**Design and Analysis of Computer Algorithms Questions**

1. Provide an example of a recursive function in which the amount of work on each activation is constant. Provide the recurrence equation and the initial condition that counts the number of operations executed. Specify which operations you are counting and why they are the critical ones to count to assess its execution time. Draw the recursion tree for that function and determine the Big-Θ by determining a formula that counts the number of nodes in the tree as a function of n.
2. Sorting algorithms are one kind of algorithm whose performance may depend upon the data. Choose one of the sorting algorithms or any other algorithm and explain whether the there are any differences in the best, average and worst cases. If there are no differences, explain why not. If there are differences, describe the data in the different cases and explain how the performance differs in each case.
3. Give an example of an application of a graph in computer science. Indicate whether the graph is directed or undirected. What significance, if any, does the presence of cycles have in this graph? Also indicate what significance, if any, there is to whether the graph is connected.
4. Choose a problem that lends to an implementation that uses dynamic programming. Clearly state the problem and then provide high-level pseudocode for the algorithm. Explain why this algorithm can benefit from dynamic programming. Try to choose an algorithm different from any already posted by one of your classmates.
5. Give an example of an application of a graph, in which the minimum spanning tree would be of importance. Describe what the vertices, edges and edge weights of the graph represent. Explain why finding a minimum spanning tree for such a graph would be important.
6. Give an example of an application of a graph, in which determining all pair’s shortest paths would be of importance. Describe what the vertices, edges and edge weights of the graph represent. Explain the significance of the shortest path for such a graph and why it would be important.
7. The traveling salesperson problem is a harder problem than Dijkstra's single-source shortest path problem. In other words, the typical Greedy algorithm approach does not work for this problem. It is even harder than the all-points shortest path algorithm implemented with Floyd's algorithm. Give an example of a graph that shows that the path that would be chosen by relying on shortest-path information by choosing the closest vertex each time isn't sufficient to find the shortest circuit. What makes this problem harder? Why are the straight forward approaches to this problem exponential?