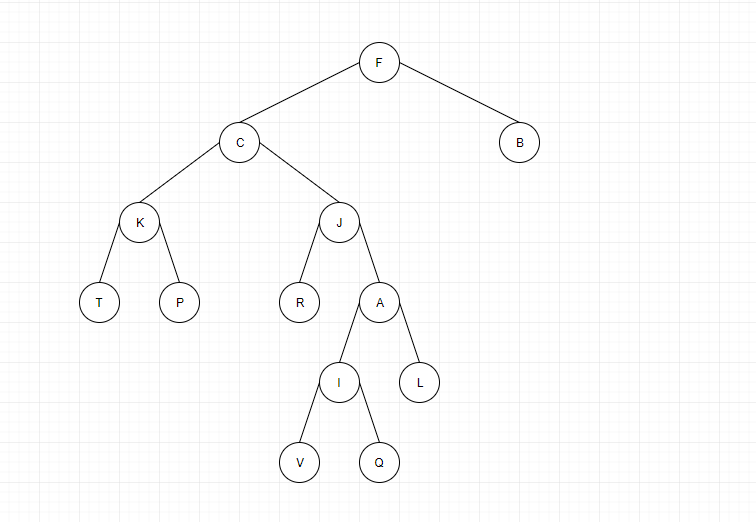
Andrew Ha – 40088418  
Nora Houari  
COMP 352  
Assignment 3 - Theory

1. 

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **-** | **F** | **C** | **B** | **K** | **J** | **-** | **-** |
| **T** | **P** | **R** | **A** | **-** | **-** | **-** | **-** |
| **-** | **-** | **-** | **-** | **-** | **-** | **I** | **L** |
| **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| **-** | **-** | **-** | **V** | **Q** | **-** | **-** | **-** |
| **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |
| **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** |

Question 3:

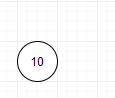
3A) This will depend on which method you will use more. If you are going to use more insertion operations based on indices go with the array based, but if you are going to be doing more insertions at the beginning then use the doubly linked list implementation. This is because adding at the beginning is faster for doubly linked lists, while indices are faster for arrays.

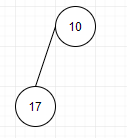
3B) The better implementation to go with would be a doubly linked list. This is because in the scenario you only care about the position of an element. Since we only care about positions, doubly linked lists are best at this.

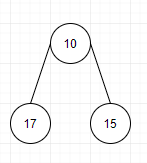
3C) Once again, it will depend on which operations you’re going to be doing more of. If you are going to be doing more removal at positions, such as beginning and end then go with the doubly linked list, However if you are doing more setting values then go with the array implementation. This is because there is no real benefit unless you are doing more of one operation then the other.

Question 4)

