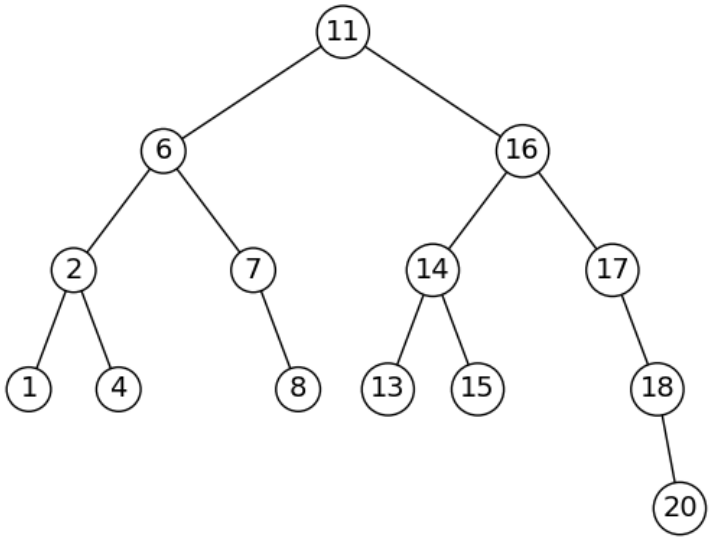
CS2302 - Data Structures

Spring 2020

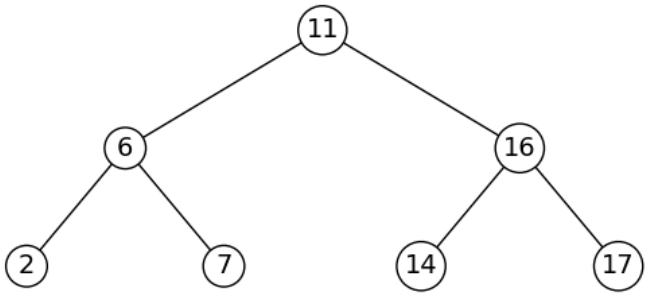
Practice Exam # 2

1. Write the function *path_to_largest(T)* that receives a reference to a binary search tree T and returns a list containing the elements in the path from the root to the largest element in the tree. For example, if T is the tree in the figure, *path_to_largest(T)* should return the list [11,16,17,18,20].

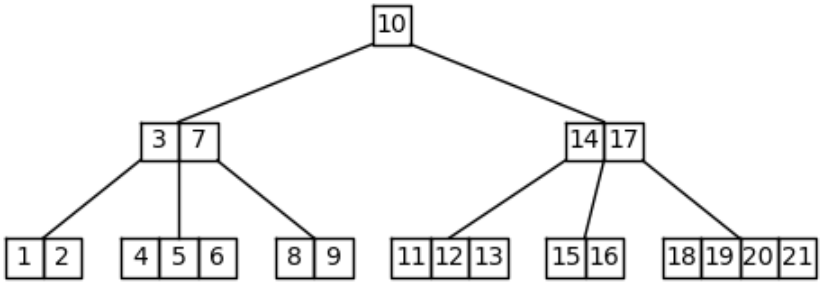


2. Write the function *path_to_k(T,k)* that receives a reference to a binary search tree T and an integer k returns a list containing the elements in the path from the root of the tree to k, if k is in the tree, otherwise the function should return an empty list. For example, if T is the tree in the figure, *path_to_k(T,15)* should return the list [11,16,14,15] and *path_to_k(T,5)* should return [].

3. Write the function *prune_BST(T,d)* that receives a reference to a binary search tree T and an integer d and removes from T all the nodes that have depth greater than d. For example, if T is the tree in the figure above, after executing *prune_BST(T,2)*, T should be the tree in the figure below.

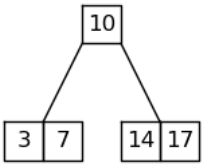


4. Write the function *keys_in_path_to_smallest(T)* that receives a reference to a B-tree T and returns a sorted list containing all the keys that are stored in nodes in the path from the root to the node that contains the smallest element in the tree. For example, if T is the tree in the figure below, *smallest_in_nodes(T)* should return [1, 2, 3, 7, 10].



5. Write the function *smallest_in_nodes(T)* that receives a reference to a B-tree T and returns a sorted list containing the smallest key in each of the nodes of T. For example, if T is the tree in the figure, *smallest_in_nodes(T)* should return [1, 3, 4, 8, 10, 11, 14, 15, 18].

6. Write the function *prune_Btree(T,d)* that receives a reference to a B-tree T and an integer d and removes from T all the nodes that have depth greater than d. For example, if T is the tree in the figure above, after executing *prune_Btree(T,1)*, T should be the tree in the figure below.



7. Write the function *item_status(H,k)* that receives a hash table H and an integer k and returns -1 if key k is not in H, 0 if key k is in H and it's the only record in its bucket, and 1 if key k is in H and k is not the only record in its bucket. For example, if H is table below, *item_status(H,99)* should return -1, *item_status(H,3)* should return 0, and *item_status(H,11)* should return 1.

```
Table contents:
bucket 0: [ [35, Bellinger] [50, Betts] ]
bucket 1: [ [31, Pederson] [11, Pollock] ]
bucket 2: [ ]
bucket 3: [ [3, Taylor] ]
bucket 4: [ [14, Hernandez] ]
```

8. All DNA is composed of a series of nucleotides abbreviated as A, C, G, and T, for example: *GACCGAATCCG*. When studying DNA, it is sometimes useful to identify repeated sequences within the DNA. Write the function *repeats(S,c)* that receives a string S and an integer c and returns all the sequences of length c that appear more than once in S. Use a hash table to implement this function in time $O(n)$. For example, if S = *'GACCGAATCCG'*, *repeats(S,1)* should return ['C', 'G', 'A'], *repeats(S,2)*should return ['GA', 'CC', 'CG'], *repeats(S,3)* should return ['CCG'], and *repeats(S,4)* should return [].

9. Write the function *build_index_table(L)* that receives a list L and returns a hash table containing the indices of the items in L. For example, if $L = [2, 4, 6, 1, 2, 3, 1, 12]$, after executing h = *build_index_table(L)*, the contents of the table should be as shown below and *h.retrieve(2)* should return the list [0,4]; *h.retrieve(4)* should return the list [1] and *h.retrieve(23)* should return None.

```
Table contents:
bucket 0: [ ]
bucket 1: [ [1, [3, 6]] ]
bucket 2: [ [2, [0, 4]] ]
bucket 3: [ [3, [5]] ]
bucket 4: [ [4, [1]] [12, [7]] ]
bucket 5: [ ]
bucket 6: [ [6, [2]] ]
bucket 7: [ ]
```