Prepared by,

Chetan Shetty

Assistant Professor

Department of CSE

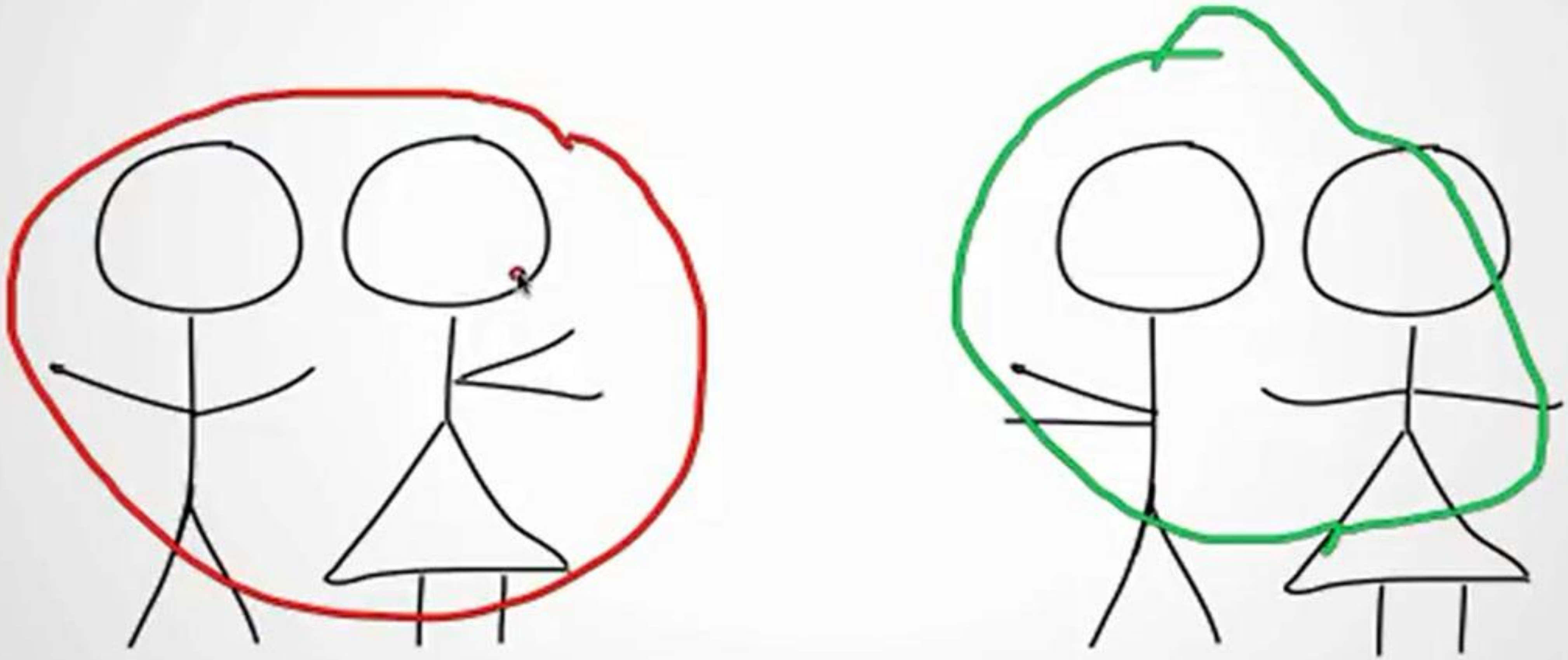
MSRIT

Bangalore

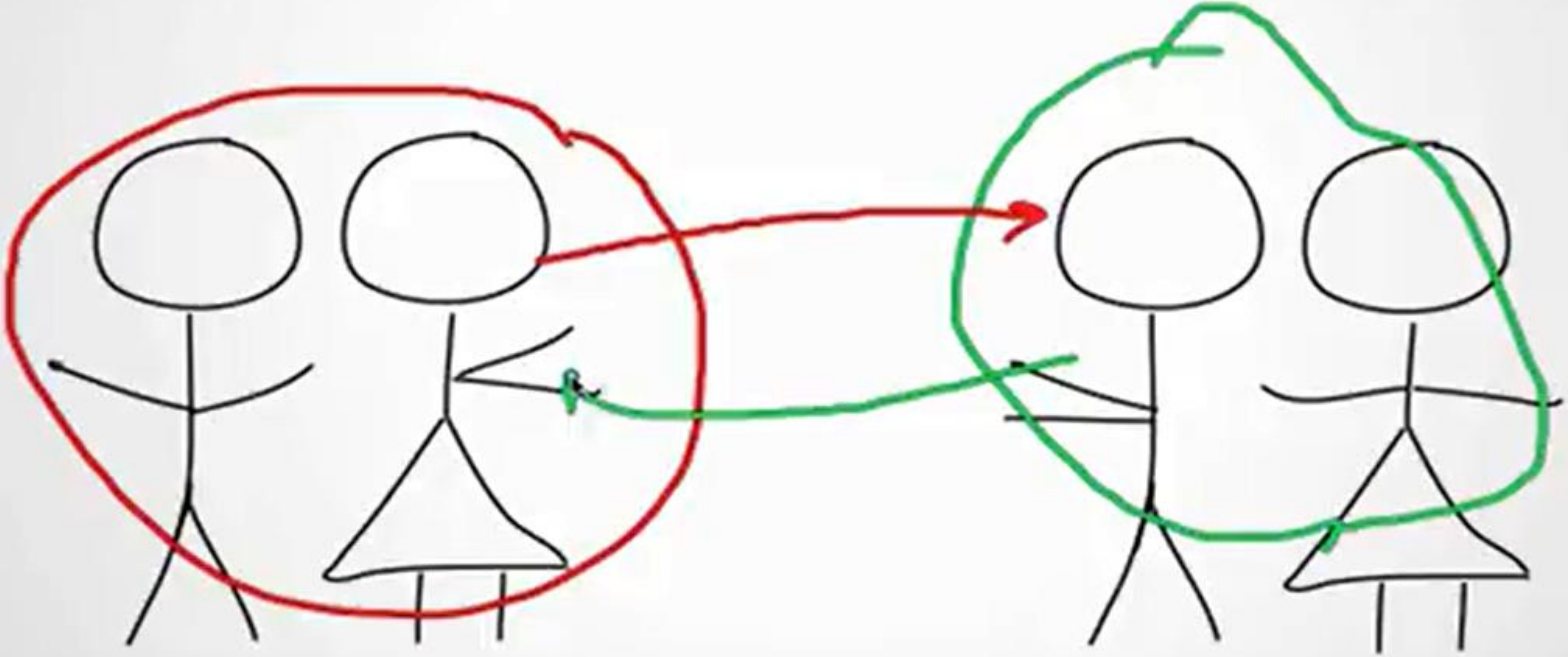
A Good matching/pairing

- Maximize the number of people who get their first match?
- Maximize the average satisfaction?
- Maximize the minimum satisfaction?
- Can anything go wrong?

An unstable situation



An unstable situation



INSTABILITY

- There are two pairs *(m, w)* and *(m', w')* in *S* with the property that
  - *m* prefers *w'* to *w*
  - *w'* prefers m to m'
- pair *(m, w)* is an *instability* with respect to *S*:*(m, w)* does not belong to *S.*
- Our goal, then, is a set of marriages with no instabilities. A matching *S* is *stable* if

  (i) it is perfect

  (ii) there is no instability with respect to *S*.

Example 1

- Suppose we have a set of two men, {*m*, *m'*}, and a set of two women, {*w*, *w'*}. The preference lists are:
  - *m* prefers *w* to *w'*.
  - *m'* prefers *w* to *w'*.
  - *w* prefers *m* to *m'*.
  - *w'* prefers *m* to *m'*.
- There is a unique **stable** matching here, consisting of the pairs *(m, w)* and *(m', w')*.

- *(m', w)* and *(m, w')*, would not be a stable match, because the pair *(m, w)* would form an instability with respect to this matching.

M	W	W'
M'	W	W'

W	M	M'
W'	M	M'

(m', w) (m, w')

(m, w) (m', w')

- As both m and want to leave their partners and pair up.

