Stable Matching Problem; Five Representative Problems: PS-1

Suchintan Mishra

Department of Computer Science and Engineering
Institute of Technical Education and Research,
Siksha 'O' Anusandhan (Deemed to be University), Bhubaneswar, Odisha, India

Text Books

Text book and References:

- (T1) Algorithm Design by Jon Kleinberg and Eva Tardos, Pearson Publication Reference book
- (R1) The Algorithm Design Manual by Steven Skiena, Springer Publication
- (R2) Introduction to Algorithms by CLRS, PHI Publication

Outline

Problems on Running Time

What is the worst-case running time of bubblesort? How does it compare to the running time of insertion sort?

Solution

The ith iteration of the for loop of lines 1-4 will cause n—i iterations of the for loop of lines 2-4, each with constant time execution, so the worst-case running time of bubble sort is $\Theta(n^2)$ which is same as the worst-case running time of insertion sort.

Although insertion also runs at $\Theta(n^2)$ worst-case time, the **number of assignments (swaps) performed in bubblesort is way more than that of insertion sort**. So, the constant factors in the running time will be much larger for bubblesort compared to that of insertion sort. **This means, for the same input size, insertion sort will run faster than bubblesort.**

A Hypothetical Situation

To set up the question, let's first think informally about the kind of situation that might arise as a group of friends, **all juniors in college majoring in computer science, begin applying to companies for summer internships.** The crux of the application process is the **interplay between two different types of parties: companies (the employers) and students (the applicants).**
Each applicant has a preference ordering on companies, and each company—once the applications come in—forms a preference ordering on its applicants. Based on these preferences, **companies extend offers to some of their applicants, applicants choose which of their offers to accept**, and people begin heading off to their summer internships.

Summary of the Problem Statement

Given a set of preferences among employers and applicants, can we assign applicants to employers so that for every employer E, and every applicant A who is not scheduled to work for E, at least one of the following two things is the case?

- E prefers every one of its accepted applicants to A; or
- A prefers her current situation over working for employer E.

If this holds, the outcome is stable: individual self-interest will prevent any applicant/employer deal from being made behind the scenes.

Stable Matching Problem

"we observe that this special case can be viewed **as the problem of devising a system by which each of n men and n women can end up getting married: our problem naturally has the analogue of two "genders"—the applicants and the companies—and in the case we are considering, everyone is seeking to be paired with exactly one individual of the opposite gender"**

Formal Definition of the Problem

So consider a set $M = m_1, ..., m_n$ of n men, and a set $W = w_1, ..., w_n$ of n women. Let MxW denote the set of all possible ordered pairs of the form (m, w), where $m \in M$ and $w \in W$. A matching S is a set of ordered pairs, each from MW, with the property that each member of M and each member of W appears in at most one pair in $S^{'}$. A perfect matching $S^{'}$ is a matching with the property property that each member of M and each member of W appears in exactly one pair in $S^{'}$.

The Notion of Preferences

Now we can add the notion of preferences to this setting. Each man $m \in M$ ranks all the women; we will say that m prefers w to $w^{'}$ if m ranks w higher than $w^{'}$.We will refer to the ordered ranking of m as his preference list.We will not allow ties in the ranking. Each woman, analogously, ranks all the men.

Man	Preference
Adam	Cindy
	Allie
	Betty
Blake	Betty
	Allie
	Cindy
Carl	Cindy
	Betty
	Allie

Woman	Preference
Allie	Adam
	Blake
	Carl
Betty	Carl
	Blake
	Adam
Cindy	Adam
	Carl
	Blake

Given a perfect matching S, what can go wrong?

- There are two pairs (m, w) and ($m^{'}$,$w^{'}$) in S with the property that m prefers $w^{'}$ to w, and $w^{'}$ prefers m to $m^{'}$. In this case, nothing to stop m and w from abandoning their current partners and heading off together; the set of marriages is not self enforcing. We'll say that such a pair (m, $w^{'}$) is an instability with respect to S:(m, $w^{'}$) does not belong to S, but each of m and $w^{'}$ prefers the other to their partner in S.

