**Pontifícia Universidade Católica do Paraná**

**Escola Politécnica / Bacharelado em Ciência da Computação**

**Disciplina de Complexidade de Algoritmos** / Prof. Edson Emilio Scalabrin

**Data: 04/03/2021**

**Avaliação DIAGNÓSTICA**

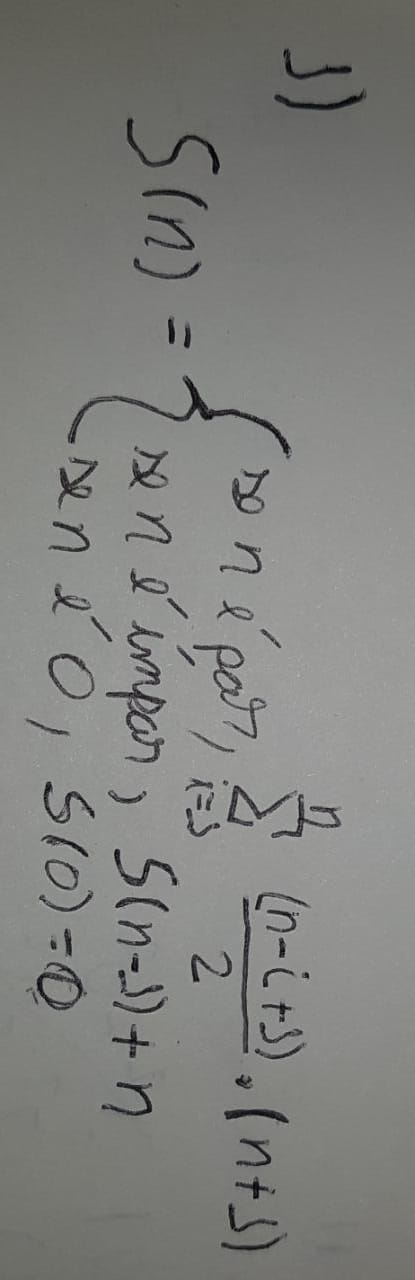
**Estudante: Gustavo Hammerschmidt.**

**Regras:** Como se trata de uma avaliação diagnóstica, ela não gera nenhuma pontuação efetiva para a disciplina. Entretanto, ela é importante para indicar o ponto de início da disciplina e também servir de referência no final da disciplina para sabermos o avanço realizado. Sendo assim, convido cada estudante tentar fazer individualmente e sem consultar nenhum material (livro, caderno, internet, colega) para responder as perguntas desta avaliação. Deve-se ter em mente que os assuntos constante neste avaliação serão abordados durante a disciplina independente do resultado que cada aluno(a) alcance. O que poderá mudar é o tempo dedicado em cada tema, que poderá ser maior ou menor.

A avaliação pode ser preenchida a mão em folha de papel, fotografada, as fotos inseridas em um arquivo Word, gerar um PDF e postá-lo no ***Blackboard***. Pode-se também fazer cópia em scanner da resolução e postá-lo no ***Blackboard*** em apenas um arquivo PDF. Outra forma, de preencher a avaliação é usando um editor de texto e postar o PDF correspondente no ***Blackboard***.