- **Example 2:**
  - m prefers w to w'.
  - m' prefers w' to w.
  - w prefers m' to m.
  - w' prefers m to m'.

- (m, w) and (m', w') is stable, because both men are as happy as possible, so neither would leave their matched partner.

- (m', w) and (m, w') is also stable, for the complementary reason that both women are as happy as possible.

- Possible for an instance to have more than one stable matching.

M	W	W’
M’	W’	W

W	M’	M
W’	M	M’

- **<u>Basic Steps:</u>**
- **Initially, everyone is unmarried.**
  - if an unmarried man '*m'* chooses the woman '*w'* who ranks highest on his preference list and *proposes* her.
  - A man '*m'* whom '*w'* prefers, may or may not receive a proposal from m'.
  - So a natural idea would be to have the pair *(m, w)* enter an intermediate state—***engagement***.

- Suppose we are now at a state in which some men and women are *free*—not engaged
  - An arbitrary free man '*m'* chooses the highest-ranked woman '*w'* and propose her.
  - If *w* is also free, then '*m'* and '*w'* become engaged.
  - Otherwise, *w* is already engaged to some other man '*m'* i.e. she determines which of *m* or *m* ranks higher on her preference list.
  - Finally, the algorithm will terminate when no one is free.

Designing the Algorithm

1. Initially all $m \in M$ and $w \in W$ are free
2. While there is a man *m* who is free and hasn't proposed to every woman
3. Choose such a man *m*
4. Let *w* be the highest-ranked woman in *m*'s preference list
5. to whom *m* has not yet proposed
6. If *w* is free then
7. *(m, w)* become engaged
8. Else *w* is currently engaged to *m'*
9. If *w* prefers *m'* to *m* then
10. *m* remains free
11. Else *w* prefers *m* to *m'*
12. *(m, w)* become engaged
13. *M'* becomes free
14. Endif
15. Endif
16. Endwhile
17. Return the set *S* of engaged pairs

Analyzing the Algorithm

1. **w remains engaged from the point at which she receives her first proposal; and the sequence of partners to which she is engaged gets better and better (in terms of her preference list).**

2. **The sequence of women to whom m proposes gets worse and worse (in terms of his preference list).**

3. ***The G-S algorithm terminates after at most n^2 iterations of the* While *loop.***
   - ***Proof:***
   - let P*(t)* denote the set of pairs *(m, w)* such that '*m*' has proposed to '*w*' by the end of iteration '*t*'.
   - For all *t*, the size of P*(t* + 1*)* is strictly greater than the size of P*(t)*.
   - But there are only n^2 possible pairs of men and women in total, so the value of P*(·)* can increase at most n^2 times over the course of the algorithm.
   - It follows that there can be at most n^2 iterations.

MENS PREFERENCE LIST

V	A	B	C	D	E
W	B	C	D	A	E
X	C	D	A	B	E
Y	D	A	B	C	E
z	A	B	C	D	E

WOMENS PREFERENCE LIST

A	W	X	Y	Z	V
B	X	Y	Z	V	W
C	Y	Z	V	W	X
D	Z	V	W	X	Y
E	V	W	X	Y	Z

N(N-1)+1

N:Number of Men

(N-1) Number of proposes left out

1: Proposal during first iteration

4. ***If m is free at some point in the execution of the algorithm, then there is a woman to whom he has not yet proposed.***

Proof:

- Suppose there comes a point when *m* is free but has already proposed to every woman.
- Then by (1), each of the *n* women is engaged at this point in time.
- Since the set of engaged pairs forms a matching, there must also be *n* engaged men at this point in time.
- But there are only *n* men total, and *m* is not engaged, so this is a contradiction.

