Algorithms and Data Structures
Solutions to some exercises on heapsort, binary search trees and hash tables

Lecturer: Michel Toulouse

Hanoi University of Science & Technology
michel.toulouse@soict.hust.edu.vn

20 juin 2020

Exercises on heapsort

1. Insertion sort and merge sort are stable algorithms while heapsort and quicksort are not. Can you explain why this is so ?

   Solution : Because for insertion and merge sort, two objects with equal keys appear in the same order in sorted output as they appear in the input array to be sorted, which is not necessary the case for heapsort and quicksort.

2. Does this array $A = [23, 11, 14, 9, 13, 10, 1, 5, 7, 12]$ is a heap (max-heap) ? If not makes it a heap.

   Solution : No. Since Parent(13) is 11 in the array. This violates the max-heap property.

3. Run $\mathit{Heapify}(A, 3)$ on the array $A = [27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0]$

   Solution :

   $A[27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0]$
   $A[27, 17, 10, 16, 13, 3, 1, 5, 7, 12, 4, 8, 9, 0]$
   $A[27, 17, 10, 16, 13, 9, 1, 5, 7, 12, 4, 8, 3, 0]$

Exercises

4. *Heapify*(A, i) in the class notes is a recursive algorithm. Write an equivalent iterative algorithm.

```
MAX-HEAPIFY(A, i)
  while true
    l = LEFT(i)
    r = RIGHT(i)
    if l <= A.heap-size and A[l] > A[i]
      largest = l
    else largest = i
    if r <= A.heap-size and A[r] > A[largest]
      largest = r
    if largest == i
      return
    exchange A[i] with A[largest]
    i = largest
```

Exercises

5. Run the algorithm *BuildHeap*(A) on the array $A = [5, 3, 17, 10, 84, 19, 6, 22, 9]$

   Solution :

   $A[5, 3, 17, 10, 84, 19, 6, 22, 9]$
   $A[5, 3, 17, 22, 84, 19, 6, 10, 9]$
   $A[5, 3, 19, 22, 84, 17, 6, 10, 9]$
   $A[5, 84, 19, 22, 3, 17, 6, 10, 9]$
   $A[84, 5, 19, 22, 3, 17, 6, 10, 9]$
   $A[84, 22, 19, 5, 3, 17, 6, 10, 9]$
   $A[84, 22, 19, 10, 3, 17, 6, 5, 9]$

Exercises

6. Run *Heapsort*(A) on the the array $A = [3, 15, 2, 29, 6, 14, 25, 7, 5]$

$A[5, 13, 2, 25, 7, 17, 20, 8, 4]$
$A[5, 13, 20, 25, 7, 17, 2, 8, 4]$
$A[5, 25, 20, 13, 7, 17, 2, 8, 4]$
$A[25, 5, 20, 13, 7, 17, 2, 8, 4]$
$A[25, 13, 20, 5, 7, 17, 2, 8, 4]$
$A[25, 13, 20, 8, 7, 17, 2, 5, 4]$
$A[4, 13, 20, 8, 7, 17, 2, 5, 25]$
$A[20, 13, 4, 8, 7, 17, 2, 5, 25]$
$A[20, 13, 17, 8, 7, 4, 2, 5, 25]$
$A[5, 13, 17, 8, 7, 4, 2, 20, 25]$
$A[17, 13, 5, 8, 7, 4, 2, 20, 25]$
$A[2, 13, 5, 8, 7, 4, 17, 20, 25]$
$A[13, 2, 5, 8, 7, 4, 17, 20, 25]$
$A[13, 8, 5, 2, 7, 4, 17, 20, 25]$
$A[4, 8, 5, 2, 7, 13, 17, 20, 25]$
$A[8, 4, 5, 2, 7, 13, 17, 20, 25]$
$A[8, 7, 5, 2, 4, 13, 17, 20, 25]$
$A[4, 7, 5, 2, 8, 13, 17, 20, 25]$
$A[7, 4, 5, 2, 8, 13, 17, 20, 25]$
$A[2, 4, 5, 7, 8, 13, 17, 20, 25]$
$A[5, 4, 2, 7, 8, 13, 17, 20, 25]$
$A[2, 4, 5, 7, 8, 13, 17, 20, 25]$
$A[4, 2, 5, 7, 8, 13, 17, 20, 25]$
$A[2, 4, 5, 7, 8, 13, 17, 20, 25]$

