CSCI255 **Sample Final Exam Questions**

You may use your books and notes for the exam. Give *complete*, *concise*, *clear*, and *orderly* solutions to all of the problems, and *justify all of your answers*. The actual exam will be shorter than this.

1. True false questions, with partial credit for brief justifications. (Sample questions omitted).
2. What is the coefficient of x^{11} in $(3x-9)^{19}$?
3. The divide-into-thirds algorithm for finding a fake coin turned out to be more efficient than the divide-in-half algorithm. However, the ternary search algorithm is actually worse than binary search. Explain why splitting into thirds turned out to be better in the one case but not the other.
4. Compute $13^{123} \bmod 100$ using one of the binary exponentiation algorithms we discussed. *Show all of your work!*
5. Imagine an AVL tree with 4 nodes: The root is 5 and it has left child 3 and right child 9. The node with value 9 has left child 7. First, draw the tree. Then insert 8 into the tree, showing the intermediate steps and what the final tree will look like after it has been rebalanced.
6. Given a graph, be able to perform BFS and/or DFS, showing your work.
7. Prove that $\sum_{k=0}^{n} 3^k \binom{n}{k} = 4^n$.
8. Prove that the sum of the first n odd integers is n^2 by
   (a) Evaluating a summation.
   (b) Using mathematical induction.
9. I have a pile of n **bolts** of different sizes. For each bolt, I have a **nut** and **washer** that fits only that bolt. Unfortunately, they all got mixed together. I can compare nuts and washers with bolts to determine if they fit, but I cannot compare nuts with nuts, washers with washers, bolts with bolts, or washers with nuts. Give an algorithm with an average-case efficiency of $\Theta(n \log n)$ to match all of the nuts, bolts, and washers.
10. Use induction to prove that the following program correctly computes $\sum_{k=1}^{n} k$ for all $n \geq 1$.

    You are NOT proving what the summation is. You are proving that the code correctly produces it.

    ```
    int sum(int n) {
        if(n==1) {
            return 1;
        } else {
            return n + sum(n-1);
        }
    }
    ```

    Also, give the complexity of this algorithm. Then come up with a more efficient algorithm to solve the problem, giving the complexity of your algorithm.
11. Describe an algorithm that can determine whether or not a graph is bipartite. Give the computational complexity of your algorithm and specify which algorithmic technique it employs.
12. Given an array of n integers, give an algorithm with complexity *better than* $O(n^2)$ to find the smallest difference between any pair of elements in the array. For instance, if the input is $[9, 6, 1, 10, 7, 12]$, the would output would be 1 since the difference between 6 and 7 (as well as 9 and 10) is 1.
13. Find an optimal prefix code to encode the string "this is a string in it is a sting". You should ignore the spaces.
14. Give an efficient algorithm to delete all of the nodes from a binary tree. What is the complexity of your algorithm? Does the complexity depend on how well balanced the tree is? Explain. What design technique is your algorithm based on?

15. The following method returns true if and only if none of the entries of the array are 0. In other words, it determines the truth value of the expression $\forall x(a[x] \neq 0)$.

```
boolean noZeroElements(int[] a, int n) {
    for(int i=0;i<n;i++) {
        if(a[i] == 0 )
            return false;
    }
    return true;
}
```

The two methods below implement a similar idea for two arrays. Assume `a` and `b` have the same size.

```
boolean unknown1(int[] a, int[] b, int n) {
    for(int i=0;i<n;i++) {
        if( a[i]==0 && b[i]==0 )
            return false;
    }
    return true;
}
```

```
boolean unknown2(int[] a, int[] b, int n) {
    if(noZeroElements(a, n)) {
        return true;
    } else if(noZeroElements(b, n) {
        return true;
    } else {
        return false;
    }
}
```

(a) Use `a`, `b`, quantifier(s) and/or logical operators to specify what `unknown1` determines.

(b) Use `a`, `b`, quantifier(s) and/or logical operators to specify what `unknown2` determines.

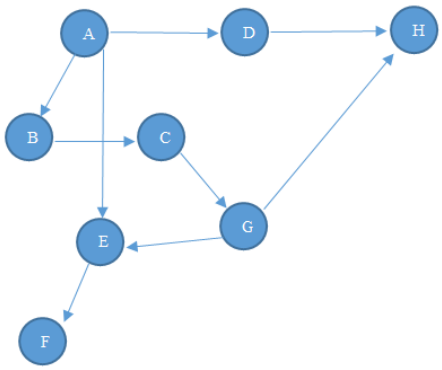
(c) Prove or disprove that `unknown1` and `unknown2` are determining the same thing.

