

# Emparelhamento estável

Formulação do problema, algoritmos e implementação

**Prof. Marcelo de Souza**

`marcelo.desouza@udesc.br`



Algoritmos e Estruturas de Dados  
Bacharelado em Engenharia de Software  
Universidade do Estado de Santa Catarina



## **Leitura obrigatória:**

- Capítulo 1 de [Kleinberg and Tardos \(2006\)](#) – Introdução.

## **Leitura complementar:**

- Capítulo 2 de [Kleinberg and Tardos \(2006\)](#) – Grafos.

# Problema do emparelhamento estável



- Formalizado em 1962 por David Gale e Lloyd Shapley.
- Algoritmo ganhou o Prêmio Nobel em Economia de 2012.
- **Cenário:** candidatos a estágio  $\times$  empresas de tecnologia.
  - Candidatos possuem uma lista de preferência das empresas.
  - Empresas possuem uma lista de preferência dos candidatos.

# Problema do emparelhamento estável



- Formalizado em 1962 por David Gale e Lloyd Shapley.
- Algoritmo ganhou o Prêmio Nobel em Economia de 2012.
- **Cenário:** candidatos a estágio  $\times$  empresas de tecnologia.
  - Candidatos possuem uma lista de preferência das empresas.
  - Empresas possuem uma lista de preferência dos candidatos.

Candidato	1º	2º	3º
<b>Xavier</b>	Boston	Atlanta	Chicago
<b>Yolanda</b>	Atlanta	Boston	Chicago
<b>Zeus</b>	Atlanta	Boston	Chicago

Preferências dos candidatos

Empresa	1º	2º	3º
<b>Atlanta</b>	Xavier	Yolanda	Zeus
<b>Boston</b>	Yolanda	Xavier	Zeus
<b>Chicago</b>	Xavier	Yolanda	Zeus

Preferências das empresas

# Problema do emparelhamento estável



- Formalizado em 1962 por David Gale e Lloyd Shapley.
- Algoritmo ganhou o Prêmio Nobel em Economia de 2012.
- **Cenário:** candidatos a estágio  $\times$  empresas de tecnologia.
  - Candidatos possuem uma lista de preferência das empresas.
  - Empresas possuem uma lista de preferência dos candidatos.

Candidato	1º	2º	3º
<b>Xavier</b>	Boston	Atlanta	Chicago
<b>Yolanda</b>	Atlanta	Boston	Chicago
<b>Zeus</b>	Atlanta	Boston	Chicago

Preferências dos candidatos

Empresa	1º	2º	3º
<b>Atlanta</b>	Xavier	Yolanda	Zeus
<b>Boston</b>	Yolanda	Xavier	Zeus
<b>Chicago</b>	Xavier	Yolanda	Zeus

Preferências das empresas

- **Problema:** Xavier aceita proposta de Atlanta, mas recusa após receber nova proposta de Boston. Boston contrata Xavier, mas o demite após receber o currículo de Yolanda. Xavier se obriga a ir para Chicago.



- Dado um conjunto de candidatos e um conjunto de empresas, cada um com sua lista de preferências, encontra um **emparelhamento** de candidatos e empresas que seja **estável**.
- Um emparelhamento é estável com ao menos uma dessas condições:
  1. Candidato prefere sua empresa às outras que o aceitariam;
  2. Empresa prefere seu funcionário aos outros que a aceitariam.

# Problema do emparelhamento estável



- Dado um conjunto de candidatos e um conjunto de empresas, cada um com sua lista de preferências, encontra um **emparelhamento** de candidatos e empresas que seja **estável**.
- Um emparelhamento é estável com ao menos uma dessas condições:
  1. Candidato prefere sua empresa às outras que o aceitariam;
  2. Empresa prefere seu funcionário aos outros que a aceitariam.

Candidato	1º	2º	3º
Xavier	Boston	Atlanta	Chicago
Yolanda	Atlanta	Boston	Chicago
Zeus	Atlanta	Boston	Chicago

Um emparelhamento **não estável**

Empresa	1º	2º	3º
Atlanta	Xavier	Yolanda	Zeus
Boston	Yolanda	Xavier	Zeus
Chicago	Xavier	Yolanda	Zeus

Um emparelhamento **não estável**

- Atlanta prefere Xavier a Zeus e Xavier prefere Atlanta a Chicago.

