CSc 345 Final

Name	Key
UA email	

General Instructions.

- Put your name and email in the spaces provided.
- For all questions, read the instructions carefully and write legibly. Illegible answers will not receive credit. Make sure you use a pen or dark pencil.
- Do not tear any pages from the exam.
- Assume that all logarithms are base-2 unless specifically indicated otherwise.
- Please read all multiple choice answers carefully.
- If you need extra space, you can use the back of the exam, but make sure you make a note in the question that you used the extra space. Also, make sure the work is labeled with the appropriate Section and Question number.
- Unless otherwise noted, sorting is in ascending order.
- Unless otherwise noted, all algorithms are as presented in class.
- Unless otherwise noted, linked lists do not have tail pointers.

Some formulas that might be useful:

Summations:

$$\sum_{k=1}^{N} k = \frac{N(N+1)}{2}$$

$$\sum_{k=0}^{N} r^k = \frac{r^{N+1}-1}{r-1},\ r \neq 1$$

The Master Theorem: Given a function of the form $f(N) = af(N/b) + cN^d$, $f(N)$ is

- $O(N^d)$ if $a < b^d$
- $O(N^d logN)$ if $a = b^d$
- $O(N^{log_b a})$ if $a > b^d$

Other: $log(N!)\ is\ O(NlogN)$

Part 1. Multiple Choice. (4 points each) Circle the best answer for each question.

1. There is a flaw in the reasoning of the following proof by induction. On which line is the flaw located?

Conjecture: $d^n = 1$, where d is a non–zero real number, and n is a non–negative integer.

```
1  Proof (strong induction):
2      Basis Step:  Let n = 0.  d^0 = 1.
3      Inductive Step: If d^i = 1 ∀i such that 0 ≤ i ≤ k, then d^(k+1) = 1.
4
5          d^(k+1)
6          = d^(k+2) / d^1
7          = (d^k · d^k) / d^1
8          = (1 · 1) / 1
9          = 1
10
11     Therefore, d^n = 1, where d is a non--zero real number,
12                and n is a non--negative integer.
```

A. Line 2 B. Line 3 C. Line 6 D. Line 7

Questions 2 & 3 refer to the pseudocode below.
~~**You can assume that *n* is a positive power of 2.**~~

```
sum = 0
for i from 1 to n:
    for j from 1 to i:
        sum = sum + 1
```

$1 + 2 + 3 + \ldots + n$

2. What is the final value of *sum*?

A. $n - 1$ B. $n(n-1)/2$ C. $n(n+1)/2$ D. n^2

3. What is the runtime of this pseudocode in big-Theta notation?

A. $\theta(log_2 n)$ B. $\theta(n)$ C. $\theta(nlog_2 n)$ D. $\theta(n^2)$

Questions 4 & 5 refer to the pseudocode below.
You can assume that *n* is a positive power of 2.

```
sum = 0
while n ≥ 1:
    for j from 1 to n:
        sum = sum + 1
    n = n/2
```

$n + n/2 + n/4 + \ldots + 1$

$= \sum_{k=0}^{\log n} 2^k = 2^{\log n + 1} - 1 = 2n - 1$

4. What is the final value of *sum*?

A. $n - 1$ B. $2n - 1$ C. $log_2 n$ D. $n + log_2 n$

5. What is the runtime of this pseudocode in big-Theta notation?

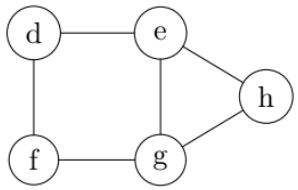
A. $\theta(log_2 n)$ B. $\theta(n)$ C. $\theta(nlog_2 n)$ D. $\theta(n^2)$

Questions 6-9 refer to the Anonymous Algorithm shown below, in pseudocode.
Input to the algorithm is an undirected graph $G = (V, E)$, a source vertex s, and G's representation using an adjacency list $adjList[]$. Other data structures uses are arrays $a[]$, $b[]$, and $c[]$, and an initially empty vertex queue Q. A sample graph and its adjacency list representation are provided for your convenience.

