Algorithm Analysis

LECTURE 1
Introduction to Algorithm Design and Analysis

Joon Edward Sim

1. Introduction

In this short article, we seek to introduce students to algorithm design and analysis by working through a fairly non-trivial example - the Stable Matching problem. This writeup is based on material found both in the required textbook[**1**](Section 10.4) and the recommended text[**2**].

1.1. Understanding the problem

Consider the following scenarios of companies selecting from a pool of students for Summer Internship. Every company has a limited quota of interns and will seek to employ the best.

Scene A:

(1) John accepts an internship offer from BubiSoft.
(2) John receives an offer from EB, a company he prefers to BubiSoft.
(3) John withdraws his acceptance from BubiSoft.
(4) John accepts EB's offer.
(5) Bubisoft has to find another intern to replace John.

Scene B:

(1) Geeks Inc. interviewed Alan and offered him the last place of internship.
(2) Alan accepted the offer.
(3) However, Cindy applied for the internship after the application deadline.
(4) Geeks Inc. realized that Cindy is actually better than Alan for them.
(5) They withdraw offer for Alan(!) and offer Cindy instead. (...)

(6) Now Alan is free...

Question is ...does this cycle ever ends? Is it possible to assign students to companies such that the matching is so-called *self-enforcing*? i.e., whenever a student is assigned to a company the self-interest of both the student and the company prevents them from breaking the match.

Example 1. Student A is assigned to Company X. This pairing is "stable" if either of the following conditions are met:

- Consider all the companies that A prefers to X. All such companies prefer students that they are currently hiring than student A.
- Consider all the students that Company X prefers to the current hired batch. All these students prefer the company they are matched to to X.

1.2. Problem formulation

We note that the problem described above may actually takes many forms. For example, it applies to students forming groups for projects and for matching prospective single men and women for marriage. Here we consider a simple form of the problem – how to form stable marriages between a set of single men and single women[1].

Using set notation, we denote the set of males as $M = \{m_1,...,m_n\}$ and set of females as $F = \{f_1,...,f_n\}$. The set of all possible ordered pairs will then be the cartesian product of the two sets:

$$M \times F = \{(m,f) | m \in M \wedge f \in F\}$$

As such, we define a matching as the following:

Definition 1. A matching S is any subset of $M \times F$ where

$$(m,f) \in S \rightarrow (\nexists (m,f') \in S : f' \neq f) \wedge (\nexists (m',f) \in S : m' \neq m)$$

i.e., each member of M and each member of F appears in at most one pair in S.

[1] Please note that this is a special case of the problem described above: where each company only accepts 1 student. This special case is also known as the Stable Marriage problem.

Definition 2. A **perfect matching** S' is a matching where every male $m \in M$ is paired with some female $f \in F$.

Example 2. Say $M = \{m_1, m_2\}$ and $F = \{f_1, f_2\}$. $M \times F = \{(m_1, f_1), (m_1, f_2), (m_2, f_1), (m_2, f_2)\}$. An possible matching (but imperfect!) might be $\{(m_1, f_1)\}$ (m_2 is not paired). A possible matching might be $\{(m_1, f_2), (m_2, f_1)\}$.

Now, having defined perfect matchings, we need to formulate the notion of preferences. Each man $m \in M$ ranks all the women $f \in F$ in a particular order with *no ties* allowed. So each man has a *preference list* of all the women. Each woman keeps a similar list.

Furthermore, we need a notion of the stability of matching S.

Definition 3. A matching S is considered to be **instable** if there are pairs $(m_1, f_1) \in S$ and $(m_2, f_2) \in S$ while m_1 prefers f_2 to f_1 and f_2 prefers m_1 to m_2. A matching is stable if no such pairs exists.

Goal. Given a set of males M, a set of women F and their individual preference lists, find a perfect and stable matching.

2. Designing the algorithm

We have now come to the stage of solving the problem. Unfortunately, there are no set formulaic steps for solving problems. It is often a mixture of trial and error, experience, wit and good fortune. Most often, a good and elegant solution relies on having an insight into the deeper structures of the problem.

