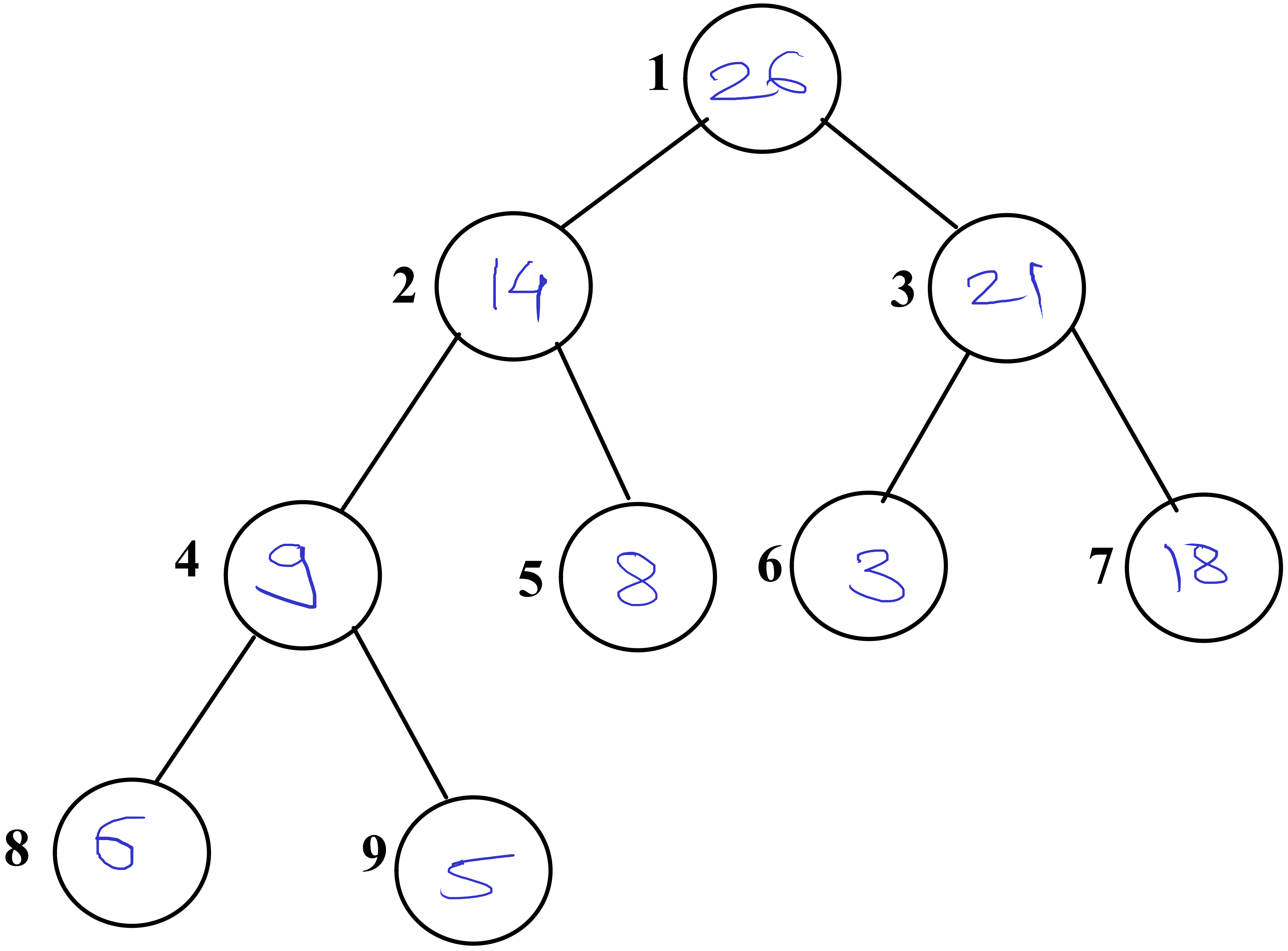
Heap sort - Create Heap

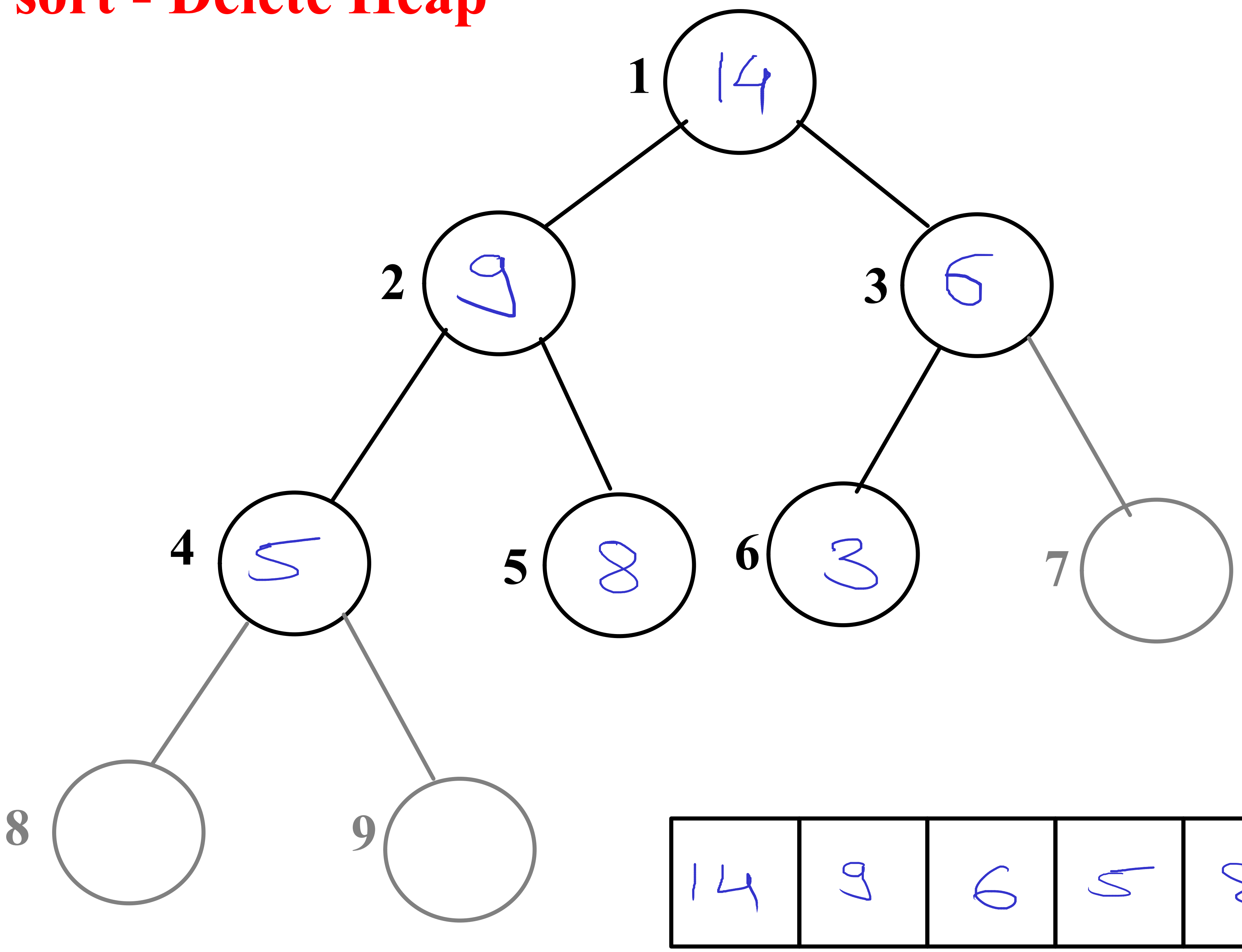
6	14	3	26	8	18	21	9	5



26	14	21	9	8	3	18	6	5
1	2	3	4	5	6	7	8	9

Heap sort - Delete Heap

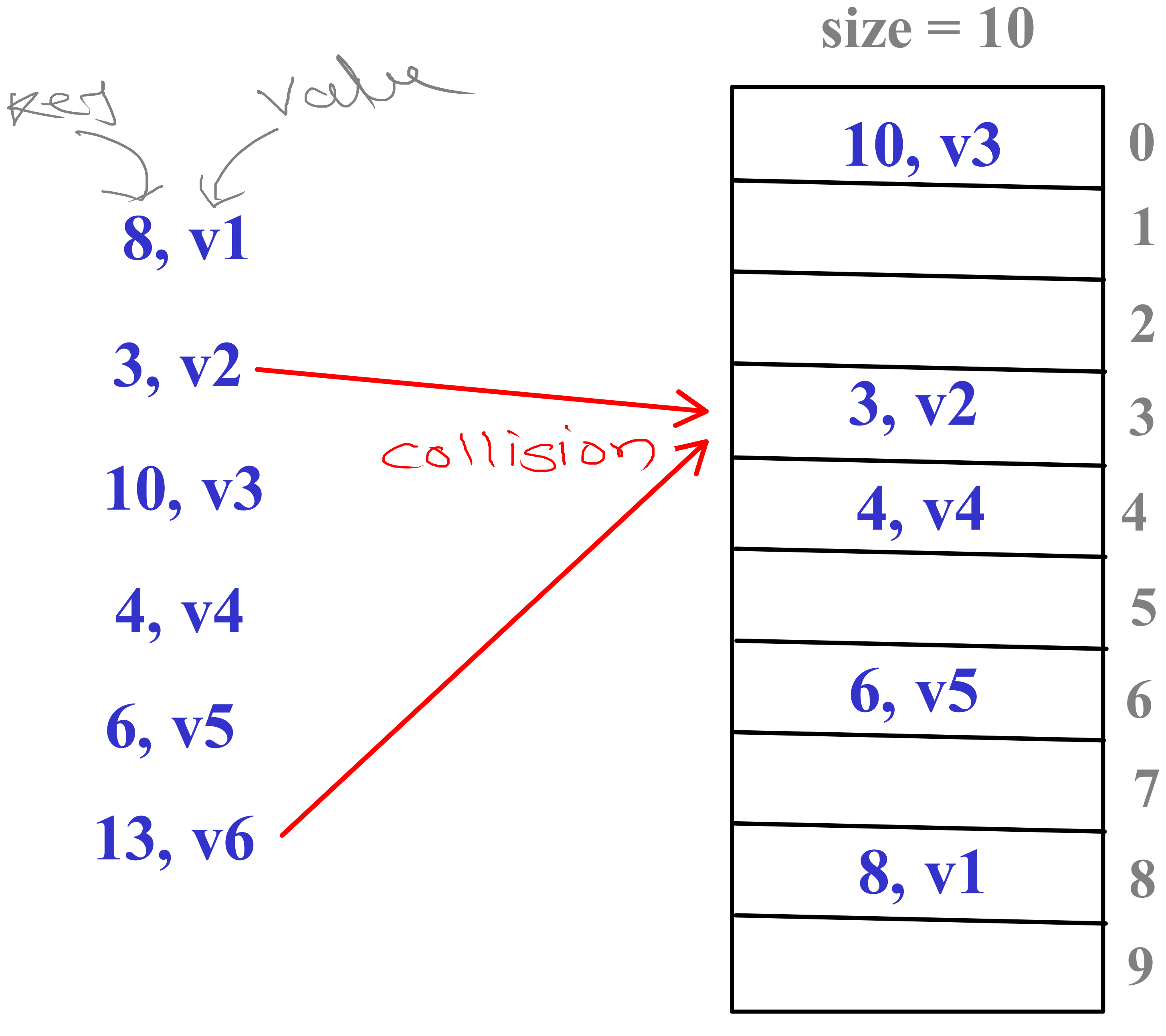
Max =



14	9	6	5	8	3	18	21	26
1	2	3	4	5	6	7	8	9

3	5	6	8	9	14	18	21	26
1	2	3	4	5	6	7	8	9

Hashing



size = 10

8, v1

3, v2

10, v3

4, v4

6, v5

13, v6

10, v3	0
	1
	2
3, v2	3
4, v4	4
	5
6, v5	6
	7
8, v1	8
	9

Hash Table

Collision:

-if any key yield slot which is already occupied then collision has occured

$h(k) = k \% size$

$h(8) = 8\%10 = 8$

$h(3) = 3\%10 = 3$

$h(10) = 10\%10 = 0$

$h(4) = 4\%10 = 4$

$h(6) = 6\%10 = 6$

$h(13) = 13\%10 = 3$

add: — O(1)

```
slot=k%size;
arr[slot]=data;
```

search: — O(1)

