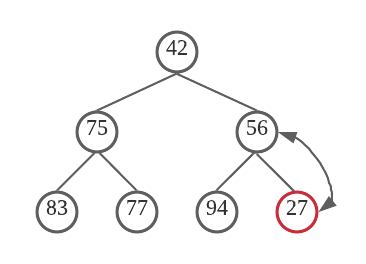
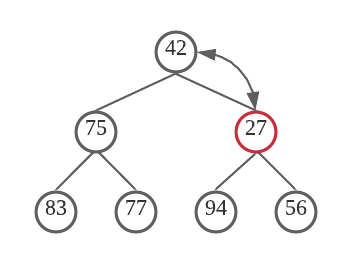
# Homework 4

## Problem 1

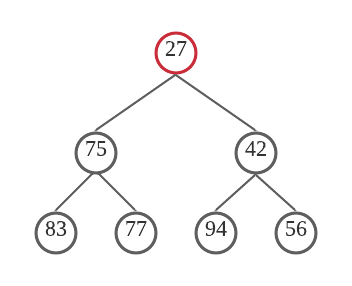
Step 1: add the new entry at the end of the heap, new entry (27) is added to the end of the heap.



Step 2: reorganize the heap, since 27 < 56, (27) is swapped with its parent (56)

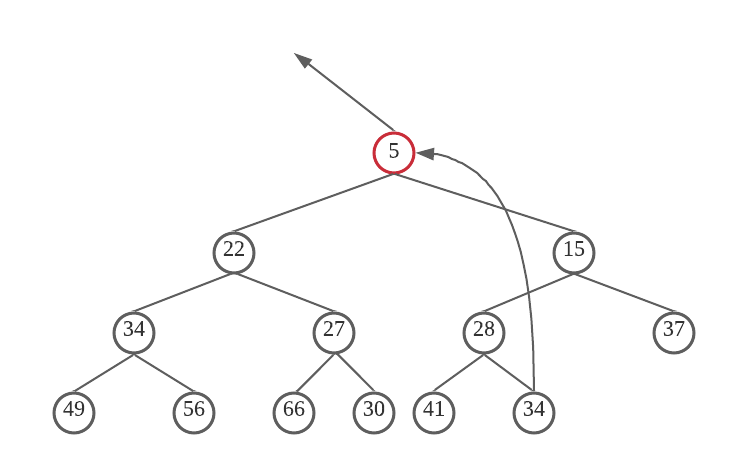


Since 27 < 42, (27) is swapped with its parent (42)

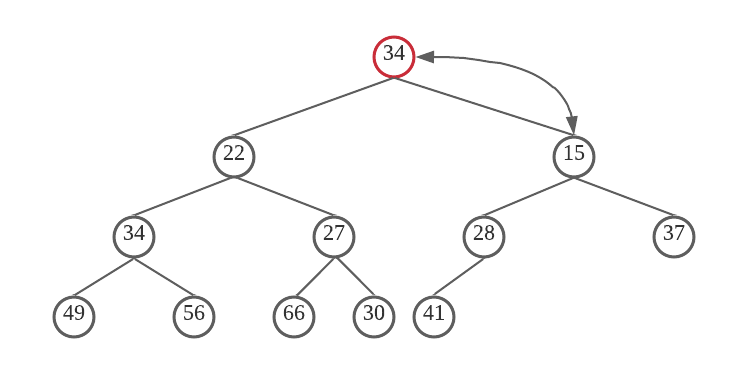


## Problem 2

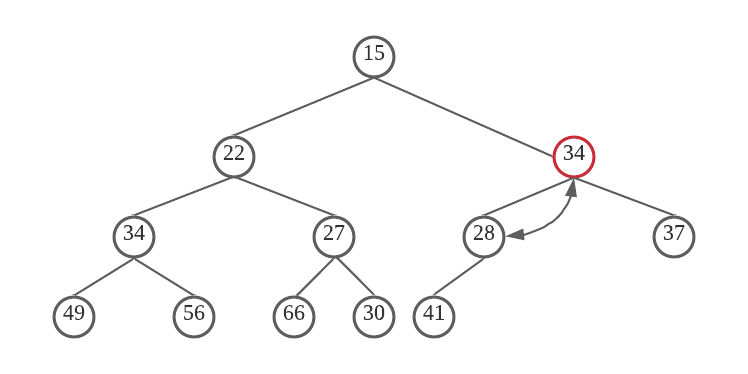
Step 1: remove the root, root entry (5) is removed, the last entry (34) is moved up to be the new root



