Universidade Tecnológica Federal do Paraná (UTFPR)
Departamento Acadêmico de Informática (DAINF)
Estrutura de Dados I
Professor: Rodrigo Minetto
Lista de exercícios (ordenação linear)

Exercícios (seleção): necessário entregar TODOS (moodle)!

Exercício 1) Implemente em linguagem C o algoritmo Counting-Sort visto em aula e utilize-o para ordenar sequências com 10, 100 mil, 1 milhão e 10 milhões de números inteiros com as seguintes restrições:

- números no intervalo de 0 a 10
- números no intervalo de 0 a 1.000
- números no intervalo de 0 a 1 milhão

Neste exercício, utilize o programa **counting-sort-int.c** (em **arquivos.zip**).

Exercício 2) Descreva um algoritmo que, dados n inteiros em qualquer ordem em um intervalo de 0 a k, realiza um pré-processamento do array A de entrada e depois responde a qualquer consulta sobre quantos dos n inteiros recaem em um intervalo $[a \ldots b]$ em tempo $\mathcal{O}(1)$ (tempo constante). Seu algoritmo deve utilizar o tempo de pré-processamento $\Theta(n+k)$. Para auxiliar a resolução deste exercício use o programa **process-interval.c** (em **arquivos.zip**). Exemplo de execução:

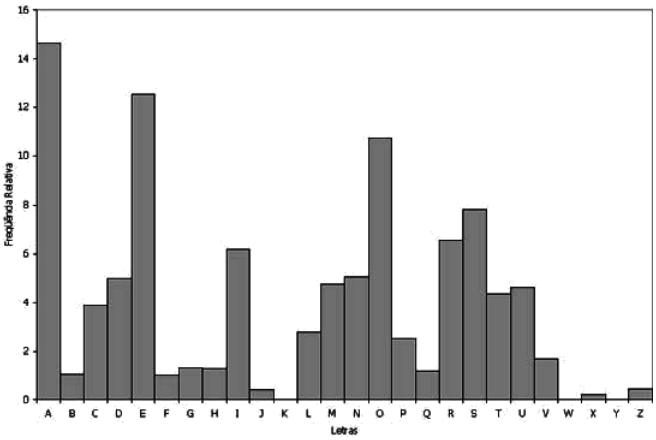
```
Input A: {9,2,0,8,7,9,3,2,0,7,5,0,2,6,0}
Consulta: qtd elems no intervalo [0 - 3] = 8
Consulta: qtd elems no intervalo [4 - 8] = 5
Consulta: qtd elems no intervalo [0 - 0] = 4
Consulta: qtd elems no intervalo [6 - 6] = 1
Consulta: qtd elems no intervalo [2 - 4] = 4
Consulta: qtd elems no intervalo [9 - 9] = 2
```

Exercício 3) A análise de frequência é um método empregado para decifrar mensagens criptografadas por meio da análise, no texto criptografado, de padrões que se repetem constantemente, que podem indicar a ocorrência de letras ou de palavras de uso corriqueiro. Em um algoritmo de substituição simples, cada letra do texto em claro é substituído por outra letra, ou símbolo, na mensagem cifrada. Por exemplo, suponha uma cifra que substitua todas letras 'a' por 'X', 'b' por 'C´, e assim por diante (sendo que este emparelhamento pode ser dar ao acaso ou através do uso de uma senha):

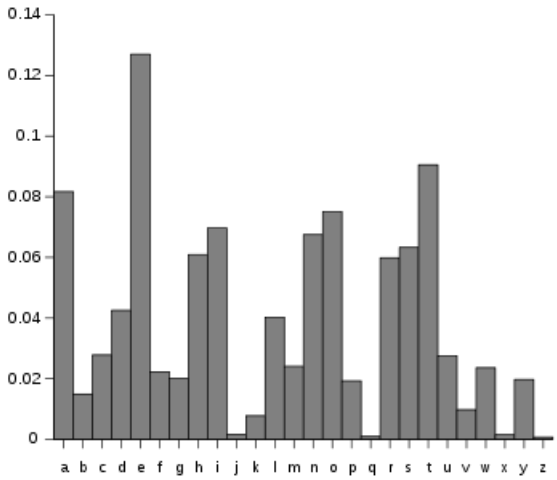
```
                  Substituindo o 'a'
Mensagem cifrada         ->        MensXgem cifrXdX
                        ....
```

Estudiosos árabes inventaram a **criptoanálise** (algoritmo para quebrar cifras por substituição conforme explicado acima). **Al-Kindi** no século XI publicou ***Um manuscrito sobre como decifrar mensagens criptográficas***: ... "um meio de decifrar uma mensagem codificada, quando conhecemos o seu idioma, é encontrar um texto diferente, na mesma língua, longo como uma página. Analisa-se em ambos a frequência com que cada letra/símbolo aparece. O símbolo que mais aparecer no criptograma deve ser transformado na letra mais frequente do texto de amostra, e assim por diante".

