Computação 1, 2021.2
Lista 8

Data limite para entrega: 11/2 às 18:00

Submeta suas soluções colocando os arquivos correspondentes na sua pasta do Google Drive*

Lembre-se de escolher bons nomes para suas funções e variáveis, e de documentar seu código com *docstrings* (documentação de função) e comentários onde for apropriado.

Parte 1 — Obrigatória

Questão 1.

a. Implemente uma função que receba um natural n e retorne o último dígito de n (sem converter n para nenhum outro tipo!)

b. Implemente uma função que receba um natural $n \geq 10$ e retorne o número n sem seu último dígito (sem converter n para nenhum outro tipo!)

c. Implemente uma função **recursiva** que receba um natural n e retorne a lista dos dígitos de n. Não utilize nenhum tipo de dado de Python que não sejam `lists` ou números `ints`.

Questão 2 (Merge sort)**.**

a. Faça uma função em `python` que receba como entrada duas listas ordenadas e retorne seu "*merge*": uma lista ordenada cujos elementos são exatamente os elementos das listas dadas como entrada. Não use nenhuma função de ordenação (como `list.sort` ou `sorted`) em sua solução.

b. Implemente em `python` o seguinte algoritmo, chamado *merge sort*: dada uma lista `L` a ser ordenada, quebre `L` na "metade" (ou o mais próximo disso que for possível), recursivamente ordene cada metade, e depois faça o *merge* das metades já ordenadas. Não se esqueça de fazer um caso base!

Questão 3 (Busca binária)**.** Como vimos em aula, caso tenhamos uma tupla, lista ou string `S` e queiramos saber em qual posição ela contém um certo elemento `e`, não há nada muito melhor a se fazer a não ser olhar para os elementos de `S` um por um e compará-los com `e`. Assim, se a sequência `S` tem aprox. 10^6 elementos, esse procedimento pode levar até aprox. 10^6 passos antes de terminar.

Se soubermos de antemão que a sequência `S` está *ordenada* em ordem não-decrescente e que `e` de fato é um elemento de `S`, podemos fazer *muito* melhor:

*Link recebido por email em 24/11/2021 — o nome é parecido com `<seu nome> - Comp 1 2021.2 - Submissões e Feedback.`

1. calculamos um índice i que corresponda à "metade" (aproximadamente) de `S`; se `S[i]` coincidir com `e`, retornamos i;

2. se não, ou `S[i]` é maior que `e` ou menor que `e`; no primeiro caso, podemos (recursivamente) repetir o processo apenas na primeira metade de `S`, e no segundo caso apenas na segunda metade.

Desta forma, a cada "passo" em que ainda não encontramos o elemento `e` buscado, aprox. metade da sequência `S` é descartada da busca. Com isso, buscar em uma sequência de tamanho aprox. 10^6, no pior caso, não leva mais do que aprox. 20 passos!

a. Implemente este algoritmo em `python` (lembre-se que você pode assumir que o elemento buscado pertence à sequência dada).

b. Implemente o algoritmo de busca binária descrito acima, mas sem assumir que o elemento buscado pertence à sequência ordenada dada: caso o elemento não pertença à sequência, retorne `False`.

Parte 2 — Desafio opcional

Questão 4 (Torre de Hanói de 4 pinos)**.** Suponha que você tem 4 pinos para jogar a Torre de Hanói. Isso deveria reduzir o número de passos necessários. Suponha que temos $N = n(n+1)/2$ discos.

a. Mostre que é possível (i.e., que está correto) usar o caso de 3 pinos como "caso base" da seguinte recursão:

- Transferir $n(n-1)/2$ discos do pino `origem` para o pino `auxiliar_1`, usando os 4 pinos
- Transferir os n maiores discos do pino `origem` para o pino `destino`, usando apenas os pinos `origem`, `destino` e `auxiliar_2`
- Transferir de volta os $n(n-1)/2$ discos mais leves do pino `auxiliar_1` para o pino `destino`, usando os 4 pinos,

b. Implemente o algoritmo acima em `python`.