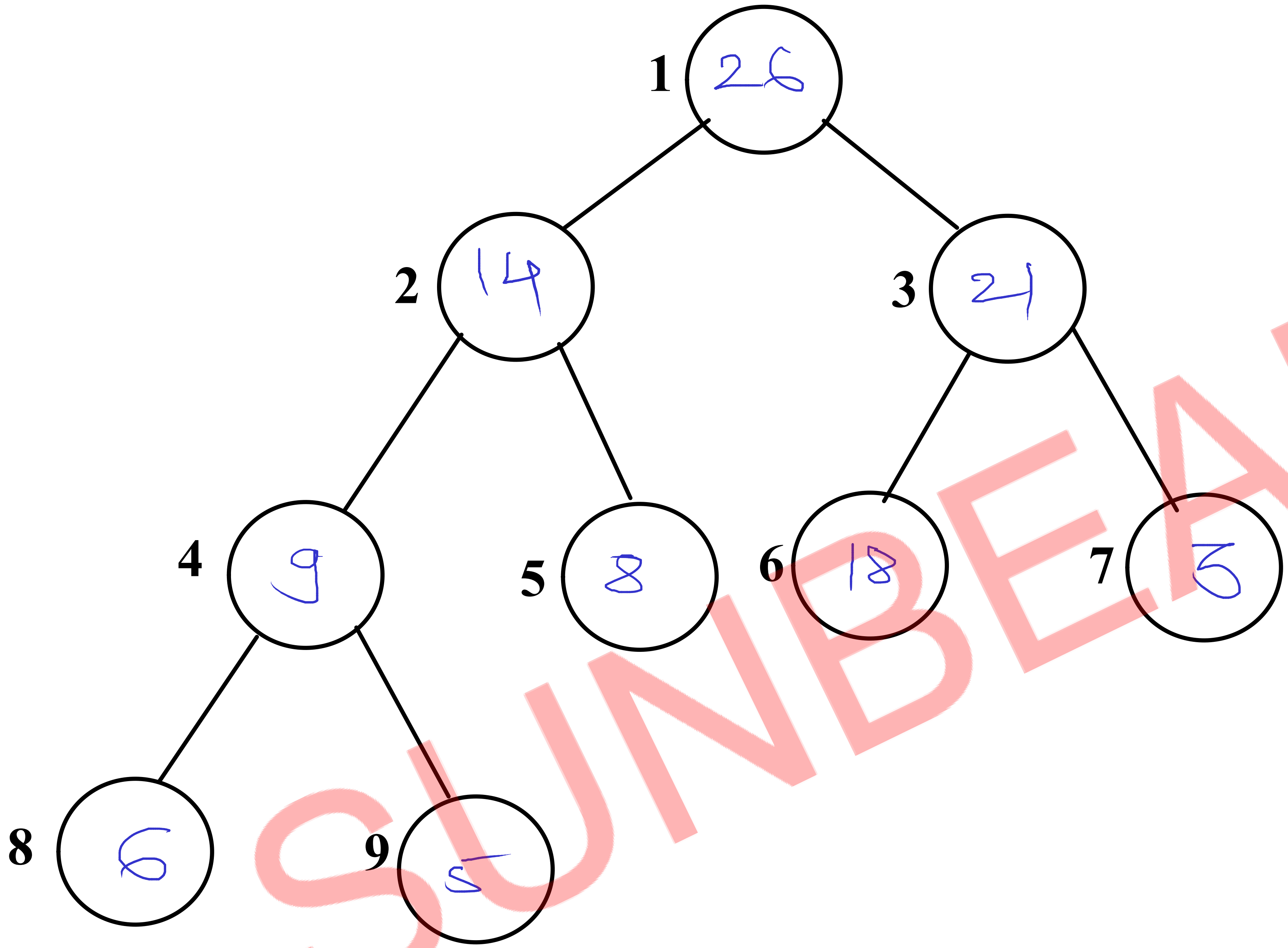
Heapify



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

$9/2 = \boxed{4}$

$T(n) = O(n)$

```
for(i=size/2;
    i>=1;
    i--)
```

SUNBEAM

linear Search :- $T(n) = O(n)$

Binary search :- $T(n) = O(\log n)$

Linked List search :- $T(n) = O(n)$

Binary Tree search :- $T(n) = O(n)$

BSTree search :- $T(n) = O(\log n)$

