CSC 360 Assignment 6

Names: Kevin Andor, Nathaniel DeHart, Cole Stewart

# **12.5 points each**

1. Solve, using the iteration method, and EXPLAIN each step:

T(n) = T(n-2) + 1

2nd iteration: T(n - 2) + 1 = T(n - 4) + 2

3rd iteration: T(n - 4) + 2 = T(n - 6) + 3

T(n) appears to have the form: T(n - 2k) + k for the kth iteration

If we pick k = 1/2n - 1/2, then we have:

T(n) = T(n - 2(1/2n - 1/2)) + 1/2n - ½ = T(1) + 1/2n - ½

T(1) + 1/2n - ½ = Θ(n)

1. Show work, and apply the Master Method to solve:

T(n) = 16T(n/4) + n2

a = 16, b = 4, k = 2

16 = 4^2

since a = b^k in this case T(n) = O(n^2 log n)

1. Suppose we have 512 numbers where each number is a 6-digit number. Will the Radix Sort perform better than n log n, worse than n log n, or at the level of n log n? Explain how you arrive at your answer.

- There are 512 numbers being sorted and the max number of positions in each number is six

- Because 512 is bigger than six, it can be sorted in or be close to sorted in linear time

- This means that it runs better than nlogn because nlogn is not linear

1. For the recurrence: T(n) = 5T(n/3) + log n

Explain what this recurrence implies is happening recursively. Hint: What does this recurrence tell us about a recursive call being made?

T(n) is calling 5 iterations of itself with its data set cut into a third in each call and merged with the subdividing of its data set.

1. Provide good definitions for the following terms as they relate to graphs:
   1. Connected

A graph where a path exists between any two vertices.

* 1. Disjoint

A graph that is not connected.

* 1. Strongly Connected

A directed graph such that there is a directed path between any 2 vertices in both directions.

* 1. Weakly Connected

A directed graph that is not strongly connected, but the underlying undirected graph is connected.

1. Explain the differences between depth-first search and breadth-first search. As part of your answer, explain when it would be appropriate to choose one over the other.

- A depth-first search starts by pushing a vertex onto a stack and marks it as visited. It then pops a vertex, v, from the stack and processes it. Whenever it comes across an unmarked neighbor of v, u, it marks u and places it onto the stack. This repeats until the stack is empty.

- A breadth-first search starts by placing a start vertex into a queue and marking it as visited. It then dequeues a vertex, v, from the queue and processes it. For each unmarked neighbor of v, u, it marks u and places it into the queue. This repeats until the stack is empty.

- Because depth-first uses a stack and breadth-first uses a queue, a breadth-first is better for smaller sets of data whereas a depth-first might be better for larger sets of data because it could require more time in a breadth-first search. A breadth-first search would also be better than a depth-first search if you were to know that the solution is closer to where you start. Both, however, have the same running time of O(V+E).

1. What does it mean to say that an algorithm is NP-Complete? Name at least two problems that fall into this category of problems.

If an algorithm is NP-Complete, it means that they’re the hardest algorithms to implement (the run time would take too long) and can only verify in polynomial time.

MIN Dom Set and MAX Ind Set would fall into this category.

1. What is an approximation algorithm? Why are they important?

An approximation algorithm finds A POTENTIAL solution to a problem that we can consider “good enough”, but not always THE BEST solution.

This allows us to at least verify possible solutions from problems that are considered NP/NP-Complete in polynomial time, as running an NP problem (especially with large data sets) would take an unreasonable amount of time to finish if run with a brute force method.