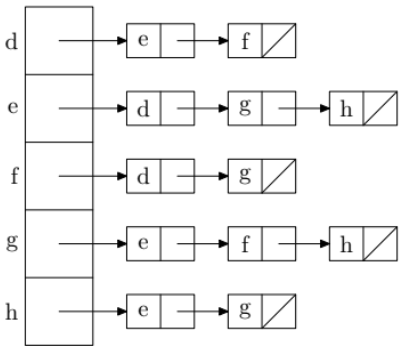
ANONYMOUS ALGORITHM

```
1   for each v ∈ V:
2     a[v] ← 'W'
3     b[v] ← ∞
4     c[v] ← nil
5   a[s] ← 'G'
6   b[s] ← 0
7   Q.enqueue(s)
8   while Q.size() ≠ 0:
9     v ← Q.dequeue()
10    for each n in adjList[v]:
11      if a[n] holds 'W':
12        a[n] ← 'G'
13        b[n] ← b[v] + 1
14        c[n] ← v
15        Q.enqueue(n)
16    a[v] ← 'B'
```

A sample graph:



And its adjacency list representation:



6. After the algorithm has been executed, what information about **G** does the array **b[]** contain?
A. The path distances from **s** to each vertex
B. The number of adjacent vertices each vertex has in the graph
C. The vertex from which each vertex was reached
D. A count of how many of each vertex's adjacent vertices were examined

7. After the algorithm has been executed, what information about **G** does the array **c[]** contain?
A. The path distances from **s** to each vertex
B. For each edge **(x, y)**, the vertex (**x** or **y**) that was examined last
C. For each vertex, the vertex that is most strongly connected to it
D. The vertex from which each vertex was reached

8. What is the actual name of this algorithm?
A. Breadth-First Search (BFS)
B. Depth-First Search (DFS)
C. Prim's Minimal Cost Spanning Tree Algorithm
D. Kruskal's Minimal Cost Spanning Tree Algorithm

9. What is the theta approximation of the runtime of this algorithm?

A. $\theta(|V|^2)$ B. $\theta(|V| + |E|)$ C. $\theta(|E|log_2|V|)$ D. $\theta(|E| + |V|log_2|V|)$

10. In which order are the numbers 1 to 5 printed by the pseudocode below?

```
S := an empty Stack
Q := an empty Queue
for i from 1 to 5:
    if(i % 2 == 0)
        S.push(i)
    else
        Q.enqueue(i)
end for
while Q is not empty:
    print Q.dequeue()
end while
while S is not empty:
    print S.pop()
end while
```

