CS340 - Analysis of Algorithms

Stable Matching Algorithm

Logistics:

Prof Office Hours: Friday 3-5pm

TA Office Hours: TBD

Join the piazza and gradescope if you have not already

Make sure you have a goldengate account

- Email David Diaz ddiaz1@brynmawr.edu if you encounter issues

HW1 due next Monday (9/15)

Lab0 due Thursday (9/11)

Agenda

1. Lab 0 review

2. Gale-Shapley Algorithm

Lab0 Review

1. Formalize an algorithm for deciding if a matching is stable
    a. Given a set of employers E, a set of applicants A, and a matching M, decide if the matching is stable
    b. Writeup in latex
2. Give a proof for termination and correctness
3. Discuss time analysis
    a. Worst case

Stability Definition

For every employer E and every applicant A who is not assigned to intern at E,

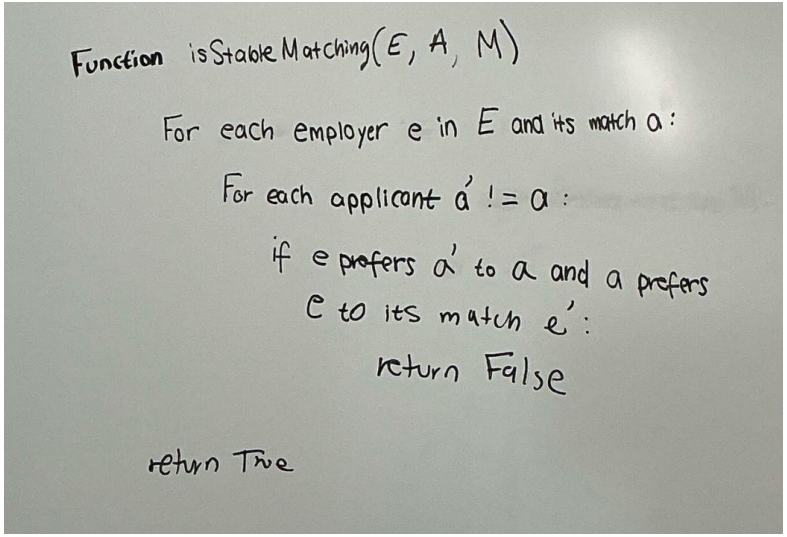
at least one of the following should be true:

- E prefers its accepted intern to A or
- A prefers her current internship over working for employer E

Individual self interest will lead to stability

Lab 0

Employers			Applicants		
Google(G)	Intel(I)	Apple(A)	Kate(K)	Clara(C)	Lisa(L)
K	K	L	A	A	G
C	L	K	G	I	I
L	C	C	I	G	A



M: (G, C), (I, L), (A, K) Output: True

M: (G, L), (I, K), (A, C) Output: False

Lab 0 - An alternate Algorithm

Function isStableMatching(E, A, M):

for each (e_i, a_j) in M:

construct two lists L_{e_i} and L_{a_j}, where

L_{e_i} = all applicants in e_i's preference list who rank higher than a_j

L_{a_j} = all employers in a_j's preference list who rank higher than e_i

for each e_i:

for each a_k in L_{e_i}:

if L_{a_k} contains e_i:

output no

break

output yes

Employers			Applicants		
Google(G)	Intel(I)	Apple(A)	Kate(K)	Clara(C)	Lisa(L)
K	K	L	A	A	G
C	L	K	G	I	I
L	C	C	I	G	A