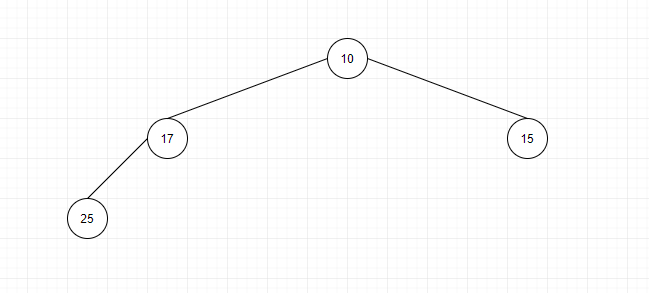
1. Insert 10



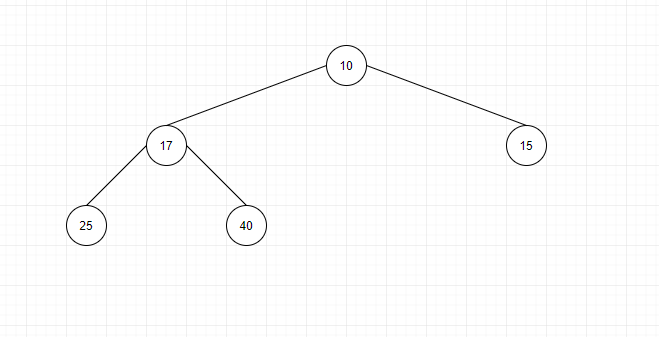
1. Insert 17  
     
   
2. Insert 15



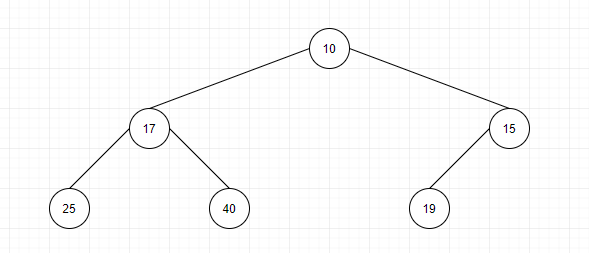
1. Insert 25



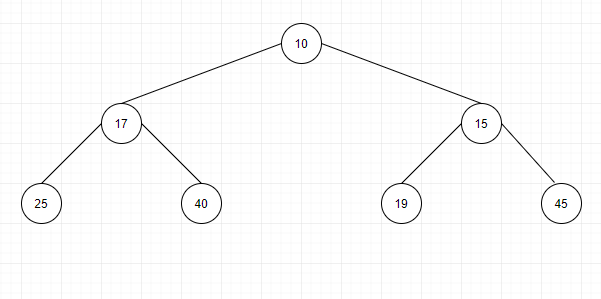
1. Insert 40

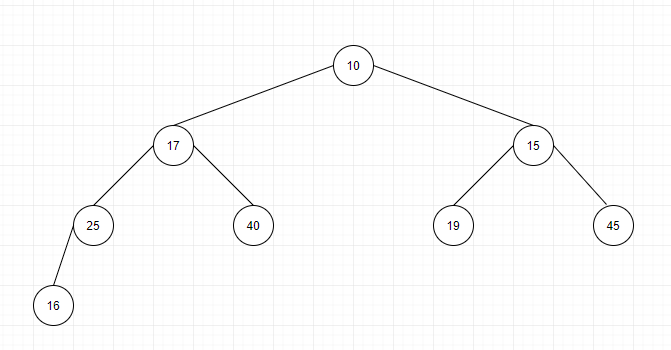
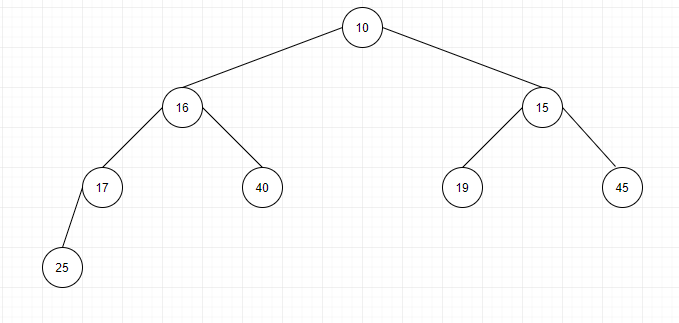
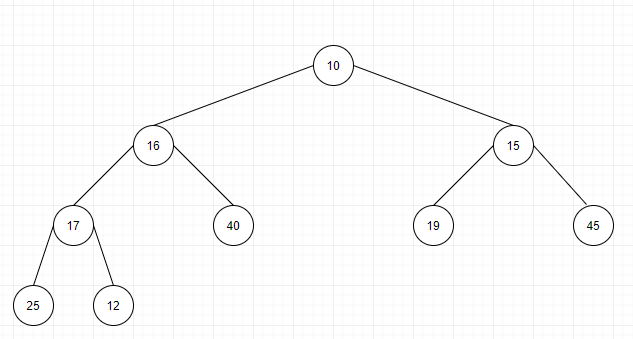
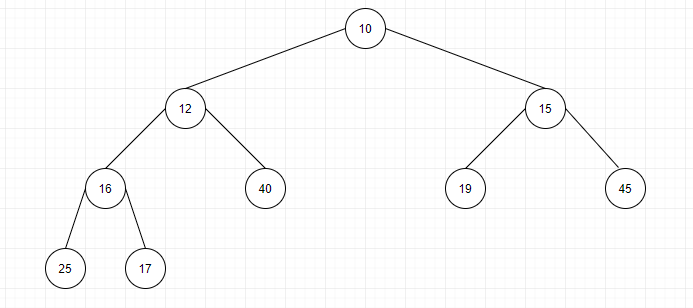
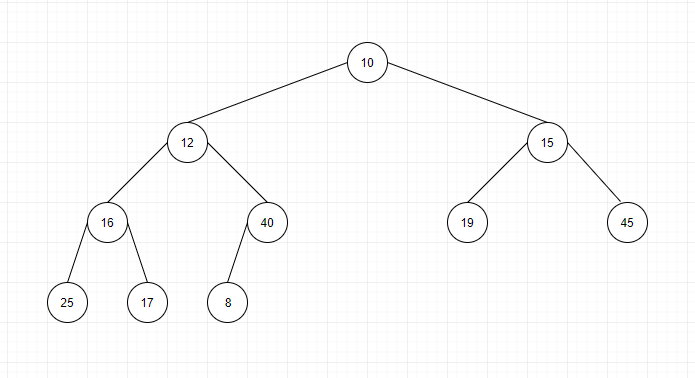
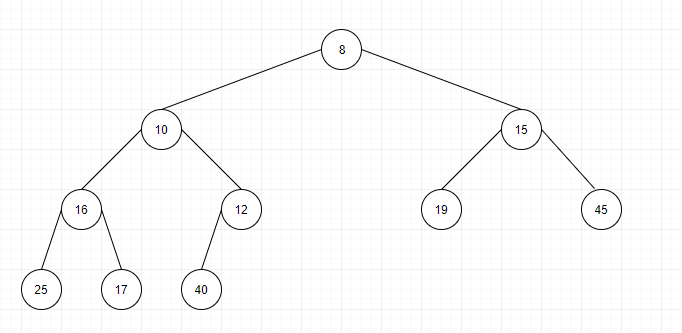
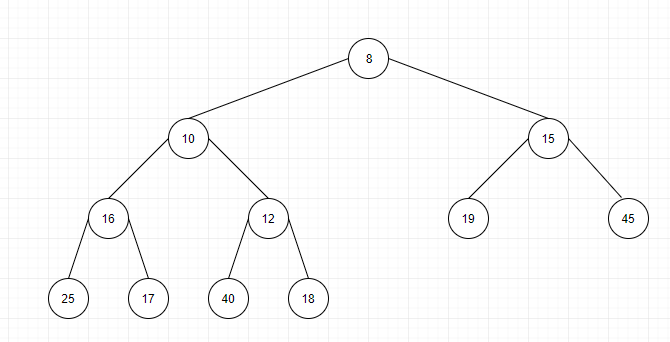
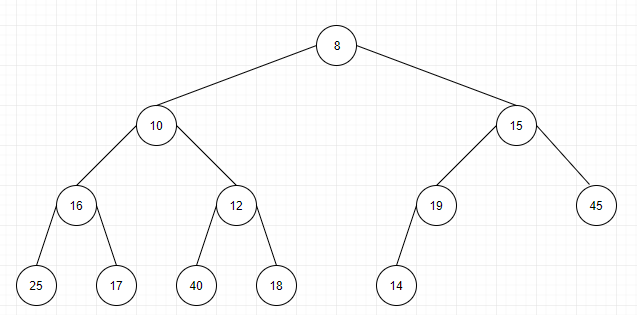
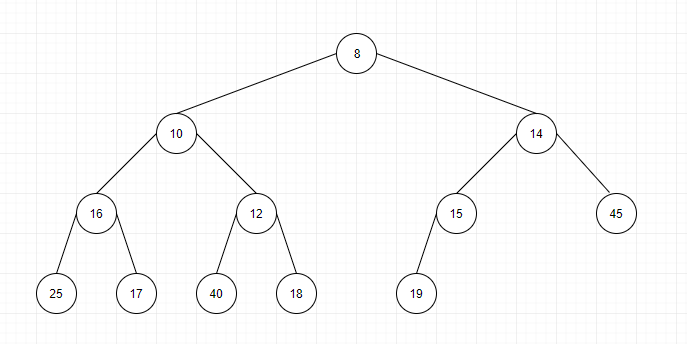
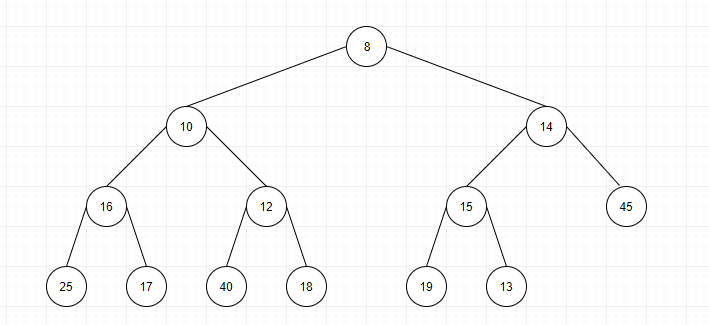
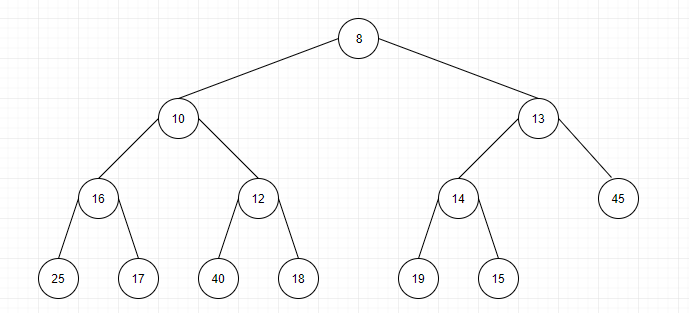
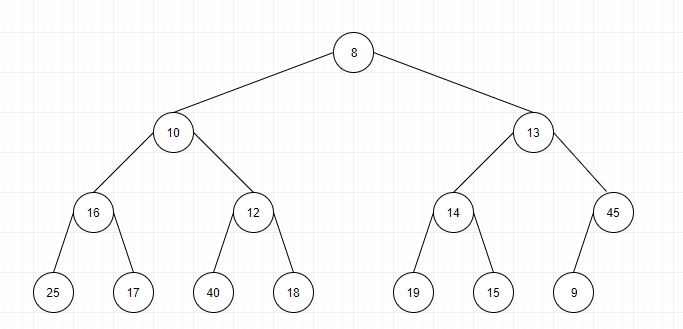
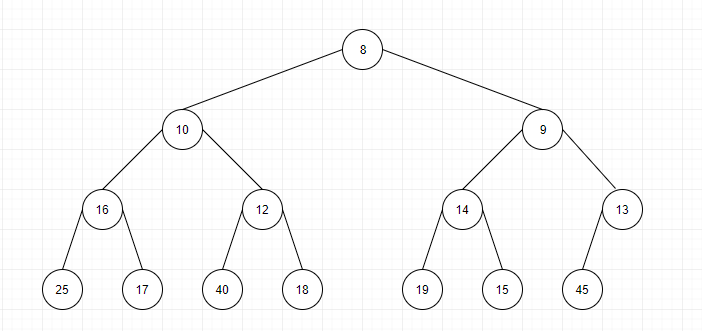
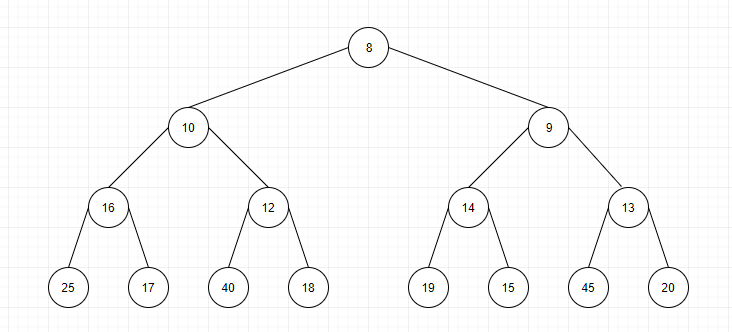
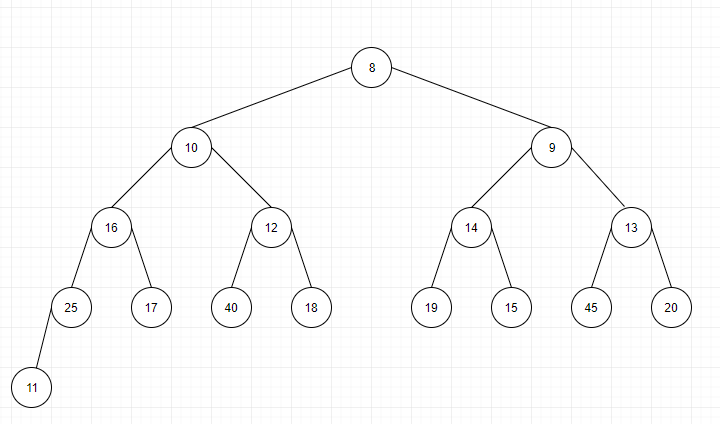
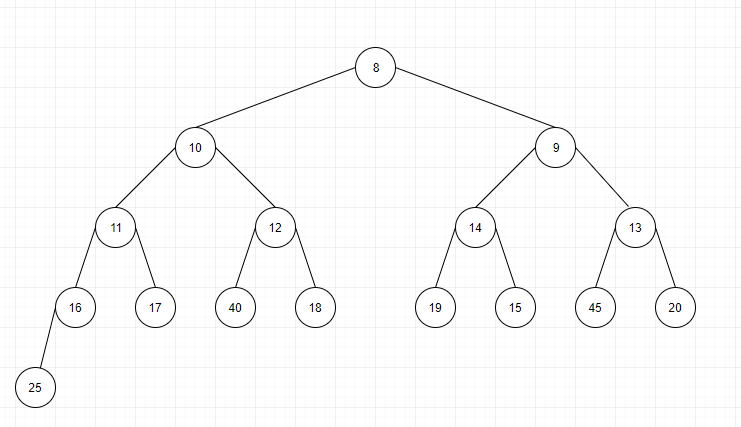
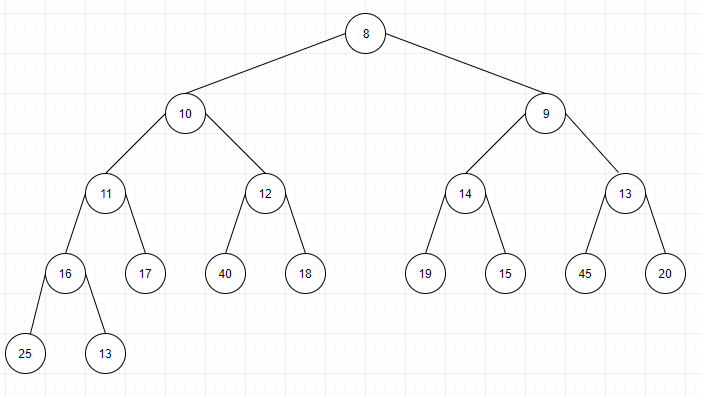
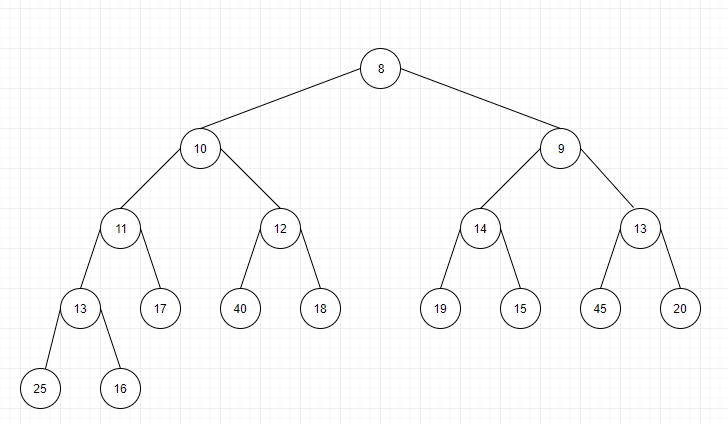
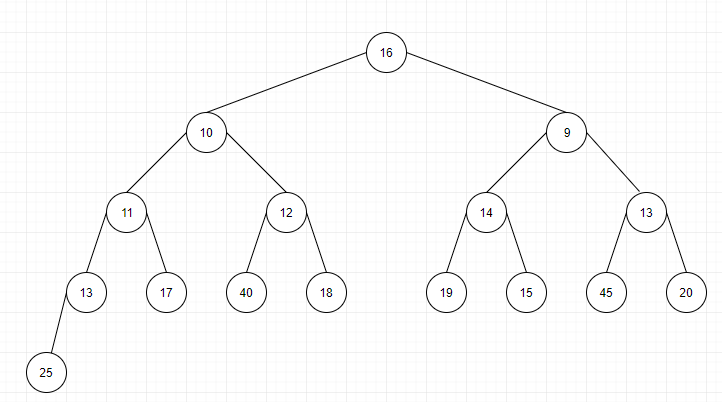
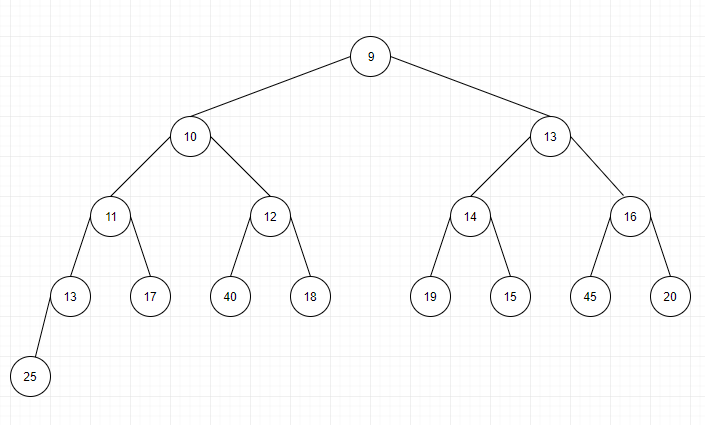
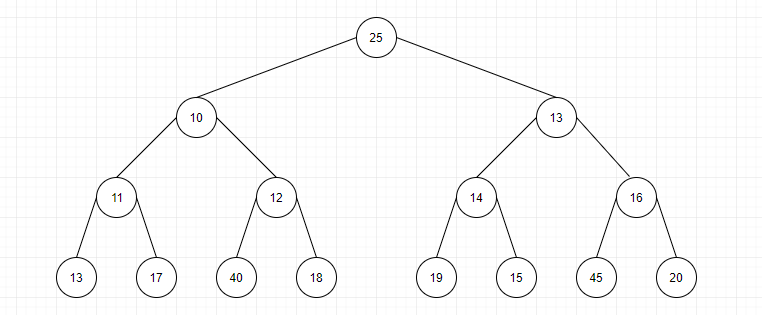