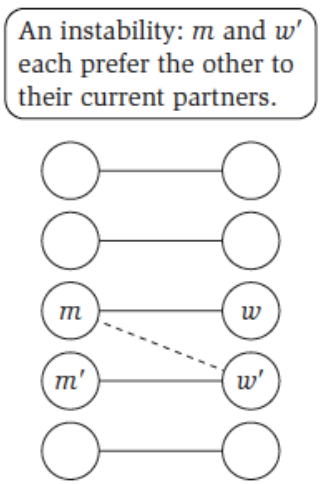


Figure 1.1 Perfect matching S with instability (m, w').

Why Stable

Our goal, then, is a set of marriages with no instabilities. We'll say that a matching S is stable if (i) it is perfect, and (ii) there is no instability with respect to S. Two questions spring immediately to mind: Does there exist a stable matching for every set of preference lists?. Given a set of preference

Design of the Algorithm I

- Initially, everyone is unmarried. Suppose an unmarried man m chooses the woman w who ranks highest on his preference list and proposes to her. Can we declare immediately that (m,w) will be one of the pairs in our final stable matching? Not necessarily: at some point in the future, a man m' whom w prefers may propose to her. dangerous for w to reject m right away; she may never receive a proposal from someone she ranks as highly as m. So a natural idea would be to have the pair (m, w) enter an intermediate state—engagement.
- Suppose we are now at a state in which some men and women are free— not engaged—and some are engaged. The next step could look like this. An arbitrary free man m chooses the highest-ranked woman w to whom he has not yet proposed, and he proposes to her. If w is also free, then m and w become engaged. Otherwise, w is already engaged to some

Design of the Algorithm II

other man m'. In this case, she determines which of m or m' ranks higher on her preference list; this man becomes engaged to w and the other becomes free.

- Finally, the algorithm will terminate when no one is free; at this moment, all engagements are declared final, and the resulting perfect matching is returned.

Design of the Algorithm III

```
Initially all m ∈ M and w ∈ W are free
While there is a man m who is free and hasn't proposed to
every woman
   Choose such a man m
   Let w be the highest-ranked woman in m's preference list
      to whom m has not yet proposed
   If w is free then
      (m, w) become engaged
   Else w is currently engaged to m'
      If w prefers m' to m then
         m remains free
      Else w prefers m to m'
         (m, w) become engaged
         m' becomes free
      Endif
   Endif
Endwhile
Return the set S of engaged pairs
```

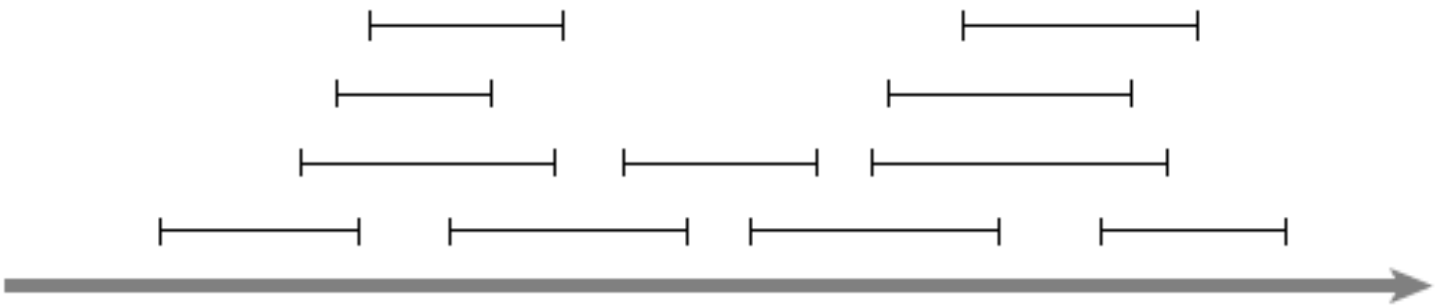
Analysis

- w remains engaged from the point at which she receives her first proposal; and the sequence of partners to which she is engaged gets better and better (in terms of her preference list).
- The sequence of women to whom m proposes gets worse and worse (in terms of his preference list).
- The G-S algorithm terminates after at most n^2 iterations of the While loop.

