Pedro Daniel da Silva Gohl
1201324420

Questão 1

O problema do *n-ésimo* número da sequência de *Fibonacci* é dado por:

$$fib(n) = \begin{cases} 1, & se\ n < 3 \\ fib(n-1) + fib(n-2), & se\ n \geq 3 \end{cases}$$

Função recursiva:
Complexidade O(ϕn).
Melhor caso e pior caso é 2n.

```
int fib( int n){
    if(n <= 2)
    return 1;
else
    return (fib (n -2) + fib(n - 1));
}
```

Função interativa:
Complexidade O(n).
Melhor caso e pior caso é n.

```
int fib( int n ){
    int a, b, c, d, i;
    if (n <= 2)
    return 1;

else{
    a = 0;
    b = 1;
    for( i = 3, i <= n, i++){
        c = a + b;
        a = b;
        b = c;
    }
    return c;
    }
}
```

Gerando todas as permutações de um determinado número é o arranjo de elementos distintos de um conjunto.
Se temos um conjunto { [5] , [4] , [3] }, então a permutação de todos os seus elementos resultará em novos arranjos:

{ [5] , [4] , [3] }, { [5] , [3] , [4] }, { [4] , [5] , [3] }, { [4] , [3] , [5] }, { [3] , [4] , [5] }, { [3] , [5] , [4] }

Questão 2

a)

$$T(\boldsymbol{n}) = \begin{cases} \boldsymbol{c},\ \boldsymbol{n} = \boldsymbol{1} \\ \boldsymbol{aT}\left(\frac{\boldsymbol{n}}{\boldsymbol{b}}\right) + \boldsymbol{c},\ \boldsymbol{n} > \boldsymbol{1} \end{cases}$$

$$T(n) = a\left[aT\left(\frac{n}{b^3}\right) + c\right] + ac + c = a^2T\left(\frac{n}{b^2}\right) + a^1c + a^0c$$
$$= a^2\left[aT\left(\frac{n}{b^3}\right) + c\right] + ac + c = a^3T\left(\frac{n}{b^3}\right) + a^2c + a^1c + a^0c$$

$$a^kT\left(\frac{n}{b^k}\right) + \sum_{i=0}^{k-1} a^ic$$

$$n = b^k, \qquad k = \log_b n$$

$$a^kT(1) + \sum_{i=0}^{k-1} a^ic$$

$$c\left(\frac{a^{(\log_b n)+1} - 1}{a - 1}\right)$$

b)

$$\boldsymbol{T(n)} = \begin{cases} \boldsymbol{\Theta(1)},\ \boldsymbol{n} = \boldsymbol{1} \\ \boldsymbol{2T}\left(\frac{\boldsymbol{n}}{\boldsymbol{2}}\right) + \boldsymbol{\Theta(n)},\ \boldsymbol{n} > \boldsymbol{1} \end{cases}$$

Método mestre:

$n^{\log_2 2} = n = \Theta(n)$

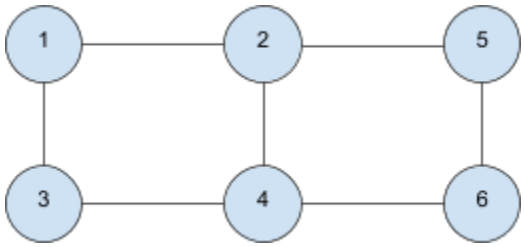
$T(n) = \Theta(n \log n)$

Questão 3

Grafo:

Um grafo simples G é um par ordenado G = (V,E), onde E é um conjunto de pares não ordenados de elementos de V, os elementos de V são chamados de vértices e os elementos E são chamados de aresta.

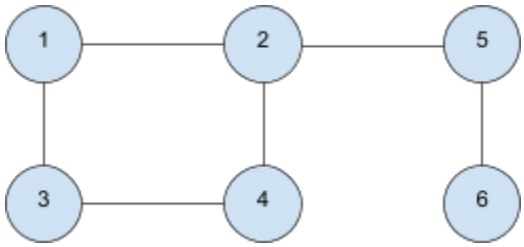
Supondo que as cidades são vértices, e as estradas para cada cidade são minhas arestas, e que um vértice se conecta com outro através dessas arestas, temos um grafo simples, ou seja, não direcionado.



