Questão 1

a) 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 $O(\phi^n)$.

Melhor caso e pior caso é 2^n.

```
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;
        }
}
```

b) Geração de todas as permutações de um número é o arranjo de elementos distintos de um conjunto. Se esse conjunto for [1,2,3] então a permutação de seus elementos resultará em: [1,2,3], [1,3,2], [3,1,2], [3,2,1], [2,3,1], [2,1,3].

Questão 2

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

$$aT\left(\frac{n}{b}\right) + c$$

1ª= $aT\left[aT\left(\frac{\frac{n}{b}}{b}\right) + \frac{c}{b}\right] + c$

= $aT\left[aT\left(\frac{n}{b^2}\right) + \frac{c}{b}\right] + c$

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

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

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

= $a^4\left[\left(\frac{n}{b^4}\right) + \frac{c}{b^3} + \frac{c}{b^2} + \frac{c}{b}\right] + c$

= $a^kT[\ \left(\frac{n}{b^k}\right) + \sum_{i=1}^{k-1}\frac{c}{b^i}\ ] + c$ k= log n, a^k = n, b^k = n.

= $a^{\log n}[\ \left(\frac{n}{b^{\log n}}\right) + \sum_{i=1}^{k-1}\frac{c}{b^i}\ ] + c$

=n[$\left(\frac{\mathrm{n}}{\mathrm{n}}\right) + \sum_{i=1}^{k-1}\frac{c}{b^i}\ ] + c$ =$1 + n\sum_{i=1}^{k-1}\frac{c}{b^i} + c$

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

$2T\left(\frac{n}{2}\right) + \Theta(n)$

1ª =$2T\left[2T\left(\frac{\frac{n}{2}}{2}\right) + \frac{n}{2}\right] + n$

=$4T\left(\frac{n}{4}\right) + 2n$

2º$=4T\left[2T\left(\frac{n}{\frac{4}{2}}\right)+\frac{n}{2}\right]+2n$

$= 4T\left[2T\left(\frac{n}{8}\right)\right]n+2n$

$=8T\left(\frac{n}{8}\right)+3n$

3ª$=8T\left[2T\left(\frac{n}{\frac{8}{2}}\right)+\frac{n}{2}\right]+3n$

$=8T\left[2T\left(\frac{n}{16}\right)+\frac{n}{2}\right]+3n$

$=8T\left[2T\left(\frac{n}{16}\right)\right]+n+3n$

$=16T\left(\frac{n}{16}\right)+4n$

$= 2^k(n/2^k) + kn$ $\qquad$ $k = \log n$, $2^k = n$.

$= 2^{\log n}(n/2^{\log n}) + n * \log n$

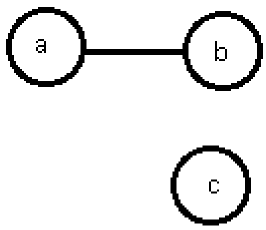
$= n(n/n) + n * \log n$

$= n * 1 + n * \log n$

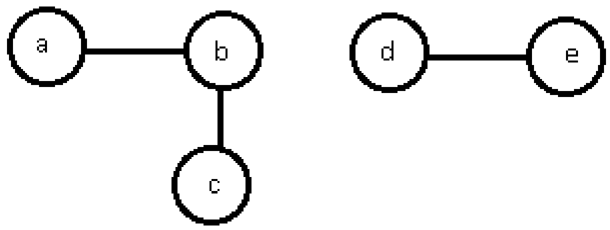
$= n + n * \log n$

Questão 3

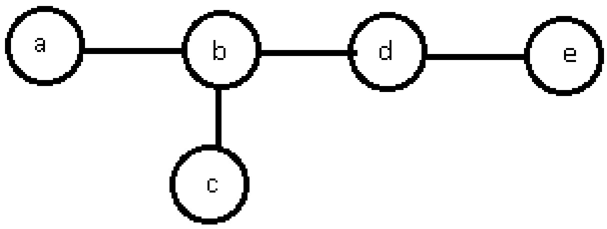
a) Grafo é um conjunto de vértices e arestas que ligam ou não pares de vértices.
Exemplo:



b) Grafo conexo: um grafo é conexo se existir um caminho entre qualquer par de vértices.
Exemplo:
Desconexo