# Problema do emparelhamento estável



- Dado um conjunto de candidatos e um conjunto de empresas, cada um com sua lista de preferências, encontra um **emparelhamento** de candidatos e empresas que seja **estável**.
- Um emparelhamento é estável com ao menos uma dessas condições:
  1. Candidato prefere sua empresa às outras que o aceitariam;
  2. Empresa prefere seu funcionário aos outros que a aceitariam.

Candidato	1º	2º	3º
Xavier	Boston	Atlanta	Chicago
Yolanda	Atlanta	Boston	Chicago
Zeus	Atlanta	Boston	Chicago

Um emparelhamento **estável**

Empresa	1º	2º	3º
Atlanta	Xavier	Yolanda	Zeus
Boston	Yolanda	Xavier	Zeus
Chicago	Xavier	Yolanda	Zeus

Um emparelhamento **estável**

- Nenhum indivíduo consegue “melhorar” seu emparelhamento.





- Problema em outros contextos:
  - Estudantes de residência em hospitais;
  - Formação de equipes;
  - Alocação de servidores a aplicações;
  - **Casamento de pares.**
- Problema do *casamento estável*:
  - Conjunto  $M = \{m_1, \dots, m_n\}$  de homens.
  - Conjunto  $W = \{w_1, \dots, w_n\}$  de mulheres.
  - **Emparelhamento:** conjunto de pares ordenados  $(m, w)$ , onde  $m \in M$  e  $w \in W$  e cada indivíduo está em no máximo um par.
  - **Emparelhamento perfeito:** cada indivíduo está em um par.
  - Cada homem possui uma lista de preferências de mulheres e vice-versa.
  - **Problema:** encontra um emparelhamento perfeito estável.



- **Instabilidade:** caso que resulta em um emparelhamento não estável.
  - Dois pares  $(m, w)$  e  $(m', w')$ , em que:
    - ▷  $m$  prefere  $w'$  a  $w$  e  $w'$  prefere  $m$  a  $m'$ ; ou
    - ▷  $w$  prefere  $m'$  a  $m$  e  $m'$  prefere  $w$  a  $w'$ .
  - Em ambos os casos, indivíduos têm incentivo de desfazer seu casamento.
- Um emparelhamento estável é um emparelhamento perfeito que não possui instabilidade.



- **Instabilidade:** caso que resulta em um emparelhamento não estável.
  - Dois pares  $(m, w)$  e  $(m', w')$ , em que:
    - ▷  $m$  prefere  $w'$  a  $w$  e  $w'$  prefere  $m$  a  $m'$ ; ou
    - ▷  $w$  prefere  $m'$  a  $m$  e  $m'$  prefere  $w$  a  $w'$ .
  - Em ambos os casos, indivíduos têm incentivo de desfazer seu casamento.
- Um emparelhamento estável é um emparelhamento perfeito que não possui instabilidade.
- **Exemplo:**
  - $m$  prefere  $w$  a  $w'$ .
  - $m'$  prefere  $w'$  a  $w$ .
  - $w$  prefere  $m'$  a  $m$ .
  - $w'$  prefere  $m$  a  $m'$ .
  - **Emparelhamento estável 1:**  $(m, w)$  e  $(m', w')$  – homens felizes.
  - **Emparelhamento estável 2:**  $(m, w')$  e  $(m', w)$  – mulheres felizes.

