THE STABLE MATCHING ALGORITHM

The Matching Problem

- Two heterogenous populations (X and Y)
- Every $x \in X$ has a preference ordering over the elements of Y.
- Every $y \in Y$ has a preference ordering over the elements of X.
- Questions: Who should be matched with whom?

The Matching Problem

- Two heterogenous populations (X and Y)
- Every $x \in X$ has a preference ordering over the elements of Y.
- Every $y \in Y$ has a preference ordering over the elements of X.
- Questions: Who should be matched with whom?

- Examples:

i. Marriage / Dating market (X women, Y men)

The Matching Problem

- Two heterogenous populations (X and Y)
- Every $x \in X$ has a preference ordering over the elements of Y.
- Every $y \in Y$ has a preference ordering over the elements of X.
- Questions: Who should be matched with whom?

- Examples:

i. Marriage / Dating market (X women, Y men)

ii. Labor contract (X CEOs, Y firms; ...)

The Matching Problem

- Two heterogenous populations (X and Y)
- Every $x \in X$ has a preference ordering over the elements of Y.
- Every $y \in Y$ has a preference ordering over the elements of X.
- Questions: Who should be matched with whom?

- Examples:

i. Marriage / Dating market (X women, Y men)
ii. Labor contract (X CEOs, Y firms; ...)
iii. Credit (X firms, Y banks)
iv. (X buyers, Y sellers, Z products), etc.

The Matching Problem

- Two heterogenous populations (X and Y)
- Every x ∈ X has a preference ordering over the elements of Y.
- Every y ∈ Y has a preference ordering over the elements of X.
- Questions: Who should be matched with whom?

- Examples:

i. Marriage / Dating market (X women, Y men)
ii. Labor contract (X CEOs, Y firms; ...)
iii. Credit (X firms, Y banks)
iv. (X buyers, Y sellers, Z products), etc.

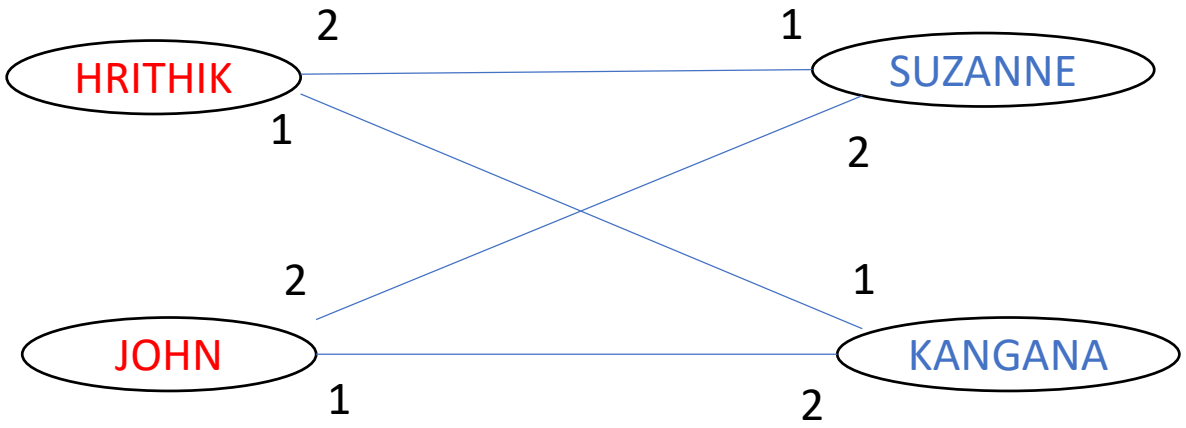
- Extensions:

Many to many: s (x1, ..., xn, y1, ..., yk): Campus Placements.

The Matching Problem: Marriage

- This presentation: marriage market.
- Assume Monogamy: One to One Matching
- Assume n(X) = n(Y): equal sized population.
- How to “optimally” match?
- What do we mean by “optimal” ?
- Let’s look at an example.

