Fundação Getulio Vargas
Estruturas de Dados e Algoritmos
Prof. Jorge Poco
Aluno: Davi Sales Barreira

Homework 1
Due: July 21, 2019

Homework 1

20/07/2019

1 Induction [3pts]

Answers should be written in this document.

1. Prove by Induction that: $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \qquad \forall n \geq 0$
2. Prove by Induction that: $\forall n \geq 7$ it is true $3^n < n!$
3. Prove by Induction that $\forall n \geq 0$

$$\left\lceil \frac{n}{2} \right\rceil = \begin{cases} \frac{n}{2} & \text{si } n \text{ es par} \\ \frac{n+1}{2} & \text{si } n \text{ es impar} \end{cases}$$

4. Prove by induction that a number is divisible by 3 if and only if the sum of its digits is divisible by 3.
5. Prove that any integer greater than 59 can be formed using only 7 and 11 cent coins.
6. Prove by induction that $F_{n+k} = F_k F_{n+1} + F_{k-1} F_n$
7. Prove by induction in n that $\sum_{m=0}^{n} \binom{n}{m} = 2^n$
8. Prove by induction that a graph with n vertices can have at most $\frac{n(n-1)}{2}$ edges.
9. Prove by induction that a complete binary tree[1] with n levels has $2^n - 1$ vertices.
10. A polygon is convex if each pair of points in the polygon can be joined by a straight line that does not leave the polygon. Prove by induction in $n > 3$ that the sum of the angles of a polygon of n vertices is $180(n-2)$.

2 Correctness of bubblesort [2pts]

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

[1] http://web.cecs.pdx.edu/ sheard/course/Cs163/Doc/FullvsComplete.html

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 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?

The next two parts will prove inequality (1).

B 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.

C 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.

D What is the worst-case running time of BUBBLESORT? How does it compare to the running time of insertion sort?

3 Growth of Functions [2pts]

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.

- $f(n) = \log n^2$; $g(n) = \log n + 5$
- $f(n) = \log^2 n$; $g(n) = \log n$
- $f(n) = n \log n + n$; $g(n) = \log n$
- $f(n) = 2^n$; $g(n) = 10n^2$

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

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

4 Insertion Sort - Mergesort - Quicksort [3pts]

Implement the insertion sort, merge sort and quicksort using the template `test.py` (use Python 3.X). Create a `test.cpp` file and write the equivalent code from `test.py` in C++, ie., the functions: main, `insertion_sort`, `merge_sort`, `quicksort` and `is_sorted`. For the random number generations you can use the `rand` function from `cstdlib`[2]. Your code should print the tuple (number of objects, time insertion_sort, time merge_sort, time quicksort)
You must submit both `test.py` and `test.cpp`. Graphs and descriptions must be included in this document.

[2] `http://www.cplusplus.com/reference/cstdlib/rand/`

4.1 Random Order

1. Create 10 sets of numbers in random order. The sets must have {10k, 20k, 30k, ..., 100k} numbers.
2. Sort these numbers using the 3 algorithms and calculate the time each algorithm takes for each set of numbers.
3. Generate a plot (using excel or another tool) showing a *linechart*, where the x-axis is the "number of elements", and the y-axis is the time that the algorithms took in C++ and Python. This plot must have 6 lines of different colors with a legend.
4. Write a small paragraph (3 to 4 lines) describing the results.

4.2 Ascending Order

Do the same experiment when the numbers are ordered in ascending order.

4.3 Descending Order

Do the same experiment when the numbers are ordered in descending order.

Respostas

Questão 1

Questão 2

Questão 3

Questão 4

Os gráficos abaixo comparam os resultados obtidos para os diferentes algoritmos de sorting implementados tanto em C++ como em Python. Como esperado, o código de C++ performa melhor que em Python, com a diferença maior sendo no algoritmo de *insert sort*.
Além disso, observamos que o *quicksort* apresentou o melhor desempenho considerando tanto o vetor com valores aleatórios como o com valores descendentes. No vetor já ordenado (*ascending*), o tempo dos três algoritmos é muito baixo, como era de se esperar, porém, nesse caso o *insert sort* foi quem apresentou o melhor resultado.

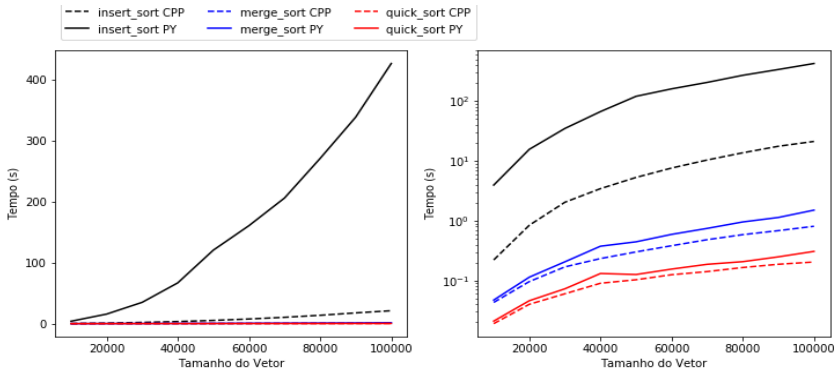

Figure 1: Comparação de performance dos algoritmos de *sorting* para vetores aleatórios

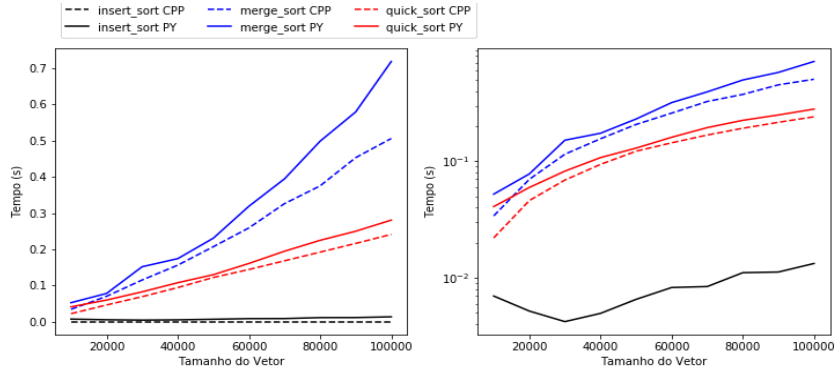

Figure 2: Comparação de performance dos algoritmos de *sorting* para vetores crescentes

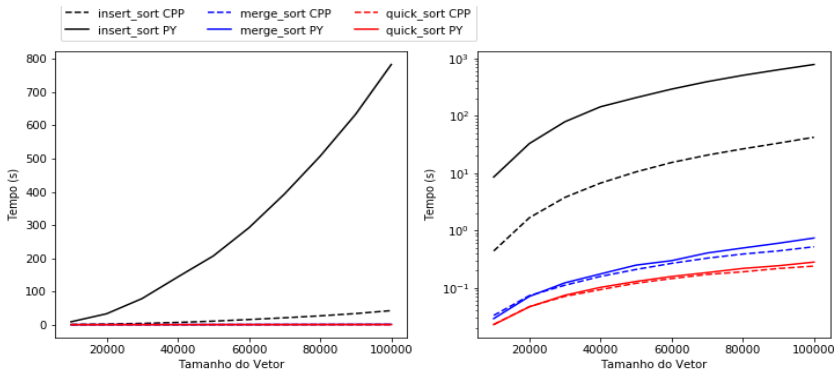

Figure 3: Comparação de performance dos algoritmos de *sorting* para vetores decrescentes