MO417 - Ata de Exercício

Pedro Henrique Del Bianco Hokama

22 de abril de 2010

Exercício 15.4-5 - Subsequência monotonicamente crescente

Enunciado

Forneça um algoritmo de tempo $O(n^2)$ para encontrar a subsequência monotonicamente crescente[1] mais longa de uma sequência de n números.

Resolução

A idéia utilizada nesse algoritmo é utilizar o Algoritmo de *Longest common subsequence (LCS)*, entre a sequência original e uma cópia ordenada dela, para encontrar a subsequência monotonicamente crescente mais longa.

Algoritmo 1: LONGEST-INCREASING-SUBSEQUENCE(X)

1. $Y \leftarrow X$; ▷ Y recebe uma cópia de X
2. Ordena(Y);
3. **retorna** LCS (X, Y);

O algorítmo LONGEST-INCREASING-SUBSEQUENCE é $O(n^2)$, faremos a análise linha a linha.

- Linha 1 $Y \leftarrow X$;

  Apenas copia a sequência, o que pode ser feito em tempo $O(n)$

[1] Pela definição dada na sessão 3.2 do livro texto, uma função $f(n)$ é monotonicamente crescente se $m \leq n$ implica que $f(m) \leq f(n)$

- Linha 2 Ordena(Y);

  Nessa parte precisamos ordenar a sequência Y utilizando um algoritmo $O(n^2)$, vamos considerar o HeapSort que é $O(n \lg n)$

- Linha 3 LCS (X, Y);

  Nessa parte utilizamos o algoritmo LCS que executa em tempo $O(|X||Y|)$; como $|X| = |Y| = n$, a linha 3 leva tempo $O(n^2)$

Como cada linha é executada uma única vez, a complexidade total do algoritmo LONGEST-INCREASING-SUBSEQUENCE é $O(n+n \lg n+n^2) = O(n^2)$