Example



Preferences:

Hrithik: Kangana , Suzanne
John: Kangana , Suzanne

Kangana: Hrithik , John
Suzanne: Hrithik , John

Possible Matchings

- Matching 1: {Hrithik – Suzanne , John – Kangana}
- Matching 2: {Hrithik – Kangana , John – Suzanne}

Possible Matchings

- Matching 1: {Hrithik – Suzanne , John – Kangana}
- Matching 2: {Hrithik – Kangana , John – Suzanne}

- In Matching 1: Hrithik prefers Kangana over Suzanne. Kangana too prefers Hrithik over John. So both prefer each other over their mates.

  So they'll cheat !

Possible Matchings

- Matching 1: {Hrithik – Suzanne , John – Kangana}
- Matching 2: {Hrithik – Kangana , John – Suzanne}

- In Matching 1: Hrithik prefers Kangana over Suzanne. Kangana too prefers Hrithik over John. So both prefer each other over their mates.

  So they'll cheat !

- ***Rogue Couple:*** *Given a matching M, two individuals X & Y form a rogue couple if they prefer each other over their mates.*

Possible Matchings

- Matching 1: {Hrithik – Suzanne , John – Kangana}
- Matching 2: {Hrithik – Kangana , John – Suzanne}

- In Matching 1: Hrithik prefers Kangana over Suzanne. Kangana too prefers Hrithik over John. So both prefer each other over their mates.

  So they'll cheat !

- ***Rogue Couple:*** *Given a matching M, two individuals X & Y form a rogue couple if they prefer each other over their mates.*

- In Matching 1: Hrithik & Kangana form a "rogue couple"
- Matching 2 has no rogue couples.

