AED2 2023 - EXERCÍCIO 6 - ORDENANDO PALAVRAS 2.0

Entrega: 02/05/2023 até 23:59:59

Instruções:

1. E/S: tanto a entrada quanto a saída de dados devem ser "secas", ou seja, não devem apresentar frases explicativas. Siga o modelo fornecido e apenas complete as partes informadas (veja o exemplo abaixo).
2. Identificadores de variáveis: escolha nomes apropriados
3. Documentação: inclua cabeçalho, comentários e indentação no programa.
4. Submeta o programa no sistema judge: `https://judge.unifesp.br/aediiS1A23`.
5. O código-fonte pode ser escrito em C, C++ ou Java.
6. **O código-fonte DEVE implementar uma solução usando o algoritmo Radix-Sort e Counting sort.**

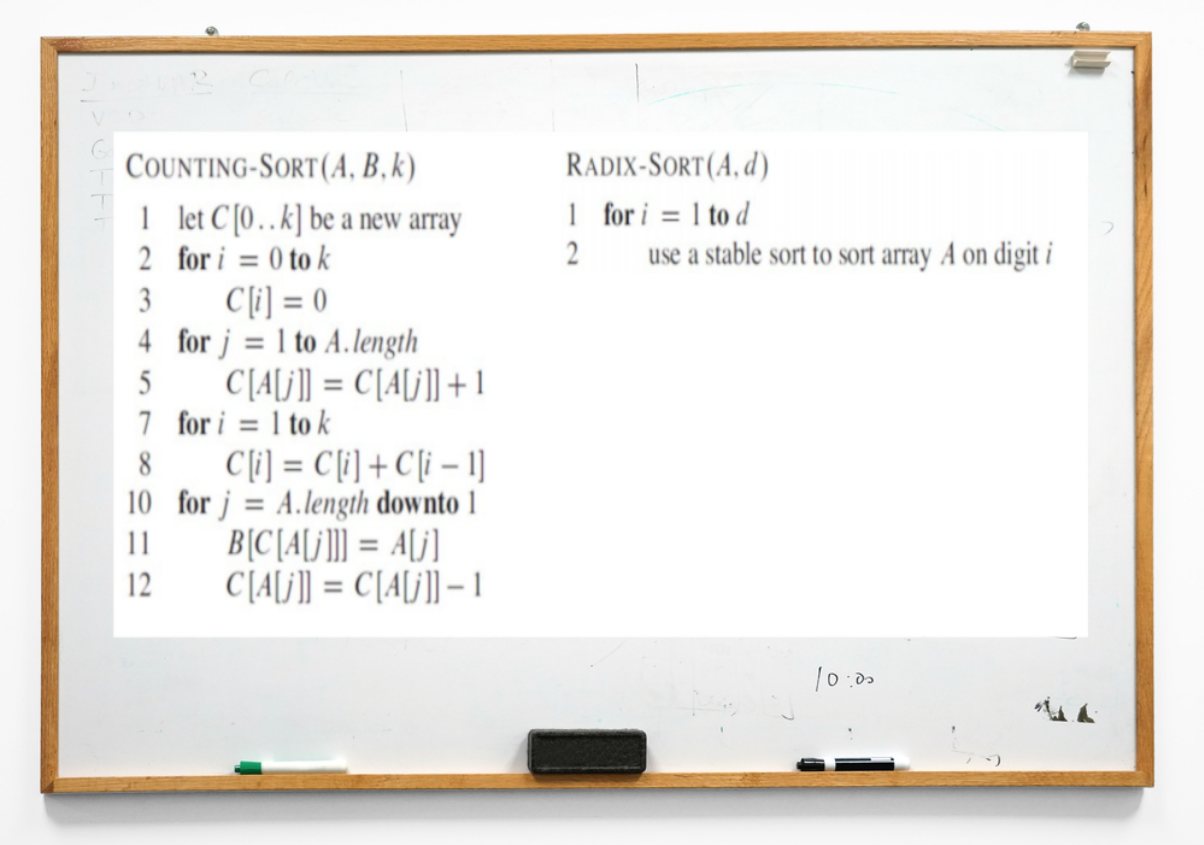
Descrição:

O grupo de amigos achou divertido praticar algoritmos de ordenação nos sábados à noite, e resolveu continuar trabalhando no problema anterior. Aproveitando o problema da semana passada, resolveram adicionar algumas regras à brincadeira:

1. Desta vez, as palavras de entrada podem conter letras maiúsculas (mas não caracteres especiais, como acentos). Caso isso ocorra, as letras maiúsculas devem ser convertidas para minúsculas.
2. Espaços em branco devem ser utilizados para deixar todas as palavras com o mesmo comprimento da maior palavra. Cada palavra tem no máximo 20 caracteres.
3. As chaves lexicográficas fornecidas, desta vez, devem conter todas as 26 letras do alfabeto.
4. É requisito obrigatório usar o algoritmo que o monitor escreveu no quadro para ordenar, e nenhum outro pode ser usado. Felizmente, um dos amigos tirou uma foto do quadro durante a semana, e trouxe a imagem para a reunião de sábado (Figura 1).

Cada amigo terá um papel na solução. O problema começa com um dos participantes sorteando um número inteiro positivo N que representa a quantidade de palavras; depois, o segundo escolhe a chave de ordenação dos caracteres; o terceiro escreve as N palavras em um papel, e assim por diante. Mais detalhes serão fornecidos no exemplo abaixo.

O jogo tem algumas restrições:



```
COUNTING-SORT(A, B, k)
 1  let C[0..k] be a new array
 2  for i = 0 to k
 3      C[i] = 0
 4  for j = 1 to A.length
 5      C[A[j]] = C[A[j]] + 1
 7  for i = 1 to k
 8      C[i] = C[i] + C[i − 1]
10  for j = A.length downto 1
11      B[C[A[j]]] = A[j]
12      C[A[j]] = C[A[j]] − 1
```

```
RADIX-SORT(A, d)
1  for i = 1 to d
2      use a stable sort to sort array A on digit i
```

Figura 1: Algoritmo

1. Cada palavra tem no máximo 20 caracteres.
2. As palavras conterão apenas letras sem acento. É garantida a ausência de caracteres especiais (acentos, etc) e números.
3. A chave fornecida tem tamanho de 26 caracteres e contem todas as letras do alfabeto.
4. O número de dígitos d deve ser o comprimento da maior string presente na entrada de cada caso de teste.
5. Para cada "dígito", você deve imprimir os valores em cada posição do vetor auxiliar C após a execução da linha 8 do algoritmo *Counting sort*. O vetor C deve ser de tamanho $k = 27$, sendo a primeira posição destinada ao caractere adicional (vazio) e as posições restantes referentes às 26 letras minúsculas **na ordem fornecida pela chave lexicográfica**.

Exemplo:

- O primeiro amigo especifica o número 7 como o número de palavras contidas na entrada.
- O segundo amigo especifica a ordem de valor dos caracteres do alfabeto, por exemplo "wnustqefyxgamhczivjlpkorbd".
- O terceiro amigo escreve as 7 palavras que formarão o conjunto de entrada: "programar VAMOS palavra eh futebol computador legal"
- O quarto verifica se há alguma letra maiúscula nas palavras, e converte para minúscula ("VAMOS"$\rightarrow$ "vamos").
- O quinto descobre o tamanho da maior palavra e completa todas com espaços em branco, para que todas tenham o tamanho máximo de 20 caracteres.
- O sexto ordena o conjunto de entrada e exibe o vetor auxiliar C após a execução da linha 8 do algoritmo *Counting sort* apresentado pelo monitor.
- O sétimo exibe a saída, seguindo o seguinte padrão: primeiro, exibe todas as palavras convertidas para letras minúsculas, na ordem em que aparecem na entrada, acrescidas de um ponto final. Depois, mostra o número inteiro "d"que corresponde ao tamanho da maior palavra. Posteriormente, exibe as d linhas do vetor C do *Counting sort*. Por fim, mostra o vetor de palavras ordenado, lembrando que o valor dos caracteres é determinado pela chave fornecida, e não por ordem alfabética.