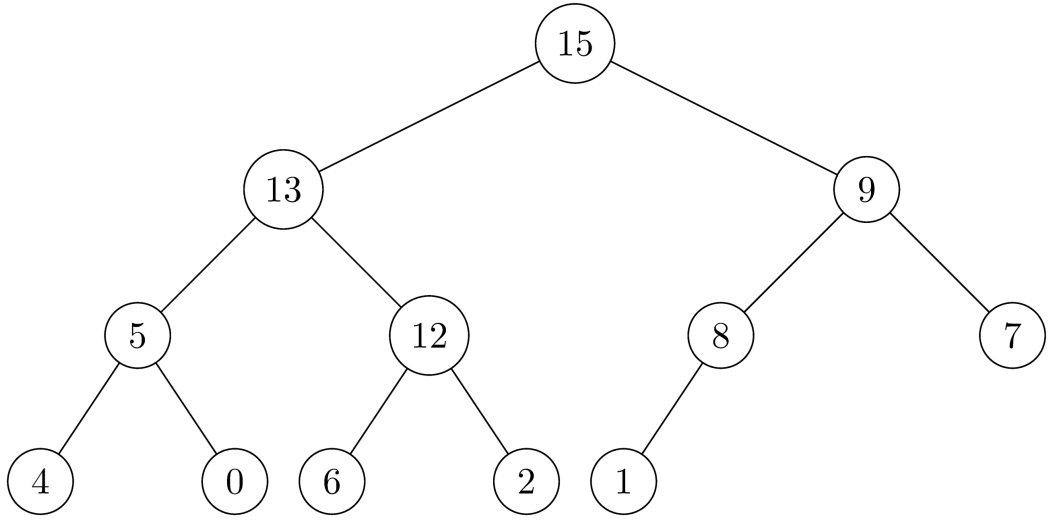
Exercises

7. What is the running time of *Heapsort* on an array A of length n that is sorted in decreasing order ?

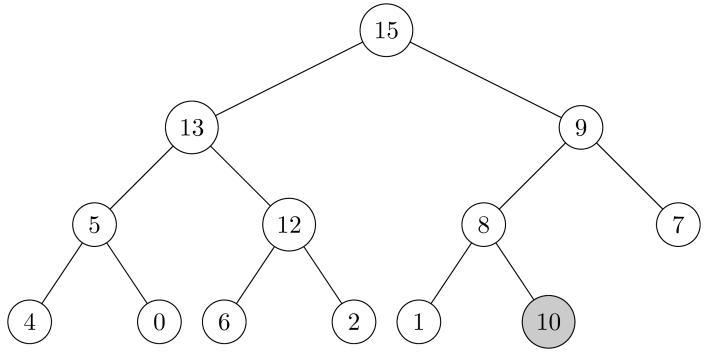
   Solution : It is the same as if the array was already sorted in increasing order. The algorithm will need to convert it to a heap that will take $O(n)$. Afterwards, however, there are $n-1$ calls to *heapify*(), each call performs $\log n$ operations.

8. Show what happens when *Insert*$(A, 10)$ is run on the heap $A = [15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1]$

