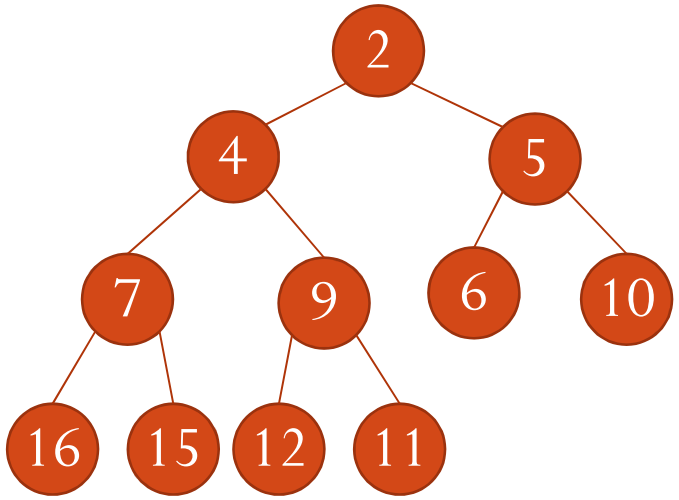
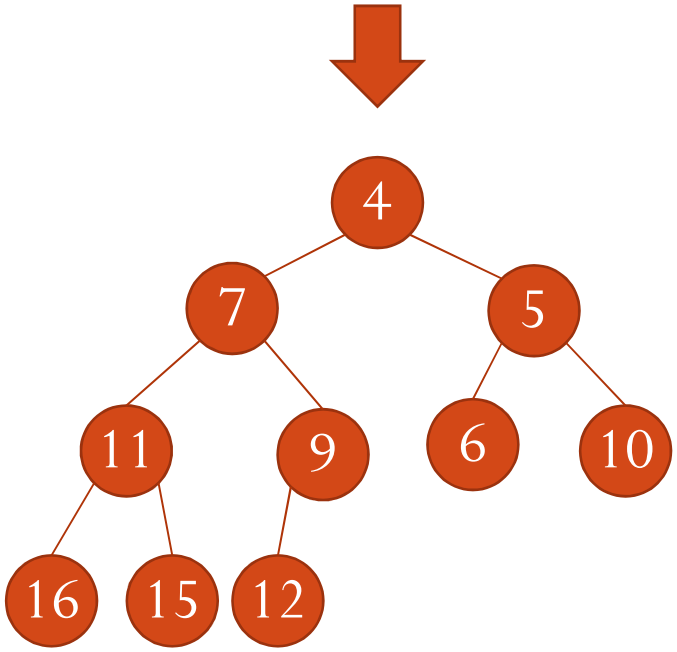
EXERCISE - DIJKSTRA

- Using priority queue (min-heap)
  - Complete binary tree
  - Min-Heap property: Key of each node is smaller than or equal to the key of its children
  - Operations:
    - deleteMin(): return the node having minimum key, remove this node from the min-heap
    - insert(k): insert a key into the min-heap
    - upHeap(u): adjust the heap (upward) when the key of a node u decreases
    - downHeap(u): adjust the heap (downward) when the key of a node u increases

