CPSC 320 Sample Solution, The Stable Matching Problem, Part II

Here you'll analyze the Gale-Shapley (G-S) algorithm for the Stable Matching Problem, and learn principles for constructing proofs more generally. You've already read the analysis provided in the text. It's best not to look it up now, but see if you can reconstruct the analysis yourself, come up with alternative ways to reason about the algorithm, and uncover new properties of the algorithm.

Algorithm G-S(n,P_E, P_A) // return a stable matching M for the input instance, where:
// $n \geq 1$ is the number of employers and also the number of applicants
// P_E is the collection of complete preference lists of the employers
// P_A is the collection of complete preference lists of the applicants

```
M ← ∅ // matching M is initially empty
While some employer is unmatched and has not considered every applicant
    Choose any such employer e
    Let a be the highest-ranked applicant in e's preference list that e has not yet considered
    // e now considers a (i.e., "makes an offer" to a) as follows:
    If a is unmatched
        Add match e:a to M                  // a accepts e's offer
    Else // a is matched
        Let a be currently matched to e'
        If a prefers e to e'
            Remove match a:e' from M        // a rejects e''s offer
            Add match a:e to M              // a accepts e's offer
    // Else a prefers e' to e, in which case M does not change
Return M
```

1 Correctness Conditions

Write down what it means for an algorithm for the Stable Matching Problem to be correct on a valid iput.

SOLUTION A correct algorithm has the following properties on any valid input:

- The algorithm terminates.
- The output is a valid solution, i.e., a perfect matching.
- The output is a good solution, i.e., a perfect matching with no instabilities.

Note. The answer to this question does not in any way depend on the G-S algorithm. But it's helpful to break down the different aspects of correctness, as a first step in coming up with proofs.

2 The G-S Algorithm Always Terminates

Prove that algorithm G-S terminates on every valid input.

SOLUTION Consider inputs with n employers and n applicants. The algorithm terminates if the condition of the While loop eventually becomes false—that is, when all employers are matched or when all employers have considered all applicants, whichever comes first. On each iteration of the loop, some employer e considers some applicant a that e has not considered in previous iterations. There are n^2 employer-applicant pairs, so the condition of the While loop must become false within n^2 iterations.
Note. This proof uses a very versatile strategy, namely showing that ***progress is made at every iteration of the While loop****. Also, the proof just uses English sentences; it could be written more formally but that's not necessary.*

3 True or False

Choose whether you think each of the following statements about Algorithm G-S is true or false for all valid inputs. Do your best to also write down explanations of your answers. (Get in the practice of explaining your answers to true/false or multiple-choice questions such as this, and always include short explanations in assignments and exams unless explicitly asked not to.) If you don't have convincing justification, just write down your best intuition - some of these may be hard to justify.

1. Every applicant has been considered at least once when the algorithm terminates.

   SOLUTION True. This is easy to see by examining the condition of the While loop, which is false when the algorithm terminates. When the condition is false, one of two things must be true. Either all employers have considered all applicants, in which case certainly every applicant has been considered at least once. Otherwise, all employers must be matched, and so all applicants must be matched. But an applicant can only be matched if it has been considered, and again the statement must be true.

2. Some employer gets its highest-ranked applicant.

   SOLUTION False. Here's an input on which the statement does not hold:

   ```
   e1: a1,a2,a3        a1: e3,e2,e1
   e2: a1,a3,a2        a2: e1,e2,e3
   e3: a2,a1,a3        a3: e2,e1,e3
   ```

   There is an execution of the algorithm in which e1 is initially matched with a1, but a1 later rejects e1 in favour of e2. So e1 is matched with a2, which is not its highest-ranked applicant. Also, a2 rejects e3, so e3 is not matched with its highest-ranked applicant. Then e3 makes an offer to a1 and a1 accepts this offer. As a result, e2 is not matched with its highest-ranked applicant, but is matched with a3.

   Note. When proving that a statement about an algorithm's behaviour on all inputs is false, it's sufficient to show one example on which the statment is false.

3. When the algorithm terminates, each employer is matched with the last applicant it considered.

   SOLUTION True. An employer e certainly cannot be matched with an applicant that it considered before the last applicant, since those applicants must have rejected e. Also, e cannot be matched with applicants that it did not consider. So the only possible applicant that e can be matched with is the last applicant, say a, that e considered.