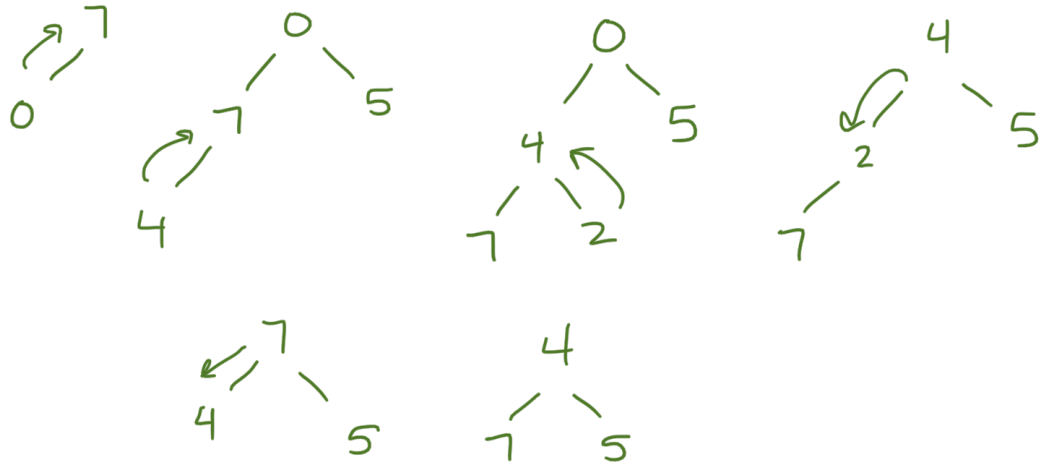
S: 2 4
Q: 1 3 5

1 3 5 4 2

A. 1, 2, 3, 4, 5 B. 5, 4, 3, 2, 1 C. 1, 3, 5, 4, 2 D. 5, 3, 1, 4, 2 E. 1, 2, 4, 3, 5

11. What is the in-order traversal of a min binary heap after the following operations?
`insert 7, insert 0, insert 5, insert 4, insert 2, delMin, delMin`

A. 4, 5, 7 B. 5, 4, 7 C. 4, 7, 5 D. 5, 7, 4 E. Answer not listed.



Questions 12 & 13 refer to the information below.
Recall that a full binary tree is a tree whose internal nodes must have exactly 2 children. Also, recall the definitions given below.
Let T be a full binary tree and let $i(T)$ be the number of internal nodes in T and $e(T)$ be the number of external nodes in T.

- If T is only made up of the root, then $i(T) = 0$ and $e(T) = 1$.
- If T is the tree made by joining two smaller full binary trees together (T_1 and T_2) by making each one a child of a new root, then $i(T) = i(T_1) + i(T_2) + 1$ and $e(T) = e(T_1) + e(T_2)$.

Consider the following proof by structural induction that for any full binary tree T, $e(T) = i(T) + 1$.

1 **Basis Step.**
2 Let T_0 be a full binary tree with just the root. Then
3 $e(T_0) = 1 = 0 + 1 = i(T_0) + 1$.
4 **Inductive Step.**
5 Let T_1 and T_2 be full binary trees and let T_3 be the full binary tree
6 formed by making each of the two trees a child of a new root. Assume
7 as the inductive hypothesis (IH) that $e(T_1) = i(T_1) + 1$ and
8 $e(T_2) = i(T_2) + 1$.
9 $e(T_3) = e(T_1) + e(T_2)$
10 $= i(T_1) + 1 + i(T_2) + 1$
11 $= i(T_3) + 1$

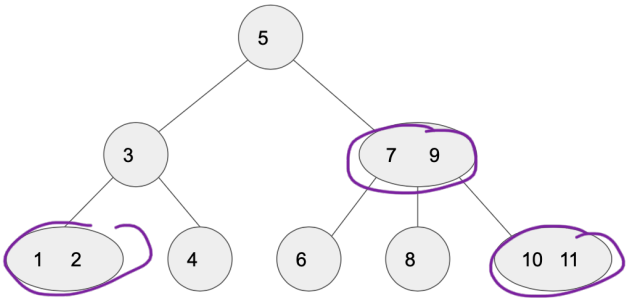
12. Which of the following statements is true about this proof?
A. The proof is correct.
B. The proof is incorrect because there is an error in the Basis Step.
C. The proof is incorrect because there is an error in the Inductive Hypothesis.
D. The proof is incorrect because the inductive hypothesis is applied incorrectly.
E. The proof is incorrect because of an error in the algebra or application of definitions.

13. In which line(s) is the inductive hypothesis applied?
A. Lines 2 & 3 B. Lines 5-7 C. Line 9 D. Line 10 E. Answer not listed.

14. Given that $f(n)$ *is* $O(g(n))$ and $h(n)$ *is* $O(g(n))$, which of the following statements *must be true*?
A. $f(n)$ *is* $\theta(h(n))$
B. $f(n)$ *is* $O(h(n))$
C. $f(n)$ *is* $\Omega(h(n))$
D. All of the above.
E. None of the above.