Hash Table :- $T(n) = O(1)$

SUNBEAM

Hashing

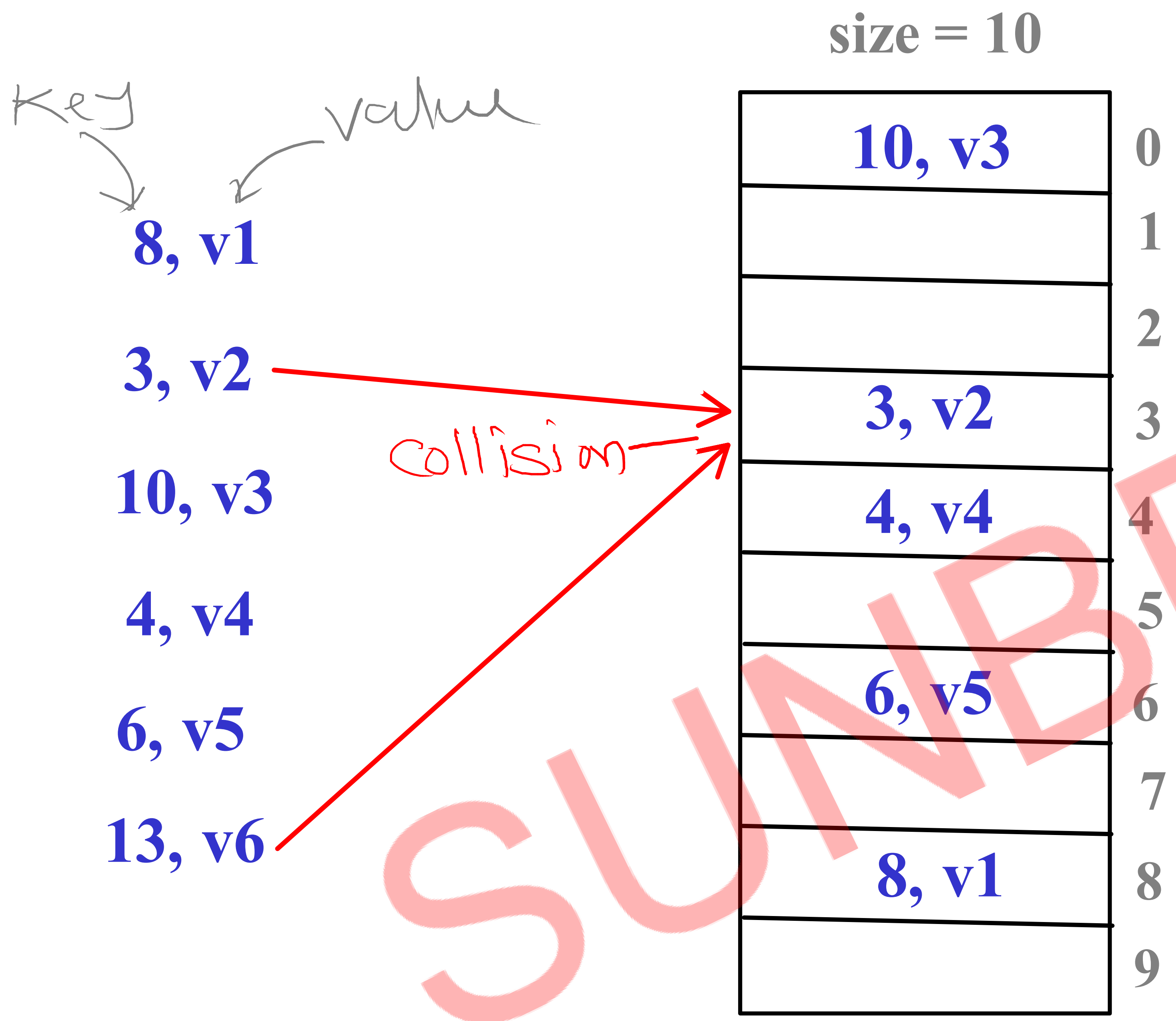
- hashing is a technique in which data can be inserted, removed and searched in constant avearge time (O(1))
- implementation of this technique is known hash table
- hash table is nothing but fixed size array in which elements are stored in key-value pair