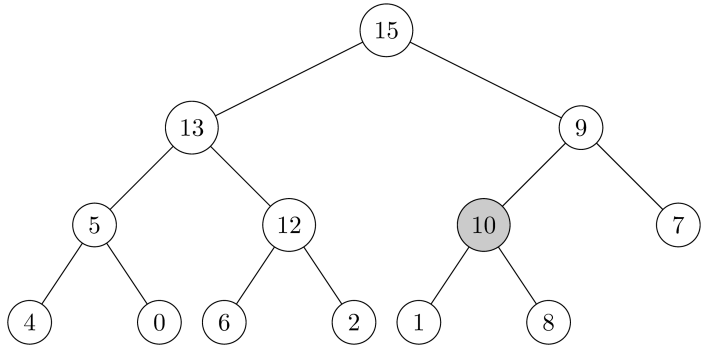
   - Original heap



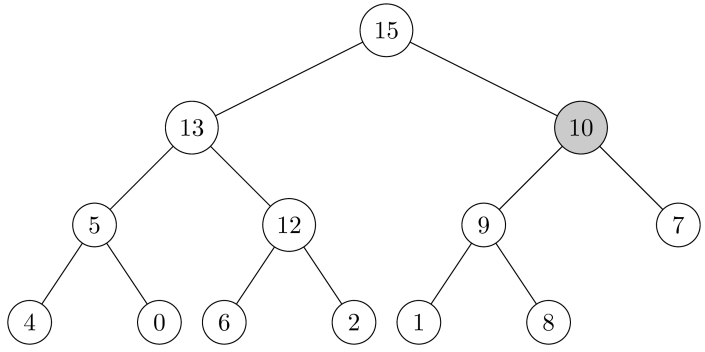
- Add the key 10 to the next position in the heap (corresponding to a new entry extending the array A).



