UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE CIÊNCIAS FÍSICAS E MATEMÁTICAS

Jogos Trancados de Dominó

TRABALHO DE CONCLUSÃO DE CURSO

Saimon Marcolino Ventura

Florianópolis, 2013

Saimon Marcolino Ventura

Jogos Trancados de Dominó

Trabalho de Conclusão de Curso apresentado ao Curso de Matemática do Departamento de Matemática do Centro de Ciências Físicas e Matemáticas da Universidade Federal de Santa Catarina para obtenção do grau de Licenciado em Matemática.

Orientador:
Aldrovando Luis Azeredo Araújo

UNIVERSIDADE FEDERAL DE SANTA CATARINA

Florianópolis

2013

Esta monografia foi julgada adequada como TRABALHO DE CONCLUSÃO DE CURSO no Curso de Matemática - Habilitação Licenciatura, e aprovada em sua forma final pela Banca Examinadora designada pela Portaria n° 33/CCM/13.

Prof. Silvia Martini de Holanda Janesch
Professora da disciplina

Banca Examinadora:

Prof. Aldrovando Luis Azeredo Araújo
Orientador

Prof. Nereu Estanislau Burin
Membro

Prof. Márcio Rodolfo Fernandes
Membro

Sumário

1 Introdução

O Dominó é um jogo formado por peças retangulares, dotadas normalmente de uma espessura que lhes dá a forma de paralelepípedo, em que uma das faces está marcada por pontos indicando valores numéricos. O termo é também usado para designar individualmente as peças que compõem este jogo. O nome provavelmente deriva da expressão latina *domino gratias* (*graças ao Senhor*), dita pelos padres europeus para assinalar a vitória em uma partida.

O jogo aparentemente surgiu na China e sua criação é atribuída a um santo soldado chinês chamado Hung Ming, que viveu de 243 a.C a 182 a.C. O conjunto tradicional de dominós, conhecido como sino-europeu, é formado por 28 peças, ou pedras. Cada face retangular de dominó é dividida em duas partes quadradas, ou *pontas*, que são marcadas por um número de pontos de 1 a 6, ou deixadas em branco. Um jogo de dominó é equivalente a um baralho de cartas ou jogo de dados, que podem ser jogados em uma diversidade indeterminada de maneiras. As pedras são geralmente denominadas de acordo com os números em suas pontas. Assim, uma pedra com um 3 de um lado e um 4 do outro, é chamada de três-quatro, por exemplo. Peças com números iguais em ambas as pontas são chamadas *duplos*. Em um jogo de peças, nenhuma delas apresenta uma combinação de *pontas* igual a outra.

Pedras com o mesmo número em uma das pontas são consideradas do mesmo naipe. No caso do conjunto conhecido como duplo-seis, em que a pedra de maior valor é aquela com seis pontos nas duas pontas, as pedras $1-0, 1-1, 1-2, 1-3, 1-4, 1-5$ e $1-6$ pertencem todas ao naipe de **1**, sendo que cada peça, exceto os duplos, sempre irão pertencer a dois naipes.

A forma mais comum de jogar o dominó no Brasil é entre duplas (4 jogadores 2×2), onde cada jogador recebe 7 peças, ou jogar-se em 2 jogadores com 7 pedras cada um e 14 pedras para comprar no caso do oponente não ter a pedra da vez. O jogador que começa a partida é o que tem a peça 6-6. Ele inicia a partida colocando esta peça no centro da mesa. A partir daí, joga-se no sentido horário. O objetivo é baixar todas as peças primeiro, ou fechar o jogo (menos habitual). Jogar para o *fecha* não é modalidade comum nas mais nobres mesas de jogos, sendo permitido somente o *fecha* natural. Quem baixar todas as peças ganha os pontos da soma de

todas as peças que sobrarem na mão do adversário.

O jogo fica fechado quando não é mais possível baixar peças, geralmente quando as duas pontas do jogo têm o mesmo número e não existem mais peças com este número na mão dos jogadores.

Quando o jogo fica fechado, quem tiver menos pontos em peças na mão ganha e leva a pontuação em peças na mão do adversário, no caso de jogo por pontos. Geralmente uma disputa de dominó é feita em várias partidas consecutivas, onde a dupla que acumular 100 pontos primeiro é a vencedora.

Os Jogos sempre foram um tema comum na matemática. De fato, algumas teorias matemáticas se desenvolveram motivadas por problemas de jogos. Relativamente ao nosso trabalho estamos interessados na relação do jogo de dominó com a teoria da contagem ou combinatória. Mais precisamente, o problema tratado neste trabalho é o da contagem do número de jogos trancados possíveis no dominó.

Um problema fácil de enunciar, mas muito complexo de ser resolvido. Inicialmente pretendíamos encontrar uma fórmula combinatória que expressasse a relação do número de jogos trancados com a quantidade de peças envolvidas nestes jogos. Por exemplo, quantos jogos trancados existem com 15 peças. Veremos, no desenvolvimento do trabalho, que para se ter um jogo trancado precisamos no mínimo de 10 peças. Este objetivo inicial se revelou complicado em demasia e assim partimos para a busca de um algoritmo que calculasse o número de jogos trancados em relação ao número de peças no jogo, bem como, exibisse todos estes jogos possíveis.

Foi nesta direção que obtivemos sucesso. Nesse trabalho foram usados alguns resultados como equações lineares com coeficientes unitários. Foram também utilizadas três linguagens de programação, PHP, Matlab e Java.

A princípio, criamos uma maneira de reconhecer e manipular o dominó, em seguida calculamos o número mínimo de peças para se fechar o jogo e a partir desse número mínimo calculamos quantas maneiras distintas existiam. Em seguida fomos adicionando uma nova peça e tentando calcular o número de jogos trancados existentes com este número de peças. Observamos que, adicionando apenas cinco peças, já ficava altamente complexo de se calcular o número de jogos trancados. Foi então que decidimos criar um algoritmo para obter todos os jogos trancados e calcular este número. Neste processo resolvemos algumas questões interessantes relacionadas. Mencionamos duas aqui nesta introdução.

Proposição 1. Em um jogo fechado, os números nas duas extremidades são iguais.

Precisamos de uma definição antes de enunciarmos o segundo problema tratado. Dizemos que uma pedra é ímpar quando a soma de seus números for ímpar. Por exemplo, a pedra 3:2 é uma pedra ímpar.

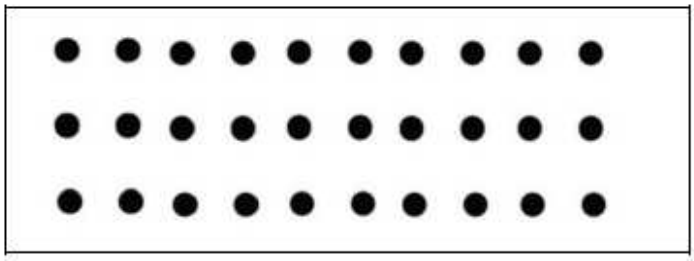
Proposição 2. Em um jogo fechado, a quantidade de pedras ímpares no jogo é par.

2 *Elementos de Combinatória*

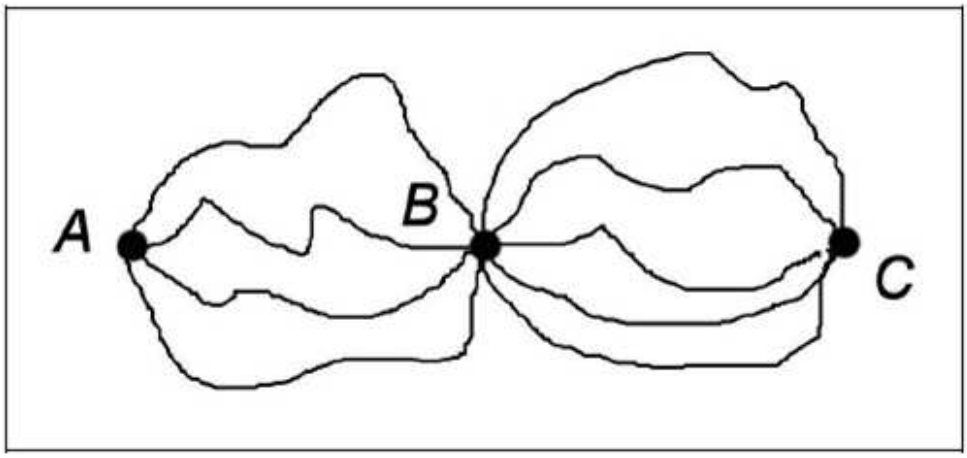
Sempre que nos deparamos com uma pergunta que começa ”De quantos modos possíveis...”, sabemos que estamos frente a um problema de combinatória. A análise combinatória está associada com inúmeros problemas importantes das ciências, em especial a química, biologia, física e ciência da computação. Problemas de química surgem no estudo da estrutura dos átomos, moléculas e cristais, em física no estudo das mecânicas quânticas e estatística, em biologia no estudo da estrutura dos gens e na ciência da computação nas áreas das redes de computadores e design de microcircuitos. Quase todos estes problemas se reduzem a determinação da quantidade de elementos de um dado conjunto. Por isto, uma excelente ainda que não completa definição de análise combinatória é: A Análise Combinatória é simplesmente a Teoria da Contagem. A Análise Combinatória visa essencialmente desenvolver métodos que permitam contar - de uma forma indireta - o número de elementos de um conjunto. Seus métodos são baseados em um certo conjunto de princípios, dos quais o que nos interessa é o princípio multiplicativo.

Definição 2.1 (Princípio Multiplicativo)**.** Considere uma sequência finita de decisões. Suponha que o número de escolhas para cada decisão individual é independente das decisões anteriores. Então o número de maneiras de se fazer o conjunto total de decisões é o produto dos números destas escolhas.

Conte a quantidade de pontos na imagem abaixo.



Se você contou os dez pontos de uma linha e multiplicou por três você acabou de usar o princípio multiplicativo. Quantos caminhos possíveis existem de A até C passando por B na figura a seguir?



Isso não deve ter sido um problema, pois de A até B existem 4 caminhos e de B até C existem 5 caminhos então de A até C passando por B existem 4x5 = 20 caminhos. E novamente você usou o princípio multiplicativo. De um modo em geral, se um processo pode ser dividido em sucessivas etapas, e para cada etapa existe um número finito de possibilidades, então a quantidade total de maneiras de realizar tal processo é a multiplicação das etapas. Uma aplicação para o principio multiplicativo é o cálculo das permutações.

2.1 Permutações

Usaremos a seguinte notação:

$$\#X = \text{ número de elementos de } X.$$

Definição 2.2 (Permutações)**.** Considere um conjunto X com n elementos, i.é $\#X = n$. Seja $1 \leq k \leq n$. Então uma sequência de comprimento k formada com distintos elementos de X denomina-se de uma k-permutação de X. O número de tais permutações denota-se por P_n^k.

O princípio multiplicativo permite determinar este número.

Teorema 2.1. *O número de permutações de k elementos escolhidos de um conjunto com n elementos é:*

$$P_n^k = \frac{n!}{(n-k)!} = n(n-1)(n-2)\cdots(n-k+1).$$

Demonstração: Para escolhermos os k elementos de um conjunto com n elementos fazemos:

- ▷ o primeiro elemento pode ser escolhido de n modos

- ▷ o segundo elemento pode ser escolhido de $n-1$ modos (pois agora temos $n-1$ disponíveis, já que não podemos repetir o primeiro escolhido)
- ▷ o terceiro elemento pode ser escolhido de $n-2$ modos (pois agora temos $n-2$ disponíveis)
- ▷ $\cdots \quad \cdots$
- ▷ o k-ésimo elemento pode ser escolhido de $n-k+1$ modos

Segue do princípio multiplicativo o resultado. ■

2.2 Permutações com repetição

Considere a palavra SATA. Chamamos de anagrama de SATA a todo rearranjamento obtido com as letras desta palavra. no caso de SATA temos: SATA, STAA, SAAT, ASTA, ASAT, TSAA, TASA, ATSA, AAST, TAAS, ATAS e AATS

Exemplo 2.1. Determine o número de anagramas da palavra PAPA.

