CSCI 3110
Fun with Algorithms

Christopher Whidden
cwhidden@dal.ca

Faculty of Computer Science
Dalhousie University

Summer 2019

(Adapted from Slides by Norbert Zeh)

Stable Matching: An Introductory Example

Given:

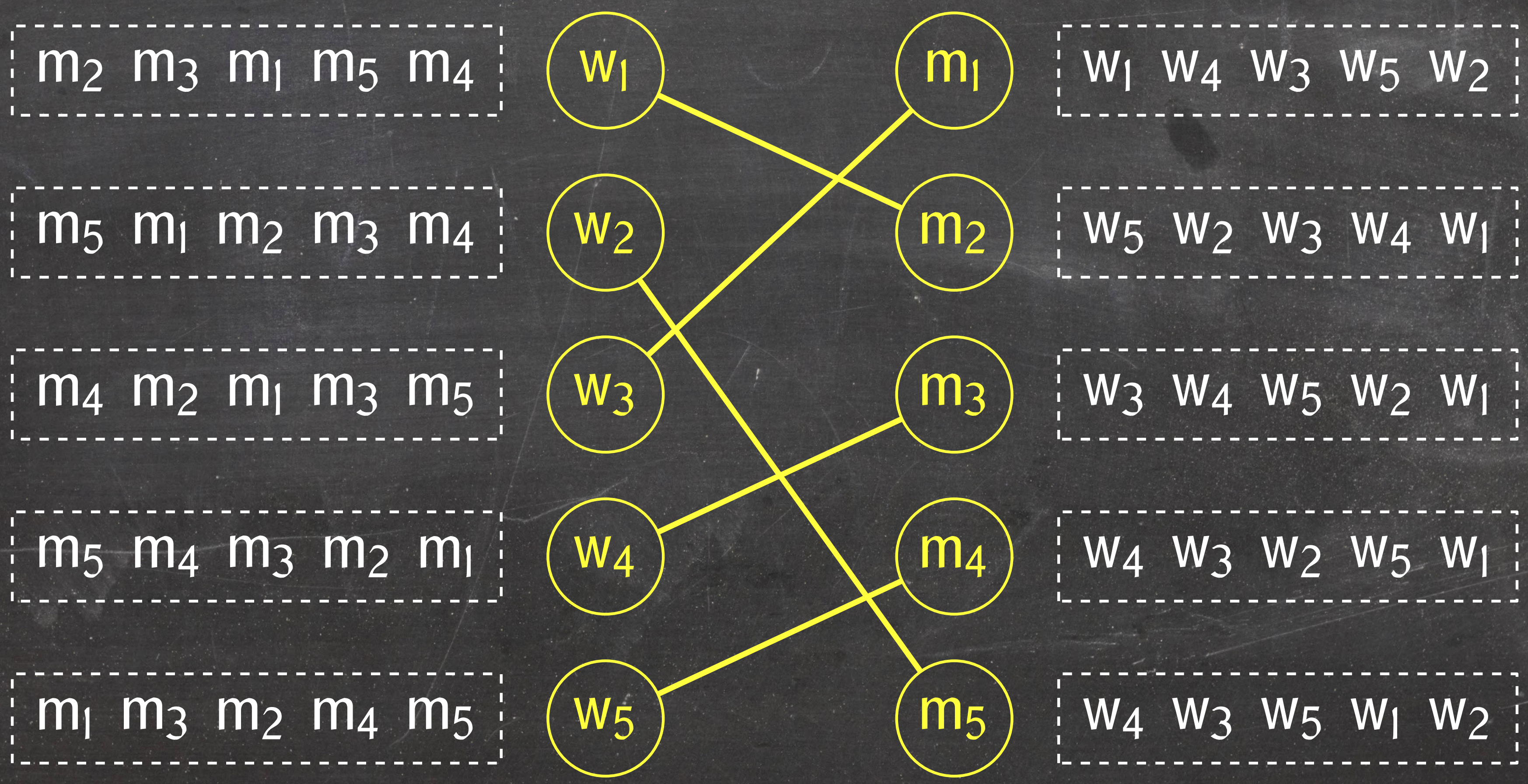
- n women $w_1, w_2, \ldots, w_n$
- n men $m_1, m_2, \ldots, m_n$
- A preference list for each

