Prova do Funcionamento do Radix Sort

Yuri Crystian Ribeiro e Silva, 18/0029479
Luis Filipe Siqueira Ribeiro, 18/0053906
Gabriel Oliveira do Espírito Santo, 18/0041835

1 Introdução

Este projeto trata de provar o correto funcionamento do algoritmo de ordenação *Radix Sort*, utilizando como sub-rotina de ordenação o *Merge Sort*.

Isto, é, devemos provar que:

1. *Merge Sort* tem como resultado uma permutação da lista a ser ordenada.
2. *Radix Sort* é uma permutação da lista a ser ordenada.
3. *Radix Sort* de fato ordena.

Para isso será utilizado o *Prototype Verification System*, um *software* específico para provas matemáticas para algoritmos de computadores.

2 Radix Sort

O *Radix Sort* é um algoritmo de ordenação não-comparativo estável - um algoritmo estável ao ordenar dois elementos iguais mantém a ordem original entre eles - que ordena um grupo de números analisando-os dígito a dígito do menos significativo ao mais significativo e então aplica uma sub-rotina para ordená-los.

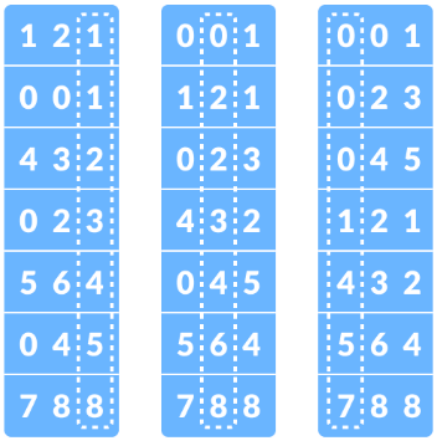

Figure 1: Funcionamento do Radix Sort

Dependendo da sub-rotina utilizada, como o *Counting Sort*, por exemplo, pode-se obter uma ordenação em tempo linear.

A sub-rotina escolhida para a ordenação dos dígitos individuais, neste projeto, foi o *Merge Sort*.

2.1 Merge Sort

O *Merge Sort* é um algoritmo de ordenação baseado em comparações que produz uma ordenação estável, criado em 1945 por John Von Neumann.

Este algoritmo usa a técnica de Dividir para Conquistar, ou seja, dividir um problema maior em partes menores para poder resolvê-los de forma mais fácil e, assim, juntar a solução para que o problema maior esteja resolvido.

Seu funcionamento se dá dividindo uma lista não ordenada em várias sub-listas menores, até obter varias listas que contenham apenas um elemento, pois uma lista de um único elemento é considerada ordenada, e depois une essas sub-listas (daí seu nome, pois *"merge"* é inglês para "fundir") de forma ordenada, através de comparações, até que retorne à lista original ordenada.

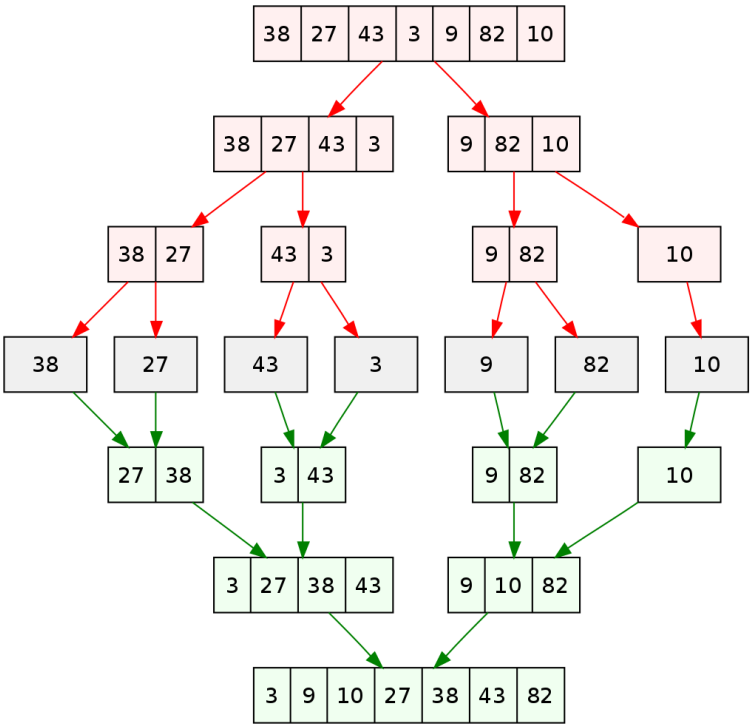

Figure 2: Funcionamento do Merge Sort

3 Prototype Verification System

O *Prototype Verification System* é uma linguagem desenvolvida no intuito de provar teoremas matemáticos, e foi desenvolvido na SRI International em Menlo Park, California. Utilizamos ele no trabalho para realizar as provas necessárias para comprovar o funcionamento do algoritmo.

4 Resolução

Para construir as provas necessárias para a verificação do funcionamento do *Radix Sort* o grupo se reuniu e analisou, juntamente, o problema proposto e utilizou de lemas já disponibilizados.