Solução: Para obtermos as permutações da palavra PAPA podemos usar as permutações de PETA trocando T por P e o E por A.

PETA	TEPA	PATE	TAPE
EPTA	ETPA	APTE	ATPE
PTEA	TPEA	TPAE	PTAE
EAPT	AEPT	EATP	AETP

PAPA	PAPA	PAPA	PAPA
APPA	APPA	APPA	APPA
PPAA	PPAA	PPAA	PPAA
AAPP	AAPP	AAPP	AAPP

No entanto, algumas permutações aparecem repetidas depois que efetuamos as trocas. Especificamente cada linha das permutações de PETA dá origem a uma mesma permutação de PAPA ou anagrama de PAPA. Concluímos que os anagramas de PAPA são em número de 4. Nossa solução foi engenhosa, mas necessita de uma generalização para ser aplicável a outros anagramas. Vamos fazê-la através de um lema.

Lema 2.1. *Suponhamos que existem k diferentes tipos de objetos. O número de n-permutações de n_1 elementos do primeiro tipo, n_2 elementos do segundo tipo,...., n_k elementos do k-ésimo tipo, onde $n = n_1 + n_2 + \cdots + n_k$ é*

$$P_n^{n_1,n_2,\cdots,n_k} = P(n_1,n_2,\cdots,n_k) = \frac{n!}{n_1! \times n_2! \times \cdots \times n_k!}$$

Demonstração: Considere a permutação

$$\underbrace{a_1 a_1 \cdots a_1}_{n_1}\underbrace{a_2 a_2 \cdots a_2}_{n_2} \cdots \underbrace{a_k a_k \cdots a_k}_{n_k},$$

onde os elementos do primeiro tipo aparecem primeiro, os do segundo tipo aparecem logo após e assim sucessivamente até os do k-ésimo tipo por último. Os elementos do primeiro tipo podem ser permutados de $n_1!$ maneiras. Como são iguais darão origem à mesma permutação global. O mesmo acontece se permutarmos os elementos do segundo tipo entre eles, cujo número de permutações seria $n_2!$. Depois os do terceiro tipo e assim sucessivamente. Logo, para cada permutação onde consideramos os elementos de cada tipo como distintos existem $n_1! \times n_2! \times \cdots \times n_k!$ permutações que dão origem a mesma permutação quando consideramos os elementos de cada tipo identificados. Portanto, podemos concluir que,

$$P(n_1,n_2,\cdots,n_k) = \frac{n!}{n_1! \times n_2! \times \cdots \times n_k!}$$

■

2.3 Combinações

Combinações são essencialmente objetos bem diferentes de permutações.

Definição 2.3. Seja X um conjunto com $\#X = n$. Seja $1 \leq k \leq n$. Então uma k-combinação de X é simplesmente um subconjunto de X com k elementos. Denotamos por C_n^k o número de k-combinações de um conjunto com n elementos.

Em um conjunto a ordem dos elementos não importa, portanto o mesmo vale para combinações. Ainda que pareça estranho podemos provar a seguinte fórmula para o número de k-combinações de um conjunto com n elementos usando o princípio multiplicativo.

Teorema 2.2. *O número de k-combinações de um conjunto com n elementos é*

$$C_n^r = \frac{n!}{(n-r)!r!}$$

Exemplo 2.2. Encontre as combinações de tamanho $k = 1, 2$ e 3 do conjunto $X = \{a, b, c\}$.

$k = 1$	$k = 2$	$k = 3$
$\{a\}$	$\{a, b\}$	$\{a, b, c\}$
$\{b\}$	$\{a, c\}$	
$\{c\}$	$\{b, c\}$	

2.4 Combinações com repetição

Consideremos o seguinte exemplo:

Exemplo 2.3. Numa sorveteria o cliente pode montar seu próprio sorvete com 4 porções de sorvete que ele pode escolher entre 7 sabores oferecidos. Ele pode repetir o sabor quantas vezes ele quiser. Quantos sorvetes possíveis podemos formar nesta sorveteria?

Resolução. Este problema é diferente dos até aqui estudados, pois estamos permitindo repetições dos elementos. Vamos resolver este problema usando um código de 0´s e 1´s. Denotemos os sabores por suas iniciais, a saber: chocolate=C, creme=Cr, morango=M, nata=N, abacaxi=A, pistache=P e flocos=F. A cada sorvete possível associamos o seguinte código: entre os sabores colocamos 0´s e para cada colher de um sabor colocamos um 1, começando na ordem, C|Cr|M|N|A|P|F. Por exemplo, para um sorvete formado de 2 colheres de morango 1 de abacaxi e 1 de flocos associamos o código 0011001001. Temos então 6 zeros separadores e 4 uns denotando os sabores e suas quantidades escolhidas. Observe que diferentes sorvetes acarretam em diferentes sequências de 6 zeros e 4 uns. Igualmente dada uma sequência de 6 zeros e 4 uns, associamos uma escolha possível de sabores para formar um sorvete. Segue que o número de sorvetes possíveis é igual de permutações com repetições de 6 0's e 4 1's. Logo o número de sorvetes é

$$\frac{(6+4)!}{4!6!} = \frac{10 \times 9 \times 8 \times 7}{4 \times 3 \times 2} = 210.$$

O problema anterior pertence a uma classe de problemas que envolvem combinações com repetições. Podemos definir uma combinação com repetição como a extensão natural de um subconjunto com repetição.

Definição 2.4. Seja X um conjunto com n elementos. Seja k um inteiro positivo. Uma combinação com repetição de comprimento ou tamanho k, ou também uma k-combinação com repetição do conjunto X, é um *subconjunto com repetição* com k elementos escolhidos dos elementos de X, onde estamos admitindo que os elementos possam aparecer repetidos, sem restrição do número de repetições.

Teorema 2.3. *Seja X um conjunto com n elementos. Seja k um inteiro positivo. Então, o número de k-combinações com repetição dos elementos do conjunto X é*

$$CR_n^k = P(n, n-k) = \frac{(k+n-1)!}{k!(n-1)!}.$$

Demonstração: Seja $X = \{x_1, x_2, \cdots, x_n\}$. Então para cada k-combinação com repetição associamos o seguinte código: separamos os elementos de X usando o símbolo 0 e para cada elemento de X na sua posição correspondente a ordenação acima colocamos tantos *uns* quantas forem as repetições deste elemento na combinação. Assim obtemos uma sequência formada de $(n-1)$ *zeros* pois precisamos de $n-1$ separadores e k uns onde os *uns* são colocados na posição correspondente ao elemento que eles estão associados com as repetições exatas que este elemento aparece na combinação. Segue que o número de k-combinações com repetição de um conjunto com n elementos é igual ao número de sequências de $n-1$ zeros e k *uns*, isto é o número de permutações com repetição de $n-1$ *zeros* e k *uns*, $P(n-1, k)$. Segue que

$$CR_n^k = P(n-1, k) = \frac{(n-1+k)!}{(n-1)!k!}.$$

■

2.5 Equações lineares com coeficientes unitários

Considere a seguinte equação $x + y + z = 12$. Observe que a equação é linear com três variáveis e que os coeficientes são todos uns.

Definição 2.5. Uma equação do tipo

$$x_1 + x_2 + \cdots + x_n = k \qquad \text{com } k > 0 \tag{2.1}$$

denomina-se de equação linear com coeficientes unitários.

O problema que se coloca é determinar o número de soluções inteiras não-negativas (ou positivas) de uma equação destas. Por exemplo considere a equação:

$$x + y + z = 3.$$

Problema 1. Vamos tratar primeiro do problema das soluções não-negativas da equação

$$x_1 + x_2 + \cdots + x_n = k.$$

Teorema 2.4. *O número de soluções não-negativas da equação*

$$x_1 + x_2 + \cdots + x_n = k \qquad com\ k > 0$$

é

$$\#S = \frac{(k+n-1)!}{k!(n-1)!}$$

Demonstração: Seja um conjunto com n elementos $X = \{x_1, x_2, \cdots, x_n\}$. Então para cada k-combinação com repetição de X seja $k_i \in \mathbb{N}$ o número de vezes que o elemento x_i aparece na combinação escolhida para $i = 1..n$. Como o comprimento da combinação é k, segue que

$$k_1 + k_2 + \cdots + k_n = k,$$

e assim $x_1 = k_1, x_2 = k_2, \cdots, x_n = k_n$ é claramente uma solução da equação estudada. Reciprocamente se $x_1 = k_1, x_2 = k_2, \cdots, x_n = k_n$ é uma solução então podemos formar uma k-combinação com repetição do conjunto X escolhendo k_1 representantes do elemento x_i para cada $i = 1, 2, ..., n$. Segue que as soluções não-negativas da equação 2.1 estão em correspondência biunívoca com as k-combinações com repetição de um conjunto com n elementos. Logo o número de soluções da equação 2.1 é

$$\#(\text{soluções equação linear}) = \frac{(n-1+k)!}{(n-1)!k!} \tag{2.2}$$

■

No próximo exemplo exibiremos um modo gráfico de resolver o problema do número de soluções da Eq 2.1.

Exemplo 2.4. Exiba todas as soluções não-negativas da equação $x_1 + x_2 + x_3 = 2$.

Resolução. Seja uma sequência de $n + k - 1 = 3 + 2 - 1 = 4$ caixinhas:

□□□□

das quais marcaremos com um X duas delas. Observe na tabela abaixo a relação das soluções com as sequências de caixinhas.

$$\begin{array}{lll} aa & (2,0,0) & \square\square\boxtimes\boxtimes \\ ab & (1,1,0) & \square\boxtimes\square\boxtimes \\ ac & (1,0,1) & \square\boxtimes\boxtimes\square \\ bb & (0,2,0) & \boxtimes\square\square\boxtimes \\ bc & (0,1,1) & \boxtimes\square\boxtimes\square \\ cc & (0,0,2) & \boxtimes\boxtimes\square\square \end{array}$$

A relação é fácil de entender. A quantidade de caixinhas desmarcadas até a primeira caixinha marcada é o valor de x_1, a quantidade de caixinhas desmarcadas entre a primeira marcada e a segunda marcada é o valor de x_2 e finalmente a quantidade de caixinhas desmarcadas após a segunda caixinha marcada é o valor de x_3. O leitor certamente pode através deste procedimento estabelecer outro método de prova do resultado geral.

Para o caso de soluções positivas temos que impor a condição $k \geq n$.

Teorema 2.5. *O número de soluções positivas da equação linear de coeficientes unitários Eq 2.1 com a condição $k \geq n$ é*

$$\#S = C_{k-1}^{n-1} = \binom{k-1}{n-1}$$

Demonstração: Para demonstrarmos este teorema, considere k *uns* dispostos em uma sequência horizontal. Existem exatamente $k-1$ espaços entre os *uns*. Selecione $n-1$ destes $k-1$ espaços e em cada um destes buracos coloque um *zero*. Defina x_i como o número de *uns* que existem entre o *zero* $i-1$ e o *zero* i, para $i = 2..n-1$. Defina x_1 como o número de *uns* que antecedem o primeiro *zero* e x_n como o número de *zeros* que sucedem o último *zero*. Observem que cada $x_i \neq 0$ e que

$$x_1 + x_2 + \cdots + x_n = k$$

pois estamos escolhendo todos os *uns* e a sua soma era k. Segue que o número de soluções positivas da equação é o número de possibilidades de escolhas de $n-1$ buracos entre $k-1$ buracos disponíveis e portanto

$$\#S = C_{k-1}^{n-1} = \binom{k-1}{n-1}.$$

■

3 *Elementos do dominó*

Quem não sabe jogar dominó? Talvez um dos jogos mais conhecidos do planeta, suas origens são obscuras justamente por ser um dos jogos praticados pelos seres humanos há mais tempo. Mesmo pessoas que não gostam do jogo ou crianças que ainda não entendem as regras, gostam de usar suas peças como blocos de construção ou como elementos em intrincados desenhos que se desenrolam diante de todos com o simples movimento de queda sequencial das peças; o famoso efeito dominó.

