

Exercise 8 - Solutions

1. Introduce a counter variable c , with arbitrary initialization.

When adding an item to the priority queue, use c as key. Then increment c .

When dequeuing the minimum, the oldest (smallest value of counter) will leave the queue.

(This works until the counter overflows...)

2. No, as soon as one or more items are removed, there will be duplicate keys.

The priority queue ADT does not give any guarantees about priority among the set of same keys.

As soon as two consecutive removals are done, key order is strictly wrong:

add (A)	→	(0, A)	
add (B)	→	(0, A), (1, B)	
add (C)	→	(0, A), (1, B), (2, C)	
remove, remove	→	(2, C)	
add (D)	→	(2, C), (1, D)	
remove	→	(2, C)	WRONG ☹

3. The two implementations behave differently when there are duplicate keys.

E.g. $L = [1, 2, 2, 4, 5]$, search for key $k=2$.

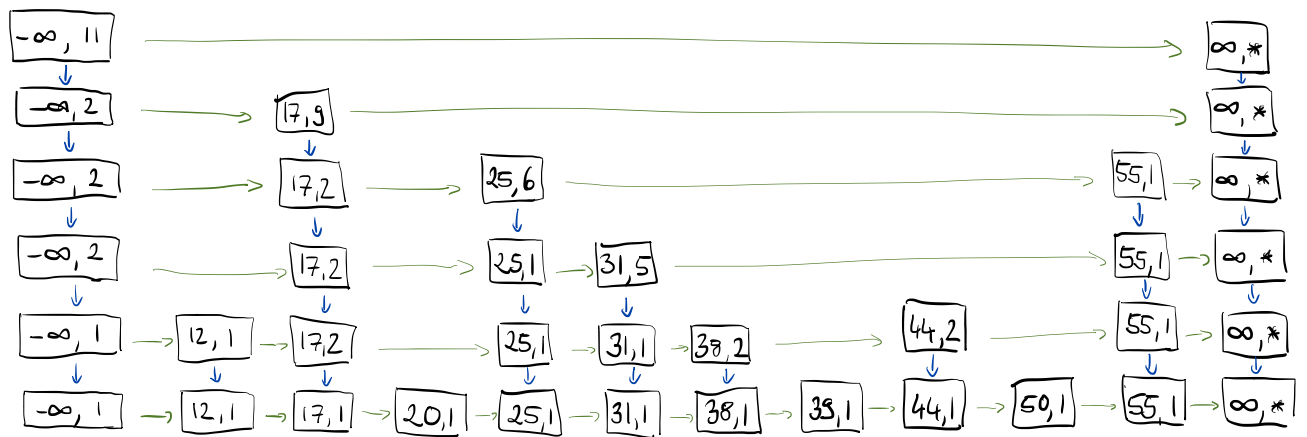
Implementation 1 will compute $\text{mid} = (0+4)/2 = 2$ and immediately find the requested key at position 2.

Implementation 1 will recurse on the sublist $[1, 2]$ and then find the requested key at position 1.

4. Only separate chaining can store more elements than there are positions in the hash table. All others will degrade much before load factor 1 is achieved.

5. The order of hash values has no relation to an order of keys. Thus, there is no way a 'next' operation can be implemented except for a brute-force linear search.

6. At each node in the skip list, add a counter as to how many nodes at lowest level lie between the node and its right neighbor. Eg.



To access the element of index i , start at position p at top left with index $j=0$.

- If $j + p.\text{step} > i$, drop down one level
- If $j + p.\text{step} = i$, you have found the element as the right neighbor of p
- If $j + p.\text{step} < i$, update $p := \text{right-neighbor}(p)$
 $j := j + p.\text{step}$
 and repeat.

The number of steps required is exactly as for search, so $O(\log n)$ on average.

Updating the step counts on removal or insertion requires an update on all nodes along the (index) search path, so is also $O(\log n)$.