16. Give a tight bound on the worst-case complexity of each of the following algorithms.

(a)
```
void foo(int n,int m) {
    int foo = 0;
    for(int i = 0 ; i < n ; i++) {
        for(int j=0;j<n/2;j++) {
            foo++;
        }
    }
    for(int j = 0 ; j < m ; j++)
        foo++;
}
```

(b)
```
clearTree(BinarySearchTree T) { // Assume T initially contains n elements
    while(!T.empty()) {
        int v = T.maximumValue();
        T.delete(v);
    }
}
```

17. Consider the graph to the right.



(a) Perform a topological sort on this graph using the decrease-and-conquer method. Show intermediate steps.

(b) Perform a topological sort on this graph using the DFS-based method. Show intermediate steps.

(c) Did you get the same answer in (a) and (b)? In general, will you always get the same answer? Explain.

(d) Modify the graph so that it cannot be topologically sorted (changing it as little as possible). Explain why the change makes it impossible.

18. *Ferzle Art Studio* needs to evaluate which of several art projects to pursue. The possible projects are in the chart to the right, where P is the estimated profit and C is the cost to create the artwork (figures are given in thousands of dollars). Assume the estimated profits are accurate and that the studio can spend up to \$10,000.

#	Project	P	C
1	Legopolis	1	1
2	Legoku	4	6
3	Twin Fight	3	2
4	The Square	6	2
5	Fibonacci Rectangle	1	2
6	Opposites Attract	8	5
7	Latin Squared Square	10	4
8	Raised Asquareness	2	1

Use a *dynamic programming* approach to determine which works of art the studio should create so that the profit is maximized. In particular, you need to define a function $M[i, j]$ such that the optimal solution is $M[a, b]$ for some values of a and b. Assume in general there are n projects and m dollars (given in thousands).

(a) Describe $M[i, j]$ in words, including conditions on the values of i and j.

(b) Give a recursive formula to compute $M[i, j]$, including the base case(s).

(c) Compute the optimal solution, showing all of your work. Give the maximal profit and specify which projects should be funded.

19. This is a continuation of the *Ferzle Art Studio* problem, but now we will assume that works of art can be partially funded, resulting in a smaller piece that will result in partial profit proportional to the percent of the budget that was funded (e.g. if \$1000 is spent to fund *Fibonacci Rectangle*, the profit would be \$500). We still have \$10,000 to spend.

#	Project	P	C
1	Legopolis	1	1
2	Legoku	4	6
3	Twin Fight	3	2
4	The Square	6	2
5	Fibonacci Rectangle	1	2
6	Opposites Attract	8	5
7	Latin Squared Square	10	4
8	Raised Asquareness	2	1

(a) Describe an efficient algorithm to solve this problem.

(b) What algorithmic technique are you using?

(c) What is the computational complexity of your algorithm, assuming there are n project and m dollars?

(d) Show computations that determine the projects that should be funded. Give the maximal profit and the amount of each project that should be funded.

(e) You should have noticed that the profits from this and the previous problem were not the same. What can you say in general about the relative solutions to these two versions of the problem? Explain.

20. Consider the `TwoThirdsSort` algorithm below, which correctly sorts a list of numbers.

```
TwoThirdsSort(A,L,R) {
      if(R<=L) return; // Array has at most one element
      if(A[R]<A[L]) {  // Swap first and last element if they are out of order
          Swap(A,L,R);
      }
      if(R-L>1){ // If the list has at least 2 elements, continue sorting.
        int third=(R-L+1)/3;
        TwoThirdsSort(A,L,R-third) // sort first two-thirds
        TwoThirdsSort(A,L+third,R) // sort last two-thirds
        TwoThirdsSort(A,L,R-third) // sort first two-thirds again
    }
}
```

(a) Give *and prove* a tight bound on the efficiency of `TwoThirdsSort(A,0,n-1)` (i.e. on a list of size n).

(b) Is `TwoThirdsSort` a good choice for a sorting algorithm? Explain.