m_2 m_3 m_1 m_5 m_4	w_1	m_1	w_1 w_4 w_3 w_5 w_2
m_5 m_1 m_2 m_3 m_4	w_2	m_2	w_5 w_2 w_3 w_4 w_1
m_4 m_2 m_1 m_3 m_5	w_3	m_3	w_3 w_4 w_5 w_2 w_1
m_5 m_4 m_3 m_2 m_1	w_4	m_4	w_4 w_3 w_2 w_5 w_1
m_1 m_3 m_2 m_4 m_5	w_5	m_5	w_4 w_3 w_5 w_1 w_2

Stable Matching: An Introductory Example

Output:

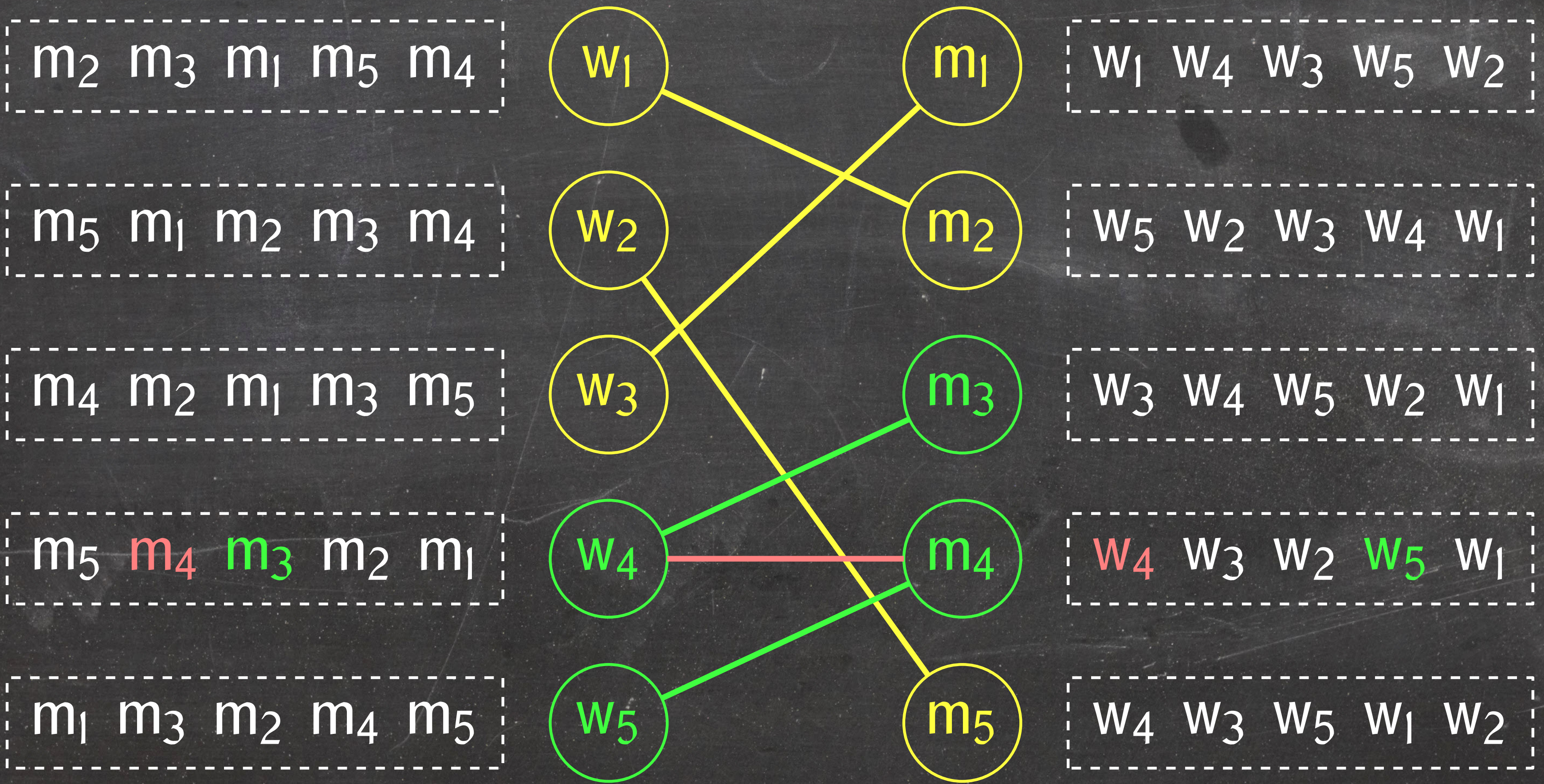
- A set of n marriages $\{(w_{i_1}, m_{j_1}), ((w_{i_2}, m_{j_2}), \ldots, (w_{i_n}, m_{j_n})\}$
- Every man is married
- Every woman is married
- The marriages are stable