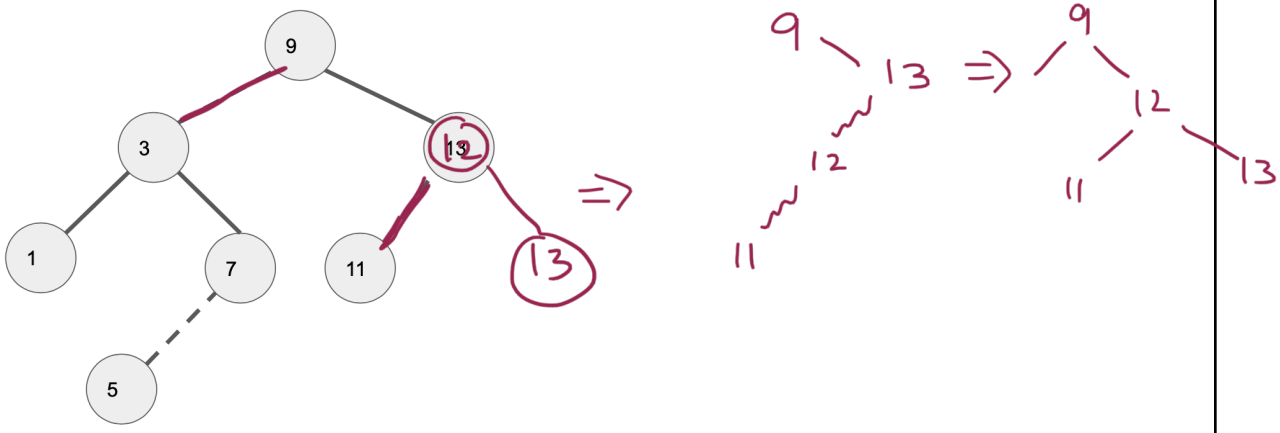
g h
f g h
h

15. The following is a 2-3 tree. If this tree is implemented as a LLRB tree, how many red nodes would it have?



A. 0 B. 1 C. 2 D. 3 E. More than 3

16. The following is a left-leaning red-black tree. The red links are shown with a dotted line, while the black links are shown with a solid line.



How many red nodes will there be in the tree after inserting 12 into the tree?

A. 0 B. 1 C. 2 D. 3 E. More than 3.

17. Below is a directed graph given as an adjacency list. How many topological orderings does the graph have?

0	3, 4
1	0, 2
2	3, 4
3	1, 4
4	

4

A. 0 B. 1 C. 2 D. 3 E. More than 3.

Questions 18-20 refer to the pseudocode below.

```
1   Q = an empty min priority queue that holds integers
2   for i from 1 to N:
3       Q.insert(N-i)
4   end for
5   while !Q.isEmpty():
6       Q.delMin()
7   end while
```

18. If *Q* is implemented as an unsorted array (of size *N*), what is the runtime of the pseudocode?

A. $O(logN)$ B. $O(N)$ C. $O(NlogN)$ D. $O(N^2)$ E. The answer is not listed.

19. If *Q* is implemented as a sorted array (of size *N*), what is the runtime of the pseudocode? Note that the sorting here would be in descending order so that the minimum would always be farthest to the right.

A. $O(logN)$ B. $O(N)$ C. $O(NlogN)$ D. $O(N^2)$ E. The answer is not listed.

20. If *Q* is implemented as a min binary heap, what is the runtime of the pseudocode?

A. $O(logN)$ B. $O(N)$ C. $O(NlogN)$ D. $O(N^2)$ E. The answer is not listed.

21. Let *A* be an array of integers containing the integers from 1 to 100 in reverse order. (i.e. [100, 99, 98,...,3, 2, 1]. How many swaps are required to sort *A* using Selection Sort?