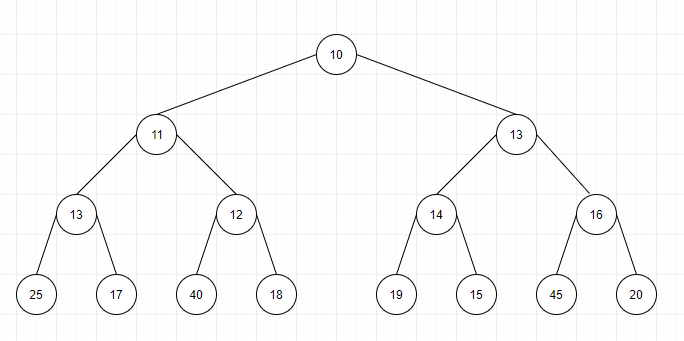
1. Insert 19



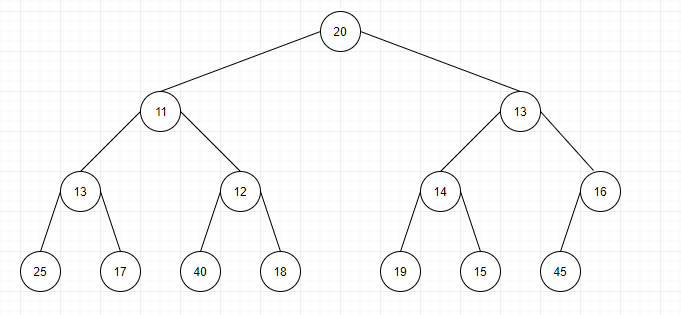
1. Insert 45

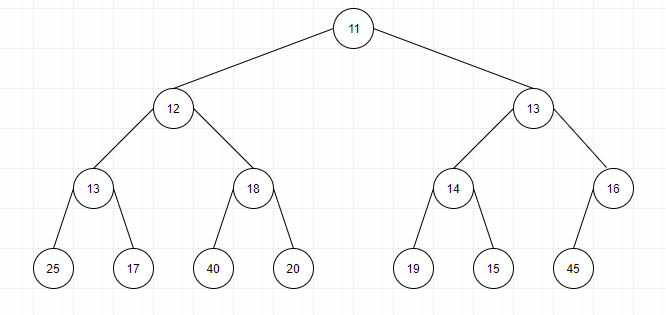


1. Insert 16 and swap  
     
   
2. Insert 12 and Swap  
     
     
   