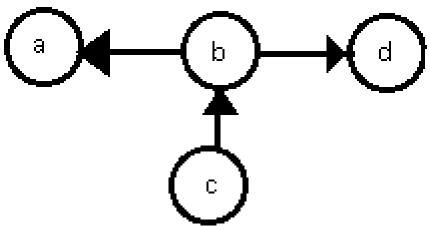
Conexo



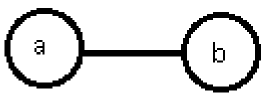
Grafo acíclico: um grafo é acíclico quando não possui ciclos.
Exemplo:

Grafo direcionado: um grafo é direcionado se cada aresta so possuir um sentido, ou seja, existe uma aresta ligando os vértices A e B, mas so é possível ir de A para B e não de B para A.

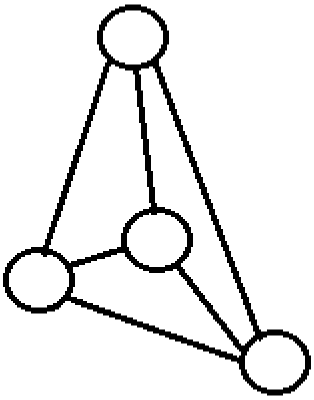
Exemplo:



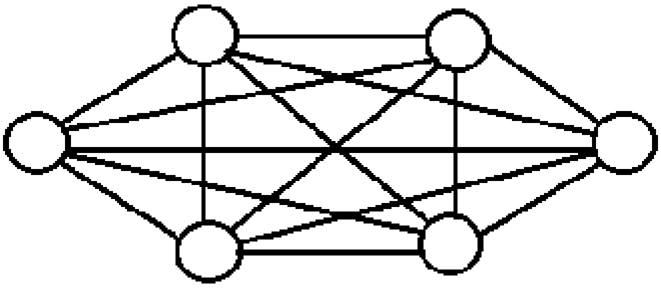
c) Adjacência ou vizinhança: refere-se aos vértices, dois vértices são adjacentes se existe uma aresta ligando os dois.
Exemplo:



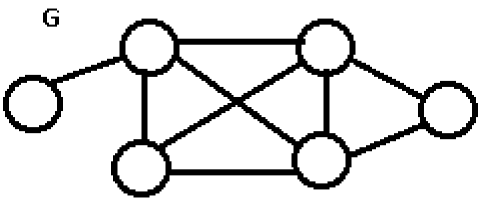
d) Grafo plana: é um grafo que pode ser imerso no plano de tal forma que suas arestas não se cruzem.
Exemplo:



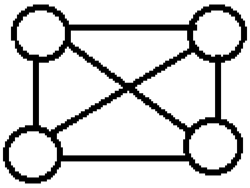
e) Grafo completo: um grafo é completo quando existe uma aresta ligando qualquer par de vértice.
Exemplo:



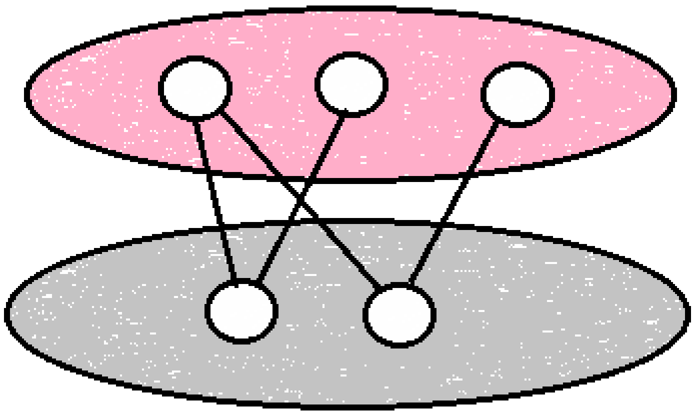
Clique: dado um grafo G, o clique seria um sub-grafo completo desse grafo G.
Exemplo:



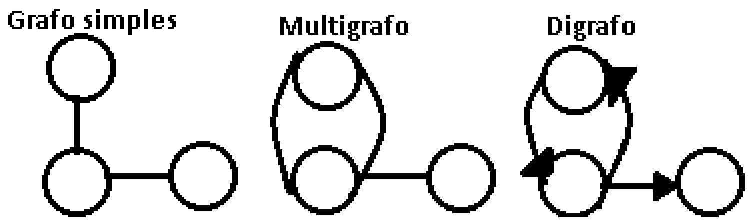
Clique de G:



Grafo bipartido: Um grafo é bipartido se for possível dividir tal grafo em dois grupos, sendo que cada vértice de um grupo só possui arestas para os vértices do outro grupo.
Exemplo:



f) Grafo simples possui apenas uma aresta para cada par de vértices, já Multigrafo pode conter mais de uma aresta para o mesmo par de vértices, e o Dígrafo pode mais de uma aresta para cada par de vértices e essas arestas possuem direção.
Exemplo:



Questão 4

<u>Matriz de incidência</u>: corresponde a uma representação matricial do grafo.

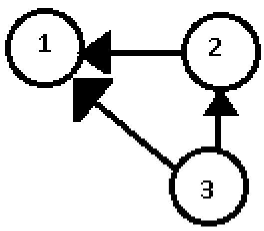
Cada elemento da matriz $\boldsymbol{a_{ij}}$ da matriz de incidência vale:

“0” se não há incidência;

“+1” se o vértice **j** sai do vértice **i;**

“-1” se o vértice **j** entra no vértice **i**.

Exemplo:

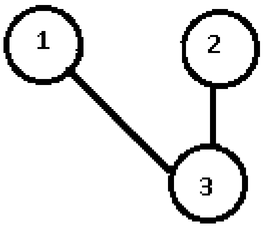


$$A = \begin{pmatrix} 0 & -1 & -1 \\ +1 & 0 & -1 \\ -1 & +1 & 0 \end{pmatrix}$$

<u>Matriz de adjacência</u>: corresponde a uma representação de um grafo de n vértices, onde cada elemento $\mathbf{a_{ij}}$ é determinado da seguinte maneira:

“1” se uma aresta ligando o **a_i** e **a_j** e “0” se não há aresta.

Exemplo:



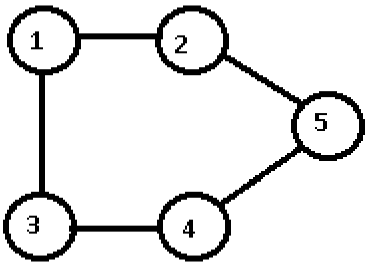
$$A = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

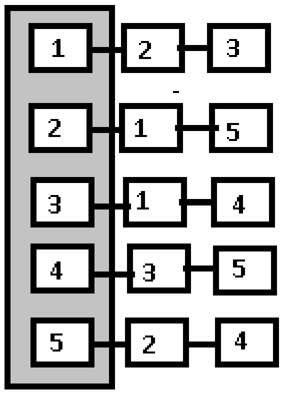
<u>Vantagem:</u> inserção, remoção e consulta de adjacência entre par de vértices levam tempo O(1).

<u>Desvantagem:</u> espaço de alocação é igual ao número de vértices ao quadrado.

<u>Lista de adjacência</u>: corresponde a uma representação de um grafo de n vértices, em que cada vértice possui uma lista e os elementos da lista são os seus vértices adjacente.

Exemplo:





<u>Vantagem:</u> espaço de alocação menor se comparado com matriz de adjacência.