5. ***The set S returned at termination is a perfect matching*.**

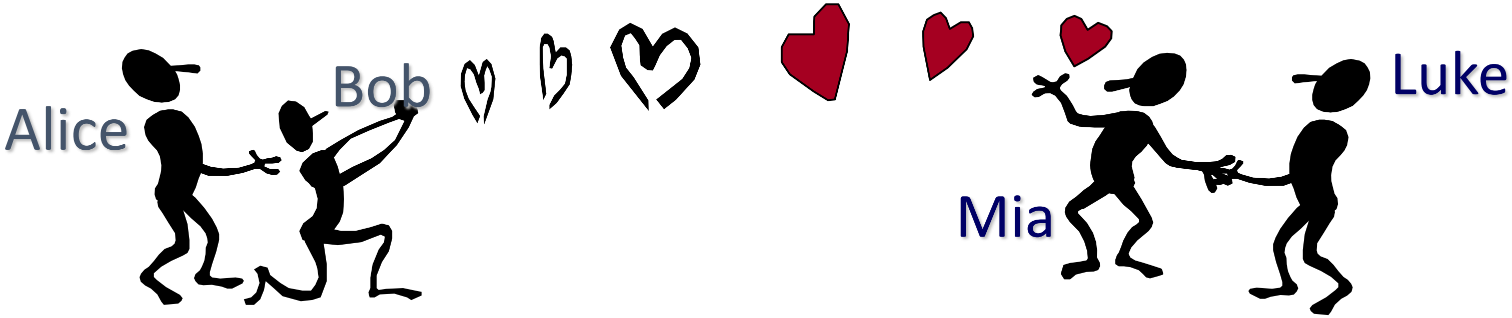
Proof:

- The set of engaged pairs always forms a matching.
- Let us suppose that the algorithm terminates with a free man *m*.
- At termination, it must be the case that '*m'* had already proposed to every woman, for otherwise the While loop would not have exited.
- But this contradicts (4), which says that **there cannot be a free man who has proposed to every woman**.

6. ***Consider an execution of the G-S algorithm that returns a set of pairs S. The set S is a stable matching.***

Proof:

- We have already seen, in (5), that *S* is a perfect matching.
- Thus, to prove ***S*** is a stable matching, we will assume that there is an instability with respect to *S* and obtain a contradiction.
- Instability would involve two pairs, *(m, w)* and *(m′,w′)*, in "*S*" with the properties that
  - *m* prefers *w′* to *w*,
  - *w′* prefers *m* to *m′*



-

- In the execution of the algorithm that produced ***S***, ***m*'s** last proposal was, by definition, to ***w***.

- Did *m* propose to *w'* at some earlier point in the execution:

  - **NO**: then *w* must occur higher on *m*'s preference list than *w'*, contradicting our assumption that *m* prefers *w'* to *w*.

  - **YES:** then he was rejected by *w'* in favor of some other man *m''*, whom *w'* prefers to *m*.

- *m'* is the final partner of *w'*, so either *m''* = *m'* or, by (1), *w'* prefers her final partner *m'* to *m''*; either way this contradicts our assumption that *w'* prefers *m* to *m'*.

- It follows that *S* is a stable matching.

Extensions

All Executions Yield the Same Matching

- We'll show that each man ends up with the **"best possible partner".**

- First, we will say that a woman w is a *valid partner* of a man m if there is a stable matching that contains the pair (m, w). We will say that w is the *best valid partner* of m if w is a valid partner of m, and no woman whom m ranks higher than w is a valid partner of his.

- We will use *best(m)* to denote the best valid partner of m.

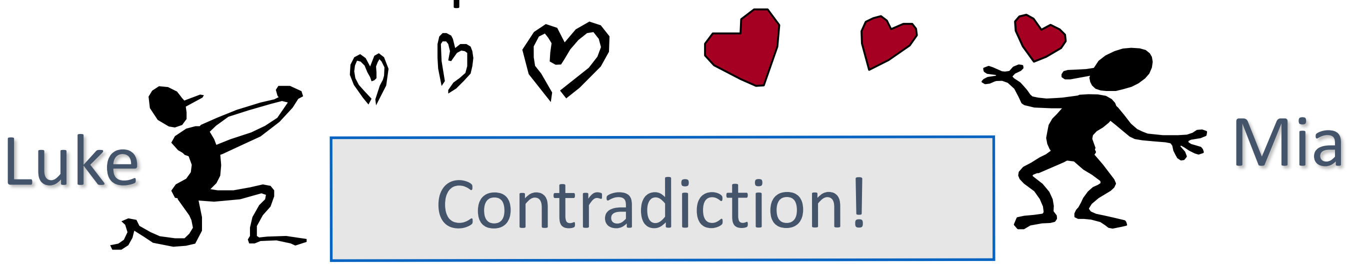
- Now, let $S*$ denote the set of pairs $\{(m, best(m)) : m \in M\}$.