Stable Matching: An Introductory Example

A pair of marriages (m, w) and (m', w') is unstable if

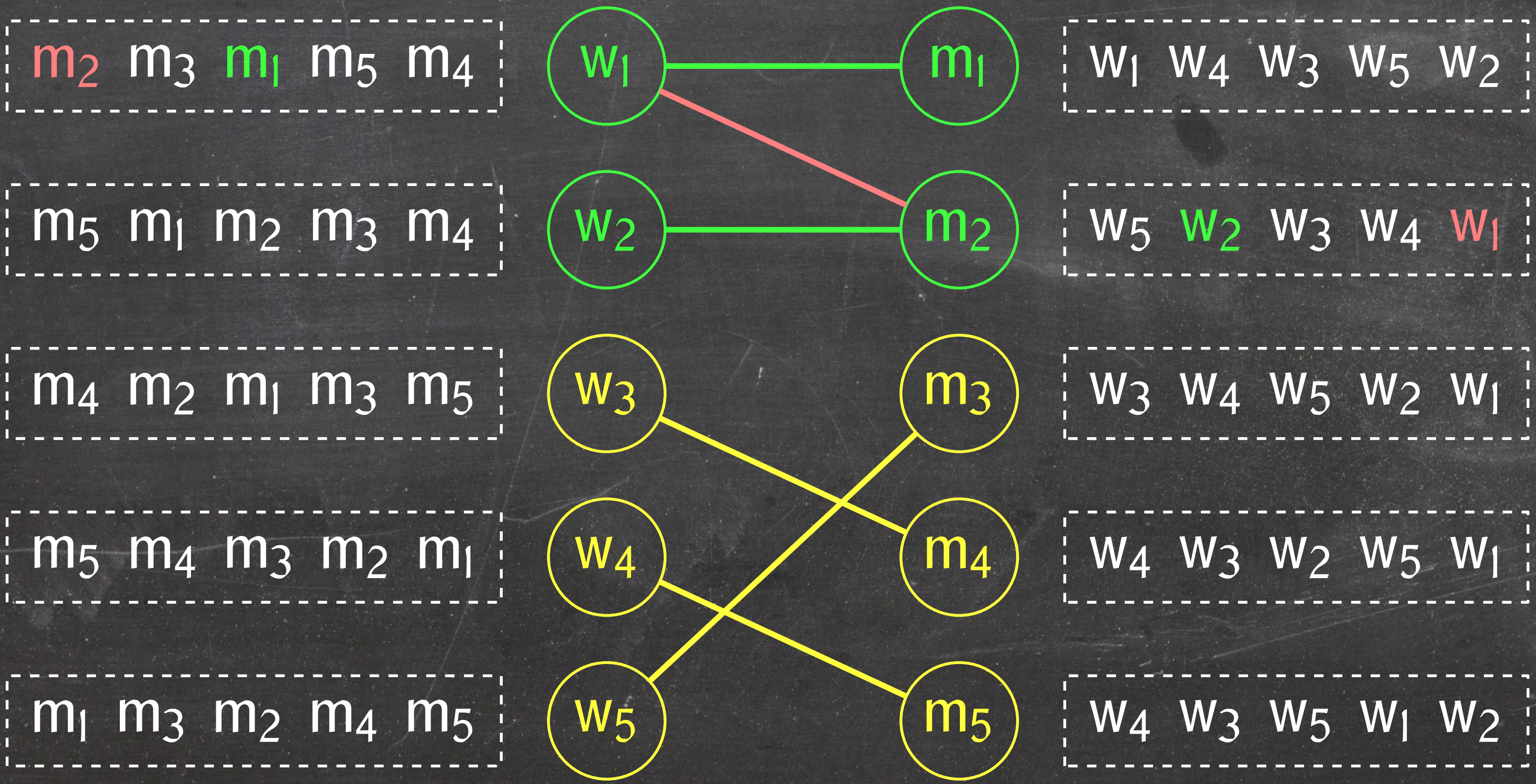
- w prefers m' over m ($m' \prec_w m$)
- m' prefers w over w' ($w \prec_{m'} w'$)



