Fundação Getulio Vargas

Estruturas de Dados e Algoritmos **Homework 2**

Prof. Jorge Poco **Due:**: March 22, 2019

Homework 2

Lucas Machado Moschen

1 LinkedList Improvements

As implementações necessárias encontram-se no arquivo "list.cpp". A função main() contém os valores postos no Homework. A linguagem padrão escolhida em Tools > Compiler Optios > Settings > Code Generation > Language standart: GNU C++ 11. ISO C++ 11 também funciona. Foram incluídos "iostream" e "exception" no escopo.

2 Correctness of bubblesort

Bubblesort is a popular, but inefficient, sorting algorithm. It works by repeatedly swapping adjacent elements that are out of order.

Algorithm 1: BUBBLESORT(A)

```
1 for i = 1 to A.length − 1 do
2 |   for j = A.length downto i + 1 do
3 |   |   if A[j] < A[j − 1] then
4 |   |   |   exchange A[j] with A[j − 1]
5 |   |   end
6 |   end
7 end
```

A **(0.5pt)** Let A' denote the output of BUBBLESORT(A). To prove that BUBBLESORT is correct, we need to prove that it terminates and that

$$A'[1] \leq A'[2] \leq ... \leq A'[n] \tag{1}$$

where $n = A.length$. In order to show that BUBBLESORT actually sorts, what else do we need to prove?

Resposta: Nós precisamos provar que A' possui todos os elementos de A, porém permutados. Observe que um algoritmo de ordenação poderia basicamente copiar um único elemento para todas as posições, o que não seria desejável. Logo, precisamos provar que A' não perde nenhum elemento de A.

The next two parts will prove inequality (1).

B **(1.0pt)** State precisely a loop invariant **for** the for loop in lines 2–6, and prove that this loop invariant holds. Your proof should use the structure of the loop invariant proof presented in this chapter.

Resposta: Inner Invariant (for loop 2-6): é assegurado que no final da iteração $n - j + 1$ (observe que é decrescente e a iteração 1 ocorre quando $j = n$), $A[j : n]$ contém os mesmos elementos de $A[j : n]$ inicial, embora possivelmente permutados, e que $A[j]$ é o menor valor da sublista $A[j : n]$. Agora, seguirei os passos para demonstração, como nos slides.

Inicialization: É trivial que na primeira iteração, $A[n : n] = A[n]$ contém todos os elementos dessa sublista de A. Também vê-se que é o valor mínimo, pois é único.

Maintenance: Suponha que esse loop invariant seja verdade na iteração $n - j + 1$. Assim $A[j : n]$ mantém os elementos e $A[j]$ é o menor deles. Considere a iteração $n - j + 2$. Se $j \leq i$, a iteração não ocorre, logo o loop invariant é satisfeito. Se $j > i$, Há dois casos:

- $A[j - 1] \leq A[j]$. Nesse caso, não há alteração. Como $A[j]$ é o menor elemento da sublista $A[j : n]$ e $A[j - 1] \leq A[j]$, tem-se que $A[j - 1]$ é o menor elemento da sublista $A[j - 1 : n]$. $A[j - 1 : n]$ manteve todos os elementos, pois $A[j - 1 : n] = A[j - 1]$ concatenado com $A[j : n]$.
- $A[j - 1] > A[j]$. Há alteração. Os elementos da lista $A[j - 1 : n]$ são mantidos, pois $A[j : n] =$ concatenação de $A[j - 1]$ que é copiado para o index j com $A[j + 1 : n]$ e o valor de $A[j]$ é copiado para o index $j - 1$. Também vê-se que $A[j]$ é o menor elemento, porém como ele é copiado para a posição $j - 1$, $A[j - 1]$ é o menor elemento da sublista $A[j - 1 : n]$.

Está provado o passo indutivo.

Termination: No final das iterações, quando $j = i$, $A[i]$ é o menor elemento da sublista $A[i : n]$ e os elementos da sublista $A[i : n]$ são os mesmos, possivelmente permutados, o que prova o invariante.

C **(1.0pt)** Using the termination condition of the loop invariant proved in part (B), state a loop invariant for the for loop in lines 1–7 that will allow you to prove inequality (1). Your proof should use the structure of the loop invariant proof presented in this chapter.

