

Fundação CECIERJ - Vice Presidência de Educação Superior a Distância
Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação com Interfaces Gráficas
AP2X 2º semestre de 2020.

Nome -

Assinatura -

Questão 1) (6 pontos) Implemente um algoritmo de ordenação, baseado em busca binária¹.
 Como a complexidade da busca binária é $\log(n)$, a ordenação é:

$$\sum_{i=1}^n \log(i) = \log\left(\prod_{i=1}^n i\right) = \log(n!) \approx {}^2 n \log(n).$$

ordenada	-12	-5	0	1	2	3	4	4	9	10	100	
original	9	3	4	10	100	-5	2	1	4	0	-12	6

Tabela 1: Último elemento 6 vai ser inserido.

ordenada	-12	-5	0	1	2	3	4	4		9	10	100
original	9	3	4	10	100	-5	2	1	4	0	-12	6

Tabela 2: Busca binária retorna pos = 8

ordenada	-12	-5	0	1	2	3	4	4	6	9	10	100
original	9	3	4	10	100	-5	2	1	4	0	-12	6

Tabela 3: Elemento 6 é inserido no espaço aberto.

Pede-se implementar os métodos indicados com reticências da classe `sorter`.

Questão 2) (4 pontos) Implemente uma interface gráfica usando `tkinter` que permita gerar uma lista aleatória de inteiros, com um certo comprimento, fornecido pelo usuário, e exibir a lista ordenada correspondente. Os inteiros gerados devem estar no intervalo $[-\text{limite}, \text{limite}]$, onde `limite` é um número aleatório entre 10 e 300.

¹<https://pt.khanacademy.org/computing/computer-science/algorithms/binary-search/a/binary-search>

²https://en.wikipedia.org/wiki/Stirling's_approximation

```

class sorter:
    """Ordenador eficiente baseado em busca binária.

    @staticmethod
    def randomList(k):
        """
        Retorna uma lista aleatória de inteiros de comprimento k.
        Se k <= 0, retorna:
            [9, 3 , 4, 10, 100, -5, 2, 1, 4, 0, -12].
        """
        ...

    def __init__(self, debug=False):
        """
        Construtor. Salva o estado de depuração do programa.
        A depuração pode ser usada durante a fase de teste
        do programa.
        """
        ...

    def sort(self, x):
        """
        Cada elemento da lista de entrada original 'x' é inserido
        em uma outra lista, 'a' (que é mantida ordenada),
        na posição apropriada 'p', dada pela busca binária.
        Move elementos consecutivos, da posição 'p' em diante,
        na lista ordenada 'a', uma posição para a direita,
        para abrir espaço para o novo elemento, conforme a Tabela 2.
        Insere o novo elemento na posição criada,
        conforme a Tabela 3.
        """
        ...
        return a

    def binarySearch(self, arr, larr, x):
        """
        Função iterativa que executa uma busca binária.
        Retorna o índice de 'x' na lista ordenada 'arr',
        de comprimento 'larr', se presente, ou então
        retorna a posição 'pos' onde ele deveria ser inserido.
        Como consequência, se 'arr[pos] == x'
        então 'x' pertence a lista 'arr'.
        """
        ...
        return pos

```

```

def main():
    s = sorter()
    lista1 = sorter.randomList(0)
    lista2 = s.sort(lista1)
    print("original list = %s" % lista1)
    print("sorted list = %s" % lista2)

    n = lista1[len(lista1)//2]
    b = s.binarySearch(lista2, len(lista2), n)
    print("pos({}) -> sorted list [{}] and found = {}".format(n,b,lista2[b]==n))

    n = 6
    b = s.binarySearch(lista2, len(lista2), n)
    print("pos({}) -> sorted list [{}] and found = {}".format(n,b,b<len(lista2) and lista2[b]==n))

if __name__=="__main__":
    sys.exit(main())

```

————— Resultado da execução de main(): —————

```

original list = [9, 3, 4, 10, 100, -5, 2, 1, 4, 0, -12]
sorted list = [-12, -5, 0, 1, 2, 3, 4, 4, 9, 10, 100]
pos(-5) -> sorted list [1] and found = True
pos(6) -> sorted list [8] and found = False

```