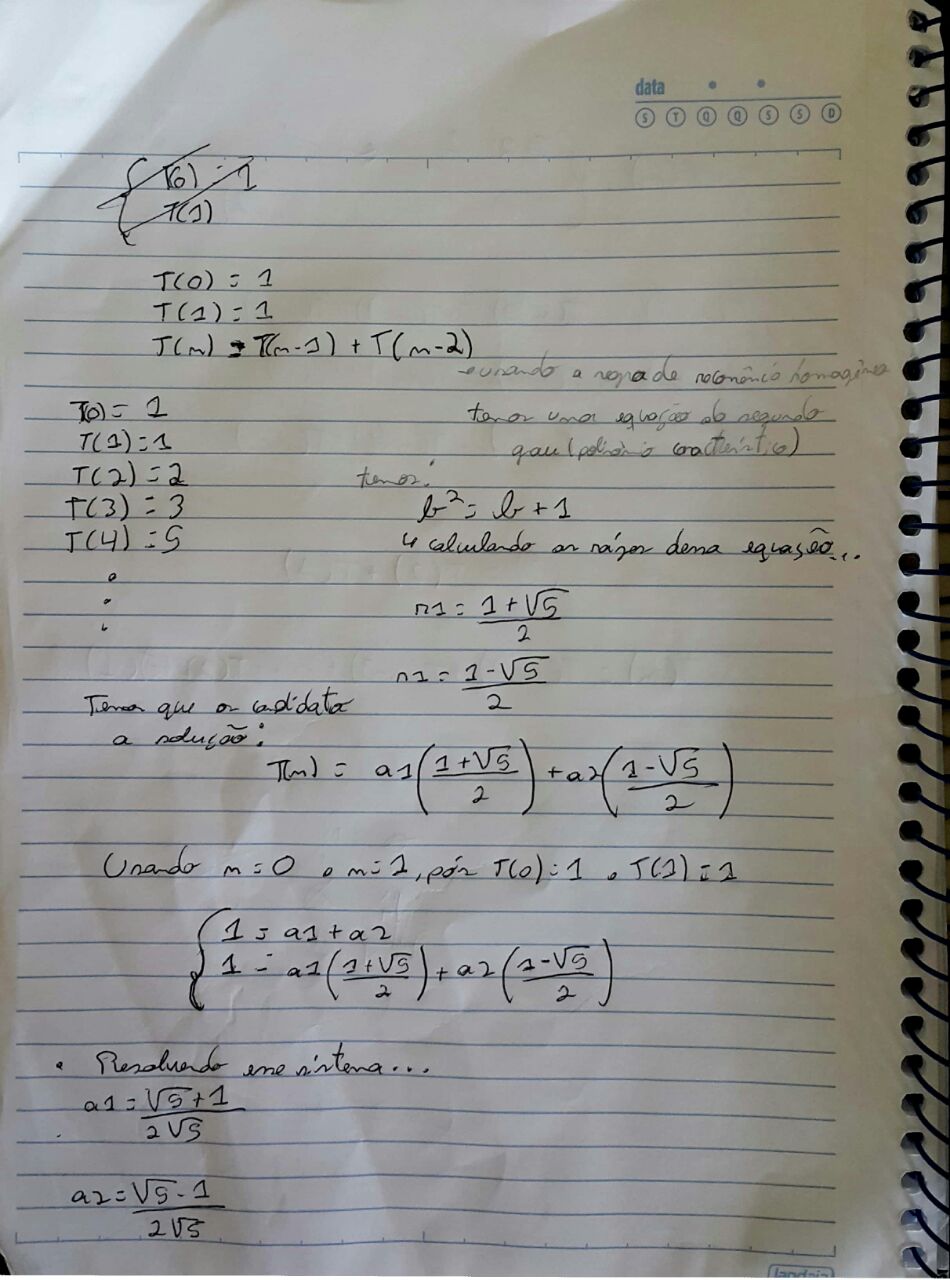
