Seminar 06

Thursday, May 19, 2022 2:52 PM

Hashtables

Coalesced chaining

Implementation: Map - we will use only the keys (natural numbers).

Table ▭ T : TElem

next ▭ integer

firstFree integer

hash function (mod m (capacity)) TFunction

	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

firstFree = ~~0~~ ~~1~~ ~~4~~ ~~6~~ 7

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

h(k): 5, 5, 3, 2, 0, 5, 0

k%m

k%13

Map

```
function search(map, k) is:
    pos ← map.h(k)
    while map.T[pos] ≠ k ∧ pos ≠ -1 execute
        pos ← map.next[pos]
    endwhile
    if pos = -1 then
        search ← false
    else
        search ← true
    endif
```

endfunction

1)

0	1	2	3	4
3	11		8	
-1	-1	-1	-1	-1

2)

0	1	2	3	4
~~11~~	56	12	8	
-1	~~0~~	-1	-1	-1

-1

k:	11	56	12	8
h:	1	1	2	3

3)

56

0	1	2	3	4
20	~~11~~	~~56~~		
-1	~~2~~	-1		

-1

k:	20	11	56
h:	0	1	1

4)

0	1	2	3	
~~11~~	56	12	1	
~~3~~	~~0~~	-1	-1	-1

-1 ←

3

k:	11	56	12	1
h:	1	1	2	1

5)

20 ←

0	1	2	3	4
~~11~~	56	~~20~~	13	
~~2~~	0	-1	-1	-1

-1

k:	11	56	20	13
	1	1	0	3

To remove Key k:

- find the position where k is (p);
- search forward for an element that hashes at p;
- if we find an element we move backwards then we remove the element;
- otherwise we remove the element and set the links.

```
subalgorithm remove(map, k) is:
    pos ← map.h(k)
    prev ← -1
                              ≠ k execute:
```

```
while $pos \neq -1 \wedge map.T[pos] \neq k$ execute :
    prev ← pos
    pos ← map.next[pos]
endwhile
if pos = -1 then
    @keydoesnotexist
else
    index ← pos
    while $index \neq -1 \wedge map.h(map.T[index]) != pos$ execute :
        index ← map.next[index]
    endwhile
    if index = -1 then
        map.T[pos] ← NULL_TELEM
        ( if map.firstFree < pos then
            map.firstFree ← pos )
        if $prev \neq -1$ then
            map.next[prev] ← map.next[pos]
            map.next[pos] ← -1
        else
            i ← 0
            while $i < map.m \wedge map.next[i] \neq pos$ execute
                i ← i+1
            endwhile
            if i < map.m then
                map.next[prev] ← map.next[pos]
            endif
            map.next[pos] ← -1
        endif
```

else