HashTable:
T: TKey[]
m: Integer
h: TFunction

```
subalgorithm insert (ht, e) is:
//pre: ht is a HashTable, e is a TKey
//post: e was added in ht
    i ← 0
    pos ← ht.h(e, i)
    while i < ht.m and ht.T[pos] ≠ -1 execute
    //-1 means empty space
        i ← i + 1
        pos ← ht.h(e, i)
    end-while
    if i = ht.m then
        @resize and rehash and compute the position for e again
    else
        ht.T[pos] ← e
   end-if
end-subalgorithm
```

- Removing an element from a hash table with open addressing is not simple:
  - we cannot just mark the position empty - *search* might not find other elements
  - you cannot move elements - *search* might not find other elements
- Remove is usually implemented to mark the deleted position with a special value, *DELETED*.
- In a hash table with open addressing with load factor $\alpha = n/m$ ($\alpha < 1$), the *average* number of probes is at most
  - for *insert* and *unsuccessful search*

$$\frac{1}{1-\alpha}$$

  - for *successful search*

$$\frac{1}{\alpha} * ln\frac{1}{1-\alpha}$$

- If α is constant, the complexity is $\Theta(1)$
- Worst case complexity is $\Theta(n)$