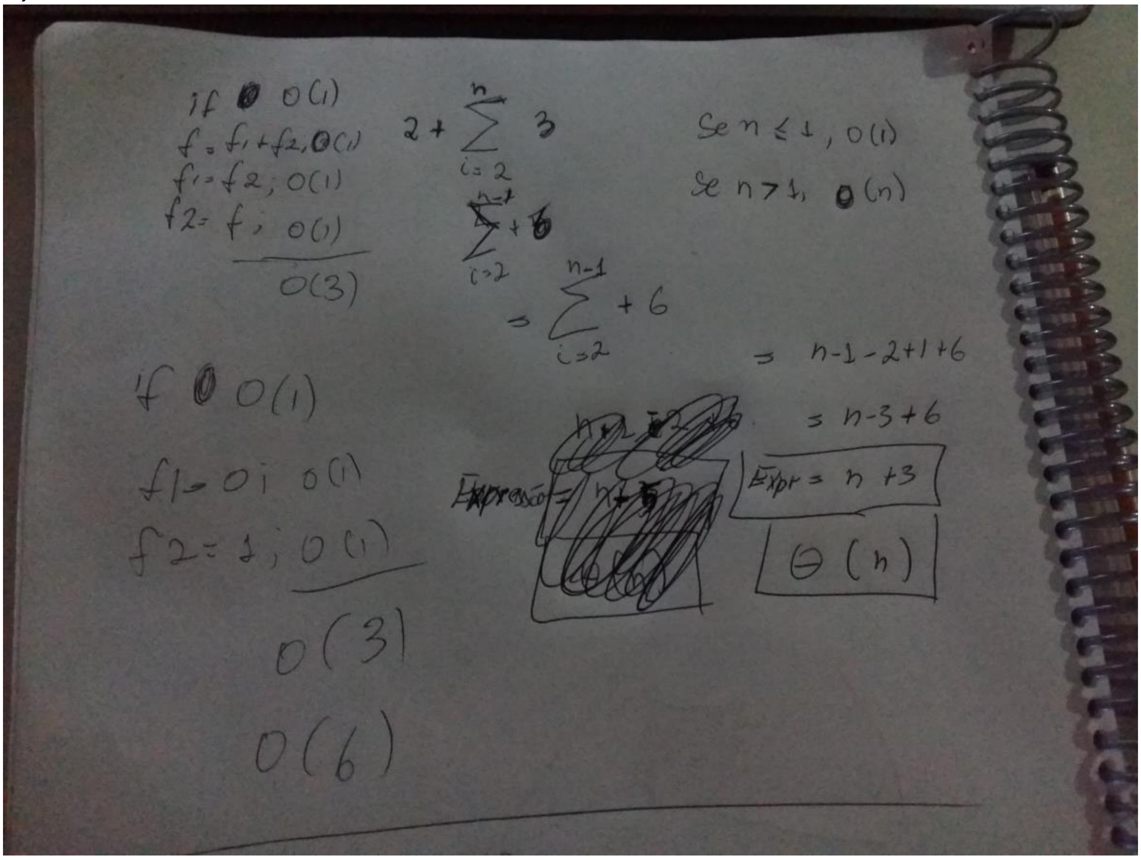
**Questão 2**

a)




Complexidade O(n).

função de custo f(n) = n+3.

b)

$$T(n) = \begin{cases} 0, & \text{se } n = 0 \\ 2T(n-1)+1, & \text{se } n > 0 \end{cases}$$

$$2T(n-1)+1$$

$$1 \quad = 2T[2T(n-2)+1]+1$$
$$= 4T(n-2)+2+1$$
$$= 4T(n-2)+3$$

$$2 \quad = 4T[2T(n-3)+1]+3$$
$$= 8T(n-3)+4+2+1$$
$$= 8T(n-3)+4+2+1$$

$$3 \quad = 8T[2T(n-4)+1]+4+2+1$$
$$= 16T(n-4)+\underbrace{8+4+2+1} = 2^{k+1}-1$$
$$= 2^{k}T(n-k)+$$
$$= 2^{k}T(n-k)+2^{k+1}-1$$
$$= 2^{n-1}T(n-k)+2^{n-1+1}-1$$

$$\boxed{\text{Expressão} = 2^{n-1}T(1)+2^{n}-1}$$

$$\boxed{\Theta(2^{n})}$$

$$n-k=1$$
$$k=n-1$$
$$T(1)$$

Complixadade $O(2^n)$.