   The trickier thing to establish is that e is not also rejected by a, in which case e is matched with no applicant. It's helpful to **take a different tack** and establish another true fact about the algorithm. Namely, that **all applicants must be matched when the algorithm terminates**.

   To see why, note that if an applicant a is unmatched when a is considered, then a becomes matched. If a is already matched, say to employer e and is considered again, say by e', then a either remains matched with e or is rematched with e'. Either way, a remains matched by the end of the iteration. We've already shown above that every applicant is considered at least once by the algorithm and so all applicants must be matched when the algorithm terminates.

   Let's go back to our original goal of showing that e is matched with the last applicant a that e considers. We now know that all applicants are matched when the algorithm terminates. Moreover, each applicant is matched with just one employer, since matched applicants always reject one employer before accepting an offer with another. So, **all employers must be matched when the algorithm terminates**. Therefore, since e must be matched when the algorithm terminates, and e cannot be matched with any applicant other than the last applicant a that it considers, it must be that e is matched with a.

   *Note: There are two things to take away from this. First, showing half of the proof is quite straightforward (namely that e can't be matched with any applicant that is **not** the last applicant that e considers). Always look for the easier parts even if you can't complete the proof. Second, when things get tricky, try a different tack: find other things to prove that can ultimately be helpful.*

4 The G-S Algorithm Always Outputs a Valid Solution

Prove that the G-S algorithm always outputs a valid solution, i.e., a perfect matching.

SOLUTION We've already established above that when the algorithm terminates, every employer is matched with exactly one applicant. Therefore, the matching is a perfect matching, and thus valid.

5 The G-S Algorithm Always Outputs a Good Solution

Prove that the G-S algorithm always outputs a a perfect matching M such that there are no instabilities with respect to M.

1. Recall and write down the definition of instability with respect to M.

   SOLUTION A pair (e, a') in $E \times A$ is an instability with respect to M if e:a and e':a' are distinct pairs in M, and also e prefers a' to a, and a' prefers e to e'. That is, e and a' would prefer to be matched with each other than with their matches in M.

2. Using your definition of instability, show that there are no instabilities with respect an output M of the G-S algorithm.

   One natural approach is to show that the (partial) matching constructed after each iteration of the While loop avoids instabilities. Let M_k be the matching at the end of iteration $k \geq 1$ of the While loop, and let E_k and A_k be the set of employers and the set of applicants, respectively, that are matched in M_k. Show by induction that there is no instability in $E_k \times A_k$ with respect to M_k.

   Claim: There is no instability in $E_k \times A_k$ with respect to M_k.

   SOLUTION

   Base case: $k = 1$. Then M_k has exactly one pair, and there can be no instabilities with respect to a set of size 1.

   Inductive step: We show that at the end of some iteration $k \geq 2$, there are no instabilities in $E_k \times A_k$ with respect to M_k, given the induction hypothesis that there are no instabilities in $E_{k-1} \times A_{k-1}$ with respect to M_{k-1}. Suppose that at iteration k, e considers (makes an offer to) a. There are three cases:

   Case 1. a is unmatched and accepts e's offer. No instability can involve a, because the other employers that are matched in M_k have not yet considered a and so prefer their own matches to a. No instability can involve e, since if e prefers a' to a, then a' has already been matched with an employer that a' prefers to e.

   There can be no instabilities that involve neither e nor a, since such instabilities would also have arisen in M_{k-1}, which is stable by the induction hypothesis.

   Case 2. a is already matched, say to e', and accepts e's offer, rejecting e'. In this case no instability can involve a, since a prefers e to its match in M_{k-1}. No instability can involve e for the same reason as in Case 1. Employer e' is now unmatched and so not in E_k, so e' does not concern us.

   There can be no instabilities that involve neither e nor a, for the same reason as in Case 1.

   Case 3. a is already matched, say to e', and prefers e' to e. In this case $M_k = M_{k-1}$ and since there are no instabilities with respect to M_{k-1}, there can be no instabilities with respect to M_k.

   This completes the inductive proof.

   Choosing k to be the last iteration of the While loop, we have that $M = M_k$, $E = E_k$ and $A = A_k$. Therefore there are no instabilities in $E \times A$ with respect to M.