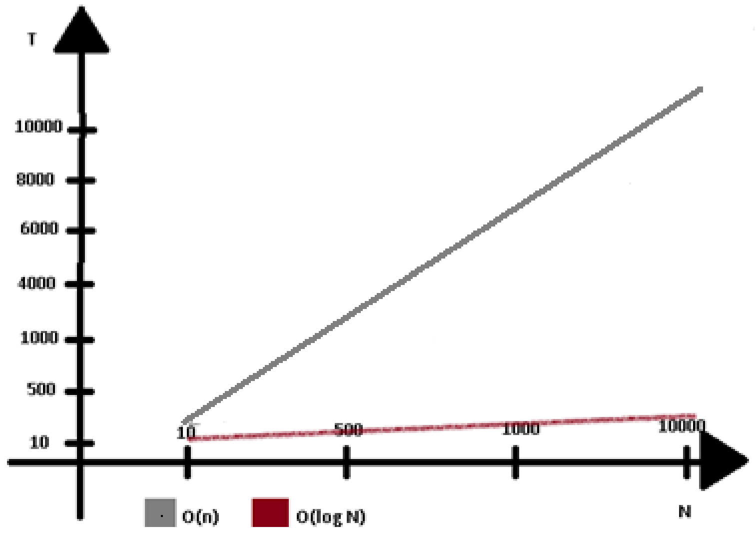
<u>Desvantagem:</u> Se os vértices possuírem muitas adjacências a lista será muito grande e o tempo de acesso para descobrir se dois vértices são adjacentes vai aumentar.

Questão 5

Tabela hash é um arranjo em que qualquer chave pode ser localizada com complexidade O(1), Funciona basicamente como um dicionário onde se pode fazer consulta a elementos.

Para fazer essa organização por chaves, é criado uma função de hashing. O grande problema é que pode haver de duas chaves apontarem para o mesmo endereço na memória, uma das alternativas é utilizar outra estrutura de dados junto com a tabela hash.

A estrutura de lista é uma das alternativas para utilizar junto com a tabela, a desvantagem dessa estrutura seria para lista grandes em que a busca, inserção e remoção passaria a ser O(n). Outra estrutura ser arvores balanceadas, no caso das colisões essa estrutura levaria vantagem sobre a lista, pois nas operações de busca, inserção e remoção seria O(log n).



Questão 6

a) Enumeração explicita X Implícita

Os dois métodos fazem uma enumeração das soluções para um determinado problema, mas há uma diferença de um para outro.

O método de enumeração explicita, faz uma enumeração de todas as possíveis soluções, já em enumeração implícita faz uma enumeração “inteligente”, com as melhores soluções para o problema.

b) Programação dinâmica

É um esquema de enumeração de soluções, em utiliza de uma técnica de decomposição minimizar o montante de computação, assim evita que um mesmo subproblema seja resolvido diversas vezes. Resolve o subproblema uma vez e reutiliza a solução toda vez que o mesmo problema aparece.

c) Algoritmo guloso

É a técnica de algoritmo que para resolver problemas de otimização, realizando as escolhas de acordo com que parece ser melhor naquele momento.

d) Backtracking

É um refinamento do algoritmo de busca por força bruta, em que boa parte das soluções podem ser eliminadas sem que seja preciso examina-las.

A principal característica desse algoritmo é técnica em procedimentos de busca que corresponde ao retorno de uma exploração.

Questão 9

a) 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:

$A = (x1 \lor \neg x2 \lor x3) \land (\neg x1 \lor x2 \lor \neg x3)$

x1 = 1, x2 = 1 e x3 = 1.

A = TRUE.

Essa solução é satisfativa para a expressão A.

b) Classes P, NP, NP-Difícil e NP-Completo.

Classes P, NP (Não-determinismo**)**, NP-completo, são definidas para problemas de decisão, P exclusivamente para algoritmos determinísticos em tempo polinomial, NP para problemas que não podem ser resolvidos nesse termo. Os problemas NP-completos possuem soluções, mas nenhuma em tempo polinomial. A classe dos problemas NP-difíceis contém os problemas de complexidade maior ou igual à do problema SAT.

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 $\alpha = c1 \wedge c2 \wedge ... \wedge cm$

$V \leftarrow$ {vi j, onde *i* é a cláusula e *j* é a variável }

E $\leftarrow$ { (vi j, v k l), onde $i \neq k$ e

vi j $\neq \neg$v k l }.

Exemplo:

$\alpha = (x \vee y \vee \neg z) \wedge (\neg y \vee z) \wedge (\neg x \vee y \vee \neg z)$