  Array - hash table
  index - slot
- keys are always unique but values can be duplicates
- every key is mapped with one slot of the hash table.
- this mapping is done by a mathematical function known as "hash function"

SUNBEAM

Hashing

h(k) = k % size



Hash Table

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

$h(3) = 8 \% 10 = 3$

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

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

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

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

Add: — O(1)
1) find slot
2) arr[slot] = (key, value)

Search: — O(1)
1) find slot
2) return arr[slot](value)

Delete: — O(1)
1) find slot
2) arr[slot] = null;

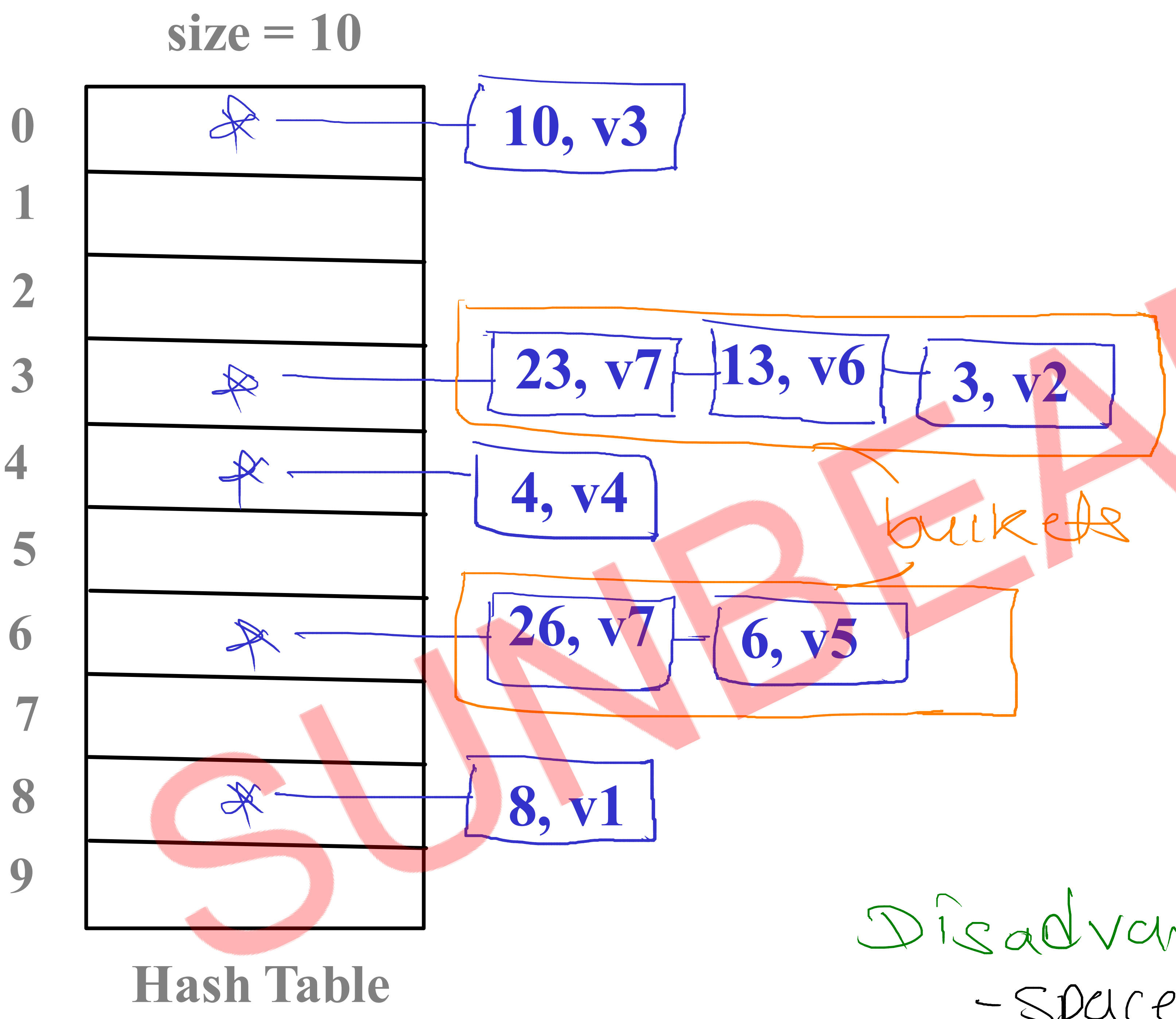
Collision:
- if multiple distinct keys yeild same slot