To give an intuition for the solution to be presented, consider an arbitrary pair (m,w). Is this a stable pair? From m's point of view, it is stable so long as all women he prefers to w rejects him i.e., m cannot run off to marry another woman he prefers. From w's point of view, it is stable only if every other man m' she prefers to m is paired with w' and m' prefers w' to w.

Therefore, the key insights that our solution depends on are:

- No pairing is considered *final* i.e., married, until everyone is paired. Temporary pairings are considered as engagements.
- Therefore, there are 2 states in which each man and woman can be – free or engaged.
- Man will take turns to propose to every women as long as he is free. Now, he proposes to every woman in the descending order of preference i.e., start from most preferred until the least preferred (if he is not engaged by then).

- A woman will accept a proposal from a man m in 2 cases: 1) She is free or 2) She is currently engaged to m' but she prefers m to m'. Otherwise she rejects m's proposal.

Gale and Shapley considered the same problem in 1962 in the context of college admissions or a job recruiting process. We now present algorithm for solving the Stable Marriage Problem (henceforth called the G-S algorithm) in the form of a pseudo-code:

```
Data: The set of man and women – $M$ and $F$ and $S$ the current set of
      engaged couples
Result: A perfect and stable matching
1  Initially, all $m \in M$ and $w \in F$ are free;
2  Consequently, $S$ is initially empty;
3  while there is a man m who is free and has not proposed to every woman do
4      Select such a man $m$;
5      Let $w$ be the most preferred woman that $m$ has not proposed to yet;
6      if $w$ is engaged to $m'$ then
7          if $w$ prefers $m$ to $m'$ then
8              $m$ and $w$ becomes engaged;
9              $m'$ becomes free;
10         end
11     else
12         $m$ and $w$ becomes engaged;
13     end
14 end
15 return $S$
```

3. Algorithm Analysis

Observation 1. Consider any woman w. Once engaged, w remains engaged and progressively gets engaged to a better and better man (in terms of her preference list) if possible.

This can be shown from how the G-S algorithm works. At the beginning, w is free. However, she progressively gets engaged to better man because she always breaks the current engagement to be engaged with someone whom she prefers.

Observation 2. The women a man proposes to becomes worse and worse (in terms of his preference list).

This can be seen from how the G-S algorithm works. A man is always proposing to the most-preferred woman he has not yet proposed to. Consequently, he can only propose to each woman at most once only and the sequence of women he proposed to is worse and worse.

Theorem 1. The while loop of the presented G-S algorithm terminates after at most n^2 iterations where n is the number of men.

Proof. There is only one proposal for each iteration. In the worst case, a man m may have to proposed to every woman i.e., n proposals. Given that there are n men, there are at most n^2 proposals. Therefore, the while loop terminates after at most n^2 iterations. □

Now, we want to prove that the above G-S algorithm returns a perfect matching that is stable. To do this, we need to prove certain intermediate results along the way.

Lemma 1. If m is free at some point during the execution of the G-S algorithm, there exists a woman to whom he has yet to propose.

Proof. We do this by contradiction. Suppose that at some point m is free and yet he has already proposed to every woman already. This implies that every woman is now engaged with someone else other than m. However, this is impossible (or we have reached a contradiction) since m is free and no two women can be engaged with the same person. □

Lemma 2. The set S returned at the termination of the while loop is a perfect matching.

Proof. Again, we prove this by contradiction. Suppose that the algorithm returns an imperfect matching i.e., there exists some man m and woman w who does not have a pair in S. However, the while loop only terminates under two possible conditions: 1) No free men or 2) There exist a free man but he has already proposed to every woman. The latter contradicts the lemma we have proven above. The former implies that every woman is engaged as well at the termination of the while loop. Hence, we have reached a contradiction and we have proven that the set S returned by the algorithm is a perfect matching. □

Theorem 2. Consider an execution of the G-S algorithm. The set S returned by an execution of the algorithm is a perfect and stable matching.

Proof. From the above lemmas, we have already proven that the returned S is a perfect matching. By way of contradiction, let us assume that S is an unstable, perfect matching i.e., there exists at least two tuples $(m,f) \in S$ and $(m',f') \in S$ s.t. the following is true:

- m prefers f' to f
- f' prefers m to m'

  m is finally married to f but m prefers f'. Based on our observation made above, m must have proposed to f' before f because m prefers f' to f. When m proposed to f', did she accept the proposal? We will show that either way, we will reach a contradiction.
- If yes, f' was engaged to m for a while before finally engaged to m'. This contradicts our observation above that a woman, once engaged, becomes engaged to better and better man.
- If no, f' must be engaged to some m'' whom she prefers to m. Somehow later, she got engaged with m', which is a contradiction for the same reason as above.