3.0.1 Qual é a origem do dominó?

Existem várias versões que tentam decifrar de onde veio o jogo, mas nenhuma delas até hoje pôde ser confirmada. Acredita-se, porém, que ele tenha surgido na China. A primeira vez que um registro escrito menciona a existência do jogo de dominó remonta aos anos de 234 a 181 a.C. na época do imperador Hui Tsung inventado por um soldado chamado Hung Ming, o jogo de dominó era conhecido como Kwat p'ai (tabletes de ossos). Os primeiros indícios da presença do dominó na Europa são de meados do século XVIII, quando era jogado nas cortes de Veneza e Nápoles. As peças eram feitas de ébano, com pontos de marfim, representando os números. O antigo dominó chinês traz todas as 21 combinações que podem ser obtidas ao lançar dois dadoscubicos de seis faces, sugerindo que um jogo possa ter nascido do outro. Já na Europa, há sete peças a mais, combinando esses números também com o zero. Alguns estudiosos sustentam até que, por ser extremamente simples, o jogo pode ter aparecido simultaneamente em várias partes do mundo.

O nome dominó provavelmente deriva da expressão latina "domino gratias", que significa "graças a Deus", dita pelos padres europeus enquanto realizavam boas jogadas. Atualmente, o dominó é jogado em quase todos os países do mundo, mas é mais popular na América Latina. Na China, ele deu também origem a outro jogo, mais complexo: o mah jong.

Características construtivas

Tradicionalmente feito de marfim, osso ou madeiras escuras, como ébano, com os pontos

marcados em cores contrastantes, hoje o dominó é facilmente encontrado em uma diversidade de materiais que vão desde versões semidescartáveis em papel cartão a modelos de luxo em pedras como mármore, granito, pedra-sabão, ou metais diversos, além de plásticos variados. É comum também às peças trazerem pontos de cores diferentes associadas ao número representado, ou ainda a substituição dos pontos por imagens.

3.0.2 Conjuntos

Embora raramente encontrados no Brasil, os jogos de dominós podem se apresentar em outras versões, além do conhecido duplo-seis. Além deste, existem também o duplo-nove, duplo-doze, duplo-quinze e duplo-dezoito. A quantidade de pedras em cada conjunto varia de acordo com as combinações possíveis entre as pontas disponíveis.

Conjunto	Pedras	Pontos
Duplo 6	28 pedras	168 pontos (será usado nessa pesquisa)
Duplo 9	55 pedras	495 pontos
Duplo 12	91 pedras	1092 pontos
Duplo 15	136 pedras	2040 pontos
Duplo 18	190 pedras	3420 pontos

3.0.3 Regras

Atualmente existem muitas regras distintas para o jogo de dominó. A que adotaremos nesse estudo será a regra utilizada no Centro Acadêmico de Matemática da Universidade Federal de Santa Catarina. O jogo é jogado por quatro pessoas. O dominó é composto de vinte e oito peças, no qual cada peça é composta de dois lados, e cada lado tem um número de zero a seis. Após embaralhar as vinte e oito peças com suas faces para baixo, cada jogador escolhe sete peças. A primeira partida é iniciada por quem possuir o duplo de seis. As demais partidas são iniciadas por quem ganhou a partida anterior. O jogo tem andamento em sentido horário. Quando o jogo é trancado as duplas somam todos os números que tem em posse, a dupla que tiver menos pontos vence, no caso de ambas as duplas terem o mesmo números de pontos, perde quem trancou o jogo. Toda vez que uma dupla ganha a partida, a dupla que perdeu a mesma deve somar todos os números em suas mãos e anotar como pontos que perdeu. Ambas as duplas começam com cem pontos. Ganha a partida a dupla que fizer a outra dupla perder cem pontos.

3.0.4 Peça de dominó

É uma peça composta por dois lados, sendo que os lados variam de zero a seis. As pedras são geralmente denominadas de acordo com os números em suas pontas. Assim, uma pedra com um 3 de um lado e um 4 do outro, é chamada de três-quatro, por exemplo. Peças com números iguais em ambas as pontas são chamadas *duplos*. Em um jogo de peças, nenhuma delas apresenta uma combinação de *pontas* igual a outra.

Pedras com o mesmo número em uma das pontas são consideradas do mesmo naipe. No caso do conjunto conhecido como duplo-seis, em que a pedra de maior valor é aquela com seis pontos nas duas pontas, as pedras 1-0, 1-1, 1-2, 1-3, 1-4, 1-5 e 1-6 pertencem todas ao naipe de *1*, sendo que cada peça, exceto os duplos, sempre irão pertencer a dois naipes. Na forma clássica do jogo, são sete números (de zero a seis), combinados entre si. Matematicamente: C(7,2) + 7 = C(8,2) = 28. Observando isso é fácil ver que cada número aparece oito vezes. Sendo seis peças distintas e a sétima peça o duplo.

3.0.5 Extremidades do jogo

São as peças das pontas, as que estão com uma ponta solta, ou seja, as que definem as próximas peças a serem encaixadas.

3.0.6 Encaixar peça

Quando o jogo é iniciado e uma peça é jogada na mesa, ela possui dois números, a próxima peça a ser jogada deve ter o um dos números da peça na mesa. Só é permitido encaixar uma peça em cada lado de outra peça.

3.0.7 Passar a vez

Quando o jogador da vez não tem posse de uma peça que contenha os números das extremidades, ele é obrigado a passar a vez. Normalmente se dá três batidinhas na mesa para demonstrar tal falta de posse das peças.

3.0.8 Bater

Quando um jogador consegue jogar suas sete peças, ele é o ganhador do jogo. E portanto iniciará a próxima partida.

3.1 Jogo trancado

Quando acontecer das duas extremidades serem o mesmo número e todas as peças que contenham tal número já tenham sido jogadas, não há mais possiblidades de jogar. Dizemos então que o jogo está trancado.

3.1.1 Representação decimal de peças

As peças de dominó serão representadas pelos números em suas pontas, como exemplo a peça que tem os números um e dois respectivamente é a peça 12, e a peça que tem os números dois e um respectivamente é a peça 21. Sabendo que existe apenas uma peça com esses números, então a peça 21 é a peça 12.

3.1.2 Bloco

Definição 3.1. Um bloco de Z é um conjunto de peças ligadas que contém em ambas as pontas o naipe Z, mas não contém o naipe Z em seu interior.

Exemplo 3.1. Vamos analisar o pedaço de jogo 01-12-25-56-64-43-32-22-24-40 quanto a existência de blocos.

Analisando quanto ao naipe 0, é fácil verificar que os únicos 0 que existem estão nas extremidades, então é um bloco de 0.

Analisando quanto ao naipe 1, note que existem 2 peças com o naipe 1 mas elas estão dispostas de tal maneira que não há peças entre elas. Logo não existe sub-bloco de 1.

Analisando quanto ao naipe 2, o trecho 01-12 não é bloco pois só contém uma peça de 1 do naipe 2, mas no trecho 25-56-64-43-32 existem peças no interior e as extremidades são do naipe 2, logo esse pedaço é um bloco de 2.

Usando o mesmo raciocínio, não há sub-blocos de 3, o trecho 43-32-22-24 é bloco de 4, não há sub-bloco de 5, e não há sub-bloco de 6.

Um bloco qualquer tem uma peça que abre o bloco, um peça que fecha o bloco, e obrigatoriamente peças no seu interior que começam e terminam com a ligação com as pontas.

Definição 3.2. O número que representa o bloco é a quantidade de peças do bloco subtraído de 3.

Exemplo 3.2. O bloco 01-12-20 é um 0 bloco de 0, pois só contém 3 peças. Esse é o menor bloco possível. O bloco 01-16-62-20 é um 1 bloco de 0. O bloco 01-16-65-52-20 é um 2 bloco de 0.

Definição 3.3. Quando na mesa encontram-se jogados o duplo Z e três Z blocos, o jogo é dado como fechado, porque as extremidades são em número de 6 mais o duplo, correspondendo a 7 possíveis representações do Z.

Exemplo 3.3. Por exemplo: 01-11-12-20 - 03-34-40 - 05-55-54-46-60 uma vez encaixados estes blocos de alguma maneira, a única peça que resta a ser jogada é o duplo zero. Neste momento o jogo está fechado.

Lema 3.1. *Em um jogo trancado com dois 0 blocos, que seja maximal ,isto é, que usa todas as 28 peças do dominó, o bloco maior terá 21 peças e será um 18 bloco.*

3.1.3 Jogo modelo (exemplo de jogo trancado com mínimo de peças)

O jogador um começa com a peça duplo seis, 66.

O jogador dois joga a peça 65 ficando então, 66-65 note que se ele tivesse jogado do outro lado dizemos que ele jogou a peça 56.

O jogador três jogou a peça 46, ficando o jogo 46-66-65.

O jogador quatro por não ter peças com os números quatro ou cinco, passa a vez.

O jogador um joga a peça 34, e o jogo está assim 34-46-66-65.

O jogador dois ao notar que seu parceiro passou com quatro e cinco, tenta eliminar a ponta de cinco jogando a peça 50, deixando jogo 34-46-66-65-50.

O jogador três por não ter peças com o número três, ele joga a peça 06 ficando o jogo 34-46-66-65-50-06.

O jogador quatro joga sua peça 63, note que essa peça pode ser encaixada em ambos os lados, portanto tenho que definir qual lado ele jogou, ou seja, ele joga a peça 63 a esquerda do jogo. Ficando o jogo 63-34-46-66-65-50-06.

O jogador um joga a peça 61 deixando o jogo 63-34-46-66-65-50-06-61.

O jogador dois por não ter mais peças com o número seis é obrigado a jogar na ponta de um, jogando a peça 12 e o jogo ficando 63-34-46-66-65-50-06-61-12.

O jogador três ao perceber que só tem peças de número pequeno na sua mão resolve jogar a peça 26, mas ele pode jogar ela a esquerda do jogo e continuar o jogo, ou pode jogar a direta e trancar o jogo pois todas as peças que contem o seis já estão jogadas, e claro que ele joga a direita trancando o jogo, pois tendo peças pequenas a soma delas tem mais chance de vencer.

Ficando o jogo modelo 63-34-46-66-65-50-06-61-12-26.

Observe que o jogo tem 10 peças.

Definição 3.4. Usando três 0 blocos e um duplo do mesmo naipe dos blocos, obtemos o menor jogo possível ou jogo minimal. Como cada 0 bloco tem 3 peças então no total esse jogo tem $3x3+1=10$ peças.

