Universidade de Brasília
Instituto de Ciências Exatas

Departamento de Ciência da Computação

117536 - Projeto e Análise de Algoritmos
Turma: B

Relatório sobre Formalização da prova da complexidade temporal do algoritmo Bubblesort usando *PVS*

Thiago Pereira Martins - 18/0163876

5 de dezembro de 2019

1 Introdução

Este relatório descreve as etapas da formalização da prova da complexidade temporal do algoritmo Bubblesort usando o assistente de provas *PVS*. Bubblesort é um algoritmo de ordenação por comparação, seu funcionamento é de fácil entendimento, o que facilita tanto na implementação quanto na análise assintótica.

As provas mecânicas, como as realizadas através do *PVS*, são importantes em sistemas vitais, já que elas servem como garantia de que o software é capaz de realizar uma dada tarefa. Assistentes de prova também podem ser utilizadas durante o desenvolvimento de um software para previr o comportamento dele em relação a tempo de execução e espaço ocupado na memória.

2 Apresentação do Problema

A proposta apresentada ao projeto foi a de "formalizar a complexidade de tempo no pior caso, e se possível, a correção de um algoritmo em assistente de provas". A complexidade temporal pode ser entendida como o tempo que um dado algoritmo leva para ser executado. Seguindo a sugestão do professor, foi escolhido o algoritmo Bubblesort.

2.1 Análise assintótica

A implementação do Algoritmo Bubblesort analisada foi disponibilizada pelo Professor no GitHub, junto com outros arquivos bases para o desenvolvimento do projeto.

O Algoritmo Bubblesort é implementado com três funções. A função *bubblesort* serve como um wrapper para a função *bubblesort_aux*. Ela recebe como parâmetro uma lista l, composta por n elementos, verifica se a lista é válida e então chama a função *bubblesort_aux*. A função base para a ordenação chama-se *bubbling*, ela passa pela lista levando o maior elemento para a última posição. Essa última posição é um índice passado ao *bubbling* pela função que o chama. Esse o valor inicial desse indice é $n - 1$, em cada chamada posterior, esse índice recebe depreciação de 1, até que ele chegue a zero. Com isso, a complexidade dessa função é delimitada pelo tamanho total da lista. A função *bubblesort_aux* ordena a lista chamando recursivamente a si mesma e executando *bubbling* a cada chamada recursiva. Como o *bubling* possui complexidade temporal $O(n)$, a complexidade temporal do *bubblesort_aux* é $O(n^2)$. Por não possuir repetições, a complexidade de *bublesort* é a mesma de *bubblesort_aux*.

2.2 Correção da solução proposta

Baseado no código compartilhado para a execução da tarefa, foi realizada uma implementação de uma função que tem como finalidade proporcionar a contagem das comparações que o algoritmo realizará dentro da lista, no momento da prova. Essa contagem será importante justamente para a avaliação da complexidade temporal algoritmo.

Após a conclusão da implementação, foram definidos teoremas sobre os valores de retorno de cada uma das funções do algoritmo. Esses teoremas

são peças fundamentais para a conclusão da prova da complexidade temporal deduzida anteriormente.

A última etapa foi a utilização do *PVS* para realizar a prova do teorema que diz que a complexidade temporal do algoritmo BubbleSort é $O(n^2)$. A segunda e a terceira etapas foram as que demandaram mais esforço, já que por muitas vezes foi necessário modificar ou adicionar novos teoremas, assim como refazer as provas, fazendo com que o desenvolvimento do projeto não tenha sido exatamente sequencial

3 Conclusão

Apesar da prova não ter sido concluída, muito progresso foi feito na formalização da mesma. Com alguns teoremas provados e outros que por muito pouco não conseguiram ser provados a tempo da entrega.

Ao iniciar a atividade, foi observado que o site indicado para o download da ferramenta de prova possui muitos links "quebrados". Outra observação relevante de ser comentada é a limitação de funcionalidades nativas, falta de intuitividade e difícil aprendizado de utilização da ferramenta *Emacs*, que é necessária para execução do *PVS*.

O Projeto serviu para aprofundar o conhecimento em Analise de Algoritmos, Formalização de Provas, a linguagem LISP, a ferramenta GitHub, entre outros, assim como serviu como campo de testes para aplicar conceitos estudados ao longo do curso.