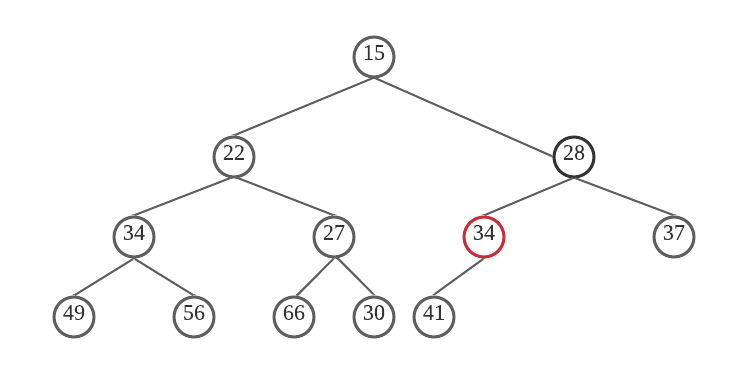
Step 2: last node is move up to the root and perform down-heap bubbling, right child has smaller key. So, (15) is swapped with (34).



Left child has smaller key. So, (28) is swapped with (34)

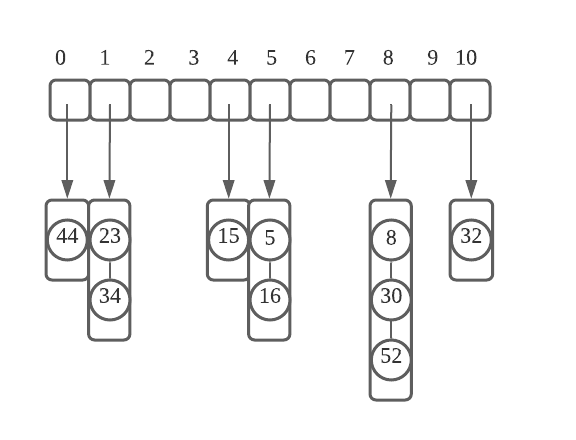


Final



## Problem 3

The compression function h(k) = k mod 11



## Problem 4

The compression function h(k) = k mod 11

*collision*

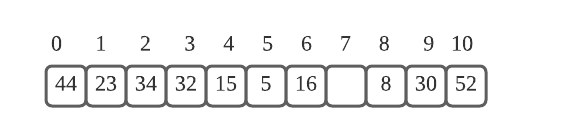
*collision*

*collision collision*

*collision collision*

*collision collision*

*collision*



## Problem 5

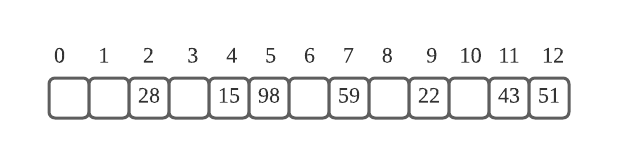
H(k) = k mod 13, h’(k) = 1 + (k mod 11), h(k, i) = (h(k) + I \* h’(k)) mod 13

Inserting k = 15, h(15) = 2, occupied

I = 1: (2 + 1 \* 5) mod 13 = 7, occupied

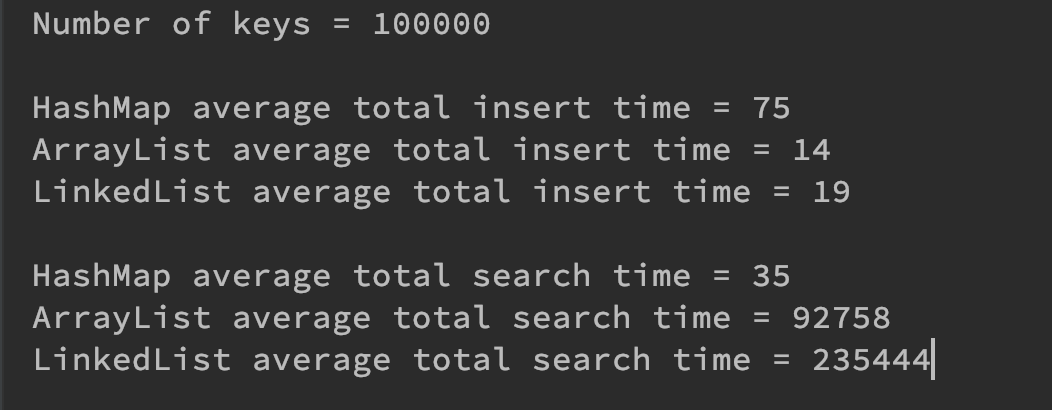
I = 2: (2 + 2 \* 5) mod 13 = 12, occupied

I = 3: (2 + 3 \* 5) mod 13 = 4, empty, store 15 here



## Problem 6

Result



So, we can find in the result above, that the HashMap has the largest insert time but it’s the fastest for searching. And the LinkedList is the fastest for insert number, but it’s the slowest for searching. Although, LinkedList is faster than the HashMap and ArrayList for insert, but the searching time is almost 6000 times of HashMap. By comparing the total time, we can find the HashMap can be used to store content with a large amount of data. When the numbers of element is not very larger, such as 100, these three data structures may has smaller different.