Collision handling techniques:
1) Closed Addressing
2) Open Addressing

SUNBEAM

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

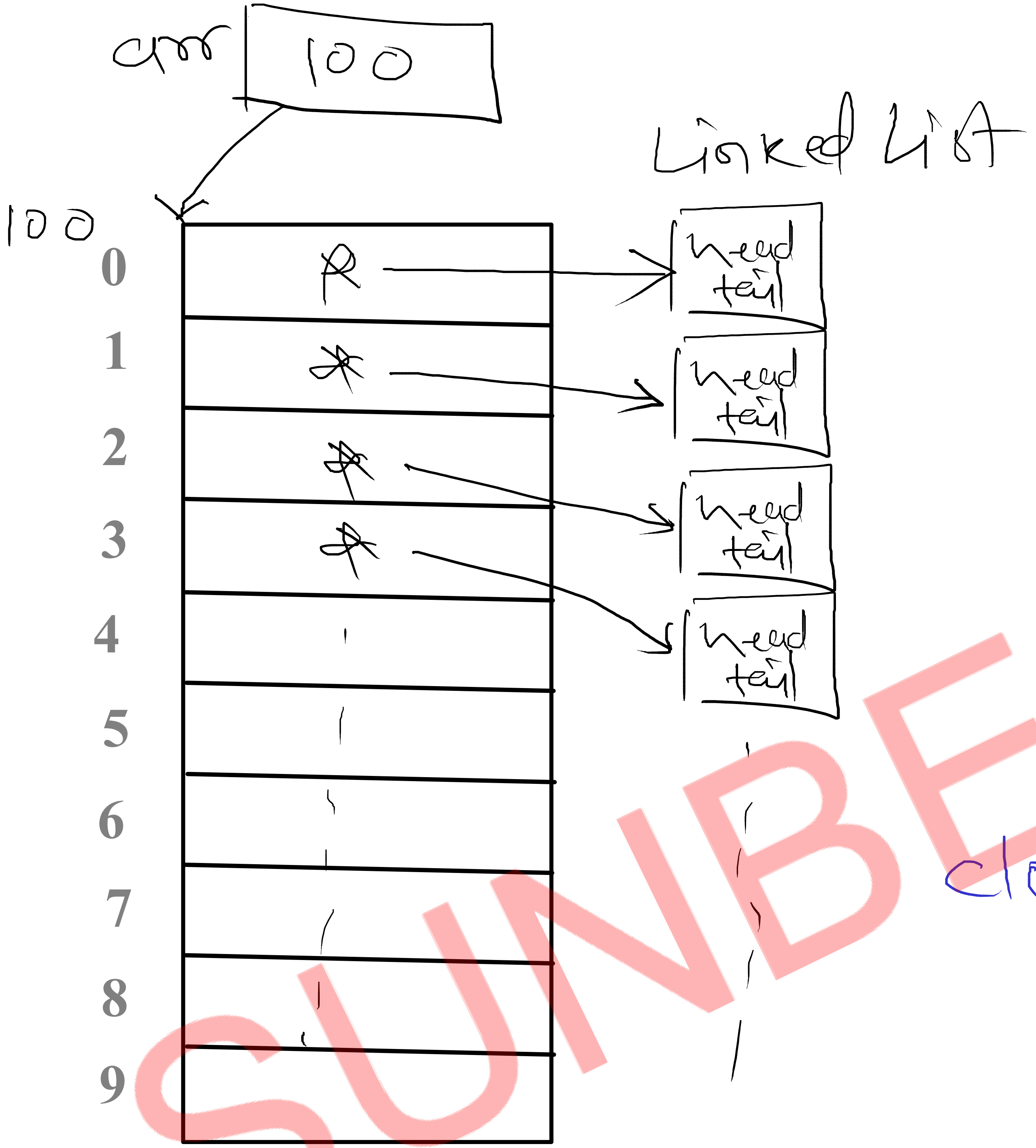
Advantage:

