

# Solutions!

## Voortentamen Data Structures and Algorithms

29 September 2009

### Task 1.

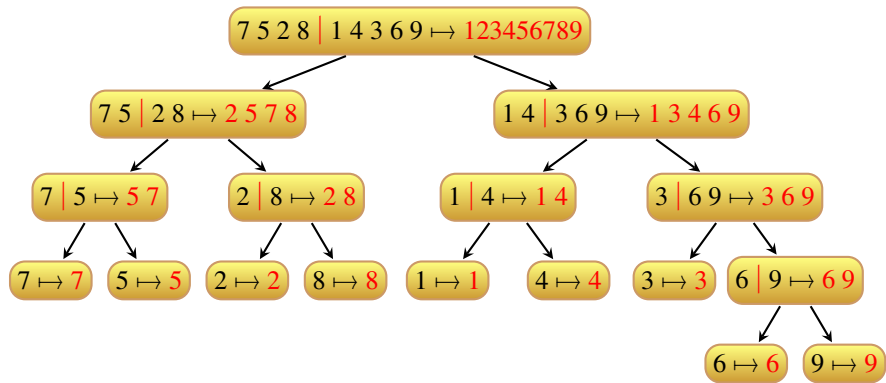
- (a) best-case:  $O(1)$   
worst-case:  $O(n)$
- (b) worst-case:  $O(n^2)$
- (c) (i)  $5 \cdot n^2 + n^3 + 2 \in O(n^3)$   
(ii)  $2 \cdot n + 5 \cdot \log_2 n \in O(n)$   
(iii)  $2^n + 7 \cdot n \cdot \log_2 n \in O(2^n)$
- (d) No, this is not the case. Two possible justifications are:
- The  $O$ -notation does not speak about small inputs. For example  $100 \cdot n \in O(n)$  and  $n^2 \in O(n^2)$ , but for  $n \leq 10$  the linear algorithm is slower  $100 \cdot n > n^2$ .
  - If we say ‘an algorithm is  $O(f(n))$ ’ we refer to the worst-case time complexity. For certain (best-case) inputs the  $O(n^2)$  algorithm could be faster (e.g.  $O(1)$ ) than the  $O(n)$  algorithm.

### Task 2.

- (a) • **enqueue** is  $O(n)$  since we have to find the insertion position ( $O(\log n)$  using binary search) and we have to make space for the inserted element ( $O(n)$  for moving  $n/2$  elements in worst case). (Thus the important part is not finding the position, but making space for the new element.)
- **dequeue** is  $O(1)$  since we remove the first element.
- (b) The complexity is  $O(n^2)$  since we have to:
- insert  $n$  times, that is,  $n$  times  $O(n)$ ,
  - and remove  $n$  times, that is,  $n$  times  $O(1)$ .

Thus the important part is inserting  $O(n^2)$ .

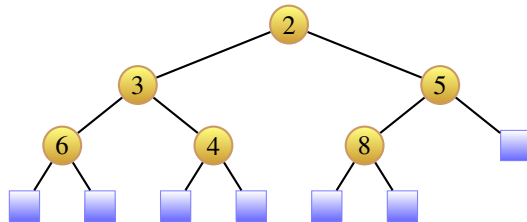
(c)



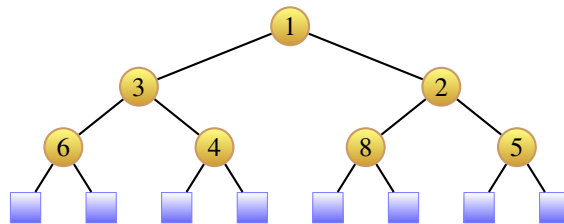
### Task 3.

Draw the final heap after every addition/removal step; intermediate steps are not needed.

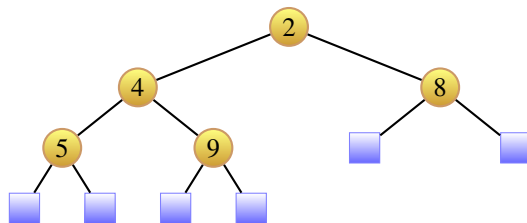
(a) After adding 5:



After adding 1:



(b) After removing 1:



#### Task 4.

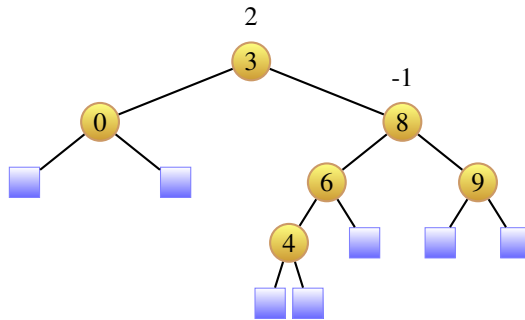
- (a) (i) inorder traversal: FDBHGARCE  
(ii) preorder traversal: ADFHBGCRE
- (b) Pseudo-code for postorder traversal:

```
postOrder(v):  
  for each child w of v do  
    postOrder(w)  
  visit(v)
```

