Revista **Ciências Exatas**
V.30, N°1, 2024 | Taubaté/SP - Brasil | **ISSN:** 1516-2893 | **DOI:** https://doi.org/10.69609/1516-2893.2024.v30.n1.a3842

JOGOS DE SALÃO E O CASAMENTO ESTÁVEL

Jeferson Lesbão de Siqueira

Nei Yoshihiro Soma

RESUMO

O problema do casamento estável, também conhecido como SM (Stable Marriage) consiste em resolver o problema do casamento entre um número par de pessoas, formando casais, de forma que ninguém possa unilateralmente dissolver seu casamento cuja aplicabilidade inclui a atribuição de médicos recém-formados a hospitais, estudantes a escolas e órgãos humanos disponíveis para transplante a receptores. Este artigo visa apresentar uma introdução aos jogos de salão de Von Neumann, com enforque no problema do casamento estável. O texto faz uma introdução ao histórico do problema e sua relevância, a apresenta a descrição e um caso de exemplo do principal algoritmo para resolução, junto de um algoritmo para verificação de estabilidade.

Palavras chave: Algoritmo, Estrutura de Dados, Jogos de Salão, Casamento Estável, Gale-Shapley

ABSTRACT

The problem known as SM (Stable Marriage) consists of solving the problem of marriage between an even number of people, forming couples, so that no one can unilaterally dissolve their marriage whose applicability includes the assignment of newly graduated doctors to hospitals, students to schools and human organs available for transplantation to recipients. This article aims to present an introduction to Von Neumann's 'Parlour Games', while focusing on the Stable Marriage problem. The text gives a brief introduction to the problem's history and relevance, presenting its description, an example case, and the implementation of its most common resolution algorithm, along with a stability check algorithm implementation.

Keywords: Algorithm, Data Structures, Parlor Games, Stable Marriage, Gale-Shapley

1. INTRODUÇÃO

Trataremos aqui de um problema bastante conhecido e pesquisado em diferentes áreas do saber, indo da Economia até a Ciência da Computação, o qual é conhecido como o *Problema do Casamento Estável*. Nossa contribuição será o de fazer uma breve resenha do problema e apresentar as formas já conhecidas de resolvê-lo de maneira rápida e no caso ótima. Adicionalmente, abordaremos o contexto no qual este aparece, *Teoria dos Jogos*, incluindo outros *jogos* bastante conhecidos com suas políticas ótimas de resolução. Para o problema específico do *Casamento Estável* traremos um sumário de sua história e relevância, a resolução de casos ilustrativos até a implementação em linguagem de computação que poderá ser utilizada a partir do texto.

O problema do casamento estável ou emparelhamento estável, também conhecido como SM (*Stable Marriage*) foi proposto pelos pesquisadores David Gale e Lloyd Shapley em 1962. O problema consiste em resolver o problema do casamento entre um número par de pessoas, formando casais, de forma que ninguém possa unilateralmente dissolver seu casamento. Há inúmeras aplicações práticas para esse problema, incluindo a atribuição de médicos recém-formados a hospitais, estudantes a escolas e órgãos humanos disponíveis para transplante a receptores. Desde então, o problema é muito estudado por pesquisadores da Matemática, Ciência da Computação, Economia e da Pesquisa Operacional.

A importância e relevância do problema para a Ciência foi o reconhecimento das pesquisas de Alvin Roth e Lloyd Shapley para receberem o prêmio Nobel de Economia do ano de 2012, lendo-se no sumário daquela Academia para tal feito: “A Teoria das Alocações Estáveis e a Configuração Prática de Mercados” (“*The Theory of Stable Allocations and the practice of Market Design*”). Observe-se que David Gale já era falecido à época da premiação, sendo esse um impeditivo da Fundação Nobel, isto é, o de laurear somente a pessoas que estejam vivas. (Nobel, 2023).

2. FUNDAMENTAÇÃO TEÓRICA

2.1 JOGOS DE SALÃO

O problema pode ser entendido como pertencente a uma classe ampla de problemas conhecido como "*Jogos de Salão*" (*Parlor Games*) que foi introduzido por Von Neumann em 1928 (Von Neumann, 1959). Sendo que essa ampla classe deu origem à *Teoria de Jogos* criada também por John von Neumann em colaboração com Oskar Morgenstern. John von Neumann é considerado como um dos maiores cientistas do século XX, tendo influenciado diretamente múltiplas áreas do conhecimento, por exemplo, os atuais computadores utilizam a chamada *arquitetura von Neumann*, há uma definição sua para os números ordinais, foi um dos pioneiros na previsão numérica do tempo.