Resposta: Outer Invariant(for loop 1-7): é assegurado que no início da iteração i, a sublista $A[1:i-1]$ está ordenada e esses elementos são menores dos da lista $A[i:n]$

Inicialization: Na iteração 1, quando $i = 1$, a sublista $A[1:0]$ é vazia, logo vale O invariante.

Maintenance: Suponha que na iteração i, a sublista $A[1:i-1]$ esteja ordenada. Na iteração $i+1$, realiza-se o inner loop (2-6). Nessa iteração, pelo provado, sabemos que ao fim dela $A[i+1]$ é o menor elemento da sublista $A[i+1:n]$. Como $A[1:i-1]$ está ordenada e $A[i-1] \leq A[i]$, tem-se que $A[1:i+1]$ está ordenada e os elementos continuam menores do que da sublista $A[i+2:n]$

Termination: No inicío da iteração $i = n$, tem-se que a lista $A[1:n-1]$ está ordenada e esses elementos são menores do que $A[n]$. Logo $A[1:n] = A$ está ordenada.

D **(0.5pt)** What is the worst-case running time of BUBBLESORT? How does it compare to the running time of insertion sort?

Resposta: No pior caso para o Bubble Sort, dada a iteração i, a sublista $A[i:n]$ nunca está ordenada. Nesse caso, há $n-1$ iterações (Outer Loop) e para cada iteração i, há $n-i$ iterações, logo o número de operações é $n-1+n-2+...+1 = \frac{n(n-1)}{2}$. A completicade, portanto, é $\Theta(n^2)$, pois encontram-se $c1$ e $c2$, tal que $c1n^2 \leq \frac{n^2-n}{2} \leq c^2n^2$.

Como observado anteriormente, a complexidade do Insertion Sort também é $\Theta(n^2)$ para o worst case, logo eles possuem running time parecidos. Porém, o best-case para o Insertion Sort tem complexidade $\Theta(n)$, enquanto para o Bubble Sort, essa complexidade ainda é $\Theta(n^2)$

3 Growth of Functions

A For each of the following pairs of functions, either $f(n)$ is in $O(g(n))$, $f(n)$ is in $\Omega(g(n))$, or $f(n) = \Theta(g(n))$. Determine which relationship is correct and briefly explain why.

- **(0.5pt)** $f(n) = \log n^2$; $g(n) = \log n + 5$

  Resposta: $f(n) = \Theta(g(n))$. Devo comparar $f(n)$ com $cg(n)$, para alguma contante c. $f(n) = \log n^2 = 2 \cdot \log n$. Ao tomar $c_1 = 2$, $f(n) = 2\log n \leq 2\log n + 10 = 2g(n), \forall n \in \mathbb{N}$ e $n_0 = 1$. Por outro lado, tomando $c_2 = \frac{1}{2}$, seja $n_0 = 4$. $n \geq n_0 \implies \log n \geq \frac{5}{3}$, pois a função logaritmo é crescente. Assim $3\log n \geq 5 \implies 4\log n \geq \log n + 5 \implies 2\log n = f(n) \geq \frac{1}{2}\log n + \frac{1}{2}5 = c_2 g(n)$.

Desta forma, $c_1 g(n) \leq f(n) \leq c_2 g(n) \implies f(n) = \Theta(g(n))$.

- **(0.5pt)** $f(n) = \log^2 n$; $g(n) = \log n$

  Resposta: $f(n)$ está em $\Omega(g(n))$. Utilizando o fato de que a função logaritmo é crescente, $\forall c \in \mathbb{R}^+$, $n_0 = \lceil 2^c \rceil, n \geq n_0 \implies \log n \geq c$. Portanto $f(n) = (\log n)(\log n) \geq c \cdot \log n = cg(n), \forall c$.Desta maneira, $f(n)$ está em $\Omega(g(n))$.