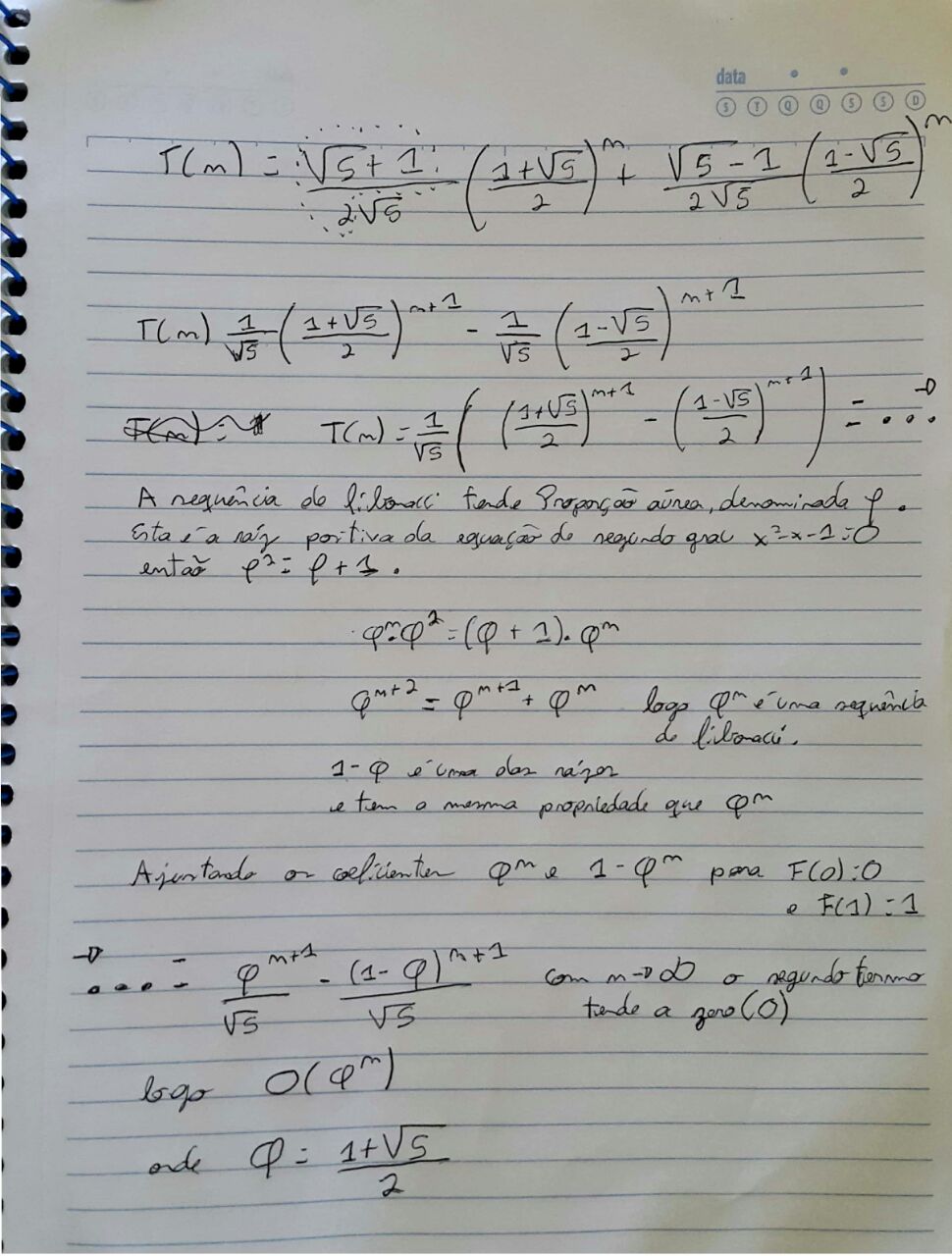
**Lista 2**

