**Section 1: General Terminology**

**The following are questions about terms used in our course.**

1.What is your instructor's last name?

**FIEDLER**

2.Our course name is abbreviated to ADAA. What does ADAA stand for?

**ADVANCED DESIGN AND ANALYSIS OF ALGORITHMS**

3.We have looked at ADTs and data structures to analyze run-time. What does ADT stand for?

**ABSTRACT DATA TYPE**

4.Which Python IDE are we using for programming and homework submission?

**PYCHARM,** **JUPYTER**

5.Which JAVA IDE are we using for programming and homework submission?

**ECLIPSE**

**Section 2: Data**

**2. (4 points) The following are True/False questions about data in Computer Science. Mark either (for**

**True) or G (for False).**

1. An ADT defines the behavior of a data implementation from the point of the **programmer.**
2. An ADT defines the behavior of a data implementation from the point of the **user**.---**TRUE**

C) A data structure is a representation of the data (organization, storage, and management)

from the point of the **programme**r**.**

D)A data structure is a representation of the data (organization, storage, and management)

from the point of the **user**.--**FALSE**

E) A data structure is a collection of data values, the relationships among them, and the

Functions or operations that can be applied to the data.

F) An ADT is implementation dependent.---**FALSE**

G) An ADT can have several data structures associated with it.---**TRUE**

**3. (4 points) The following are questions about** **ADT / data structures. Mark either ADT or DS to indicate which one is applicable.**

ADT DS String

ADT DS. List ---------**ADT**

ADT DS. Integer----ADT

ADT DS. Scapegoat-Tree----DS

ADT DS. Dictionary

ADT DS. HASH TABLE

ADT DS Binary search tree----DS

ADT DS stack,QUEUE--------**ADT**

ADT DS LINKED LIST-------ADT

ADT DS. AVL TREE---------**DS**

**ADT DS. TREE----ADT**

**Section 3: Heaps and Trees**

**4. (4 points) A binary heap is a nearly complete binary tree filled on all levels except possibly the lowest level where leaves are pushed left-most. Heaps are often implemented as an array.**

a. Each node in a heap satisfies the heap property. What is the relationship between a node and its right child in a min-heap?

b. Which element is the minimum in a min-heap?

c. If a binary heap contains 28 elements (nodes), what is the height of the corresponding binary tree

d. How many leaves does a binary heap of size n have?

e. Each node in a heap satisfies the heap property. What is the name of a heap in which every node's value is smaller than the values of its children?----**MIN HEAP**

f. An array A that implements a heap is usually of length power-of-two. In that case, if a heap has size 26 elements, what is the (minimum) size of the corresponding array?------**32**

g. If a binary heap contains 50 elements (nodes), what is the height of the corresponding binary tree?

h. In order to implement the operations insert, delete, find using an array A that represents a heap, we need to keep track of two attributes associated with our data structure. What are the two attributes needed?---------ARRAY SIZE--------HEAP SIZE

i. If a heap is implemented in an array A, and the maximum in that heap is the first element of the array, what type of heap is it?

**5. (4 points) A 2-3 tree is a tree in which each non-root node which is not a leaf has 2 or 3 sons. The following are True/False questions about 2-3 trees. Mark either I (for True) or E (for False).**

a. Each node is labeled with the largest value in the middle subtree and the largest value in

the right subtree.------TRUE-------DOUBT

b. Each node is labeled with the smallest value in the left subtree and the largest value in

the middle subtree.-------FALSE-----DOUBT

c. Every path from the root to a leaf has the same length.----**TRUE**

d. Every path from the root to a leaf has length 2 or length 3.--- 2h to 3h

e. Data is ordered left-to-right.---**TRUE**

f. Data is stored only in leaves.-----**TRUE**

**6. (4 points) A binary search tree (BST) is a linked-node based binary tree which stores key-value pairs (or just keys) in each node. Left and right children are roots of left and right subtrees, respectively.**

The following are True/False questions about BSTs. Mark either (for True) or B (for False).