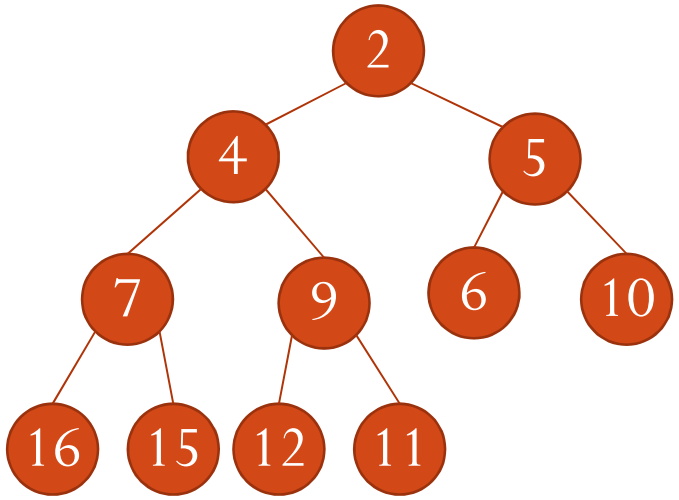
EXERCISE - DIJKSTRA



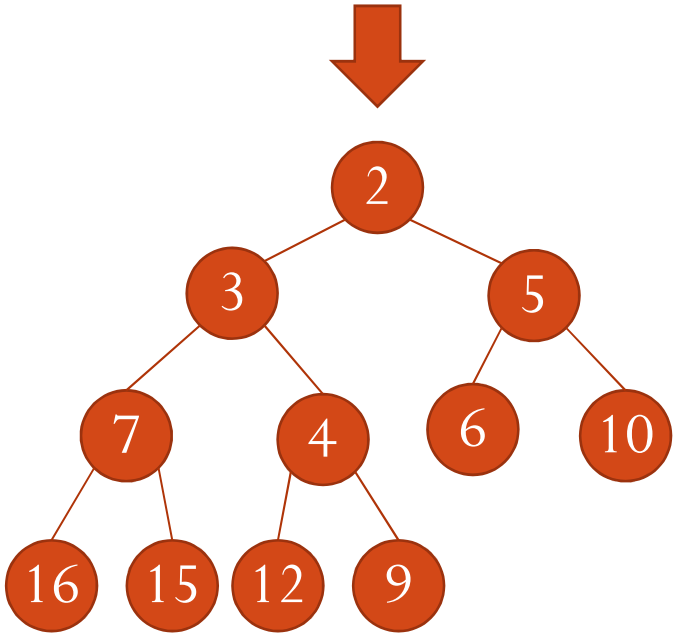
- deleteMin():
  1) swap(2,11)
  2) downHeap from 11 to recover Min-Heap property
  3) return 2



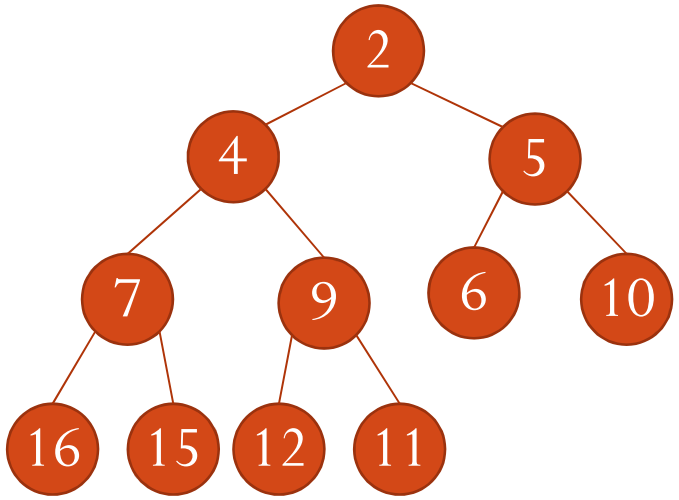
EXERCISE - DIJKSTRA



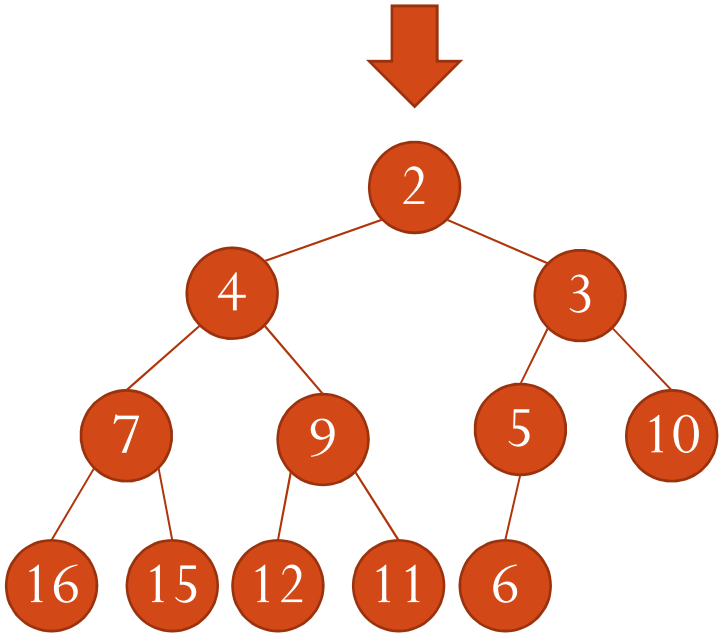
- decreaseKey(11,3):
  1) upHeap(3)



EXERCISE - DIJKSTRA



- insert(3):
  1) attach 3 as left-child of 6
  2) upHeap(3)



EXERCISE - DIJKSTRA

```
DIJKSTRA-HEAP(G = (V,E), s, t){
  initHeap(H);
  for v ∈ A(s) do  H.insert(v, c(s,v));
  while(not H.empty()){
    v = H.deleteMin(); fixed[v] = true;
    if (v = t) break;
    for x ∈ A(v)  do if (not  fixed[v]) then {
      if (not H.contains(x)) then insert(x, d[v] + c(v,x));
      else{
        if (d[x] > d[v] + c(v,x)) then{
          d[x] = d[v] + c(v,x); H.decrease(x, d[x]);
        }
      }
    }
  }
  output(d[t]);
}
```