Fazendo uma busca nas bases indexadas do saber, notamos que somente na IEEE além de sua biografia escrita pelo igualmente conhecido cientista Stanislaw Ulam, há 811 resultados para seu nome, 10206 referências que o citam na base *Web of Science Clarivate™*, 69988 documentos na base Scopus, sendo que von Neumann faleceu há mais de 65 anos. Os acessos a essas bases foram feitos no mês de setembro de 2023 através do Portal de Periódicos da CAPES.

Antes, porém, apresentamos a seguir três jogos mais conhecidos e em um certo sentido mais simples de jogos de salão que todos conhecemos e a estratégia, eventualmente, ótima para resolvê-los.

2.1.1 TESOURA, PEDRA E PAPEL

O primeiro exemplo deste tipo de jogo é o do '*Tesoura*, *Pedra* e *Papel*'. Não será feita a explicação desse jogo, já que é muito conhecido. A estratégia ótima para cada um dos jogadores consiste em proceder à escolha do que ser feito a partir do uso de um gerador de números (pseudo) aleatórios. Por exemplo, através de um programa que simule um dado com 6 faces, se os números que aparecerem forem *1* ou *2* escolhe-se *Tesoura*, se forem *3* ou *4* escolhe-se *Pedra* e caso sejam *5* ou *6* a escolha recai em *Papel*.

Note que as escolhas devem usar um gerador de números (pseudo) aleatórios, se uma pessoa decidir jogar sempre em uma mesma ordem: *Pedra*, *Tesoura* e *Papel*, o adversário notará um padrão e vencerá o jogo após algumas rodadas. Muito embora, possa ocorrer na prática uma sequência de dados como 3, 2, 6, 4, 2, 5, 4, 1, 6, 3, 1, 5 parece ser gerada usando um gerador daquele tipo

de números, como a política é a de escolher uma série fixa de *Pedra*, *Tesoura* e *Papel* o adversário notará que há um padrão, mesmo que quem usar tal política declarar que está usando um dado para obter sua sequência.

Perecebe-se que aqui ficará patente que as escolhas ocorrem como a partir de *Pedra*: (3 ou 4); *Tesoura*: (1 ou 2) e *Papel*: (5 ou 6). Ainda, que a política de usar o "*componente aleatório*" pode ser provada ser ótima, no sentido que todos os participantes têm igual chance de vencer após uma quantidade muito grande de rodadas.

Se fizermos somente um único teste de frequência de ocorrências poderíamos ser induzidos a pensar que de fato a sequência numérica obtida é de 1/3. Mas, haveria a suspeita de que a sequência repetida de *Pedra*, *Tesoura* e *Papel* sempre nesta ordem deve ter um padrão que está sendo seguido. Para o caso há o *Teste de Séries* que indicaria que não se tem uma sequência obtida por um gerador de números (pseudo) aleatórios.

2.1.2 DIVISÃO DE BOLO

O segundo exemplo que é bastante conhecido é o da Divisão do bolo de maneira justa (Steinhaus, 1948). Aqui, deve-se cortar um bolo entre n pessoas de tal forma que ao final cada participante esteja feliz com seu pedaço e que ninguém possa ter inveja ao final da divisão. Quanto à inveja, ninguém no jogo poderá ter argumentos que o quinhão de outra pessoa lhe é maior. Note que a satisfação de cada jogador depende das ações dos outros e que uma política ótima deve ser igualmente conhecida por todos.

Para o caso em que só há duas pessoas a política ótima, que é a do *bom senso*, consiste em que uma pessoa faz o corte do bolo e a outra escolhe qual pedaço ficará para si. Ambos os jogadores têm que se convencer que essa regra é a melhor para ambos. Uma argumentação é a de que ninguém poderá reclamar que o pedaço da outra pessoa é maior que o seu, já que ao cortar, a primeira pessoa tentará ser a mais correta possível já que quem não cortou irá escolher antes e, portanto, qualquer pedaço lhe será indiferente. E quem não cortou, mas escolheu primeiro não poderá reclamar, pois foi a primeira pessoa a escolher. Para o caso em que se têm 3 pessoas, a estratégia ótima só foi encontrada após 32 anos que o problema surgiu na literatura, (Robertson; Webb 1998).

Uma solução muito conhecida para o caso da divisão do bolo entre 3 pessoas consiste em se ter um árbitro que irá deslocar uma lâmina de tamanho suficiente para cortar o bolo em uma de suas dimensões. O árbitro começará a deslocar a lâmina da esquerda para a direita o primeiro jogador que disser 'pare' ficará com o pedaço cortado à esquerda. Na sequência utiliza-se a política anterior de um corta e o outro escolhe. A ideia desse algoritmo é a de todos os jogadores gostariam de dizer 'pare' o mais à direita possível, mas há eventualmente alguém falará 'pare' antes e isto deve ocorrer após ter-se chegado ao seu valor ótimo. Isto é, individualmente, cada um pensará somente no tamanho que lhe é ideal.

