Universidade de Brasília
Instituto de Ciências Exatas

Departamento de Ciência da Computação

117366 - Lógica Computacional 1 Turma: B

Relatório sobre o **Algoritmo RadixSort**

Otávio Souza de Oliveira - 15/0143401
15/0126298 - Gabriel dos Santos Martins
160052289 - Lucas Ribas

18 de novembro de 2019

1 Introdução

Nesso projeto mostramos através da ferramenta PVS, o que está por trás do algoritmo radixsort, no sentido lógico de sua construção. Mas para isso ocorrer primeiramente, radixsort necessita de um algoritmo de ordenação auxiliar, e que seja estável; nesse caso foi usado o algotimo de ordenação mergesort.

Como dito mergesort é um algoritmo estável e usado como auxiliar no algoritmo radixsort. Porém tivemos que demonstrar essa estabilidade dele, mostrando que ele preserva o conteúdo de uma lista qualquer sendo ordenada por ele mesmo. E a partir dessas definições mostrar que radixsort é capaz de ordenar uma lista qualquer com respeito a ordem lexicográfica construída.

2 Apresentação do Problema

O problema consiste em demonstratar formalmente a funcionalidade do algoritmo radixsort sobre uma lista qualquer, para isso é usado um algoritmo de ordenação auxiliar o mergesort.

Mas primeiramente o que é radixsort? Radixsort é um algoritmo recursivo que utiliza de outro algoritmo de ordenação para aprensetar uma lista de elementos qualquer ordenada. E utilizando esse algoritmo auxilar (mergesort), ele ordena cada elemento da lista comparando primeiro as unidades de cada elemento dessa mesma lista.

Em seguida, o processo é repetido para as unidades seguintes que ainda não foram comparadas desses elementos. Lembrando que a ordenação é feita a partir da comparação das unidades dos elementos da lista, até as ultimas unidades sejam comparadas. Depois disso, a lista de elementos estará ordenada corretamente, dada uma ordem qualquer. Segue abaixo um exemplo:

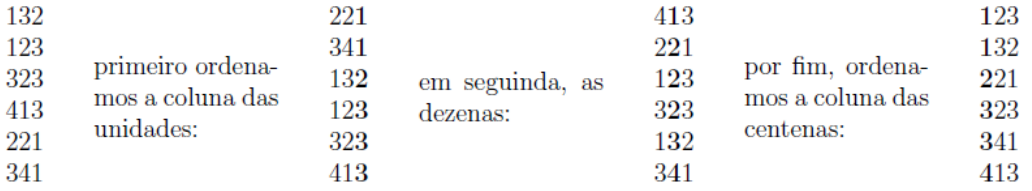


Figura 1: RadixSort

E o que é mergesort? Mergesort numa linguagem popular é definido como "dividir para conquistar", é também um algoritmo recursivo, e possui 3 passos para definir sua funcionalidade, que são:

1. Dividir: Fazer divisão dos dados da lista em subsequência;
2. Conquistar: Classificação das duas metades de uma lista, aplicando mergesort recursivamente.
3. Combinar: Juntar as duas metades ordenadas, em uma única lista classificada.

Porque a utilização de um algoritmo de ordenação estável? Isso pois, um algoritmo estável garante que ao ordenar uma lista de entrada qualquer,

que não haverá perde de elementos nessa lista, assim como manter a ordem de elementos que não necessitam de ordenação, ou seja, elementos iguais, preservando assim a ordem natural dos elementos.

Exemplo:

