CS658 Problems

Some of the graph questions ask for linear time algorithms. This should be taken to mean linear in the number of edges, i.e., \( O(m) \). For problems labeled with Programming Option you can get extra points by writing a program that implements the algorithm. For these problems, create a directory named Homework (upper case H) and save your programs in files named pX.c (or pX.py), where X is the number of the problem.

1. Find the shortest path tree for the graph below. Use vertex A as the starting vertex. The shortest path tree consists of all edges used on shortest paths from A to any other vertex.

   ![Figure 1](image1.png)

2. Answer the following for the graph shown in the figure below.

   (a) What is the cost of its minimum spanning tree?
   (b) How many minimum spanning trees does it have?
   (c) Suppose Kruskals algorithm is run on this graph with A as the starting vertex. In what order are the edges added to the MST?

   ![Figure 2](image2.png)

3. Design a \( O(m) \) algorithm that takes as input the adjacency list of a connected, undirected graph \( G \) and determines if there is an edge you can remove from \( G \) while still leaving \( G \) connected?
4. Suppose we have a computer that can multiply only 8-bit numbers. How many 8-bit multiplications will be required to multiply two 64-bit numbers using the following methods?
   (a) The usual longhand multiplication method.
   (b) The Gauss divide and conquer method.
   (c) FFT.

5. How many times would the following function print hello?
   (a) Give an asymptotic estimate (big-O notation).
   (b) Give an exact answer.

   function f(n):
   if n > 0:
       f(n/3)
       f(n/3)
       f(n/3)
       f(n/3)
   for i = 1 to n:
       for j = 1 to n:
           print "hello"

6. The reverse of a directed graph $G = (V, E)$ is another directed graph $GR = (V, ER)$ on the same vertex set, but with all edges reversed; that is, $ER = \{(v, u) | (u, v) \in E\}$. Give a linear-time algorithm for computing the reverse of a graph in adjacency list format. In other words, devise an algorithm that reads the adjacency list of a graph, and which prints the adjacency list of the reverse graph.

7. Here is a proposal for how to find the length of the shortest cycle in an undirected graph with unit edge lengths. When a back edge $(v, w)$ is encountered during a depth-first search, it forms a cycle with the tree edges from $w$ to $v$. The length of the cycle is $\text{level}[v] - \text{level}[w] + 1$, where the level of a vertex is its distance in the DFS tree from the root vertex. This suggests the following algorithm:
   
   - Do a depth-first search, keeping track of the level of each vertex.
   - Each time a back edge is encountered, compute the cycle length and save it if it is smaller than the shortest one previously seen.

   Is this algorithm correct? If it is, prove it. If not, give an example where the algorithm fails.
8. **(Programming Option)** The following problem is an application from automated program analysis. For a set of variables $x_1, \ldots, x_n$, you are given some equality constraints, of the form $x_i = x_j$ and some inequality constraints, of the form $x_i \neq x_j$. Is it possible to satisfy all of them? For instance, the constraints $x_1 = x_2$, $x_2 = x_3$, $x_3 = x_4$, and $x_1 \neq x_4$ cannot be satisfied.

   Give an efficient algorithm that takes as input $m$ constraints over $n$ variables and decides whether the constraints can be satisfied.

9. **(Programming Option)** Design and analyze an algorithm that takes as input an undirected graph $G = (V,E)$ and determines whether $G$ contains a simple cycle (that is, a cycle which does not intersect itself) of length four. Its running time should be at most $n^3$. You should assume that the input graph is represented as an adjacency list.

10. **(Programming Option)** Design a linear-time algorithm which, given an undirected graph $G$ and a particular edge $e$ in it, determines whether $G$ has a cycle containing $e$. You should assume that the input graph is represented as an adjacency list.

11. **(Programming Option)** For each node $u$ in an undirected graph, let $\text{twostep}[u]$ be the sum of the degrees of the neighbors of $u$. Show how to compute the entire array $\text{twostep}$ in linear time. You should assume that the input graph is represented as an adjacency list.