CSEP 521: Applied Algorithms
Lecture 5 Average Case Analysis

Richard Anderson

January 19, 2021

Announcements

- Office hours
  - Oscar: 5-6 pm, Monday and Wednesday
  - Richard: 11am-noon, Monday, 2-3 pm Friday
- Homework 3 is available

- Today, Stable Matching (Stable Marriage)
  - Recommended reading: Kleinberg-Tardos, Chapter 1
- Thursday, Random algorithm for primality testing

Average Case Performance of Algorithms

- Main topics for today
  - Average case of stable marriage algorithm
  - Coupon Collector Problem

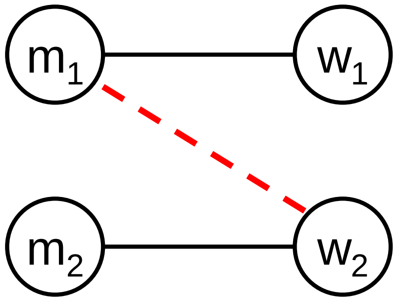
- Formal setting, input is drawn randomly from a probability distribution on legal inputs
- Standard runtime model
  - T(N) = max {over inputs I of size N} $T_A(I)$
- Average case runtime
  - T(N) = average {over inputs I of size N using probability distribution P} $T_A(I)$

Stable Matching

- Setting:
  - Assign TAs to Instructors
  - Avoid having TAs and Instructors wanting changes
    - E.g., Prof A. would rather have student X than her current TA, and student X would rather work for Prof A. than his current instructor.

Formal notions

- Perfect matching
- Ranked preference lists
- Stability



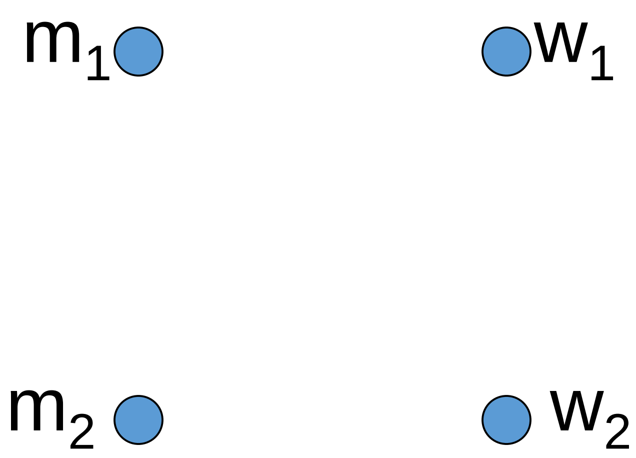
Example (1 of 3)

m_1: w_1 w_2

m_2: w_2 w_1

w_1: m_1 m_2

w_2: m_2 m_1



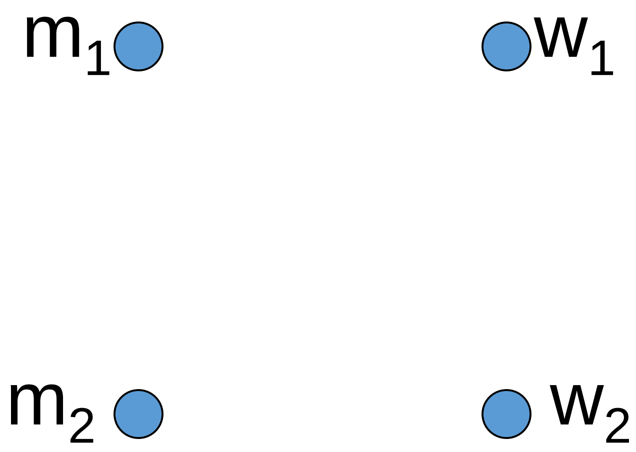
Example (2 of 3)

