Assignment #2

Greg Plaxton

September 12, 2012

NOTE ADDED 9/12/12: This version of the assignment corrects two errors that existed in the original version handed out in class. Both of these errors occurred in the paragraph preceding Section 1.1. Most importantly, the edge set of $trade(G, C)$ was incorrectly defined in the original version. End of NOTE.

NOTE ADDED 9/18/12: The definition of $trade(G, C)$ referred to in the preceding paragraph was not right for the case of a cycle of length 2. Fortunately we only care about this definition for cycles C of length greater than 2. Nevertheless I have repaired the definition to work for any cycle C. End of NOTE.

There are two parts to this assignment. The first part is described in Section 1 below, and may be done with a partner. If you are having trouble finding a partner, send me an email and I will match you up with someone else in the same situation.

The second part, described in Section 2 below, consists of textbook exercises, and is due at the beginning of class on Friday, September 28. Each student should work on this part separately.

Any corrections or clarifications related to the assignment will be posted in the *Assignments* section of the class website.

1 Programming & Problem Solving: Mooov Around

This part of the assignment has two subparts: a paper-and-pencil subpart involving proofs (see Section 1.2) and a programming subpart (see Section 1.3).

Our goal is to develop a polynomial-time algorithm for the special case of the Mooov Around problem in which each student has *strict preferences*: For any student i, and any spaces j and j' such that $j \neq j'$, either i prefers j to j' or i prefers j' to j.

As in Assignment 1, we let n denote the number of students/spaces. For any student i, we define $top(i)$ as i's most preferred space, and we define $bottom(i)$ as i's least preferred space. For any student i and any space j not equal to $bottom(i)$, we define $next(i, j)$, as the next-most-preferred space of student i after space j.

We define a *configuration* $G = (U, V, E)$ as a bipartite graph with a set U of vertices on the "left", one for each student, a set V of vertices on the "right", one for each space, and a set E of directed edges satisfying the following constraints: (1) each edge goes from a

student to a space, or from a space to a student; (2) each vertex has exactly one outgoing edge; (3) each student vertex has exactly one incoming edge. The n space-to-student edges encode an allocation, as follows: If the outgoing edge of space j goes to student i, then space j is allocated to student i.

The *initial configuration* has an edge from each student i to space $top(i)$, and an edge from each space j to student j. Thus the space-to-student edges in the initial configuration encode the university allocation.

For any configuration G and student i, we define $space(G, i)$ as the space reached by following the unique outgoing edge from student i.

For any configuration G, we define $exhausted(G)$ as the set of all students i such that i does not belong to a cycle of length 2 in G and $space(G, i)$ belongs to a cycle of length 2 in G. (Remark: Throughout this assignment, we use the term "cycle" to refer to a directed cycle.)

For any student i in $exhausted(G)$ such that the space $j = space(G, i)$ is not equal to $bottom(i)$, we say that a *revelation action is enabled for student i in G*. The effect of performing a revelation action that is enabled for student i in G is to replace the current configuration $G = (U, V, E)$ with the configuration $reveal(G, i) = (U, V, E')$ where $E' = E - (i, j) + (i, next(i, j))$.

For any cycle C of length $2k$ in $G = (U, V, E)$, we can partition the edges of C into a set E_0 of k student-to-space edges, and a set E_1 of k space-to-student edges. Letting E'_0 denote the set of k space-to-student edges corresponding to the reversals of the edges in E_0, we define $trade(G, C)$ as the configuration $G' = (U, V, (E \setminus E_1) \cup E'_0)$. For any cycle C of length greater than 2 in G, we say that a *trading action is enabled for C in G*. The effect of performing a trading action that is enabled for C in G is to replace the current configuration G with $trade(G, C)$.

1.1 Algorithm A

In this assignment we study the following nondeterministic algorithm, referred to as Algorithm A, that starts from the initial configuration, and iteratively updates the current configuration by performing an arbitrary enabled action until a configuration is reached in which no actions are enabled. Once the algorithm terminates, the output allocation is defined by the space-to-student edges in the final configuration.

1.2 Proofs

In this subpart of the assignment you will prove a subset of the eight lemmas stated below. When proving a given lemma, you are allowed to make use of any lower-numbered lemma, but you are not allowed to make use of a higher-numbered lemma.

Each team is required to turn in proofs for any two of Lemmas 1, 2, 6, and 7 (these lemmas are relatively easy to prove), and any two of Lemmas 3, 4, and 5 (these lemmas are

a bit harder). Lemma 8 is somewhat more challenging, and will be graded as an optional bonus problem.

Your solutions to this subpart are due at the beginning of class on Friday, September 21.

Lemma 1: If Algorithm A reaches a configuration containing a cycle C of length 2, then all subsequent configurations reached by Algorithm A contain C.

Lemma 2: If Algorithm A reaches a configuration G containing an edge from student i to space j, and j' is a space that i prefers to j, then j' belongs to a cycle of length 2 in G.

Lemma 3: Assume that Algorithm A reaches a configuration G containing edge $(i, \mathit{bottom}(i))$ for some student i. Then no actions are enabled in G and every vertex belongs to a cycle of length 2.

Lemma 4: Algorithm A terminates after performing $O(n^2)$ revelation actions and $O(n)$ trading actions.

Lemma 5: If G is a configuration such that some vertex does not belong to a cycle of length 2, then at least one action is enabled in G.

Lemma 6: If Algorithm A terminates in configuration G, then every vertex belongs to a cycle of length 2 in G.

Lemma 7: The output allocation produced by Algorithm A is individually rational.

Lemma 8: The output allocation produced by Algorithm A is strictly Pareto-efficient.

1.3 Programming Task

Implement Algorithm A. Your implementation is required to have polynomial worst-case time complexity. Use the same input-output format as in Assignment 1, bearing in mind that you will only be asked to process inputs with strict preferences.

Your program is due by 8pm on Friday, September 21.

2 Textbook Exercises

1. Problem 1.3, page 22.
2. Problem 1.8, page 27.
3. Problem 3.10, page 110.
4. Problem 3.12, page 112.