- multiple key value pairs can be stored

Disadvantage:

- space inside memory is more
- data(key, value) is stored outside the table area(space)
- worst case T(n) = O(n)

SUNBEAM

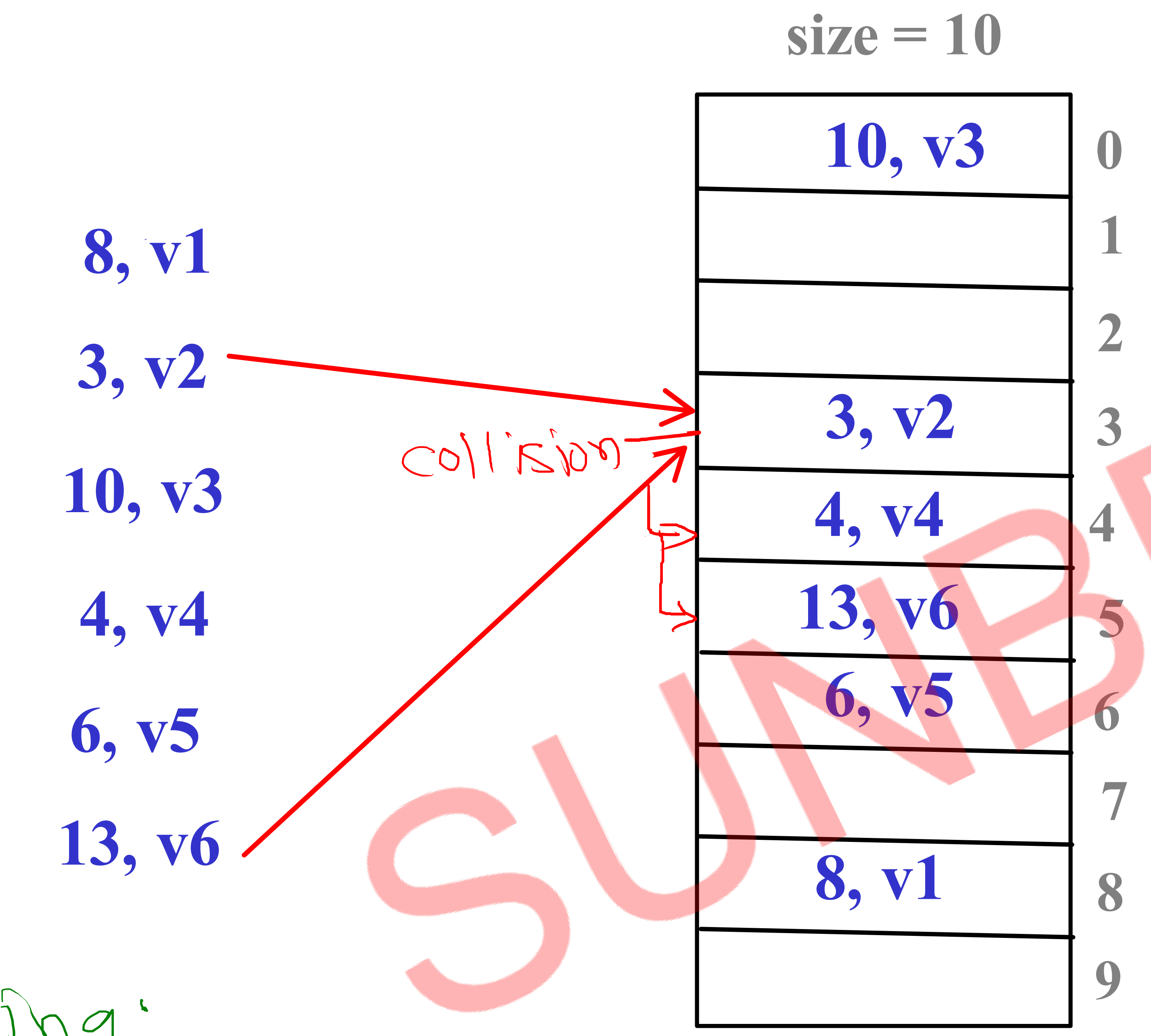


closed Addressing
↳ open probing

open Addressing
↳ closed probing

SUNBEAM

Open Addressing - Linear Probing



Hash Table

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