3.2 Possibilidades distintas de jogos

3.2.1 Possibilidades distintas do jogo mínimo (10 peças)

Sejam três 0 blocos, e um duplo do mesmo naipe. Existem algumas escolhas que fazemos quando montamos esse jogo. A primeira decisão é qual o naipe dos blocos. Existem 7 naipes disponíveis. Com isso já está decidido qual duplo será usado. A segunda é quais as possíveis maneiras de se criar três 0 blocos. Ou seja as permutações das peças que definem os blocos. Como são três blocos, são definidos por 6 peças, então, só existem $P_6^1 = 6!$ maneiras de se construir estes blocos. Agora precisamos colocar o duplo entre os blocos. Para isso existem 4 possibilidades para colocá-lo ou seja $C_4^1 = 4$. Até aqui estamos contando todas as possibilidades, mas como no dominó não há ordem, então estamos contando duas vezes cada possibilidade. Para resolver esse problema podemos colocar o duplo em apenas dois lugares dos 4 possíveis. No total, então, temos $7 \times 6! \times 2 = 7! \times 2 = 10080$ maneiras diferentes de escrever os jogos trancados com 10 peças.

Definição 3.5. A representação de um jogo trancado será na forma X-Y-Z, sendo X o primeiro bloco, Y o segundo bloco e Z o terceiro bloco.

Lema 3.2. *A quantidade de jogos trancados de N peças é múltiplo da quantidade de jogos com 10 peças.*

Demonstração: Seja um jogo trancado com N peças, sendo $10 < N \leq 28$, então existem 7 possibilidades de escolher o naipe que fecha, existem 2 lugares para o duplo ser colocado, e para criar os blocos, existem 6 peças do naipe do fecha cujas permutações são em número de 6 peças é $P_6^1 = 6!$. O resto da contagem é de acordo com as $N - 10$ peças restantes. Sendo então $7 \times 6! \times 2 = 10080$ um divisor da quantidade total de possibilidades. ∎

Observação 3.1. Um jogo com 11 peças é formado por um duplo, dois blocos com 3 peças (0 blocos) e um bloco com 4 peças (1 bloco). Então adotaremos a notação 0-0-1 ou 001 para esse tipo de jogo. Se o jogo contém um 4 bloco, um 5 bloco e um 7 bloco, podemos denotar como 457, 475, 745, 754, 574 e 547.

3.2.2 Possibilidades distintas do jogo de (11 peças)

Tirando as 10 peças do jogo mínimo existe 1 peça disponível, essa peça pode ser adicionada em qualquer um dos 3 blocos, e o bloco em que a peça for adicionada terá 4 peças. Então o bloco que era 01-12-20 passa a ser 01-1x-x2-20. O x tem que ser qualquer naipe diferente do que está

fechando. Existem então 6 possibilidades para cada bloco com 4 peças. Usando a equação 2.1 sabemos que existem 3 maneiras de escrever 1 como a soma de 3 números não negativos. São elas 100, 010, 001 e para cada uma dessas possibilidades existem 6 possibilidades diferentes de adicionar o x no bloco. Portanto temos 7(do naipe que fecha), 6!(da permutação das peças), 2(dos lugares pro duplo), 3(da solução a+b+c=1) e para cada solução temos 6 possibilidades, logo existem $7x6!x2x(3x6) = 10080x18 = 181440$ possibilidades de jogos com 11 peças.

Lema 3.3. *A permutação dos blocos gera o mesmo número de possibilidades de jogos trancados.*

Demonstração: Seja a,b,c inteiros não negativos e $a+b+c \leq 21$. Seja n a quantidade de combinações possíveis para o jogo a-b-c, e m a multiplicidade da solução. O jogo trancado a-b-c tem $7! \times 2 \times m \times n$. ■

Exemplo 3.4. O jogo 00 - 01-1x-xy-y2-20 - 03-3z-z4-40 - 05-56-60 tem m possibilidades de ser escrito. O jogo 00 - 03-3z-z4-40 - 01-1x-xy-y2-20 - 05-56-60 tem m possibilidades de ser escrito.

3.2.3 Possibilidades distintas do jogo de (12 peças)

Estudando o caso que o jogo é composto de um bloco com cinco peças e dois blocos com três peças (2-0-0) ou 01-1x-xy-y2-20 - 03-34-40 - 05-56-60.
Quando x é igual a 1, então y pode ser igual a {2, 3, 4, 5, 6 }.
Quando x é igual a 2, então y pode ser igual a {3, 4, 5, 6 }.
Quando x é igual a 3, então y pode ser igual a {1, 2, 5, 6 }.
Quando x é igual a 4, então y pode ser igual a {2, 4, 5, 6 }.
Quando x é igual a 5, então y pode ser igual a {2, 3, 4, 5 }.
Quando x é igual a 6, então y pode ser igual a {2, 3, 4, 6 }.

Totalizando 21 maneiras de escrever com 2-0-0.

Estudando o caso que o jogo é composto de um bloco com três peças e dois blocos com quatro peças (1-1-0) ou 01-1x-x2-20 - 03-3y-y4-40 - 05-56-60
Quando x é igual a 1, então y pode ser {1, 2, 3, 4, 5, 6 }.
Quando x é igual a 2, então y pode ser {1, 2, 3, 4, 5, 6 }.
Quando x é igual a 3, então y pode ser {3, 4, 5, 6 }.
Quando x é igual a 4, então y pode ser {3, 4, 5, 6 }.
Quando x é igual a 5, então y pode ser {1, 2, 3, 4, 5, 6 }.
Quando x é igual a 6, então y pode ser {1, 2, 3, 4, 5, 6 }.

Totalizando 32 maneiras de escrever com 1-1-0.

No total então existem $7!x2x3x(21+32) = 1602720$ possibilidades de jogos distintos com 12 peças.

3.2.4 Possibilidades distintas do jogo de (13 peças)

Estudando o caso que o jogo é composto de um bloco com seis peças e dois blocos com três peças (3-0-0) ou 01-1x-xy-yz-z2-20 - 03-34-40 - 05-56-60.
Quando x é igual a 1 e y é igual a 3 então z pode ser {2, 3, 5, 6 }.
Quando x é igual a 1 e y é igual a 4 então z pode ser {2, 4, 5, 6 }.
Quando x é igual a 1 e y é igual a 5 então z pode ser {2, 3, 4, 5 }.
Quando x é igual a 1 e y é igual a 6 então z pode ser {2, 3, 4, 6 }.
Quando x é igual a 2 e y é igual a 3 então z pode ser {5, 6 }.
Quando x é igual a 2 e y é igual a 4 então z pode ser {5, 6 }.
Quando x é igual a 2 e y é igual a 5 então z pode ser {3, 4 }.
Quando x é igual a 2 e y é igual a 6 então z pode ser {3, 4 }.
Quando x é igual a 3 e y é igual a 3 então z pode ser {2, 5, 6 }.
Quando x é igual a 3 e y é igual a 5 então z pode ser {1, 2, 4, 5 }.
Quando x é igual a 3 e y é igual a 6 então z pode ser {1, 2, 4, 6 }.
Quando x é igual a 4 e y é igual a 4 então z pode ser {2, 5, 6 }.
Quando x é igual a 4 e y é igual a 5 então z pode ser {1, 2, 3, 5 }.
Quando x é igual a 4 e y é igual a 6 então z pode ser {1, 2, 3, 6 }.
Quando x é igual a 5 e y é igual a 3 então z pode ser {1, 2, 3, 6 }.
Quando x é igual a 5 e y é igual a 4 então z pode ser {1, 2, 4, 6 }.
Quando x é igual a 5 e y é igual a 5 então z pode ser {2, 3, 4 }.
Quando x é igual a 6 e y é igual a 3 então z pode ser {1, 2, 3, 5 }.
Quando x é igual a 6 e y é igual a 4 então z pode ser {1, 2, 4, 5 }.
Quando x é igual a 6 e y é igual a 6 então z pode ser {2, 3, 4 }.

Sendo 68 maneiras de escrever 3-0-0

No segundo caso que o jogo é composto de um bloco com cinco peças, um bloco com quatro peças e um bloco com três peças (2-1-0) ou 01-1x-xy-y2-20 - 03-3z-z4-40 - 05-56-60
Quando x é igual a 1 e y é igual a 2 então z pode ser {1, 2, 3, 4, 5, 6 }.
Quando x é igual a 1 e y é igual a 3 então z pode ser {3, 4, 5, 6 }.
Quando x é igual a 1 e y é igual a 4 então z pode ser {3, 4, 5, 6 }.
Quando x é igual a 1 e y é igual a 5 então z pode ser {1, 2, 3, 4, 5, 6 }.

Quando x é igual a 1 e y é igual a 6 então z pode ser {1, 2, 3, 4, 5, 6 }.
Quando x é igual a 3 e y é igual a 2 então z pode ser {3, 4, 5, 6 }.
Quando x é igual a 3 e y é igual a 3 então z pode ser {4, 5, 6 }.
Quando x é igual a 3 e y é igual a 4 então z pode ser {5, 6 }.
Quando x é igual a 3 e y é igual a 5 então z pode ser {2, 3, 4, 6 }.
Quando x é igual a 3 e y é igual a 6 então z pode ser {2, 3, 4, 5 }.
Quando x é igual a 4 e y é igual a 2 então z pode ser {3, 4, 5, 6 }.
Quando x é igual a 4 e y é igual a 3 então z pode ser {5, 6 }.
Quando x é igual a 4 e y é igual a 4 então z pode ser {3, 5, 6 }.
Quando x é igual a 4 e y é igual a 5 então z pode ser {2, 3, 4, 6 }.
Quando x é igual a 4 e y é igual a 6 então z pode ser {2, 3, 4, 5 }.
Quando x é igual a 5 e y é igual a 2 então z pode ser {1, 2, 3, 4, 5, 6 }.
Quando x é igual a 5 e y é igual a 3 então z pode ser {1, 3, 4, 6 }.
Quando x é igual a 5 e y é igual a 4 então z pode ser {1, 3, 4, 6 }.
Quando x é igual a 5 e y é igual a 5 então z pode ser {1, 2, 3, 4, 5, 6 }.
Quando x é igual a 6 e y é igual a 2 então z pode ser {1, 2, 3, 4, 5, 6 }.
Quando x é igual a 6 e y é igual a 3 então z pode ser {1, 3, 4, 5 }.
Quando x é igual a 6 e y é igual a 4 então z pode ser {1, 3, 4, 5 }.
Quando x é igual a 6 e y é igual a 6 então z pode ser {1, 2, 3, 4, 5, 6 }.

Sendo 100 maneiras de escrever 2-1-0

No terceiro caso que o jogo é composto por três blocos com quatro peças (1-1-1) ou 01-1x-x2-20 - 03-3y-y4-40 - 05-5z-z6-60
Quando x é igual a 1 e y é igual a 1 então z pode ser {1, 2, 3, 4, 5, 6 }.
Quando x é igual a 1 e y é igual a 2 então z pode ser {1, 2, 3, 4, 5, 6 }.
Quando x é igual a 1 e y é igual a 3 então z pode ser {1, 2, 3, 4, 5, 6 }.
Quando x é igual a 1 e y é igual a 4 então z pode ser {1, 2, 3, 4, 5, 6 }.
Quando x é igual a 1 e y é igual a 5 então z pode ser {1, 2, 5, 6 }.
Quando x é igual a 1 e y é igual a 6 então z pode ser {1, 2, 5, 6 }.
Quando x é igual a 2 e y é igual a 1 então z pode ser {1, 2, 3, 4, 5,6 }.
Quando x é igual a 2 e y é igual a 2 então z pode ser {1, 2, 3, 4, 5, 6 }.
Quando x é igual a 2 e y é igual a 3 então z pode ser {1, 2, 3, 4, 5, 6 }.
Quando x é igual a 2 e y é igual a 4 então z pode ser {1, 2, 3, 4, 5, 6 }.
Quando x é igual a 2 e y é igual a 5 então z pode ser {1, 2, 5, 6 }.
Quando x é igual a 2 e y é igual a 6 então z pode ser {1, 2, 5, 6 }.
Quando x é igual a 3 e y é igual a 3 então z pode ser {1, 2, 3, 4, 5, 6 }.