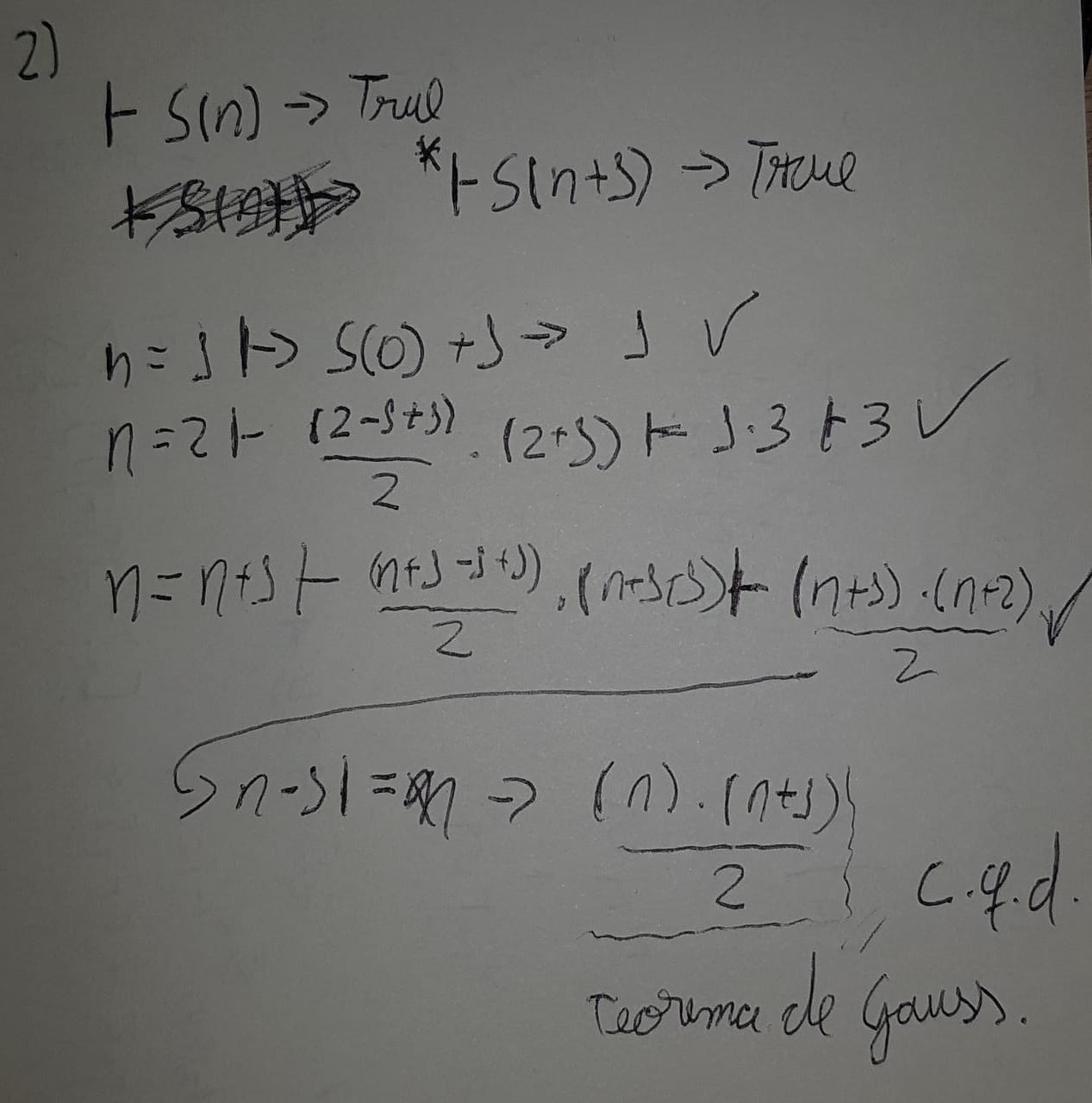
**ITEM 01**:

Escreva a fórmula fechada para a soma dos números inteiros positivos, onde S = 1 + 2 + 3 + ... + n.



ITEM 02:

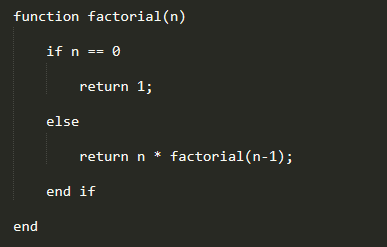
Prove por indução matemática que a fórmula para a soma dos números inteiros positivos do ITEM 01 é verdadeira para qualquer valor de n.



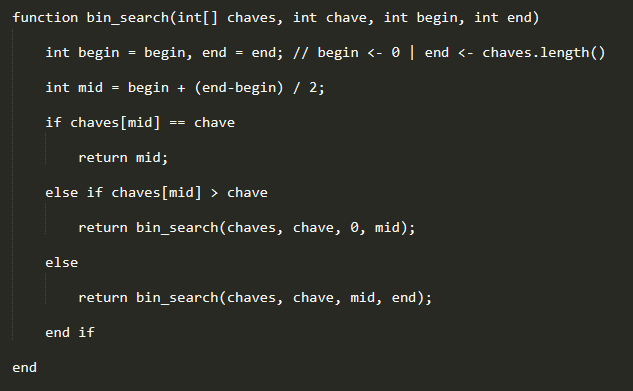
ITEM 03:

