

## CS 570 (Spring 2009) — Final Exam (Takehome)

Unless you are specifically asked to reprove a known fact, you can use all facts from class, homeworks and the textbook without proof. There are five questions on this exam. The only legitimate sources in solving this exam are (1) your textbook and course notes, (2) any handouts provided by us, (3) discussions with the TA or instructor. You cannot discuss the problems with anyone else (whether in the class or not), nor can you look for solutions in other textbooks, on the Internet, or in other sources. The exam is due, in paper, in David Kempe's office no later than 12:00 noon on Tuesday, 05/12/2009. If you submit after that, your exam will automatically be graded as 0.

You can view your exam in David's office on Thursday, 05/14/2009, between 10:00-12:00, and 2:00-5:00. After that, your grade will be considered final. If you cannot make these times yourself, you can designate another student to view your exam for you, but must send an e-mail to David including the name of the student.

G O O D L U C K

(1) [15 points]

You have two robots on an infinite line. Let's say they start at the origin. You have a sequence of requests coming in. Each request  $i$  is identified with a point  $x_i$  on the line. A request  $i$  is served by moving a robot to the point  $x_i$ . As soon as you serve  $x_i$ , the next request  $i+1$  arrives at location  $x_{i+1}$ . Being an online problem, you do not know the future requests until you have served the current one. Your goal is to minimize the sum of distances traveled by the two robots.

(a) [5 points] The simplest *Greedy Algorithm* is the following: always serve the next request  $i$  with the robot currently closest to  $x_i$  (breaking ties arbitrarily). Prove that the Greedy Algorithm is not constant-competitive for any constant.

(b) [10 points] Here is the *Proactive Algorithm*. At any time, let  $L \leq R$  be the positions of the two robots, and  $x_i$  the location of the new request  $i$ . If  $x_i \leq L$ , then the robot at  $L$  serves request  $i$ . If  $x_i \geq R$ , then the robot at  $R$  serves request  $i$ . If  $L < x_i < R$ , then *both* robots start moving toward  $x_i$  simultaneously, at the same speed. As soon as one of them reaches  $x_i$ , it serve the request  $i$ ; the other robot stops moving at that moment as well.

Prove that the Proactive Algorithm is 2-competitive. (Hint: define a suitable potential function.)

(2) [10 points]

For the same problem, assume now that you are given the entire sequence of request locations  $x_1, x_2, \dots, x_n$  ahead of time. (However, they still have to be served in the order  $1, 2, \dots, n$ .) Give and analyze a polynomial-time algorithm to find the optimum sequence of robot moves to serve all the requests in order.

(3) [10 points]

Now, let's make the robot problem a bit more realistic. Each request  $i$  also comes with a time  $t_i$ : the time it takes to fulfill request  $i$  (once the robot reaches the point  $x_i$ ). In addition, we now also care about the travel time, and assume that each of the robots can travel a distance of  $v$  within one unit of time, so it takes time  $d/v$  to travel distance  $d$ . In return, we assume that the requests can now be served *in any order*.

The input thus consists of  $n$  pairs  $(x_i, t_i)$  and the number  $v$ . (We still assume that the robots start at the origin.) The goal of the robots is to jointly finish the requests as quickly as possible, i.e., minimize the time by which all requests have been serviced.

Phrase this problem as a decision problem, and prove that it is NP-complete.

(4) [15 points]

Recall the KNAPSACK problem: you are given  $n$  items  $i = 1, \dots, n$  with *weights*  $w_i \geq 0$  and *values*  $v_i \geq 0$ , and a total weight  $W$ . Without loss of generality, we assume that  $w_i \leq W$  for all  $i$ . The goal is to find a set  $S$  with  $\sum_{i \in S} w_i \leq W$ , and maximizing  $\sum_{i \in S} v_i$ .

- (a) Write the KNAPSACK problem as an integer LP.
- (b) Use LP-rounding on your LP to obtain a  $\frac{1}{2}$ -approximation for KNAPSACK. (Note: there are some simple greedy algorithms that give a  $\frac{1}{2}$ -approximation as well. Here, you are supposed to use LP-rounding, though.)
- (c) Prove that the integrality gap of the LP can be arbitrarily close to  $\frac{1}{2}$ .
- (d) Show how to compute an optimal solution to the (fractional) LP in time  $O(n \log n)$ , without actually using an LP solver.

(5) [10 points]

Imagine that you want to deploy  $n$  sensors to monitor large wildfires in a park. More specifically, for each of  $d$  days, you want to have an estimate of what fraction of the park is currently burning. When you deploy a sensor, we assume that it will always accurately report whether it is inside a fire or not. However, it has no geographical information. So all you receive is a count of how many of your sensors are currently in the burning area of the park.

We said above that you want to monitor “large” wildfires. By that, we mean that you only need an accurate estimate of the burning area if that area is at least an  $\epsilon$  fraction of the total park area. Whenever the burning area is less than that, it does not matter how accurately you report it.

Given that you don’t know ahead of time which areas of the park will be burning, it is hard to know how to deploy the sensors. One simple strategy, though, is to place them independently and randomly over the entire park. We assume that which areas will burn is determined before you place your sensors, i.e., there will not be an adversary reacting to your random choices.

Prove that by placing  $n = O(k \cdot 1/\epsilon \cdot \ln d)$  sensors in this way, you can get an estimate accurate to within  $\pm 10\%$  (for all  $d$  days simultaneously), with probability at least  $1 - 1/d^k$ .