- **(0.5pt)** $f(n) = n \log n + n$; $g(n) = \log n$

  Resposta: $f(n)$ está em $\Omega(g(n))$. Neste caso, $\forall c \in \mathbb{R}^+, f(n) = n \log n + n \geq c \log n = cg(n)$, pois $(n - c) \log n + n \geq 0$, $\forall n \in \mathbb{N}, n \geq n_0 = c$. Logo, $f(n)$ está em $\Omega(g(n))$

- **(0.5pt)** $f(n) = 2^n$; $g(n) = 10n^2$

  Resposta: $f(n)$ está em $\Omega(g(n))$. Provemos inicialmente que $n \geq 2 \log n$, $n \geq 4$ por indução. Para o caso base, $4 \geq 2 \log 4 = 4$, verdade. Suponha que $n \geq 2 \log n$. Assim $n + 1 \geq 2 \log n + 1 = \log n^2 + \log 2 = 2 \log \sqrt{2} n$. Como $\sqrt{2} n \geq n + 1$, para $n \geq 4$, tem-se que $n + 1 \geq 2 \log(n + 1)$. Logo está provado pelo Princípio de Indução.

  Agora veja que dado qualquer constante d, basta tomar $n_0 = 4 + \lceil \log d \rceil$, que teremos $n \geq \log d + 2 \log n$. Como a função exponencial é crescente, $2^n \geq 2^{\log d + 2 \log n} = 2^{\log dn^2} = dn^2$. Desta forma, tomando $d = 10c$, $f(n) = 2^n \geq c \cdot 10n^2 = c \cdot g(n)$ e, portanto, $f(n)$ está em $\Omega(g(n))$.

B **(0.5pt)** Prove that $n^3 - 3n^2 - n + 1 = \Theta(n^3)$.

Resposta: $f(n) = n^3 - 3n^2 - n + 1$ e $g(n) = n^3$. Devo mostrar que existem $c_1, c_2 \in \mathbb{R}^+$, tal que existe n_0, onde $n \geq n_0 \implies c_1 g(n) \leq f(n) \leq c_2 g(n)$. Note que $c_2 = 1$ satifaz a segunda desigualde, pois $n^3 - 3n^2 - n + 1 \leq n^3 \Leftrightarrow 1 \leq n(1 + 3n), \forall n > 0$. Para a outra desigualdade, $n^3 - 3n^2 - n + 1 \geq c_1 n^3 \implies (1 - c_1)n^3 - 3n^2 - n + 1 \geq 0$. Tome $c_1 < 1$. Nesse caso,$(1 - c_1) - \frac{3}{n} - \frac{1}{n^2} + \frac{1}{n^3} \geq 0 \Leftrightarrow 1 - c_1 \geq \frac{3}{n} + \frac{1}{n^2} - \frac{1}{n^3}$, que é válido para algum n_0. Para completar o conceito, veja que $lim_{n \to \infty}(1 - c_1) - \frac{3}{n} - \frac{1}{n^2} + \frac{1}{n^3} = 1 - c1 > 0$, que implica que $\exists n_0 \in \mathbb{N}$, tal que $(1 - c_1) - \frac{3}{n} - \frac{1}{n^2} + \frac{1}{n^3} > 0$. Desta forma, $f(n) = \Theta(g(n))$.

C **(0.5pt)** Prove that $n^2 = O(2^n)$.

Resposta: Nesse caso, basta provar que $n^2 \leq c \cdot 2^n$, para algum c. Tomando $c = 1$ e $n_0 = 5$, vou provar por indução que $n \geq 5 \implies 2^n \geq n^2$. O caso base é verdade, pois $32 \geq 25$. Suponha $2^n \geq n^2$, hipótese de indução. Assim, $2^{n+1} \geq 2 \cdot n^2 = n^2 + n^2 \geq n^2 + 2n + 1 = (n + 1)^2$. Note que $n^2 \geq 2n + 1$, pois $n(n - 2) \geq 1$, para valores de n maiores do que 3, em particular $n \geq 5$. Assim, pelo Princípio da Indução, esta provado. Seja $d = \frac{1}{c}$. Portanto, pelo que foi visto no quarto item de A, $dn^2 \leq 2^n, \forall d \in \mathbb{R}^+, n \geq n_0$, para algum n_0. Como $\nexists c \in \mathbb{R}^+; n^2 \geq c \cdot 2^n$, conclui-se que $n^2 = O(2^n)$.