A. 50 B. 99 C. 100 D. 99(100)/2 = 4950 E. 99(100) = 9900

22. What is the expected runtime of the *put* operation in a hashtable that uses separate chaining to handle collisions if the size of the table is *M* and there are *N* elements in the table? You can assume that the hashtable is implemented well and that it uses a good hash function.

A. $O(1)$ B. $O(N/M)$ C. $O(M/N)$ D. $O(M)$ E. $O(N)$

23. Consider a binary search tree that contains the integers 1 to 7. Which of the following insert orders will result in the shortest tree?

A. 1, 2, 3, 4, 5, 6, 7
B. 7, 6, 5, 4, 3, 2, 1
C. 7, 1, 6, 2, 5, 3, 4
D. 4, 2, 6, 1, 3, 5, 7
E. 1, 3, 2, 4, 5, 7, 6

Multi-Select. For each question, circle all the correct answers.

1. (10 points) Which of the following structures provide an O(logN) (or better) worst-case guarantee for finding (but not removing) the maximum?

A. A sorted array
B. A sorted linked list
C. A hashtable
D. A max binary heap
E. A min binary heap
F. An unsorted array
G. An unsorted linked list
I. A binary search tree
J. A skiplist
K. A red-black tree

2. (10 points) Which of the following structures provide an O(logN) (or better) expected runtime for finding (but not removing) the maximum given that the data is random and the structure is implemented well?

A. A sorted array
B. A sorted linked list
C. A hashtable
D. A max binary heap
E. A min binary heap
F. An unsorted array
G. An unsorted linked list
I. A binary search tree
J. A skiplist
K. A red-black tree

3. (10 points) Which of the following structures provide an O(logN) (or better) worst-case guarantee for finding (but not removing) an arbitrary element?

A. A sorted array
B. A sorted linked list
C. A hashtable
D. A max binary heap
E. A min binary heap
F. An unsorted array
G. An unsorted linked list
I. A binary search tree
J. A skiplist
K. A red-black tree

4. (10 points) Which of the following structures provide an O(logN) (or better) expected

runtime for finding (but not removing) an arbitrary element given that the data is random and the structure is implemented well?

A. A sorted array
B. A sorted linked list
C. A hashtable
D. A max binary heap
E. A min binary heap
F. An unsorted array
G. An unsorted linked list
I. A binary search tree
J. A skiplist
K. A red-black tree

5. (8 points) Let **G = (V, E)** be a simple, connected, undirected graph with unique edge weights. Assume that $|V| > 3$ and $|E| > 2$. Let the edge weights be named so that $weight(e_1) < weight(e_2) < \ldots < weight(e_m)$. Note that *m* is the number of edges. Also, let **T** be a minimum spanning tree of **G**.
Circle all of the statements below that **must be true**.

A. **T** is unique.
B. e_1 is in **T**.
C. e_2 is in **T**.
D. e_3 is in **T**.
E. e_m might be in **T**. (and might not be in **T**)
F. Let **P** be the shortest path from vertex *u* to vertex *v* in **G**. **P** must be in **T**.
G. Given all the edges that are incident on a vertex *v*, let e_{v-max} be the one with the greatest weight. Then e_{v-max} cannot be in **T**.
H. Given all the edges that are incident on a vertex *v*, let e_{v-min} be the one with the smallest weight. Then e_{v-min} must be in **T**.

6. (6 points) The following is an adjacency matrix for a directed graph **G***. **G*** is the transitive

closure of another directed graph **G**. Which of the following statements **must be true** about **G**? (The vertices of the graph are 0, 1, 2, 3, 4, 5 and correspond to the indices of the matrix.

