Estruturas de Dados

lista de exercícios 21

1. Operação de atualização

   Imagine que `A[1..n]` é uma multilista.

   Dados dois números x, y, a tarefa consiste em

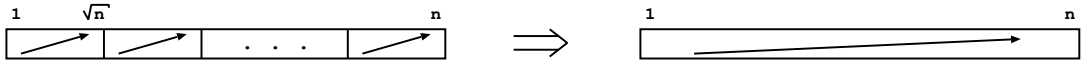
   - localizar a posição do número x na multilista
   - modificar o se valor para y — (*se ele estiver lá*)
   - e deslocar o número y para a sua posição correta — (*se necessário*)

   Apresente um algoritmo que realiza essa tarefa.

   E depois analise o tempo de execução do seu algoritmo.

2. Ordenação de multilistas

   Considere a tarefa de transformar uma multilista em uma lista completamente ordenada.



   Apresente um algoritmo que realiza essa tarefa.

   E depois estime o tempo de execução do seu algoritmo.

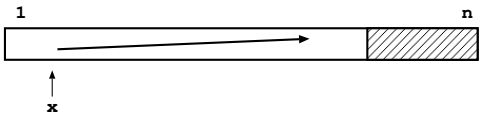
   Dica: Faça chamadas sucessivas ao procedimento de intercalação.

   Fazendo as coisas de maneira esperta, é possível ordenar a lista em tempo menor que $O(n \cdot \sqrt{n})$.

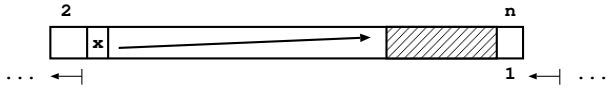
   Você consegue fazer isso?

3. Listas circulares

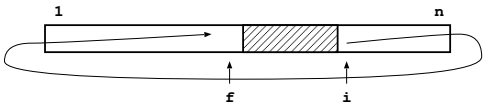
   Nesse exercício, nós vamos ver uma solução alternativa para o problema de inserir um novo elemento `x` no início de uma lista ordenada.



   A ideia consiste em deslocar os elementos menores que `x` para frente, jogando o primeiro elemento para o final da lista, e inserir o elemento `x`:



Em geral, após fazer esse tipo de coisa várias vezes, o início da lista não está mais no começo, e o final da lista não está mais no fim:



Isto é, o início e o final da lista passam a ser indicados por duas variáveis `i`,`f`, que não precisam estar nas posições 1 e n, respectivamente.

Escreva programas que realizam as seguinte operações na lista circular

a) busca binária b) inserção c) remoção

4. O problema da mediana

Lembre que a *mediana* de um conjunto A de tamanho n é aquele elemento que

- *é menor do que outros* $n/2$ *elementos*
- *é maior do que outros* $n/2 - 1$ *elementos*

Por exemplo, se

$$\mathsf{A} \quad = \quad \{\,1,\ 3,\ 6,\ 7,\ 10,\ 11,\ 15,\ 17,\ 19,\ 21\,\}$$

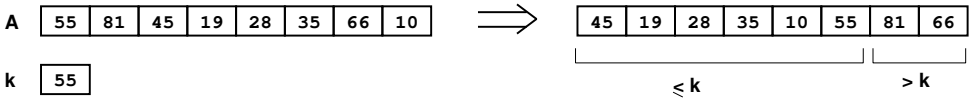
a mediana do conjunto A é o elemento 10.

Certo.

Nesse exercício, nós vamos ver uma estratégia para encontrar da lista `A[1..n]`

A ideia é a seguinte.

Aplicando o procedimento de partição sobre a lista (que vimos na aula 02), utilizando o primeiro elemento como *pivô* k



nós observamos que a mediana certamente está no maior lado da partição — (*ou do lado esquerdo, caso os dois lados tenham o mesmo tamanho*)

A ideia, então, é continuar a busca pela mediana fazendo uma nova partição no lado em que ela se encontra, reduzindo ainda mais a lista de candidatos, até que reste apenas a mediana.

a) Apresente um programa que implementa esse método para encontrar a mediana da lista.

b) Assumindo que em cada etapa (aprox.) 1/4 dos canditatos atuais são eliminados, estime o número de comparações realizados pelo seu programa.

5. Lista Sudoku

A lista Sudoku é um arranjo quadrado de sublistas onde

- todas as linhas estão organizadas em ordem crescente
- e todas as colunas estão organizados em ordem crescente

Por exemplo,

2	5	11	55
7	15	42	58
21	31	68	90
23	70	83	99

a) Apresente um programa que realiza a operação de busca na lista Sudoku.

Nota: A ideia é que o seu programa seja mais rápido do que a busca na multilista — (*pois a lista Sudoku possui mais estrutura que a multilista*)

b) Apresente um programa que constrói uma lista Sudoku com os elementos de uma lista desordenada.

6. Espalhamento de posições vazias

Imagine que `L[1..n]` é uma lista esparsa com 10% de posições vazias.

E imagine que, por alguma razão, existem grandes regiões da lista onde não se encontra quase nenhuma posição vazia.

E existem outras regiões da lista com uma grande quantidade de posições vazias.

Nessas condições, nós não podemos mais garantir um desempenho eficiente das operações de inserção e remoção.

A solução, então, é redistribuir as posições vazias por toda a lista, para restaurar as boas propriedades da lista esparsa.

a) Apresente um algoritmo que reorganiza a lista de modo que exista uma posição vazia a cada 10 posições da lista.

b) E depois analise o tempo de execução do seu algoritmo.

7. Intercalação de listas esparsas

Imagine que `A[1..n]` e `B[1..n]` são duas listas esparsas, ambas com 10% de posições vazias.

A tarefa consiste em

→ *Combinar os elementos de* `A` *e* `B` *em uma única lista esparsa* `C[1..2n]`*, com* 10% *de posições vazias (bem distribuídas)*

Apresente um algoritmo que realiza essa tarefa.

E depois analise o tempo de execução do seu algoritmo.