CS472 Spring 2019 Exam #1

This exam has 16 questions, for a total of 200 points.

Please answer the questions in the space provided on the question sheets. If you run out of room for an answer, continue on the back of the page.

You may use one page of handwritten notes as an aid during this exam. Otherwise, the exam is "Closed book, Closed notes".

Grades will be posted on Blackboard within one week of the administration of this exam.

Name: _____

Question	Points	Score
1	9	
2	12	
3	10	
4	21	
5	12	
6	10	
7	12	
8	11	
9	24	
10	12	
11	13	
12	9	
13	14	
14	8	
15	15	
16	8	
Total:	200	

1. (9 points) Explain the relationship between program, data, and algorithm.

Solution: Best way to remember this is the title to N. Wirth's book on data structures: "Program+data = Algorithms".

2. (12 points) What is the formal mathematical definitions of the rate-of-growth functions $O(f(x))$, $\Theta(f(x))$, and $\Omega(f(x))$?

Solution:

$$\begin{aligned} O(f(n)) \to g(n) &\implies \exists (c > 0) \texttt{and} (n_0 > 0) \texttt{s.t.} f(n) \leq c * g(n) \forall n \geq n_0 \\ \Omega(f(n)) \to g(n) &\implies \exists (c > 0) \texttt{and} (n_0 > 0) \texttt{s.t.} f(n) \geq c * g(n) \forall n \geq n_0 \end{aligned} \tag{1}$$

$\Theta(f(n)) \to g(n)$ implies both $O(n)$ and $\Omega(n)$. So have to show the definition of $O(f(n))$ and $\Omega(f(n))$ both apply. That means four constants!

3. (10 points) What are the differences between worst-case, average-case, and best-case behavior and explain the relationship between each behavior and each rate-of-growth function ($O(f(x))$, $\Theta(f(x))$, and $\Omega(f(x))$)?

Solution:

Best Case	Best your program can do	$\Omega(f(n))$
Worst Case	Worst your program can do	$O(f(n))$
Average Case	Average your program can do	$\Theta(f(n))$

4. Identify which of the following functions has a greater rate-of-growth (in some cases, the rate-of-growth may be the same):
   (a) (7 points) $n^2 + 3n^3$ or n^4
   (b) (7 points) 2^n or $n!$
   (c) (7 points) 2^{n-1} or 2^{n+1}

Solution: n^4, $n!$, and equal.

The behavior of exponents is weird when dealing with rate-of-growth functions.

5. (12 points) List the following functions in order of increasing rate-of-growth: $n!$, $\log_2(n)$, $n * \log_2(n)$, n^2, n, n^3, 2^n.

Solution: $log_2(n)$, n, $n * \log_2(n)$, n^2, n^3, 2^n, $n!$.

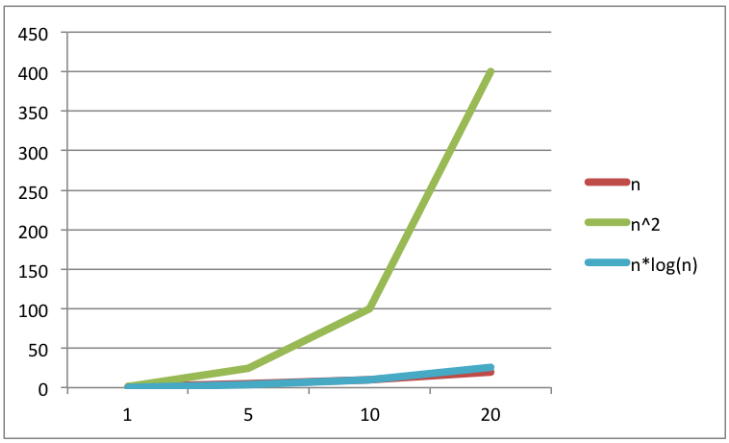

Figure 1: A graph of n, $n * log(n)$, and n^2

6. (10 points) A student in CS372 attempts to figure out the meaning of $O(x)$ by graphing n, $n * log(n)$, and n^2 as shown in Figure 1. Use the formal definition of $O(f(x))$ to explain some of the seeming contradictions in the figure (for instance, why do the lines for n and $n * log(n)$) appear to overlie each other, are you using large enough values of n for your sample size, and so on).

Solution: Explain the definition of "big-oh" and show how the choice of constants can get "unexpected" behavior for small problem sizes.

7. (12 points) Given an integer N, write a brute-force algorithm for finding all integers $a < 100000$ and $b < 100000$ such that $a^2 + b^4 = N$ is true. Assume the existence of a function `power(a,n)` which will return the value of a^n. The `power()` function is $O(n)$. What is the basic operation of your algorithm and the algorithm's time complexity?

Solution: The trick with this question to remember that a brute-force algorithm will enumerate all possible solutions searching for the "best" solution:

Algorithm 1: Find when $a^2 + b^4 = N$ is true

```
Input: Some positive integer N
Output: All combinations of a and b that satisfy the equation
for a ← 0..100000 do
    for b ← 0..10000 do
        t ← power (a,2) + power (b,4);
        if t = N then
            Output a, b;
```

It's important to note that the basic operation of this algorithm is the call to the `power()` function. But `power()` is $O(n)$! So, the entire algorithm has an order of $O(n^3)$ because the two-level nested loops make things $O(n^2)$ while doing a extra $2 * O(n)$.

8. (a) (5 points) Explain the set-theoretic definition of an Abstract Data Type (ADT).
   (b) (3 points) What is the relationship between an ADT, an object, and a class?
   (c) (3 points) What is the relationship between an ADT and an algorithm?

Solution: An **ADT** is a pair of sets $< V, O >$ where V is the set of values this ADT operates upon and O is the set of operations that work upon this set.