7. ***Every execution of the G-S algorithm results in the set S∗.***

Proof:

- Let us suppose, by way of contradiction, that some execution E of the G-S algorithm results in a matching *S* in which some man is paired with a woman who is **not his best valid partner**.

(m, w') is not the best valid pair.



- Since men propose in decreasing order of preference, this means that some man is rejected by a valid partner during the execution E of the algorithm.

- So consider the first moment during the execution E in which some man, say *m*, is rejected by a valid partner *w*.
    - **m** was rejected by **w.**

- Again, since men propose in decreasing order of preference, and since this is the first time such a rejection has occurred, it must be **that *w* is *m*'s best valid partner *best(m)*.**

- The rejection of *m* by *w* may have happened either because ***m* proposed and was turned down** in favor of ***w*'s existing engagement**, or because ***w* broke her engagement to *m* in favor of a better proposal m'**.

- But either way, at this moment *w* forms or continues an engagement with a man *m* whom she prefers to *m*.

M	W	W'

W	M'	m

- (m, best(m)) (w, m')

- Since *w* is a valid partner of *m*, there exists a stable matching *S* containing the pair *(m, w)*.

S* ={(m,w) , (m',w')prefered**}**

- Now we ask: Who is *m′* paired with in this matching?
- Suppose it is a woman ***w′ != w*.**

W	M’	M
W’	M	M’

- Since the rejection of *m* by *w* was the first rejection of a man by a valid partner in the execution E, it must be **that *m’* had not been rejected by any valid partner** at the point in E when he became engaged to *w*.

M	w	W’
m’	w	w’

W	M’	M
W’	M	M’

- m’ was not rejected by a valid partner.
- So m’ prefers w to w’ and w prefers m’ to m

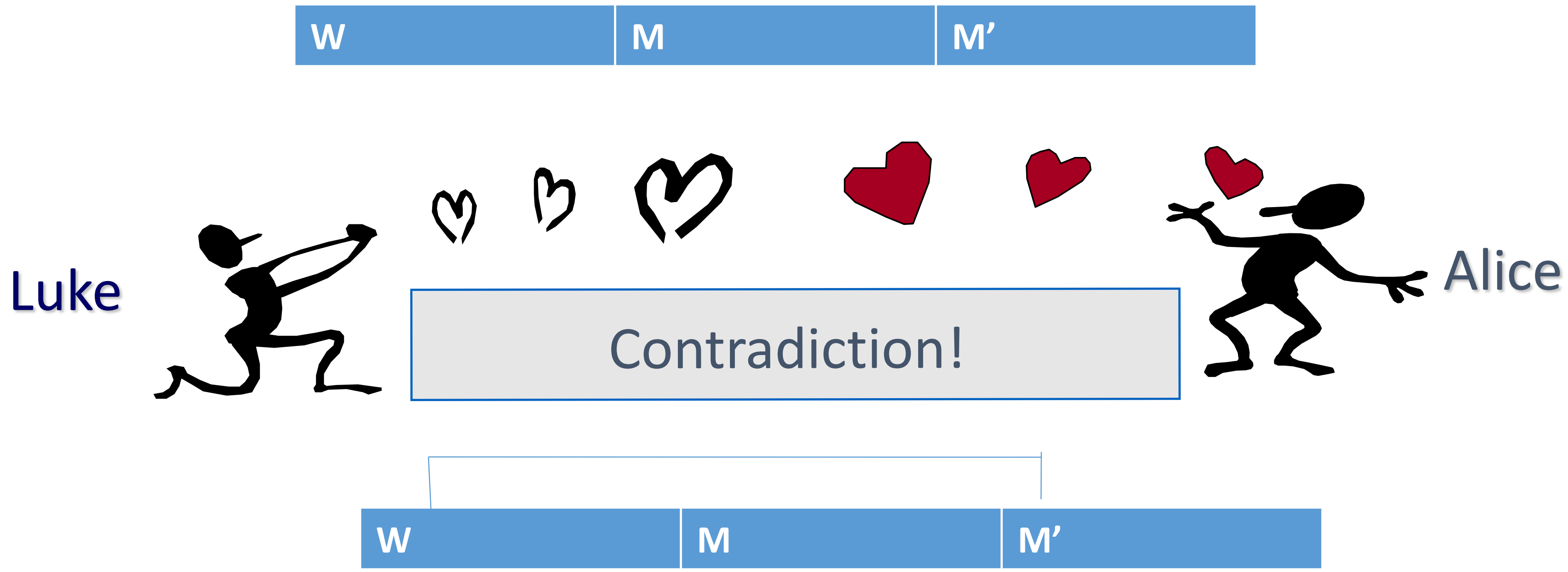
- Since *(m', w) is instability in s* and !E s**.