3. Insert 8 and Swap  
     
   
4. Insert 18   
   
5. Insert 14 and swap  
     
   
6. Insert 13 and swap  
     
   
7. Insert 9 and swap  
     
   
8. Insert 20  
   
9. Insert 11 and Swap  
     
   
10. Insert 13 and Swap and final Tree  
      
    
11. RemoveMin and DownHeap  
      
    
12. Remove min and downheap  
    

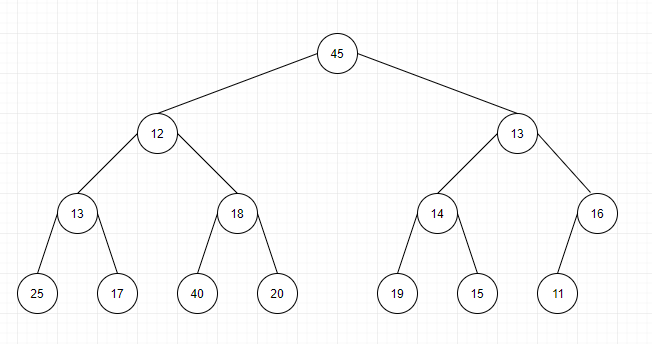
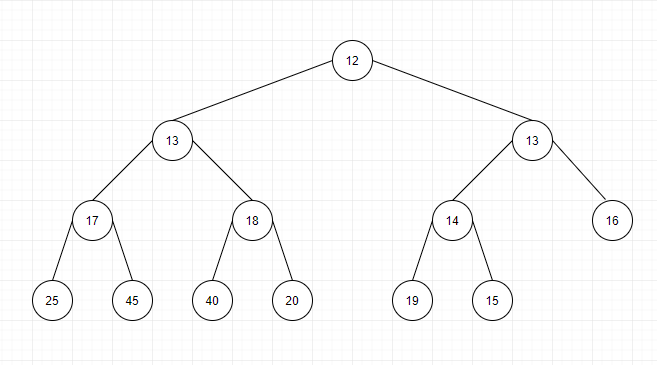


1. Remove Min and DownHeap

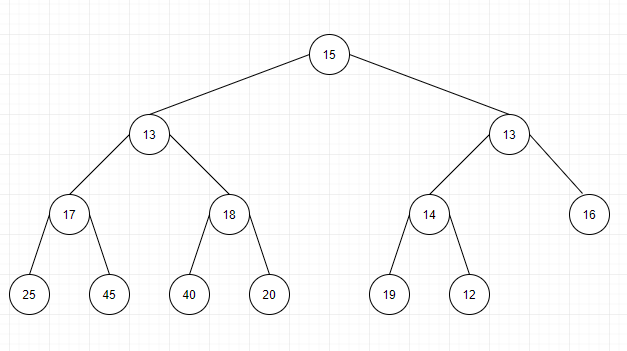


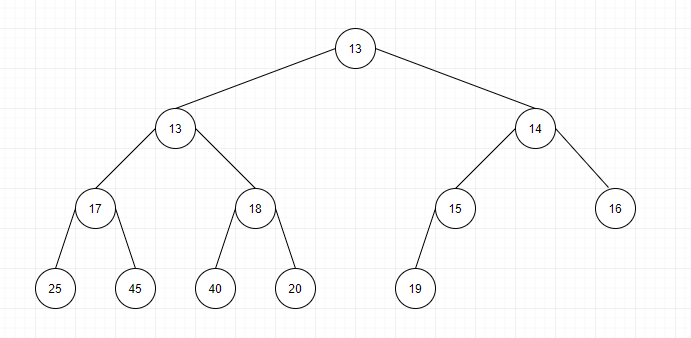


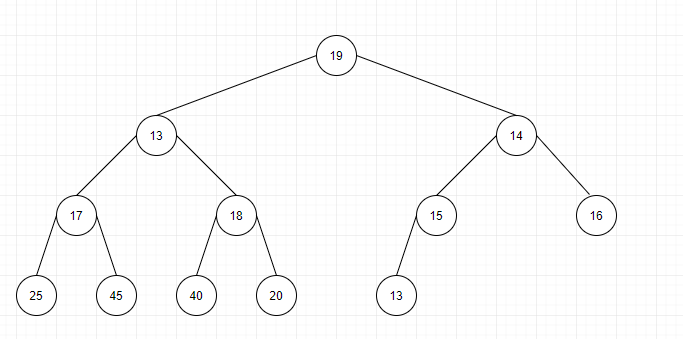
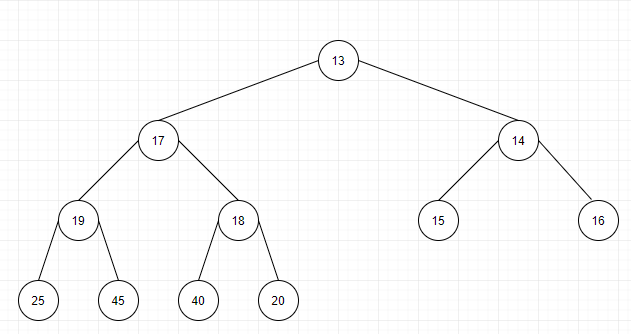
1. Remove Min and DownHeap

1. Remove Min and Swap

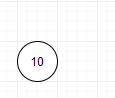


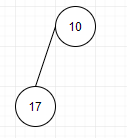


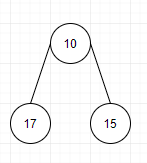
1. Remove Min and Swap / Final Tree  
     
   

Question 5)

