http://www.cs.unh.edu/~ruml/cs758

1 handout: slides
Summary
### What We've Covered

#### Summary

<table>
<thead>
<tr>
<th>Topics</th>
<th>Criteria</th>
<th>Everything Else</th>
<th>AI</th>
<th>Break</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Topics

- **Stable Matching**

#### Criteria

- **AI**
- **Break**
- **Evaluation**

#### Everything Else

- **Aside**

### Topics

<table>
<thead>
<tr>
<th>Week</th>
<th>Class</th>
<th>Date</th>
<th>Topic</th>
<th>Reading</th>
<th>Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Sept 3</td>
<td>intro, big-O</td>
<td>2, 3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>Sept 8</td>
<td>linear sorting</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>Sept 11</td>
<td>heaps</td>
<td>6</td>
<td>asst 1 (sorting)</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>Sept 15</td>
<td>hashing</td>
<td>11</td>
<td>asst 2 (float addition)</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>Sept 17</td>
<td>binary trees</td>
<td>12, 13</td>
<td>asst 3 (babbler)</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>Sept 22</td>
<td>red-black trees</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>Sept 29</td>
<td>red-black deletion</td>
<td>12, 13</td>
<td>asst 4 (I/O scheduling)</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>Oct 1</td>
<td>dynamic programming</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>Oct 6</td>
<td>knapsack</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>Oct 8</td>
<td>parsing</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>13</td>
<td>Oct 13</td>
<td>greedy</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>14</td>
<td>Oct 15</td>
<td>greedy</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>15</td>
<td>Oct 20</td>
<td>Midterm Exam</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>16</td>
<td>Oct 22</td>
<td>union-find, components</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>17</td>
<td>Oct 27</td>
<td>spanning trees</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>18</td>
<td>Oct 29</td>
<td>shortest paths</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>19</td>
<td>Nov 3</td>
<td>all pairs paths</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>20</td>
<td>Nov 5</td>
<td>network flow</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>21</td>
<td>Nov 10</td>
<td>matching</td>
<td>26.3</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>22</td>
<td>Nov 12</td>
<td>NP-completeness</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>23</td>
<td>Nov 17</td>
<td>satisfiability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>24</td>
<td>Nov 19</td>
<td>clique</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>25</td>
<td>Nov 24</td>
<td>undecidability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>26</td>
<td>Dec 1</td>
<td>approximation</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>27</td>
<td>Dec 3</td>
<td>backtracking</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>28</td>
<td>Dec 8</td>
<td>wildcard slot</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>29</td>
<td>Dec 10</td>
<td>everything else</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>Dec 17</td>
<td>Final Exam, 1-3pm?</td>
<td></td>
<td>(consult registrar for details)</td>
</tr>
</tbody>
</table>

**Evaluation**

- asst 1 (sorting)
- asst 2 (float addition)
- asst 3 (babbler)
- asst 4 (I/O scheduling)
- asst 5 (spelling correction)
- asst 6 (sequence alignment)
- asst 7 (knapsack)
- asst 8 (make)
- asst 9 (tree hatching)
- asst 10 (route planning)
- asst 11 (flow)
- asst 12 (NP proof)
- asst 13 (NP proof)
- asst 14 (algorithm design)
How to Choose an Algorithm

Summary

- Topics
- Criteria
- Everything Else
- AI
- Break
- Evaluation

Stable Matching

- running time
- memory use
- solution quality (for optimization problems)
- guarantees on time, memory, or cost
- implementation complexity
  - correctness
  - ease of testing
  - time to write
  - ease of maintenance
- generality
- popularity
  - ease of maintenance
  - correctness
- input required
topics
- geometry
- strings
- linear programming
- cryptography
- numerical analysis
- FFT

approaches
- randomized algorithms
- on-line algorithms
- parallel, distributed
- external memory, cache-oblivious
- models: quantum, DNA
CS 730/730W/830 Introduction to Artificial Intelligence
spring 2015, fall 2015

Chris Amato: CS 980 Multiagent AI
spring 2015, fall 2015

CS 980 Planning for Robots
spring 2016

Chris Amato: CS 7??/8?? Probabilistic AI and ML
spring 2016

And the UNH AI Research Group meets weekly all year round
Google “UNH AI Group” for details
Final Exam: Wed Dec 17, 1-3pm, Kingsbury N113
no books, notes, gadgets, ...
bring questions from practice final to (last!) recitation
We do read these (and so does my boss).

A. Class
1. Things you liked
2. Suggestions for improvement

B. Wheeler Ruml
1. Things you liked
2. Suggestions for improvement

C. Scott Kiesel
1. Things you liked
2. Suggestions for improvement

Duplication from CS 515? Background assumed?
How to make class appropriate for juniors?
Advice for students next year?
Stable Matching

The Problem

Stable Matchings

Gale-Shapley

EOLQs
match \( n \) people to \( n \) openings

- each person has ranked preferences, each employer has ranked preferences
- **stable**: no incentive to change: for each employer and each person not employed there, either employer prefers every current employee over than person or person prefers current job
- medical residencies, NYC public schools. “marriage”

note that people are autonomous: act in their own interest

internet, economics, game theory, multi-agent systems

example:

\[
\begin{array}{c|c}
\text{a:} & x, y \\
\text{b:} & y, x \\
\text{x:} & b, a \\
\text{y:} & a, b
\end{array}
\]
one always exists. there can be more than one!

**Gale-Shapely algorithm**

1. while there’s a free employer who hasn’t tried every candidate
2. employer proposes to most preferred candidate it hasn’t yet tried
3. if candidate is free or prefers this to current, match them

employer situation gets worse and worse
candidate situation gets better and better
running time:
running time: each employer proposes at most once to each candidate

correctness:
1. if employer is unmatched, then it has an untried candidate: if it had tried all, it would be matched unless all candidates were matched, which is impossible

2. GS finds perfect matching: GS runs until every free employer tries every candidate and by 1 must therefore be matched

3. GS finds stable matching: Given match with \( a, y \) and \( b, x \), assume an instability where \( a \) prefers \( x \) to \( y \) and \( x \) prefers \( a \) to \( b \). \( a \)'s last proposal was to \( y \). \( a \) must have proposed to \( x \) earlier and been rejected in favor of someone better. But then it’s impossible for \( x \) to end up with \( b \). So no such instability is possible.
For example:

- What’s still confusing?
- What question didn’t you get to ask today?
- What would you like to hear more about?

Please write down your most pressing question about algorithms and put it in the box on your way out.

*Thanks!*