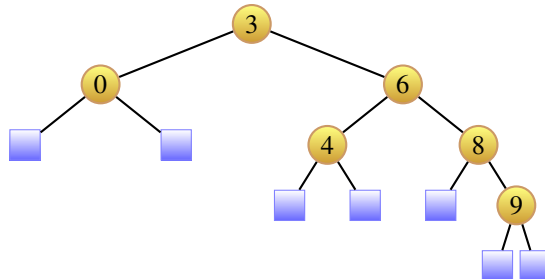
Also accepted is postorder traversal for binary trees:

```
postOrder(v):  
  if v.isInternal() then  
    postOrder(v.leftChild())  
    postOrder(v.rightChild())  
  visit(v)
```

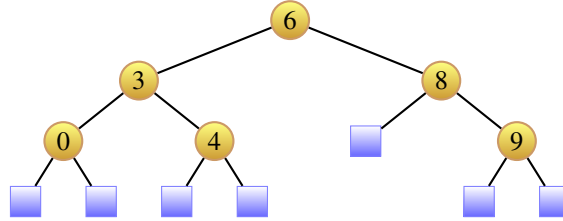
- (c) After adding 4:



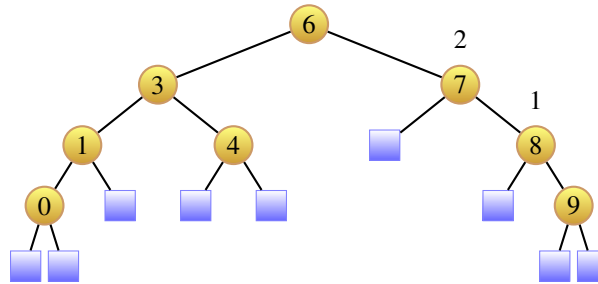
We rotate right 6 and 8:



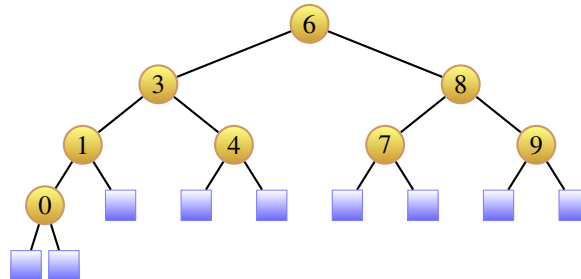
We rotate left 3 and 6:



(d) After removing 5:



We rotate left 7, 8:



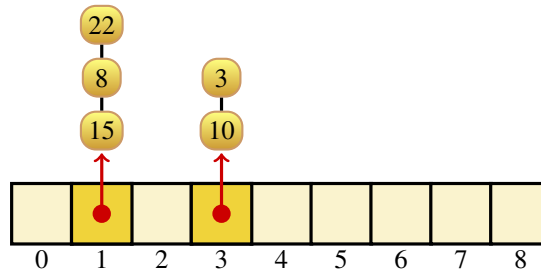
**Task 5.**

(a) Make a hash table of size 9. Use the hash function  $h(k) = k \bmod 7$ . Add to the initial empty hash table the following numbers in this order:

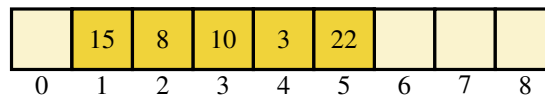
15, 10, 8, 3, 22

and use the following 3 different ways for solving collisions:

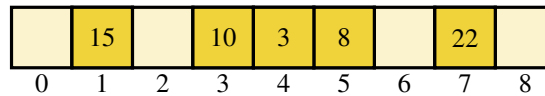
(i) chaining:



(ii) linear probing



(iii) double hashing with secondary hash function  $d(k) = 4 - (k \bmod 4)$



(b) Time complexity of searching a key  $k$  in a hash table:

- (i) best-case:  $O(1)$
- (ii) average (expected):  $O(1)$
- (iii) worst-case:  $O(n)$

The worst case happens if all items collide, that is, are mapped to the same index of the hash table. Then we get a big chain (or group) of elements and we have to search through all these elements to find a key.

### Task 6. Optional Tasks for Extra Points

- (a) An inorder traversal visits for every node first the elements in the left subtree, then the node itself, and then the elements in the right subtree. For search trees all elements in the left subtree are smaller than the key of the node itself and the elements in the right subtree are all greater. Hence the inorder traversal visits the elements in ascending order.
- (b) Let  $f(h)$  be the minimal number of inner nodes of an AVL tree of height  $h$ . Given minimal AVL trees  $A$  of height  $h$  and  $B$  of height  $h+1$  we can construct a minimal AVL tree of height  $h+2$  by taking a root node with  $A$  as left and  $B$  as right child. As a consequence we have:

$$f(h+2) = 1 + f(h) + f(h+1)$$

For  $f(0)$  and  $f(1)$  it depends whether we consider AVL trees with leaves or without. (In the lecture we have considered AVL trees with leaves, as the book does.)

- For AVL tree with leaves we have  $f(0) = 0$ ,  $f(1) = 1$ . Thus we obtain:

h	0	1	2	3	4	5	6	7	8
f(h)	0	1	2	4	7	12	20	33	54

Thus the answer is 54.

- For AVL tree without leaves we have  $f(0) = 1$ ,  $f(1) = 2$ . Thus we obtain:

h	0	1	2	3	4	5	6	7	8
f(h)	1	2	4	7	12	20	33	54	88

Thus the answer is 88.