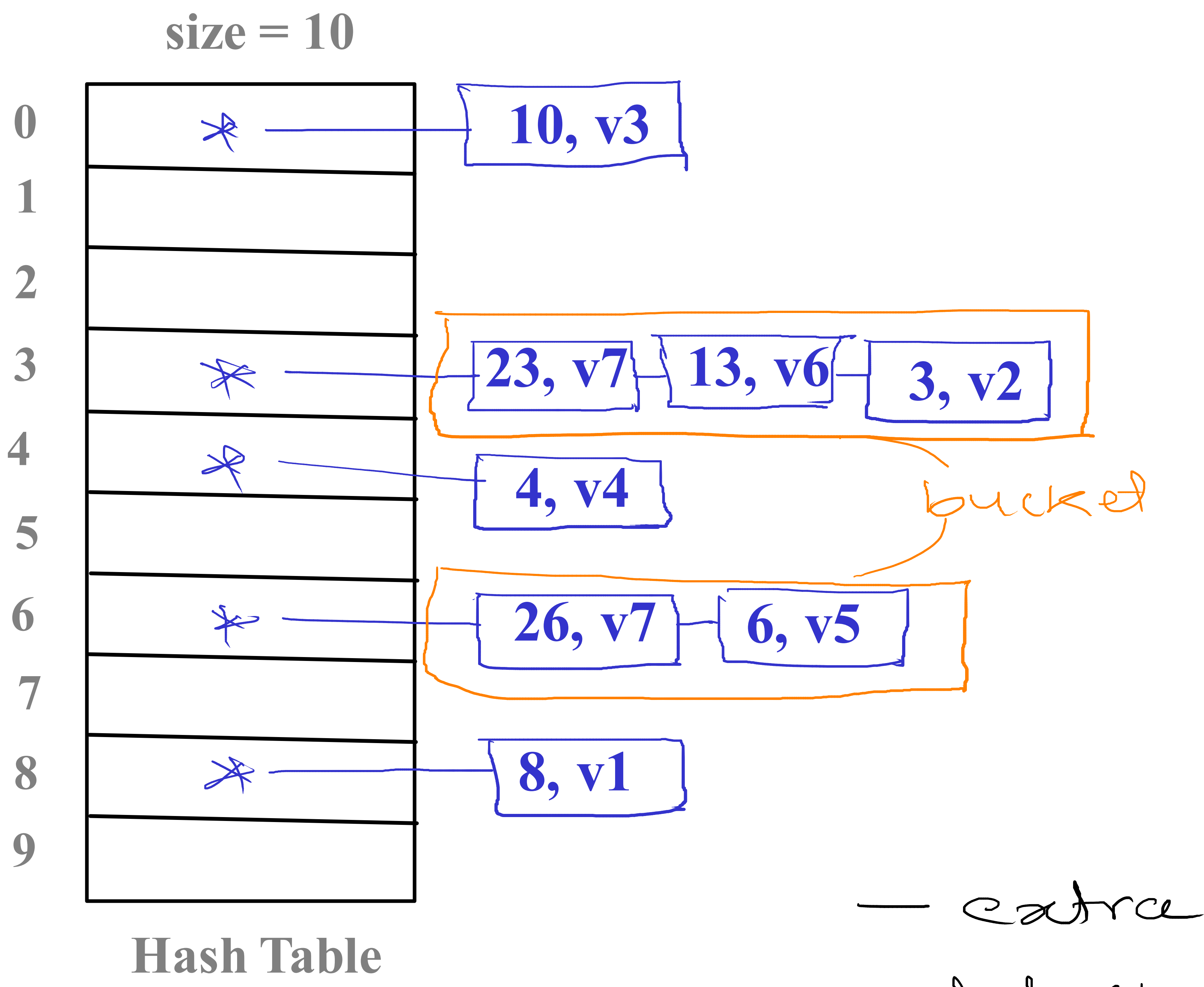
```
slot = k % size;
return arr[slot].value;
```