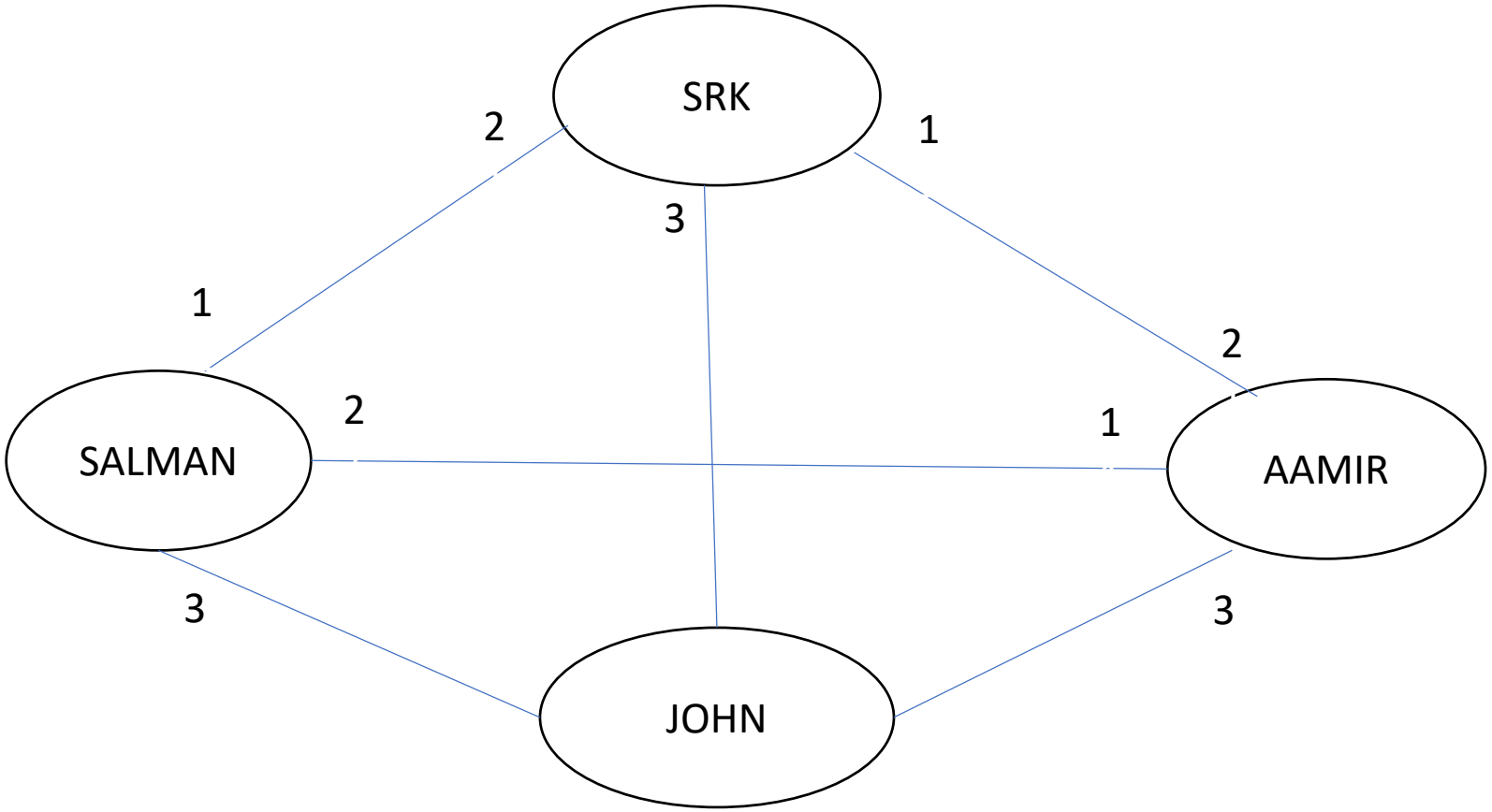
Stable Matching

- **Perfect Matching:** A matching where all individuals are paired is called a perfect matching.

- **Stable Matching:** A perfect matching is said to be stable if there are no “rogue couples”

- In the example: Matching 2 is “stable”.

Example



Preferences:

SRK: Aamir , Salman , John

Salman: SRK , Aamir , John

Aamir: Salman , SRK , John

John’s preferences are inconsequential.

Theorem 1

There does NOT exist a stable match.

Theorem 1

There does NOT exist a stable match.

Proof by contradiction:

Assume that there exists a stable match M.

Theorem 1

There does NOT exist a stable match.

Proof by contradiction:

Assume that there exists a stable match M.

$\Rightarrow$John will be matched with someone. (WLOG by symmetry let's say John matches with SRK)

Theorem 1

There does NOT exist a stable match.

Proof by contradiction:

Assume that there exists a stable match M.

$\Rightarrow$John will be matched with someone. (WLOG by symmetry let's say John matches with SRK)

$\Rightarrow$ Salman is matched with Aamir.

Theorem 1

There does NOT exist a stable match.

Proof by contradiction:

Assume that there exists a stable match M.

$\Rightarrow$John will be matched with someone. (WLOG by symmetry let's say John matches with SRK)

$\Rightarrow$ Salman is matched with Aamir.

$\Rightarrow$ Salman & SRK form a "rogue couple".

Theorem 1

There does NOT exist a stable match.

Proof by contradiction:

Assume that there exists a stable match M.

$\Rightarrow$John will be matched with someone. (WLOG by symmetry let's say John matches with SRK)

$\Rightarrow$ Salman is matched with Aamir.

$\Rightarrow$ Salman & SRK form a "rogue couple".

$\Rightarrow$ M is NOT stable.

Theorem 2

If we only allow opposite gender relationships

i.e the representative graph is bipartite (with the set of girls & the set of boys being the two partitions) then a stable match necessarily exists.

What Next?

- Proof of Existence of a stable match.
- Algorithm to find the stable match.
- First we'll propose the algorithm & then claim that it leads to a stable match.