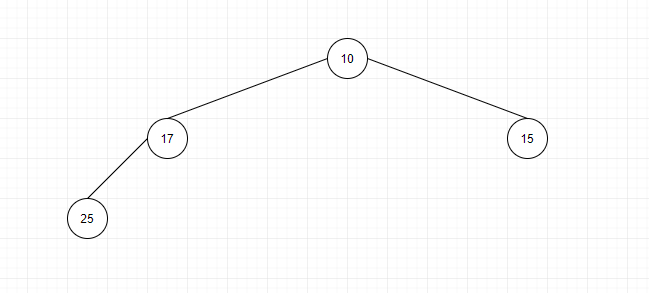
1. Insert 10



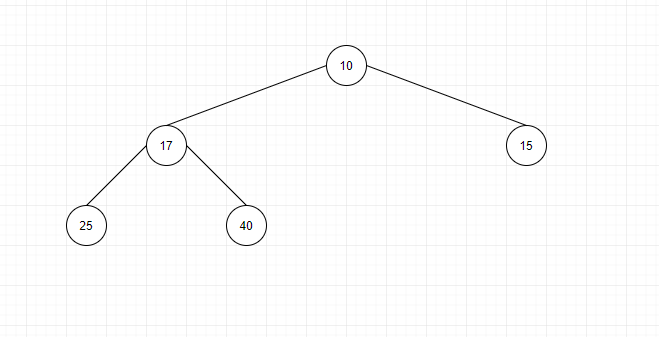
1. Insert 17  
     
   
2. Insert 15



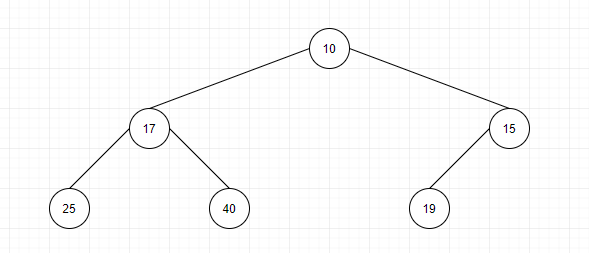
1. Insert 25



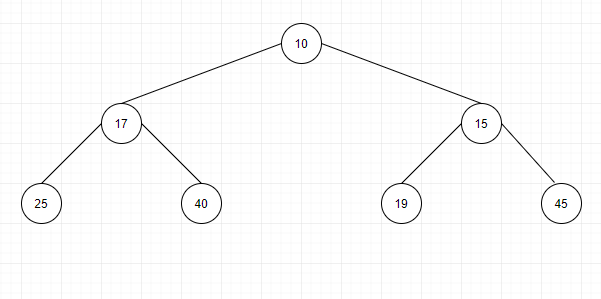
1. Insert 40

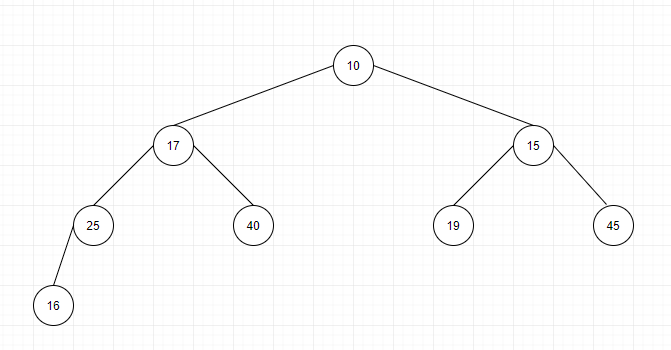
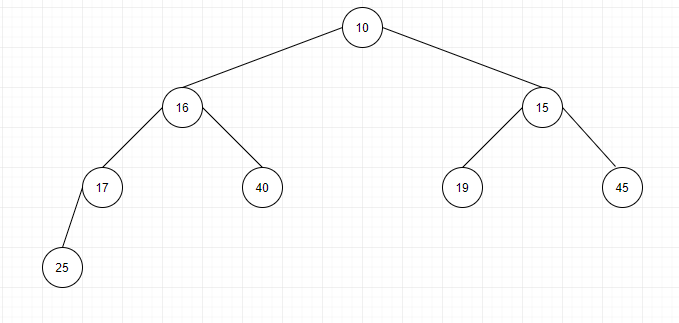
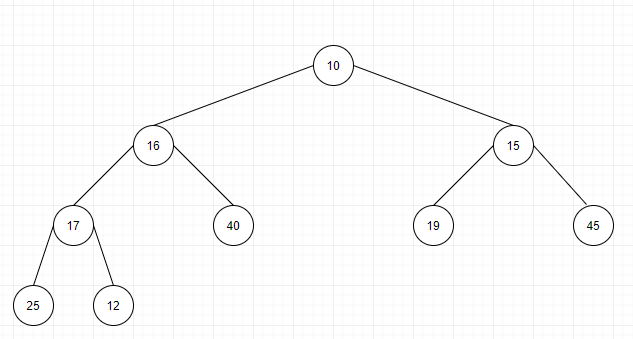
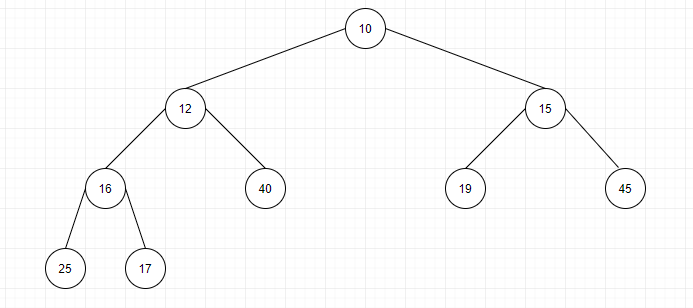
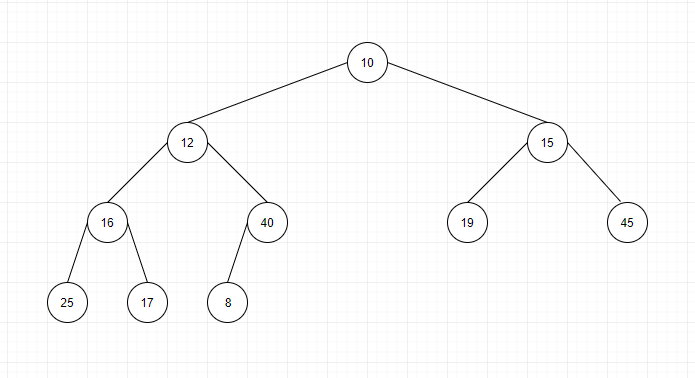
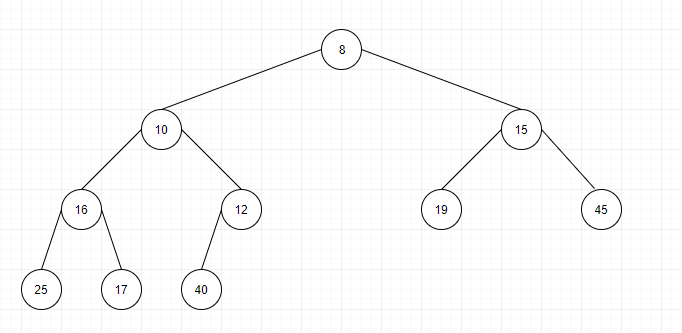
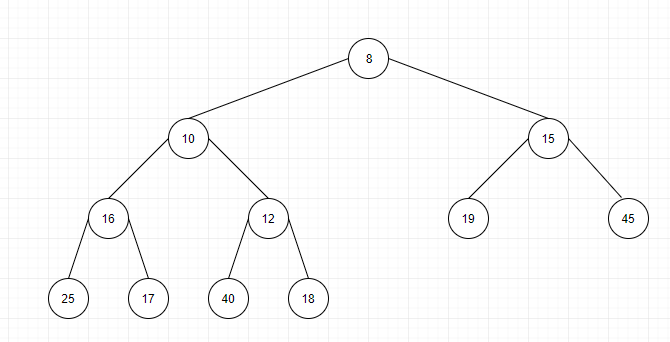