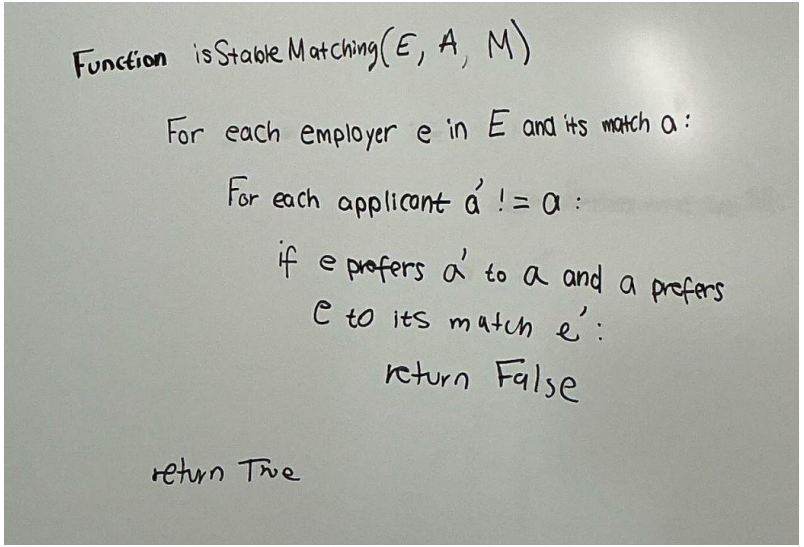
M: (G, C), (I, L), (A, K) Output: True

M: (G, L), (I, K), (A, C) Output: False

Data Structures:

Algorithm 1: For each employer, the algorithm loops over each applicant. Let n be |E| and |A|. Each for loop runs n times for a total of n^2 iterations. For the algo to be $O(n^2)$, the following operations must be done in O(1) time:

1. Retrieving an employer's match a'
2. Retrieving an applicants match e'
3. Checking if e prefers a' to a
4. Checking if a prefers e to e'



Data Structures:

Algorithm 2: For each employer, the algorithm loops over each applicant. Let n be |E| and |A|. Each for loop runs n times for a total of n² iterations. For the algo to be O(n²), the following operation must be done in O(1) time:

1. Checking if e is contained in L_a

The first for-loop for constructing the L_e and L_a lists executes n times. Creating both hashsets will take O(n) time, so the loop will be O(n²).

Function isStableMatching(E, A, M):

for each (e_i, a_j) in M:

construct two lists L_{e_i} and L_{a_j}, where

L_{e_i} = all applicants in e_i's preference list who rank higher than a_j

L_{a_j} = all employers in a_j's preference list who rank higher than e_i

for each e_i:

for each a_k in L_{e_i}:

if L_{a_k} contains e_i:

output no

break

output yes

Data Structures:

Which algorithm is better? What is the tradeoff?

Algorithm 1: Proof of Termination and Correctness

Termination: The number of employees and applicants is finite, so the algorithm terminates

Correctness: Suppose the function returns True, but there is actually an instability. By the definition of instability, there must exist some e and some a’ (e, a’) such that e prefers a’ to its match a and a’ prefers e to its match e’. But if this was the case, the if statement would evaluate to True and the function would return False.

Algorithm 2: Proof of Termination and Correctness

Termination: The number of employees and applicants is finite, so the algorithm terminates

Correctness: Suppose the function returns Yes, but there is actually an instability. By the definition of instability, there must exist some e_i and some a_k such that e_i prefers a_k to its match a_j and a_k prefers e_i to its match. But if this a_k preferred e_i to its match, L_ak would contain e_i and the function would output no

Lab 0

- Review the algorithm writeup guidelines
- Review the sample report
- Writeup in Latex
- Submit by 9/11 before next lab

An algorithm to produce a stable matching

Input:

- a set of employers: E
- a set of applicants: A
- $|E| = |A| = n$
- $2n$ preference lists, each of size n

Output: M, a matching of n pairs

Small example

Employers			Applicants		
Google(G)	Intel(I)	Apple(A)	Kate(K)	Clara(C)	Lisa(L)
K	K	L	A	A	G
C	L	K	G	I	I
L	C	C	I	G	A

Gale-Shapely: an algorithm to produce a stable matching

```
Function findStableMatching(E, A)
    All e ∈ E and a ∈ A are unpaired
    while there is an e unpaired that hasn't made an offer to every a do
        choose such an e
        let a be the highest-ranked applicant in e's preference list who e has not
        made an offer to yet
        if a is unpaired then
            pair a and e
        end
        else
            a is currently paired with e'
            if a prefers e' to e then
                e remains unpaired
            end
            else
                a is paired with e and e' becomes unpaired
            end
        end
    end
    return the set of pairs
```

Employers			Applicants		
Google(G)	Intel(I)	Apple(A)	Kate(K)	Clara(C)	Lisa(L)
K	K	L	A	A	G
C	L	K	G	I	I
L	C	C	I	G	A