- Since the parent key is smaller than 10, the nodes are swapped

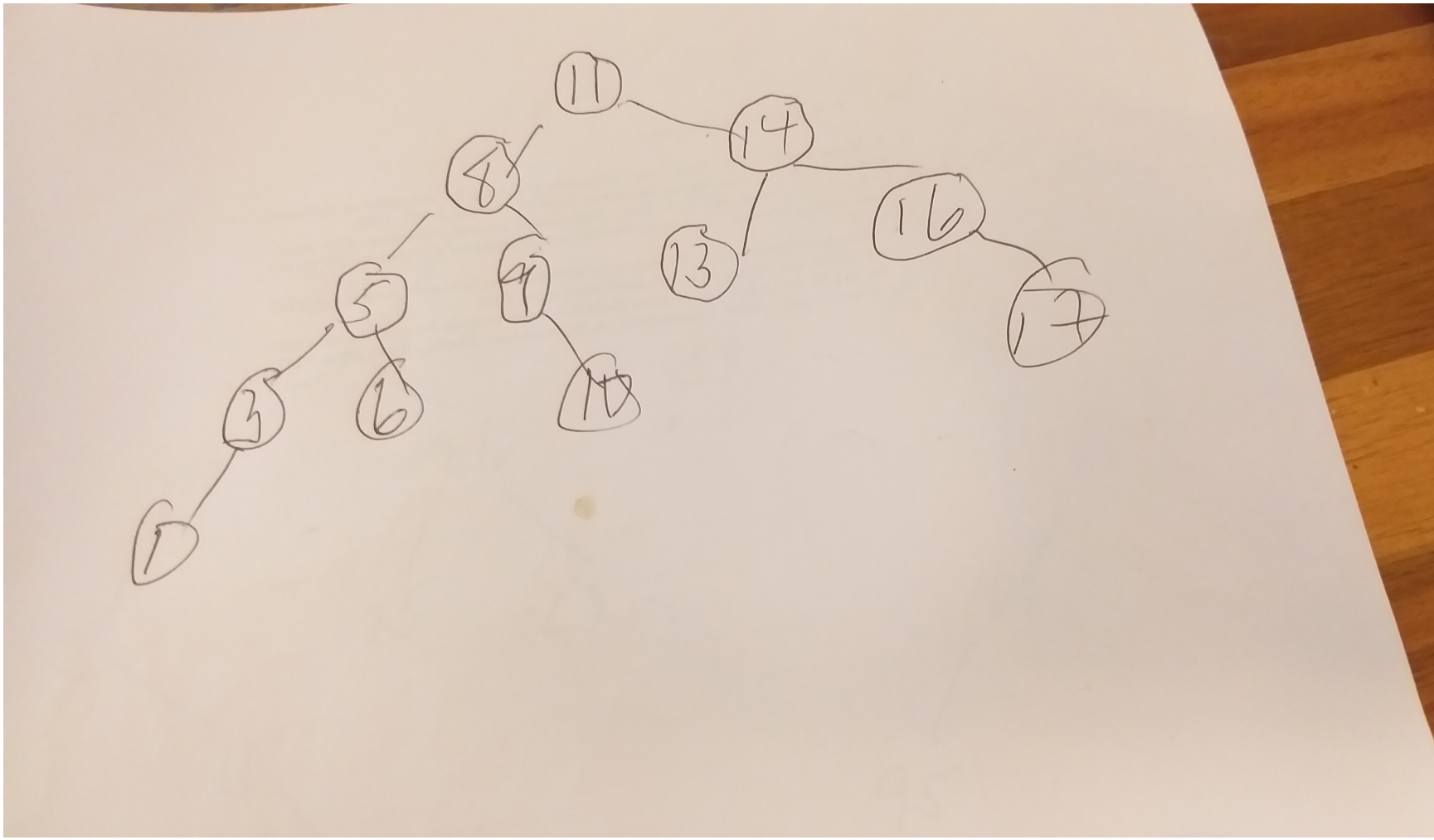


- Since the parent key is smaller than 10, the nodes are swapped