Todavia, essa solução envolve um árbitro e além disto não é livre de inveja, já que a primeira pessoa a dizer 'pare' pode eventualmente ao final do jogo argumentar que o primeiro que escolheu após sobrarem somente duas pessoas acabou ficando com o maior pedaço.

Por completude, transcreveremos o algoritmo de Conway, Guy e Selfridge que foi publicado por Stromquist (1980) sem proceder à demonstração de sua corretude e que é ótimo. No jogo há 3 jogadores, *A*, *B* e *C* e um bolo.

1. *A* fará dois cortes que lhe *parecem* ser iguais.
2. *B* ordenará os pedaços por tamanho, sendo eles, P_1, P_2 e P_3, do maior para o menor. No caso para *B* o pedaço P_1 é o maior e P_3 o menor.
3. *B* cortará o eventual excesso, segundo sua percepção, de P_1, chamaremos esse excesso de *E*. Nesse caso, para *B* agora $P'_1 = P_1 - E = P_2$.
4. Os novos pedaços a serem escolhidos será agora: P'_1, P_2 e P_3 que serão escolhidos na seguinte ordem dos jogadores: *C* (primeiro), *B* (segundo) e *A* (terceiro). Se P'_1 não tiver sido escolhido por *C*, então *B* terá que ficar com P'_1.
5. Se *B* ou *C* ficarem com P'_1 quem tiver escolhido P'_1 será chamado agora de J_1 e o outro de J_2. J_2 agora cortará o pedaço *E* em 3 pedaços que considera igual e para esse excesso que foi cortado, cada um dos jogadores escolherá seu pedaço adicional conforme a seguinte ordem: J_1, *A* e J_2.

Observe que o algoritmo não se enquadra na ideia usual que temos daquele tipo de procedimento. Mais ainda, que, embora esse contenha somente 5 passos, a demonstração que ele é correto e ótimo é muito elaborada.

No primeiro jogo que apresentamos a estratégia ótima não é determinística e no segundo introduziu-se a situação de não inveja. O problema a ser mencionado aqui com mais detalhes, Casamento Estável, será determinístico e, também, não terá inveja. A seguir mostramos o último jogo deste tipo e que também é o *mais atual e perigoso*.

2.1.3 O MEDO DA ANIQUILAÇÃO TOTAL COMO POLÍTICA

O caso mais extremo desse tipo de jogo, sendo bastante atual e veiculada nas mais diferentes medias, é o que resultou na política conhecida como M.A.D. (*Mutual Assured Destruction*) ou Destruição Mútua Assegurada que surgiu durante a Guerra Fria e acarretou o acúmulo de armamento nuclear por diversos países. A ideia é a de a utilização de um armamento nuclear é tão devastador que, em princípio, nenhum governo irá dela querer lançar mão.

Nenhum dos lados envolvidos atacará o(s) outro(s) com suas armas nucleares porque todos os lados têm a garantia de serem totalmente destruídos no conflito. Isso leva a um impasse estratégico, onde a estratégia ótima para todos é o de *não* lançar um ataque nuclear. Observe que nenhum lado terá o que ganhar mudando unilateralmente sua própria estratégia enquanto os outros lados mantiverem as suas inalteradas. Neste caso, o equilíbrio se dá através de iterações estratégicas e decisões que levam a resultados estáveis, porém implicando, também, em situações do mais alto risco a todos os participantes.

3. O PROBLEMA DO CASAMENTO ESTÁVEL

Retornando ao jogo que abordaremos aqui, este, consiste em uma situação que há 2n pessoas que devem se casar e formar *n* pares de casais. Aqui casar tem uma conotação mais ampla que o do casamento entre seres humanos conforme mencionado acima. Como outro exemplo, têm-se *n* vagas de empregos a serem preenchidos por *n* pessoas. Mas, somente para simplificar o entendimento e manter a formulação original, admita que haja n pessoas de um grupo A que deve se casar com n pessoas do grupo B.

Uma pessoa do grupo A só pode se casar com uma pessoa do grupo B, cada pessoa tem sua própria lista de preferências em relação a *todas* as outras pessoas do outro grupo. Por notação, a_k é a pessoa *k* do grupo A, b_j é uma pessoa *j* do grupo B. Cada pessoa tem uma relação própria de preferencias, por exemplo, $a_3 = (3, 2, 4, 1)$ significará que para a pessoa a_3 ela gostaria de se casar primeiro com a pessoa b_3, seguido da pessoa b_2, depois pela pessoa b_4 e sua última opção será b_1. Mesmo que haja mais pessoas no exemplo, para a_3 as eventuais demais pessoas lhe são indiferentes, pois assumiremos que ela não as conhece.