Questão 1

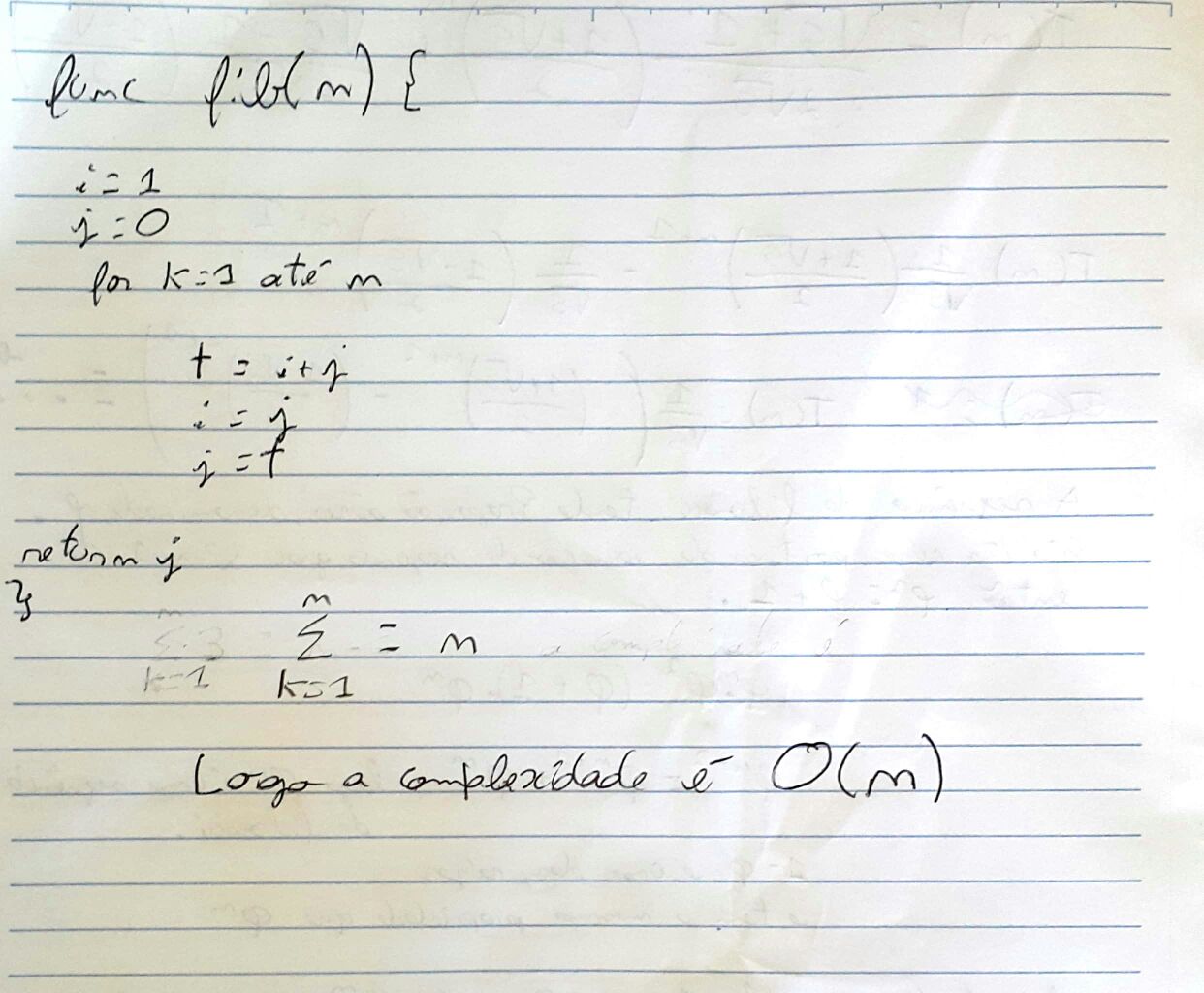
Letra (A)

Complexidade da versão recursiva





Calculo da complexidade para a versão iterativa



O limite inferior para ele é O(log(n)), pois é a melhor complexidade entre os algoritmos de Fibonacci, neste caso é um algoritmo que usa exponenciação de matrizes.

**Questão 2**

**(A)**

É um modelo matemático que representa relações entre objetos G = (V,E).

V é o número de vértices e E é o número de arestas.

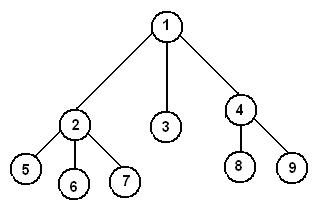


**(B)**

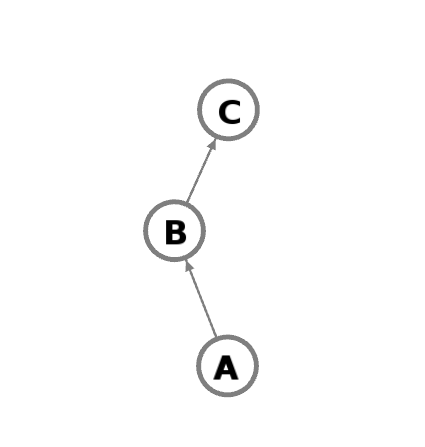
Grafo conexo: Grafo que possui arestas



Grafo acíclico: Grafo que não possui ciclos, ou seja, uma árvore



Grafo direcionado: Grafo onde as arestas possuem uma direção definida



**(C)**

Adjacência em vértices é quando dois vértices x e y forem ligados por uma mesma aresta e=(x,y).