1. A BST with n nodes has height at most n.
2. Keys in a BST must be comparable. -----TRUE
3. Keys in a BST must be integers.
4. The minimum key in a BST is in the root.
5. BSTs form doubly linked lists.
6. All keys of nodes in the right subtree of a node N are smaller than the key of N.-----FALSE

**7. (4 points) A binary search tree (BST) is a linked-node based binary tree which stores key value pairs (or just keys) in each node. Left and right children are roots of left and right subtrees, respectively.**

**The following are True/False questions about BSTs. Mark either D (for True) or D (for False).**

1. The size of the left subtree must not differ by more than 1 from the size of the right subtree for any node in a BST.
2. The size of the left subtree must be larger than the size of the right subtree since nodes must be pushed left.
3. In-order walks provide the correct key order (smallest to largest) regardless of the tree balance.-----TRUE
4. post-order walks provide the correct key order regardless of the tree balance.
5. Post-order walks provide the reverse key order to a pre-order walk

f. Keys and values in a BST must be of the same type.

g. BSTs can take on the form of a linked list where each node has only one child.-----FALSE

h. All keys of nodes in the left subtree of a node N are smaller than the key of N.

I. The minimum in a BST can be found by following the left child pointers from root until we

encounter a leaf.-----TRUE

j. Keys in a BST must be comparable.-----TRUE

k. A BST with n nodes has height at least =log2(n)

**8. (4 points) A binary search tree (BST) is a linked-node based binary tree which stores key-value pairs (or just keys) in each node. Left and right children are roots of left and right subtrees, respectively.**

a. What is the relationship between the key of a parent and the key of its right child?(2points)

**PARENT.KEY<=RIGHT.KEY**

OR

KEY OF RIGHT CHILD >KEY.PARENT

b. What is the relationship between the key of a parent and the key of its left child?(2points)

**PARENT.KEY>=LEFT.KEY (OR) KEY OF LEFT CHILD SHOULD BE LESS THAN THE KEY OF A PARENT NODE**

**OR**

**KEY OF LEFT CHILD (NODE)<KEY.PARENT(NODE)**

c. Give a short description of an algorithm to find the predecessor of a node N (by key) after the

node I has been located. No programming on paper(2points)

d. Give a short description of an algorithm to find the successor of a node N (by key) after the node N has been located. No programming on paper! (2points)

**QUESTIONS\_NEW**

Q1). A binary search tree (BST) is a linked-node based binary tree which stores key-value pairs (or just keys) in each node. What is the relationship between the key of a node and the key of its **left** child?--------- **node.left.key < node.key**

Q2). A binary search tree (BST) is a linked-node based binary tree which stores key-value pairs (or just keys) in each node. What is the relationship between the key of a node and the key of its **right** child?--------- **node.right.key > node.key**

**Section 4: Self-Balancing Trees and Forests**

**9.** **(4 points) A self-balancing tree (forest) is a (collection of) search tree data structures) in which insert/delete operations may trigger a partial tree rebuild. Name four self-balancing search tree data structures we discussed in class or have been** **assigned as presentation topics**

a. 2-3 TREES

b .SCAPEGOAT TREES

C .AVL TREES

D .RED-BLACK TREES

**10.(4****points) Scapegoat trees which upon insert/delete operations rarely but expensively choose a scapegoat node and completely rebuild the subtree rooted at it into a complete tree. The following are True/False questions about** **Scapegoat trees Mark either OD (for True) or (for False).**

a. Scapegoat trees are binary search trees.-----**TRUE**

b. Scapegoat trees store the size of the whole tree in the root node.

c. Scapegoat trees store the size of the tree since the last rebuild in the root node.

d. Scapegoat trees store the weight of the subtree rooted at a node N in that node N.-----**FALSE**

e. Scapegoat trees store the height of the subtree rooted at a node N in that node N.------**FALSE**

**f.** Give n data values stored in n nodes,scapegoat trees have average performance of O(n) for lookup,insertion,and deletion of values.------**FALSE**

**11. (4 points)** **Scapegoat trees are search trees which upon insert/delete operations rarely but expensively choose a scapegoat node and completely rebuild the subtree rooted at it into a complete tree. The** **Following are True/False questions about Scapegoat trees. Mark either D (for True) or B (for False).**

a. If T is an a-weight-balanced binary search tree, then T is also a-height-balanced.---**-TRUE**

b. A measure of tree balance is the parameter a. For a Scapegoat tree, ½ <=a<=1.----**FALSE**

c. A measure of tree balance is the parameter a. For a Scapegoat tree, size(left(node)) <=a.size (node).----**TRUE**

d. If a partial tree rebuild is triggered by insertion of a deep node N, the scapegoat node is a

descendant of the node N.

e. If a partial tree rebuild is triggered by insertion of a deep node N, the scapegoat node is a

Ancestor of the node N.

**12. (4 points) A priority queue is a special type of queue in which each element is associated with a priority value. Elements are served on the basis of their priority. Higher priority elements are served** **first. Elements with the same priority are served according to their order in the queue. Priorities can be encoded with keys.**

Name two algorithms or applications for which priority queues are used.

