Universidade de Brasília
Instituto de Ciências Exatas

Departamento de Ciência da Computação

117536 - Projeto e Análise de Algoritmos
Turma: B

Relatório sobre ***Bubble Sort***

Fábio Oliveira Guimarães - 19/0099330
Geraldino Antônio da Silva - 13/0112267

5 de dezembro de 2019

1 Introdução

Uma das preocupações dos projetistas, tanto de *Hardware* quanto de *Software* é garantir que os seus produtos sejam corretos. Para isso, os pesquisadores utilização uma formulação matemáticas, mais especificamente os métodos formais, para garantir a corretude dos seus produtos.

No entanto, a prova formal desenvolvida de forma manual, com lápis e papel, é suscetível a erros e é muito demorada. Para contornar este problema foi desenvolvido Assistentes de Prova, como o Coq e o PVS, para auxiliar nesse processo.

Este trabalho, foi desenvolvido em cima do PVS, software desenvolvido pela NASA. O PVS é utilizado pela NASA e pela AirBus para verificação dos programas de aviação.

2 Apresentação do Problema

Um dos problemas da Ciência da Computação é a ordenação de listas, vários algoritmos foram propostos ao longo dos anos e esse trabalho falará sobre o mais simples deles, e em contrapartida um dos menos eficiente, o *bubble sort*.

O seu funcionamento é bem simples, os elementos são comparados do inicio ao fim da lista, par a par, quando o elemento analisado é maior que o seu predecessor, eles são trocados de lugar. Essa comparação é realizada até o final da lista, quando o algoritmo chega ao fim da lista o processo é repetido, retirando o último elemento que após a primeira passagem é o maior número. A seguir, tem-se uma implementação do algoritmo *bubble sort* em C.

```c
//Algoritmo de ordenacao Bubble Sort em C

void bubble_sort (int vetor[], int n) {
    int k, j, aux;

    for (k = 1; k < n; k++) {
        printf("\n[%d] ", k);

        for (j = 0; j < n - 1; j++) {
            printf("%d, ", j);

            if (vetor[j] > vetor[j + 1]) {
                aux          = vetor[j];
                vetor[j]     = vetor[j + 1];
                vetor[j + 1] = aux;
            }
        }
    }
}
```

2.1 Análise assintótica

Avaliando a implementação do algoritmo *Bubble Sort*, do capítulo anterior, é possível notar que existe um *for* dentro de outro *for* e que o número de execuções de cada *for* é proporcional ao tamanho do vetor, ou seja é proporci-

onal a n. Com isso pode-se afirmar que o algoritmo é limitado superiormente a $O(n^2)$.

O algoritmo implementado não está otimizado, mas é o mais fácil para visualizar a sua complexidade.

2.2 Correção da solução proposta

A correção de qualquer algoritmo pode ser avaliada observando os seguintes aspectos: Se os laços são invariantes, se o programa termina e se o resultado final da execução retorna o que é esperado para o algoritmo.

Todos os três itens informados no paragrafo anterior foram provados no arquivo bubblesort.pvs, conforme códigos abaixo:

```
%bubbling mantem o tamanho
bubbling_preserves_length : LEMMA
FORALL  (l: list[nat], n :below[l`length]) :
  length(bubbling(l,n)) = length(l)

%bubbling invariante ao tamanho
bubbling_preserves_suffix : LEMMA
FORALL  (l: list[nat], n :below[l`length]) :
 LET l1 = bubbling(l,n) IN
   FORALL (i:below[l`length]) :  i > n IMPLIES nth(l1,i)
       = nth(l,i)
```

3 Conclusão

Conclui-se que o algoritmo de ordenação *bubble sort* é correto, ou seja ao final de sua execução mantém todos os elementos da lista, em ordem crescente e também possui a sua execução em tempo finito, limitado a $O(n^2)$, onde n é o tamanho da lista de elementos a serem ordenados.