

INF 431

PC – 1

Stable marriages...

Based on the PC-notes by B. Salvy & G. Schaeffer

M. Vazirgiannis

mvazirg@lix.polytechnique.fr

Gale & Shepley algorithm

function stableMatching

Initialize all $m \in M$ and $w \in W$ to *free*

while \exists *free* man m who still has a woman w to propose to ($w = m$'s highest ranked such woman to whom he has not yet proposed)

{

if w is *free*

(m, w) become *engaged*

else some pair (m', w) already exists

if w prefers m to m'

(m, w) become *engaged*

m' becomes *free*

else

(m', w) remain *engaged*

}

}

See interesting demo for stable marriages at:

- <http://www.dcs.gla.ac.uk/research/algorithms/stable/EGSapplet/EGS.html>

Gale & Shepley algorithm

- The algorithm *terminates* after $\max n^2$ iterations ...
- The matching is
 - ***perfect***
 - every m gets engaged to a w and vice versa
 - ***Stable***
 - *proof by contradiction: assume (m, w) and m prefers w' while w prefers m' – m has proposed to w' before and therefore they must have been engaged.*

Stable marriages -Q1

- Assume n candidates for n departments.
- Each candidate has a ranking of the departments
- Each department has a ranking of the candidates
- A stable matching can be found by applying the Gale-Shapley algorithm.
- Assume that each department has k positions and for each of them there is a preference list of n_i candidates.
- (sum of positions = # candidates)
- How can we treat this situation to match the n candidates to the k positions?

Stable marriages - Q1

- Assume departments D1, D2 and students $\{s_1, s_2, s_3, s_4, s_5\}$.
- The students preferences are: $s_1: \{D1, D2\}$, $s_2: \{D1, D2\}$, $s_3: \{D1, D2\}$, $s_4: \{D1, D2\}$, $s_5: \{D1, D2\}$
- Depts advertise positions D1(p_1, p_2) and D2(p_1, p_2, p_3)
- The Departments preference lists for the positions are: D1: $\{s_3, s_1, s_2, s_4, s_5\}$, D2: $\{s_1, s_4, s_2, s_3, s_5\}$
- *How can we modify the matching algorithm to achieve a stable matching here?*

Stable marriages -1 - solution

- Make the two parts “even” i.e.: number of positions available = number of candidates
- For each position assign a copy of the list of students that the department had:
 - D1. $p_1 : \{s_3, s_1, s_2, s_4, s_5\}$, D1. $p_2 : \{s_3, s_1, s_2, s_4, s_5\}$
 - D2. $p_1 : \{s_1, s_4, s_2, s_3, s_5\}$, D2. $p_2 : \{s_1, s_4, s_2, s_3, s_5\}$, D2. $p_3 : \{s_1, s_4, s_2, s_3, s_5\}$
- For each student distribute her preference to the positions of the departments according to the original preferences:
 - $s_1 : \{D1. p_1, D1. p_2, D2.p_1, D2.p_2, D2.p_3\}$,
 - $s_2 : \{D1. p_2, D1. p_1, D2.p_1, D2.p_3, D2.p_2\}$,
 - ...
- Apply the stable marriage algorithm

Stable marriages -Q2

- Assume that each department has k positions and for each of them there is a preference list of n_i candidates.
- (sum of positions = # candidates)
- How can we treat this situation to match the n candidates to the k positions?
- Simply to assign to each position the same list

Stable marriages –Q2

- Assume
 - departments that has positions available
 - A population of candidates among which some are from school Y .
 - The departments systematically favor (put in the top of their preferences) Y candidates
 - The candidates Y rank the director positions above the other ones
- Let k director positions and k candidates of type Y . Prove that all k director positions will be occupied by the k Y candidates.

Stable Marriage – Q2

- Let k director positions and k candidates of type Y . Prove that all k director positions will be occupied by the k Y candidates.
- Assume a matching $(d, A), (n-d, Y1)$, d : director, $n-d$: non director position, A non Y candidate.
 - d prefers $Y1$ to A
 - $Y1$ prefers d to $n-d$
- Thus there is instability! Hence all d positions will be occupied by Y candidates

Stable Marriage –

Incomplete lists of preferences - Q3

- Assume the case where a man m / woman w prefer to stay single than get engaged to a partner they do not wish (incomplete preference lists).
- For instance: let $M = \{m_1, m_2, m_3, m_4, m_5\}$, $S = \{s_1, s_2, s_3, s_4, s_5\}$ and the incomplete preference lists:
- $m_1 : \{s_3, s_1, s_2\}$, $m_2 : \{s_4, s_1, s_2, s_3\}$...
- One way to represent this follows:
- we can assume that m participates in **his** list just after the last acceptable partner, i.e. $m_1 : \{s_3, s_1, s_2, m_1, s_4, s_5\}$. This implies that m_1 does not accept s_4, s_5 as fiancés and would rather remain single.