**a.PRIMS ALGORITHM**

**b. dijkstra's algorithm**

Name two data structures that we looked at for implementation of a priority queue in class.

c.ADT STACK

d.DYNAMIC TABLE

**13. (4 points) Fibonacci heaps are a collection of trees. The following are True/False questions about**

**Fibonacci heaps. Mark either (for True) or F (for False).**

1. The roots of the trees in a Fibonacci heap are stored in a doubly linked list.-------TRUE
2. Children nodes in a Fibonacci heap are stored in a doubly linked list.-------TRUE
3. Nodes in a Fibonacci heap have parent pointers.
4. A node N in a Fibonacci heap has pointers to one of its children.
5. A node N in a Fibonacci heap has pointers to each of its children.
6. Fibonacci heaps consolidate trees after each DELETE\_ MIN operation
7. Fibonacci heaps consolidate trees after each INSERT\_ MIN operation-------FALSE

**Section 5: Amortized Analysis and Runtime Analysis**

**14. (4 points) Run-time analysis is an estimation of running time of an algorithm as a function of its input size (usually denoted as r). The following are four True/False** **questions about runtime analysis. Mark either M or true or false**

a. FIND/Search/Get in an array with n keys always has runtime of 0(1).

b. FIND/Search/Get in an array with n keys always has runtime of 0(n).---------TRUE

c. FIND/SEARCH/Get in a hash table with chaining and n keys always has runtime of 0(1).

d. FIND/SEARCH/GET in a BST with n nodes and height h always has runtime of O(h).

e. FIND/SEARCH/GET in a BST with n nodes and height h always has runtime of O(log n).---------TRUE

f. Find/Search/GET in a 2-3 tree with n nodes always has runtime of O(log(n))------------TRUE

g. In a BST with n nodes, the BST key property affords us to retrieve all data in order.-------FALSE

**15. (4 points) Amortized analysis is a method for analyzing an algorithm's complexity. The following are**

**Four True/False questions about amortization analysis. Mark either true or false**

1. Amortized analysis evaluates the average costs.------**TRUE**
2. Amortized analysis evaluates the worst-case cost-------**FALSE**
3. Amortization is used for the evaluation of one operation only, such as push, pop, or multipop----**FALSE**

d. Scapegoat trees achieve O(log(n)) amortized run-rime complexity for all operations

INSERT, DELETE, SEARCH.--------TRUE

e. DELETE\_ MIN in a Fibonacci heap has amortized runtime O(log(n)).-----------FALSE

f. Amortization is used for the evaluation of sequence of operations. -----**TRUE**

16. (4 points Name the three types of amortized analysis covered in class.

a.**AGGREGATE**

b.**POTENTIAL**

c**.ACCOUNTING**

**UNION-FIND AND GRAPH THEORY**

14. Union-Find data structures for a partition of a set provide operations for adding new sets, merging sets, and finding a representative member of a set.

A. The union operation merges all elements of the data structure.--------FALSE

B. The find operation returns a random element from the set it belongs to.--------FALSE

C All objects/elements may be represented by integers.--------TRUE

D. Union-Find data structures may be implemented using a list of trees.---------TRUE

15, A graph traversal is a systematic procedure for exploring a graph by examining all of its vertices and edges. Name (the generic names of two standard **graph traversal algorithms.**

ANSWER -----Breadth first seaseb

ANSWER-------DEPTH first search.

17.Name two improvements that can be used on **QuickUnion** and **QuickFind** (one each) to get a better worst-case run-time.

Your Answer:

**WEIGHTED QUICK UNION**

**PATH COMPRESSION**

18.Name FOUR applications for Union-Find structures.

Your Answer:

**FRIENDSHIP NETWORKS**

**IMAGE ANALYSIS**

**KRUSHKAL’S ALGORITHM**

**percolation theory**

19.Union-Find data structures may be implemented using one stack.-------FALSE

20. Union-Find data structures may be implemented using one singly-linked list.----TRUE

21. The union operation merges **all** elements of the data structure.--------FALSE

22. A Union-Find data structure stores a collection of disjoint sets.-------TRUE

**Name two standard structures to represent graphs (the vertex-edge relationships) in computer science.**

Your Answer:

Adjacency Matrix

Collection of adjacency lists

**GRAPH TEORITICAL ALGORITHMS**

General Applications By Field Physics and Chemistry

•**Crystallography (Automorphisms)**

**•Chemical Graph Theory (Shortest Path)**

**•Protein Folding (Automorphisms)**

**•Protein Classification (Knowledge Graph)**

16. A flow network can be modeled as a directed graph where each arc has a capacity and each arc receives a flow.

A. Network capacity is the maximum amount of flow that can be reliably transferred between different locations over a flow network.-----FALSE