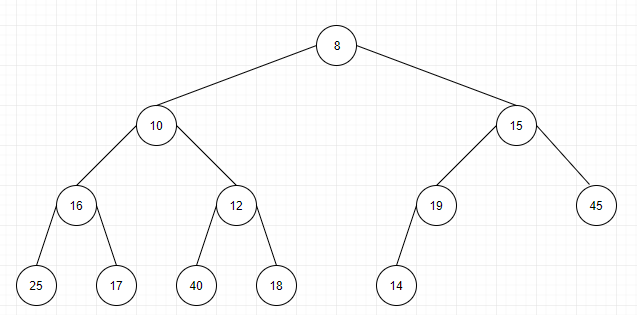
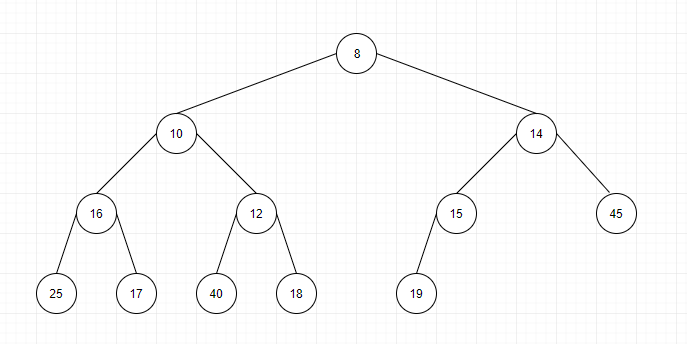
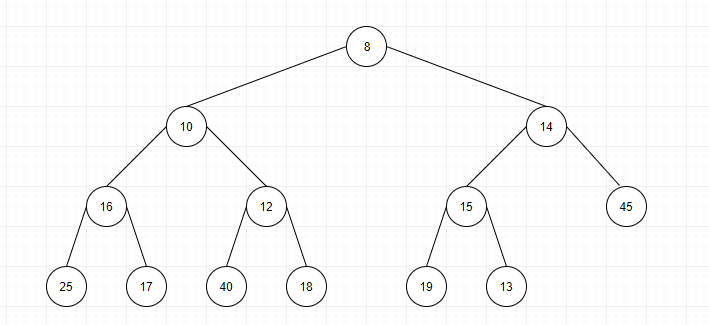
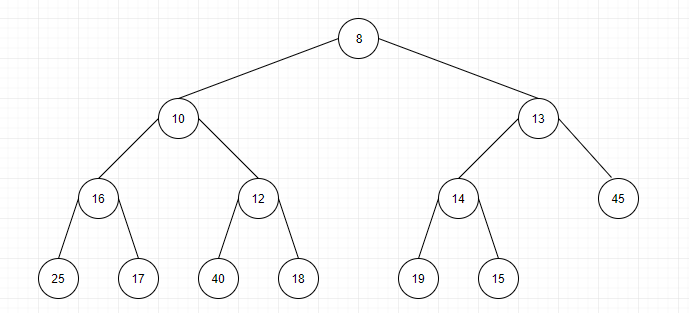
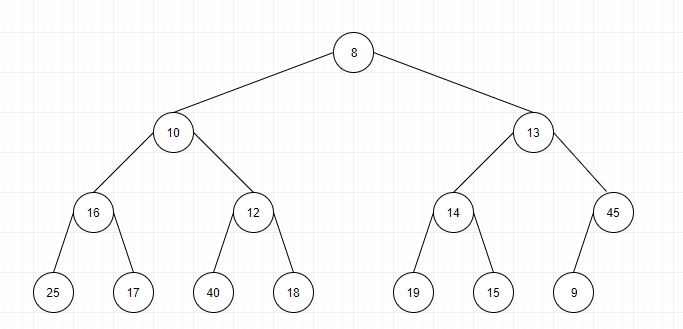
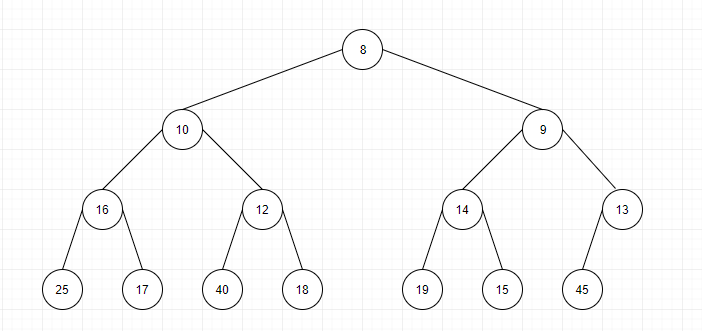
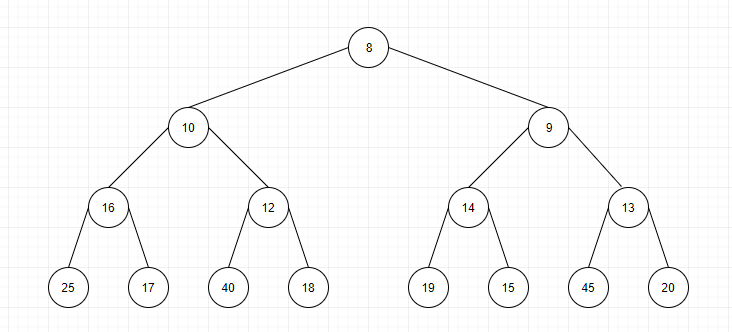
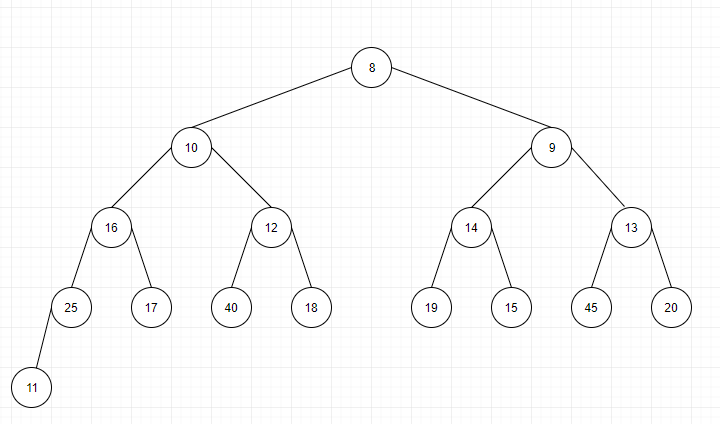
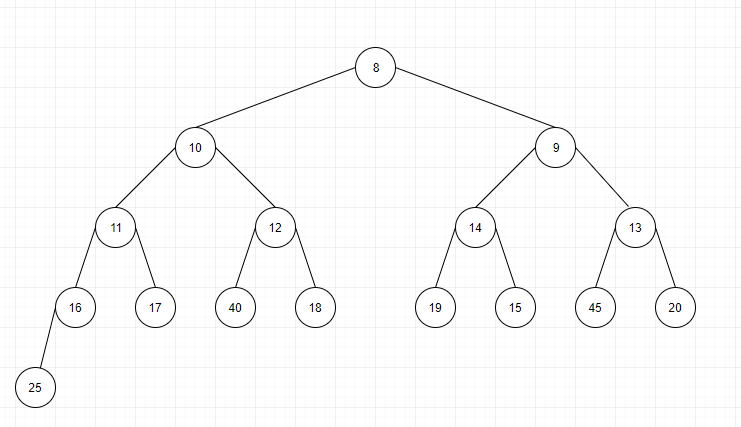
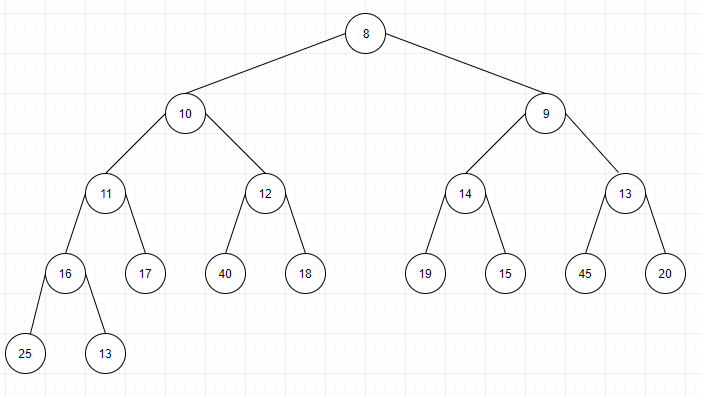
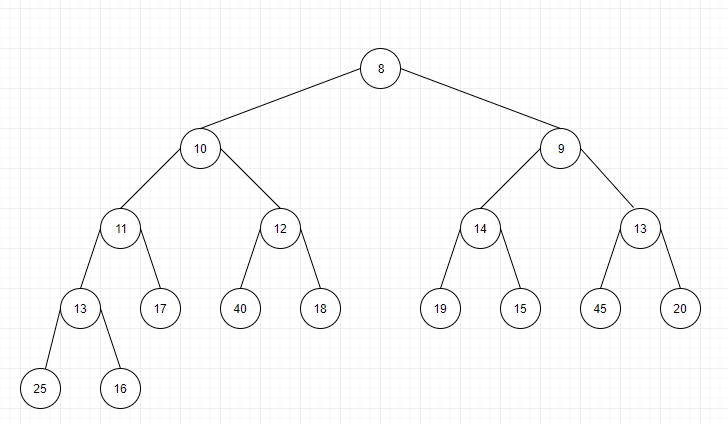
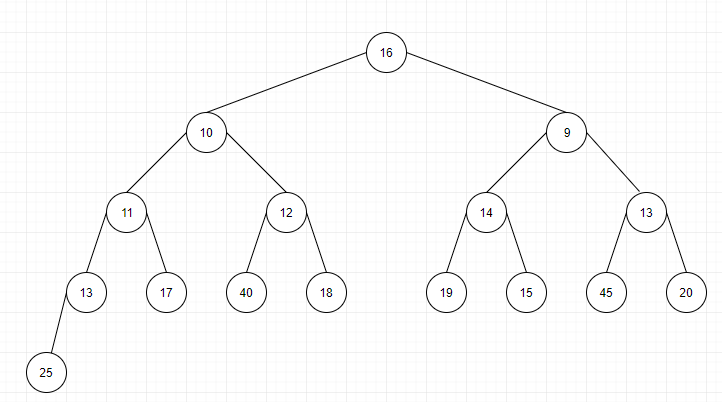
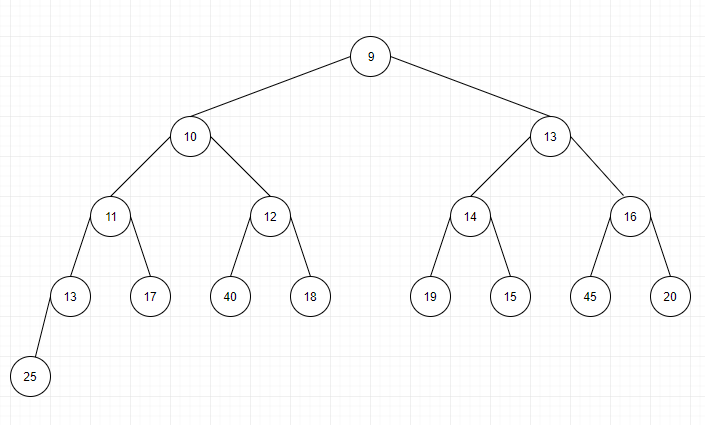
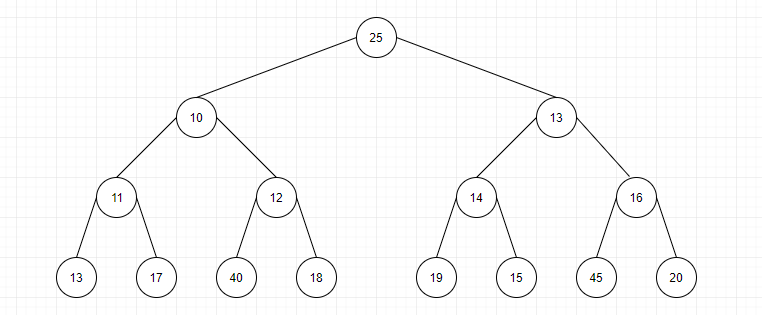


