Complexity:

The **worst case** complexity of the **find-max** operation on standard binary search trees of size N => O (N)

The **average case** complexity of the **find-min** operation on standard binary search trees of size N is => O(logN)

The **average case** complexity of the **find-max** operation on hash tables is => O(N)

The **average case** complexity of the **delete** operation on hash tables is => O(1)

The **average case** complexity of the **find-max** operation on AVL trees of size N is => O(log N)

The **average case** complexity of the **find-min** operation on AVL trees of size N is => O(log N)

The height of an AVL tree containing n nodes is O(n) in the worst case => false

The **average case** complexity of the **find** operation on linked lists of size N is => O(N)

The **worst case** complexity of the **delete** operation on linked lists of size N is => O(N)

In disjoint sets, when unions are done arbitrarily, the worst case complexity of the find operation => O(N)

Quadratic probing suffers from primary clustering => false

How much additional amount of space does a binary search tree of *n* elements require **in addition**to the space required to store the elements => O(N)

Amount of **additional** memory required by binary heaps is => O(1)

Heapsort can be implemented in a way that requires O(1) additional amount of memory => true

In Hash Tables, the complexity of rehashing is => O(N)

The complexity of the merge operation (not the merge sort algorithm) is => O(N)

The complexity of the partitioning step in Quicksort is => O(N)

What is the maximum height of a complete binary tree containing N nodes? => O(logN)

A binary heap can always be constructed out of a given array of items in O(N) time.  => t

In disjoint sets, what will be the worst case complexity of the find operation if the unions are done by size. => O(log N)

In Disjoint Sets, if unions are done arbitrarily, the **worst-case**complexity of the **union** operation will be => O(1)

In Disjoint Sets, if unions are done **by height**, the **worst-case**complexity of the **find** operation will be => O(log N)

In Disjoint Sets, **path compression** changes the algorithmic complexity of the **find** operation. => false

When the median element is always chosen as the pivot, Quicksort takes => O(n logn)

When the input is **sorted** and the first element is always chosen as the pivot, Quicksort takes => O( n^2 )

The median of a given sequence of unnecessarily sorted items can be found in linear time on average => true

In Shell’s increments, the pairs of increments are relatively prime => false

In Shell Sort, the choice of increment does not affect the complexity of the algorithm =>f

Hash Tables can store duplicated keys. => false

During rehashing, existing keys in the old hash table may map to different indices in the new table => true

The number of nodes in a complete binary tree of height *h*is always smaller than or equal to that of a perfect binary of the same height => true

Quadratic probing always grantees to find an empty cell if there is any present => false

T(0) = T(1) = 3 /  T(n) = T(n-1) + T(n-2) + 5 when n > 1 => exponentially

An algorithm, the resource function of which is 10n2 + 25n + 3, is also Θ(n2) => true

In separate chaining, the hash table is an array of linked lists => true

In linear probing there is no guarantee of finding an empty cell even if there is some => f

Every problem that is Ω(log100N) is also Ω(N2) => false

Every algorithm that is O(nn) is also O(n2) => false

Every algorithm that is θ(n) is also Ω(n) => true

logkn is O(n) for any constant k => true

In a perfect binary tree, leafs can be located at different depths => false

The height of a binary search tree may depend on the order in which elements are inserted => true