Algorithms
Computer Science 140 & Mathematics 168
Spring 2007
Homework 12a (OPTIONAL)
Due Friday, April 27 by 5 PM in the CS Office

This homework is ENTIRELY optional. It is worth 70 bonus points. Please note that it is due at an unusual time: Friday, April 27 at 5 PM.

This problem set is all about the k-server problem described in class. Recall that in this problem we are given k "servers" (or robots) in a metric space. You don't need to worry too much about the details of metric spaces other than to recall that metric spaces satisfy the triangle inequality.

A request sequence is a *finite* sequence of points in our metric space. We use $S = p_1 p_2 \ldots p_n$ to represent a sequence of n points. The points are requests for service. When a point request arrives, we must move at least one server to that point. An online algorithm, ALG, is one that services the current request before the next request is given.

For a request sequence S, $ALG(S)$ denotes the total distance travelled by all robots in servicing the requests in S using online algorithm ALG. Similarly, $OPT(S)$ denotes the total distance travelled by all of the robots in an optimal *offline* algorithm.

We say that an online algorithm ALG is α-competitive if

$$ALG(S) < \alpha \times OPT(S) + C$$

for all finite request sequences S and some fixed constant C.

In class we defined a particular online algorithm, called DC, for the k-server problem on the real line. This algorithm works as follows: If a request is outside of the convex hull of the k servers, dispatch the server closest to that request (if there are multiple servers at the same point, dispatch just one of them). Otherwise, the request is inside the convex hull. In this case, dispatch the two robots immediately to the left and right of the request. Move those two robots at equal speed until one reaches the request. The second robot will stop short of the request, unless the two robots were equidistant from the request.

1. **[10 Points OPTIONAL Bonus Problem] Laziness is a Virtue!** Before looking more closely at DC, let's take a close look at a property called "laziness". An algorithm (online or offline) for dispatching

servers is called "lazy" if it has the following property: On each request, only the one robot that will service this request moves. All other robots remain stationary. Notice that the DC algorithm described above is **NOT** lazy.

Prove that for any algorithm A, there exists a corresponding lazy algorithm A' such that $A'(S) \leq A(S)$ for all S. In other words, laziness is a virtue!

2. **[30 Points OPTIONAL Bonus Problem] DC is k-competitive!** Prove that DC with k servers is k-competitive. You'll need to use the observations that we made in class.

3. **[30 Points OPTIONAL Bonus Problem] DC-Tree!** You've been hired by `Nile.com`, a new online bookstore. The floor of a Nile warehouse has tracks embedded in the floor and a collection of k robots move along these tracks picking up books, etc. The network of tracks form a rectilinear tree: There are no cycles and each piece of track is straight. Notice that the tracks are continuous so a robot can be located at any point along the track. In other words, any (graph theoretic) tree in which the edges are drawn as straight line segments forms a legitimate network of track except that the robots can be located anywhere along any edge of the tree.

   Generalize the DC algorithm for the real line to an algorithm for rectilinear trees. Describe the algorithm very carefully. Then show that your algorithm is still k-competitive. How cool is that!?!