Escreva duas funções recursivas:

a) uma função para o cálculo do fatorial; e

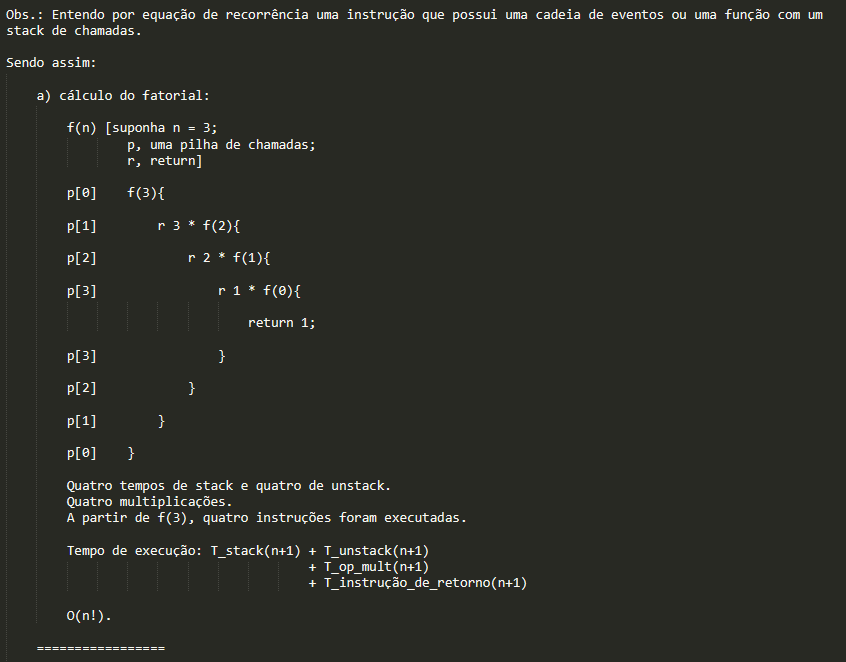


b) uma função para a busca binária de uma chave sobre um arranjo ordenado de chaves.

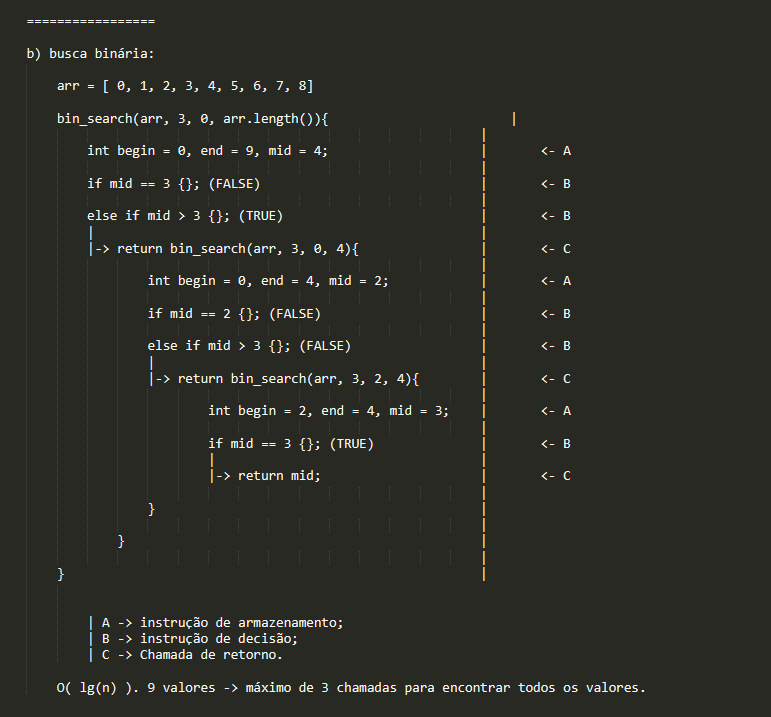


ITEM 04: Para a contagem de tempo ou de instruções, escreva as equações de recorrência para as seguintes funções:

1. cálculo do fatorial; e

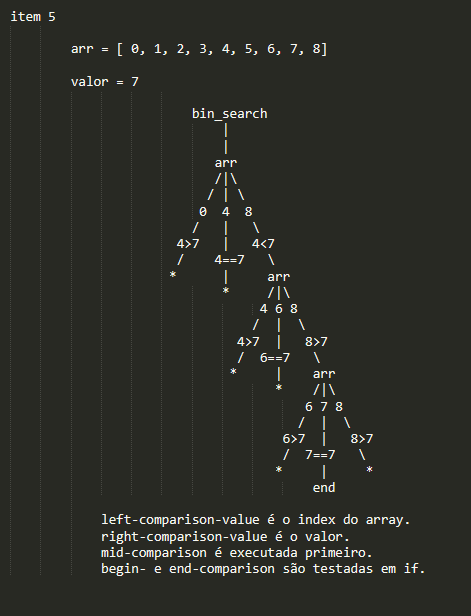


b) busca binária.