m_1: w_1 w_2

m_2: w_1 w_2

w_1: m_1 m_2

w_2: m_1 m_2



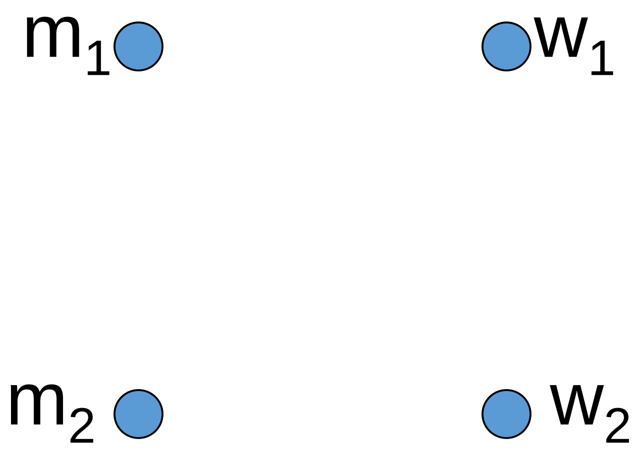
Example (3 of 3)

m_1: w_1 w_2

m_2: w_2 w_1

w_1: m_2 m_1

w_2: m_1 m_2



Formal Problem

- Input
  - Preference lists for $m_1, m_2, \ldots, m_n$
  - Preference lists for $w_1, w_2, \ldots, w_n$
- Output
  - Perfect matching M satisfying stability property:

If $(m', w') \in M$ and $(m'', w'') \in M$ then
(m' prefers w' to w'') or (w'' prefers m'' to m')
[In other words, m' and w'' do not want to pair up.]

Idea for an Algorithm

m proposes to w

- If w is unmatched, w accepts
- If w is matched to m_2
  - If w prefers m to m_2 w accepts m, dumping m_2
  - If w prefers m_2 to m, w rejects m

Unmatched m proposes to the highest w on its preference list that it has not already proposed to

Algorithm

Initially all m in M and w in W are free
While there is a free m
    w highest on m's list that m has not proposed to
    if w is free, then match (m, w)
    else
        suppose (m_2, w) is matched
        if w prefers m to m_2
            unmatch (m_2, w)
            match (m, w)

