Homework 4

ECE 1762 Algorithms and Data Structures
Fall Semester, 2016

Due: December 5, 4PM, in-box (near SFB-560)

This is another LONG homework. Start immediately!

Unless otherwise stated, for each algorithm you design you should give a detailed description of the idea, proof of correctness, termination, analysis and proof of time and space complexity. If not, your answer will be incomplete and you will miss credit!

1. [**Graph Algorithms, 15 Points**] When an adjacency matrix representation is used, most graph algorithms require time $O(n^2)$ (where n is the number of vertices), but there are some exceptions. Here's one.

   The race for the election of Democratville's mayor is heating up and everyone in Democratville is competing for it. Each candidate has a few preferences (people who the person would be willing to accept as a mayor). Of course, the set of preferences for a person includes him/her self all the time.

   What we are looking for is a "perfect" mayor who is in the set of preferences of every person and who does not prefer anyone but him/her self (wouldn't that make a good mayor?). In fact, all we want to know whether such a person exists or not. Otherwise, we're willing to live in anarchy. Define a directed graph with the set of candidates as the vertices and a directed edge from vertex a to vertex b if and only if b is in the set of preferences of a.

   Suppose that the number of people (also the number of candidates) is n. Give an algorithm which executes in $O(n)$ time and determines if such a perfect chairman exists or not. Assume that you are given the graph described above in the form of an $n \times n$ adjacency matrix.

2. [**Graph Algorithms, 15 Points**] A *scorpion* is an undirected graph having the following form: there are three special vertices, called the *sting*, the *tail* and the *body*, of degree 1, 2 and $n - 2$ respectively. The sting is connected only to the tail, the tail is connected only to the sting and body, and the body is connected to all the vertices except the sting. The other vertices of G may be interconnected arbitrarily. Given an adjacency matrix representation of a graph G, give an algorithm that probes the adjacency matrix at most $O(n)$ times and determines whether G is a scorpion or not.

3. [**Shortest Paths, 15 Points**] Alice wants to fly from city A to city B in the shortest possible time. Alice turns to the traveling agent who knows all the departure and arrival times of all the flights on the planet. Give an algorithm that will allow the agent to choose an optimal route.

4. [**Spanning Trees, 10 Points**] Consider the following problem. Given a graph $G = (V, E)$ with weight function $b : E \rightarrow \Re^+$ (bandwidth), we are interested in finding for each pair of vertices $u, v \in V$ a path $P_{u,v}$ in G such that the minimum of b on the edges on $P_{u,v}$ is maximized (that is, we want a path that maximizes the bottleneck bandwidth).

   (a) Show that there is a spanning tree T such that for any pair $u, v \in V$ the unique path from u to v in T realizes the maximum bottleneck bandiwdth.

(b) Describe an efficient algorithm to compute such tree (as always, verify correctness and analize runing time).

5. [**Maximum Flow, 10 Points**] Problem 26-1, (CLRS 3rd edition page 760), (CLRS 2nd edition page 692), (CLRS 1st edition Problem 27-1, page 625).

6. [**Maximum Flow, 20 Points**] We consider the following variant of the flow problem: In addition to the network $G = (V, E)$, source and sink $s, t \in V$, and the edge capacity $c : E \to \Re^{\geq 0}$, there is also an edge cost function $w : E \to \Re^{\geq 0}$. The cost of a flow f is:

$$W(f) = \sum_{(u,v)\in E: f(u,v)>0} f(u,v) \cdot w(u,v).$$

The sum is over $f(u,v) > 0$ because by convention we have $f(v,u) = -f(u,v)$ and so there is an edge with negative flow, for each one with positive flow. The problem is, given K, to determine a flow f with value $|f| = K$ whose cost $W(f)$ is smallest. So K must be smaller than the max flow.

We outline an approach to solve this problem: A *cycle-flow* or *circulation* in a network G is a flow such that it is zero in all edges except in (around) a cycle C; the flow just goes around the cycle, so it does not add to the value of the flow. The idea is as follows: suppose we have f with $|f| = K$; then f is not min-cost iff there is a cycle-flow f_C in the reduced network G_f such that $f + f_C$ has the same value but smaller cost (must prove !). So, the outline of the algorithm would be:

```
MIN-COST-FLOW $(G, s, t; c, w; K)$
    find an $s - t$ flow $f$ in $G$ with $|f| = K$; if not possible return “failed”
    while there is a cycle-flow $f_C$ in $G_f$ with “reduced negative cost” then
        augment flow through $C$
```

Make the outline above into a precise argument and algorithm (among other things, you'll need to make precise “reduced negative cost”). For the correctness, you'll need appropriate lemmas/theorems analogue to those for max-flow. For the algorithm, give details of implementation (for example, how is the cycle-flow determined ?) and analyze running time.

7. [**Machine Learning, 10 Points**]

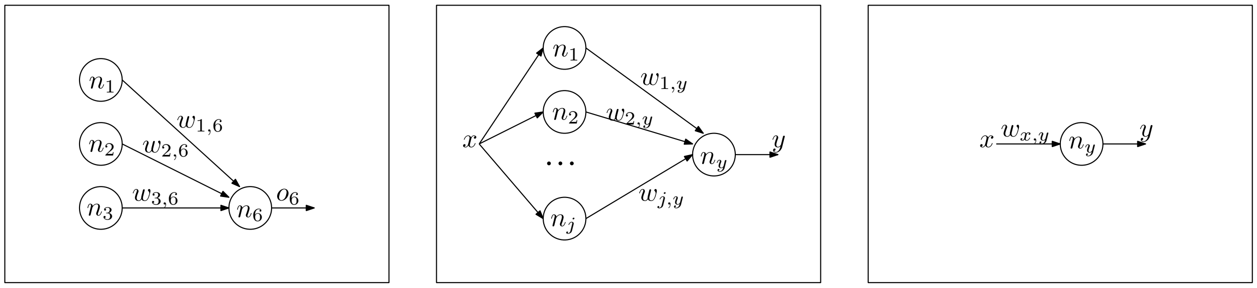
In a neural network, the output of a node i is the result of passing the weighted sum of its inputs through an activation function. That is:

$$o_i = \sum_j \Phi(w_{j,i} \cdot o_j)$$

Where j iterates over the inputs to neuron i. For instance, in Figure 1(a), we have $o_6 = \Phi(w_{1,6} \cdot o_1 + w_{2,6} \cdot o_2 + w_{3,6} \cdot o_3)$. This question asks you to consider a neural network with a linear activation function $\Phi(x) = c \cdot x$ for some constant c. Assume every node uses the same activation function with the same constant c.

(a) Consider the network N using linear activation functions with one hidden layer, one input x, and one output y, as shown in the Figure 1(b) below. This network computes some function $y = f(x)$. Show that there is a network using linear activation functions with no hidden layers (*i.e.,* a single neuron) that computes the same function as N. In other words, show that there is a value of $w_{x,y}$ such that the two networks in Figures 1(b) and 1(c) compute the same function.

(b) Prove that a neural network using linear activation functions with any number of hidden layers is equivalent to a single neuron with a linear activation function. *Hint:* use induction with part (a) as your basis step.



(a) (b) (c)

Figure 1: (a) Example neural network (b) Neural network with one hidden layer (c) Neural network with no hidden layers