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

$f(i) = i$

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

↳ probe number

$h(13) = 13 \% 10 = 3$ ⓒ

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

$= [3 + 1] \% 10$

$= 4$ (1st probe) ⓒ

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

$= 5$ (2nd probe)

Probing:

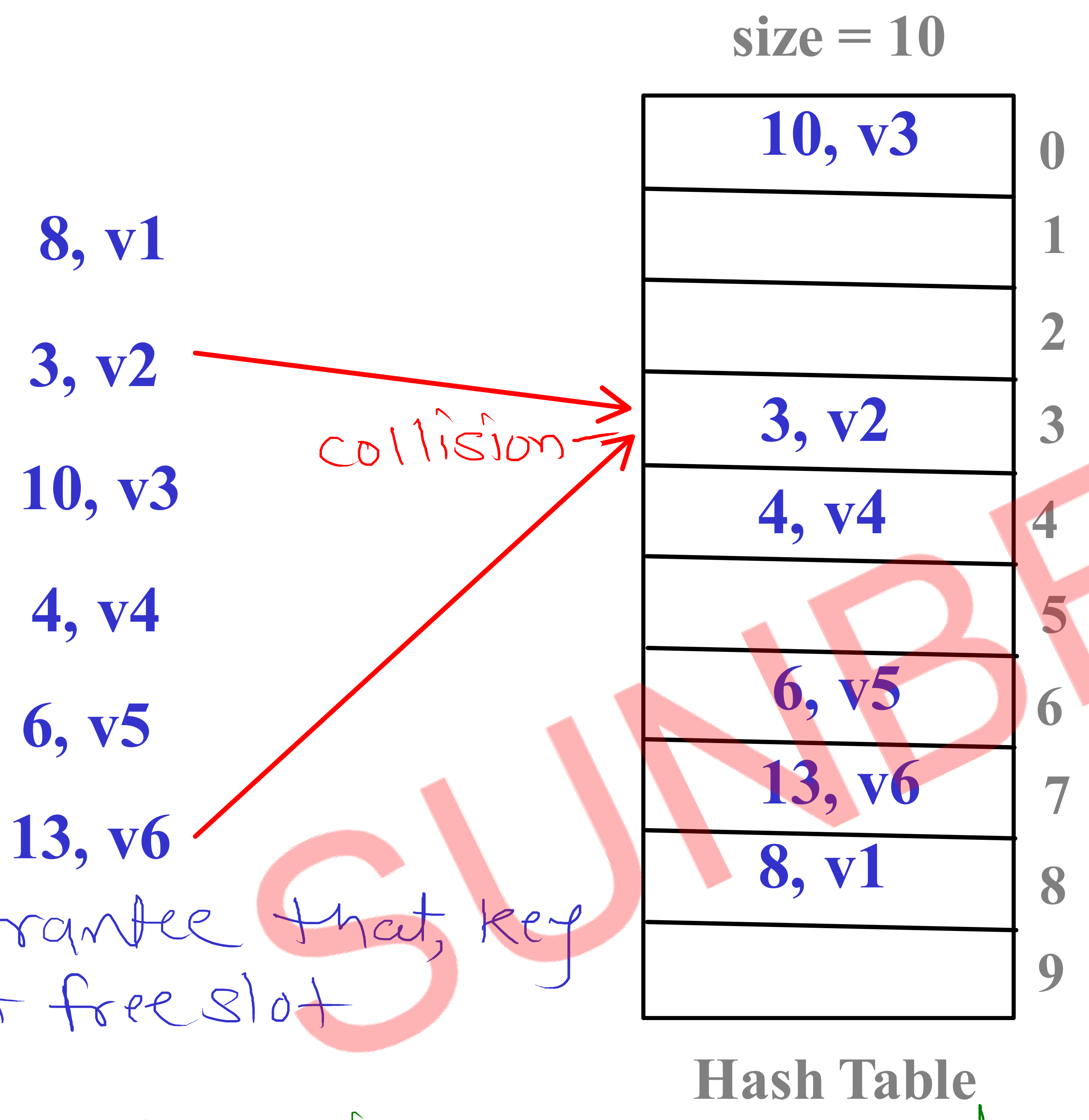
- finding next free slot whenever collision occurs

