

# CS 573: Graduate Algorithms, Fall 2008

## Homework 3

Due at 11:59:59pm, Wednesday, October 22, 2008

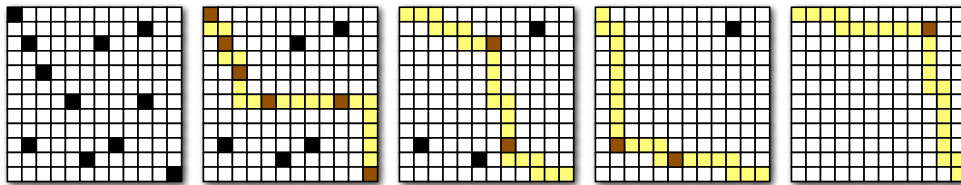
- Groups of up to three students may submit a single, common solution. Please neatly print (or typeset) the full name, NetID, and the HW0 alias (if any) of every group member on the first page of your submission.

1. Consider an  $n \times n$  grid, some of whose cells are marked. A *monotone* path through the grid starts at the top-left cell, moves only right or down at each step, and ends at the bottom-right cell. We want to compute the minimum number of monotone paths that cover all marked cells. The input to our problem is an array  $M[1..n, 1..n]$  of booleans, where  $M[i, j] = \text{TRUE}$  if and only if cell  $(i, j)$  is marked.

One of your friends suggests the following greedy strategy:

- Find (somehow) one “good” path  $\pi$  that covers the maximum number of marked cells.
- Unmark the cells covered by  $\pi$ .
- If any cells are still marked, recursively cover them.

Does this greedy strategy always compute an optimal solution? If yes, give a proof. If no, give a counterexample.



Greeditly covering the marked cells in a grid with four monotone paths.

2. Let  $X$  be a set of  $n$  intervals on the real line. A subset of intervals  $Y \subseteq X$  is called a *tiling path* if the intervals in  $Y$  cover the intervals in  $X$ , that is, any real value that is contained in some interval in  $X$  is also contained in some interval in  $Y$ . The *size* of a tiling path is just the number of intervals.

Describe and analyze an algorithm to compute the smallest tiling path of  $X$  as quickly as possible. Assume that your input consists of two arrays  $X_L[1..n]$  and  $X_R[1..n]$ , representing the left and right endpoints of the intervals in  $X$ . If you use a greedy algorithm, you must prove that it is correct.



A set of intervals. The seven shaded intervals form a tiling path.

3. Given a graph  $G$  with edge weights and an integer  $k$ , suppose we wish to partition the vertices of  $G$  into  $k$  subsets  $S_1, S_2, \dots, S_k$  so that the sum of the weights of the edges that cross the partition (i.e., that have endpoints in different subsets) is as large as possible.
- Describe an efficient  $(1 - 1/k)$ -approximation algorithm for this problem. [Hint: Solve the special case  $k = 2$  first.]
  - Now suppose we wish to minimize the sum of the weights of edges that do *not* cross the partition. What approximation ratio does your algorithm from part (a) achieve for this new problem? Justify your answer.
4. Consider the following heuristic for constructing a vertex cover of a connected graph  $G$ : **Return the set of all non-leaf nodes of any depth-first spanning tree.** (Recall that a depth-first spanning tree is a *rooted* tree; the root is not considered a leaf, even if it has only one neighbor in the tree.)
- Prove that this heuristic returns a vertex cover of  $G$ .
  - Prove that this heuristic returns a 2-approximation to the minimum vertex cover of  $G$ .
  - Prove that for any  $\varepsilon > 0$ , there is a graph for which this heuristic returns a vertex cover of size at least  $(2 - \varepsilon) \cdot OPT$ .
5. Consider the following greedy approximation algorithm to find a vertex cover in a graph:

```

GREEDYVERTEXCOVER( $G$ ):
   $C \leftarrow \emptyset$ 
  while  $G$  has at least one edge
     $v \leftarrow$  vertex in  $G$  with maximum degree
     $G \leftarrow G \setminus v$ 
     $C \leftarrow C \cup v$ 
  return  $C$ 

```

In class we proved that the approximation ratio of this algorithm is  $O(\log n)$ ; your task is to prove a matching lower bound. Specifically, for any positive integer  $n$ , describe an  $n$ -vertex graph  $G$  such that  $\text{GREEDYVERTEXCOVER}(G)$  returns a vertex cover that is  $\Omega(\log n)$  times larger than optimal. [Hint:  $H_n = \Omega(\log n)$ .]

- \*6. [Extra credit] Consider the greedy algorithm for metric TSP: Start at an arbitrary vertex  $u$ , and at each step, travel to the closest unvisited vertex.
- Prove that this greedy algorithm is an  $O(\log n)$ -approximation algorithm, where  $n$  is the number of vertices. [Hint: Show that the  $k$ th least expensive edge in the tour output by the greedy algorithm has weight at most  $OPT/(n - k + 1)$ ; try  $k = 1$  and  $k = 2$  first.]
  - \*Prove that the greedy algorithm for metric TSP is no better than an  $O(\log n)$ -approximation. That is, describe an infinite family of weighted graphs that satisfy the triangle inequality, such that the greedy algorithm returns a cycle whose length is  $\Omega(\log n)$  times the optimal TSP tour.