delete: — O(1)

```
slot= k%size;
arr[slot]=null;
```

Closed Addressing/ Seperate Chaining / Chaining

8, v1
3, v2
10, v3
4, v4
6, v5
13, v6
23, v7
26, v7

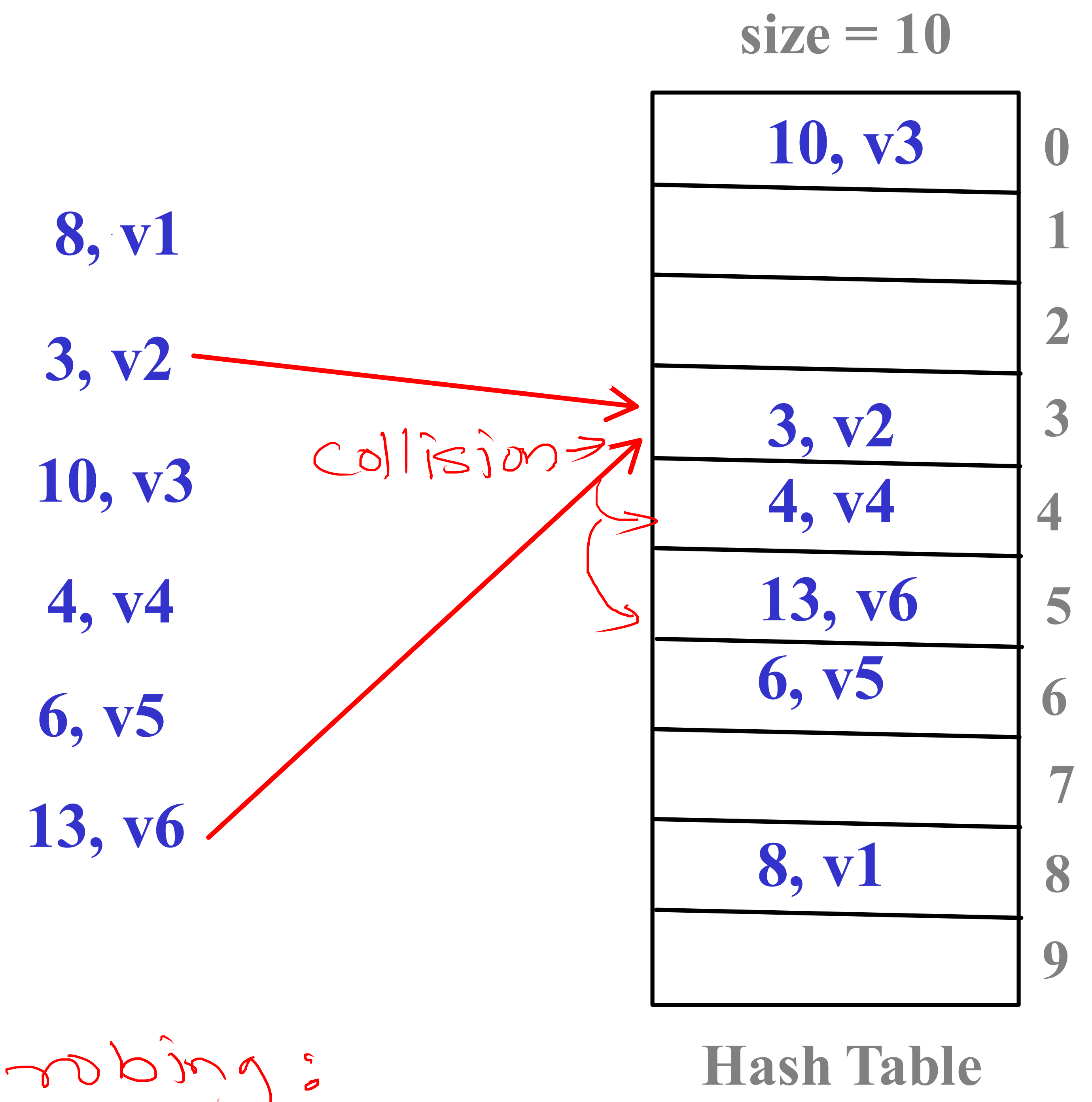


h(k) = k % size

h(8) = 8 % 10 = 8
h(3) = 3 % 10 = 3
h(10) = 10 % 10 = 0
h(4) = 4 % 10 = 4
h(6) = 6 % 10 = 6
h(13) = 13 % 10 = 3
h(23) = 23 % 10 = 3
h(26) = 26 % 10 = 6

- extra space required
- data (key, value) is kept/ stored outside the table
- worst case : O(n)
  ↳ if multiple keys yield same slot

Open Addressing - Linear Probing



8, v1

3, v2

10, v3

4, v4

6, v5

13, v6

0	10, v3
1	
2	
3	3, v2
4	4, v4
5	13, v6
6	6, v5
7	
8	8, v1
9	

$h(k) = key \ \% \ size$

$h(k, i) = [\ \underline{h(k)} + \underline{f(i)}\ ] \ \% \ size$

$f(i) = \underline{i}$

where i = 1, 2, 3, ...

