Universidade de Brasília
Instituto de Ciências Exatas

Departamento de Ciência da Computação

Projeto e Análise de Algoritmos

Prova acerca da Complexidade de Bubblesort

Danilo Santos - 14/0135910
Anne Gontijo - 14/0016546

6 de dezembro de 2019

1 Introdução

O projeto de algoritmos embora possa ser guiado por boas práticas e conhecimento prévios, é um processo criativo, sendo assim, não temos certeza a princípio, se o algoritmos desenvolvido, de fato resolve o problema para todos os casos, ou se mesmo resolvendo, os problemas, se o faz em tempo útil. Diante destas questões é necessários verificar de maneira sistemática, formal, se o algoritmo cumpre estes requisitos.

2 Descrição do Problema

Neste projeto, foi executada a prova formal sobre a complexidade e corretude de um algoritmo de ordenação por comparação de chaves, o Bubblesort. Utilizando o assitente de prova, PVS, buscou-se provar que o Bubblesort, pertence a classe de complexidade de tempo $O(n^2)$, e quanto a corretude, buscamos verificar as seguintes propriedades, que a lista ordenada é uma permutação dos elementos da lista inicial, e que a lista de saída está ordenada.

3 Estratégia Utilizadas

Para efetuar a prova de complexidade de tempo, se fez necessário construir uma função, de comportamento semelhante a originl, ou seja, realizando a ordenação

da lista, com a adição de um comportamento, a contagem da quantidade de comparações realizadas.

```
bubblesort_count_bubblesort_equiv: LEMMA
    FORALL (l:list[nat]):
        bubblesort(l) = count_bubblesort(l)`1
```

Isso se fez necessário, pois apenas assim, poderiamos mostrar ao assistente de prova, a relação entre a quantidade de elementos da lista, e a quantidade máxima de comparações possíveis de serem realizadas.

```
bubble_sort_is_quadratic: LEMMA
    FORALL (l:list[nat]):
        member(
            LAMBDA(n:nat): count_bubblesort(l)`2,
            Omicron(LAMBDA(n:nat):length(l)^2)
        )
```

Para que haja um prova completa da complexidade do Bubblesort, é necessário também provar a equivalência entre as funções *bubblesort* e $count_bubblesort$.

Para provar a corretude do algoritmo, foram utilzidas duas funções da biblioteca fornecida junto ao problema, em elas verificam a ordenação da lista, bem como se a lista obtida na saída é uma permutação da lista de entrada.

```
bubblesort_works: CONJECTURE
FORALL (l:list[nat]):
    sorted?(bubblesort(l)) AND permutations(l, bubblesort(l))
```

A possibidade de prosseguir com prova dessa maneira, facilita a execução; Primeiro por poder dividir em objetivos menores, e por não ter de efetuar a prova desse lemas auxiliaress

4 Dificuldades Encontradas

Embora tenhamos enunciado lemas auxiliares na prova da complexidade do bubblesort, não foi possível completar a prova da equivalênica entre *bubblesort* e $count_bubblesort$. A estratégia de indução-forte pareceu interessante, pois o algoritmo utiliza o tamanho da lista para tomar decisões, e isso possibilitou um desdobramento da prova, entretanto não foi possível finaliza-la. O mesmo ocorreu com a prova da correção do algoritmo; Embora tivesse a hipótese de indução a disposição, não foi possível encontrar a situação adequada para sua aplicação.

Referências