Para a prova do *Radix Sort* foi necessário provar as seguintes três conjecturas

4.1 Primeira Conjectura

A primeira questão conjectura que a sub-rotina *Merge Sort* tem como saída uma permutação da lista de entrada.

A definição de permutação se dá da seguinte forma: dadas duas listas, para cada elemento de uma lista, as ocorrências desse elemento na lista igual as ocorrências deste mesmo elemento na outra lista. Ou seja, os mesmos elementos estão presentes em ambas as listas, não necessariamente na mesma ordem.

Dada esta definição, percebemos que o que queremos provar é que ao realizar um *Merge Sort* em uma lista mantêm-se os mesmos elementos na nova lista, porém, com ordem possivelmente diferente. Isto é um tanto quanto lógico, afinal, caso a lista original não esteja ordenada, a nova lista estaria.

Para conseguir tal prova, devemos, de maneira simplificada, provar que:

As ocorrências de um elemento x na junção do prefixo de uma lista **L** com seu sufixo é igual às ocorrências de **x** no prefixo de **L** somado às ocorrências no sufixo, que, por fim, é igual às ocorrências de **x** na lista **L**

Para tal, ocorreram alguns momentos chave que serão discutidos a seguir.

4.1.1 Ocorrências na Junção de Duas Listas

Após a expansão de algumas definições, obteve-se um consequente que dizia:

As ocorrências de um elemento **x**
na junção do *merge sort* do prefixo de uma lista **L**
com o *merge sort* do sufixo desta mesma lista
é igual a ocorrência de **x** na lista **L**.

A partir daí aplicou-se um lema que garante que as ocorrências de um elemento **x** na junção de duas listas **L1** e **L2** é igual as ocorrências de **x** em **L1** somado às ocorrências de **x** em **L2**.

Conseguindo assim instanciar o consequente no antecedente, a lista **L1** assume o prefixo de **L**, e **L2** assume o sufixo, obtendo-se então que as ocorrências de um elemento **x** na junção do *merge sort* do prefixo de **L** com o *merge sort* do sufixo de **L** é igual a soma das ocorrências de **x** nessas listas individualmente.

4.1.2 A Junção do Prefixo e do Sufixo de uma Lista Resulta na Lista

Chegando ao ponto em que temos que a ocorrência de um elemento no prefixo de uma lista somado a ocorrência do mesmo elemento no sufixo da lista é igual à ocorrência do elemento na lista, usamos então um lema que diz:

Uma lista é igual a junção de seu prefixo com seu sufixo.

Instanciando o consequente no antecedente e fazendo a devida substituição, obtém-se uma igualdade que garante, enfim, a prova da conjectura.

4.2 Segunda Conjectura

A segunda questão conjectura, de forma similar à primeira (seção 4.1), que o *Radix Sort* faz uma permutação dos dados de entrada.

Sendo assim, que ao executar um *Radix Sort* em uma dada lista, o resultado obtido será a mesma lista, mas com os elementos em ordem possivelmente diferente.

Ao expandir as definições de permutação e de *radix sort* obtemos como consequente:

A ocorrência de um elemento no *merge sort* do *merge sort* de uma lista é igual a ocorrência desse elemento na lista.

Ou seja, que o *merge sort* do *merge sort* de uma lista é uma permutação dessa lista.

Usando a primeira conjectura provada (4.1) podemos provar isso, visto que ela nos garante que o *merge sort* de uma lista é uma permutação desta dada lista.

4.3 Terceira Conjectura

A terceira questão conjectura que o *Radix Sort* realmente ordena.

Para tal, começamos por aplicar a definição de *radix sort* e de ordenação. Esta última diz que para cada elemento de uma lista, este deve ser lexicograficamente menor que o elemento depois dele.

Então obtemos que um elemento no *merge sort* de um *merge sort* deve ser lexicograficamente menor que o elemento próximo a ele.

Usando o lema que nos garante que *Merge Sort* ordena e fazendo as devidas instanciações chegamos ao ponto que devemos mostrar que os elementos são lexicograficamente menores que os próximos.

Ao aplicar a definição de *lexicográfico* percebemos que essa diz que um elemento deve ser menor ou igual o elemento que vem após, mas nunca maior.

Usamos então o lema que garante que o *Merge Sort* é um algoritmo estável, isto é, que para dois elementos iguais, após a ordenação, suas posições relativas não serão trocadas em relação à lista original.

Foi usado então mais um lema, que diz que, para todo elemento de uma lista ordenada, todos os elementos anteriores a este são menores ou iguais.

Basta então fazer a correta instanciação e aplicar novamente o lema que garante que o *merge sort* ordena e a prova se dá por completa.

5 Conclusões

Pode-se perceber por meio deste projeto a importância da linguagem PVS, que pode ser usada para provar o correto funcionamento de um algoritmo, evitando assim qualquer erro inesperado.

6 Referências

1. Radix Sort - Wikipedia
2. Merge Sort - Wikipedia
3. Radix Sort Algorithm