Grafo Conexo:

Um grafo G=(V, E) é conexo se existir um caminho entre qualquer par de vértices. Caso Contrário é desconexo.

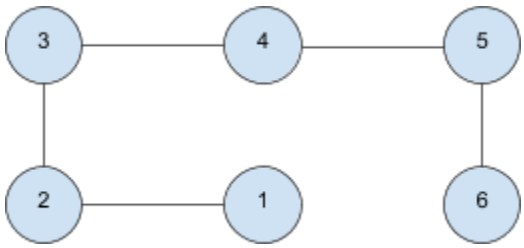
Supondo que duas ilhas tem seus distritos, e que esses distritos são os vértices, e que esses vértices são ligados por estradas, que são nossas arestas, supondo que tenha uma ponte que ligue essas duas ilhas, essas ilhas se tornarão um exemplo de grafo conexo.



Grafo Acíclico:

Um grafo G=(V, E) é acíclico se o mesmo não formar um “caminho” fechado, mais precisamente um ciclo sem vértices repetidos.

Supondo que uma linha de ônibus seja nosso grafo, onde suas paradas são os nossos vértices, e o caminho entre eles são nossas arestas.Esse caminho não leva de volta ao começo, então o ônibus sempre fará uma viagem só de ida. Esse é um exemplo de grafo acíclico.

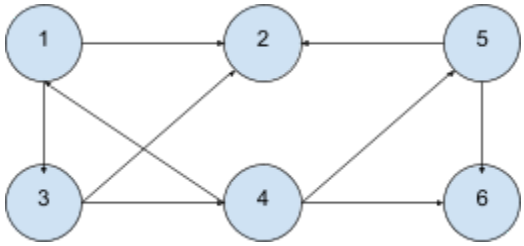


Grafo Direcionado:

Um grafo G=(V, E) é direcionado quando suas arestas possuim somente uma sentido para um vértice.

Supondo que uma linha de ônibus seja nosso grafo, onde suas paradas são os nossos

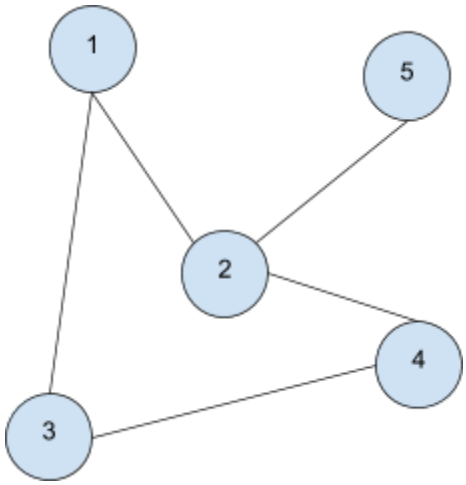
vértices, e o caminho entre eles são nossas arestas, mas cada um desses caminhos tem somente uma direção que leva a somente uma parada.



Grafo Planar

Grafo planar é um grafo que pode ser representado de forma que suas arestas não se cruzem.

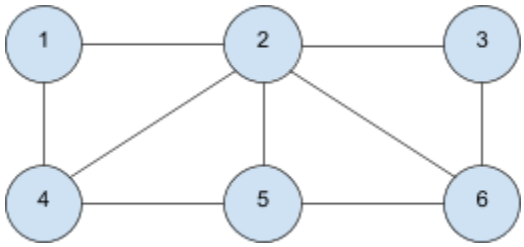
Supondo que temos uma linha de trem, que será nossas arestas, e os estações, que serão nossos vértices, as linhas de trem que normalmente podem passar uma por cima das outras, nesse caso não pode se sobrepor umas as outras, elas tem que chegar em cada estação sem se cruzarem.



Adjacência x Vizinhança

Um grafo G=(V,E), onde V é um vértice e E são suas arestas, e um vértice está ligado a outro, este vértice está adjacente e vizinho.

