1/7/16

HW 1 assigned

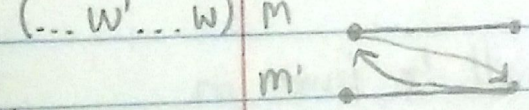
designing an algorithm
- understand problem
- use an algorithm
- analyze
- improve -- check if optimal algorithm is possible
  - prove this is the best you can do

matching: match 2 people to each other within a group
- match 1 to 1
- usually certain objectives/criteria
- not everyone may get matched
- ranges in complexity

example: basic matching problem
- 2 groups: male + female w/ same number of people
- want perfect matching (everyone is matched)

ordering pair — M — W — always do the simplest thing (arbitrary)

all combos allowed
(1243)
(2314) — 2 — (3142) — each person has a preference list
(2314) — 3
(4312) — 4



a match is represented by a straight line
higher priority gets an arrow

this matching (if in output) is called unstable due to the arrows pointing in a loop

example: stable matching

objectives
- perfect matching
- stable matching for all pairs

find a perfect stable matching
everyone is initially unmatched
or start w/ algorithm and fix instability

2 step process
- m proposes to a w
- w will choose to accept depending on algorithm

proposed algorithm

pick an arbitrary m
go to the highest priority w or an arbitrary w

generic i^{th} step: (no assumption of past unless proven)
if m' is unmatched pick highest w who he hasn't proposed to
if w is unmatched, accept the proposal
if w is matched, check w's preference list if m' is higher on the list than the current m in the match, w breaks the match and accept m's proposal

stopping condition: m has asked all preferences and is still unmatched

properties of algorithm:
as new matches occur, they will be lower in the priority list for M but higher in priority for W
if a W is matched, will always be matched

prove no unstable pairs:

by contradiction, assume algorithm outputs an unstable pair

(...W'...W) M W

M' W' (...M...M')

Case 1: M did not propose to W'

Case 2: M did propose to W'

case 1: contradiction -- did not follow algorithm b/c M always asks from high to low priority

case 2: W' was already matched but did not accept (to M") b/c M is lower in priority -- contradiction!

W's matches always get better in priority

M" ≥ M ≥ M'

W' would not go lower in priority in our algorithm

prove perfect match: