MC102 - Algoritmos e Programação de Computadores

MC102 Oferecimento anterior

Tarefa de laboratório 11 - Opcional

Nesta tarefa, construiremos um visualizador ASCII para o algoritmo de busca binária. Dado um vetor ordenado, a busca binária é feita da seguinte forma:

- Compara-se o valor do elemento procurado com o que está na posição central da lista.
- Se for igual, indica-se esta posição.
- Se for menor, o processo é repetido, considerando-se apenas os elementos da lista que estão à esquerda da posição central.
- Se for maior, o processo é repetido, considerando-se apenas os elementos da lista que estão à direita da posição central.
- Caso não haja mais elementos para percorrer, conclui-se que o elemento procurado não está na lista.

Suponha que temos o vetor abaixo e estamos procurando o elemento 11.

```
+-----+-----+-----+-----+-----+-----+-----+
| 002 | 003 | 004 | 008 | 011 | 015 | 017 |
+-----+-----+-----+-----+-----+-----+-----+
```

Para ilustrar o processo, iremos marcar inicialmente o elemento central, cujo valor é 8. Como 8 é menor do que 11, prosseguiremos com os valores à direita da posição central, marcando o elemento 15. Finalmente, ficaremos apenas com o elemento 11, que é a chave procurada.

```
+-----+-----+-----+=====+-----+-----+-----+
| 002 | 003 | 004 ||008|| 011 | 015 | 017 |
+-----+-----+-----+=====+-----+-----+-----+
                      +-----+=====+-----+
                      | 011 ||015|| 017 |
                      +-----+=====+-----+
                      +=====+
                      ||011||
                      +=====+
```

Note que em caso de um número par de chaves, temos duas chaves centrais. Iremos trabalhar com a de menor índice.

```
+-----+-----+-----+=====+-----+-----+-----+-----+
| 002 | 003 | 004 ||008|| 011 | 015 | 017 | 020 |
+-----+-----+-----+=====+-----+-----+-----+-----+
```

Descrição da entrada

A primeira linha da entrada conterá um inteiro a ser procurado e a segunda linha um vetor de inteiros. Uma de suas tarefas será verificar se este vetor está ordenado. Veja o exemplo abaixo:

```
4
2 3 4 8 11 15 17
```

Descrição da saída

A primeira linha da saída conterá uma linha descrevendo o valor a ser procurado.

```
Elemento procurado: <elem>
```

Em seguida, o vetor lido deverá ser representado, com inteiros com três dígitos e a moldura formada por caracteres `+`, `|` e `-`conforme os exemplos dados.

```
+-----+-----+-----+-----+-----+-----+-----+
| 002 | 003 | 004 | 008 | 011 | 015 | 017 |
+-----+-----+-----+-----+-----+-----+-----+
```

Utilizaremos um número fixo de dígitos para facilitar a escrita e a comparação caso você prefira armazenar as strings e não inteiros (Veja as dicas).

Caso o vetor não esteja ordenado, você deve emitir a mensagem `"Vetor nao estah ordenado"` e interromper o processo.

Os passos dos algoritmos serão ilustrados como no exemplo da primeira seção. A moldura do elemento central conterá caracteres `=` na parte de cima e na parte de baixo. Nas laterais, teremos duas barras, sem espaço em branco.

```
+=====+
||008||
+=====+
```

O visualizador irá mostrar apenas os elementos que estão participando da busca em cada passo. Note que precisaremos imprimir espaços em branco à esquerda, mas não haverá espaços em branco após o último elemento. Observe os passos da busca do elemento 4 no vetor fornecido.

```
Elemento procurado: 004
+-----+-----+-----+-----+-----+-----+-----+
| 002 | 003 | 004 | 008 | 011 | 015 | 017 |
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+=====+-----+-----+-----+
| 002 | 003 | 004 ||008|| 011 | 015 | 017 |
+-----+-----+-----+=====+-----+-----+-----+
+-----+=====+-----+
| 002 ||003|| 004 |
+-----+=====+-----+
            +=====+
            ||004||
            +=====+
```

A última parte da saída será indicar a posição do elemento, caso encontrado, com a string `"O elemento estah na posicao <pos>"`. Caso o elemento não esteja no vetor, a string final será `"O elemento nao foi encontrado"`.

Dicas de Python 3 para esta tarefa:

- Para escrever um inteiro com um número fixo de casas decimais, você pode usar a função `zfill` Veja o exemplo:

```
>>> str(9).zfill(3)
'009'
```

- Você já deve ter visto que a comparação de strings contendo números nem sempre retorna um resultado coerente com o esperado na matemática.

```
>>> '20' < '100'
False
```

  No entanto, se estivermos trabalhando com o mesmo número de casas decimais, o resultado será sempre coerente.

```
>>> '020' < '100'
True
```

  Você sabe explicar a razão?

Testes para o SuSy

Esta tarefa contém 13 testes abertos, exercitando vários caminhos de busca. Contém também um teste fechado em que uma chave está no vetor e outro em que não está.

Orientações para submissão

Veja aqui a página de submissão da tarefa. Lembre-se que o arquivo a ser submetido deve se chamar **`lab11.py`**.

No link Arquivos auxiliares há um arquivo `aux-11.zip` que contém todos os arquivos de testes abertos e seus respectivos resultados compactados. Os arquivos `executa-testes.py` e `executa-testes-windows.py` também estão neste pacote.

Observe o limite máximo de 20 submissões.

A nota final é proporcional ao número de testes que executaram corretamente, desde que o código esteja coerente com o enunciado. **A submissão de um código que não implementa o algoritmo requisitado, mas que exibe as saídas esperadas dos testes abertos a partir da comparação de trechos da entrada será considerada fraude e acarretará a atribuição de nota zero à média final da disciplina.**

O peso desta tarefa é 2. Como é uma tarefa **opcional**, esta nota entrará com peso 2 para o numerador do cálculo da média de tarefas de laboratório. O denominador será dado pela soma dos pesos das tarefas de número 0 a 10.

O prazo final para submissão é 01/12/2018 (último dia para o cumprimento da carga horária/programa das disciplinas). Recomenda-se a realização desta tarefa antes da Prova 2.