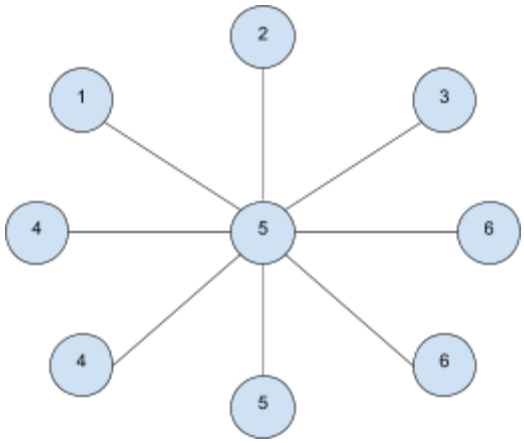
Supondo que tenhamos várias cidades, com várias estradas, onde nossas cidades são os vértices e as estradas as arestas, supondo que uma cidade seja interligada à outra, essas são vizinhas e adjacentes.



Grafo Bipartido, Completo

Um grafo bipartido, G := (V_1 + V_2, E), é um grafo bipartido tal que para quaisquer dois vértices, $v_1 \in V_1$ e $v_2 \in V_2$,v_1v_2 é uma aresta em G. O grafo bipartido completo com partições de tamanho $|V_1|$=m e $|V_2|$=n, é denotado $K_{m,n}$.

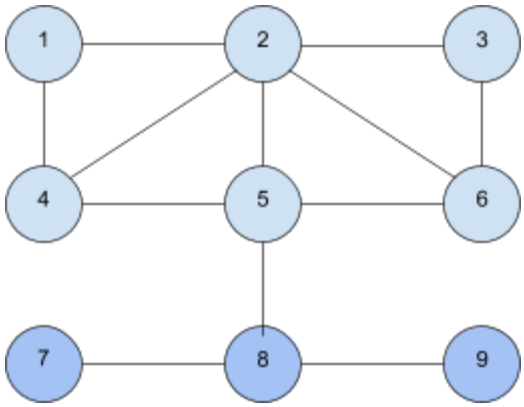
Supondo que tenhamos várias cidades, e que nossas cidades são os vértices e que as estradas que ligam essas cidades são as arestas, existe uma cidade central que é ligada a todas as outras cidades.



Clique

Um clique em um grafo não direcionado G = (V, E) é um subconjunto de vértices C $\subseteq$ V, tal que para cada dois vértices em C, existe uma aresta os conectando. Isso se equivale a dizer que um subgrafo induzido de C é completo (em alguns casos, o termo clique também pode ser referência ao subgrafo).

Supondo que vários bairros formem um distrito, e que nossos bairros são os vértices e que as estradas que ligam esses bairros são as arestas, supondo que tenhamos uma cidade que possua vários distritos, o clique é o que vai dividir esses dois distritos em um novo distrito.



Grafo Simples

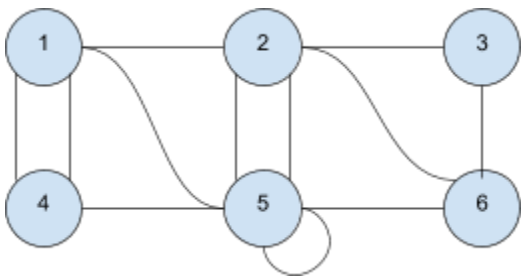
Em teoria dos grafos, um grafo é simples se ele não tem laços nem mais de uma aresta ligando dois vértices.

Supondo que tenhamos duas cidades que são ligados por uma estrada, e somente uma estrada.

Multigrafo

Multigrafo é um grafo não diricionado que pode possuir arestas múltiplas (ou paralelas), ou seja, arestas com mesmos nós finais. Assim, dois vértices podem estar conectados por mais de uma aresta. Formalmente, um multigrafo G é um par ordenado G = (V,E) sendo V os vértices e E as arestas.

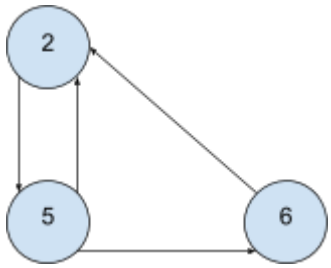
Supondo que tenhamos várias cidades, e que nossas cidades são os vértices e que as estradas que ligam essas cidades são as arestas, essas cidades são ligadas por não uma estrada de ida e volta, mas sim duas estradas, uma de ida e outra de volta, e além disso elas possuem estradas que levam a elas mesmas.



Dígrafo

Um dígrafo G=(V,E) é constituído de um conjunto finito de vértices V e um conjunto E de arestas, onde cada aresta corresponde a um par ordenado de vértices.