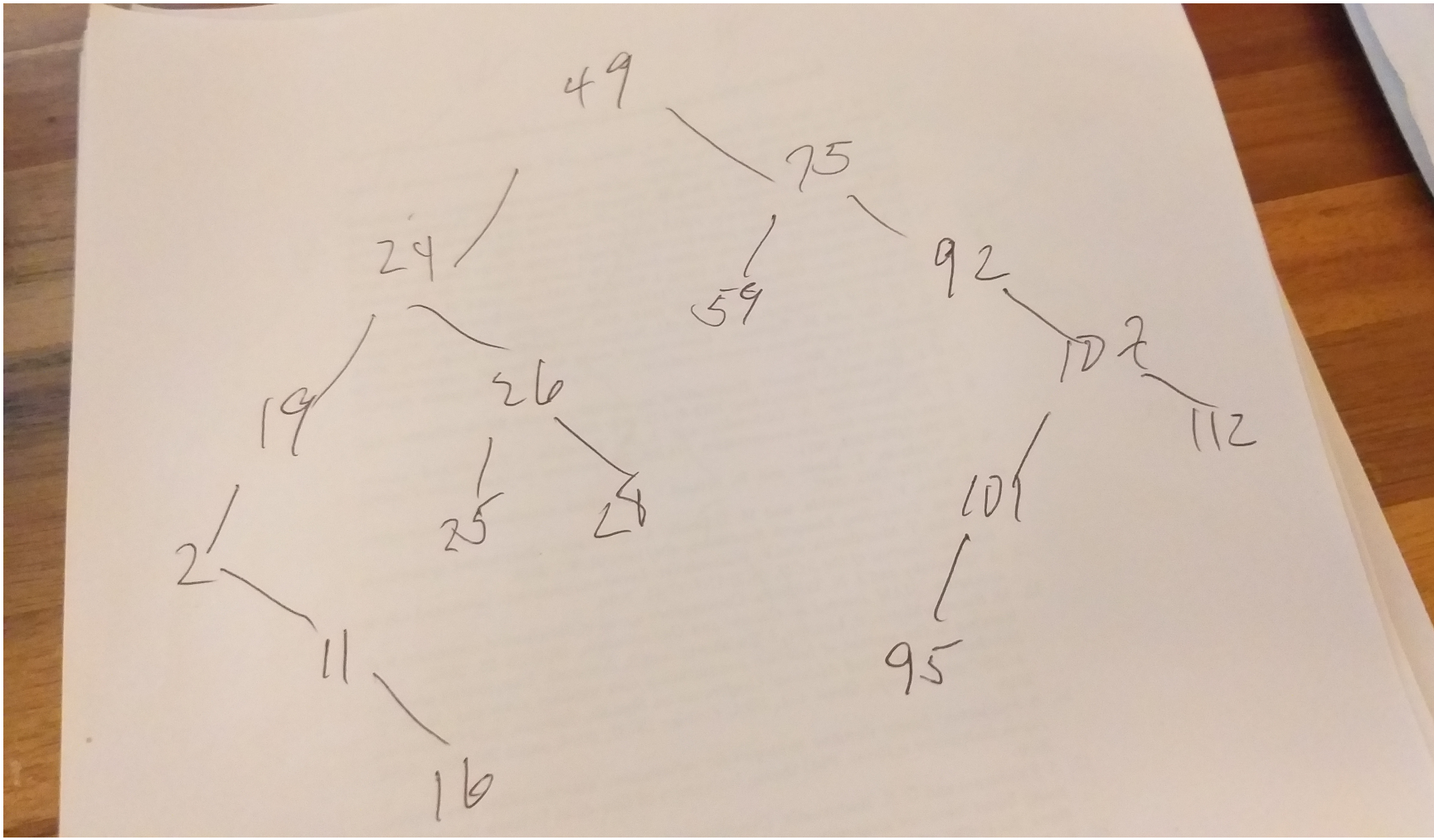
Exercises on binary search trees

1. Draw a binary search tree for the nodes 11, 8, 14, 5, 9, 13, 16, 3, 6, 10, 17, 1