Primary clustering:

- need long runs of filled slots "near" key position to find empty slot

SUNBEAM

Open Addressing - Quadratic Probing



h(k) = key % size

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

f(i) = i^2

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

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

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

$= [3+1] \% 10$

$= 4$ (1st probe) c

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

$= 7$ (2nd probe)

- no garantee that, key will get free slot

- primary clustering is removed

Secondary clustering

- need long runs of filled slots "away" key position to find empty slot

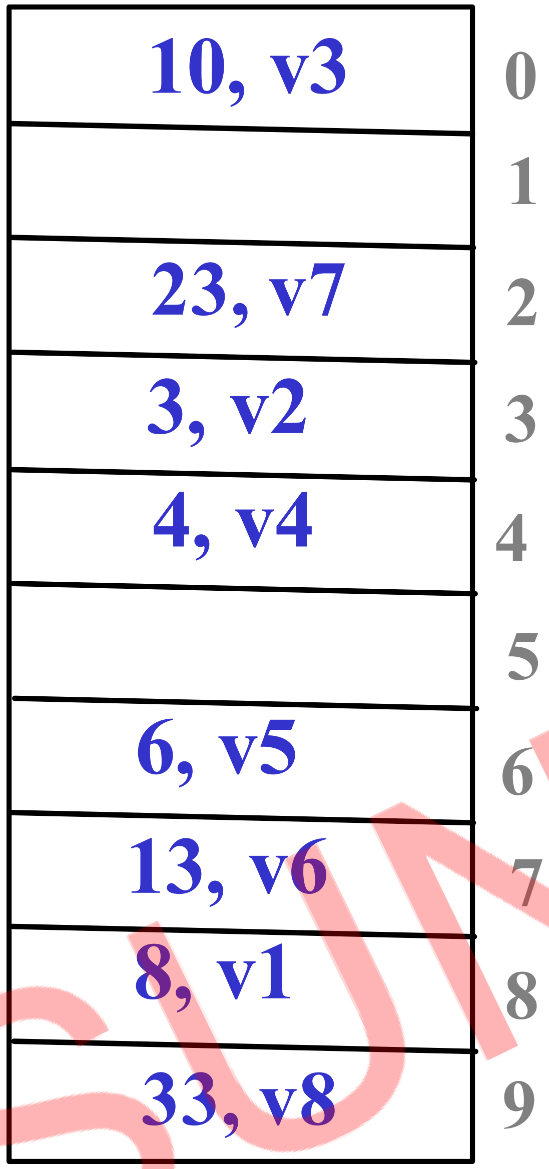
SUNBEAM

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) = 23 % 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