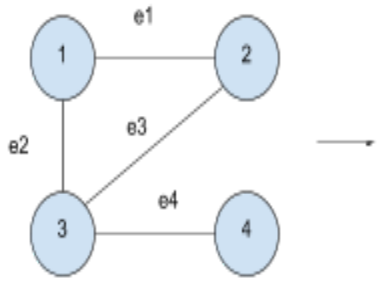
Supondo que tenhamos três bairros de uma cidade, onde os bairros são nossos vértices e a cidade é o nosso grafo, e os nossos bairros são ligados por vias expressas, que são nossas arestas. cada mão da via expressa representa uma aresta, cada via tem duas mãos.



Questão 4

Matriz de incidência:

Uma matriz de incidência representa computacionalmente um grafo através de uma matriz bidimensional, onde as colunas representam os vértices e a e as linhas representam as arestas.

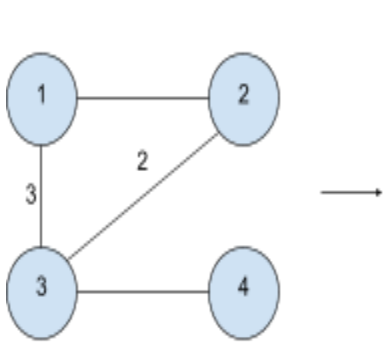


	e1	e2	e3	e4
1	1	1	0	0
2	1	0	1	0
3	0	1	1	1
4	0	0	0	1

Dado um grafo G com n vértices e m arestas, podemos representá-lo em uma matriz n x n M. A definição precisa das entradas da matriz varia de acordo com as propriedades do grafo que se deseja representar, porém de forma geral guarda informações sobre como os vértices se relacionam com cada aresta (isto é, informações sobre a incidência de uma aresta em um vértice[1]).

Matriz de adjacência:

A matriz de adjacência de um grafo com |V|∣V∣ vértices é uma matriz |V| \times |V|∣V∣×∣V∣ de 0 e 1, na qual a entrada na linha ixi e coluna jxj é 1 se e somente se a aresta (i,j)(i,j) estiver no grafo. Se você quiser indicar o peso da aresta, coloque-o na entrada da linha ixi, coluna jxj.



	1	2	3	4
1	0	1	3	0
2	1	0	2	0
3	3	2	0	1
4	0	0	0	1

Lista de adjacência:

Consiste de um array Adj de |V| listas, um para cada vértice de V. Para cada u em V, Adj[u] consiste de todos os vértices de G adjacentes a u. Vértices armazenados de forma arbitrária na lista e também pode ser utilizada no caso de grafos dirigidos.

Uma matriz de adjacência é uma abstração de um grafo definido em uma matriz bidimensional nxn, onde quando uma aresta está ligando os vértices o elemento que identifica essa ligação será 1, ex: uma matriz 3x3, e os vértices 3 e 2 são ligados por arestas, o elemento n5 será preenchido com o valor 1, caso contrário será preenchido com o valor 0. Isso irá garantir rápido acesso a informação.