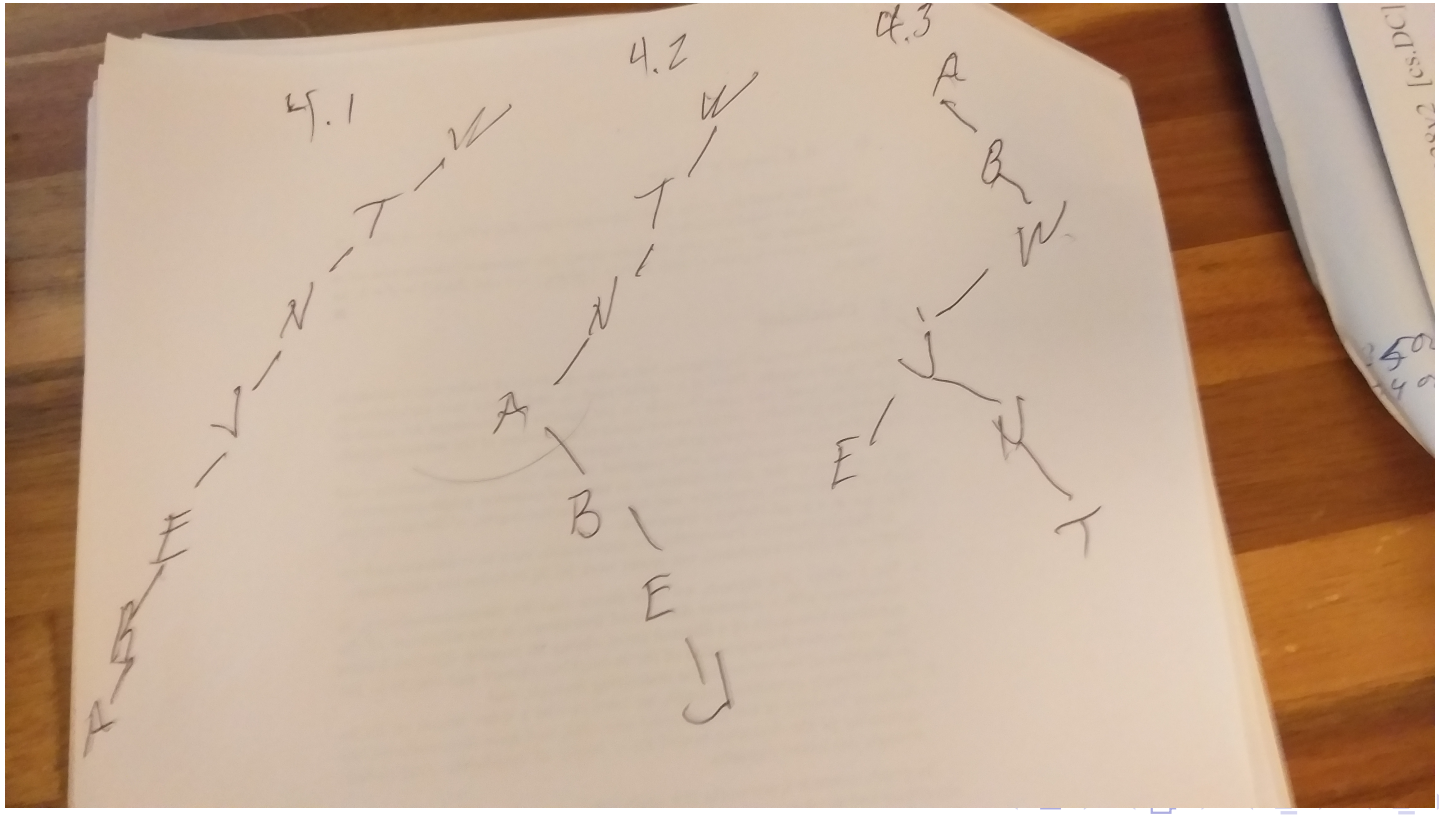
Exercises

2. Draw the binary search tree from inserting the following sequence of keys : 49 75 92 24 107 26 112 101 19 2 11 25 59 16 28 95