↳ probe number

$h(8) = 8 \ \% \ 10 = 8$

$h(3) = 3 \ \% \ 10 = 3$

$h(10) = 10 \ \% \ 10 = 0$

$h(4) = 4 \ \% \ 10 = 4$

$h(6) = 6 \ \% \ 10 = 6$

$h(13) = 13 \ \% \ 10 = 3$ (c)

$h(13, 1) = [3 + 1] \ \% \ 10$

$= 4$ (1^{st} probe) (c)

$h(13, 2) = [3 + 2] \ \% \ 10$

$= 5$ (2^{nd} probe)

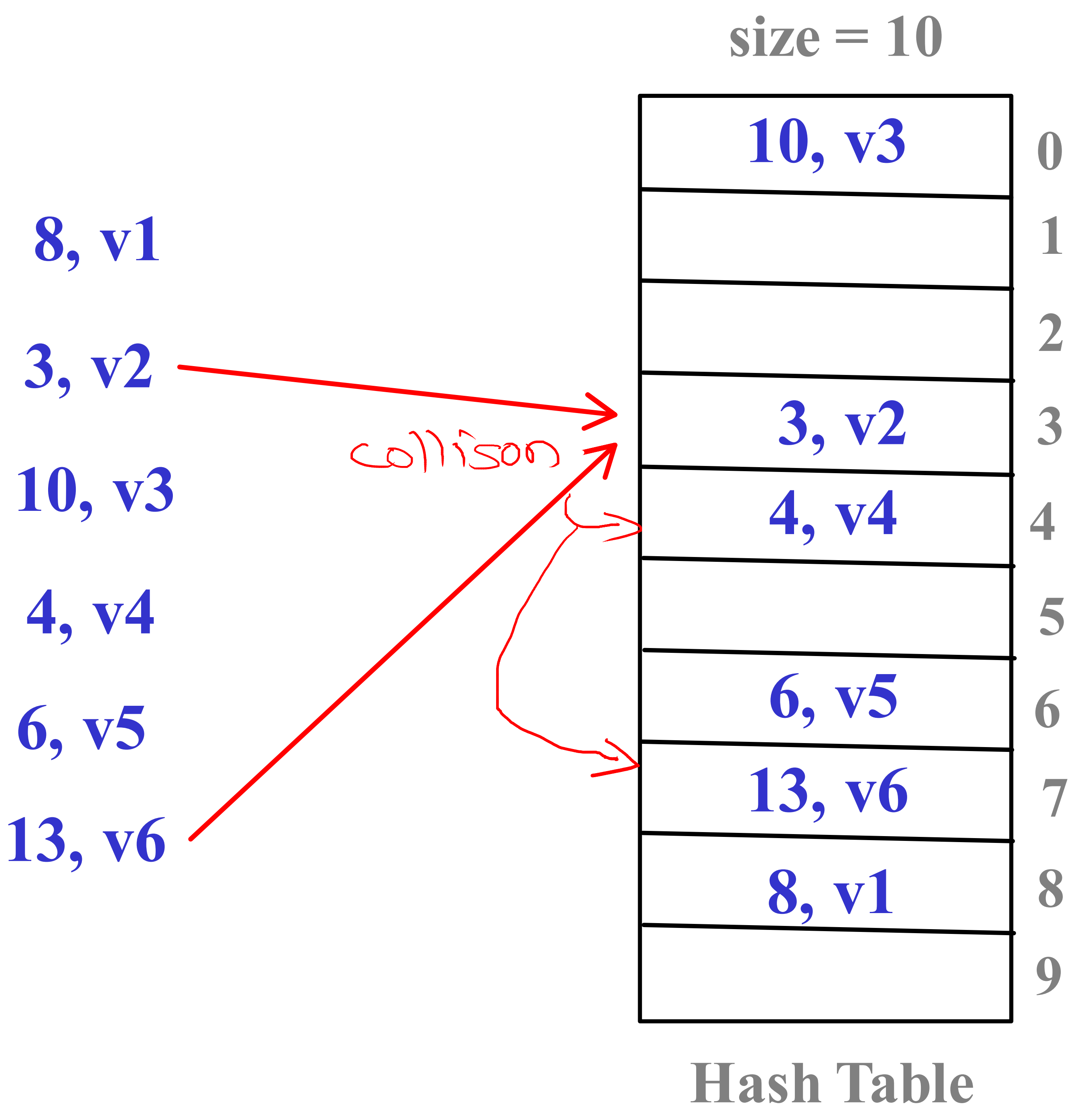
probing :