Matriz x Lista

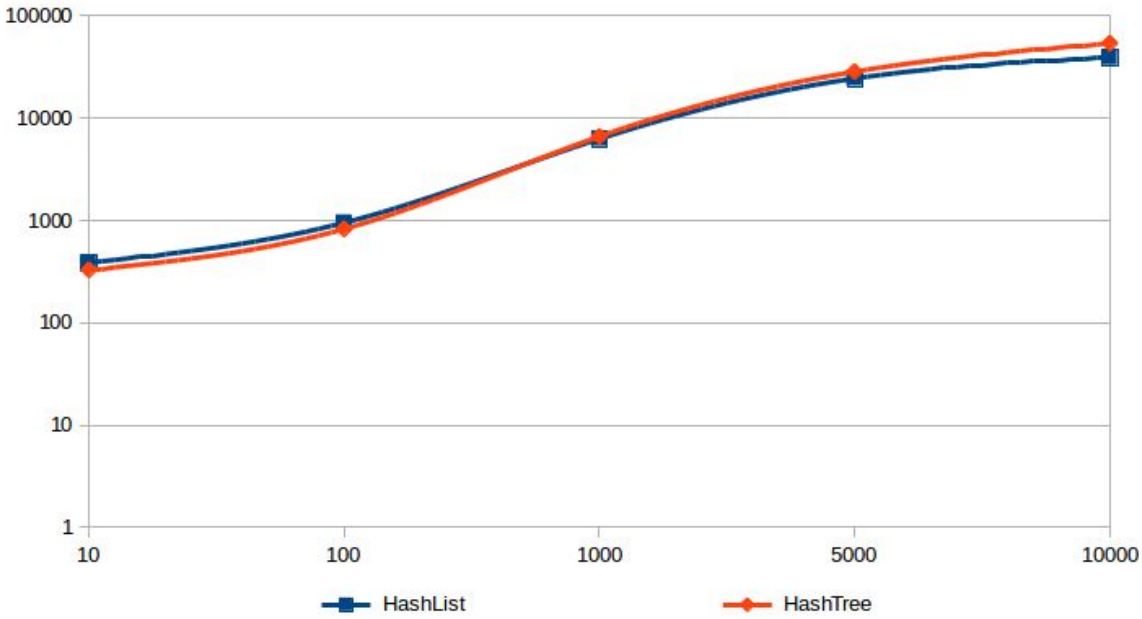
De acordo com as propiedades da matriz de adjacência, os valores da diagonal principal da matriz são 0, logo ela não teria dificuldade em mostrar loops, para fazer isso basta mudar um valor da diagonal principal para um valor diferente de 0

O fato de que as linhas da matriz de adjacência contêm normalmente muitos 0 sugere outras implementações mais compactas. Uma delas é é a estrutura de adjacência. Seja G=(V,E) um grafo. A estrutura de adjacência A de G é um conjunto de *n* listas A(v), uma para cada vértice *v* de V. Cada lista A(v) é denominada *lista de adjacências* de *v*, e contém todos os vértices que são adjacentes a *v*. A lista de adjacência pode ser problemática pois nem sempre podemos representar uma aresta especificando somente os dois vértices que ela liga. É preciso também rotular as aresta.

Questão 5

Uma tabela hash é uma associação entre chaves e valores. As chaves, quando usadas como argumento de uma função hash, retornam um valor da tabela, o que permite que se possa fazer uma “busca” com complexidade $O(1)$. Uma função hash pode acabar gerando o mesmo resultado para chaves diferentes, causando conflito na tabela. Uma maneira de resolver este conflito seria usando uma lista de valores no lugar de apenas um valor, o que aumentaria a complexidade das operações para $O(n)$. Como alternativa à lista,

pode-se usar também árvores balanceadas para os valores das chaves, garantindo uma menor complexidade $O(\log n)$.



Questão 6

Enumeração:

Enumeração é quando temos elementos de um conjunto finito com valores atribuídos, normalmente esses números são atribuídos como inteiros.

Explicita x Implícita:

Um exemplo de enumeração explicita x implícita. Tendo um baralho espalhado na mesa, o jogador já tem conhecimento do valor de cada carta, não irá precisar virar carta alguma para saber seu valor, já na implícita, o jogador teria que virar carta por carta para descobrir seu valor.

Programação Dinâmica:

Utiliza um esquema de enumeração de soluções que visa, através de uma abordagem de divisão-e-conquista (decomposição), minimizar o montante de computação a ser feito. A sua aplicação é indicada quando há casos em que um mesmo subproblema aparece diversas vezes ao longo do processo, onde a decomposição pura e simples é incapaz de reconhecer este fato, pois os subproblemas são resolvidos uma vez só e reutiliza a solução toda vez que o mesmo aparecer novamente.

Programação dinâmica é programação recursiva usando tabelas, ao invés de resolver recursivamente, ela resolve de forma sequencial armazenando as soluções em uma tabela, assim toda vez que uma solução para um subproblema for necessária ela já vai estar na tabela.

Programação dinâmica é muito utilizada em problemas de divisão e conquista que tendem a ter um número de subproblemas exponenciais, mas esses subproblemas se repetem frequentemente, assim a tabela irá reduzir bastante o tempo de execução.