Example

m_1: w_1 w_2 w_3

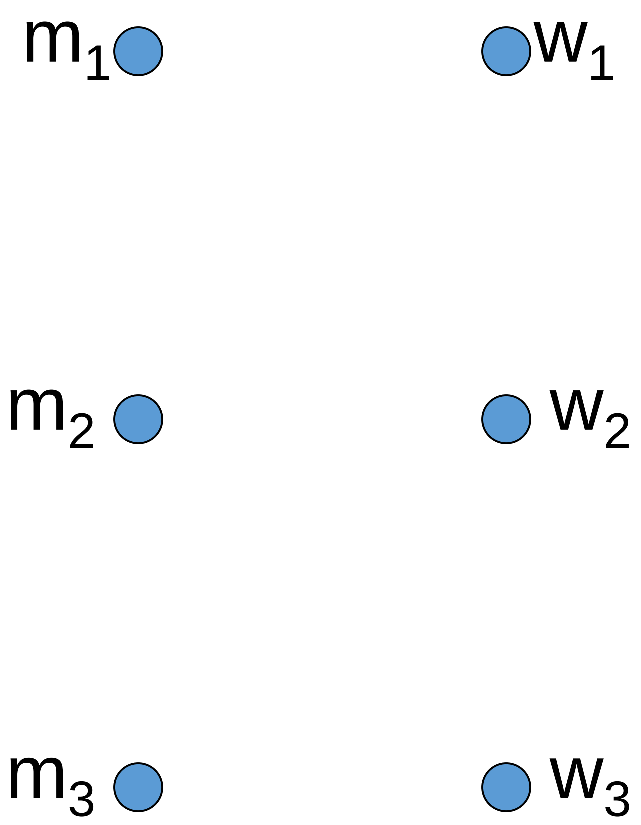
m_2: w_1 w_3 w_2

m_3: w_1 w_2 w_3

w_1: m_2 m_3 m_1

w_2: m_3 m_1 m_2

w_3: m_3 m_1 m_2



Cleaned up example

m_1: w_1 w_2 w_3

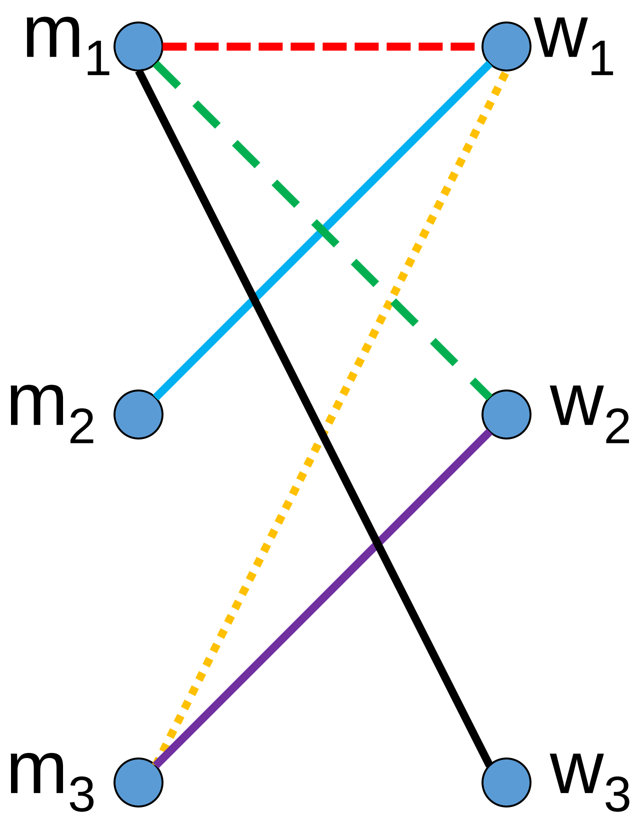
m_2: w_1 w_3 w_2

m_3: w_1 w_2 w_3

w_1: m_2 m_3 m_1

w_2: m_3 m_1 m_2

w_3: m_3 m_1 m_2



Order: m_1, m_2, m_3, m_1, m_3, m_1

Does this work?

- Does it terminate?
- Is the result a stable matching?

- Begin by identifying invariants and measures of progress
  - m’s proposals get worse (have higher m-rank)
  - Once w is matched, w stays matched
  - w’s partners get better (have lower w-rank)

Claim: If an m reaches the end of its list, then all the w's are matched

Claim: The algorithm stops in at most n^2 steps

When the algorithms halts, every w is matched

Why?

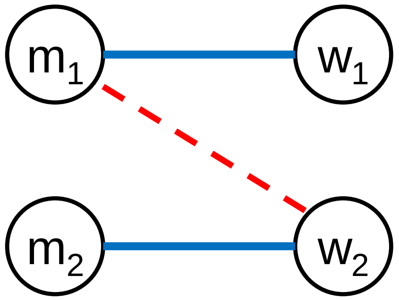
Hence, the algorithm finds a perfect matching

The resulting matching is stable

Suppose

$(m_1, w_1) \in M, (m_2, w_2) \in M$

m_1 prefers w_2 to w_1



How could this happen?

Result

- Simple, $O(n^2)$ algorithm to compute a stable matching
- Corollary
  - A stable matching always exists

Algorithm under specified

- Many different ways of picking m’s to propose
- Surprising result
  - All orderings of picking free m’s give the same result

- Proving this type of result
  - Reordering argument
  - Prove algorithm is computing something mores specific
    - Show property of the solution – so it computes a specific stable matching

M-rank and W-rank of matching

- m-rank: position of matching w in preference list
- M-rank: sum of m-ranks
- w-rank: position of matching m in preference list
- W-rank: sum of w-ranks

m_1: w_1 w_2 w_3

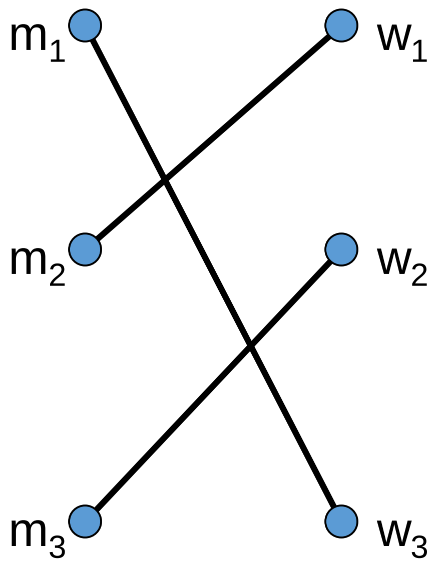
m_2: w_1 w_3 w_2

m_3: w_1 w_2 w_3

w_1: m_2 m_3 m_1

w_2: m_3 m_1 m_2

w_3: m_3 m_1 m_2



What is the M-rank?