O vértice 6 é adjacente ao vértice 4

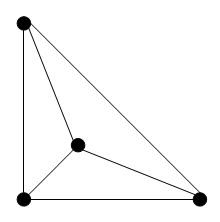
Já adjacência em arestas é quando duas arestas possuem o mesmo extremo (um mesmo vértice)



(6,4) é adjacente a (4,5)

**(D)**

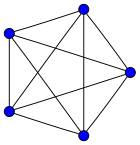
Um grafo planar é quando um grafo é colocado em um plano onde suas arestas não se cruzem



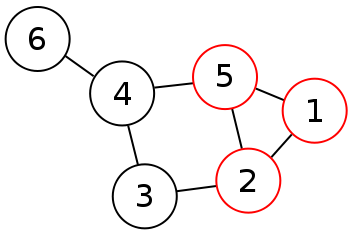
**(F)**

Grafo Completo: Um grafo é completo quando todos os seus vértices forem adjacentes.

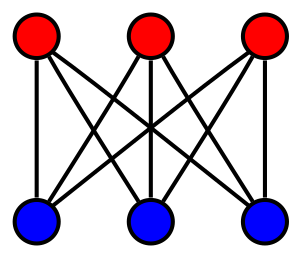
Um grafo completo Kn possui n(n-1)/2 arestas.



Clique: é um subgrafo de um grafo G, que é completo.



Grafo bipartido: Um grafo G=(V,E) é bipartido quando seu conjunto de vértices V pode ser dividido em dois subconjuntos de vértices tais que toda aresta conecta um vértice de um subconjunto com o do outro subconjunto.

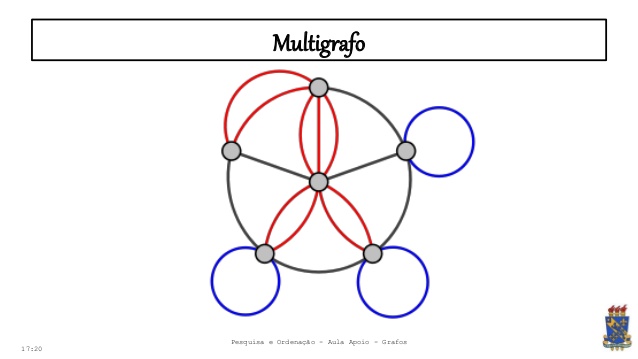


**(G)**

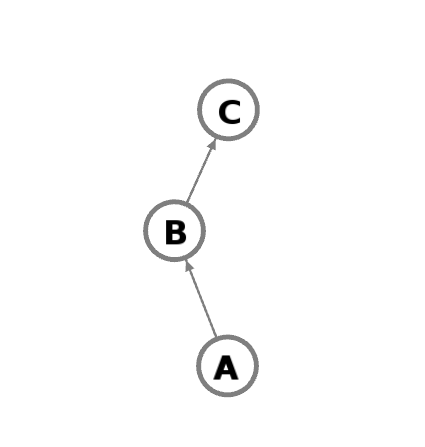
Grafo Simples: É um grafo que não possui arestas múltiplas.



Multigrafo: Quando um grafo possui mais de uma aresta interligando os mesmos dois vértices dizem-se que esse grafo possui arestas múltiplas (ou arestas paralelas).

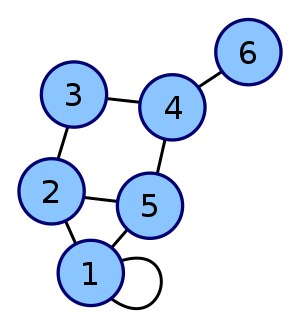


Dígrafo: quando o grafo é direcionado, ou seja, quando as arestas possuem direções.

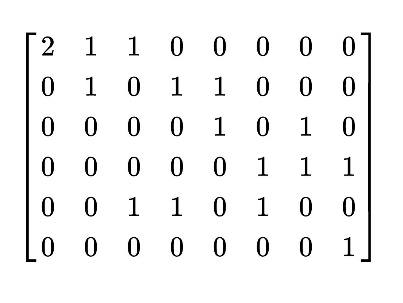


**Questão 3**

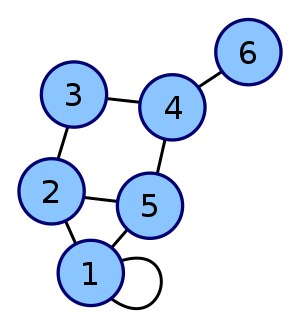
Matriz de incidência: É a representação computacional de um grafo através de uma matriz bidimensional, uma dimensão são os vértices e a outra são as arestas. De forma geral ela guarda informações sobre a relação de cada vértice com cada aresta (a incidência de uma aresta sobre um vértice)



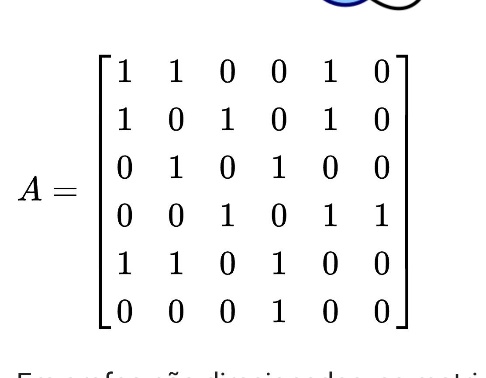
A Matriz de incidência abaixo é a representação do grafo acima.