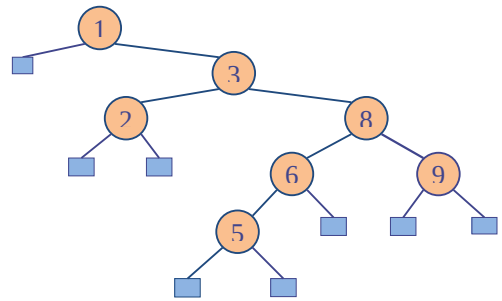
Exercises

3. For the BST of the previous question, run the *PostorderTreeWalk*(x) algorithm and list the keys in the same order as they are printed by this algorithm 16, 11, 2, 19, 25, 28, 26, 24, 59, 95, 101, 112, 107, 92, 75, 49
4. Beginning with an empty binary search tree, what binary search tree is formed when you insert the following values in the order given ?
   - 4.1 W, T, N, J, E, B, A
   - 4.2 W, T, N, A, B, E, J
   - 4.3 A, B, W, J, N, T, E

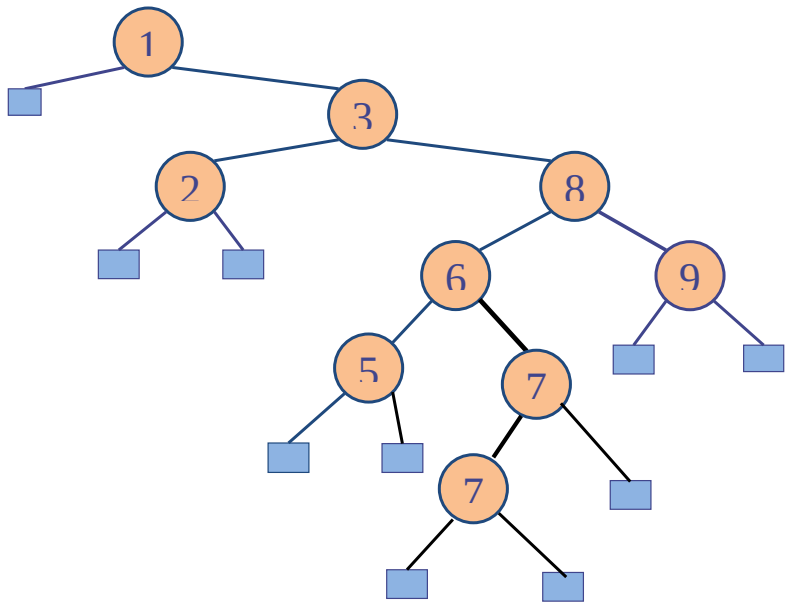


Exercise 5

Considering the BST below

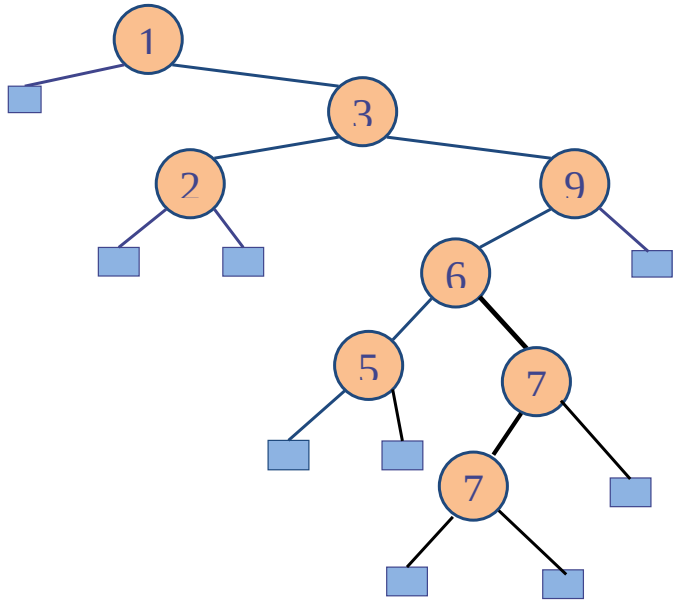


5.1 Show the steps to insert two nodes with both keys as 7.

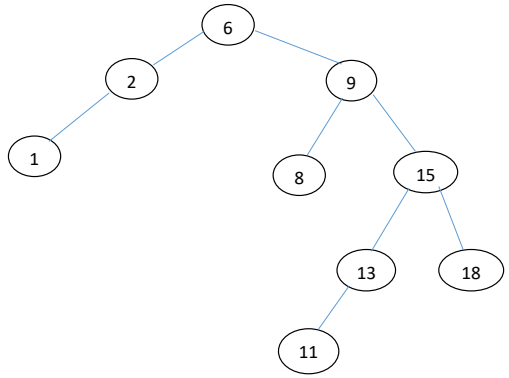


Exercise 5

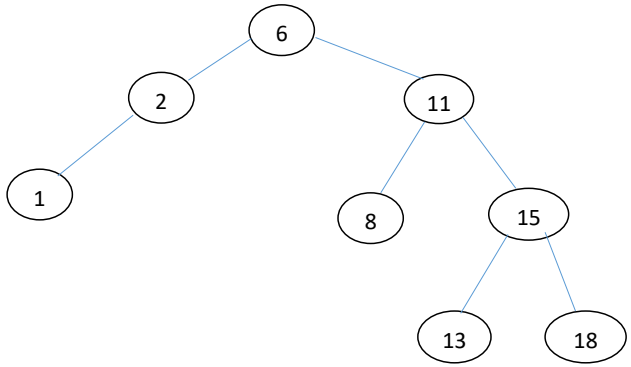
5.2 Show the steps to delete the node with key=8 (after inserting two nodes with key=7).



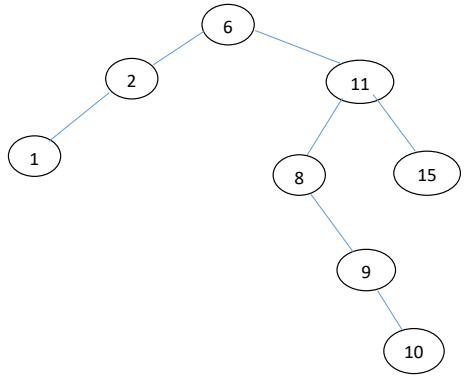
Exercise 6



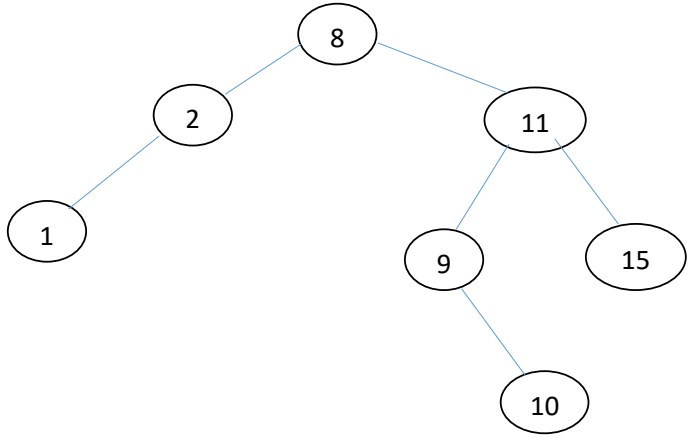
Considering the BST above, show the steps to delete the node with key=9.
Solution : Swap 9 with its successor node 11 and delete 9 using the case of a node with no child



Exercise 7



Considering the BST above, show the steps to delete the node with key=6.
Solution : swap 6 with its successor and delete 6 using the case where the node has one child.



Exercises

8. Describe an implementation of quicksort in which the comparisons to sort a set of elements are exactly the same as the comparisons to insert the elements into a binary search tree
9. Suppose that we have numbers between 1 and 1000 in a binary search tree, and we want to search for the number 363. Which of the following sequences could not be the sequence of nodes examined ?