Quando x é igual a 3 e y é igual a 4 então z pode ser {1, 2, 3, 4, 5, 6 }.
Quando x é igual a 3 e y é igual a 5 então z pode ser {1, 2, 5, 6 }.
Quando x é igual a 3 e y é igual a 6 então z pode ser {1, 2, 5, 6 }.
Quando x é igual a 4 e y é igual a 3 então z pode ser {1, 2, 3, 4, 5, 6 }.
Quando x é igual a 4 e y é igual a 4 então z pode ser {1, 2, 3, 4, 5, 6 }.
Quando x é igual a 4 e y é igual a 5 então z pode ser {1, 2, 5, 6 }.
Quando x é igual a 4 e y é igual a 6 então z pode ser {1, 2, 5, 6 }.
Quando x é igual a 5 e y é igual a 1 então z pode ser {3, 4, 5, 6 }.
Quando x é igual a 5 e y é igual a 2 então z pode ser {3, 4, 5, 6 }.
Quando x é igual a 5 e y é igual a 3 então z pode ser {3, 4, 5, 6 }.
Quando x é igual a 5 e y é igual a 4 então z pode ser {3, 4, 5, 6 }.
Quando x é igual a 5 e y é igual a 5 então z pode ser {5, 6 }.
Quando x é igual a 5 e y é igual a 6 então z pode ser {5, 6 }.
Quando x é igual a 6 e y é igual a 1 então z pode ser {3, 4, 5, 6 }.
Quando x é igual a 6 e y é igual a 2 então z pode ser {3, 4, 5, 6 }.
Quando x é igual a 6 e y é igual a 3 então z pode ser {3, 4, 5, 6 }.
Quando x é igual a 6 e y é igual a 4 então z pode ser {3, 4, 5, 6 }.
Quando x é igual a 6 e y é igual a 5 então z pode ser {5, 6 }.
Quando x é igual a 6 e y é igual a 6 então z pode ser {5, 6 }.

Sendo 144 maneiras de escrever 1-1-1

No total então existem $7! \times 2 \times ((3 \times 68) + (6 \times 100) + (1 \times 144))) = 9555840$ possibilidades de jogos distintos com 13 peças onde os números 3 6 e 1 são os fatores multiplicantes, vistos no Lema 3.3, que correspondem as permutações dos blocos.

3.2.5 Qual o total de verificações que devem ser executadas?

Temos que calcular o número de jogos de 10 peças até 25 peças, e para cada quantidade de peças existem várias maneiras de construir os blocos. Para calcular a quantidade total de jogos trancados distintos, é necessário calcular quantas possibilidades existem para cada combinação de blocos. Usando o lema 3.2 que diz que a quantidade de jogos de x-y-z é igual a quantidade de jogos de todas suas permutações. Podemos definir a multiplicidade como a quantidade de permutações distintas dos blocos. Quando os três blocos tem a mesma quantidade de peças a multiplicidade é 1. Quando dois blocos contem quantidades iguais e um diferente a multiplicidade é 3. Quando os três blocos tem quantidades distintas de peças então a multiplicidade é 6.

Segue abaixo a lista das permutações que devem ser calculadas.

10 Peças

0-0-0

11 Peças

1-0-0

12 Peças

2-0-0, 1-1-0

13 Peças

3-0-0, 2-1-0, 1-1-1

14 Peças

4-0-0, 3-1-0, 2-2-0, 2-1-1

15 Peças

5-0-0, 4-1-0, 3-2-0, 3-1-1, 2-2-1

16 Peças

6-0-0, 5-1-0, 4-2-0, 4-1-1, 3-2-1, 3-3-0, 2-2-2

17 Peças

7-0-0, 6-1-0, 5-2-0, 5-1-1, 4-3-0, 4-2-1, 3-3-1, 3-2-2

18 Peças

8-0-0, 7-1-0, 6-2-0, 6-1-1, 5-3-0, 5-2-1, 4-4-0, 4-3-1, 4-2-2, 3-3-2

19 Peças

9-0-0, 8-1-0, 7-2-0, 7-1-1, 6-3-0, 6-2-1, 5-4-0, 5-3-1, 5-2-2, 4-4-1, 4-3-2, 3-3-3

20 Peças

10-0-0, 9-1-0, 8-2-0, 8-1-1, 7-3-0, 7-2-1, 6-4-0, 6-3-1, 6-2-2, 5-5-0, 5-4-1, 5-3-2, 4-4-2, 4-3-3

21 Peças

11-0-0, 10-1-0, 9-2-0, 9-1-1, 8-3-0, 8-2-1, 7-4-0, 7-3-1, 7-2-2, 6-5-0, 6-4-1, 6-3-2, 5-5-1, 5-4-2, 5-3-3, 4-4-3

22 Peças

12-0-0, 11-1-0, 10-2-0, 10-1-1, 9-3-0, 9-2-1, 8-4-0, 8-3-1 ,8-2-2, 7-5-0, 7-4-1, 7-3-2, 6-6-0, 6-5-1, 6-4-2, 6-3-3, 5-5-2, 5-4-3, 4-4-4

23 Peças

13-0-0, 12-1-0, 11-2-0, 11-1-1, 10-3-0, 10-2-1, 9-4-0, 9-3-1, 9-2-2, 8-5-0, 8-4-1, 8-3-2, 7-6-0, 7-5-1, 7-4-2, 7-3-3, 6-6-1, 6-5-2, 6-4-3, 5-5-3, 5-4-4

24 Peças

14-0-0, 13-1-0, 12-2-0, 12-1-1, 11-3-0, 11-2-1, 10-4-0, 10-3-1, 10-2-2, 9-5-0, 9-4-1, 9-3-2, 8-6-0, 8-5-1, 8-4-2, 8-3-3, 7-7-0, 7-6-1, 7-5-2, 7-4-3, 6-6-2, 6-5-3, 6-4-4, 5-5-4

25 Peças
15-0-0, 14-1-0, 13-2-0, 13-1-1, 12-3-0, 12-2-1, 11-4-0, 11-3-1, 11-2-2, 10-5-0, 10-4-1, 10-3-2, 9-6-0, 9-5-1, 9-4-2, 9-3-3, 8-7-0, 8-6-1, 8-5-2, 8-4-3, 7-7-1, 7-6-2, 7-5-3, 7-4-4, 6-6-3, 6-5-4, 5-5-5

Totalizando 6 de multiplicidade 1, 66 de multiplicidade 3 e 102 de multiplicidade 6. Ou seja, 174 verificações separadas para conseguir o total de trancas possíveis.

A cada pedra adicionada, o número de restrições aumentou bastante resultando num problema de combinatória com restrições que não soubemos resolver. Por isso, a partir da peça 14 tivemos que optar por solução computacional.

3.3 Parte Programacional

A cada pedra adicionada a quantidade de possibilidades aumentava muito, ou seja, listá-las a mão seria inviável, então foi quando decidimos usar programação para calcular estes resultados.
Restou o problema da escolha da linguagem de programação a ser utilizada. Após algumas pesquisas no Google descobri que uma das linguagens mais fáceis de se aprender é PHP.
Decidida a linguagem passei a estudar PHP. Alguns comandos do PHP são simples e muito importantes. O comando FOR que cria um laço, repetindo algo quantas vezes for desejado.
O comando IF que faz alguma coisa, caso o que esteja no teste seja verdadeiro.
O comando ELSE que pode ser usado após o IF fazendo outra coisa caso o IF seja falso.

Então tentando construir o código que verifica 4-0-0 o seguinte esquema foi criado:
1ª parte do código
Contar de 1111 até 6666. (usando apenas 1, 2, 3, 4, 5, 6)
2ª parte do código
Criar peças
3ª parte do código
Comparar as peças para que não hajam peças repetidas
4ª parte do código
Caso não existam peças repetidas, imprime a sequencia de peças e conta a quantidade de jogos até o momento

3.3.1 PHP

Na 1ª Parte do código a seguinte estrutura
faça $x = 1$, enquanto $x < 7$
faça $y = 1$, enquanto $y < 7$
faca $z = 1$, enquanto $z < 7$
faca $w = 1$, enquanto $w < 7$
(código a ser executado)
final dos comandos

Utilizando apenas o comando FOR foi feito a primeira parte do código, na qual são executados comandos com as variáveis x, y, z e w variando de 1 a 6.

Na 2ª Parte do código precisava criar as peças a serem comparadas, o seguinte problema foi encontrado, a peça 12 e a peça 21 são a mesma peça, como verificar se existem peças iguais sendo que nem as peças iguais é possível verificar. Esse problema foi resolvido colocando a representação decimal da pedra com a dezena maior ou igual do que a unidade.
Ficando o código assim:
peça1 é $(10 \times x) + 1$.
se $x < y$ então peça2 é $(10 \times y) + x$ caso contrário peça2 é $(10 \times x) + y$.
se $y < z$ então peça3 é $(10 \times z) + y$ caso contrário peça3 é $(10 \times y) + z$.
se $z < w$ então peça4 é $(10 \times w) + z$ caso contrário peça4 é $(10 \times z) + w$.
se $w < 2$ então peça5 é $(20) + w$ caso contrário peça5 é $(10 \times w) + 2$.
peca6 = 43.
peca7 = 65.

Criadas as peças vamos compará-las. Na 3ª parte do código faça $xx = 1$, Enquanto $xx <$ total de pecas somada de um.
faça $yy = 1$, enquanto $yy <$ total de peças somada de um. Se xx é diferente de yy então verifica se a peça(xx) é igual a peça(yy), se sim erro=1.
Na 4ª parte vem a parte fácil se erro diferente de 1 então imprime:
00-01-1x-xy-yz-zw-w2-20-03-34-40-05-56-60
define erro=0
e aqui fechamos todos os laços.

Em PHP fica assim:

```
<?
for( $x = 1 ; $x < 7 ; $x++){
for( $y = 1 ; $y < 7 ; $y++){
for( $z = 1 ; $z < 7 ; $z++){
for( $w = 1 ; $w < 7 ; $w++){
$peca1 = 10 + $x ;
if( $x < $y ){ $peca2 = $y \times 10 + $x ; }
else { $peca2 = $x \times 10 + $y ; }
if( $y< $z ){ $peca3 = $z \times 10 + $y ; }
else { $peca3 = $y \times 10 + $z ; }
if( $z < $w ){ $peca4 = $w \times 10 + $z ; }
else { $peca4 = $z \times 10 + $w ; }
if( $w < 2 ){ $peca5 = 20 + $w ; }
else { $peca5 = 10 \times $w + 2 ; }
$peca6 = 43;
$peca7 = 65;
total_de_pecas_somado_de_um = 8;
for($xx=1;$xx<$total_de_pecas_somado_de_um;$xx++){
for($yy=1;$yy<$total_de_pecas_somado_de_um;$yy++){
if($xx != $yy){
if(${'peca'.$xx} == ${'peca'.$yy}){ $erro = 1;}}}}
if($erro != 1){
$cont = $cont+1;
echo $cont.
"00-01-1".$x."-".$x.$y"-".$y.$z-".$z-$w-"-".$w-"2-20-03-34-40-05-56-60";
erro=0;
}}}}
?>
```

Esse código imprime o seguinte resultado:

1 - 01-11-12-23-35-52-20-03-34-40-05-60

2 - 01-11-12-23-36-62-20-03-34-40-05-60

3 - 01-11-12-24-45-52-20-03-34-40-05-60

...