ITEM 05:

Resolva a equação de recorrência para busca binária do ITEM 3b por substituição ou árvore de resolução.



ITEM 06:

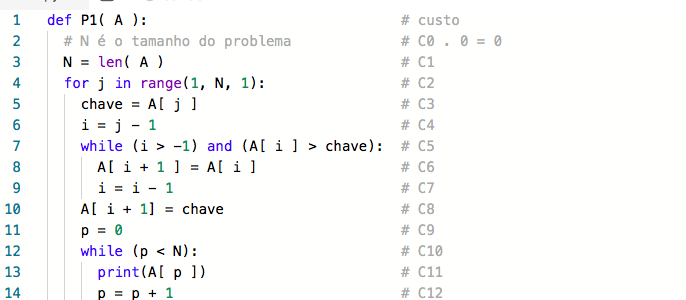
Revolva os somatórios a seguir indicando para cada um deles a sua fórmula fechada:

|  |  |
| --- | --- |
| a) | |-  - 1 surge da primeira interação, quando n = 0, somatório é igual a -1. |
| b) | = |
| c) | = - + . . . |

ITEM 07: Dado o programa **P1** abaixo escrito em Python:

1. indique (para cada linha) ao lado de cada constante Ci = 0, 1, 2, 3, ..., 12 o número de vezes que cada instrução é executada.
2. A partir do que foi feito em item (a), expresse o custo total na forma de um somatório.
3. Reescreva o somatório do item (b), assumindo que o custo de cada instrução/comando é 1.
4. Revolva o somatório reescrito no item (c).

Item a)



C1 = 1

C2 = 1

C3 = S1

C4 = S1

C5 = S1

C6 = S1 \* S2

C7 = S1 \* S2

C8 = S1

C9 = S1

C10 = S1

C11 = S1 \* N

C12 = S1 \* N

S1 = (N-1)/1 = N-1

S2 = no pior caso, o total de loops será N². O algoritmo segue a lógica de um bubblesort, ou seja, em cada interação, pode executar a mesma instrução de 1 a N vezes.

Item b)

Custo( P1(A) ), n = len(A)

Item c)

Custo( P1(A) ) , n = len(A)

Item d)

Custo( P1(A) ) , n = len(A)

, n = len(A)

Instruções de comando , n = len(A)

| Considere que o tempo de execução de instruções de comando tende a 0 ms.

Custo( P1(A) ) , n = len(A)

Note que o primeiro somatório itera de j=1 até n, e o segundo somatório nos limites de 0 até n-1, sendo assim, temos que o custo de P1 – em termos de A – é de n valores para o primeiro somatório vezes n valores (no pior cenário) para o segundo somatório vezes duas instruções. Portanto, o custo é equivalente a n \* n \* 2, desconsidere a magnitude das duas instruções, e teremos, logo, que o custo é de n² no pior cenário em termos de A para a função P1.

Custo( P1( A ) ) = [*Worst-Case*] O( n² ), [*Best-Case*] O( n ), n = len(A)

*c.q.d.*

ITEM 08: Qual é a importância da passagem de parâmetros por referência com vista a complexidade de um programa? Ilustre a situação usando textualmente e/ou graficamente.

A importância se dá quando um programa utiliza um conjunto de dados grande em memória ram, se não passados por referência, haverá uma duplicação desses elementos em memória ram – aumentando, subsequentemente, o uso de memória pelo programa e o tempo de leitura e escrita dos dados: o que pode intensificar uma piora no tempo de execução -; não somente isto, mas, se o programa utilizar recursividade, então, haverá possivelmente um uso de memória ram não só para armazenar o stack de chamadas de função mas, também, a escrita em memória dos dados de modo cascata: a cada chamada uma nova escrita dos mesmos dados é feita em memória – o que, logicamente, impacta significativamente na performance de um programa – logo, o impacto espacial será alto.