Método usando decomposição pura

```
int fibo(int n){
    if ( n <= 2){
        return (n-1);
    }
    else
        return (fibo(n-1)+fibo(n-2));
}
```

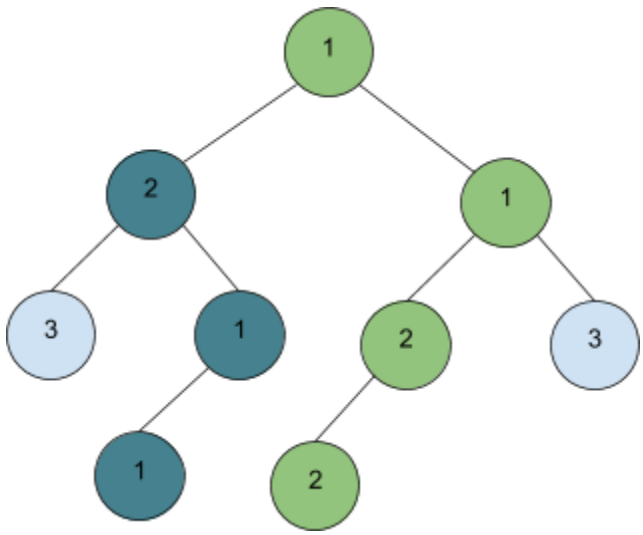
Método usando programação dinâmica

```
int fibo(int n){
    int fib[n];
    fib[0] = 0;
    fib[1] = 1;
    for (j = 2; j <= n; j++){
        fib[j] = (fib[j-1] + fib[j-2]);
    }
    return fib[j-1];
}
```

Algoritmo Guloso:

É um algoritmo procedural que toma suas decisões pela escolha que mais trará benefícios.

Esses algoritmos não analisam um problema como um todo, apenas o próximo passo para tentar informar a saída o mais rápido possível. Uma vez que ele escolhe o próximo passo, ele não volta atrás.

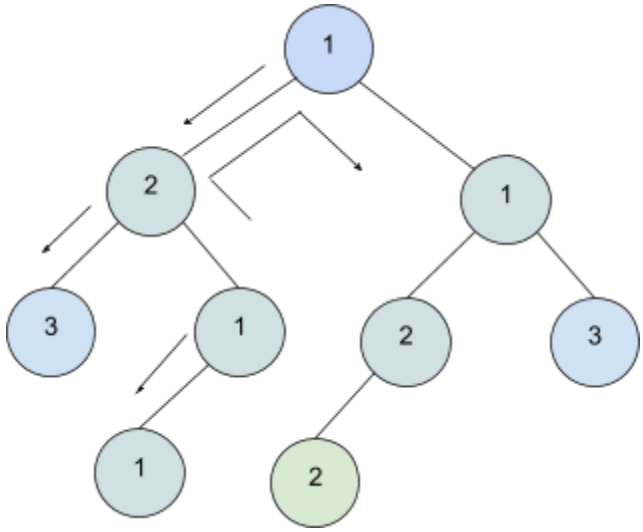


O caminho verde é o caminho que o algoritmo escolhe. Perceba que o melhor caminho é o caminho azul, mas como ele escolhe a melhor opção de acordo com a situação, ele escolheu o caminho verde, porque na primeira bifurcação ir para direita era melhor do que ir pra esquerda.

Backtracking:

É um algoritmo de força bruta, que toma várias decisões entre várias possibilidades, onde não há informação suficiente para saber qual decisão tomar, mas dentre umas dessas

decisões uma é a satisfatória. O algoritmo usa recursão para dividir o processo em várias subtarefas, assim pode explorá-las assim conseguindo gerar uma árvore de tentativas.



O algoritmo vai tentar todas as possibilidades até achar uma que seja satisfatória.

Questão 9

Problema SAT x Teoria da NP-Completude:

O problema da satisfatibilidade é o problema central da teoria da NP-completude, tal problema consiste em dada uma expressão booleana, pergunta-se se há alguma atribuição de valores para as variáveis que torne a expressão verdadeira. O algoritmo para a resolução desse problema consiste em testar todas as possibilidades de atribuição de valores para as variáveis. As soluções para esses problemas são exponenciais, que o classifica como problema NP-completo.

Exemplo:
A = (x1 ˅ ¬x2 ˅ x3) ^ (¬x1 ˅ x2 ˅ ¬x3)
x1 = 1, x2 = 1 e x3 = 1.
A = TRUE.
Essa solução é satisfativa para a expressão A.