262 - 01-16-66-64-42-22-20-03-34-40-05-60

263 - 01-16-66-64-44-42-20-03-34-40-05-60

264 - 01-16-66-64-45-52-20-03-34-40-05-60

Ou seja existem 264 maneiras de escrever 4-0-0.
Esse código calculou 1x-xy-yz-zw-w2-34-56 , mas pelo que vimos ainda falta calcular 3-1-0, 2-1-1e 2-2-0. Fazendo pequenas alterações no código obtemos
Para 3-1-0
1 - 01-11-13-32-22-20-3-33-34-40-5-60
2 - 01-11-13-32-22-20-3-34-44-40-5-60
3 - 01-11-13-32-22-20-3-35-54-40-5-60
...
310 - 01-16-66-64-42-20-3-33-34-40-5-60
311 - 01-16-66-64-42-20-3-34-44-40-5-60
312 - 01-16-66-64-42-20-3-35-54-40-5-60
Existem 312 possibilidades de escrever 3-1-0 combinações de dominó.
Para 2-1-1
1 - 01-11-12-22-20-3-31-14-40-5-51-60
2 - 01-11-12-22-20-3-31-14-40-5-52-60
3 - 01-11-12-22-20-3-31-14-40-5-53-60
...
398 - 01-16-66-62-20-3-34-44-40-5-55-60
399 - 01-16-66-62-20-3-35-54-40-5-55-60
400 - 01-16-66-62-20-3-36-64-40-5-55-60
E para 2-2-0
1 - 01-11-12-22-20-3-31-14-44-40-5-60
2 - 01-11-12-22-20-3-31-15-54-40-5-60
3 - 01-11-12-22-20-3-31-16-64-40-5-60
...
295 - 01-16-66-62-20-3-35-54-44-40-5-60
296 - 01-16-66-62-20-3-35-55-54-40-5-60
297 - 01-16-66-62-20-3-36-64-44-40-5-60

Então calculando 14 peças obtemos $3 \times (297 + 400 + 264) \times 2 \times 7! + (6 \times 312) \times 2 \times 7! = 4755 \times 2 \times 7! = 47930400$ maneiras de fechar o jogo com 14 peças.

Utilizando o mesmo código foi criando uma tabela sendo a primeira coluna o fator multi-

plicativo, a segunda coluna a combinação a ser executada no código na terceira coluna o valor obtido pelo código e na quarta coluna a multiplicação do fator multiplicativo pelo número obtido no código.

Tabela obtida a partir de códigos PHP				
Fator Multiplicativo	15 Peças	Possibilidades		Multiplicado
3	5 0 0	1048		3144
6	4 1 0	1222		7332
6	3 2 0	944		5664
3	3 1 1	1296		3888
3	2 2 1	1094		3282
Fator Multiplicativo	16 Peças	Possibilidades		Multiplicado
3	6 0 0	3556		10668
6	5 1 0	4584		27504
6	4 2 0	3516		21096
3	4 1 1	5064		15192
6	3 2 1	3608		21648
3	3 3 0	2900		8700
1	2 2 2	2983		2983
Fator Multiplicativo	17 Peças	Possibilidades		Multiplicado
3	7 0 0	9968		29904
6	6 1 0	13792		82752
6	5 2 0	11512		69072
3	5 1 1	17072		51216
6	4 3 0	9100		54600
6	4 2 1	12578		75468
3	3 3 1	10280		30840
3	3 2 2	9188		27564
Fator Multiplicativo	18 Peças	Possibilidades		Multiplicado
3	8 0 0	24216		72648
6	7 1 0	34124		204744
6	6 2 0	29932		179592
3	6 1 1	45288		135864
6	5 3 0	24628		147768
6	5 2 1	35808		214848
3	4 4 0	23730		71190
6	4 3 1	29124		174744
3	4 2 2	27204		81612
3	3 3 2	22796		68388

O código estava demorando mais de 2 minutos por execução nesse ponto e eu necessitava de quase 10 minutos para fazer as alterações no código que se aplicassem as outras situações, ou seja, consumia um grande tempo essa programação usando o PHP. O tempo se multiplicava por 6 cada vez que adicionava uma variável, então precisaria de meses para conseguir calcular até as 25 peças.

Não contente com os resultados, e pesquisando um pouco sobre o assunto, existiam três coisas que poderiam estar erradas.

1) O computador que está executando código é lento. Mas meu computador é bem rápido ou seja esse não era esse o problema.

2) PHP é uma linguagem lenta, talvez Matlab seja a solução.

3) O código está mal escrito, necessitaria ser otimizado para que o computador executasse menos cálculos e demorasse menos tempo.

Passei então a estudar e escrever os códigos na linguagem do MATLAB.

3.3.2 Matlab

A estrutura dos códigos em Matlab são mais limpas e mais práticas.

Poucas dificuldades foram encontradas para transcrever o código para a nova linguagem.

Em Matlab o código que pede para computador calcular o mesmo 4-0-0 calculado a pouco fica assim:

```
clear
erro=0;
p=0 ;
tic
for a=1:6
for b=1:6
for c=1:6
for g=1:6
x=3; p(x)=10+a;if(1<a) peca(x)= a*10+1; else peca(x)=10+a; end
x=x+1; p(x)=a*10+b;if(a<b) peca(x)= b*10+a; else peca(x)=a*10+b; end
x=x+1; p(x)=b*10+c;if(b<c) peca(x)= c*10+b; else peca(x)=b*10+c; end
x=x+1; p(x)=c*10+g;if(c<g) peca(x)= g*10+c; else peca(x)=c*10+g; end
x=x+1; p(x)=g*10+2;if(2<g) peca(x)= g*10+2; else peca(x)=2*10+g; end
x=x+1; p(x)=43; peca(x)=43;
x=x+1; p(x)=65; peca(x)=65;
for pp=3:x
for qq=3:x
```

```
18 if(pp ~= qq)
   if(peca(pp)==peca(qq)) erro=1;
20 end
   end
22 end
   end
24 if(erro ~= 1)
   p(1)=p(1)+1;
26 p
   end
28 erro=0;
   end
30 end
   end
32 end
   toc
```

O código em PHP e em Matlab não apresentaram grandes diferenças no tempo de execução, mas só foi verificado isso quando tentamos verificar os jogos com 19 peças.
Surgiu então a possibilidade de usar JAVA, que é uma linguagem mais rápida e mais complexa de se aprender.

3.3.3 JAVA

Tendo um código já pronto em PHP e MATLAB, reescrevê-lo em JAVA não foi tão difícil e para minha surpresa o JAVA conseguiu executar o código até as 20 peças sem problemas. Foi no jogo de 21 peças que percebi que o JAVA era a linguagem certa, mas que eu deveria otimizar meu código facilitando o processamento do computador.

3.3.4 Otimizando o Código

Verificação de peças, quando tentei interpretar o que o computador estava fazendo percebi que ele estava comparando todas as peças entre si, mas era um pouco mais do que eu pensava. Na primeira bateria de testes ele verifica se a peça 1 é diferente das peças 2,3,4,...,N. Na segunda bateria de testes ele verifica se a peça 2 é diferente das peças 1,3,4,5,6,...,N. Na terceira bateria de testes ele verifica se a peça 3 é diferente das peças 1,2,4,5,6,....,N. Se prestar bem atenção, o

código verifica se a peça 1 é diferente da peça 2 duas vezes, e o mesmo acontece para todas as peças. A solução para esse problema é a peça a ser verificada só é verificada com peças maiores do que ela. Ou seja a peça 1 será verificada com todas as outras, e a peça dois será verificadas com todas as outras exceto a peça 1, e assim por diante até que a peça N-1 seja verificada apenas com a peça N. Evitando, assim, metade das verificações comparado ao código anterior.
Segue abaixo uma tabela de como era e de como passou a ser feita a comparação dos números.

Antes							
1	2	3	4	5	6	...	N
1	2	3	4	5	6	...	N
1	2	3	4	5	6	...	N
1	2	3	4	5	6	...	N
1	2	3	4	5	6	...	N
1	2	3	4	5	6	...	N
1	2	3	4	5	6	...	N

Depois							
1	2	3	4	5	6	...	N
	2	3	4	5	6	...	N
		3	4	5	6	...	N
			4	5	6	...	N
				5	6	...	N
					6	...	N
						...	N

Sendo os números vermelhos comparado aos números pretos da mesma linha.

Nessa mesma parte do código se o código achar peças repetidas ele continua verificando. Para arrumar isso tive de aprender o comando BREAK, que simplesmente para o laço FOR em que se encontra, ou seja, quando o erro é igual a um então o comando BREAK é executado fazendo com que o código pare de verificar, pois já encontrou peça repetida. Apenas com essas duas alterações no código o desempenho do código melhorou perto de 60%. Um código que demoraria 5 minutos para ser executado, agora é executado em menos de 2 minutos. Quanto maior o código maior a economia dos cálculos.

Na parte dos laços FOR, que são responsáveis por fazer as variáveis permutarem de um a seis a evolução foi pequena, porém, muito eficiente na alteração de um modelo para o outro. Anteriormente cada laço de FOR usava uma letra para definir sua variável, com a mudança de

linguagem para o Matlab tive que usar vetores, ou seja, uma única letra contém as N entradas que fazem os comandos FOR funcionarem.
Para evitar fazer cálculos desnecessários, observei que as seguintes peças xy-yz ambas contem o naipe y, então x não pode ser igual a z, pois isso acarreta em duas peças iguais. Resolvi o problema fazendo com que o código verifique se a variável N é igual a variável N-2, ou seja, eliminando esse tipo de verificações. Essa alteração apesar de pequena produziu um ganho de tempo computacional significativo. Além disso, quanto maior o número de peças melhor esse ajuste operou.

Na parte da criação de peças, o que evoluiu foram as rotinas que geravam as peças. No primeiro projeto, elas eram totalmente escritas, ou seja, era definida peça1 igual a alguma coisa. No segundo modelo de projeto, a numeração das peças passou a ser automática, restando apenas a definição de uma variável atribuída à quantidade de peças do jogo.

Certamente esse código não era o mais eficiente nem o mais otimizado, mas era bom o suficiente para terminar os cálculos.

