Computer Science 142 & Mathematics 167
Theory of Computation
Fall 2000
Homework 11 Lite
Due Wednesday, December 6

1. **[35 Points] The Complexity of the Alpha Game!** In this problem we will examine two variants of a computer game called "Alpha". The first version of the game is a one-player game and the second version is a two-player game.

   (a) The one-player version of "Alpha" works as follows. We first specify an alphabet (English alphabet, Arabic alphabet, binary alphabet, etc.) The game uses a grid with m rows and n columns. The computer puts zero or more characters from the alphabet in each cell in the grid. The same character may appear in multiple grid cells. Below the grid are n spaces or "player cells" for the player to fill in as explained below. Here is a picture of a game using a 3 by 4 grid and the English alphabet.

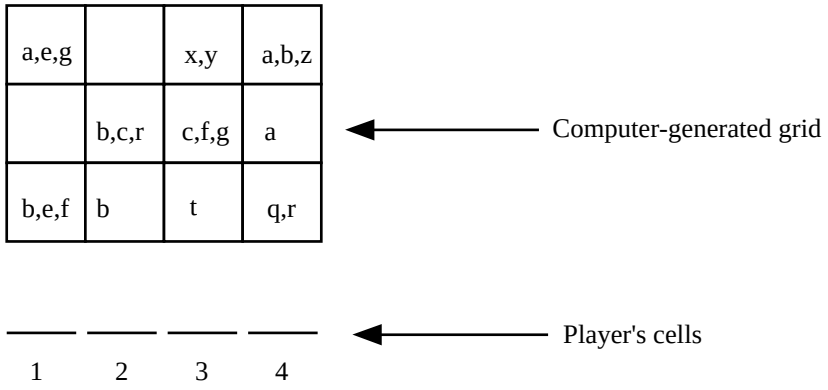

Figure 1: A sample "Alpha" Game.

The player's objective is to write one symbol from the alphabet in each of the n player cells. The n symbols written in these n cells must have the following property: For each row i in the computer-generated grid, there exists some column j such that cell (i, j) contains a symbol that matches the symbol chosen by the player in player cell j. For example, for the game above, the player could choose the string "ebgz". This satisfies the constraints of the game for the following reason: For row 1 of the game board, column 1 contains an "e" and there is an "e" in position 1 of the player cells. For row 2 of the game board, column 3 contains a "g" and player cell 3 contains a "g" as well. Finally, for row 3 of the game board, column 1 contains an "e" and position 1 of the player cells contains an "e".

The **One-Player Alpha Problem (OPAP)** is the following: Given an alphabet Σ and a $m \times n$ game board with each cell containing zero or more characters from

the alphabet, does there exist a string that the player can write in the player cells that permits her to "win" the game (in the sense that the game constraints above are satisifed)? Prove that OPAP is NP-complete. *Hint: A reduction from 3SAT works well.*

(b) The two-player version of Alpha works as follows: We are again given a computer-generated grid and player cells as shown in the figure. Player one begins by writing a symbol in player cell 1. Player 2 then writes a symbol in player cell 2, followed by player 1 writing a cell in player cell 3, etc. Player 1 wins if the final string written in the player cells satisfies the matching properties described above. Otherwise Player 2 wins. The **Two-Player Alpha Problem (TPAP)** is the following: Given an alphabet Σ and a $m \times n$ game board with each cell containing zero or more characters in the alphabet, does there exist a winning strategy for Player 1? Prove that TPAP is PSPACE-complete.

2. **[15 Points] Tightness of Approximation Ratios.** In class we showed a polynomial time approximation algorithm for the Travelling Salesman Problem with Triangle Inequality. (TSPTI) The ratio bound for this algorithm was also shown to be at most 2. In this problem you will show that the ratio of 2 is tight. In particular, show that for any $\delta > 0$ it is possible to construct an instance of TSPTI such that the approximation algorithm finds a solution that is at least $2 - \delta$ times larger than optimal. *Hint: Construct an instance of TSPTI where the points are arranged as shown below and use the Euclidean distance metric. This is convenient since the Euclidean distance metric satisfies the triangle inequality.*

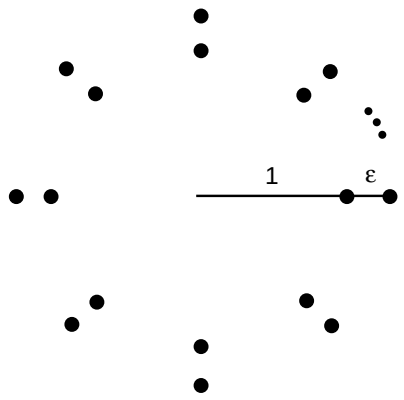

Figure 2: An arrangement of points on two concentric circles with radii 1 and $1 + \epsilon$.

3. **[20 Points] The Disk Storage Problem!** First the gratuitious story. You have been hired by Moon Microsystems to work on their new operating system, Lunaris. When Lunaris does a backup, it typically uses two large backup disks and a tape drive. Since disks have lower access time than tape, it is desirable to store as many files as possible on disk and the remainder will go on tape. Let $F = \{f_1, \ldots, f_n\}$ denote the n

files to be backed up and let ℓ_i denote the length of file i. Without loss of generality, let $\ell_1 \leq \ell_2 \leq \ldots \leq \ell_n$.

There are two identical disks, each with storage capacity of L. A file stored on disk cannot be broken up - it must be stored entirely on one of the two disks. The Disk Storage Optimization Problem is to store the maximum number of files from F onto the disks.

Your boss has suggested that you implement a greedy algorithm for the Disk Storage Optimization Problem: Since the files in F appear in non-decreasing order of size, simply go through F from shortest file to longest file, putting as many of the files on the first disk as possible. When the first disk fills up, continue iterating through F putting as many of the remaining files as possible on the second disk.

(a) Give an example that demonstrates that the greedy algorithm does not necessarily find an optimal solution. That is, give a set of specific file sizes, show the solution found by the greedy algorithm, and then the optimal solution. Show that the solution found by the greedy algorithm is not optimal.

(b) State the decision problem corresponding to this optimization problem. Then prove that the decision problem is NP-complete. Use a reduction from one of: Vertex Cover, 3SAT, Partition, Clique, or Undirected Hamiltonian Cycle.

(c) Prove that the greedy algorithm is an *absolute* approximation algorithm which finds solutions that are at most 1 larger than optimal.