Stable Marriage –

Incomplete lists of preferences - Q3

- Prove that in a matching with incomplete lists no one m can be married to someone w that is lower than him self m in her preference list.

Proof:

- m has proposes only to w 's that are desired and thus higher than herself in the preference list.

Stable Matching – Q 4

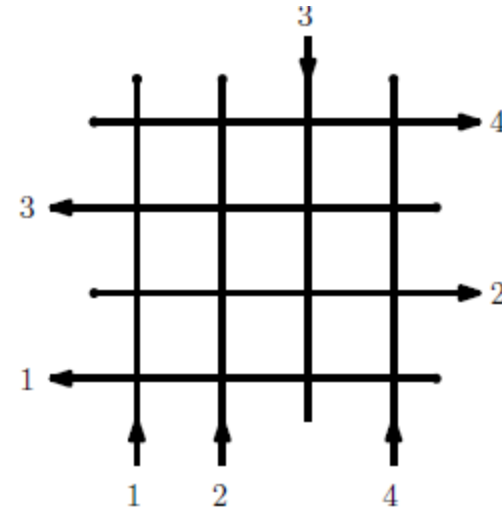
- **Verify that you can extend the algorithm of Gale and Shepley for incomplete lists, and that the evidence of termination and stability of the matching is obtained immediately.**
 - The algorithm is the same:
 - men propose to women not engaged (*or decide to remain single*) in the order of their preference list.
 - Women accept at any time the proposal most advantageous to them made (taking into account the possibility for them to remain single). Here is the pseudo-code:
- ```
while there is a free man and he has not requested all women that he appreciates in marriage {
 let h is such a man
 let f is the favorite wife of h among those he likes and he has not yet proposed to her
 if f is free and appreciates h
 then h and f get engaged
 else
 if f is engaged to h' but prefers h
 then
 f leaves h' (which becomes free) and becomes engaged to h
}
```
- The *Termination* is assured from the fact that the same proposal **h** to **f** is not made twice.
  - The result is a matching (not a perfect one..).
  - The proof of stability is identical to that of the course.

## Reference

- Roth, A.E. and Vande Vate, J.H., "Random Paths to Stability in Two-Sided Matching," *Econometrica*, 58, 1990, 1475-1480.

# Stability and perfect routing – Q5

- Consider a network of  $n$  *incoming* pipes and  $n$  *outgoing* ones:
- the in pipe  $i$  successively crosses by each of the  $n$  out pipes following a permutation  $\sigma(i)$ ;
- the out pipe  $j$  is crossed by each of the  $n$  input pipes, based on a permutation  $\tau(j)$  representing the order in which these pipes are meeting from the output.
- Each intersection consists of a valve which can be either closed or open in one of *six* positions connecting two pipes of 4 incidents.



$$\sigma_1 = 1, 2, 3, 4$$

$$\sigma_2 = 1, 2, 3, 4$$

$$\sigma_3 = 4, 3, 2, 1$$

$$\sigma_4 = 1, 2, 3, 4$$

$$\tau_1 = 1, 2, 3, 4$$

$$\tau_2 = 4, 3, 2, 1$$

$$\tau_3 = 1, 2, 3, 4$$

$$\tau_4 = 4, 3, 2, 1$$

# Stability and perfect routing

- Show that there is a way to open valves to allow the simultaneous flow of  $n$  independent streams in the network, from the  $n$  in pipes to the  $n$  out pipes.

## Solution.

- Junctions: Stable marriage among in/out pipes based on the permutation  $\sigma(i)$  and  $\tau(j)$  as lists as preference
- Then we open the valves a way that if  $i$  and  $j$  are married and the flow from  $i$  continues in  $j$  en exits from it
- \*we have to make sure that the flows do not meet!

