Problem Set 2

Due: March 1

The goal of this problem set is to help you understand how competitive algorithms work. We will consider the game from class. A *request* sequence is a series $r_1, ..., r_t$ of integers between 1 and 100. You have two servers that initially are both at 1. At each request you must move at least one of the servers to the requested point. The cost of the move is the distance, i.e. if the server is at 45 and the request is for 48 the cost is 3. The goal is to minimize the total cost.

1. You are to write a routine that computes the cost of the "optimal" sequence of moves. Thus, given the request sequence $r_1, ..., r_t$ this routine computes $OPT(r_1, ..., r_t)$ the cost of the optimal moves.
Hint:compute first the following function: $C(r_1, ..., r_t, k)$ which is defined as the cost of the optimal sequence of moves that handles the requests $r_1, ..., r_t$ and leaves the other server at k. Note, one server by definition is at r_t at the end. The reason that this is useful is that it satisfies a simple recursion:

$$C(r_1, ..., r_t, k) = min_l C(r_1, ..., r_{t-1}, l) + M$$

where M is the cost of moving the servers from r_{t-1} and l to r_t and k in some order. Then, use $C()$ to compute OPT.

2. Write a routine that computes the cost of $A(r_1, ..., r_t)$ where A is each of the following algorithm.

1. Move the closest server to the request.
2. Move both servers until one hits the request. This is one we discussed in class.
3. Move one server for real and the other virtually. This is also one we discussed in class.

3. Finally, compare the performance of the algorithms to OPT. What happens for random request sequences of length 200? What happens for worst case sequences?