Gale-Shapely: Observations

Observation 1: each applicant takes their first offer and remains employed

Observation 2: the sequence of employers each applicant accepts gets better and better (in terms of her preference list)

Observation 3: the sequence of applicants the employers offer jobs to gets worse and worse (in terms of their preference list)

Runtime Analysis

1. How many times does the while-loop execute?

Define a measure of *progress*

Each step the algorithm takes brings it closer to termination

```
Function findStableMatching(E, A)
    All e ∈ E and a ∈ A are unpaired
    while there is an e unpaired that hasn't made an offer to every a do
        choose such an e
        let a be the highest-ranked applicant in e's preference list who e has not
        made an offer to yet
        if a is unpaired then
            pair a and e
        end
        else
            a is currently paired with e'
            if a prefers e' to e then
                e remains unpaired
            end
            else
                a is paired with e and e' becomes unpaired
            end
        end
    end
    return the set of pairs
```

Runtime Analysis

In order for the algorithm to run in $O(n^2)$ time, the following operations need to be done in O(1):

1. Selecting / checking existence of an unpaired e that hasn't made an offer to every a
2. Choosing a who is highest-ranked applicant in e's preference list who e has not made an offer to yet
3. Checking if a is paired or unpaired
4. Checking if a prefers e' to e

```
Function findStableMatching(E, A)
    All e ∈ E and a ∈ A are unpaired
    while there is an e unpaired that hasn't made an offer to every a do
        choose such an e
        let a be the highest-ranked applicant in e's preference list who e has not
        made an offer to yet
        if a is unpaired then
            pair a and e
        end
        else
            a is currently paired with e'
            if a prefers e' to e then
                e remains unpaired
            end
            else
                a is paired with e and e' becomes unpaired
            end
        end
    end
    return the set of pairs
```

Termination

The algorithm terminates when there are no unpaired employers or each employer has made an offer to every applicant.

Since employers never ask the same applicant twice and there are at most n applicants, the algorithm terminates in at most n^2 iterations of the while-loop.

```
Function findStableMatching(E, A)
    All e ∈ E and a ∈ A are unpaired
    while there is an e unpaired that hasn't made an offer to every a do
        choose such an e
        let a be the highest-ranked applicant in e's preference list who e has not
        made an offer to yet
        if a is unpaired then
            pair a and e
        end
        else
            a is currently paired with e'
            if a prefers e' to e then
                e remains unpaired
            end
            else
                a is paired with e and e' becomes unpaired
            end
        end
    end
    return the set of pairs
```

Proof of Correctness:

For this algorithm to be correct, it must produce a perfect and stable matching.

A matching is perfect if every element of X and Y occurs in some pair.

Proof by contradiction: suppose there is some employer e who was not matched.

```
Function findStableMatching(E, A)
    All e ∈ E and a ∈ A are unpaired
    while there is an e unpaired that hasn't made an offer to every a do
        choose such an e
        let a be the highest-ranked applicant in e's preference list who e has not
        made an offer to yet
        if a is unpaired then
            pair a and e
        end
        else
            a is currently paired with e'
            if a prefers e' to e then
                e remains unpaired
            end
            else
                a is paired with e and e' becomes unpaired
            end
        end
    end
    return the set of pairs
```

Proof of Correctness:

Proof of stability by contradiction: Suppose there exists some e and some a' such that e prefers a' to its match a and a' prefers e to its match e.

Either: (1) e never offered a job to a' or (2) a' rejected the offer

```
Function findStableMatching(E, A)
    All e ∈ E and a ∈ A are unpaired
    while there is an e unpaired that hasn't made an offer to every a do
        choose such an e
        let a be the highest-ranked applicant in e's preference list who e has not
        made an offer to yet
        if a is unpaired then
            pair a and e
        end
        else
            a is currently paired with e'
            if a prefers e' to e then
                e remains unpaired
            end
            else
                a is paired with e and e' becomes unpaired
            end
        end
    end
    return the set of pairs
```

Summary

1. Logistics
    a. Gradescope
    b. Piazza
    c. Goldengate account

2. Continue reading textbook

3. Lab 0 due Sep 11th

4. HW1 due Sep 15th (next Monday)

5. Next class: discrete math and graphs review