Example:

- Five Girls: A,B,C,D,**E**
- Five Boys: 1,2,3,4,5

Example:

- Five Girls: A,B,C,D,**E**
- Five Boys: 1,2,3,4,5
- Preference of Boys:

1: C , B , E , A , D

2: A , B , E , C , D

3: D , C , B , A , E

4: A , C , D , B , E

5: A , B , D , E , C

Preference of Girls:

A: 3 , 5 , 2 , 1 , 4

B: 5 , 2 , 1 , 4 , 3

C: 4 , 3 , 5 , 1 , 2

D: 1 , 2 , 3 , 4 , 5

E: 2 , 3 , 4 , 1 , 5

The Greedy Algorithm

- Start with Boy 1 and allocate him the “best” possible girl (i.e highest in his list)
- Now allocate to Boy 2 the “best available” girl.

The Greedy Algorithm

- Start with Boy 1 and allocate him the “best” possible girl (i.e highest in his list)
- Now allocate to Boy 2 the “best available” girl.

Preference List

1: C , B , E , A , D	A: 3 , 5 , 2 , 1 , 4
2: A , B , E , C , D	B: 5 , 2 , 1 , 4 , 3
3: D , C , B , A , E	C: 4 , 3 , 5 , 1 , 2
4: A , C , D , B , E	D: 1 , 2 , 3 , 4 , 5
5: A , B , D , E , C	E: 2 , 3 , 4 , 1 , 5

The Greedy Algorithm

- Start with Boy 1 and allocate him the “best” possible girl (i.e highest in his list)
- Now allocate to Boy 2 the “best available” girl.

…..

1 -> C

Preference List

1: C , B , E , A , D	A: 3 , 5 , 2 , 1 , 4
2: A , B , E , C , D	B: 5 , 2 , 1 , 4 , 3
3: D , C , B , A , E	C: 4 , 3 , 5 , 1 , 2
4: A , C , D , B , E	D: 1 , 2 , 3 , 4 , 5
5: A , B , D , E , C	E: 2 , 3 , 4 , 1 , 5

The Greedy Algorithm

- Start with Boy 1 and allocate him the “best” possible girl (i.e highest in his list)
- Now allocate to Boy 2 the “best available” girl.

…..

1 -> C

2 -> A

Preference List

1: C , B , E , A , D	A: 3 , 5 , 2 , 1 , 4
2: A , B , E , C , D	B: 5 , 2 , 1 , 4 , 3
3: D , C , B , A , E	C: 4 , 3 , 5 , 1 , 2
4: A , C , D , B , E	D: 1 , 2 , 3 , 4 , 5
5: A , B , D , E , C	E: 2 , 3 , 4 , 1 , 5

The Greedy Algorithm

- Start with Boy 1 and allocate him the “best” possible girl (i.e highest in his list)
- Now allocate to Boy 2 the “best available” girl.

…..

1 -> C

2 -> A

3 -> D

Preference List

1: C , B , E , A , D	A: 3 , 5 , 2 , 1 , 4
2: A , B , E , C , D	B: 5 , 2 , 1 , 4 , 3
3: D , C , B , A , E	C: 4 , 3 , 5 , 1 , 2
4: A , C , D , B , E	D: 1 , 2 , 3 , 4 , 5
5: A , B , D , E , C	E: 2 , 3 , 4 , 1 , 5

The Greedy Algorithm

- Start with Boy 1 and allocate him the “best” possible girl (i.e highest in his list)
- Now allocate to Boy 2 the “best available” girl.

…..

1 -> C

2 -> A

3 -> D

4 -> B

Preference List

1: C , B , E , A , D	A: 3 , 5 , 2 , 1 , 4
2: A , B , E , C , D	B: 5 , 2 , 1 , 4 , 3
3: D , C , B , A , E	C: 4 , 3 , 5 , 1 , 2
4: A , C , D , B , E	D: 1 , 2 , 3 , 4 , 5
5: A , B , D , E , C	E: 2 , 3 , 4 , 1 , 5

The Greedy Algorithm

- Start with Boy 1 and allocate him the “best” possible girl (i.e highest in his list)
- Now allocate to Boy 2 the “best available” girl.

…..