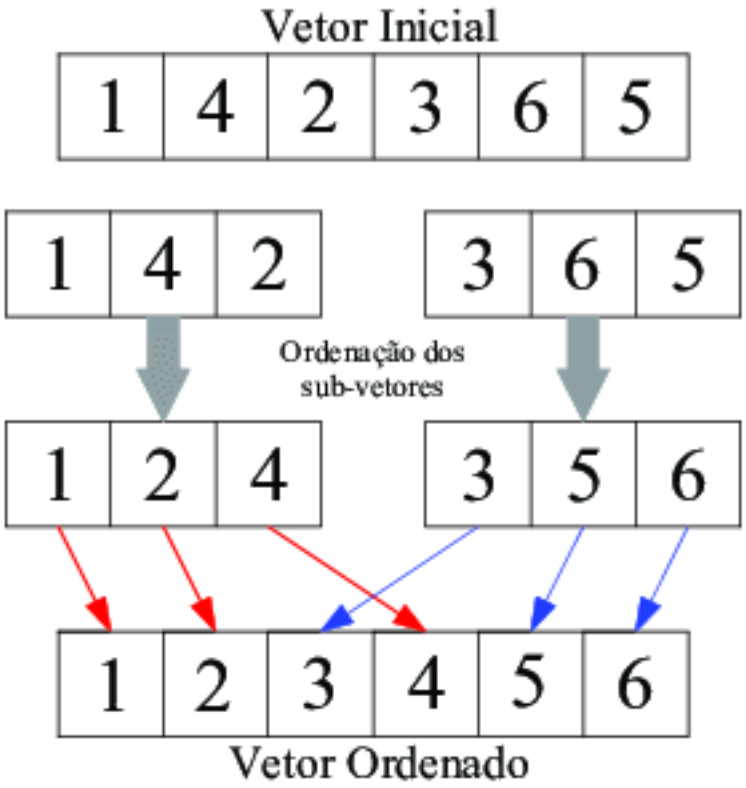


Figura 2: MergeSort

2.1 Análise assintótica

Nossa prova foi baseada na aplicação de indução no tamanho da nossa lista presente na estrutura de merge sort, que é o algoritmo usado na ordenação dos elementos de RadixSort.

Pois a partir dessa definição, garantiríamos o fato da estabilidade de merge sort, e assim facilitária na aplicação desse algoritmo que faz parte

da definição dada de radixsort.

2.2 Correção da solução proposta

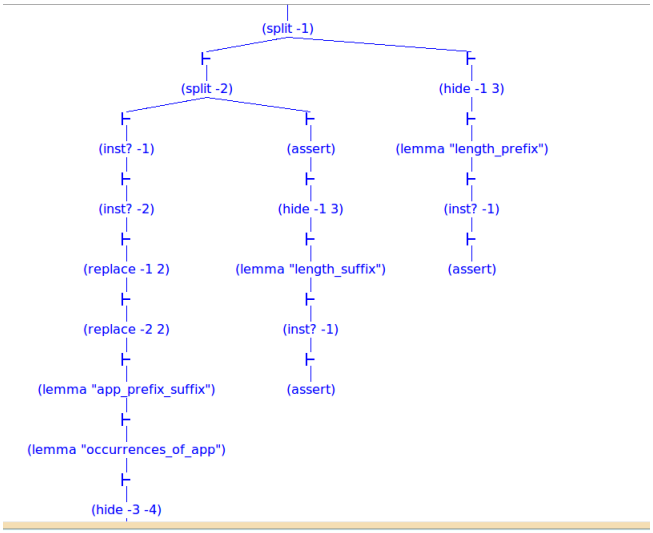
Nossa primeira prova era mostrar que merge sort preserva o conteúdo da lista sendo ordenada, ou seja, mostrar a estabilidade do merge sort.

Ao aplicadar indução no tamanho da lista de merge sort, construimos nossa prova baseada em mostrar que de fato o número de elementos iniciais da lista, continuava igual ao aplicar o algoritmo merge sort. Isso pois já mostrado na figura 2, merge sort divide uma lista inicial em outras duas, definidas por prefixo e sufixo. Logo, se mostrar-mos que não haverá perde de elementos nesse processo; pode-se dizer que existe estabilidade nesse algoritmo.

A segunda prova era mostrar que a função radixsort produz uma permutação da lista de entrada. Como a estrutura de ordenação de radixsort, utiliza o merge sort para ordenar seus elementos; de fato utilizamos disso para mostrar que radix permuta seus elementos, baseado na ordenação feita por merge sort.

2.3 Exemplos de Aplicação da Prova no PVS

Após uma explicação detalhada do problema e de sua solução proposta, iremos mostrar em termos gráficos como foi a experiência de usar uma aplicação para provar teoremas, no caso, o PVS. Não querendo entrar em detalhes técnicos, exemplificamos abaixo com é a árvore gerada pelos comandos de prova executados no PVS. Assim podemos ter uma idéia de como é usar uma aplicação de provas.



A estrutura é bastante familiar a que usamos em uma prova no "papel", isso torna a experiência muito mais fácil de assimilar e de praticar, deixando o trabalho apenas de aprender os comandos.

3 Conclusão

Com esse projeto podemos perceber a complexidade que algoritmos de ordenação nos oferece em determinados problemas, e a real necessidade de analisá-los adequadamente para tais problemas.

A ultilização da ferramenta PVS auxília bastante esse processo de análise da estrutura de tais algoritmos, vide que muitas delas foram dadas através dos lemas já criados; possibilitando também no acréscimo das mesmas.

As dificuldades encontradas foram justamente relacionadas aos lemas; no aspecto de saber utiliza-lás de maneira correta dentro de cada estrutura a ser provada. Após isso o processo é facilitado, usando adequadamente os comandos de prova da ferramenta PVS; que aplica as definições a cada estrutura da prova.