O objetivo do jogo é o de se determinar uma alocação de pares de pessoas tal que os casamentos sejam estáveis, no sentido que não há pares de pessoas que estão casadas, mas que há outras pessoas com maior prioridade e que também gostariam de aumentar suas satisfações o que fazer com que aqueles pares desejem buscar o divórcio.

Por exemplo, suponha que todas as pessoas do grupo A tenham como primeira opção a mesma pessoa do grupo B. Tal pessoa do grupo B ao examinar sua própria lista de prioridades formará um casamento estável, já que fará a escolha de sua primeira prioridade. Se formos repetir a mesma ideia para as demais pessoas do grupo A, todas também têm as mesmas preferencias desde a segunda opção até a enésima a ocorrência de pares estáveis será ditado somente pelas escolhas do grupo B. Porém, esse tipo de situação é extremo.

Um caso de instabilidade ocorre em uma situação na qual há dois pares de casais (a_1, b_1) e (a_2, b_2) na qual a_2 e b_1 têm, b_1 precedendo a b_2 na lista de a_2 e o mesmo ocorrendo para b_1, isto é, a_1 precede à a_2. Neste caso ocorrerá um *divórcio* já que (a_2, b_1) formarão um casal em que seus antigos parceiros tinham precedência menor que a atual configuração. O objetivo é o de se encontrar uma alocação na qual não haja divórcio algum, o que à primeira vista parece não ser possível.

Em uma primeira abordagem, parece que a solução do problema só é possível caso tentássemos gerar todas as alternativas possíveis e tentar descobrir se há solução ou não. Deve ser notado que tal quantidade cresce de maneira exponencial no número de pessoas. Para três pares de pessoas há mais de 40 mil casos a serem examinados, mais ainda, há situações na qual a alocação pode fazer com que haja uma sequência de divórcios com geração de ciclo e nesse caso, teríamos que guardar todas os possíveis divórcios e procurar se um ciclo já apareceu ou não. Ainda, em uma primeira abordagem, poderíamos concluir que além de grande, poderia haver problema em que não há uma solução sequer.

Vamos ver um exemplo em que tal situação em que se dá a ocorrência de ciclos. Nas duas figuras a seguir, temos as preferências das pessoas do grupo A, sejam elas, a_1, a_2 e a_3 que aparecem na primeira coluna da Figura 1. Devemos interpretar da seguinte forma, para a pessoa a_1 a pessoa que ela prefere do Grupo B em primeiro lugar é b_2, seguido por b_1 e sua última opção é a pessoa b_3. Para as demais pessoas do Grupo A e aquelas do grupo B faremos a mesma interpretação de acordo com as Figuras 1 e 2 respectivamente.

Figura 1 | Preferências do Grupo A

Grupo A			
a_1	b_2	b_1	b_3
a_2	b_3	b_2	b_1
a_3	b_1	b_2	b_3

Figura 2 | Preferências do Grupo B

Grupo B			
b_1	a_1	a_3	a_2
b_2	a_3	a_1	a_2
b_3	a_2	a_3	a_1

Vamos supor agora o que acontece com os seguintes pares alocados para se casarem:

$$(a_1, b_3); (a_2, b_1) \text{ e } (a_3, b_2).$$

Cada uma das seis pessoas irá consultar sua própria lista de prioridades. Note o que ocorrerá com a pessoa a_3 que está alocada para b_2 que é sua segunda opção. Ela irá tentar se casar com b_1 que é sua primeira opção. Também, ao fazer tal pedido, b_1 também irá querer trocar de pessoa que foi a ela alocada, pois para b_1 a alocação foi para a_2 que é sua última opção. Teríamos uma situação em que há divórcio pois há um par insatisfeito. Nesse caso, a alocação ficaria após tal rearranjo:

$$(a_1, b_3); (a_2, b_2) \text{ e } (a_3, b_1).$$

Note que isso ocorreu devido ao fato de b_2 e a_2 ficarem sem par e, portanto, terem que ser alocados um ao outro. Repetindo o processo de verificação, desta vez a iniciativa partindo de b_1 que foi recém-alocada, perguntará para a_1 que é a primeira pessoa de sua lista. Ao examinar sua própria lista, a_1 aceitará o pedido de b_1 já que b_3 é sua última preferência. Assim, teremos outro rearranjo resultando em