SUNBEAM

Hashing - Double Hashing

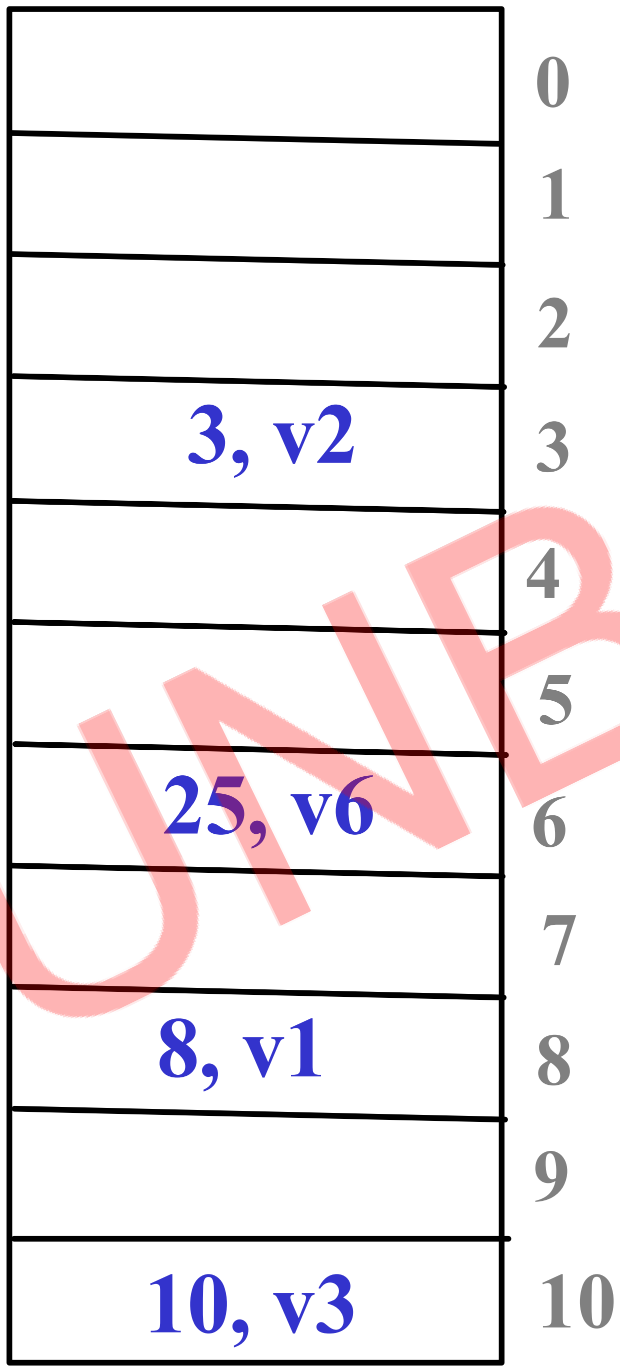
8, v1

3, v2

10, v3

25, v6

size = 11



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

Hash Table

$h1(k) = key \% size$

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

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

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

$h1(3) = 3 \% 11 = 3$

$h1(10) = 10 \% 11 = 10$

$h1(25) = 25 \% 11 = 3$ ⓒ

$h2(25) = 7 - (25 \% 7) = 3$

$h(25, 1) = [3 + 1 * 3] \% 11$

$= 6 \% 11 = 6$ (1st)

SUNBEAM

Rehashing

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

(λ)

$$\frac{6}{10} = \underline{\underline{0.6}} \rightarrow 60\% \text{ full}$$

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

SUNBEAM