CS340 Analysis of Algorithms

Handout:	**-1**	**Professor:**	**Dianna Xu**
Title:	**Stable Matching Decision**	**E-mail:**	dxu@brynmawr.edu
Date:		**URL:**	**http://cs.brynmawr.edu/cs340**

Sample description, pseudo code and time analysis of the Stable Matching Decision problem. Recall that given a set of employers E, a set of applicants A, $|E|=|A|=n$, $2n$ preference lists, each of size n, and a set of perfect matchings M, which is a set of n pairs (e,a), where $e \in E$ and $a \in A$, we ask for a yes/no on whether M is stable. Please refer to lecture notes and your text book for definitions of matching, perfect matching and stable matching.

Throughout this course, we assume the standard implementation technique that all data with known size can (and should/will) be assigned unique integer IDs that can be used to index into appropriately-sized arrays for $O(1)$ access. You can assume that these integer IDs will be stored with the objects representing the associated data elements (in this problem, employers and applicants) and therefore each data element has a mapping to an integer. Pseudocode will use data elements to index arrays without considering the details of index assignment.

A related remark is dictionaries/hashmaps are less efficient than directly indexing into arrays, especially in terms of space. Dictionaries/hashmaps require load factors to be below 70%-80% in order to resolve collisions, which means 20%-30% wasted space. If the size of the data is known, arrays should ALWAYS be the choice for $O(1)$ operations. For example, hashmaps are an overkill for this problem. Arrays have no collisions and lookup in arrays is $\theta(1)$ instead of $O(1)$.

1 Description

For each employer and applicant pair in M, construct a list of more preferred matches compared to the one they are currently assigned to. One such "better matches" list is generated for the employer and one for the applicant per matched pair, for a total of $2n$ lists. Examine the "better matches" lists. If any employer e has a better match a', where a''s "better matches" list also contains e, then we have found a possible switch and the matching is unstable.

2 Pseudocode

```
1  Function StableMatchingDecision(M)
2      for each (e_i, a_j) in M do
3          construct two lists L_{e_i} and L_{a_j}, where
4          L_{e_i} = all applicants in e_i's preference list who rank higher than a_j
5          L_{a_j} = all employers in a_j's preference list who rank higher than e_i
6      end
7      for each e_i in E do
8          for each a_k in L_{e_i} do
9              if L_{a_k} contains e_i then
10                 output no
11                 exit
12             end
13         end
14     end
15     output yes
```

3 Time Analysis

We store all preference lists as sorted arrays (sorted linked lists work too) of size n, in order of preference. Better matches lists are stored as linked lists.

There are two main loops in this algorithm. The first `for` loop runs n times because there are n pairs in M. In the body of the loop, construction time of each of the "better matches" list (lines 4 or 5) is $O(n)$ because it requires a linear scan (upto the assigned match) of a single preference list, which is of size n. Therefore the first `for` loop runs in $n \times (n+n) = 2n^2 = O(n^2)$

The second `for` loop runs a total of $O(n^2)$ times because $|E| = n$ and any "better matches" list is of size $n-1$ in the worst case. In the body of the loop, line 9 requires another linear search in a list of size n and therefore the second `for` loop runs in $n^2 \times n = O(n^3)$

Overall, the algorithm runs in $O(n^3)$. The size of the data structures used is $O(n^2)$, for a total of $2n$ "better matches" lists, each of size n.