$$\sigma_1 = 1, 2, 3, 4$$

$$\sigma_2 = 1, 2, 3, 4$$

$$\sigma_3 = 4, 3, 2, 1$$

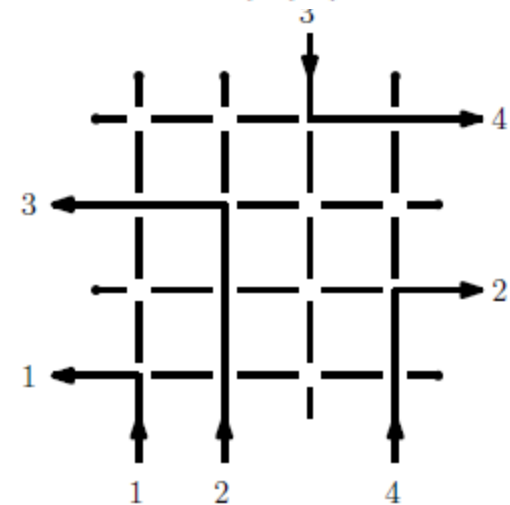
$$\sigma_4 = 1, 2, 3, 4$$

$$\tau_1 = 1, 2, 3, 4$$

$$\tau_2 = 4, 3, 2, 1$$

$$\tau_3 = 1, 2, 3, 4$$

$$\tau_4 = 4, 3, 2, 1$$



# Stable Marriages - Termination

- Consider a group of  $n$  persons. Some of them are friends while some others are enemies.
- The goal is to divide the group into two partitions so that everyone “feels good” about her team, in the sense that a individual  $x$  is satisfied:
  - Satisfied( $x$ ): #friends of  $x$  in  $x$ 's group + # enemies of  $x$  in the *other group* (1)
  - Dis-satisfied( $x$ ): #enemies of  $x$  in  $x$ 's group + # friends of  $x$  in the *other group*
- The algorithm we want to study works iteratively based on an arbitrary partition of the set: As long as there are unhappy people, we chose one and change her team.

*Interesting animation at:*

<http://www.physorg.com/news/2011-01-mathematical-groups-factions.html>

# Stable Marriages – Termination –Q7

- Question 7: Show that the algorithm terminates, producing a stable group.

*Solution.*

- The quantity to consider  $\# \{pairs\ of\ friends\ in\ the\ same\ team\} + \# \{enemies\ in\ different\ teams\}$  is equivalent to the
  - $sum\ of\ the\ satisfaction(x)$  for all individuals.
- this quantity is bounded by  $n^2$ ,  $n$  is the number of people in the group.
- when a person  $x$  changes team above the above amount increases strictly as the left-hand side of condition (1)
- Thus that the algorithm stops in at most  $n^2$  steps and by definition the score obtained is stable.

# Stable Marriages – Termination –Q6

- Q6: Give an example of a situation in which a step of the algorithm increases the number of *unhappy* people.
- *Solution.*
- Take a person with two enemies and three friends, all in the other team,
- then the person wants to change teams. Suppose that the enemies that has 3 friends each and are already satisfied.
- The team change has made unhappy the two enemies: # enemies in the group increases by 1.

# Stable Marriages – Termination –Q8

- Question 8: Can we really use this algorithm to form teams?

*Solution.*

- *No* because there is no guarantee of obtaining balanced size groups
  - for example, if all people are friends, the only stable partitioning is the trivial partition with a group containing all the people.
- This algorithm was actually proposed to model the self-stabilization of a simplified network neural network called Hopfield
  - each individual represents a neuron that can take two states,
  - the connections are of two types
  - supporting the agreement or disagreement between neighboring neurons.

# The k-sum problem.

*Problem definition:*

Assume a set  $\mathbf{E}$  of  $\mathbf{N}$  integers, and another integer  $S$ . Is there a subset of  $\mathbf{E}$  whose element sum to  $S$ ?

- In computer science, the subset sum problem is an important problem in complexity theory and cryptography.
- The problem is NP-complete with regards to its time complexity
- A brute force algorithm (exponential in  $N$ )

Compute all subsets of  $\mathbf{E}$  of size  $\mathbf{N}$   
for each of them  
check if the subset sums to  $S$ .

Time complexity  $O(2^N N)$ ,

- there are  $2^N$  subsets and, to check each subset, we need to sum at most  $N$  elements.

# 2-sum – naïve approach – Q9

Naiv2sum(E, S, N)

For i=1 to N

  for j = i+1 to N

    if  $E[i]+E[j] = S$  return *true*

  return *false*

*Quadratic approach  $O(n^2)$*

*K-sum:  $O(n^k)$*

# 2-sum – for sorted lists – Q10

```
sorted2sum(E, S, N)
```

```
i=1; j=N
```

```
 while for i<j
```

```
 if E[i]+E[j] = S return true
```

```
 else if E[i]+E[j] >S
```

```
 then j=j-1
```

```
 else i=i+1
```

```
 return false
```

*Complexity linear (O(N))*