A análise de diversos textos em português indicam as seguintes frequências:



A frequência de letras para a língua inglesa é diferente



Neste exercício utilize um dos algoritmos vistos em aula para determinar a frequência de cada letra em um livro em português no formato TXT (que está dentro do diretório **livros**). Codifique a funcionalidade que falta no programa **counting-sort-char.c** para produzir uma saída conforme o exemplo abaixo

```
./counting-sort-char livros/livro_pt.txt
Letra  Contagem  Frequência
  a     66553      0.1377
  b      5975      0.0124
  c     17672      0.0366
  d     22955      0.0475
```

```
e     59624      0.1234
f      4374      0.0090
g      5769      0.0119
       ...
```

Observações: você pode ignorar conforme o exemplo acima as vogais e consoantes com acentos (ã,ç,é,...), para tanto, use a função **isalpha** (que retorna verdadeiro caso o caractere lido do arquivo de entrada seja uma letra entre 'a' e 'z' — letras com acentos, números, caracteres especiais resultam no valor **falso**). Caracteres maiúsculos devem ser convertidos para minúsculo logo após a leitura (use a função **tolower** para isso).

Exercícios (aprofundamento): não é necessário entregar mas é importante estudar!

Exercício 4) O algoritmo Counting-Sort é estável? Qual o pior e melhor caso deste algoritmo em função do tamanho do vetor n? Ele pode ser utilizado em qualquer domínio de problema?

Exercício 5) Qual é a menor profundidade possível de uma folha em uma árvore de decisão para uma ordenação por comparação?

Exercício 6) Implemente em linguagem C o algoritmo Radix-Sort visto em aula e teste a sua corretude para ordenar 20 números em um intervalo de 0 a 1.000. Neste exercício, utilize o programa **radix-sort-int.c** (em **arquivos.zip**).

Exercício 7) Vimos em aula o seguinte algoritmo de ordenação linear:

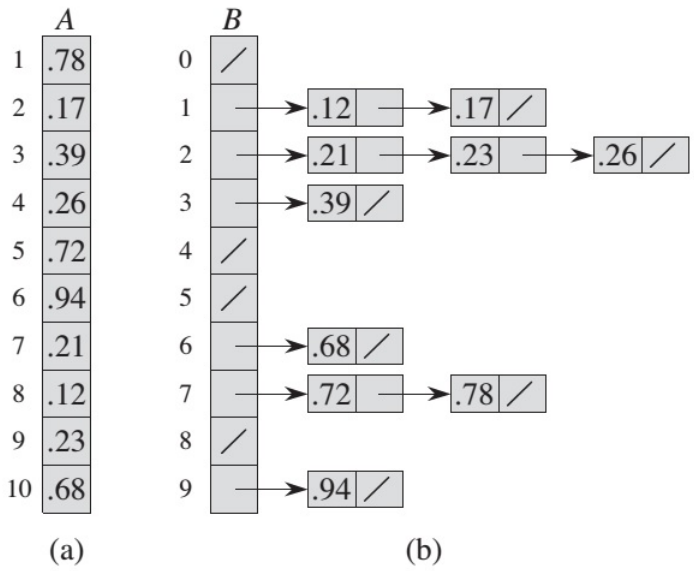
```
Counting-Sort (A[0 ... n-1], n, k)
0.  C[0..k]   /*Contador de elementos*/
1.  T[0..n-1] /*Temporário*/
2.  Para i = 0 até k faça
3.    C[i] = 0;
4.  Para j = 0 até n-1 faca
5.    C[A[j]] = C[A[j]] + 1;
6.  Para i = 1 até k faça
7.    C[i] = C[i] + C[i - 1];
8.  Para j = n - 1 até 0 faça
9.    T[C[A[j]] - 1] = A[j];
10.   C[A[j]] = C[A[j]] - 1;
11. Para j = 0 até n-1 faca
12.   A[j] = T[j]
```

Suponha que o loop **for** da linha 8 seja reescrito como:

```
7.  Para j = 0 até n-1 faça
```

Diga se o algoritmo ainda funciona corretamente e mantém as mesmas propriedades.

Exercício 8) Bucket-Sort, ou **ordenação por balde**, é um algoritmo que funciona dividindo um vetor em um número finito de recipientes (baldes). Cada recipiente é então ordenado individualmente, seja usando um algoritmo de ordenação diferente (geralmente o algoritmo Insertion-Sort), ou usando o algoritmo Bucket-Sort recursivamente. O algoritmo Bucket-Sort é geralmente utilizado (e tem complexidade **linear**) quando o objetivo é ordenar n elementos distribuídos uniformemente no intervalo $[0, 1)$. O exemplo abaixo ilustra o comportamento deste algoritmo:



O algoritmo Bucket-Sort é codificado da seguinte forma:

```
Bucket-Sort (array A, size n)
1.   n = length[A]
2.   for i = 1 to n
3.     do insert A[i] into list B[floor(n * A[i])]
4.   for i = 0 to n - 1
5.     do sort list B[i] with insertion sort
6.   concatenate the lists B[0], B[1], . . . , B[n - 1] together in order
```

Conforme o exercício do nosso livro texto, *Introdução aos Algoritmos, Cormem, MIT*, ilustre a operação do Bucket-Sort no arranjo $A = \{0.79.0.13, 0.16, 0.64, 0.39, 0.20, 0.89, 0.53, 0.71, 0.42\}$. Utilize a figura exemplo acima como guia.

Exercício 9) Qual a complexidade de pior e melhor caso do algoritmo Bucket-Sort? Utilize a notação $\mathcal{O}$ para responder.

Exercício 10) Poderíamos ordenar o array de entrada com o algoritmo Radix-Sort conforme visto em aula mas iniciando pelo dígito mais significativo? Existe alguma forma que permita a ordenação pelo dígito mais significativo?