Função de custo $f(n) = 2^{n-1} + 2^n - 1$.

**Questão 3**

A) Algoritmo guloso

Um algoritmo guloso em cada interação toma as decisões com base nas informações daquele momento e sempre escolhe a mais “apetitosa” mesmo que essa decisão não o leve para a melhor solução.

Exemplo problema da mochila fracionaria

Imagine que tenho *n* objetos que eu gostaria de colocar em uma mochila de capacidade *c*. Cada objeto *i* tem peso *pi* e valor *vi*. Posso escolher uma fração (entre 0% e 100%) de cada objeto para colocar na mochila. Quero fazer isso de modo a respeitar a capacidade da mochila e maximizar o seu valor.

Algoritmo

```
Mochila-Fracionária (p, v, n, c)
j = n
enquanto  j ≥ 1  e  pj ≤ c  faça
        xj = 1
        c = c − pj
        j = j − 1
se  j ≥ 1  então
        xj = c/pj
        para  i = j−1  decrescendo até  1  faça
                xi = 0
devolva  x
```

Complexidade O(n).

B) 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 4**

A) as etapas para provar que um problema é NP-Completo.

Para provar que **X** é um problema NP-completo é necessário seguir as seguintes etapas:

- Mostrar que X está na classe NP

  Apresentar um algoritmo que verifique em tempo polinomial se a solução é válida.

- Mostrar que **X** está na classe NP-Difícil

  Apresentar uma redução polinomial que transforma instancias de um problema **Y** que seja NP-difícil, no problema **X**

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

**Questão 5**

<u>Grafo</u>: Um grafo é uma representação abstrata de um conjunto de objetos e das relações que ocorrem entre eles, ou seja, é definido por um conjunto de vértices e pelas ligações que existem entre esses vértices, as arestas.

Seja G um grafo, G por ser definido pro G(V,E). Onde V é o número de vértices existentes no grafo e E as arestas do grafo.

**Caixeiro Viajante**

O problema do caixeiro viajante é uma questão da área de otimização. A modelagem desse problema é feita através de grafos.

O problema consiste em visitar todos os vértices de um grafo apenas uma vez e voltar ao ponto onde se iniciou a busca.

Seja G um grafo de N vértices, iniciaria uma busca em G partindo de um vértice p e passaria por todos os vértices de G e retornaria a p.

**Coloração de Grafo**

A coloração de grafos consiste em mapear o grafo e através de um conjunto finito de cores, pintar os vértices com a condição que vértices adjacentes não possuirão a mesma cor.

Uma definição formal seria, em um grafo G(V,E), mapear c: V->S, sendo que S um conjunto de cores finito, tal que se <v,w> ϵ E então c(v) ≠ c(w), cores diferentes para vértices adjacente.

**Clique em Grafo**

Para definir um clique de um grafo é necessário primeiro a definição de grafo completo.

Um grafo completo é quando existe uma aresta ligando qualquer par de vértice daquele grafo. Então a partir de um vértice pode-se alcançar os outro sem que seja necessário visitar um vértice intermediário.

Um clique de um grafo é definido por um conjunto de vértices que são dois a dois adjacente pertencente a um grafo. Seja G um grafo e C um subgrafo de G e C um grafo completo, então C é um clique de G. Cliques de grafos sempre serão subgrafos completos.