A class is the programmatic way of implementing an ADT while an object is a runtime instance of a class.

Algorithms are defined to work upon ADTs.

9. Define
   (a) (6 points) A graph.
   (b) (6 points) A directed graph.
   (c) (6 points) A graph cycle
   (d) (6 points) A complete graph.

Solution:

- A pair of sets (V, E) where V is the vertices (nodes) of the graph and E is a set of ordered pairs indicating the edges between vertices.
- A directed graph is a graph with unidirectional edges.
- A graph cycle is a path through a graph that begins and end at the same vertex.
- A complete graph is a graph where are vertices are connected to all other vertices.

10. (12 points) Sketch the data structures we use to represent a graph in an adjacency list and a graph in an adjacency matrix.

Solution:

- Adj list: A list of lists, with each node in the main list corresponding to the nodes of the graph while each element in the sub-lists are the destination of an edge.
- Adj matrix: A 2-d matrix where a non-zero value indicates the presence of a edge from the nodes indicated by the row and columns.

11. As concisely as possible, explain how you can detect each of the following situations in a graph represented by an adj. list and an adj. matrix.
    (a) (3 points) The graph is complete.
    (b) (5 points) The graph has a cycle (a loop).
    (c) (5 points) The graph has a node that does not have incoming or outgoing edges.

Solution:

- A graph is complete:
  - if in an adj. matrix, all of the elements on the main diagonal are equal to 1.
  - if in an adj. list, each of the linked lists for the vertices contain all other vertices of the graph.
- A graph has a loop:
  - if in an adj. matrix, there is an element equal to 1 on its main diagonal.
  - if in an adj. list, the list for a vertex has that vertex as an element in the list.
- An (undirected, without loops) graph has an isolated vertex:
  - if in an adj. matrix, one row consists entirely of zeroes.
  - if in an adj. list, one of the elements of the array contains the empty list.

12. (9 points) You have two different programs solving the same graph problem. Program A implements an algorithm that is $O(n^3)$ where n is the number of nodes in the graph. Problem B implements an algorithm is $O(n * \log(n))$. In what situations where you would want to use Program A rather than Program B to solve this problem?

Solution: One must look closely at the problem being solved to gauge the size of the graph. For small graphs, it may hold that $O(n^3)$ grows slower than $O(n*log(n))$. You have to plug in some numbers and check.

13. (a) (7 points) True or false: a tree is a special case of a graph. Justify your answer.

    (b) (7 points) True or false: The in-order, pre-order, and post-order traversal of a tree are the same as depth-first or breath-first search in a graph. Justify your answer.

Solution: Both questions are true. A tree is a special case of a graph (with some additional restrictions). Pre-order and post-order traverals are depth-first searches while in-order is a breath-first search.

14. Define the following:
    (a) (4 points) Computational geometry.
    (b) (4 points) Convex hull of a set of points

Solution:

- Computational geometry is the study of algorithms that can be expressed in geometric terms.
- The convex hull of a set of points is the smallest possible polygon that contains all of the points.

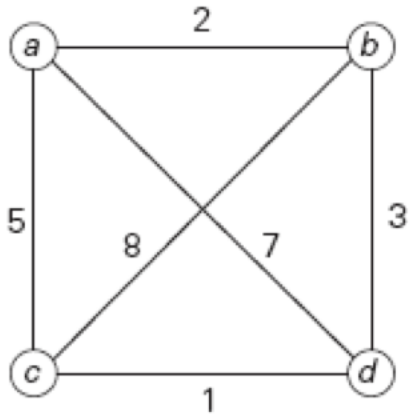

Figure 2: A Weighted Undirected Graph

15. (a) (5 points) Define: The Traveling Salesman Problem (TSP).

    (b) (10 points) The brute force algorithm for solving the TSP enumerates all possible tours (path through a graph that begin and end at some starting point), looking for the least cost tour (sum of edge weights). Using the graph in Figure 2, show how the brute force algorithm would find the least cost tour in that graph (list the tours) assuming Node 'a' is the starting node.

Solution: Traveling Salesman Problem(TSP): Find the least-cost path through a graph from a starting node back to that starting nodes which visits each node (except the start node) exactly once.

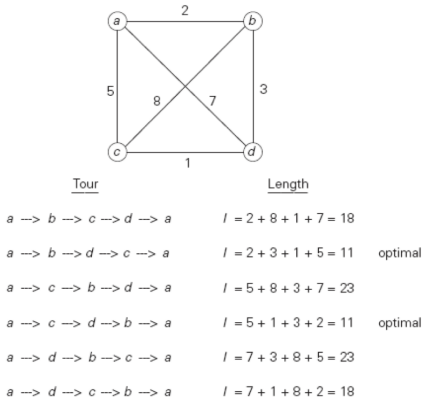

Figure 3: TSP Solution for This Graph

16. (8 points) Assume that you have a function named `dmetric(point1, point2)` that returns the Euclidean distance between two points (square root of the sum of the squares of the points). Write a brute-force algorithm that finds the points that are most close to each other in a set of points in the plane.

    HINT: Work with points as whole entity; don't try to do anything with the x and y values of the points.

Solution: The trick in this case is to realize that having an implementation of `demetric` that takes points as inputs means that we can work with points as an entity rather than doing anything with the components of the points. This makes this problem into something very similar to what we saw with the $a^2 + b^4 = N$ problem:

Algorithm 2: Brute Force Closet Pair

```
Input: A list of points P
Output: The pair of points closet to each other
mindistance = ∞;
for p1 in P do
    for p2 in P do
        t ← dmetric (p1, p2);
        if t < mindistance then
            mindistance = t;
            cpoint1 = p1;
            cpoint2 = p2;
Output cpoint1, cpoint2;
```