---

**Algorithm:** Gale-Shapley

---

Inicialmente todos  $m \in M$  e  $w \in W$  estão livres

**while** *existe um homem  $m$  livre que ainda não propôs a toda mulher*  
**do**

    Escolha um homem  $m$

$w \leftarrow$  mulher preferida de  $m$  para a qual ele ainda não propôs

**if**  $w$  *está livre* **then**

$(m, w)$  formam um par

**else**

**if**  $w$  *prefere  $m'$  a  $m$*  **then**

$m$  continua livre

**else**

$(m, w)$  formam um par

$m'$  se torna livre

Retorna o conjunto  $S$  de pares

# Algoritmo de Gale-Shapley



---

## Algorithm: Gale-Shapley

---

Inicialmente todos  $m \in M$  e  $w \in W$  estão livres

**while** *existe um homem  $m$  livre que ainda não propôs a toda mulher*  
**do**

Escolha um homem  $m$

$w \leftarrow$  mulher preferida de  $m$  para a qual ele ainda não propôs

**if**  $w$  *está livre* **then**

$(m, w)$  formam um par

**else**

**if**  $w$  *prefere  $m'$  a  $m$*  **then**

$m$  continua livre

**else**

$(m, w)$  formam um par  
         $m'$  se torna livre

Iterativamente, um homem livre propõe a uma nova mulher, seguindo sua lista de preferências. Se a mulher está livre ou prefere este homem, eles formam um casal. Caso algum homem seja abandonado pela mulher, ele volta à lista de homens livres. Quando todos os homens estiverem casados, todas as mulheres também estarão.

Retorna o conjunto  $S$  de pares

# Algoritmo de Gale-Shapley



## Detalhes de funcionamento

- O algoritmo GS retorna um emparelhamento perfeito.
  - **Observação 1:** homens propõem na ordem de sua preferência.
  - **Observação 2:** uma vez em um par, uma mulher nunca fica livre.
  - **Observação 3:** homens propõem somente quando estão livres.
  - **Observação 4:** mulheres mantêm apenas o homem de maior preferência.
- **Prova** (por contradição):
  - Supondo que um homem  $m \in M$  esteja livre ao final da execução do algoritmo. Então, uma mulher  $w \in W$  também estará livre.
  - Pela *OBS2*, ninguém propôs a  $w$ . Mas  $m$  propôs a todas as mulheres, para que o algoritmo (laço) tenha terminado [uma contradição].
  - Todos os homens terminam em pares. O mesmo ocorre para as mulheres.
  - O algoritmo não pode terminar, senão com um emparelhamento perfeito.

# Algoritmo de Gale-Shapley



## Detalhes de funcionamento

- O algoritmo GS retorna um emparelhamento estável.
  - Ou seja, o emparelhamento  $S$  não possui pares instáveis.
- Prova:
  - Consideremos qualquer par  $(m, w)$  que não está em  $S$ .
  - **Caso 1:**  $m$  nunca propôs a  $w$ .
    - ▷  $m$  prefere seu par  $w'$  a  $w$ .
    - ▷  $(m, w)$  não é instável.
  - **Caso 2:**  $m$  propôs a  $w$ .
    - ▷  $w$  rejeitou  $m$  (no momento ou mais tarde).
    - ▷  $w$  prefere seu par  $m'$  a  $m$ .
    - ▷  $(m, w)$  não é instável.
  - Logo,  $S$  é um emparelhamento estável.

# Algoritmo de Gale-Shapley

Complexidade de tempo



- O algoritmo GS termina em no máximo  $n^2$  iterações do laço.
- **Prova:**
  - A cada iteração, um homem  $m$  propõe a uma mulher  $w$ .
  - No pior caso, todas as propostas possíveis serão feitas.
  - Ou seja, todos os possíveis pares  $(m, w)$  serão “testados”.
  - Existe no máximo  $n^2$  pares possíveis.
  - Logo, o laço é executado no máximo  $n^2$  vezes.



# Algoritmo de Gale-Shapley



Complexidade de tempo

- O algoritmo GS possui complexidade quadrática –  $O(n^2)$ .
- **Prova:**
  - Sabemos que o laço executa  $n^2$  vezes no pior caso.
  - Para que a complexidade seja  $O(n^2)$ , todas as operações internas ao laço devem ser executadas em tempo constante.
    - ▷ Identificar um homem livre.
    - ▷ Identificar a mulher preferida para quem um homem  $m$  ainda não propôs.
    - ▷ Verificar se uma mulher  $w$  possui um par e recuperá-lo.
    - ▷ Verificar qual dos homens  $m$  e  $m'$  é preferido por uma mulher  $w$ .
  - Usando listas e vetores é possível obter o desempenho desejado.

