Assignment #4

Greg Plaxton

October 17, 2012

There are two parts to this assignment. The first part is described in Section 1 below, and may be done with a partner. The proofs associated with the first part are due at the beginning of class on Wednesday, October 24. The second part, described in Section 2 below, consists of textbook exercises, and is due at the beginning of class on Wednesday, October 31. Each student should work on the second 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

In Assignment 2 we developed a polynomial-time nondeterministic algorithm — referred to as Algorithm A — for the special case of the Mooov Around problem in which preferences are strict. In Assignment 3 we presented an $O(n^2)$-time determinization of Algorithm A. The main objective of the current assignment is to establish that Algorithm A is *strategyproof*: A student never gets a better space by lying about his or her preferences. In order to streamline our proof of this claim, we find it convenient to modify our formal framework a little bit. In Section 1.1 below we provide slightly revised definitions for several of the technical terms introduced in Assignment 2. With these revised definitions, we obtain a minor variant of Algorithm A — referred to as Algorithm B — which is defined in Section 1.2. The sequence of lemmas that you are asked to prove in Section 1.3 culminates with a lemma stating that Algorithm B is strategyproof. Due to the close relationship between Algorithm A and Algorithm B, it will be apparent that these two nondeterministic algorithms have the same input-output behavior; it follows that Algorithm A is also strategyproof.

1.1 A Revised Framework

Previously, when we defined a configuration, we required each student vertex to have outdegree one. In the revised sequence of definitions given below, we relax this constraint by allowing a student vertex to have outdegree zero.

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 space vertex has exactly one outgoing edge; (3) each student vertex has at most one outgoing edge and 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 space j to student j, and has no outgoing edges from any of the student vertices. Thus the space-to-student edges in the initial configuration encode the university allocation.

For any configuration G and student i, we define $\mathit{space}(G,i)$ as follows. If student i has outdegree zero in G, then $\mathit{space}(G,i)$ is defined to be *nil*. If student i has outdegree one in G, then $\mathit{space}(G,i)$ is defined as the space reached by following the unique outgoing edge of student i.

For any configuration G, we define $\mathit{exhausted}(G)$ as the set of all students i such that the following two conditions hold: (1) i does not belong to a cycle of length 2 in G; (2) either $\mathit{space}(G,i)$ is *nil* or $\mathit{space}(G,i)$ belongs to a cycle of length 2 in G.

For any configuration G and any student i in $\mathit{exhausted}(G)$ such that the $\mathit{space}(G,i)$ is not equal to $\mathit{bottom}(i)$, 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 as follows. If $\mathit{space}(G,i)$ is equal to *nil*, then the effect is to replace the current configuration $G=(U,V,E)$ with the configuration $\mathit{reveal}(G,i)=(U,V,E')$ where $E'=E+(i,\mathit{top}(i))$. If $\mathit{space}(G,i)$ is equal to a space j, then the effect is to replace the current configuration $G=(U,V,E)$ with the configuration $\mathit{reveal}(G,i)=(U,V,E')$ where $E'=E-(i,j)+(i,\mathit{next}(i,j))$.

The following definitions related to trading actions are the same as in Assignment 2; they are reproduced here for the sake of convenience only. 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 $\mathit{trade}(G,C)$ 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 $\mathit{trade}(G,C)$.

1.2 Algorithm B

Algorithm B starts from the initial configuration, and iteratively updates the current configuration by performing a nondeterministically chosen 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.

Remarks: One way that an execution of Algorithm B can begin is by performing one revelation action for each student. Once this has been done, the current configuration corresponds to the initial configuration of Algorithm A as defined in Assignment 2. Furthermore, each student vertex has outdegree one, and continues to have outdegree one thereafter. It is easy to see that the the remainder of the execution of Algorithm B corresponds to an execution of Algorithm A.

1.3 Proofs

Each team is required turn in three proofs related to the five lemmas stated in this section: (1) a proof of Lemma 1 or 2; (2) a proof of Lemma 3 or 4; (3) a proof of Lemma 5. When proving a given lemma, feel free to make use of any lower-numbered lemma.

In Assignment 2 we proved that Lemmas 1 to 8 of Assignment 2 hold for Algorithm A. Using the same arguments, it can be shown that Lemmas 1 to 8 of Assignment 2 hold for Algorithm B under the revised framework of Section 1.1. Feel free to make use of these lemmas in your proofs.

Remark: It is trivial to modify the $O(n^2)$-time determinization of Algorithm A that we analyzed in Assignment 3 to obtain an $O(n^2)$-time determinization of Algorithm B. End of Remark.

It is not difficult to prove that the following two facts hold. Feel free to make use of these facts in your proofs.

Fact 1. If two actions α and β are enabled in some configuration G, and G' is the configuration reached from G by performing action α, then action β is enabled in G'.

Fact 2. If two actions α and β are enabled in some configuration G, then the configuration reached from G by performing action α followed by action β is equal to the configuration reached from G by performing action β followed by action α.

Lemma 1. Prove that for any given initial configuration G, the final configuration reached by Algorithm B is unique, i.e., it does not depend on the nondeterministic choices made by Algorithm B.

For any initial configuration G, let $\mathit{sink}(G)$ denote the corresponding unique final configuration given by Lemma 1.

We say that an enabled action "involves student i" if it is either a revelation action enabled for student i or a trading action enabled for a cycle that includes student i.

Lemma 2. Let $G = (U, V, E)$ be an initial configuration, and let i be a student in U. Suppose that we repeatedly perform a nondeterministically chosen action that does not involve student i until we reach a configuration in which every enabled action involves student i. Then the configuration reached in this process is unique, i.e., it does not depend on the nondeterministic choices made.

For any initial configuration $G = (U, V, E)$ and any student i in U, let $\mathit{sink}(G, i)$ denote the unique configuration reached by the process associated with Lemma 2.

For any initial configuration $G = (U, V, E)$ and any student i in U, let $\mathit{reach}(G, i)$ denote the set of all spaces j in V such that there is a directed path from j to i in configuration $\mathit{sink}(G, i)$.

Lemma 3. Let $G = (U, V, E)$ be an initial configuration, let i be a student in U, and let j be a space in V. Prove that exactly one of the following two conditions holds: (1) j belongs to a cycle of length 2 in $\mathit{sink}(G, i)$; (2) j belongs to $\mathit{reach}(G, i)$.

Lemma 4. Let $G = (U, V, E)$ be an initial configuration, let i be a student in U, and let j be the space allocated to student i in the configuration *sink*(G). Then j is student i's most-preferred space in *reach*(G, i).

Lemma 5. Algorithm B is strategyproof.

2 Textbook Exercises

1. Problem 6.5, page 316.
2. Problem 6.8(b), page 319. Clarification: The only robots eligible to be destroyed in second i are the ones that arrive in second i. Thus at most x_i robots can be destroyed in second i.
3. Problem 6.14, page 324.
4. Problem 7.3, page 415.
5. Problem 7.5, page 416.