E N T R A D A:

Primeira linha contem a quantidade N do total de palavras do conjunto inicial.

A segunda linha contem a chave com as 26 letras do alfabeto.

A terceira linha contém as N palavras separadas por um espaço em branco, representando o conjunto inicial de palavras: $p_1, p_2, ..., p_N$.

S A Í D A:

Inicialmente deve-se exibir na saída o mesmo vetor de entrada com as palavras convertidas para caracteres minúsculos. Cada linha deve conter uma palavra acrescida do sinal ponto final (".") no fim. **Importante:** o ponto final é apenas para a impressão e não deve ser adicionado às palavras no vetor.

Posteriormente, deve-se exibir um valor "*d*"correspondente a quantidade de caracteres da maior palavra do vetor de entrada.

Em seguida, para cada "dígito" i do *Radix sort*, imprima uma linha com os 27 valores do vetor C do *Counting sort*. Devem ser exibidas "*d*"linhas no total.

Finalizando, nas próximas M linhas, imprima as N palavras ordenadas, uma palavra a cada linha sem os caracteres adicionais (Obs: Nesta sub-lista ordenada as palavras **não tem** o sinal de ponto final exibido no fim, **mas incluem os espaços adicionados para completar o comprimento**).

clearpage

Exemplos de entrada e saída:

Exemplos de entrada
7 abcdefghijklmnopqrstuvwxyz ProgRamAr LegAL VAMOS eH FutEbOl comPutaDor paLaVRa
Exemplos de saída
programar. legal. vamos. eh. futebol. computador. palavra. 10 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6 6 7 7 7 7 7 7 7 7 7 7 5 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 3 5 5 5 5 5 5 5 5 5 5 5 5 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 6 6 7 7 7 7 7 7 7 7 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 4 5 5 6 7 7 7 7 7 7 1 3 3 3 3 4 4 5 5 5 5 5 5 5 5 6 7 7 7 7 7 7 7 7 7 7 7 7 1 1 1 1 1 1 1 2 2 2 2 2 3 5 5 6 6 6 6 6 7 7 7 7 7 7 7 7 0 2 2 2 2 3 3 3 4 4 4 4 4 4 4 5 5 5 6 6 6 7 7 7 7 7 7 7 0 0 0 1 1 2 3 3 3 3 3 3 4 4 4 4 6 6 6 6 6 6 7 7 7 7 7 7 computador eh futebol legal palavra programar vamos

Tabela 1: Exemplos de entrada e saída 01

Exemplos de entrada
5 avdbgtcmuhqkrpefzwyoxnjsli interessantes de ORDENACAO sao algoritmos
Exemplos de saída
interessantes. de. ordenacao. sao. algoritmos. 13 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 5 5 5 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 5 5 5 5 5 5 5 2 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 2 2 2 2 2 2 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 2 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 5 2 2 2 2 2 2 2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 5 5 5 5 5 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 4 4 4 4 5 5 5 5 5 5 1 1 1 2 2 3 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 0 1 1 1 1 1 1 1 1 1 1 2 2 3 3 3 3 3 3 4 4 4 5 5 0 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 4 4 5 algoritmos de ordenacao sao interessantes

Tabela 2: Exemplos de entrada e saída 02

Exemplos de entrada
5 dvspafethkbzicrnjwuymxolgq vEz DesTa sAo AleaTORias PalAvRas
Exemplos de saída
vez. desta. sao. aleatorias. palavras. 10 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 3 3 3 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 3 3 3 3 3 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 5 5 5 5 2 2 3 3 3 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 2 2 2 2 2 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 0 0 0 1 1 1 1 2 2 2 2 2 3 3 3 3 3 3 3 3 3 4 5 5 5 0 0 0 0 0 2 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 0 1 2 3 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 desta vez sao palavras aleatorias

Tabela 3: Exemplos de entrada e saída 03