B. A source has only incoming flow.--------TRUE

C. A sink has only incoming flow.---------TRUE

D. No flow may exceed the capacity along an arc or path.---------TRUE

E. If (u,v) is an arc then (v,u) is not an arc in a flow network.----TRUE

17.In a flow network, what is the name of a node that has only outgoing flow?-----source

18.In a flow network, what is the name of a node that has only incoming flow?---------sink

19. What is the name of a partition of the vertices of a flow network?------cut

20. A flow with the largest possible value must exist.------TRUE

21.Network capacity is the minimum amount of flow that can be reliably transferred between different locations over a flow network.-----FALSE

22. No flow may exceed the capacity along an edge or path.-----TRUE

23. The graph isomorphism problem is of polynomial complexity----FALSE

24. What is the name of the Python library we used for graphs?---IGRAPH

25. What is the graph theoretical term for "what comes in must go out"?----FLOW CONVSERVATION

26. There are no loops in a flow network.------FALSE

27. What is the name of a map *f(V)→V* from the vertices of a graph to itself such that edges get mapped to edges (incidence is preserved)?------------ automorphism

**C**

**HASH TABLES**

Q1.) Hash tables are an implementation of the ADT dictionary. They are arrays which utilize hash functions.

If we implement a hash table as an array of size **713**, what does the output of our hash function have to satisfy (in addition to being a hash function) in order for our program not to constantly crash?

Your Answer:

output should be lie between the range of array indexes

Q2). Hash tables utilize hash functions. Hash functions have certain properties.

A hash function has fixed **output** size.-----TRUE

Q3). Hash tables utilize hash functions. Hash functions have certain properties.

A hash function has fixed **input** size.--------FALSE

Q4). Hash tables utilize hash functions. Hash functions have certain properties.

A hash function has fixed **input** and **output** size.------FALSE

Q5). A binomial tree of order 0 is a single node.-----TRUE

Q6). The number of nodes in a binomial tree *Bk* of order k is *2k*.------TRUE

Q7). A binomial tree of order 𝑘 has a root whose children are binomial trees of order 𝑘−1,𝑘−2,…,2,1,0 (in that order).------------TRUE

**LOWER BOUNDS**

**1. (4 points) 0, ©, and are mathematical notations to describe the asymptotic complexity of an algorithm**.

Q1) If we have an algorithm A for a problem P and this algorithm is of complexity 0 (n)

then the problem P is of complexity O (n).---------FALSE

Q2). If we have an algorithm A for a problem P and this algorithm is of complexity 0 (n)

then the problem P is of complexity at least 0 (n).-------FALSE

Q3).If we have an algorithm A for a problem P and this algorithm is of complexity O(n)

then the problem P is of complexity at most 0 (n).--------TRUE

Q4). The complexity of an algorithm A which solves a problem P is an upper bound on the complexity of P.--------TRUE

**2. (4 points) A decision tree is a full binary tree that represents the comparisons between elements that are performed by a particular sorting algorithm operating on an input of a given size.**

Q1).Internal nodes of a decision tree are labeled with a query.------TRUE

Q2).Each leaf in a decision tree is labeled with a query.---------FALSE

Q3).The length of the longest simple path from the root of a decision tree to any of its reachable leaves represents the worst-case number of comparisons that the corresponding sorting algorithm performs.--------TRUE

Q4). Any comparison sort algorithm requires N(n logn) comparisons in the worst case.-----TRUE

**QUESTIONS LOWER BOUND AND UPPER BOUND**

Q1). The complexity of an algorithm 𝐴 which solves a problem 𝑃 is an **upper bound** on the complexity of 𝑃.-------TRUE

Q2). The complexity of an algorithm 𝐴 which solves a problem 𝑃 is a **lower bound** on the complexity of 𝑃.---------FALSE

Q3). Any comparison sort algorithm requires 𝛺(𝑛log𝑛) comparisons in the worst case.

--------TRUE

Q4). Θ provides an asymptotic upper bound.-------FALSE

Q5). The length of the longest simple path from the root of a decision tree to any of its reachable leaves represents the worst-case number of comparisons that the corresponding sorting algorithm performs.----------TRUE

Q6). 𝑂, Θ, and Ω are mathematical notations to describe the asymptotic complexity of an algorithm.---------TRUE

Q7). Θ provides a tight bound.-------TRUE

Q8). 𝑂 provides an asymptotic upper bound.-------TRUE