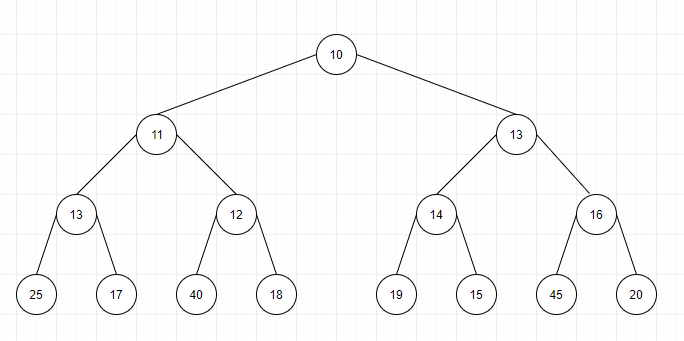
1. Insert 19



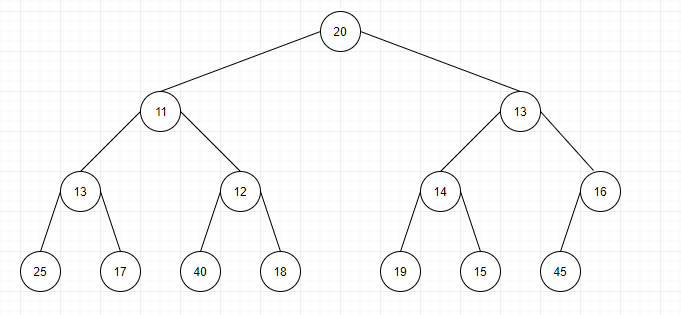
1. Insert 45

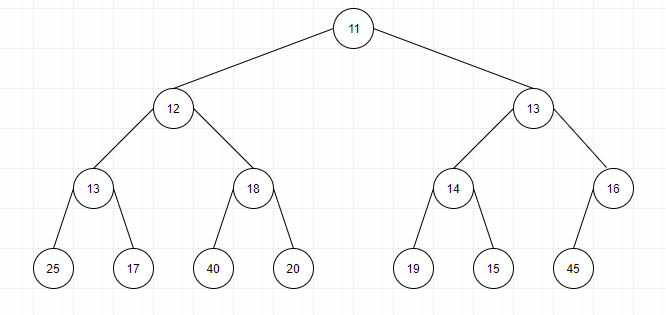


1. Insert 16 and swap  
     
   
2. Insert 12 and Swap  
     
     
   
3. Insert 8 and Swap  
     
   
4. Insert 18   
   
5. Insert 14 and swap  
     
   
6. Insert 13 and swap  
     
   
7. Insert 9 and swap  
     
   
8. Insert 20  
   
9. Insert 11 and Swap  
     
   
10. Insert 13 and swap  
      
    
11. RemoveMin and DownHeap  
      
    
12. Remove min and downheap  
    

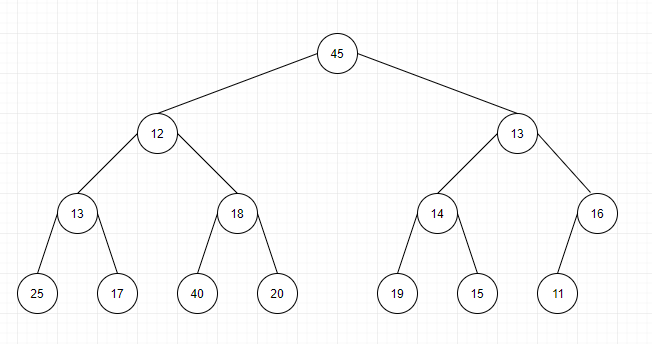
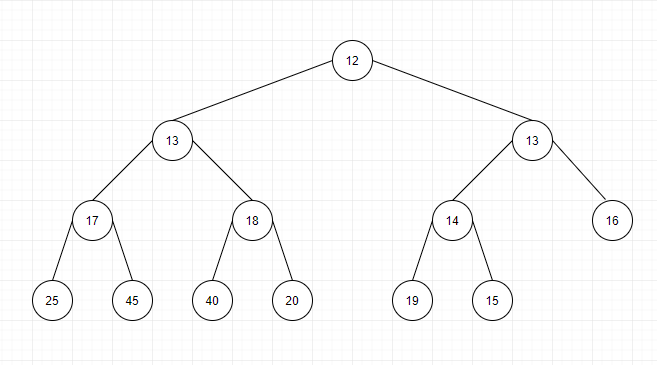


1. Remove Min and DownHeap

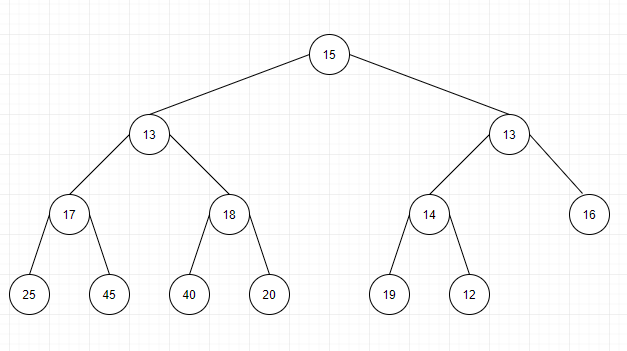


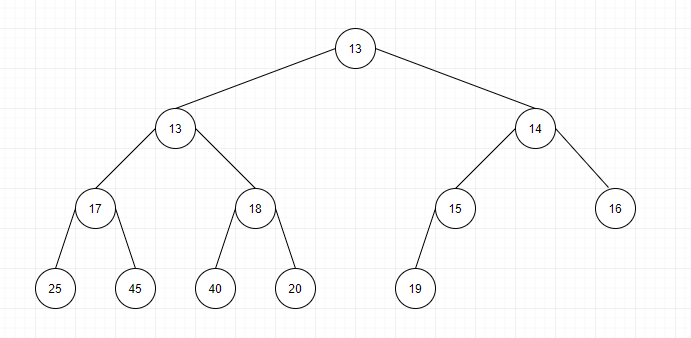


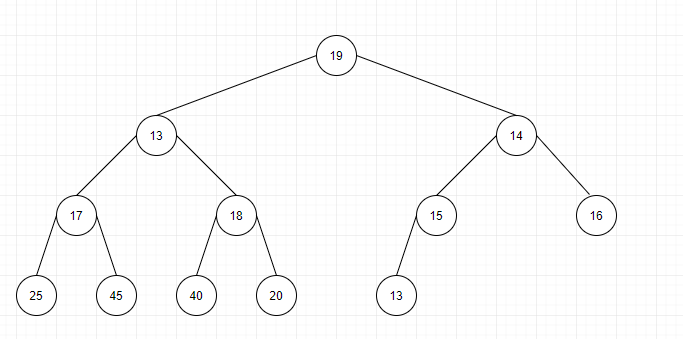
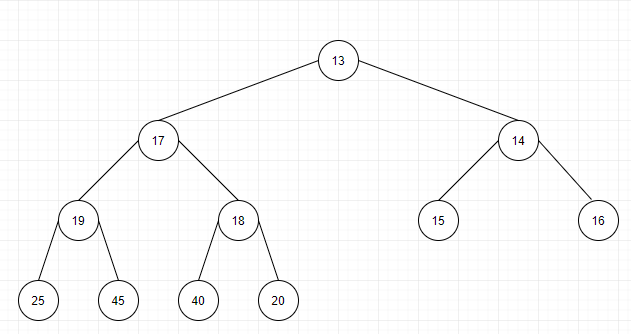
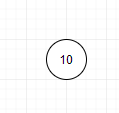
1. Remove Min and DownHeap

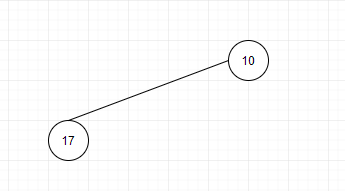
  


1. Remove Min and Swap

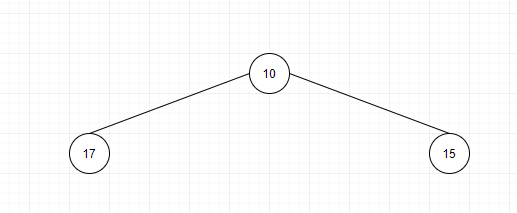


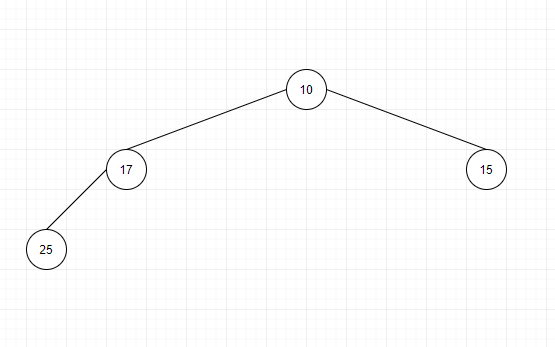
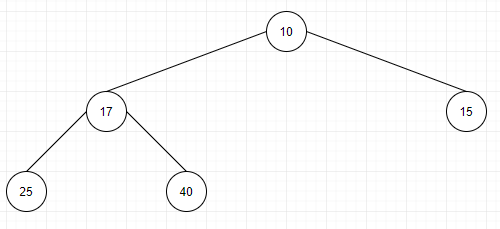