As classes P, NP, NP-Completo e NP-Difícil caracterizam problemas quanto à sua solução em tempo polinomial.

P é solucionável em tempo polinomial.

NP não é solucionável em tempo polinomial.

NP-Difícil são problemas que não são solucionáveis em tempo polinomial.

NP-Completo é a interseção entre NP e NP-Difícil: representa problemas que têm solução, mas não em tempo polinomial.

Questão 10

A redução do problema SAT ao Clique é transformar uma instancia X do

SAT em uma instancia Y do clique, da forma que se X = True se somente se Y = True.

Algoritmo polinomial para, dada uma expressão booleana na FNC, α, (instância de SAT), gerar um grafo G(V,E) e um natural k ≤ |V| tal que:

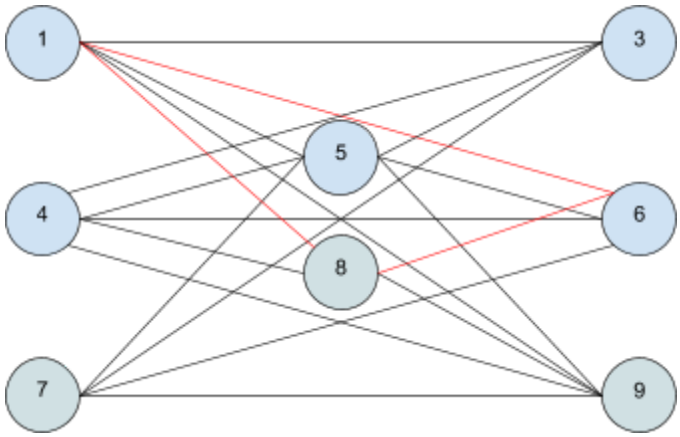
α é satisfatível se existe um k-clique em G.

Algoritmo para gerar G(V,E) e k:
Seja = C1 ^ C2 ^ ... ^ CM

V <- {vi j, onde i é a cláusula e j é a variável }
E <- { (vi j, v k l), onde l != k e vi j ¬ ˅ kl }.

Exemplo:

α = (X ˅ Y ˅ Z) (y ˅ z) (¬X˅ Y ˅ Z)



Questão 11 - extra

Sequência Fibonacci

É uma sequencia, onde tendo os dois primeiros números, o sucessor é a soma dos dois anteriores. ex: 0 1 1 2 3 5 8 13 ..