   Solution : the sequence 9.3

   9.1 2, 252, 401, 398, 330, 344, 397, 363.
   9.2 924, 220, 911, 244, 898, 258, 362, 363.
   9.3 925, 202, 911, 240, 912, 245, 363.
   9.4 2, 399, 387, 219, 266, 382, 381, 278, 363.
   9.5 935, 278, 347, 621, 299, 392, 358, 363.
10. Write recursive versions of TREE-MINIMUM and TREE-MAXIMUM.
11. Write an iterative algorithm that executes an inorder tree traversal. (Hint : Use a stack as an auxiliary data structure)

Exercises on hash tables

1. Given the values 2341, 4234, 2839, 430, 22, 397, 3920, a hash table of size 7, and hash function h(x) = x mod 7, show the resulting tables after inserting the values in the above order with each of these collision strategies :
   - 1.1 Chaining
   - 1.2 Linear probing
   - 1.3 Quadratic probing where $c_1 = 0$ and $c_2 = 1$
   - 1.4 Double hashing with second hash function $h_2(x) = (2x - 1) \mod 7$

solutions	0	1	2	3	4	5	6
Chaining	3920	22		430 $\rightarrow$ 2341	2839	397	4234
Linear	397	22	3920	2341	2839	430	4234
Quadratic	397	22	3920	2341	2839	430	4234
Double hash	430	22	3920	2341	2839	397	4234

Note : double hashing for 3920, the solution is obtained pour $i = 5$, i.e. 30 mod 7 = 2

Exercises

2. Suppose you a hash table of size $m = 9$, use the division method as hashing function $h(x) = x \mod 9$ and chaining to handle collisions. The following keys are inserted : 5, 28, 19, 15, 20, 33, 12, 17, 10. In which entries of the table do collisions occur ?

   Solution : keys 28, 19 and 10 collide on index 1, and 15 and 33 collide on index 6

3. Now suppose you use the same hashing function as above with linear probing to handle collisions, and the same keys as above are inserted. More collisions occur than in the previous question. Where do the collisions occur and where do the keys end up ?

   Solution :

0	1	2	3	4	5	6	7	8
10	28	19	20	12	5	6	33	17

   Here 19 collide with 28 on index 1, the probing sequence stores 19 on index 2. 20 collide with 19 on index 2, the probing sequence brings 20 on index 3. 10 collide with 28 on index 1, the probing sequence try index 2, 3, 4, 5, 6, 7, 8 and finally find an empty entry in index 0.

Exercises

4. Fill a hash table when inserts items with the keys D E M O C R A T in that order into an initially empty table of m $= 5$, using chaining to handle collisions. Use the hash function $11k \mod m$ to transform the kth letter of the alphabet into a table index, e.g., $hash(I) = hash(9) = 99 \mod 5 = 4$

   Solution :

0	1	2	3	4
R→M→E	A	T→D	C	D

5. Fill a hash table when inserting items with the keys R E P U B L I C A N in that order into an initially empty table of size m $= 16$ using linear probing. Use the hash function $11k \mod m$ to transform the kth letter of the alphabet into a table index.

Exercises

6. Suppose you use one of the open addressing techniques to handle collisions and you have inserted so many keys/values into your hash table such that all entries are taken. Then collisions occur every time. What can you do ?

   Solution : You need to build a larger hash table (a much bigger m). The hashing function will be mod m for a new value of m, so the hash function changes. With the new hash function, take each of the key/values from the current full hash table and put them (in some arbitrarily chosen sequence) into the new hash table. You will probably have some collisions when you do this, and you need to resolve them using some collision scheme.

7. Suppose a hash table with capacity $m = 31$ gets to be over .75 full. We decide to rehash. What is a good size choice for the new table to reduce the load factor below .5 and also avoid collisions ?

   Solution : 3/4ths full : more than ceil(3/4 * 31) = 24 elements

   Must be a prime to avoid collisions

   Two possible answers to reduce the load factor below .5 :

   1) m = next prime higher than $31 \times 2 = 62$ m = 67

   2) m = next prime higher than $24 \times 2 = 48$ m = 53