$$(a_1, b_1); (a_2, b_2) \text{ e } (a_3, b_3).$$

Sendo que essa atribuição ainda não havia ocorrido e repete-se a verificação. Agora, b_2 pode melhorar sua escolha, pois a_2 é a última pessoa de sua preferência e falará com a_1 que aceitará a oferta de trocar de par, pois b_2 é sua primeira opção e, portanto, a nova configuração passará a ser:

$$(a_1, b_2); (a_2, b_1) \text{ e } (a_3, b_3).$$

Note que b_2 ao pensar melhor, observou que na configuração atual há ainda a chance de fazer mais uma tentativa que é a de pedir para a pessoa que está em primeiro lugar na sua lista, que é a_3 que aceitará a oferta, pois a_3 prefere b_2 à b_3 esta última estando no final de sua preferência. Assim, após mais um rearranjo chegaremos em

$$(a_1, b_3); (a_2, b_1) \text{ e } (a_3, b_2).$$

Porém, aqui, temos uma repetição e esse processo não irá terminar. Teríamos que ter armazenado a todas essas tentativas para saber se há uma atribuição de pares que torne os casamentos estáveis.

É possível provar que o problema terá soluções sem ciclos se as duas tabelas das pessoas pertencentes à A e à B forem *Quadrados Latinos*. Em um *Quadrado Latino* não há nenhuma pessoa ocupando a mesma ordem em listas de diferentes pessoas. Além disto, haverá várias soluções sem ciclos. As demonstrações destes fatos, embora sejam simples, fogem ao escopo do presente artigo.

A partir do problema original do casamento estável, outros problemas podem ser derivados. São os mais conhecidos: SMI (*Stable Marriage with Incomplete Lists*), a qual permite que um pessoas tenham listas de preferência incompletas. SMT *(Stable Marriage with Ties*), onde pode haver indiferença na lista preferencias. SMTI (*Stable Marriage with Incomplete lists and Ties*), que é uma mescla das duas variantes anteriores: listas de preferência possivelmente incompletas e com indiferença. A definição de indiferença neste caso é quando uma pessoa não tem uma preferência específica entre duas outros do grupo oposto. Por exemplo, em $a_3 = ([3, 2], 4, 1)$, a_3 possui a mesma preferência por 3 ou 2.

Na obra original, Gale e Shapley (1962) além de apresentarem a versão original do problema do casamento estável, também apresentaram uma versão 'poligâmica' do problema, conhecida como o problema de emparelhamento entre estudantes de medicina e hospitais. Esta variante já era explorada anteriormente, de forma que um algoritmo semelhante ao proposto por Gale e Shapley já era usado desde 1952 pelo *NPRM* (N*ational Resident Matching Problem*). Porém, isto só se tornou de conhecimento público anos depois, quando detalhes do algoritmo usado pelo *NPRM* foram divulgados. A versão generalizada do

problema, conhecida como emparelhamento de estudantes com dormitórios também foi apresentada no trabalho, e há época não tinha solução conhecida.

Tanto o problema original quanto suas variações vêm sendo estudadas desde a publicação do artigo seminal de Gale e Shapley. Fruto dessas pesquisas, diversos livros, artigos e teses já foram produzidos, como os de Gusfield e Irving (1989) e Knuth (1976), dedicados exclusivamente ao problema.

3.1 O ALGORITMO GALE-SHAPLEY

O algoritmo de Gale-Shapely foi proposto no artigo seminal de 1962, como uma forma de se encontrar um dos possíveis emparelhamentos estáveis para uma dupla de conjuntos (A, B). O algoritmo é executado da perspectiva de um dos grupos iniciando as 'propostas' de casamento, e o outro grupo aceitando as propostas, ou terminando 'noivados'. Para facilitar a compreensão e manter a nomenclatura original, a analogia é a de um grupo propositor, no caso de Gale e Shapley, sendo de homens, que irão propor um noivado as mulheres do grupo oposto, em sua ordem de preferência pessoal.

O artigo original também demonstra que o emparelhamento estável obtido é otimizado para o grupo fazendo as propostas, já que elas são executadas em ordem decrescente de preferência. Além de também de demonstrar que um casamento estável é sempre possível para uma entrada válida do problema original. Observe, ainda mais uma vez, que mantivemos a nomenclatura original de homens e mulheres, mas que a conotação é a de somente que haja, tecnicamente, um grafo bipartido completo. Os detalhes de tais demonstrações, porém estão além do escopo deste artigo.