   Note. Proof by induction is often a nice way to show that properties are preserved throughout successive iterations of a While loop. It's natural that the cases in the proof will follow the structure of If statements in the While loop. The proof in the textbook is more streamlined than the proof presented here, but inductive reasoning about While loops can be a more structured and fool-proof approach when gaining experience with proofs. You can always streamline the argument later.

6 True or False Again

You've now proved that the G-A algorithm is correct. But there are other interesting properties of the algorithm, and stable matchings more generally, that are valuable to understand. For example, choose whether you think the following statements are true or false.

1. Any valid instance of the Stable Matching Problem has exactly one good solution.

   SOLUTION False. Here's an instance with two good solutions.

```
e1: a1, a2          a1: e2, e1
e2: a2, a1          a2: e1, e2
```

The good solution { e1:a1, e2:a2 } is better for employers, while the good solution { e1:a2, e2:a1 } is best for applicants.

2. The output of the G-A algorithm is independent of the order in which employers are chosen in the While loop.

   SOLUTION The answer is not so clear here. On the one hand, the partial matchings M_k (obtained at successive iterations k of the While loop) depend considerably on the order in which employers are chosen, suggesting that different outputs are possible, and that the answer is "false". On the other hand, trying examples suggests that the answer is "true" since in the end, the algorithm converges on the same answer for the examples tried. Of course, we can't conclude that the statement is true based on a few examples, but it's not clear that the statement can be proved by induction on M_k.

   Here again it is helpful to try a different tack. If we're aiming to prove that the statement is true, can we be more specific about the properties of the unique output of the G-S algorithm, for a given input? Intuitively, the algorithm seems to be best for employers rather than applicants, because applicants are considered in the employers' preferred order. In what follows we show that the G-S algorithm outputs the "employer optimal" matching, and that this is unique. From this it will follow that the answer is "true".

7 Employer-Optimal Matchings

1. Write down more precisely what it means for an algorithm to be "employer optimal". Specifically, for a given employer e, define what is e's "best stable applicant". We'll denote this applicant by best(e).

   SOLUTION Fix an employer e. We say that applicant a is *stable for* e if there is a stable matching containing the pair e:a. We say that a is e's *best stable applicant* if among all of those applicants that are stable for e, a is ranked highest by e. We say that a matching is *employer optimal* if every employer is matched with its best stable applicant.

2. It turns out that no two employers have the same best stable applicant. Prove this.

   Claim: Each employer e has a distinct best(e).

   SOLUTION

   Proof: By contradiction. Suppose that for some applicant a, a = best(e) = best(e') for distinct employers e and e'. Suppose also that a prefers e to e' (the argument is similar, with e and e' swapped, if a prefers e' to e).

   Consider the stable matching M containing the pair e':best(e') = e':a. We'll use the fact that a is also best(e) to show that M has an instability, and this gives us our contradiction.

   In M, e prefers a to its match, since a = best(e). Also a prefers e to its match e' by our assumption. So, (e, a) is an instability with respect to M, contradicting the fact that M is stable. We conclude that best(e) $\neq$ best(e').

   Note: Proofs by contradiction will also be very handy in this course. Use definitions and known facts to help make progress, and break things down into simpler cases.

8 The G-S Algorithm Is Optimal for Employers

We can use induction also to show that the G-S algorithm outputs the (unique) employer-optimal stable matching in which every employer e is matched with best(e).

1. Prove the following claim.

Claim: At the end of every iteration k of the While loop, every employer e has only considered applicants that it ranks at least as high as best(e). Moreover, if e has considered best(e) on or before iteration k, then e is matched with best(e) at the end of iteration k.

SOLUTION You'll do this in Assignment 1.

2. Using the claim, show that the output of the G-S algorithm must be employer-optimal.

   SOLUTION When the algorithm terminates, every employer e must be matched with best(e), because (i) e is matched with **some** applicant in M, (ii) by the Claim, e can only be matched with an applicant it ranks at least as high as best(e), and and (iii) e cannot be matched with an applicant it ranks higher than best(e) since the ouput M is stable.

9 Useful Proof Strategies

Stepping away from Stable Matching now, discuss and summarize useful things that you've learned more generally about writing proofs, while doing this worksheet.

SOLUTION Hopefully you've learned some useful things. See the notes throughout the solutions above for some possibilities. One more note: Giving English descriptions to pseudocode in our algorithm makes it possible to write more intuitive statements in our proofs; these statements can be unambigously traced back to the pseudocode.