Ordenação de Lista Bitônica

Autoria: Emanuel Catão

Um tipo interessante de sequência numérica é chamado de **Sequência Bitônica**. Uma sequência pode ser dita monotonicamente crescente quando ela é equivalente a uma sequência não-decrescente, ou seja, seus elementos nunca diminuem, mas podem ficar iguais. Uma sequência também pode ser dita monotonicamente decrescente, cuja definição é análoga ao tipo anterior. Já uma **sequência bitônica**, é uma sequência monotonicamente crescente seguida de uma monotonicamente decrescente (ou uma rotação cíclica dessa sequência).

> {1, 2, 3, 7, 6, 4} é uma sequência bitônica, formada pela sequência monotonicamente crescente {1, 2, 3} e pela monotonicamente decrescente {7, 6, 4}.
>
> {3, 7, 6, 4, 1, 2} é uma rotação cíclica da sequência original, então ela é também bitônica.

Para nosso problema, vamos assumir somente a primeira parte da definição, ou seja, trabalharemos com sequências bitônicas sem a parte da rotação cíclica. Pede-se que seu programa construa uma lista duplamente encadeada a partir da lista bitônica fornecida na entrada e imprima uma lista ordenada, partindo da lista criada e seguindo o seguinte método:

Exemplo de lista: [1, 2, 4, 3]

- Mantenha os dois ponteiros: inicio e fim da lista. Do exemplo: (inicio = 1, fim = 3)
- Compare ambos os valores dos nós e adicione o elemento menor para resultar em uma lista. Do exemplo: (inicio < fim)
- Compare o próximo elemento adjacente ao elemento adicionado com aquele que não foi adicionado e adicione o menor (*inicio->prox*, se inicio < fim; *fim->ant*, se fim < inicio).
    - Do exemplo: inicio é adicionado; compare (inicio->prox = 2, fim = 3).
    - inicio->prox é adicionado
- Repita isso até que todos os elementos da lista duplamente encadeada criada a partir da entrada sejam adicionados ao resultado da ordenação.
    - compare (inicio->prox->prox = 4, fim = 3).
    - fim é adicionado
    - compare (inicio->prox->prox, fim->ant)
    - inicio->prox->prox é adicionado
    - Todos os elementos foram adicionados. Lista final: [1, 2, 3, 4]

Obs: Perceba que toda vez que um nó for adicionado a lista de resultado, suas referências devem ser apagadas da lista original de modo que ao final de tudo a lista

original está vazia. Isso permite que se entenda onde o programa encerra as iterações.

Entrada: Cada entrada é composta por uma lista de valores (separados por um espaço em branco)

Saída: Deve-se retornar a lista ordenada valendo-se do método explicado anteriormente. O formato está expresso nos exemplos de entrada e saída.

Exemplo de entrada 1:	**Exemplo de saída 1:**
1 2 3 4 5 6	<- 1 -><- 2 -><- 3 -><- 4 -><- 5 -><- 6 ->
Exemplo de entrada 2:	**Exemplo de saída 2:**
1 5 7 6 3 2	<- 1 -><- 2 -><- 3 -><- 5 -><- 6 -><- 7 ->