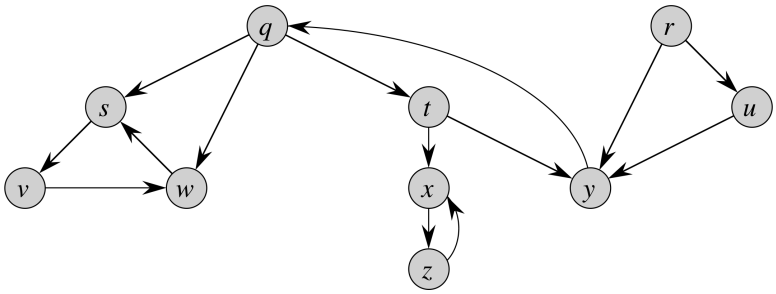
CS6033 Homework Assignment 7*

Due November 5th at 5:00 pm
No late assignments accepted



1. (10 points) Create the adjacency list for the graph above. Create the adjacency matrix for the subgraph composed of nodes v, s, w, q and t. If I had asked you to create the adjacency matrix for the entire graph, how many entries would it have?

2. (10 points) Read about `Breadth-first trees` on the bottom of page 600. Then show the d (distance) and π (predecessor) values that result from running breadth-first search on the directed graph above when *starting* at vertex q.

   Draw the predecessor subgraph.

3. (10 points) What is the running time of BFS if we represent its input graph by an adjacency matrix and modify the algorithm to handle this form of input?

4. (10 points) Show how depth-first search works on the graph of above. Assume that the for loop of lines $5-7$ of the DFS procedure considers the vertices in *alphabetical order*, and assume that each adjacency list is ordered alphabetically. Show the *discovery* and *finishing* times for each vertex.

5. (10 points) Instead of using recursion it is possible to usa a *stack* to choose the next vertex to visit when traversing a graph. Rewrite the procedure DFS, using a stack to eliminate recursion.

*Many of these questions came from outside sources.

6. (10 points) The transpose of a directed graph $G = (V, E)$ is the graph $G^T = (V, E^T)$, where $E^T = \{(u', u)$ `such that` $(u, u') \in E\}$. Thus, G^T is G with all its edges reversed.

   Describe efficient algorithms for computing G^T from G, for both the adjacency-list and adjacency-matrix representations of G. Analyze the running times of your algorithms

7. (10 points) In the $O(n)$ worst case deterministic select algorithm, the pivot was found by dividing the input elements into groups of 5 and then finding the median of their medians.

   Would the algorithm still run in $O(n)$ time if we divided the elements into groups of 7 instead of 5?

   Would the algorithm still run in $O(n)$ time if we divided the elements into groups of 3 instead of 5?

   Prove your answer.

8. (15 points) The Chancellor would like to invite the top 15% of his generals to participate in a raid.

   You are given an unsorted list of all the generals' names and the Chancellor assessment of each of the generals. The way the generals are assessed by the Chancellor is based on a number of criteria. Consequently you may only use $<$ operator **only** to compare two generals.

   Design an algorithm and justify the running time of your algorithm. Your algorithm must *efficient* with respect to the worst case running time.

9. (10 points) We can improve the running time of quicksort in practice by taking advantage of the fast running time of insertion sort when its input is "nearly" sorted. Upon calling quicksort on a subarray with fewer than k elements, let it simply return without sorting the subarray. After the top-level call to quicksort returns, run insertion sort on the entire array to finish the sorting process. Argue that this sorting algorithm runs in $O(kn + n \log(n/k))$ expected time. How should we pick k, both in theory and in practice?

10. (15 points) You have been enlisted by an organizer for a special New Years Day event; your job is to help make sure the evening goes smoothly. There are n people who will be attending the event, which will be held in two rooms. Unfortunately, some of the people attending have deep grudges against some of the other people who are attending (the previous year's event is the stuff of legends). Your job is to assign the people, if possible, to one of the two rooms so that no two people with a grudge are in the same room (The organizers don't want another billion dollars in property damage payouts).

You are given the list of g grudges. Design an $O(n+g)$ time algorithm that either assign all the people to one of two rooms so that no one with a grudge is in the same room as the person who he/she has a grudge with, or determines that this is not possible.

Your algorithm must be efficient. State the running time of your algorithm given $G = (V, E)$ and justify your running time.

11. (10 bonus points) Suppose you would like to sort n music files, but you only have an old, unreliable computer, which you have nicknamed "Rustbucket." Every time Rustbucket compares two music files, x and y, there is an independent 50-50 chance that it has an internal disk fault and returns the value 0, instead of the correct result, 1, for "true" or -1, for "false," to the question, "$x \leq y$?"' That is, for each comparison of music files that Rustbucket is asked to perform, it is as if it flips a fair coin and answers the comparison correctly if the coin turns up "heads" and answers with 0 if the coin turns up "tails." Moreover, this behavior occurs independent of previous comparison requests, even for the same pair of music files. Otherwise, Rustbucket correctly performs every other kind of operation (not involving the comparison of two music files), including if-statements, for-loops, and whileloops based on comparisons of integers. Describe an efficient algorithm that can use Rustbucket to sort n music files correctly and show that your algorithm has an expected running time that is $O(n \log n)$.

    This question is from *Algorithm Design and Applications* by Goodrich and Tamassia.

12. (3 bonus points) Think of a good exam/homework question for the material covered in Lecture 7.