A execução do algoritmo é iniciada com um dos elementos do conjunto propositor 'noivando' com sua escolha preferencial. Nesse ponto, qualquer elemento do grupo receptor estará sempre em um estado 'solteira' ou 'noiva'. A partir do momento que se tornar noiva, ela não poderá voltar a ser solteira novamente, mas poderá trocar de noivo, caso receba uma proposta de alguém que ela prefira do que o atual noivo. A execução é continuada enquanto houver homens que não estejam noivos. Sempre que houver um homem livre, ele irá propor a mulher mais bem ranqueada em sua lista de preferência a qual ele não tenha proposto um noivado ainda.

Como o número de homens e mulheres em ambos os conjuntos é igual, e uma mulher após noivar não pode voltar a ser solteira, a execução é sempre finalizada, com um pior caso de n^2 pedidos de noivado sendo feitos. Uma forma de demonstrar que o algoritmo sempre chega a um casamento é

através da contradição. É possível ver que para um casal (h, m), qualquer mulher “m” a qual h há prefira do que sua atual noiva “m”, “h” já propôs um noivado a “m” antes. E (h, m’) não permaneceram como casal porque “m” optou por um parceiro que ela prefere do que “h”. A ideia de se ter noivados, em Computação é chamada de ‘*decisões deferidas*’ ou ‘*decisões postergadas*’ do Inglês, *deferred decisions*. Se uma pessoa tiver que tomar uma decisão, às vezes se preocupar em tomá-la com muita antecedência talvez não seja o melhor. Por exemplo, quando se sai de casa ao serviço de carro, não vale a pena considerar em seu processo de tomada de decisão qual a velocidade que irá economizar mais combustível pensando em todos os sinais de trânsito que se tem ao longo do caminho.

O algoritmo Gale-Shapley sempre encontrará um emparelhamento ótimo para o conjunto propositor. E de mesma forma, sempre gera o pior emparelhamento para o conjunto recebendo as propostas. Tal propriedade é demonstrada no teorema 1.2.3 em (Gusfield; Irving, 1989). A Figura 3 apresenta uma codificação do algoritmo Gale-Shapely.

Figura 3 | algoritmo Gale-Shapely.

```
std::vector<std::pair<T, T>> gale_shapley(
    std::map<T, std::vector<T>> a_preference,
    std::map<T, std::vector<T>> b_preference) {
  std::map<T, T> matched;
  std::map<T, T> matched_reverse;
  std::map<T, int> a_preference_index;

  std::queue<T> q;

  for (const auto &a_entry : a_preference) {
    q.push(a_entry.first);
  }

  while (!q.empty()) {
    T curr_element = q.front();
    q.pop();

    if (matched.contains(curr_element)) {
      continue;
    }

    assert(a_preference_index[curr_element] <
```

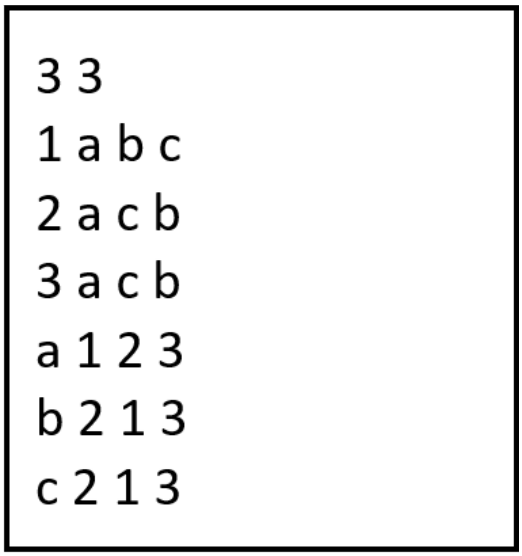
```
assert(a_preference_index[curr_element] <
     (int)a_preference[curr_element].size());

T b_element =
   a_preference[curr_element][a_preference_index[curr_element]++];

// In case b_element is already matched, check if they
// prefer the new option instead of their current match.
if (matched_reverse.contains(b_element)) {
 int curr_element_b_index =
    indexOf(b_preference[b_element], curr_element);
 int matched_element_b_index =
    indexOf(b_preference[b_element], matched_reverse[b_element]);

 // If curr_element is more desired than matched_reverse[b_element],
 // switch.
 if (curr_element_b_index < matched_element_b_index) {
  // Break proposal and set matched_reverse[b_element] to propose again.
  q.push(matched_reverse[b_element]);

  matched.erase(matched_reverse[b_element]);
  matched_reverse.erase(b_element);
assert(a_preference_index[curr_element] <
     (int)a_preference[curr_element].size());

T b_element =
   a_preference[curr_element][a_preference_index[curr_element]++];

// In case b_element is already matched, check if they
// prefer the new option instead of their current match.
if (matched_reverse.contains(b_element)) {
 int curr_element_b_index =
    indexOf(b_preference[b_element], curr_element);
 int matched_element_b_index =
    indexOf(b_preference[b_element], matched_reverse[b_element]);

 // If curr_element is more desired than matched_reverse[b_element],
 // switch.
 if (curr_element_b_index < matched_element_b_index) {
  // Break proposal and set matched_reverse[b_element] to propose again.
  q.push(matched_reverse[b_element]);

  matched.erase(matched_reverse[b_element]);
  matched_reverse.erase(b_element);
```

```
    std::vector<std::pair<T, T>> answer;

    for (auto item : matched) {
      answer.emplace_back(std::make_pair(item.first, item.second));
    }

    return answer;
}
```