1 -> C

2 -> A

3 -> D

4 -> B

5 -> E

Preference List

1: C , B , E , A , D	A: 3 , 5 , 2 , 1 , 4
2: A , B , E , C , D	B: 5 , 2 , 1 , 4 , 3
3: D , C , B , A , E	C: 4 , 3 , 5 , 1 , 2
4: A , C , D , B , E	D: 1 , 2 , 3 , 4 , 5
5: A , B , D , E , C	E: 2 , 3 , 4 , 1 , 5

The Greedy Algorithm

- Start with Boy 1 and allocate him the “best” possible girl (i.e highest in his list)
- Now allocate to Boy 2 the “best available” girl.

…..

1 -> C

2 -> A

3 -> D

4 -> B

5 -> E

- Is this match stable?
-

Preference List

1: C , B , E , A , D	A: 3 , 5 , 2 , 1 , 4
2: A , B , E , C , D	B: 5 , 2 , 1 , 4 , 3
3: D , C , B , A , E	C: 4 , 3 , 5 , 1 , 2
4: A , C , D , B , E	D: 1 , 2 , 3 , 4 , 5
5: A , B , D , E , C	E: 2 , 3 , 4 , 1 , 5

The Greedy Algorithm

- Start with Boy 1 and allocate him the “best” possible girl (i.e highest in his list)
- Now allocate to Boy 2 the “best available” girl.

…..

1 -> C

2 -> A

3 -> D

4 -> B

5 -> E

- Is this match stable?
- (Boy 4 , Girl C) form a “rogue couple”.

Preference List

1: C , B , E , A , D	A: 3 , 5 , 2 , 1 , 4
2: A , B , E , C , D	B: 5 , 2 , 1 , 4 , 3
3: D , C , B , A , E	C: 4 , 3 , 5 , 1 , 2
4: A , C , D , B , E	D: 1 , 2 , 3 , 4 , 5
5: A , B , D , E , C	E: 2 , 3 , 4 , 1 , 5

The Stable Matching Algorithm

- Everyday a boy will go and stand in front of the balcony of the girl he likes the most.
- Each day a girl will either tell a boy waiting near her balcony to “come back next day” or “reject”.
- Once the girl has “rejected” a boy, the boy crosses the girl off his list of possibilities.
- This continues till all the girls have exactly one boy standing under her balcony.
- Initially all girls are present in every boy’s list.

DAY - 1

A -> 2 , 4 , 5

B ->

C -> 1

D -> 3

E ->

Preference List

1: C , B , E , A , D
2: A , B , E , C , D
3: D , C , B , A , E
4: A , C , D , B , E
5: A , B , D , E , C

A: 3 , 5 , 2 , 1 , 4
B: 5 , 2 , 1 , 4 , 3
C: 4 , 3 , 5 , 1 , 2
D: 1 , 2 , 3 , 4 , 5
E: 2 , 3 , 4 , 1 , 5

The Updated List

1: C , B , E , A , D

2: A , B , E , C , D

3: D , C , B , A , E

4: A , C , D , B , E

5: A , B , D , E , C

Preference List

1: C , B , E , A , D	A: 3 , 5 , 2 , 1 , 4
2: A , B , E , C , D	B: 5 , 2 , 1 , 4 , 3
3: D , C , B , A , E	C: 4 , 3 , 5 , 1 , 2
4: A , C , D , B , E	D: 1 , 2 , 3 , 4 , 5
5: A , B , D , E , C	E: 2 , 3 , 4 , 1 , 5

