DSA - Seminar 6

<u>Problems</u>

Representations on a **hash table**

1. collision resolution with separate chaining
2. collision resolution with coalesced chaining

see lectures

•

discussions

Problem 1:

Iterator for a SortedMap represented on a hash table, collision resolution with separate chaining.

- Assume on our example
  - We memorize only the keys from the Map
  - Keys are integer numbers

EX:

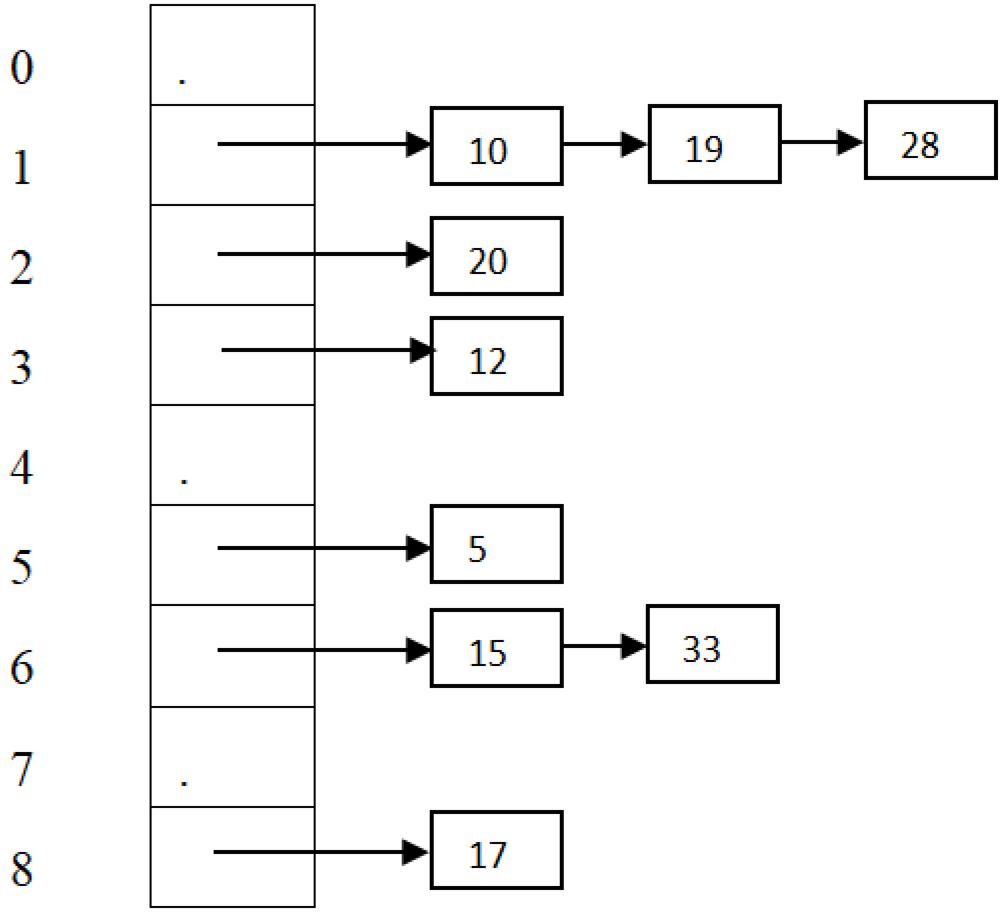
- Keys from the map: 5, 28, 19, 15, 20, 33, 12, 17, 10
  Keys have to be unique!
- HT
  - m = 9
  - Hash function defined with the division method
    - h(k) = k mod m

Problem 1.1: Representation: SortedMap on a hash table

[…] collision resolution with separate chaining.

- 5, 28, 19, 15, 20, 33, 12, 17, 10
- HT: m = 9 ; h(k) = k mod m

k	5	28	19	15	20	33	12	17	10
h(k)	5	1	1	6	2	6	3	8	1

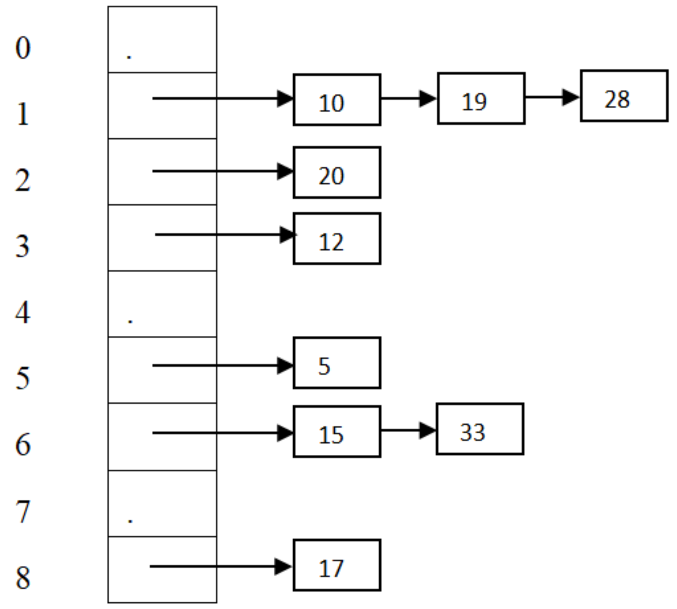


Representation?

Problem 1.2: Iterator for a SortedMap represented on a hash table

[…] collision resolution with separate chaining.

- 5, 28, 19, 15, 20, 33, 12, 17, 10



Representation:

TNode:
e: TElem // key, value
next: ↑TNode

SortedMap:
m: Integer
T : (↑TNode)[]
h: TFunction
R: relation

Structure not optimized for iteration

IteratorSortedMap:
sm: SortedMap
l: TList
currentNode: ↑TNode

Complexity for mergeList ?

```
subalgorithm init(it, sm):
  it.sm ← sm
  mergeLists (sm, it.l)
  it.currentNode ← it.l.head
end-subalgoritm
```

DSA - Seminar 6

Problem 2:

Map

– representation on a **hash table**

– collision resolution with coalesced chaining

Implement operations: init, search, remove.

Map on a **hash table** collision resolution with coalesced chaining

Representation:

<u>Map:</u>
m: Integer
t: TKey[]
next: Integer[]
firstFree: Integer
h: Tfunction

Ex: 5, 18, 16, 15, 13, 31, 26

- HT:
  - m = 13
  - firstFree is considered to be the first empty position from left to right (empty positions are no longer linked)

K	5	18	16	15	13	31	26
h(k)	5	5	3	2	0	5	0

	0	1	2	3	4	5	6	7	8	9	10	11	12
t	18	13	15	16	31	5	26						
next	1	4	-1	-1	6	0	-1	-1	-1	-1	-1	-1	-1

Map on a **hash table** collision resolution with coalesced chaining

```
Subalgorithm init (map):
        @ initialize the hash function
        @ initialize the value of m
        …
end-subalgorithm
```

Representation:

<u>Map:</u>
m: Integer
t: Tkey []
next: Integer []
firstFree: Integer
h: Tfunction

```
Function search(map, k):
        // implement a simple version:
        // return the position where the key was found, or -1
        // in case of a real map, you return the value associated to the key
        …
end-function
```

5, 18, 16, 15, 13, 31, 26

	0	1	2	3	4	5	6	7	8	9	10	11	12
t	18	13	15	16	31	5	26						
next	1	4	-1	-1	6	0	-1	-1	-1	-1	-1	-1	-1