# Algoritmo de Gale-Shapley



## Complexidade de tempo – detalhes de implementação

- Cada homem e cada mulher recebe um índice  $i$ .
- As preferências são dadas por matrizes (ManPref e WomanPref).
  - $\text{ManPref}[m, i]$  denota a  $i$ -ésima mulher na lista de preferência de  $m$ .
  - $\text{WomanPref}[w, i]$  denota o  $i$ -ésimo homem na lista de preferência de  $w$ .

# Algoritmo de Gale-Shapley



## Complexidade de tempo – detalhes de implementação

- Cada homem e cada mulher recebe um índice  $i$ .
- As preferências são dadas por matrizes (ManPref e WomanPref).
  - $\text{ManPref}[m, i]$  denota a  $i$ -ésima mulher na lista de preferência de  $m$ .
  - $\text{WomanPref}[w, i]$  denota o  $i$ -ésimo homem na lista de preferência de  $w$ .
- O conjunto de homens livres é mantido por uma lista encadeada.
  - Inserção e remoção são feitas em tempo constante.

# Algoritmo de Gale-Shapley



## Complexidade de tempo – detalhes de implementação

- Cada homem e cada mulher recebe um índice  $i$ .
- As preferências são dadas por matrizes (ManPref e WomanPref).
  - ManPref[ $m, i$ ] denota a  $i$ -ésima mulher na lista de preferência de  $m$ .
  - WomanPref[ $w, i$ ] denota o  $i$ -ésimo homem na lista de preferência de  $w$ .
- O conjunto de homens livres é mantido por uma lista encadeada.
  - Inserção e remoção são feitas em tempo constante.
- Um vetor Next armazena a próxima mulher para um homem propor.
  - O vetor é inicializado Next[ $m$ ] = 1, para todo homem  $m$ .
  - Um homem vai propor à mulher  $w = \text{ManPref}[m, \text{Next}[m]]$ .
  - Após propor, Next[ $m$ ] é incrementado em 1.

# Algoritmo de Gale-Shapley



## Complexidade de tempo – detalhes de implementação

- Cada homem e cada mulher recebe um índice  $i$ .
- As preferências são dadas por matrizes (ManPref e WomanPref).
  - $\text{ManPref}[m, i]$  denota a  $i$ -ésima mulher na lista de preferência de  $m$ .
  - $\text{WomanPref}[w, i]$  denota o  $i$ -ésimo homem na lista de preferência de  $w$ .
- O conjunto de homens livres é mantido por uma lista encadeada.
  - Inserção e remoção são feitas em tempo constante.
- Um vetor Next armazena a próxima mulher para um homem propor.
  - O vetor é inicializado  $\text{Next}[m] = 1$ , para todo homem  $m$ .
  - Um homem vai propor à mulher  $w = \text{ManPref}[m, \text{Next}[m]]$ .
  - Após propor,  $\text{Next}[m]$  é incrementado em 1.
- Um vetor Current armazena o atual par de uma mulher.
  - $\text{Current}[w]$  é o atual par da mulher  $w$ .

# Algoritmo de Gale-Shapley



## Complexidade de tempo – detalhes de implementação

- Cada homem e cada mulher recebe um índice  $i$ .
- As preferências são dadas por matrizes (ManPref e WomanPref).
  - $\text{ManPref}[m, i]$  denota a  $i$ -ésima mulher na lista de preferência de  $m$ .
  - $\text{WomanPref}[w, i]$  denota o  $i$ -ésimo homem na lista de preferência de  $w$ .
- O conjunto de homens livres é mantido por uma lista encadeada.
  - Inserção e remoção são feitas em tempo constante.
- Um vetor Next armazena a próxima mulher para um homem propor.
  - O vetor é inicializado  $\text{Next}[m] = 1$ , para todo homem  $m$ .
  - Um homem vai propor à mulher  $w = \text{ManPref}[m, \text{Next}[m]]$ .
  - Após propor,  $\text{Next}[m]$  é incrementado em 1.
- Um vetor Current armazena o atual par de uma mulher.
  - $\text{Current}[w]$  é o atual par da mulher  $w$ .
- Uma matriz Ranking mapeia a lista de preferências de uma mulher.
  - $\text{Ranking}[w, m]$  contém a posição de  $m$  nas preferências de  $w$ .
  - Verificar se  $w$  prefere  $m$  a  $m'$ :  $\text{Ranking}[w, m] < \text{Ranking}[w, m']$ ?