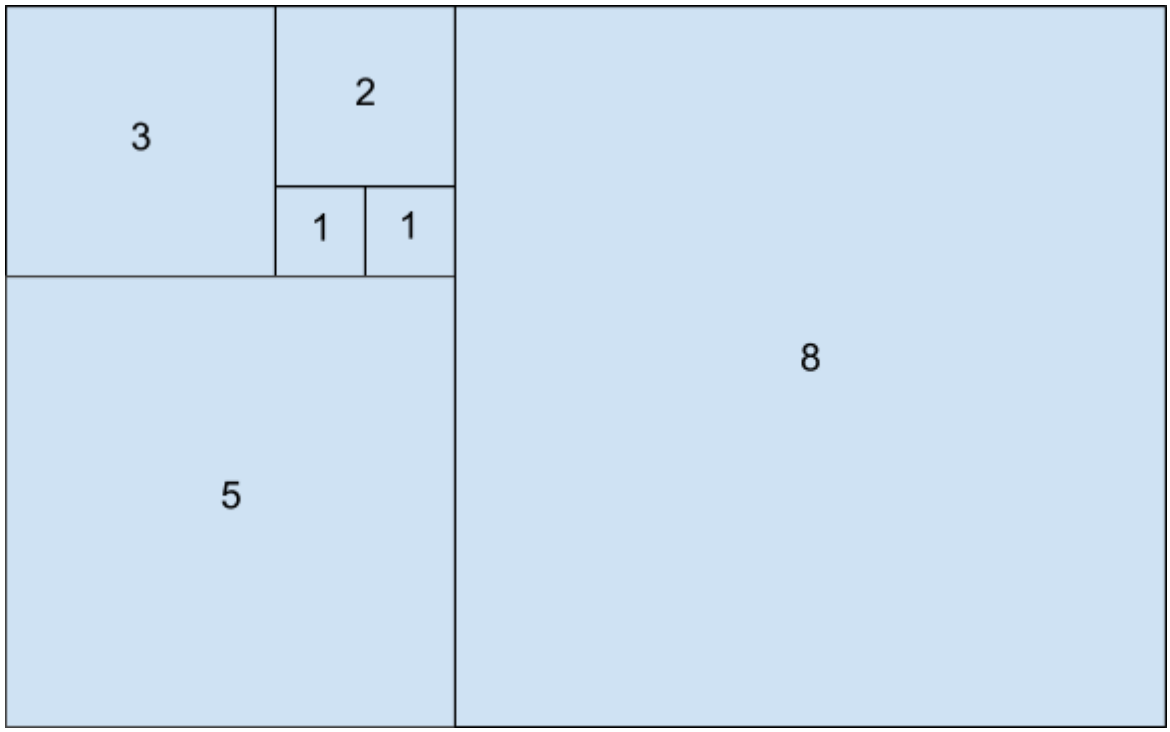
Razão Áurea

É uma constante real algébrica irracional e com o valor arredondado a três casas decimais de 1,618.

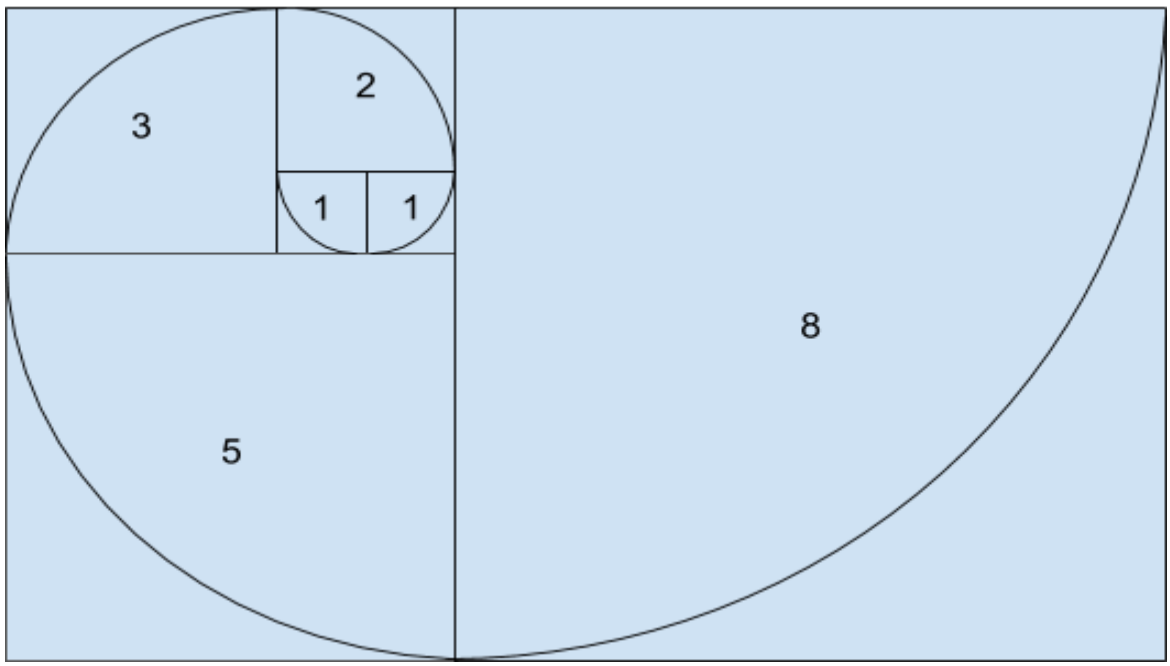
Razão Áurea x Fibonacci

A razão áurea era usada em construções para mostrar simetria e também encontrada na natureza, como em cascos de caracóis e plantas, assim como a sequência fibonacci.

Usando a sequencia fibonacci para construir um retângulo com dois números interligados, vamos ter o Retângulo de Ouro.



Passando um arco neste retângulo, teremos a espiral fibonacci.



Essa simetria pode ser encontrado na maioria das coisas como já foi mencionado.

Todas as imagens eu que fiz.