Matriz de adjacência: Seja um grafo com n vértices, a matriz de adjacência é uma matriz n x n. A matriz guarda informações sobre como o vértice v1 se relaciona com o vértice v2.



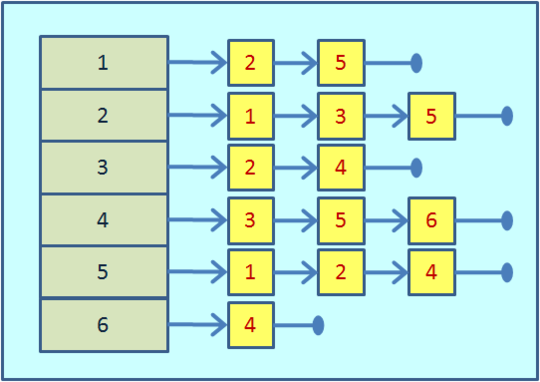
A matriz de adjacência abaixo é a representação do Grafo acima.



Lista de adjacência: É a representação computacional de um Grafo em uma estrutura de dados de lista de listas (ou vetor de listas).



A lista de adjacência a seguir é a representação do grafo acima.



A vantagem da matriz de adjacência é sua velocidade de busca, para buscar qualquer informação nela o custo é O(1). Já a desvantagen é que o gasto de memória talvez não compense isso, pois ele sempre vai armazenar um matriz n x n (n é o número de vértices), logo se o grafo possuir poucas arestas talvez não valha pena.

A vantagem da lista de adjacência é que caso o grafo possua poucas arestas o custo para a lista será baixo, por exemplo, caso possua nenhuma aresta a lista de adjacência será basicamente um vetor. A desvantagem é caso o grafo seja um grafo completo, ou seja, todos os seus vértices estão ligados entre si.

**Questão 4**

Tabela hash / tabela de dispersão / tabela de espalhamento é uma estrutura de dados que associa chaves aos elementos. Com esta chave será feito a busca do elemento. Existe vários métodos para o cálculo do espalhamento dos elementos (calcular sua posição na tabela), o mais conhecido é usar resto da divisão do elemento pelo tamanho da tabela (tamanho da tabela normalmente é um número primo).

A complexidade da inserção e remoção no pior caso é O(n).

A complexidade da busca é O(1).

**Questão 5**

Enumeração explicita: é a famosa resolução por força bruta, onde é feita todas as comparações possíveis em um conjunto de dados para se obter a resposta desejada.

Exemplo: algoritmos de força bruta.

Enumeração implícita: Quando apenas uma parte dos dados é realmente analisada, sem necessidade de se analisar todos os casos possíveis.

Exemplo: Algoritmos gulosos.

Programação dinâmica: É a técnica de resolução de problemas onde os sub-resultados são armazenados em tabelas para consulta, por exemplo, em uma resolução recursiva normalmente a mesma operação é feita mais de uma vez, nesses casos essas respostas repetidas não são calculadas mais de uma vez, pois já estão armazenadas em tabelas.

Exemplos: algoritmo de Dijkstra, algoritmo para o problema da mochila booleana.

Algoritmo Guloso: é uma resolução baseada em achar a melhor escolha local com a esperança de achar a melhor escolha global, ou seja, o primeiro resultado que satisfaz a condição ele já aceita e desconsidera as outras possíveis possibilidades.

Exemplos: problema da mochila fracionaria, problema do escalonamento de intervalos.

BackTracking: é um refinamento de busca por força bruta, onde varias soluções porem ser descartadas sem serem necessariamente analisadas.

Exemplos: N-rainhas, Caixeiro Viajante.

**Questão 6**

Pseudocódigo da multiplicação de matrizes usando programação dinâmica

MATRIXCHAINORDER (p,n )

1 para i ← 1 até n faça

2 m[i,i] ← 0

3 para l ← 2 até n faça

4 para i ← 1 até n − l + 1 faça

5 j ← i + l − 1

6 m[i, j] ← ∞

7 para k ← i até j − 1 faça

8 q ← m[i, k] + p[i − 1] p[k]p[j] + m[k+1, j]

9 se q < m[i, j]

10 então m[i, j] ← q

11 devolva m[1, n]