- finding new slot for key when collision occur

Primary clustering :

- to find next free slot needs long runs of filled slots "near" key position

Open Addressing - Quadratic Probing



10, v3	0
	1
	2
3, v2	3
4, v4	4
	5
6, v5	6
13, v6	7
8, v1	8
	9

h(k) = key % size

h(k, i) = [h(k) + f(i)] % size

f(i) = i^2

where i = 1, 2, 3, ...

$h(8) = 8 \% 10 = 8$

$h(3) = 3 \% 10 = 3$

$h(10) = 10 \% 10 = 0$

$h(4) = 4 \% 10 = 4$

$h(6) = 6 \% 10 = 6$

$h(13) = 13 \% 10 = 3$ (c)

$h(13,1) = [3+1] \% 10$

$= 4$ (1st) (c)

$h(13,2) = [3+4] \% 10$

$= 7$ (2nd)

Open Addressing - Quadratic Probing

23, v7

33, v8

size = 10

10, v3	0
	1
23, v7	2
3, v2	3
4, v4	4
	5
6, v5	6
13, v6	7
8, v1	8
33, v8	9

Hash Table

h(k) = key % size

h(k, i) = [h(k) + f(i)] % size

f(i) = i^2

where i = 1, 2, 3, ...

$h(23) = 23 \% 10 = 3$ (c)

$h(23,1) = [3+1] \% 10 = 4$ (1st) (c)

$h(23,2) = [3+4] \% 10 = 7$ (2nd) (c)

$h(23,3) = [3+9] \% 10 = 2$ (3rd)

$h(33) = 33 \% 10 = 3$ (c)

$h(33,1) = [3+1] \% 10 = 4$ (1st) (c)

$h(33,2) = [3+4] \% 10 = 7$ (2nd) (c)

$h(33,3) = [3+9] \% 10 = 2$ (3rd) (c)

$h(33,4) = [3+16] \% 10 = 9$ (4th)

- there is no garantee of getting free slot

Secondary clustering:

- to find next free slot needs long runs of filled slots "away" key position

Hashing - Double Hashing

8, v1

3, v2

10, v3

25, v6

size = 11

	0
	1
	2
3, v2	3
	4
	5
25, v6	6
	7
8, v1	8
	9
10, v3	10

Hash Table

$h1(k) = key \% size$

$h2(k) = 7 - (key \% 7)$

$h(k, i) = [h1(k) + i * h2(k)] \% size$

$h1(8) = 8 \% 11 = 8$

$h_1(3) = 3 \% 11 = 3$

$h_1(10) = 10 \% 11 = 10$

$h_1(25) = 25 \% 11 = 3$ (c)

$h_2(25) = 7 - 25 \% 7 = 3$

$h(23,1) = [3 + 1 * 3] \% 11$

$= 6$ (1st probe)

Rehashing

$\lambda = 0.75$

↳ 75% filled

$$\text{Load Factor} = \frac{n}{N}$$

(λ)

n - Number of elements (key value pairs) in hash table
N - Number of slots in hash table

if $n < N$	Load factor < 1	- free slots are available
if $n = N$	Load factor $= 1$	- no free slots
if $n > N$	Load factor > 1	- can not insert at all

- Rehashing is make the hash table size twice of existing size if hash table is 70 or 75 % full

- In rehashing existing key value pairs are again mapped according to new hash table size