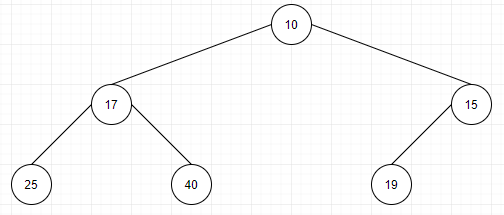
1. Remove Min and Swap / Final Tree  
     
   
2. Insert 10  
   
3. Insert 17

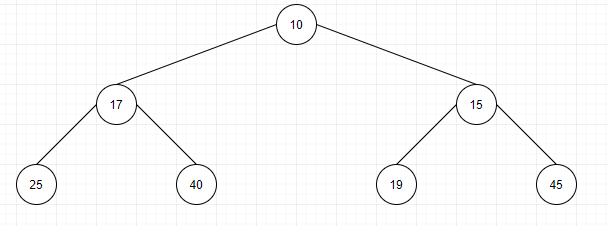
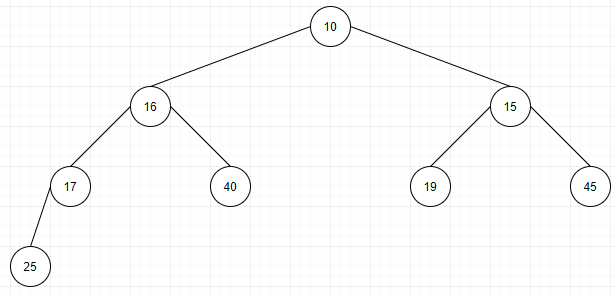
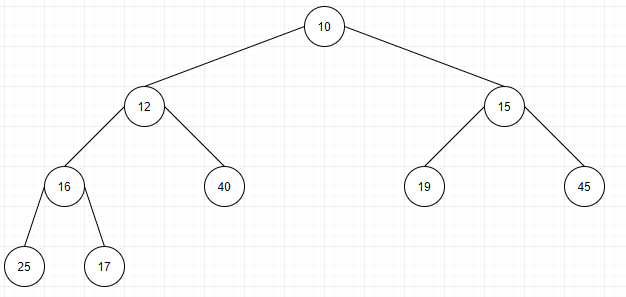
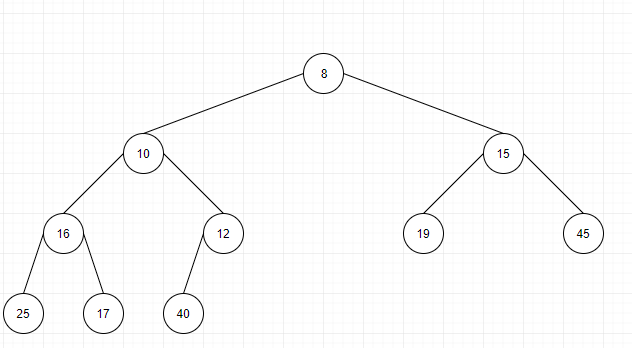
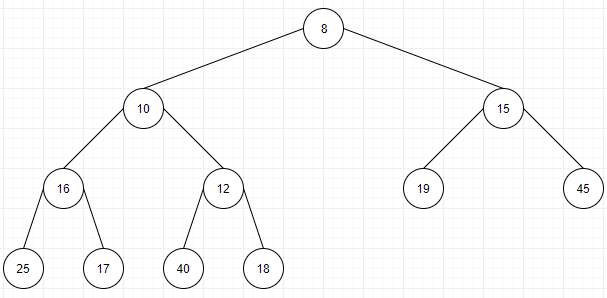


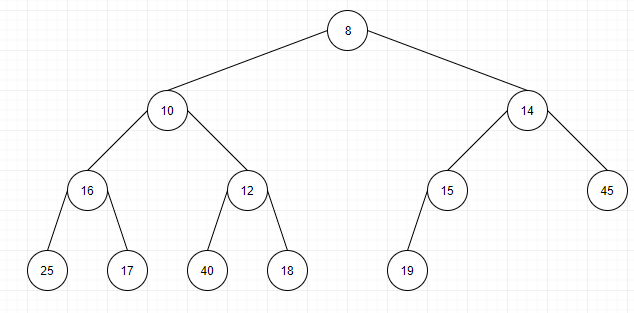
1. Insert 15

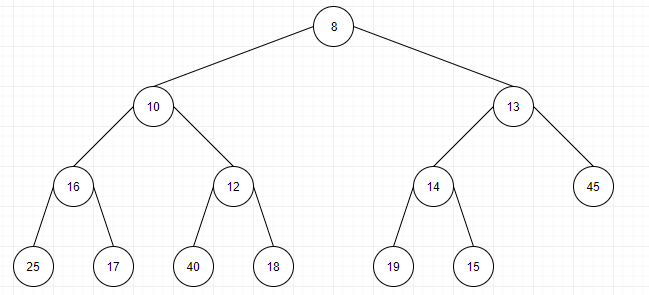
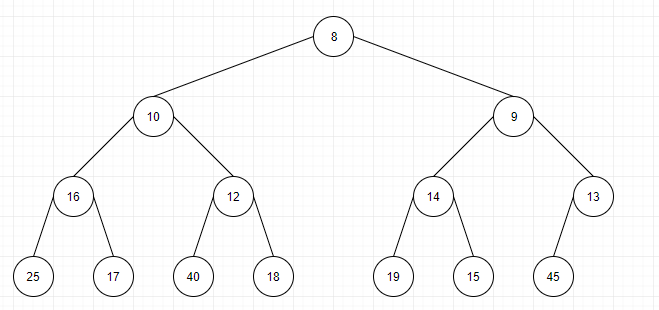
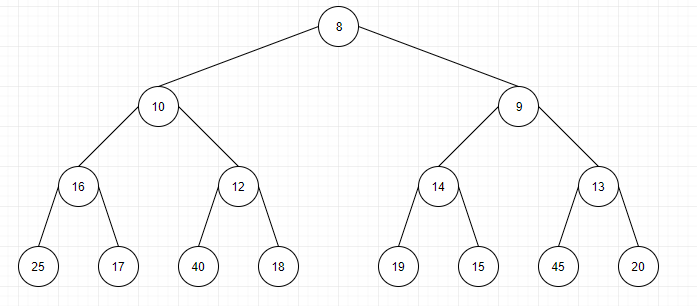
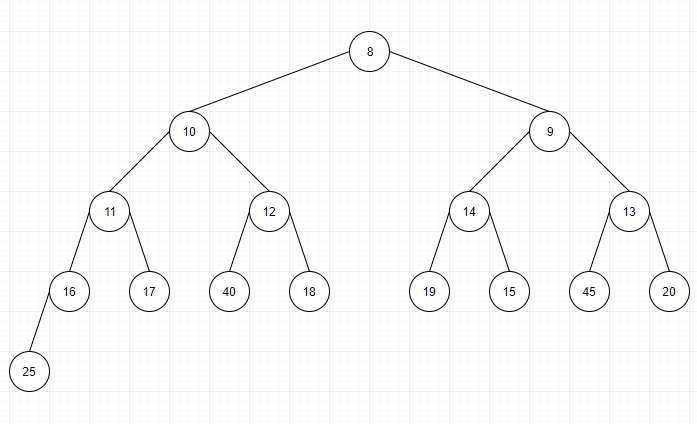


1. Insert 25  
   
2. Insert 40  
   
3. Insert 19



1. Insert 45  
   
2. Insert and Swap 16  
   
3. Insert and Swap 12  
   
4. Insert and swap 8  
   
5. Insert 18  
   
6. Insert and Swap 14



1. Insert and Swap 13  
   
2. Insert and Swap 9  
   
3. Insert 20  
   
4. Insert and swap 11  
   
5. Insert and Swap 13  
   