Análise sobre o algoritmo Gnomesort

Victor de Sá Nunes

30/08/2017

1 Introdução

Neste trabalho será explicado o funcionamento do algoritmo de ordenação GnomeSort e sua complexidade será comentada.

2 Funcionamento do algoritmo

O algoritmo é similar ao InsertionSort. Ele verifica se o elemento da posição atual é maior do que o elemento da posição anterior. Caso seja, deve-se avançar uma posição à direita. Caso contrário, deve-se trocar os elementos de posição e andar uma posição a esquerda no vetor.

Caso a posição atual seja a primeira, deve-se avançar uma posição a direita. Os passos do algoritmo são os seguintes:

1. Se estiver na primeira posição, avance uma posição a direita
2. Se o elemento atual é maior do que o elemento anterior, avance uma posição a direita
3. Se o elemento atual é menor do que o anterior, então troque-os e volte uma posição no vetor
4. Repita os passos 2 e 3 até se atingir o final do vetor
5. Se chegar ao final do vetor, este estará ordenado

A procedimento que ordena o vetor é apresentado a seguir:

```
void gnomeSort(int array[], int size){
    int i, aux;
    i = 1;
    while(i < size){
        if(array[i] >= array[i-1]){
            i++;
        }
        else{
            aux = array[i];
            array[i] = array[i-1];
            array[i-1] = aux;
            i--;
        }

    }
}
```

3 Complexidade

Embora não tenha *loops* aninhados o algoritmo possui complexidade quadrática. Isto acontece pois a variável i nem sempre será incrementada. Em alguns momentos ela pode ser decrementada. Entretanto, o algoritmo é bom para dados quase ordenados. Em relação a memória, o algoritmo possui complexidade constante. Os dados são ordenados no próprio vetor, não sendo necessário memória extra.

4 Conclusões

O algoritmo é semelhante ao algoritmo InsertionSort, já bastante estudado na literatura. Por possuir complexidade quadrática o algoritmo é bom para base de dados quase ordenados. Entretanto, como a complexidade da memória é constante, ele é bom para aplicações com memória limitada. Alémd disso, sua implementação também é bastante simples.

5 Código Fonte

Programa 1: Gnome Sort escrito em C

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void gnomeSort(int array[], int size){
    int i, aux;
    i = 1;
    while(i < size){
        if(array[i] >= array[i-1]){
            i++;
        }
        else{
            aux = array[i];
            array[i] = array[i-1];
            array[i-1] = aux;
            i--;
        }

    }
}

void generateRandomArray(int array[], int n){
        int i;
        for(i = 0; i < n; i++){
                array[i] = rand() % 1001;
        }
}

void printArray(int array[], int size){
        int i;
        for(i = 0; i < size; i++){
                printf("%d ", array[i]);
        }

}
void main(){
        srand(1);
        int n = 10;
        int array[n];
        //int array[n] = {5, 8, 12, 4, 3, 2, 21, 10, 16, 9};
        generateRandomArray(array, n);
        printf("Original array: ");
        printArray(array, n);
        gnomeSort(array, n);
        printf("\n\nSorted array: ");
        printArray(array, n);

}
```