1. Write the function \( \text{sum\_items}(T,d) \) that returns the sum of the items in a binary search tree with root \( T \) that have depth \( d \). Recall that the root has depth 0. For example, for the bst in the figure, \( \text{sum\_items}(T,0) \) would return 11, \( \text{sum\_items}(T,1) \) would return 24 and \( \text{sum\_items}(T,5) \) would return 0.

2. Write the function \( \text{max\_at\_d}(T,d) \) that returns the largest item stored in a b-tree with root \( T \) that has depth \( d \). For example, for the b-tree in the figure, \( \text{max\_at\_d}(T,0) \) would return 17, \( \text{max\_at\_d}(T,1) \) would return 34, \( \text{max\_at\_d}(T,2) \) would return 50 and \( \text{max\_at\_d}(T,3) \) would return \(-\text{math.inf}\).

3. Write the function \( \text{count\_items}(T,d) \) that returns the number of the items stored in a b-tree with root \( T \) that have depth \( d \). Recall that the root has depth 0. For example, for the b-tree in the figure, \( \text{count\_items}(T,0) \) would return 1, \( \text{count\_items}(T,1) \) would return 4 and \( \text{count\_items}(T,2) \) would return 15.
4. Write the function `checkHeap(H)` that returns true if `H` is a valid encoding of a max-heap and false otherwise.

5. Write the function `total_keys(H)` that returns the total number of items stored in a hash table `H` that solves collision by chaining.

6. Write the function `total_keys(H)` that returns the total number of items stored in a hash table `H` that solves collision by linear probing.

7. A singleton is a set that contains exactly one element. Write the function `is_singleton(S,i)` that returns true if `i` belongs to a singleton in the disjoint set forest encoded by `S` and false otherwise. For example, for the figure shown, `is_singleton(S,1)` should return true, and `is_singleton(S,2), is_singleton(S,4)` and `is_singleton(S,5)` should return false.
8. A disjoint set forest is compressed if every element is either a root or points to a root (i.e. all trees in the forest have height at most one). Write the function `is_compressed(S)` that returns true if `S` represents a compressed dsf and returns false otherwise. For example, for the figure `is_compressed(S)` should return false, since the path from 2 to its root has length 2.