Graph

Thus, G consists of a pair of sets (V, E)—a collection V of nodes and a collection E of edges, each of which "joins" two of the nodes. We thus represent an edge e E as a two-element subset of V: e = u, v for some u, v V, where we call u and v the ends of e. We typically draw graphs as in Figure 1.3, with each node as a small circle and each edge as a line segment joining its two ends.

Interval Scheduling

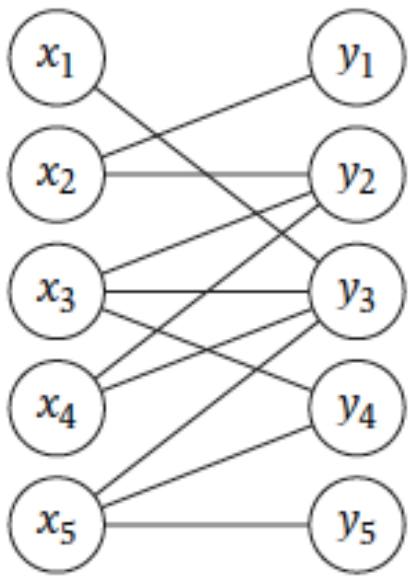


Weighted Interval Scheduling

Now, suppose more generally that each request interval i has an associated value, or weight, $v_i > 0$; we could picture this as the amount of money we will make from the ith individual if we schedule his or her request. Our goal will be to find a compatible subset of intervals of maximum total value.

Bipartite Matching

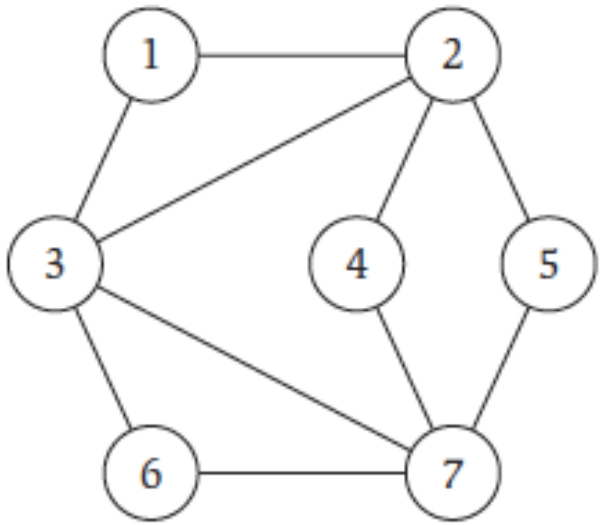
We say that a graph G = (V, E) is bipartite if its node set V can be partitioned into sets X and Y in such a way that every edge has



one end in X and the other end in Y.

Independent Set

Given a graph G = (V, E), we say a set of nodes S V is independent if no two nodes in S are joined by an edge.



Problem Solving

- Decide whether you think the following statement is true or false. If it is true, give a short explanation. If it is false, give a counterexample. True or false? **In every instance of the Stable Matching Problem, there is a stable matching containing a pair (m, w) such that m is ranked first on the preference list of w and w is ranked first on the preference list of m.**
- Decide whether you think the following statement is true or false. If it is true, give a short explanation. If it is false, give a counterexample.

Solutions

True or false? In every instance of the Stable Matching Problem, there is a stable matching containing a pair (m, w) such that m is ranked fi
list of m.

False. Consider the group m1, m2, w1, and w2.

m1 has the ranking:
1. w1
2. w2

m2 has the ranking:
1. w2
2. w1

w1 has the ranking:
1. m2
2. m1

w2 has the ranking:
1. m1
2. m2

In this situation, it's impossible to have a pair (m, w), let alone a stable matching containing a pair, such that m is ranked first on the prefere

Thank You