BSTS LIKE

The **worst-case** asymptotic time complexity to **search** an integer in a **binary search tree** of N integers is O(N).

TRUE

~~*The asymptotic time complexity to search an element in a binary search*~~

~~*tree with N nodes is O(logN)*~~

~~*FALSE*~~

The **worst-case** asymptotic time complexity to search an integer in a **red-black tree** of N integers is O(log N).

TRUE

The **master method** applies to calculate the asymptotic time complexity of **binary search**.

TRUE

~~v. Every binary search tree with n nodes has height O(log n).~~

~~FALSE~~

RECURRENCE

Consider the recurrence $T(n) = 3T(n=3)+\log n$. Its asymptotic complexity

is $T(n) = O(n)$.

BFS

Let T be a complete binary tree with n nodes. Assume we use breadth _rst search to _nd a path from the root to a given vertex v 2 T. The asymptotic runtime complexity of this search is O(log n).

FALSE

HEAPS

The worst-case asymptotic time complexity to search an integer in a max-heap of N integers is O(N log N)

TRUE

In a max-heap the depth of any two leaves differs by at most 1.

TRUE

Consider an unsorted array A[1:::n] of n integers. Building a max-heap out of the elements of A can be done asymptotically faster than building a red-black tree with the elements of A.

TRUE

The array A = [20; 15; 18; 7; 9; 5; 12; 3; 6; 2] represents a max-heap.

TRUE

SORTS

a)

«The bubble sort algorithm can be implemented using two nested while loops.»

→ for loops and while loops are interchangeable

TRUE

b)

«The insertion sort algorithm can be implemented using two nested for loops.»

→ for loops and while loops are interchangeable

TRUE

c)

~~«Given the same input, all three sorting algorithms always need the same number of comparisons.»~~

~~→ all algorithms will need a different number of comparisons in general (cf. lecture slides)~~

~~FALSE~~

d)

~~«All three sorting algorithms only compare two adjacent elements in an array.»~~

~~→ counter example: Both selection sort and insertion sort in general will compare elements at positions which are not next to eachother~~

~~FALSE~~

HASH TABLE

Assume chaining is used to resolve collisions for a hash table of size

m that stores N elements with unique keys. The worst-case asymptotic time

complexity to remove an item from the hash table is O(N).

TRUE

~~A hash table guarantees a constant lookup time.~~

~~FALSE~~

Consider an initially empty hash table of size M and hash function h(x) = x mod M. In the worst case what is the time-complexity to insert n keys into the table if chaining is used to resolve collisions. Assume that overow chains are implemented as unordered linked lists. Give a brief justi_cation for your answer.

O(n)

What is the answer for question (au dessus) if the overow lists are ordered? Give a brief justi_cation for your answer.

$O(n^2)$

Consider the same hash table and function as in task (a), but assume that collisions are resolved using linear probing, and n _ M
2 . In
the worst case what is the time complexity (in big O notation) to insert n keys into the hash table? Give a brief justi_cation for your answer.

$O(n^2)$

How big must the hash table be if we have 60000 items in a hash table that uses open addressing (linear probing) and we want a load factor of 0.75?
n/alpha = 60000/0.75 = 80 000

What is the expected number of comparisons to search for a
key if we must store 60000 items in a hash table that uses open addressing (linear probing) and we have a load factor of 0.75.

(1+(1/1-alpha))/2 = (1+(1/1-0.75))/2 = 2.5

GRAPHS

Let G = (V;E) be a weighted graph and let M be a minimum spanning tree of G. The path in M between a pair of vertices v1 and v2 does not have to be a shortest path in G.

TRUE

ARRAYS

Assume an array contains n numbers that are either -1, 0, or 1. Such an array can be sorted in O(n) time in the worst case.

TRUE

LISTS

In a doubly linked list with 10 nodes, we have to change 4 pointers of the list if we want to delete a node other than the head node and tail node.

FALSE