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
  • $T(N) = \max \{\text{over inputs } I \text{ of size } N\} \ T_A(I)$

• Average case runtime
  • $T(N) = \text{average}\ \{\text{over inputs } I \text{ of size } N \text{ using probability distribution } P\} \ T_A(I)$
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)

$m_1$: $w_1$ $w_2$

$m_2$: $w_2$ $w_1$

$w_1$: $m_2$ $m_1$

$w_2$: $m_1$ $m_2$

$m_1$

$m_2$

$w_1$

$w_2$
Formal Problem

- **Input**
  - Preference lists for $m_1, m_2, ..., m_n$
  - Preference lists for $w_1, w_2, ..., w_n$

- **Output**
  - Perfect matching $M$ satisfying stability property:
    
    If $(m', w') \in M$ and $(m'', w'') \in M$ then
    
    $(m' \text{ prefers } w' \text{ to } w'') \text{ or } (w'' \text{ prefers } m'' \text{ to } m')$

    [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 m₂
    If w prefers m to m₂, w accepts m, dumping m₂
    If w prefers m₂ 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 (m₂, w) is matched
        if w prefers m to m₂
            unmatch (m₂, 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

\[(m_1, w_1) \in M, (m_2, w_2) \in M\]

\[m_1\text{ prefers } w_2\text{ to } w_1\]

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

\[
\begin{align*}
\text{m}_1: & \quad w_1 \quad w_2 \quad w_3 \\
\text{m}_2: & \quad w_1 \quad w_3 \quad w_2 \\
\text{m}_3: & \quad w_1 \quad w_2 \quad w_3 \\
\text{w}_1: & \quad m_2 \quad m_3 \quad m_1 \\
\text{w}_2: & \quad m_3 \quad m_1 \quad m_2 \\
\text{w}_3: & \quad m_3 \quad m_1 \quad m_2 \\
\end{align*}
\]

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

\[ \begin{align*}
  m_1 &: w_8 \ w_3 \ w_1 \ w_5 \ w_9 \ w_2 \ w_4 \ w_6 \ w_7 \ w_{10} \\
  m_2 &: w_7 \ w_{10} \ w_1 \ w_9 \ w_3 \ w_4 \ w_8 \ w_2 \ w_5 \ w_6 \\
  \ldots \\
  w_1 &: m_1 \ m_4 \ m_9 \ m_5 \ m_{10} \ m_3 \ m_2 \ m_6 \ m_8 \ m_7 \\
  w_2 &: m_5 \ m_8 \ m_1 \ m_3 \ m_2 \ m_7 \ m_9 \ m_{10} \ m_4 \ m_6 \\
  \ldots \\
\end{align*} \]

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?

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

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

$$E(T) = E(t_1 + t_2 + \cdots + t_n)$$

$$= E(t_1) + E(t_2) + \cdots + E(t_n)$$

$$= \frac{1}{p_1} + \frac{1}{p_2} + \cdots + \frac{1}{p_n}$$

$$= \frac{n}{n} + \frac{n}{n - 1} + \cdots + \frac{n}{1}$$

$$= n \cdot \left( \frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{n} \right)$$

$$= n \cdot H_n.$$
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?