**Instituto Federal de São Paulo - *campus* Barretos**

**Pós-Graduação *lato senso* em Programação para *Internet* e Dispositivos Móveis**

**Programação e Estruturas de Dados**

Caderno de Atividades

# Introdução

Este caderno de atividades tem por objetivo avaliar o desempenho dos discentes na resolução de problemas de programação. Ele é composto por 5 (cinco) atividades, sendo que para cada uma delas, devem ser entregues o código-fonte e um relatório, onde deve ser explicado o problema a ser resolvido de maneira geral, e os algoritmos e estruturas de dados utilizados para resolver o problema, bem como exemplos de como os algoritmos e estruturas de dados funcionam e resolvem o problema. Também devem ser explicadas a complexidade dos algoritmos e estruturas utilizadas, bem como explicado o motivo da escolha dos algoritmos e estruturas utilizados. **TODAS** as atividades devem ser realizadas. A linguagem de programação utilizada para a resolução das atividades é de escolha dos discentes. Não devem ser utilizados conceitos fora do escopo da disciplina, como banco de dados, programação paralela, entre outros. O programa não deve fazer acesso a nenhum outro processo do mesmo ou outro computador, nem acesso a APIs ou *web services*. Em todas as atividades, o processamento só deve ser iniciado após a entrada de todos os dados de entrada. Os testes serão realizados nos PCs do laboratório onde as aulas ocorrem.

**Formação de Grupos**

Esta avaliação pode ser feita de forma individual ou em dupla, a critério dos discentes.

# Data e Forma de Entrega

O prazo máximo para a entrega de todos os documentos relacionados às atividades é o dia 19/set/2022 às 22:35. A entrega deverá ser realizada em atividade cadastrada no Moodle institucional, disponível em <https://moodle.brt.ifsp.edu.br/>.

# Descoberta de senhas

Em um certo filme de ação, para descobrir a senha digitada em um teclado alfanumérico, o personagem principal descobriu cada caractere digitado e pensou em todas as palavras possíveis de serem formadas a partir daqueles caracteres digitados. Será que isso é possível de se fazer no mundo real?

## Entrada

Seu algoritmo deve ler uma quantidade finita de caracteres, em caixa baixa ou alta, seguida da tecla de retorno. Um arquivo contendo todas as palavras do alfabeto será fornecido, devendo este ser lido pelo programa (ou seja, as palavras do alfabeto não devem estar presentes no código-fonte ou em alguma biblioteca). Desconsiderar acentos e caracteres especiais.

## Saída

Exibir, em ordem alfabética, todas as palavras do alfabeto que podem ser formadas apenas pelos caracteres informados na entrada. Ao término da exibição, solicitar novamente uma nova entrada para o usuário. O programa é encerrado ao informar o comando ”SAIR!”, em caixa alta.

## Limites

O programa deve ser iniciado imediatamente. Após informar a entrada, o programa deve exibir a saída em, no máximo, 1 segundo. Devido ao *caching* de arquivos, o PC será reiniciado antes de cada teste.

## Exemplo

|  |  |
| --- | --- |
| Entrada | Saída |
| aer | aerar  aerea ara are  arar  arara area  arear era ere  erra  errar erre ra  rar  rarear  rara re |

**Solução proposta:**

# A dupla mais famosa

A busca de palavras em textos é algo bastante trivial. No entanto, neste exercício, faremos um programa para verificar qual dupla de palavras mais se repete em um texto. Por exemplo, considere o seguinte texto:

O curso de pós-graduação lato sensu em Desenvolvimento de Sistemas para Internet e Dispositivos Móveis tem como objetivo o aperfeiçoamento das competências profissionais na área da computação, preparando os discentes para o mercado de trabalho. O curso tem duração mínima de seis semestres.

No texto acima, o conjunto de palavras que mais se repete é **o curso**, num total de duas vezes. Vamos programar este problema?

## Entrada

A entrada do problema é o nome de um arquivo, que deve estar no mesmo diretório do código executável. Alguns arquivos serão fornecidos para a realização de testes. Não há diferenças entre palavras com letras maiúsculas, minúsculas ou caracteres com acento. Palavras entre símbolos (como vírgula, ponto final, quebra de linha, etc.) não são consideradas uma dupla.

## Saída

Após informar a entrada do problema, ele deve exibir as 10 (dez) duplas de palavra mais utilizadas naquele texto. Caso não haja ao menos 10 duplas de palavras, exibir todas as que existirem. Caso haja duplas empatadas, exibi-las em ordem alfabética. A exibição deve ser feita com um número, indicando a classificação, seguido da dupla de palavras. Em caso de empate, o número de palavras empatadas deve ser considerado para a próxima classificação. Todas as palavras devem ser exibidas em caixa baixa. Por exemplo:

1. este exemplo
2. mostra um

2. caso de

4. um empate

## Limites

Após informar a entrada, o programa deve exibir a saída em, no máximo, 1 segundo. Devido ao *caching* de arquivos, o PC será reiniciado antes de cada teste.

import timeit

import string

if \_\_name\_\_ == "\_\_main\_\_":

    contador = dict()

    inicio = timeit.default\_timer()

    with open('livro.txt', 'r', encoding='utf-8') as livro:

        arquivo = livro.readlines()

        for linha in arquivo:

            linha = linha.strip().lower()

            linha = linha.translate(linha.maketrans("", "", string.punctuation))

            palavras = linha.split()

            for p in palavras:

                if p in contador:

                    contador[p] += 1

                else:

                    contador[p] = 1

        livro.close()

for chave in contador:

    if contador[chave]:

        print(f"{chave} = {contador[chave]}")

fim = timeit.default\_timer()

print('Duração: %f' % (fim - inicio))

# Escovando *bits*

Sabemos que os números são armazenados e processados como dados binários, mas qual será a distribuição de uma sequência binária, de tamanho finito, dentro de um certo intervalo de números? Este programa trará uma resposta.

