Homework 5, CS520, Fall 2013, McConnell

1. Recall Euler’s formula, which we proved in class: \( n + f = m + c + 1 \), where \( n \) is the number of vertices, \( f \) is the number of faces in a planar embedding of the graph, \( m \) is the number of edges, and \( c \) is the number of connected components.

   (a) If you have studied Chapter 2 of de Berg, you will be aware of a proof that this implies that the number of edges is \( O(n) \). Find the proof, study it, then get a stronger result by proving that there is a vertex of degree at most five. (This will imply that the number of edges is less than \( 5n \), since you can delete all the edges by iteratively deleting a vertex of degree at most five in what remains of the graph.)

   (b) Why does this imply that the number of faces is less than \( 4n \)?

2. Problem: Given a planar subdivision consisting of \( n \) segments and a set of \( m \) points in the plane, label each of the \( m \) points with the face that contains the point. A precondition is that none of the \( m \) points lie on any of the segments. The planar subdivision is represented using the DCEL structure described in the chapter.

Notice that in contrast to our problem where we found intersections among the segments, \( m \) can be much larger than \( n^2 \), and we cannot assume that \( O(\log_2 (m + n)) \) is necessarily \( O(\log n) \).

Let’s solve the problem using a sweep-line approach.

   (a) Why can you get rid of the priority queue and use an ordered list data type to implement only the sweep line?

   (b) What two kinds of objects does it make sense to keep on the sweep line?

   (c) Can any of these objects appear in more than one place on the sweep line’s ordered list?

   (d) When the sweep line reaches the top of two line segments that share an upper endpoint, how can you identify the face that lies between them in \( O(1) \) time?

   (e) What is the best time bound you can get for the algorithm implied by your answers to these questions? Express your answer in terms of \( n \) and \( m \).

3. I gave a geometric explanation of the dual problem for linear programming.

Let us assume no degeneracies, such as more than \( n \) constraints meeting at a vertex of the polyhedron.

I restated the goal of the dual problem as that of finding a weighted sum of the constraints that define an optimum vertex of the polyhedron, \textit{where all the weights are positive, and such that the weighted sum is a constraint plane that is perpendicular to the gradient of the objective function and opposing it.} Since it is perpendicular to the gradient, the value of the objective function is equal everywhere on the constraint plane, which makes it easy to evaluate, since this can be done at any point on the plane. The evaluation of the objective function at the optimum has an easy geometric interpretation in these terms. Since the constraint plane is a weighted sum of constraint planes at an optimum vertex, it goes through that vertex. Finally, the
requirement that all weights must be positive and that the inequalities of the con-
straints are $\leq$ means that this constraint plane does not go through the interior of
the polygon and that the inequality is opposite to the gradient. This means that the
vertex of the polygon that it contains is an optimum one.

I showed an example in class where everything worked out nicely.

On page 58, the dual variables are determined to be $y_1 = 11$, $y_2 = 0$, $y_3 = 6$.
This corresponds to 11 times the first constraint and 6 times the third constraint on
page 54. Adding these, we get $5x_1 + x_2 + 7x_3 + 3x_4 \leq 29$. Sure enough, 29 is the
optimum value of this problem. However, this constraint is not perpendicular to the
gradient of the objective function. If the author and I are both right, shouldn’t it be
$4x_1 + x_2 + 5x_3 + 3x_4 \leq 29$?

I came up with this geometric interpretation when preparing one of your lectures.
Maybe I’m wrong. Try to resolve the discrepancy in the two views, or show that I
fell into a trap.

(If solve this, you will have a nice understanding of Theorem 5.2. You will also have
a nice understanding of Theorem 5.5 and the economic interpretation immediately
above it.)

4. This question is meant to encourage you to review techniques you’ve learned this
semester.

The connections between computers in a large network of computers forms a graph,
where the vertices are computers and the edges are the direct connections. To route
messages between computers, a system administrator found a BFS tree, and when
computer $i$ sends a message to computer $j$, it routes it using the unique path between
$i$ and $j$ in the BFS tree when it is treated as an undirected tree.

A customer has complained that his computer is slow because his computer has high
degree in the undirected tree, so it spends a lot of time passing other people’s packets
that get routed through his computer. As a consultant, you are called in to figure out
an algorithm for finding a fair tree, that is, one that minimizes the maximum degree.

See if you can make progress on an algorithm for this problem using any of the things
you have learned in the course, such as max flow using the integrality constraint,
the ability to find maximum matchings efficiently in a graph that is not necessarily
bipartite, the linking-and-cutting tree ADT, linear programming, heaps, etc.