- This contradicts our claim that *S** is stable and hence contradicts our initial assumption that **some man is paired with a woman who is not his best valid partner (m, w').**

8. ***In the stable matching S∗, each woman is paired with her worst valid partner.***

Proof:

- Suppose there were a **pair *(m, w)*** in *S** such that *m* is not the worst valid partner to *w*.



- Then there is a stable matching *S’* in which *w* is paired with a man *m’* whom she likes less than *m*.
- **S’(m, w’)**

W	M	M’
W’	M’	M

- In *S’*, ***m*** is paired with a woman ***w’ != w***; since ***w*** is the best valid partner of ***m***, and ***w’*** is a valid partner of ***m***, we see that *m* prefers **_w_ to _w’_.**

M	W	W’

- But from this it follows that *(m, w)* is an instability in ***S’*, ** contradicting the claim that *S* is stable and hence contradicting our initial assumption that **_m_ is not the worst valid partner to _w_.**

Interval scheduling

- we have a resource lecture room, many people request to use the resources for periods of time.

- Assume that resource can be used by atmost 1 person at a time.
- Scheduler wants to accept a subset of these requests, rejecting all others, so that the accepted requests doniot overlap in time.

- **Goal:** Maximize the number if request accepted.

- There will be n requests labeled 1,2,3,4,….,n, with each request I specifying start time Si and finish time fi, such that si<fi for all i.

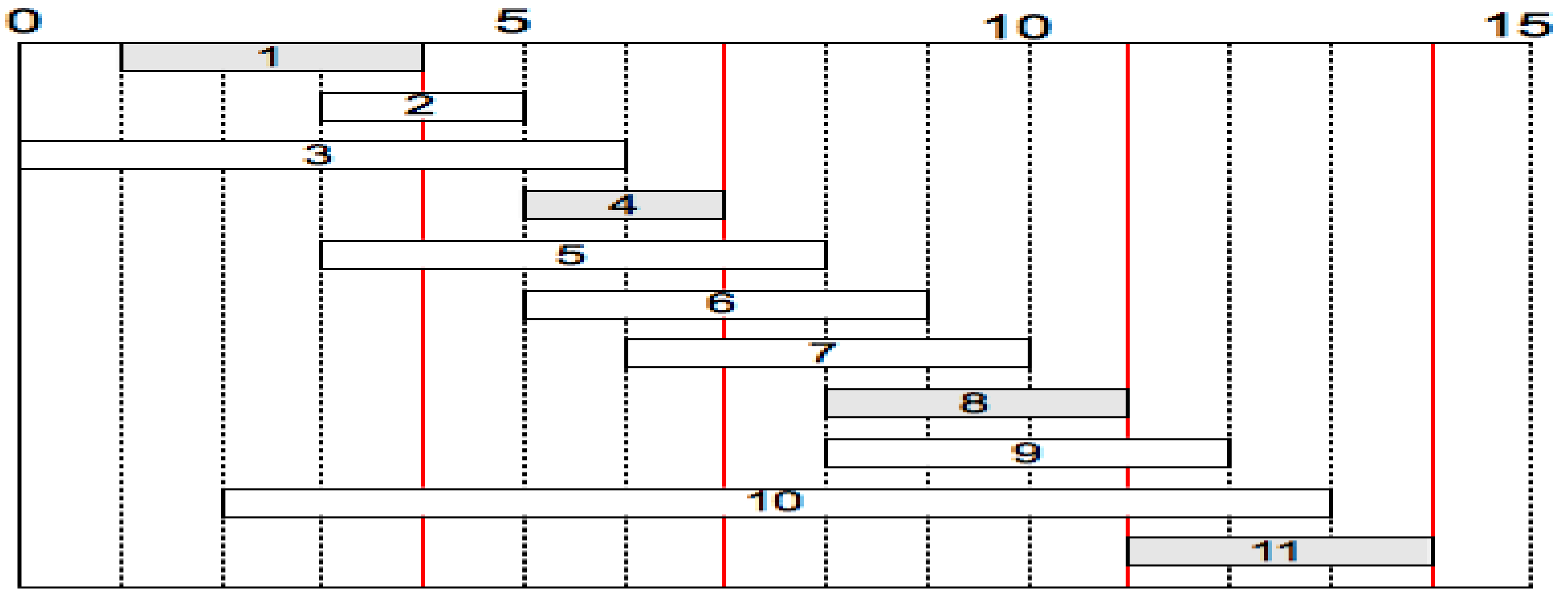
- 2 requests I and j are compatible if the requested intervals donot overlap. i.e either I is for an earlier time interval than request j (fi<=Sj) or (fj<=si)