Either way, we have reached a contradiction. Therefore, we have shown that S is a perfect and stable matching. □

4. Extended Analysis

Consider the following scenario where $M = \{m,m'\}$ and $F = \{w,w'\}$ and the following preferences:

- m prefers w to w'.
- m' prefers w' to w.
- w prefers m' to m.
- w' prefers m to m'.

What are the possible stable matchings? There are two:

- $\{(m',w),(m,w')\}$ OR
- $\{(m,w),(m',w')\}$

This example shows us two things:

- There could be multiple possible stable matchings. Then the question is: which one is returned by our proposed algorithm? The question is compounded by a vagueness in the proposed algorithm – line 3 of the algorithm did not specify at all how the free man m is to be chosen. Does the order of selecting the free man affect the result at all?
- The issue of **fairness**. Notice that in one of the matching above, the women were more satisfied while in the other it's the men.

So there are two issues at stake here: 1)Does the proposed algorithm always return the same stable matching? and 2)How does the algorithm perform with respect to the issue of fairness? The answer to the first question is yes and we shall attempt to answer both questions in one single proof. Before that, we need to formulate the idea of fairness.

Definition 4. A woman w is considered to be a **valid partner** of a man m if there exists a perfect stable matching S where $(m,w) \in S$.

Definition 5. The **best valid partner** w of a man m is a valid partner and there are no women m prefers to w that is a valid partner. We denote the best valid partner of a man m as $best(m)$.

We shall now prove that 1) every execution of algorithm *always* returns the same set and 2) the set returned is the set $S^* = \{(m,best(m)) : m \in M\}$.

Theorem 3. Every execution of the G-S algorithm results in the set S^*.

Proof. We prove by contradiction. Assume that there exist some execution ε that returns a matching S that is not S^*. Now this implies that during the execution ε, there must be an iteration where, for the first time, a man m is rejected by his $best(m) = w$. This is also the first time during the execution that the *any* man is rejected by *any* valid partner[2].

Now, m might be rejected because w is currently engaged by a better man in her opinion or w ditched him for a better man. Let this better man be m' and she must prefer m' to m.

[2] There could, of course, be more than one man who in the end is not paired with his best valid partner. For the purposes of the proof, we only consider the first rejection.

But w is a valid partner of m. There must exist a stable matching S' and $w' \in F$ such that $(m,w) \in S'$ and $(m',w') \in S'$.

m is the first man to be rejected by a valid partner. This means that m' has not yet been rejected by any valid partner. Since w' is a valid partner of m', this means that m' must prefer w to w'. However, this leads to a contradiction because S' is not a stable matching then. □

Consequently, the G-S algorithm is known to be man-optimal or proposal-optimal, depending who is the one doing the proposal in the algorithm. How does the algorithm fare for the women then?

Definition 6. A man m is considered to be a **valid partner** of a man w if there exists a perfect stable matching S where $(m,w) \in S$.
Definition 7. The **worst valid partner** m of a woman w is a valid partner and no man ranked lower than m is a valid partner. We denote the worst valid partner by $worst(w)$.

Theorem 4. In stable matching S^*, $S^* = \{(worst(w),w) : w \in F\}$.

Proof. Let $m = worst(w)$. Suppose that $(m',w) \in S^*$ such that $m' \neq m$ i.e., w prefers m' to m. Since m is a valid partner of w, there must be a stable matching S' such that $(m,w) \in S'$ and $(m',w') \in S'$.

However, we have proven above that every man is paired with his best valid partner in S^*, so it follows that m' prefers w to w'. But w prefers m' to m, thus causing an instability in S^*. Hence we have reached a contradiction. □

References

[1] Algorithm Design, Jon Kleinberg and Eva Tardos (ISBN: 9780321372918)
[2] Introduction to the Design and Analysis of Algorithms, Anany Levitin (ISBN: 9780132316811)