Utilizando a implementação base do algoritmo Gale-Shapely, podemos simular a execução de um caso de exemplo. Supondo haver um grupo de seis pessoas, três homens e três mulheres interessados entre si. Vimos anteriormente que por força bruta a quantidade total de casos possíveis seria maior do que 40.000 para esse caso pequeno, o que já é inviável de ser feito com lápis e papel à mão. A entrada é dada inicialmente pela quantidade de elementos em cada grupo, ou seja: (3 3). Cada linha subsequente contém o identificador do elemento, seguido por uma lista ordenada com sua preferência pelos elementos do outro grupo. O primeiro grupo é formado pelas primeiras três linhas de entrada, e o segundo grupo pelas próximas três.

No exemplo ilustrado na Figura 4, temos os dois grupos de três elementos formados por [1, 2, 3] e [a, b, c]. O indivíduo 1 do primeiro grupo tem em sua lista de preferência: [a, b, c], já o indivíduo ‘b’ do segundo grupo tem por preferência: [2, 3, 1].

Figura 4 | Exemplo de aplicação do algoritmo Gale_Shapely.



Na execução de exemplo, usaremos o grupo [1, 2, 3] como grupo propositor. A execução é iniciada com a inclusão dos elementos ainda não noivados na fila de execução [1, 2, 3]. Após isso, o primeiro elemento da fila é removido, neste caso 1, que faz sua proposta de noivado a sua primeira opção ‘a’. Como ‘a’ está solteira, ela aceita a proposta, formando um possível par ‘(1, a)’ e deixando a fila. Como ‘1’ é a primeira opção de ‘a’, este casal não poderá mais ser desfeito, já que não há outra opção no grupo propositor a qual ‘a’ deseja mais do que ‘1’.

O próximo elemento na fila é ‘2’, o qual faz uma proposta de noivado à ‘a’, mas ela já está noiva de ‘1’, e prefere ‘1’ a ‘2’, fazendo ‘2’ voltar a fila de execução. O próximo elemento a ser processado na fila é ‘3’, que inicialmente propõe um noivado a sua primeira opção ‘a’. Porém, ‘a’ já está noiva de sua primeira opção, o que a leva a negar o pedido e ‘3’ a voltar a fila de execução.

O processamento continua com ‘2’ de volta a execução, desta vez propondo a sua segunda opção ‘c’, que ainda está solteira e aceita a proposta. Agora temos ‘(1, a)’ e ‘(2, c)’ como possíveis casais. O último elemento restante na fila, ‘3’, volta a execução. Desta vez, ele propõe a ‘c’, que tem ‘3’ como sua última opção na lista de preferencias. Resultado disto, o pedido de noivado é negado, e ‘3’ volta a fila de execução para seu último pedido de noivado. O pedido é aceito por ‘b’, e temos a formação final de casais ‘[(1, a), (2, c), (3, b)]’.

3.2. VERIFICANDO A ESTABILIDADE

Um ponto relevante nos problemas de busca e otimização é a capacidade de verificar se uma solução é de fato válida. No caso do problema do casamento estável, podemos determinar um algoritmo de validação em tempo $O(n^2)$, para verificar se uma lista de casais forma, ou não, um casamento estável.

A verificação pode ser feita iterativamente, onde dado um casal arbitrário ‘(a, b)’, onde a pertence a A, e b pertence a B. Sendo ‘(a, b)’ um casal dentro da possível lista de que forma um casamento estável. Devemos iterar sobre A, em busca de um elemento a’ o qual a’ prefere b há seu atual par, e b prefere a’ há seu atual par. De forma análoga, a checagem pode ser feita entre os elementos pertencentes a B.

