

Hw7_Hamza_Yoğurtcuoğlu_171044086

1) Skip-List is a method that is used for data structures. It aims to create a index on the list in order to speed up access to a linked list. Each node represent a element.

That has insert,delete,search operators.

a)Insert Operator : Let's insert 10.

head						tail	
2 →	4 →	5 →	9 →	11 →	15 →	21 →	

We have a such as list.

2 →						15 →	
-----	--	--	--	--	--	------	--

We determine which domain it has .

2 →		5 →				15 →	
-----	--	-----	--	--	--	------	--

10 is not in left side (2-5)

2 →	4 →	5 →	9 →	10 →	11 →	15 →	21 →
-----	-----	-----	-----	------	------	------	------

Briefly Pointer of 9 represent to 10 value and pointer of 10 represent 11 value

Delete Operator : Let's delete 6.

head						tail	
2 →	3 →	4 →	5 →	6 →	7 →	8 →	

each key falls back in the actual structure

2 →		4 →		6 →			
6 →							

Last structure

2->	3->	4->	5->	7->	8->
-----	-----	-----	-----	-----	-----

Pointer of 5 represent to 7 value

Search Operator : Let's search 52

head						tail	
5 →	11->	25 →	42 →	52 →	65 →	88 →	100 →

	11 →		42 →			88 →	
--	------	--	------	--	--	------	--

The second linked list created contains some of the elements of the original listener and skips some elements.

	11→		42→	52→		88→	
--	-----	--	-----	-----	--	-----	--

We determine in which domains 52 is (11-42 or 52-88).

	11→		42→	52→		88→	
--	-----	--	-----	-----	--	-----	--

* it is in this domain. *

/*As a result of continuing from the lower level in node 42, the first node 52 is located and we reach the value what we are looking for.*/

--For example ,Binary Search tree is modified and it needs to balance that operation can affect large part of tree that needs a lock on many of tree nodes. But, in skip-list a node to the jump list is much more local, only nodes that are directly linked to the affected node need to be locked. Briefly, accessing so fast in this data structure method.

```

2) 1) int countInList(MyList list , int key)
      int counter is zero
      int index is zero
      if(List Is Empty) :
          Search failed
      else :
          while(list.lenght>index):           //Loop
              if(key==list[index])
                  counter is increased a point.
              index is increased a point.
      return count                           //Finish

2) int firstRepeating(myList list)
      if(List Is Empty) :
          Search failed.
      int counter is zero
      int index1 is zero
      int index2 is one
      while(list.lenght>index2)               //Loop
          If(list[index1]==list[index2])
              counter is increased a point.
              index2 are increased a point.
          else : index2 are increased a point.
                  index1 are increased a point.
      return counter                         //Finish

```

3)

```
int reverse (myList list, myList newList)
    if(List Is Empty) :
        Search failed
    int counter is zero
    int index1 is zero
    int index2 is list.lengh-1
    while(index1<list.lenght or list.lenght>=0):    //Loop
        newList[index1]=list[index2]
        index2 are decreased a point.
        index1 are increased a point.
    return newList                                //Finish
```