	0	1	2	3	4	5
0	1	1	0	1	1	1
1	1	1	0	1	1	1
2	1	1	1	1	1	1
3	1	1	0	1	1	1
4	0	0	0	0	0	0
5	1	1	0	1	1	1

A. **G** can be topologically sorted.
B. **G** is simple.
C. **G** is strongly connected.
D. {0, 1, 3, 5} is a strongly connected component of **G**.
E. {0, 1, 2, 3, 5} is a strongly connected component of **G**.
F. {0, 1, 3, 4, 5} is a strongly connected component of **G**.

Written Response A. Both of these questions will be graded.

1. (4 points) Consider the following algorithm.

```
1  Algorithm mystery
2  Input: A weighted, undirected graph G = (V, E)
3  Output: A minimum spanning tree of G
4  H = a copy of G
5  Q = a max priority queue that holds edges where the key is the weight
6  insert all of H's edges into Q
7  while H has more than |V|-1 edges:
8        maxEdge = Q.delMax()
9        if _____________________:
10             remove maxEdge from H
11       end if
12 end while
13 return H
```

What should go in the blank at line 9?

maxEdge is not a bridge

2. (10 points) Alice is trying to compute the shortest path tree from a vertex *s* to all the other vertices in a weighted, undirected graph, but she has a problem: some of the edge weights

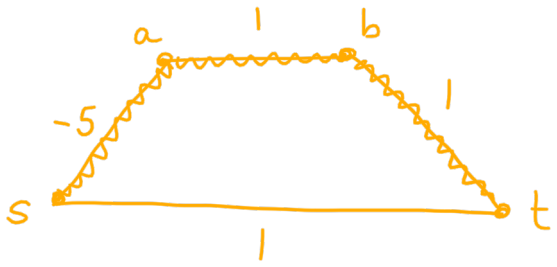
are negative and she knows that Dijkstra's Algorithm will not work if there are any negative edge weights. So she proposes the following solution:

- Determine the weight of the smallest edge. Call it *w*.
- If
- $w < 0$, add $|w|$ to all the edge weights in the graph.
- Run Dijkstra's Algorithm on the altered graph starting at *s*.

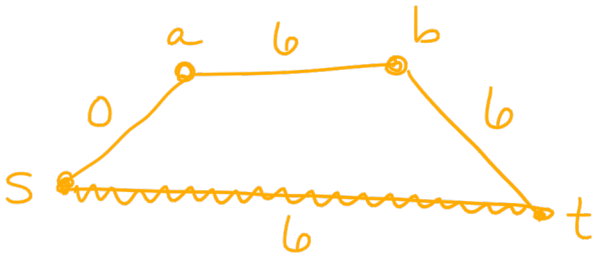
Alice claims that the shortest path tree she gets with the altered graph will be the same as the shortest path tree on the original graph.

Is Alice correct? Does her proposed algorithm work? Justify your answer.

No. Alice is not correct.
Here is a counterexample:



shortest path from s to t: s-a-b-t



shortest path from s to t: s-t

Written Response B. (40 points) Answer *two* of the four following questions. *Only two will be graded.* If you answer more than two, the first two you answer will be graded.

1. In this course, we discussed two different implementations of a Stack: one based on an array and one based on a linked list. Compare and contrast these implementations with respect to space use and the runtimes of the key operations. Make sure you include a discussion of *amortized analysis* in your answer.

2. Discuss the trade-offs between hashtables and red-black trees.

3. Explain the importance of randomness in some data structures and algorithms. Make sure you use specific examples.

4. Give recurrence relations for the worst-case runtimes of Mergesort, Heapsort, and Quicksort, and explain how each part of the recurrence relations relate to the algorithms. Also, give the worst-case runtimes in big-Oh notation and provide a mathematical justification for those runtimes with respect to the recurrence relations.

Extra Credit.

1. (6 points) What is your favorite data structure or algorithm and why? (Obviously, there is no right or wrong here. I'm just looking for a thoughtful response.)
2. (1 point) Which country hosted the 1994 FIFA World Cup? (Hint: They also hosted the 1996 Summer Olympics and the 2002 Winter Olympics.)