DAY - 2

A -> 5

B -> 2

C -> 1 , 4

D -> 3

E ->

Preference List

1: C , B , E , A , D
2: A , B , E , C , D
3: D , C , B , A , E
4: A , C , D , B , E
5: A , B , D , E , C

A: 3 , 5 , 2 , 1 , 4
B: 5 , 2 , 1 , 4 , 3
C: 4 , 3 , 5 , 1 , 2
D: 1 , 2 , 3 , 4 , 5
E: 2 , 3 , 4 , 1 , 5

The Updated List

1: C , B , E , A , D

2: A , B , E , C , D

3: D , C , B , A , E

4: A , C , D , B , E

5: A , B , D , E , C

Preference List

1: C , B , E , A , D
2: A , B , E , C , D
3: D , C , B , A , E
4: A , C , D , B , E
5: A , B , D , E , C

A: 3 , 5 , 2 , 1 , 4
B: 5 , 2 , 1 , 4 , 3
C: 4 , 3 , 5 , 1 , 2
D: 1 , 2 , 3 , 4 , 5
E: 2 , 3 , 4 , 1 , 5

DAY - 3

A -> 5

B -> 1 , 2

C -> 4

D -> 3

E ->

Preference List

1: C , B , E , A , D
2: A , B , E , C , D
3: D , C , B , A , E
4: A , C , D , B , E
5: A , B , D , E , C

A: 3 , 5 , 2 , 1 , 4
B: 5 , 2 , 1 , 4 , 3
C: 4 , 3 , 5 , 1 , 2
D: 1 , 2 , 3 , 4 , 5
E: 2 , 3 , 4 , 1 , 5

The Updated List

1: C , B , E , A , D

2: A , B , E , C , D

3: D , C , B , A , E

4: A , C , D , B , E

5: A , B , D , E , C

Preference List

1: C , B , E , A , D
2: A , B , E , C , D
3: D , C , B , A , E
4: A , C , D , B , E
5: A , B , D , E , C

A: 3 , 5 , 2 , 1 , 4
B: 5 , 2 , 1 , 4 , 3
C: 4 , 3 , 5 , 1 , 2
D: 1 , 2 , 3 , 4 , 5
E: 2 , 3 , 4 , 1 , 5

DAY - 4

A -> 5

B -> 2

C -> 4

D -> 3

E -> 1

Preference List

1: C , B , E , A , D	A: 3 , 5 , 2 , 1 , 4
2: A , B , E , C , D	B: 5 , 2 , 1 , 4 , 3
3: D , C , B , A , E	C: 4 , 3 , 5 , 1 , 2
4: A , C , D , B , E	D: 1 , 2 , 3 , 4 , 5
5: A , B , D , E , C	E: 2 , 3 , 4 , 1 , 5

- This is where the algorithm terminates.
- Is the match stable?

Stability Test

Preference of Boys:	Preference of Girls:	Matching
1: C , B , E , A , D	A: 3 , 5 , 2 , 1 , 4	A -> 5
2: A , B , E , C , D	B: 5 , 2 , 1 , 4 , 3	B -> 2
3: D , C , B , A , E	C: 4 , 3 , 5 , 1 , 2	C -> 4
4: A , C , D , B , E	D: 1 , 2 , 3 , 4 , 5	D -> 3
5: A , B , D , E , C	E: 2 , 3 , 4 , 1 , 5	E -> 1

Desirable Properties of SMA!!

i. SMA terminates.
ii. Terminates quickly.
iii. Everybody gets married / paired.
iv. No rogue couple (stability).

Theorem 3

SMA terminates is $\leq N^2$ + 1 days.

- *Proof:*

Every day (i.e in every iteration of the algorithm) at least one boy crosses out one girl from his list.

$\Rightarrow$ Every day is characterized by at least ONE cross out.

Since there are N girls & N boys (thereby N lists), so there are at max N^2 cross outs.

=> SMA NOT terminating in N^2 + 1 days is an impossibility.

Theorem 4

Everybody gets married / paired.

Proof:

Assume that there is a boy B who is NOT married at the end.

$\Rightarrow$B has been rejected by every girl.

$\Rightarrow$ Every girl is married.

$\Rightarrow$ Every boy is married (since we have equal number of boys & girls)

$\Rightarrow$ B is married (contradiction)

Theorem 5

SMA produces Stable matching.

Proof: Let’s say there exists a rogue couple (Johnny , Amber).

CASE –I: Amber rejected Johnny
⇒Amber must have had a more preferred suitor serenading her.
⇒ Amber prefers her husband over Johnny

CASE – II: Johnny never serenaded Amber
⇒Johnny never had to come down to Amber in his Pref. List.
⇒ Johnny prefers his wife over Amber