3.3.5 Versão Final do Programa

O código final em JAVA ficou:

```
int erro=0; int count=0; int xx=1; int zz=1;
int yy=1; int pt=18;
int[] v={1,7,7,7,7,7,7,7,7,7,7,2,0,3,4,0,5,6};
//0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5
int[] p={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22};
int[] pc={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22};
//faz os números rodarem :)
for(v[1]=1;v[1]<7;v[1]++){
for(v[2]=1;v[2]<7;v[2]++){
for(v[3]=1;v[3]<7;v[3]++){ if(v[0]==v[2]){ break; }
for(v[4]=1;v[4]<7;v[4]++){ if(v[1]==v[3]){ break; }
for(v[5]=1;v[5]<7;v[5]++){ if(v[2]==v[4]){ break; }
for(v[6]=1;v[6]<7;v[6]++){ if(v[3]==v[5]){ break; }
for(v[7]=1;v[7]<7;v[7]++){ if(v[4]==v[6]){ break;}
for(v[8]=1;v[8]<7;v[8]++){ if(v[5]==v[7]){ break;}
for(v[9]=1;v[9]<7;v[9]++){ if(v[6]==v[8]){ break;}
for(v[10]=1;v[10]<7;v[10]++){ if(v[7]==v[9]){ break; }
```

```
for(v[11]=1;v[11]<7;v[11]++){ if(v[8]==v[10]){ break; }
//for(v[12]=1;v[12]<7;v[12]++){ if(v[9]==v[11]){ break; }
//for(v[13]=1;v[13]<7;v[13]++){ if(v[10]==v[12]){ break; }
//for(v[14]=1;v[14]<7;v[14]++){ if(v[11]==v[13]){ break; }
//for(v[15]=1;v[15]<7;v[15]++){ if(v[12]==v[14]){ break; }
//for(v[16]=1;v[16]<7;v[16]++){ if(v[13]==v[15]){ break; }
//for(v[17]=1;v[17]<7;v[17]++){ if(v[14]==v[16]){ break; }
//for(v[18]=1;v[18]<7;v[18]++){ if(v[15]==v[17]){ break; }
//for(v[19]=1;v[19]<7;v[19]++){ if(v[16]==v[18]){ break; }
//for(v[20]=1;v[20]<7;v[20]++){ if(v[17]==v[19]){ break; }
//for(v[21]=1;v[21]<7;v[21]++){ if(v[18]==v[20]){ break; }
// criação das pecas
for(xx=0;xx<pt;xx++) {
pc[xx]=v[xx]*10+v[xx+1];
p[xx]=pc[xx];
if(v[xx]<v[xx+1]){ p[xx]=v[xx+1]*10+v[xx]; }
}
for(yy=0;yy<pt-1;yy++){ //comparação de pecas
for(zz=yy+1;zz<pt;zz++){
if(p[yy]==p[zz] ){
erro=1; break;
}} if(erro == 1){break;}} // termina verificação das peças
if(erro != 1){
count++;System.out.println(count+" - 01-"+pc[0]+"-"+pc[1]+"-"+pc[2]+"-"+
+pc[3]+"-"+pc[4]+"-"+pc[5]+"-"+pc[6]+"-"+pc[7]+"-"+pc[8]+"-"+
+pc[9]+"-"+pc[10]+"-"+pc[11]+"-"+pc[12]+"-"+pc[13]+"-"+
+pc[14]+"-"+pc[15]+"-"+pc[16]+"-"+pc[17]+"-"+pc[18]+"-"+pc[19]+"-"+
+pc[20]+"-"+pc[21]+"-60");}
erro = 0; }
}}}}}}}}}
System.out.println(count); // imprime valor total
```

Agora vamos entender exatamente o que o código faz.

Em JAVA é necessário declarar as variáveis a serem usadas.

```
"int erro=0; int count=0; int xx=1; int zz=1;"\\
```

Essa linha declara as variáveis erro, count, xx e yy sendo todas números inteiros.

```
"int yy=1; int pt=18;"\\
```

Aqui é declarado mais dois valores inteiros yy e pt

```
int[] v={1,7,7,7,7,7,7,7,7,7,7,2,0,3,4,0,5,6};
//0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5
```

é declarado uma matriz **v** com 19 colunas, e a segunda linha é apenas um comentário que nos ajuda a encontrar a posição das colunas.

```
int[] p={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22};
```

aqui é declarado o vetor **p** com 26 colunas, que será usado para criar as peças com a dezena maior ou igual a unidade.

```
int[] pc={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22};
```

aqui é declarado o vetor **pc** com 26 colunas, que será usado para saber qual é a posição certa da peça.

Aqui entram vários comandos FOR enlaçados, eles são responsáveis por fazer todas as permutações das variáveis.

```
for(v[1]=1;v[1]<7;v[1]++){
for(v[2]=1;v[2]<7;v[2]++){
for(v[3]=1;v[3]<7;v[3]++){ if(v[0]==v[2]){ break; }
for(v[4]=1;v[4]<7;v[4]++){ if(v[1]==v[3]){ break; }
for(v[5]=1;v[5]<7;v[5]++){ if(v[2]==v[4]){ break; }
for(v[6]=1;v[6]<7;v[6]++){ if(v[3]==v[5]){ break; }
for(v[7]=1;v[7]<7;v[7]++){ if(v[4]==v[6]){ break;}
for(v[8]=1;v[8]<7;v[8]++){ if(v[5]==v[7]){ break;}
for(v[9]=1;v[9]<7;v[9]++){ if(v[6]==v[8]){ break;}
for(v[10]=1;v[10]<7;v[10]++){ if(v[7]==v[9]){ break; }
```

```
for(v[11]=1;v[11]<7;v[11]++){ if(v[8]==v[10]){ break; }
```

Note que até aqui o código tem os comandos FOR abertos, ou seja eles estão sendo executados. A partir desse ponto todos os FOR vem com a barra dupla na frente, e isso em JAVA significa que é apenas um comentário.

```
//for(v[12]=1;v[12]<7;v[12]++){ if(v[9]==v[11]){ break; }
//for(v[13]=1;v[13]<7;v[13]++){ if(v[10]==v[12]){ break; }
//for(v[14]=1;v[14]<7;v[14]++){ if(v[11]==v[13]){ break; }
//for(v[15]=1;v[15]<7;v[15]++){ if(v[12]==v[14]){ break; }
//for(v[16]=1;v[16]<7;v[16]++){ if(v[13]==v[15]){ break; }
//for(v[17]=1;v[17]<7;v[17]++){ if(v[14]==v[16]){ break; }
//for(v[18]=1;v[18]<7;v[18]++){ if(v[15]==v[17]){ break; }
//for(v[19]=1;v[19]<7;v[19]++){ if(v[16]==v[18]){ break; }
//for(v[20]=1;v[20]<7;v[20]++){ if(v[17]==v[19]){ break; }
//for(v[21]=1;v[21]<7;v[21]++){ if(v[18]==v[20]){ break; }

```

Essa parte do código que nesse momento é comentário não é utilizada, mas como temos de testar várias combinações de blocos diferentes então elas ficam aqui para facilitar sua alteração. Note que na matriz **v** que foi criada no começo do código, os números dos comandos FOR que são executados são exatamente os quais o valor de v na matriz é igual a 7, caso contrário são considerados como comentários.

Essa parte do código só está inserida, pois ela facilita muito fazer alterações nos blocos, e o número 7 foi escolhido propositalmente, pois se algo de errado acontecer por falha humana, será impresso no resultado vários números 7, que não pertencem ao dominó, facilitando assim a correção dos famosos BUGs.

Aquele comando IF que está após o comando FOR, é o processo de otimização apresentado anteriormente, que evita peças do tipo XY-YX.

Finalmente chegamos a parte do código que é responsável por criar as peças e organizá-las de modo a dezena ser maior ou igual a unidade.

```
for(xx=0;xx<pt;xx++) {
pc[xx]=v[xx]*10+v[xx+1];
p[xx]=pc[xx];
if(v[xx]<v[xx+1]){ p[xx]=v[xx+1]*10+v[xx]; } }
```

Esse código é bem simples, o que evita ter que criar todas as peças manualmente, mas é muito importante lembrar-se de mudar a variável pt no caso de ser adicionada ou removida alguma peça.
Agora vem a parte que compara as peças e já se encontra otimizada como foi observado anteriormente.

```
for(yy=0;yy<pt-1;yy++){ //comparação de pecas
for(zz=yy+1;zz<pt;zz++){
if(p[yy]==p[zz] ){
erro=1; break;
}} if(erro == 1){break;}} // termina verificação das peças

```

Na próxima parte do código é verificado se existe alguma peça igual. Se não existirem peças iguais então é impresso na tela a sequência das peças, e também o jogo trancado.

```
if(erro != 1){
count++;System.out.println(count+" - 01-"+pc[0]+"-"+pc[1]+"-"+pc[2]+"-"+
+pc[3]+"-"+pc[4]+"-"+pc[5]+"-"+pc[6]+"-"+pc[7]+"-"+pc[8]+"-"+
+pc[9]+"-"+pc[10]+"-"+pc[11]+"-"+pc[12]+"-"+pc[13]+"-"+
+pc[14]+"-"+pc[15]+"-"+pc[16]+"-"+pc[17]+"-60");}
erro = 0; }
}}}}}}}}}
```

System.out.println(count); // imprime valor total, porém, se existem peças repetidas, ele simplesmente pula essa parte de código e redefine a variável erro igual a zero.
E a última linha do código imprime exatamente o número de jogadas possíveis com aqueles blocos escolhidos.

Esse código que foi explicado aqui é relativo ao cálculo do bloco 11-0-0, que imprime o seguinte resultado:
run:
1 - 01-11-12-22-23-31-14-42-25-51-16-66-62-20-3-34-40-5-56-60
2 - 01-11-12-22-23-31-14-42-25-53-33-36-62-20-3-34-40-5-56-60 3 - 01-11-12-22-23-31-14-42-25-53-36-66-62-20-3-34-40-5-56-60

...
261582 - 01-16-66-64-45-55-53-36-62-25-51-12-22-20-3-34-40-5-56-60
261583 - 01-16-66-64-45-55-53-36-62-25-51-13-32-20-3-34-40-5-56-60
261584 - 01-16-66-64-45-55-53-36-62-25-51-14-42-20-3-34-40-5-56-60
261584
BUILD SUCCESSFUL (total time: 49 seconds)

3.3.6 Alterando de 11-0-0 para 10-1-0

As seguintes partes do código sofreram alterações:

```
int[] v={1,7,7,7,7,7,7,7,7,7,7,2,0,3,4,0,5,6};
Passa a ser
int[] v={1,7,7,7,7,7,7,7,7,7,7,2,0,3,7,4,0,5,6};

for(v[11]=1;v[11]<7;v[11]++){ if(v[8]==v[10]){ break; }
Passa a ser comentário

//for(v[14]=1;v[14]<7;v[14]++){ if(v[11]==v[13]){ break; }
Deixa de ser comentário.
```

Feita essa alteração é só executar o código e conseguir o novo valor.

3.3.7 Solução do Problema

Fazendo as 143 verificações e multiplicando pelos devidos fatores obtemos a tabela seguinte com o resultado final de possibilidades distintas.

Fator Multiplicativo	10 Peças	Possibilidades	Poss Multiplicado
1	0 0 0	1	1
	11 Peças		
3	1 0 0	6	18
	12 Peças		0
3	2 0 0	21	63
3	1 1 0	32	96
	13 Peças		
3	3 0 0	68	204
6	2 1 0	100	600
1	1 1 1	144	144
	14 Peças		
3	4 0 0	264	792
6	3 1 0	312	1872
3	2 2 0	297	891
3	2 1 1	400	1200
	15 Peças		
3	5 0 0	1048	3144
6	4 1 0	1222	7332
6	3 2 0	944	5664
3	3 1 1	1296	3888
3	2 2 1	1094	3282
	16 Peças		
3	6 0 0	3556	10668
6	5 1 0	4584	27504
6	4 2 0	3516	21096
3	4 1 1	5064	15192
6	3 2 1	3608	21648
3	3 3 0	2900	8700
1	2 2 2	2983	2983

Fator Multiplicativo	17 Peças	Possibilidades	Poss Multiplicado
3	7 0 0	9968	29904
6	6 1 0	13792	82752
6	5 2 0	11512	69072
3	5 1 1	17072	51216
6	4 3 0	9100	54600
6	4 2 1	12578	75468
3	3 3 1	10280	30840
3	3 2 2	9188	27564
	18 Peças		
3	8 0 0	24216	72648
6	7 1 0	34124	204744
6	6 2 0	29932	179592
3	6 1 1	45288	135864
6	5 3 0	24628	147768
6	5 2 1	35808	214848
3	4 4 0	23730	71190
6	4 3 1	29124	174744
3	4 2 2	27204	81612
3	3 3 2	22796	68388
	19 Peças		
3	9 0 0	56168	168504
6	8 1 0	78152	468912
6	7 2 0	67724	406344
3	7 1 1	103456	310368
6	6 3 0	57548	345288
6	6 2 1	83792	502752
6	5 4 0	59088	354528
6	5 3 1	72368	434208
3	5 2 2	67000	201000
3	4 4 1	72864	218592
6	4 3 2	57688	346128
1	3 3 3	49248	49248

