CS 499 (Fall 2006)- Assignment 6
Due: Thursday, 10/12/2006

(1) Here, we explore a variation of the problem of finding good centers. The setup is the same as in Homework 1, except that we want to minimize the *maximum* distance from any node to its closest selected center (as opposed to the average). Specifically, your program should get as input (1) the possible locations L of centers, (2) the locations R that need to be served, and (3) the total number k of centers that you are allowed to open. Your goal is to find a subset $S \subseteq L$ with $|S| \leq k$ such that $\max_{i \in R} d(i, S)$ is as small as possible. (Remember that $d(i, S)$ is the minimum distance from i to its closest neighbor in S.)

In general, this problem is NP-hard. What your program is supposed to do is the following:

1. Determine if all locations lie on a line (Hint: the code given in a past assignment may be useful).
2. If they all lie on a line, then use dynamic programming to be able to solve fairly large instances.
3. If they are not on a line, use exhaustive search if it will terminate in about a minute or less.
4. Otherwise, instead of solving the problem, give the user an estimate of how long roughly it would take to solve it. Make the output somewhat legible, e.g., "106 years, 5 months" instead of "3358182871 seconds". Obviously, when the running time is in the years, you can omit the seconds and minutes and hours, and so on. (For this part, you can use the running time on your own machine. Presumably, you will want to run some tests, to see how the running time scales with various parameters. From that, you can extrapolate the running time for larger instances.)

(2) Think about heuristics that you think would be helpful for solving this problem when instances are too large. For the heuristics that you come up with, try to give an informal argument why they may be good. Also try to see whether you can draw examples in 2-D (or higher dimensions, if you prefer) where your heuristics don't do well. How badly do they seem to do compared to the best possible solution on your instances?

(Notice: there is a lot of research on this problem, some of it fairly deep. I don't expect you to come up with anything comparable to state-of-the-art here. On the other hand, I would also like to ask you not to go out and just tell me what other people have done. A bad heuristic that you came up with yourself is much more welcome than a photo-copy of the latest STOC paper on this topic.)