What is the W-rank?

Breakout groups
Suppose there are n m's, and n w's

- What is the minimum possible M-rank?

- What is the maximum possible M-rank?

- Suppose each m is matched with a random w, what is the expected M-rank?

Random Preferences

Suppose that the preferences are completely random

m_1: w_8 w_3 w_1 w_5 w_9 w_2 w_4 w_6 w_7 w_{10}
m_2: w_7 w_{10} w_1 w_9 w_3 w_4 w_8 w_2 w_5 w_6
…
w_1: m_1 m_4 m_9 m_5 m_{10} m_3 m_2 m_6 m_8 m_7
w_2: m_5 m_8 m_1 m_3 m_2 m_7 m_9 m_{10} m_4 m_6
…

If there are n m's and n w's, what is the expected value of the M-rank and the W-rank when the proposal algorithm computes a stable matching?

Stable Matching Results

- Averages of 5 runs
- Much better for M than W
- Why is it better for M?
- What is the growth of m-rank and w-rank as a function of n?

n	m-rank	w-rank
500	5.10	98.05
500	7.52	66.95
500	8.57	58.18
500	6.32	75.87
500	5.25	90.73
500	6.55	77.95
1000	6.80	146.93
1000	6.50	154.71
1000	7.14	133.53
1000	7.44	128.96
1000	7.36	137.85
1000	7.04	140.40
2000	7.83	257.79
2000	7.50	263.78
2000	11.42	175.17
2000	7.16	274.76
2000	7.54	261.60
2000	8.29	246.62

Coupon Collector Problem

- n types of coupons
- Each round you receive a random coupon
- How many rounds until you have received all types of coupons
- p_i is the probability of getting a new coupon after i-1 have been collected
- t_i is the time to receive the i-th type of coupon after i-1 have been received

$$p_i = \frac{n-(i-1)}{n} = \frac{n-i+1}{n}$$

t_i has geometric distribution with expectation

$$\frac{1}{p_i} = \frac{n}{n-i+1}$$

$$\begin{aligned}
\mathrm{E}(T) &= \mathrm{E}(t_1 + t_2 + \cdots + t_n) \\
&= \mathrm{E}(t_1) + \mathrm{E}(t_2) + \cdots + \mathrm{E}(t_n) \\
&= \frac{1}{p_1} + \frac{1}{p_2} + \cdots + \frac{1}{p_n} \\
&= \frac{n}{n} + \frac{n}{n-1} + \cdots + \frac{n}{1} \\
&= n \cdot \left(\frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{n}\right) \\
&= n \cdot H_n.
\end{aligned}$$

$$\mathrm{E}(T) = n \cdot H_n = n \log n + \gamma n + \frac{1}{2} + O(1/n).$$

Stable Matching and Coupon Collecting

- Assume random preference lists
- Runtime of algorithm determined by number of proposals until all w's are matched
- Each proposal can be viewed[1] as asking a random w
- Number of proposals corresponds to number of steps in coupon collector problem

[1]There are some technicalities here that are being ignored

A more careful analysis

- Principle of deferred randomness
  - Generate random list, traverse list
  - Traverse list, generating random elements
- Suppose that i - 1 M's are matched, expected number of proposals until i matches
  - What is the chance that X proposes to an unmatched W?
  - If X has already proposed j times, the chance is $(n - (i - j - 1))/(n-j) > (n-(i-1))/n = p_i$
  - The conditioning gives a greater probability of success, reducing the expected time to success

What about the W rank?

Balls and boxes

- N boxes, repeatedly assign balls to random boxes
- Coupon collecting – expected number of balls until every box is occupied
- How about if we assign K balls at random to N boxes
  - How many cells are occupied?
  - What is the expected number of balls in the first box?
  - What is the expected maximum for the number of balls assigned to any cell?