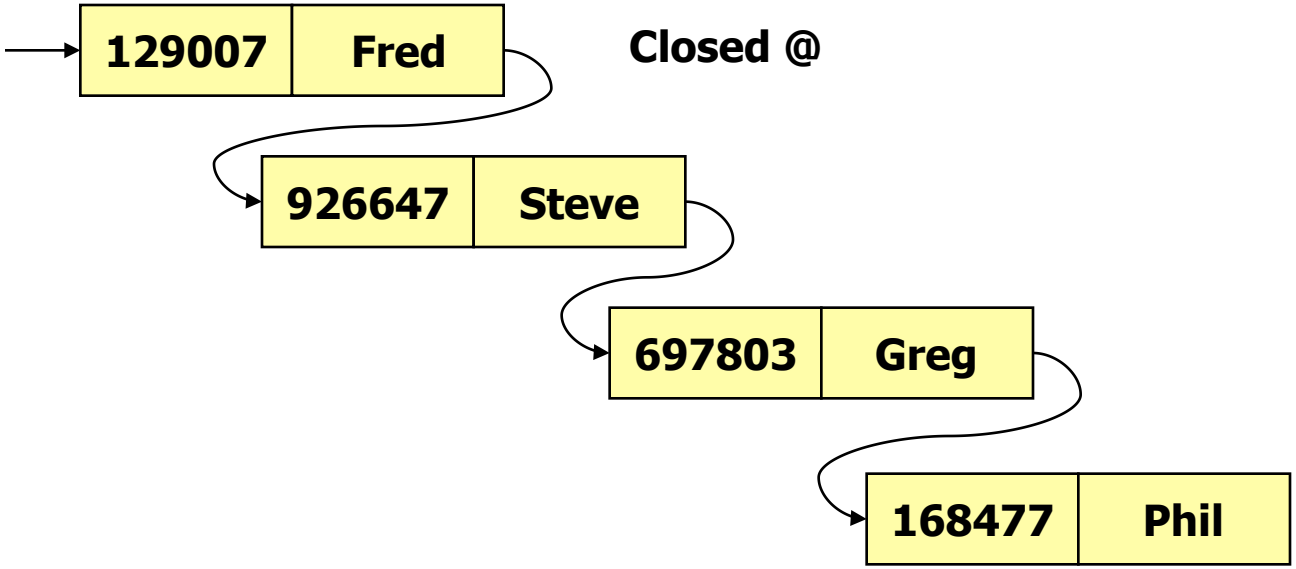


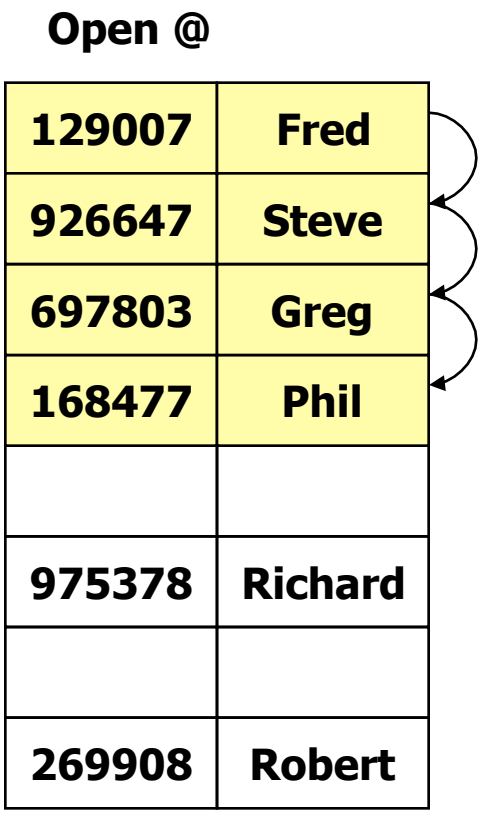
0	**129007**	**Fred**	5
1			
2	**975378**	**Richard**	
3			
4	**269908**	**Robert**	
5	**926647**	**Steve**	6
6	**697803**	**Greg**	8
7			7
8	**168477**	**Phil**	8
9			9
10			10

Hash Table components



- 1) Hash function
- 1) Data Structure
  - 1) Tables + Tables
  - 2) Tables + Chained Lists
  - 3) Chained Lists





Hash Phases

- Fill the hash table : Build phase
- Get values : "Probe" phase
  - Probe term is used in several way.