Estruturas de Dados

lista de exercícios 03

1. O terceiro maior elemento

   Imagine que nós temos duas listas U e W com tamanho n, onde os elementos são todos distintos. Por exemplo,

U	9	19	3	26	12	7	18	23	14	10
W	11	7	8	16	21	2	4	28	15	24

   A tarefa consiste em

   → *Colocar os n menores elementos na lista* U
   e os n maiores elementos na lista W

   No exemplo acima, isso nos daria

U	9	3	12	7	10	11	7	8	2	4
W	19	26	18	23	14	16	21	28	15	24

   — (*a ordem em que os elementos aparecem nas listas* U *e* W *não é importante*)

   Apresente um algoritmo que realiza essa tarefa.

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

2. Tri-partição

   Imagine que nós temos uma lista não ordenada L

L	2	21	10	5	13	18	11	1	9	26

   A tarefa consiste em

   → *Dados dois números* `x` *e* `y`*, onde* $x < y$

   - *colocar os elementos menores do que* `x` *no início da lista*
   - *colocar os elementos ente* `x` *e* `y` *no meio da lista*
   - *colocar os elementos maiores do que* `x` *no fim da lista*

   No exemplo acima, isso nos daria

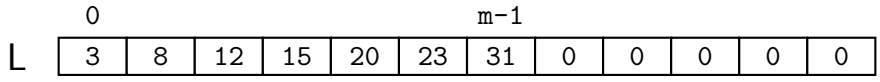
L	2	5	1	9	10	13	18	11	21	26

   a) Apresente um algoritmo que realiza essa tarefa em tempo $O(n)$ utilizando uma lista auxiliar.

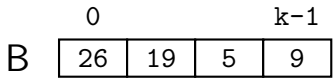
   b) Apresente um algoritmo que realiza essa tarefa em tempo $O(n)$ sem utilizar uma lista auxiliar.

3. Inserções em bloco

Imagine que nós temos uma lista ordenada que contém m elementos e posições vazias no fim



E imagine que nós temos uma outra lista com k elementos



A ideia é que k é muito menor do que m, e que k também é menor que (ou igual a) o número de posições vazias na lista L.

A tarefa consiste em

→ *Inserir todos os elementos de* B *na lista* L

Apresente um algoritmo que realiza essa tarefa.

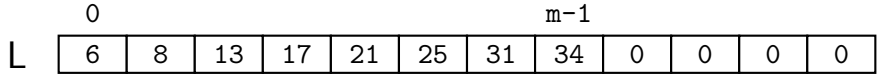
E depois analise o tempo de execução do seu algoritmo.

Dica: Você pode reorganizar os elementos da lista B.

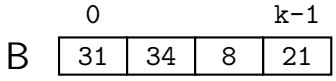
E também pode utilizar uma lista auxiliar.

4. Remoções em bloco

Imagine que nós temos uma lista ordenada que contém m elementos e posições vazias no fim



E imagine que nós temos uma outra lista com k elementos, que estão na lista L



A ideia é que k é muito menor do que m.

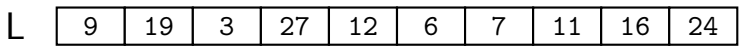
A tarefa consiste em

→ *Remover todos os elementos de* B *da lista* L

Apresente um algoritmo que realiza essa tarefa em tempo $O(m)$, sem utilizar uma lista auxiliar.

5. Par com soma k

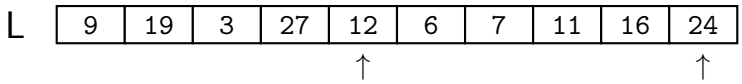
Imagine que nós temos uma lista não ordenada L



A tarefa consiste em

→ *Encontrar um par de elementos da lista cuja soma é igual a* `k`

No exemplo acima, para `k` = 32, isso nos poderia no dar



a) Apresente um algoritmo que realiza essa tarefa.

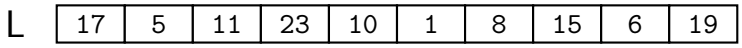
   E depois analise o tempo de execução do seu algoritmo.

b) Agora imagine que a lista está ordenada.

   Você consegue escrever um algoritmo mais rápido para essa tarefa?

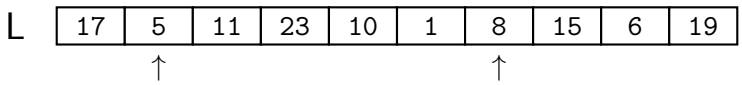
6. Contando inversões

Imagine que nós temos uma lista não ordenada L



Daí, como a lista está desordenada, nós sempre podemos encontrar dois elementos fora de ordem — (*i.e., o elemento da esquerda é maior do que o elemento da direita*)

Por exemplo,



Nós dizemos que um par desse tipo corresponde a uma *inversão*.

A tarefa desse problema consiste em

→ *Contar o número total de inversões na lista* L

No exemplo acima, isso nos daria

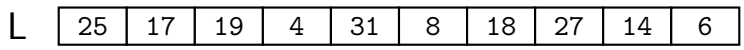
```
=>  23 inversões no total
```

Apresente um algoritmo que resolve esse problema.

E depois analise o tempo de execução do seu algoritmo.

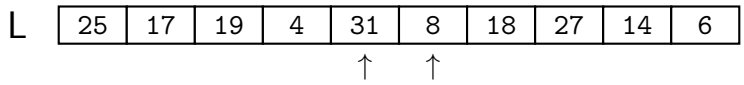
7. A ordenação da bolha

Imagine que nós temos uma lista não ordenada L



Daí, como a lista não está ordenada, nós sempre podemos encontrar uma inversão envolvendo dois elementos consecutivos de L — (*porque?*)

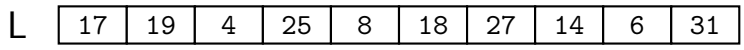
Por exemplo,



Agora, imagine que nós percorremos a lista L da esquerda para a direita, desfazendo todas as inversões entre elementos consecutivos que a gente encontra no caminho — (*i.e., trocando os dois elementos de lugar*)

Esse procedimento é chamado de *varredura*.

Abaixo nós temos o resultado da varredura no nosso exemplo



Como se pode ver, a varredura não deixa a lista ordenada.

Mas, aplicando varreduras sucessivas a lista fica eventualmente ordenada — (*porque?*)

Apresente um algoritmo que ordena a lista L por meio de varreduras sucessivas.

E depois analise o tempo de execução do seu algoritmo.