Stable Matching: An Introductory Example

A pair of marriages (m, w) and (m', w') is unstable if

- w prefers m' over m ($m' \prec_w m$)
- m' prefers w over w' ($w \prec_{m'} w'$)



Stable Matching: A Solution Inspired By Real Life

StableMatching(M, W)

```
while there exists an unmarried man m
    do m proposes to the most preferable woman w he has not proposed to yet
        if w is unmarried or likes m better than her current partner m′
            then if w is married
                    then w divorces m′
                 w marries m
```

Stable Matching: A Solution Inspired By Real Life

StableMatching(M, W)

```
while there exists an unmarried man m
    do m proposes to the most preferable woman w he has not proposed to yet
        if w is unmarried or likes m better than her current partner m′
            then if w is married
                    then w divorces m′
                 w marries m
```

Questions we can and should ask about the algorithm:

- Is there always a stable matching?
- Does the algorithm always terminate?
- Does the algorithm always produce a stable matching?
- How efficient is the algorithm? Can we bound its running time?

Course Outline

- Correctness proofs
- Analysis of resource consumption
- Algorithm design techniques
  - Graph exploration
  - Greedy algorithms
  - Divide and conquer
  - Dynamic programming
  - Data structuring
  - Randomization
- NP-completeness and intractability

General Information

Instructor:	Christopher Whidden
Office:	CS 315
Office hours:	Thursday 1:00–2:00
Email:	cwhidden@dal.ca
Textbook:	Cormen, Leiserson, Rivest, Stein. Introduction to Algorithms. 3rd edition, MIT Press, 2009.
	Zeh. Data Structures. CSCI 3110 Lecture Notes, 2005.
Website:	http://www.cs.dal.ca/~whidden/CSCI3110
TAs:	Yuhan Fu Mozhgan Saeid Younan Gao
Midterm:	July 4

Grading

- 10 Assignments (A)

  The best 8 count. Each carries equal weight.
- Midterm (M)
- Final (F)

$$\text{Final grade} = \max \begin{pmatrix} 60\% \cdot F + 40\% \cdot A \\ 40\% \cdot F + 20\% \cdot M + 40\% \cdot A \end{pmatrix}$$

Collaboration, Plagiarism, Late Assignments

Collaboration

- Groups of up to three people are allowed to collaborate on assignments.
- Every group hands in one set of solutions; every group member gets the same marks.
- Collaboration between groups is not allowed!

Plagiarism

- Plagiarism will not be tolerated.
- Collaboration between groups is a form of plagiarism.

Late assignments

. . . will not be accepted. Assignments missed for a reason documented by a Student Declaration of Absence will be covered by your final exam score.

Please see course website for a detailed discussion of these rules.

Things I Expect You To Know

- Basic rules concerning logarithms
- Basic rules concerning limits
- Basic derivatives
- Propositional logic
- Elementary combinatorics (counting permutations, combinations, ...)
- Elementary probability theory (linearity of expectation, ...)
- Elementary data structures (arrays, lists, stacks, queues, ...)
- Standard sorting algorithms (insertion sort, quick sort, merge sort)
- Binary heaps