## Entrada

A entrada é formada inicialmente por dois números inteiros, *a* e *b*, separados por um espaço, sendo que o primeiro indica o valor inicial do intervalo e o segundo o valor final do intervalo de números a serem analisados, tal que [a,b]. Em seguida, informa-se o comprimento *p* da sequência de *bits* avaliada. Utilize a,b>0, b>a, a,b<999999, e 0<p<100.

## Saída

A saída deve exibir em ordem decrescente de aparição, todas as sequências presentes no intervalo informado. Caso haja empates, exiba os valores com empate em ordem crescente de valor.

## Limites

O programa deve esperar a entrada logo após ser iniciado e, após a informação de entrada, deve ser executado em, no máximo, 1 segundo.

## Exemplo

|  |  |
| --- | --- |
| Entrada | Saída |
| 1 5 | 01 4 |
| 2 | 10 3  00 2  11 1 |

# Comparação genética

Um tipo de pesquisa que pode ser feito com amostras de DNA é calcular a quantidade de determinadas sequências que aparecem neste DNA. Este programa deve auxiliar neste cálculo.

## Entrada

A entrada dos dados de DNA é feita pela leitura de um arquivo a ser fornecido. Também deve ser informada a quantidade de bases que devem ser agrupadas para a realização do cálculo. Lembre-se que o DNA é formado pelas bases Adenina (A), Citosina (C), Guanina (G) e Timina (T).

## Saída

A saída deve exibir todas as possíveis combinações de bases que possuem ao menos uma ocorrência nos dados de entrada, em ordem decrescente de aparições. Caso haja empate, exibir em ordem alfabética, de acordo com a combinação das bases.

## Limites

A partir do fornecimento da entrada, o programa deve exibir o resultado em, no máximo, 1 segundo. Nenhum processamento deve ser feito antes de todos os dados de entrada serem informados. Devido ao *caching* de arquivos, o PC será reiniciado antes de cada teste.

## Exemplo

Considerando um arquivo formado pelas seguintes bases:

ACATCGATCGAT

O resultado do seu processamento será:

|  |  |
| --- | --- |
| Entrada | Saída |
| 2 | AT 3  CG 2  GA 2  TC 2  AC 1  CA 1 |

import timeit

import string

if \_\_name\_\_ == "\_\_main\_\_":

    contador = dict()

    inicio = timeit.default\_timer()

    with open('dna.txt', 'r', encoding='utf-8') as dna:

        arquivo = dna.readlines()

        for linha in arquivo:

            linha = linha.strip().upper()

            for p in linha:

                contador[p] = contador.get(p,0) + 1

            for chave,valor in contador.items():

                print(f"{chave} = {contador[chave]}")

        dna.close()

fim = timeit.default\_timer()

print('Duração: %f' % (fim - inicio))

# Economia de combustível

Para auxiliar na economia de combustível de um profissional de entregas, escreva um programa que leia os nomes das cidades que este profissional deve visitar, a distância entre elas e o valor do combustível utilizado.

## Entrada

A entrada inicial é um número inteiro *n*, informando a quantidade de cidades a serem digitadas. Em seguida, digita-se o nome das *n* cidades. Logo após, digita-se o um trio **cidade1, cidade2, distancia**, indicando que há uma conexão direta entre as duas cidades, sendo distantes uma da outra pelo valor informado. Para informar o término das entradas, digita-se o comando ”FIM!”. Em seguida, deve ser informado o valor do litro combustível utilizado, bem como o nome da cidade de início da viagem e o nome da última cidade a ser visitada. Considere que o caminho de ida e volta entre uma cidade e outra possui a mesma distância, caso apenas um caminho seja informado.

## Saída

A saída deve exibir o valor, em R$, da diferença de valor gasto com combustível, aproximado em quatro casas decimais, considerando um consumo de 14KM/L, do menor caminho possível que passa por todas as cidades e o caminho mais longo possível, que também passa por todas as cidades. Cada cidade deve ser visitada uma única vez, e ambos os caminhos devem ser iniciados na cidade de início e encerrados na última cidade.

## Limites

O programa deve fazer a leitura de toda a entrada antes que qualquer cálculo seja efetuado.

Após o término das informações de entrada, a saída deve ser exibida em, no máximo, 1 segundo.

## Exemplo

|  |  |
| --- | --- |
| Entrada | Saída |
| 4  Barretos  Jaboticabal  Franca  Guaíra  Barretos,Jaboticabal,83  Barretos,Franca,141  Barretos,Guaira,46  Jaboticabal,Franca,146  Jaboticabal,Guaira,129 Franca,Guaira,111 FIM!  5.599  Barretos | R$0,7999 |