- A subset A of requests is compatible if all pairs of requests I,j E A

- Goal is to select compatible subset of requests of maximum possible size.

- A single compatible set of size 4, and this is the largest compatible set.

- **Algorithm:** that orders the set of requests according to certain heuristic and then greedily process them, selecting as large compatible subset it can.

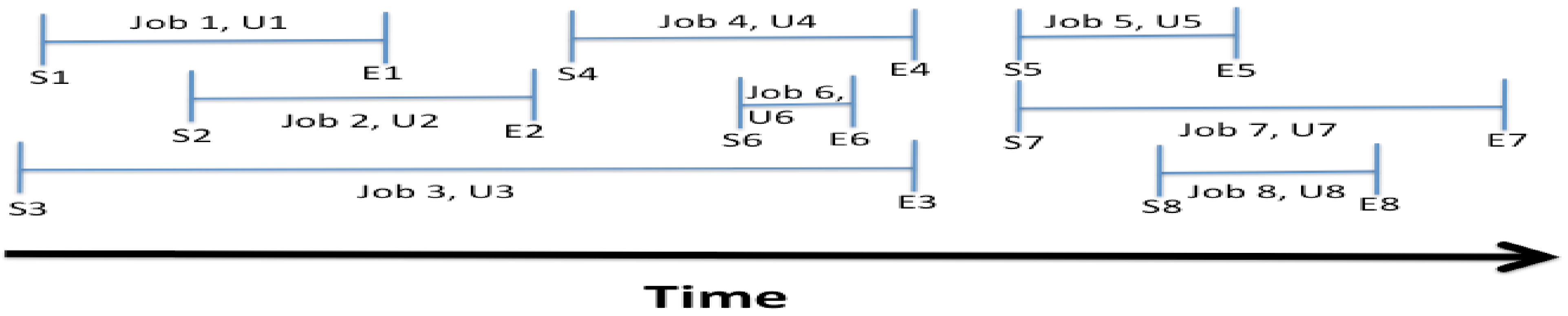


Weighted interval Scheduling

- In this, each request interval I has an associated value or weigth "Vi>0", we can picture this as amount of money we will make from the ith individual if we schedule his or her request.
- Goals is to find the a compatible subsets of intervals of maximum total value.
- If Vi=1 for each i is simply the basic interval scheduling.
- Appearance of arbitrary values changes the nature of maximization problem.
- Appearance of arbitrary values changes the nature of maximization problem.

- Ex: if V1 exceeds the sum of all other "Vi", then the optimal solution must include interval 1 regardless of the configuration of full set of intervals. So any algo for this problem must be very sensitive to the values.

- We employ dynamic programming technique that builds up the optimal value overall possible solution in a compact & tabular way, that leads to efficient algorithm.



Weighted Interval Scheduling finds the set of non-overlapping jobs that maximizes the sum of utility.