CMSC 451
Summer 2006

Homework 3

Julián Mestre

Due on Wednesday 9

Problem 1: Let $G = (V, E)$ be an undirected graph and $w : E \to R$ a weight function on the edges. We want test if G contains a negative cycle using the following algorithm:

```
HAS-A-NEGATIVE-CYCLE(G)

do DFS
For each non-tree edge e = (u, v)
  let C be the cycle defined by (u, v) and
    the unique path from u to v in the DFS tree
  If w(C) < 0
    return TRUE
return FALSE
```

Prove or disprove the correctness of this algorithm.

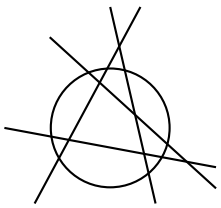
Problem 2: Strassen's algorithm for fast matrix multiplication algorithm runs in $O(n^{\log_2 7})$ time. That is much better that the naive $O(n^3)$ algorith, isn't it? Count the *exact number* of multiplications and additions of two numbers done by the naive algorithm; do the same for Strassen's algorithm by unrolling the recursion tree. You may assume n is a power of 2. Find a threshold value n_0 such that Strassen's performs less operations than the naive algorithm for all $n \geq n_0$. Assuming your computer can carry out 1 million operations per second, how long would it take to multiply two matrices of size n_0, $10\,n_0$, $100\,n_0$ and $1000\,n_0$?

Problem 3: In the previous homework you had to design an algorithm for finding maximum bandwidth paths. One way to do that is to find a maximum weight spanning tree T; for every pair of vertices u and v the tree path $T_{u,v}$ is a maximum bandwidth path.

Let $N(u, v)$ be the next vertex after u in $T_{u,v}$ (you can think of $N(u, *)$ as u's routing table); also let $B(u, v)$ be the bandwidth of $T_{u,v}$. Design an algorithm that given T computes the matrices N and B in the graph in $O(n^2)$ time.

Problem 4: From the book: Exercise 5.1.

Problem 5: From the book: Exercise 5.5.



Problem (extra credit): Here is another problem from computational geometry. You are given a collectionn of lines $\ell_1, \ell_2, \ldots, \ell_n$ on the plane, for simplicity assume no two lines are parallel to each other and no three lines intersect on the same point. We are interested in counting the number of intersections between these lines. By itself this problem is rather easy (how many intersections are there?) To make things more interesting let's add another ingredient to the picture: a circle. Design an algorithm that given a set of lines and a circle computes the number of instersections *inside* the circle in $O(n \log n)$ time. For example, in the figure on the right although the total number of intersections is six only four ocurr inside the circle. You may assume that the intersection between two lines or a line and the circle can be computed in constant time.