Fator Multiplicativo	20 Peças	Possibilidades	Poss Multiplicado
3	10 0 0	129088	387264
6	9 1 0	179320	1075920
6	8 2 0	152596	915576
3	8 1 1	232328	696984
6	7 3 0	127560	765360
6	7 2 1	183808	1102848
6	6 4 0	136012	816072
6	6 3 1	161852	971112
3	6 2 2	147572	442716
3	5 5 0	149440	448320
6	5 4 1	178372	1070232
6	5 3 2	137052	822312
3	4 4 2	141084	423252
3	4 3 3	118352	355056
	21 Peças		
3	11 0 0	261584	784752
6	10 1 0	376044	2256264
6	9 2 0	328836	1973016
3	9 1 1	504384	1513152
6	8 3 0	265896	1595376
6	8 2 1	396464	2378784
6	7 4 0	274456	1646736
6	7 3 1	334752	2008512
3	7 2 2	312096	936288
6	6 5 0	318496	1910976
6	6 4 1	377900	2267400
6	6 3 2	289032	1734192
3	5 5 1	418560	1255680
6	5 4 2	323740	1942440
3	5 3 3	264352	793056
3	4 4 3	264848	794544

Fator Multiplicativo	22 Peças	Possibilidades	Poss Multiplicado
3	12 0 0	406200	1218600
6	11 1 0	615984	3695904
6	10 2 0	566772	3400632
3	10 1 1	898640	2695920
6	9 3 0	455728	2734368
6	9 2 1	731456	4388736
6	8 4 0	442824	2656944
6	8 3 1	590680	3544080
3	8 2 2	576440	1729320
6	7 5 0	505024	3030144
6	7 4 1	632872	3797232
6	7 3 2	503608	3021648
3	6 6 0	548232	1644696
6	6 5 1	736336	4418016
6	6 4 2	570468	3422808
3	6 3 3	463232	1389696
3	5 5 2	620672	1862016
6	5 4 3	488496	2930976
1	4 4 4	467224	467224

Fator Multiplicativo	23 Peças	Possibilidades	Poss Multiplicado
3	13 0 0	516112	1548336
6	12 1 0	790160	4740960
6	11 2 0	733856	4403136
3	11 1 1	1198208	3594624
6	10 3 0	614696	3688176
6	10 2 1	1018488	6110928
6	9 4 0	585856	3515136
6	9 3 1	828480	4970880
3	9 2 2	820368	2461104
6	8 5 0	632848	3797088
6	8 4 1	838904	5033424
6	8 3 2	697368	4184208
6	7 6 0	684104	4104624
6	7 5 1	943136	5658816
6	7 4 2	747272	4483632
3	7 3 3	624640	1873920
3	6 6 1	1007808	3023424
6	6 5 2	839720	5038320
6	6 4 3	669952	4019712
3	5 5 3	705344	2116032
3	5 4 4	665680	1997040

Fator Multiplicativo	24 Peças	Possibilidades	Poss Multiplicado
3	14 0 0	905288	2715864
6	13 1 0	1241040	7446240
6	12 2 0	1056824	6340944
3	12 1 1	1602000	4806000
6	11 3 0	916128	5496768
6	11 2 1	1310832	7864992
6	10 4 0	955972	5735832
6	10 3 1	1157448	6944688
3	10 2 2	1065968	3197904
6	9 5 0	1022104	6132624
6	9 4 1	1225456	7352736
6	9 3 2	969896	5819376
6	8 6 0	1006492	6038952
6	8 5 1	1284384	7706304
6	8 4 2	1023164	6138984
3	8 3 3	878272	2634816
3	7 7 0	988912	2966736
6	7 6 1	1286744	7720464
6	7 5 2	1072152	6432912
6	7 4 3	924320	5545920
3	6 6 2	1083224	3249672
6	6 5 3	983216	5899296
3	6 4 4	986984	2960952
3	5 5 4	1041648	3124944

Fator Multiplicativo	25 Peças	Possibilidades	Poss Multiplicado
3	15 0 0	2124160	6372480
6	14 1 0	2828016	16968096
6	13 2 0	2387344	14324064
3	13 1 1	3404480	10213440
6	12 3 0	1980016	11880096
6	12 2 1	2607872	15647232
6	11 4 0	2130096	12780576
6	11 3 1	2607872	15647232
3	11 2 2	2059200	6177600
6	10 5 0	2372592	14235552
6	10 4 1	2631248	15787488
6	10 3 2	1953728	11722368
6	9 6 0	2324752	13948512
6	9 5 1	2808672	16852032
6	9 4 2	2181200	13087200
3	9 3 3	1805376	5416128
6	8 7 0	2164064	12984384
6	8 6 1	2648768	15892608
6	8 5 2	2222496	13334976
6	8 4 3	1910160	11460960
3	7 7 1	2523456	7570368
6	7 6 2	2100192	12601152
6	7 5 3	1969568	11817408
3	7 4 4	2042144	6126432
3	6 6 3	1966848	5900544
6	6 5 4	2233056	13398336
1	5 5 5	2432832	2432832
		somando tudo	619066356
	Solução do Problema	$7! \times 2 \times 619066356$	6240188868480

3.4 Problemas Extras

3.4.1 A quantidade de pontos na mesa de um jogo fechado é sempre PAR

Lema 3.4. *Em um jogo trancado a soma dos pontos na mesa é par, a soma dos pontos fora da mesa também é par.*

Demonstração: Um jogo trancado utiliza 8 peças do mesmo naipe. Seja n o naipe do fecha e x qualquer outro naipe. Serão jogadas $8 \times n$ peças do fecha e as outras peças serão ligadas aos pares. Então podemos escrever na forma $2 \times (4 \times n + x)$. Sendo múltiplo de dois e consecutivamente par. ■

Consequência: Se o total de pontos do jogo é par, e o jogo trancado é sempre par, então quando um jogo é trancado, ou ambas as duplas contém quantidades pares de pontos ou tem quantidades impares de pontos.

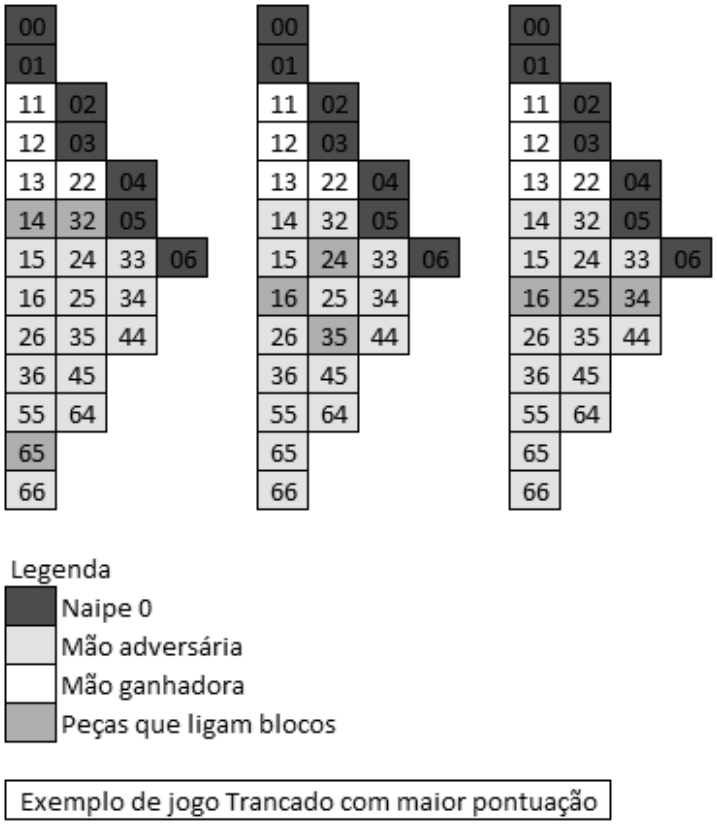
3.4.2 A maior quantidade de pontos que pode feito em uma partida

Cada peça de dominó representa um número de pontos, sendo esse número a soma dos naipes da peça, abaixo a tabela com a distribuição de pontos.

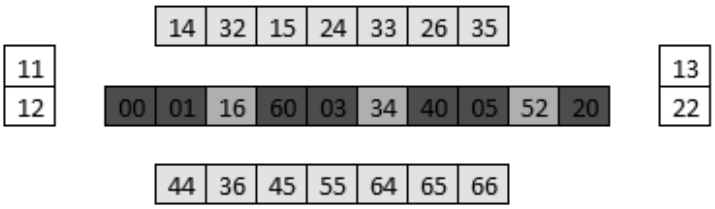
Pontos	Peça	Peça	Peça	Peça
0	00			
1	10			
2	20	11		
3	30	21		
4	40	31	22	
5	50	41	32	
6	60	51	42	33
7	61	52	43	
8	62	53	44	
9	63	54		
10	64	55		
11	65			
12	66			

Nosso objetivo é verificar qual a maior quantidade de pontos que um dupla pode receber quando o jogo é trancado. Para trancar o jogo são necessárias 10 peças, sendo 7 do mesmo

naipe. Como queremos que tenha a menor quantidade de pontos na mesa, para sobrar a maior quantidade de pontos possível nas mãos de uma das duplas usaremos o naipe do 0 para jogar. Note que a soma das peças do naipe zero é $1+2+3+4+5+6=21$ pontos. Como serão usados três 0 blocos, então precisamos de mais 21 pontos sendo as três peças que ligam um bloco ao outro. Um exemplo de 3 peças que podemos utilizar é a 21, 43, 65. Nesse exemplo estamos usando uma peça muito pequena no jogo que é a peça 21. Perceba que não importa quais sejam as três peças, a quantidade de pontos será 21. O que queremos é a maior quantidade de pontos na mão dos adversários. Vamos, por exemplo, utilizar todas as peças de peso 7, começando com a 61 depois com a 43 e por último com a 52. Considerando que as peças foram distribuídas de tal maneira que os adversários fossem obrigados a passar sua vez em todas as rodadas, eles terão 14 peças em suas mãos. Se existem 10 peças jogadas e 14 peças com os adversários, então as 4 outras peças que faltam serão escolhidas as menores que sobraram. Ficando o jogo trancado de maior pontuação assim: jogo na mesa 00-01-16-60-03-34-40-05-52-20. As peças com quem trancou o jogo: 11, 12, 13, 22 (são as 4 menores peças se removido o naipe 0). Peças na mão de quem não jogou nenhuma peça: 14, 32, 15, 24, 33, 26, 35, 44, 35, 45, 55, 46, 56, 66, totalizando 113 pontos. Nas imagens abaixo existem algumas das possíveis contagens diferentes, onde o importante é jogar o naipe zero em vermelho, e a equipe ganhadora possuir as 4 peças mais leves.



Exemplo de jogo Trancado com maior pontuação



3.4.3 A menor quantidade de pontos que pode ser feita

A maior quantidade de peças na mesa para que o jogo termine é 25, pois algum jogador bate ou tranca o jogo. Nesse caso os adversários estarão de posse de duas peças, se elas forem as duas menores do jogo como 00 e a 01, então é a menor pontuação possível durante um jogo.

4 Conclusão

Neste trabalho procuramos obter o número de jogos trancados de dominó em função do número de peças colocadas em jogo.

Inicialmente, tentamos obter isso através de uma fórmula usando os princípios de combinatória. Dada à quantidade de restrições, o problema se mostrou complexo em demasia. Consequentemente fomos levados obter esses números usando um algoritmo. Este algoritmo foi escrito em linguagem JAVA e além de calcular o número de jogos ele lista todos os jogos possíveis.

Para fazer esse trabalho foram necessários conceitos de analise combinatória e de linguagem de programação. Restou em aberto da obtenção da fórmula geral.

Referências Bibliográficas

[1] ANDRADE. A.A.S. Introdução à Álgebra Linear.

[2] HAMILTON. P. B. Funções de Matrizes, (Versão Preliminar).