# Algoritmo de Gale-Shapley



## Complexidade de tempo – detalhes de implementação

Usando essas estruturas de dados todas as operações são realizadas em tempo constante e o algoritmo possui complexidade  $O(n^2)$

Operação 1

Operação 2

Operação 3

Operação 4

- O conjunto de homens livres é mantido por uma lista encadeada.
  - Inserção e remoção são feitas em tempo constante.
- Um vetor `Next` armazena a próxima mulher para um homem propor.
  - O vetor é inicializado `Next[m] = 1`, para todo homem  $m$ .
  - Um homem vai propor à mulher  $w = \text{ManPref}[m, \text{Next}[m]]$ .
  - Após propor, `Next[m]` é incrementado em 1.
- Um vetor `Current` armazena o atual par de uma mulher.
  - `Current[w]` é o atual par da mulher  $w$ .
- Uma matriz `Ranking` mapeia a lista de preferências de uma mulher.
  - `Ranking[w, m]` contém a posição de  $m$  nas preferências de  $w$ .
  - Verificar se  $w$  prefere  $m$  a  $m'$ : `Ranking[w, m] < Ranking[w, m']`?

# Algoritmo de Gale-Shapley



## Otimalidade do emparelhamento

- **Cenário:**

- $m$  prefere  $w$  a  $w'$ .
- $m'$  prefere  $w'$  a  $w$ .
- $w$  prefere  $m'$  a  $m$ .
- $w'$  prefere  $m$  a  $m'$ .



# Algoritmo de Gale-Shapley



## Otimalidade do emparelhamento

- **Cenário:**

- $m$  prefere  $w$  a  $w'$ .
- $m'$  prefere  $w'$  a  $w$ .
- $w$  prefere  $m'$  a  $m$ .
- $w'$  prefere  $m$  a  $m'$ .

- **Emparelhamentos estáveis:**

- $(m, w)$  e  $(m', w')$ .
- $(m, w')$  e  $(m', w)$ .

# Algoritmo de Gale-Shapley



## Otimidade do emparelhamento

- **Cenário:**

- $m$  prefere  $w$  a  $w'$ .
- $m'$  prefere  $w'$  a  $w$ .
- $w$  prefere  $m'$  a  $m$ .
- $w'$  prefere  $m$  a  $m'$ .

- **Emparelhamentos estáveis:**

- $(m, w)$  e  $(m', w')$ .
- $(m, w')$  e  $(m', w)$ .

- **Qual emparelhamento o algoritmo GS retorna?**

- $(m, w)$  e  $(m', w')$ .
- Com homens propondo, o emparelhamento obtido é o melhor para eles.
- Cada homem termina com o melhor par possível.
- Ao inverter, o emparelhamento é o melhor para as mulheres.

# Algoritmo de Gale-Shapley



## Otimidade do emparelhamento

- **Cenário:**

- $m$  prefere  $w$  a  $w'$ .
- $m'$  prefere  $w'$  a  $w$ .
- $w$  prefere  $m'$  a  $m$ .
- $w'$  prefere  $m$  a  $m'$ .

- **Emparelhamentos estáveis:**

- $(m, w)$  e  $(m', w')$ .
- $(m, w')$  e  $(m', w)$ .

- **Qual emparelhamento o algoritmo GS retorna?**

- $(m, w)$  e  $(m', w')$ .
- Com homens propondo, o emparelhamento obtido é o melhor para eles.
- Cada homem termina com o melhor par possível.
- Ao inverter, o emparelhamento é o melhor para as mulheres.

- **Toda execução do algoritmo GS retorna o mesmo emparelhamento, que é o emparelhamento ótimo para o conjunto de indivíduos que propõem.**

# Exercício

## Algoritmo de Gale-Shapley



1. Implemente o algoritmo de Gale-Shapley para resolver o problema do emparelhamento estável. Implemente usando as estruturas de dados sugeridas, a fim de garantir o melhor desempenho. Teste a sua implementação em um conjunto de instâncias criadas aleatoriamente, iniciando por cada um dos conjuntos do problema.



Kleinberg, J. and Tardos, É. (2006). *Algorithm Design*. Pearson Education India.