Figura 5 | Exemplo para verificação de estabilidade do algoritmo Gale_Shapely.

```
template <typename T>
bool stability_check(std::map<T, std::vector<T>> a_preference,
                     std::map<T, std::vector<T>> b_preference,
                     std::vector<std::pair<T, T>> matching) {

 std::unordered_map<T, T> match_map;

 // Build a match_map to facilitate match lookup in O(1)
 for (auto [a, b] : matching) {
  match_map[a] = b;
  match_map[b] = a;
 }

 for (auto [a, b] : matching) {
  // Check if a prefers z than b, and vice-versa.
  int a_to_b_pref_index = indexOf(a_preference[a], b);
  // Only checking elements a prefers more than b.
  for (int i = 0; i < a_to_b_pref_index; i++) {
   // To check if z prefers a more than their match.
   T z = a_preference[a][i];
   T current_match = match_map[z];
   int z_to_current_pref_index = indexOf(b_preference[z], current_match);
   int z_to_a_pref_index = indexOf(b_preference[z], a);

   if (z_to_a_pref_index < z_to_current_pref_index) {
    return false;
   }
  }
 }

 return true;
}
```

No exemplo apresentado n Figura 5, podemos verificar a estabilidade do conjunto ‘[(1, a), (2, c), (3, b)]’ com o algoritmo proposto. Podemos iniciar com casal ‘(1, a)’, porém neste caso, ambos já são a primeira opção de cada um, o que mantém a estabilidade do conjunto válida. Para o casal ‘(2, c)’, uma opção possível seria ‘a’, porém ‘a’ já está casada com sua primeira preferência. No caso do casal ‘(3, b)’, ‘3’ tem ‘b’ como sua última preferência, da mesma forma a que ‘b’ tem ‘3’ como sua última opção. Porém, nenhum dos outros elementos disponíveis prefere ‘3’ ou ‘b’ do que seus atuais parceiros. Ou seja, o conjunto ‘[(1, a), (2, c), (3, b)]’ forma um casamento estável.

4. CONSIDERAÇÕES FINAIS

Neste texto, abordamos o grupo de problemas conhecidos como 'Jogos de Salão', com enfoque no problema do casamento estável. Na introdução, fizemos uma breve descrição do problema e sua relevância para a computação. Já na fundamentação teórica, foram descritos três problemas iniciais, para introduzir o leitor a classe de problemas de jogos de salão. No terceiro capítulo, apresentamos o do problema principal, com exemplos visuais e referências as variações clássicas do problema. Ao final, foi incluída uma implementação em linguagem C++ do algoritmo clássico para obtenção de uma solução para o problema, além de outro algoritmo para checagem de estabilidade.

Para manter um texto conciso, mantivemos de fora, propositalmente, outros problemas da classe dos jogos de salão, bem como variações do problema do casamento estável. Alguns exemplos, problemas como o da estabilidade entre colegas de quarto (Stable Roommate Problem), ou o problema da admissão (College Admission Problem) podem ser abordados semelhantemente, sendo eles extensões do problema original.

Tanto o problema do casamento estável, quanto outros jogos de salão continuam sendo pesquisados na comunidade científica. Ligações entre o problema do casamento estável e outros problemas da computação em matemática continuam sendo explorados, bem como otimizações para as ideias e algoritmos presentes. Tais feitos exemplificam o interesse e importância de problemas multidisciplinares ligados a análise de algoritmos.

REFERÊNCIAS

GALE, D.; SHAPLEY, E. L. S. College admissions and the Stability of Marriage, *The* **American Mathematical Monthly**, v. 69, n. 1, p. 9-15, 1962.

GUSFIELD, D.; IRVING, R. W. **The stable marriage problem: structure and algorithms.** [S.l.]: MIT press, 1989.

KNUTH, D. E. **Marriage stables et leurs relations avec d'autres probl`emes combinatoires**. Les Presses de l'Universit´e de Montréal, 1976.

NOBEL, The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 2012. Site Nobel Prize. Disponível em: https://www.nobelprize.org/prizes/economic-sciences/2012/summary/. Acesso em: 03 de Outubro de 2023.

ROBERTSON, J.; WEBB, W. **Cake-cutting algorithms: be fair if you can**, AK Peters, Natick, 1998.

STEINHAUS, H., **The problem of fair division**, *Econometrica*, v. **16**, p. 101-104, 1948.

STROMQUIST, W. How to Cut a Cake Fairly. **American Mathematical**, Monthly 18, no. 8 (October), p. 640-644, 1980

VON NEUMANN, J. From Parlor Games to Social Science: von Neumann, Morgenstern and the Creation of Game Theory, 1928 – 1944 J. von Neumann, 1928, Zur Theorie der Gesellschaftsspiele, *Mathematische Annalen* 1928, 100, pp. 295-320; traduzido por S. Bargmann como "On the Theory of Games of Strategy," em capítulo de *Contributions to the theory of games*. Vol. 4. Eds.: Albert Tucker and T. Duncan Luce. Princeton: Princeton U. Press, 1959, pp. 13-42.