Algorithm and Data Structures HW 1

Mario Alberto Hernandez Salamanca

February 15, 2019

1 Asymptotic Analysis

HW

a) $f(n) = 3n$ $\quad g(n) = n^3$

Prove that $3n \in O(n^3)$

Let $h(n) = (3n)/n^3$

$$\lim_{n\to\infty} \frac{3n}{n^3} = \lim_{n\to\infty} \frac{3}{n^2} = 0$$

$h(n)$ approaches 0 as n aproaches infinity.
also $h(n)$ is continuous on the interval $n > 1$.
So $h(n)$ is bounded on the interval $n > 1$
so there exist a constant c such that $3n \leq cn^3$ for all $n > 1$ (since $h(n) < c$)
So $3n \in O(n^3)$

Prove that $3n \in \Omega(n^3)$

let $h(n) = n^3/3n$

$$\lim_{n\to\infty} \frac{n^3}{3n} = \infty$$

$h(n)$ approaches ∞ as n aproches ∞
so $h(n)$ is continuous on the intervan $n > 0$
so $h(n)$ is bounded in ∞
so theres no a constant c that satisfied $3n \geq cn^3$ for all $n > 0$ (since $h(n) > c$)
So $3n \notin \Omega(n^3)$

Since $3n \in O(n^3)$ and $3n \notin \Omega(n^3)$
$3n \notin \Theta(n^3)$

Prove that $3n \in o(n^3)$

$$\lim_{n\to\infty} \frac{3n}{n^3} = \frac{3}{n^2} = 0$$

so $3n \in o(n^3)$

Prove that $3n \in \omega(n^3)$

$$\lim_{n\to\infty} \frac{3n}{3n^3} = \frac{3}{n^2} = 0$$, it has to be ∞

so $3n \notin \omega(n^3)$

Prove that $n^3 \in O(3n)$
lets $h(n) = n^3/3n$
$$\lim_{n\to\infty} h(n) = \lim_{n\to\infty} \frac{n^3}{3n} = \lim_{n\to\infty} \frac{n^2}{3} = \infty$$
that means that there no c such that $n^3 < c\,3n$

Prove that $n^3 \in \Omega(3n)$
in the prove of $3n \in O(n^3)$ we prove that
there a constant c that $3n < cn^3$ so if c is positive.
$\frac{1}{c} 3n < n^3$ and $\frac{1}{c} = c_2$ that is positive so there a constant
c_2 $n^3 > c_2 3n$ proving that.

$n^3 \in \Omega(3n)$

with the following limit we can prove $n^3 \in o(3n) \wedge n^3 \in \omega(3n)$

$$\lim_{n\to\infty} \frac{n^3}{3n} = \infty$$

this lead to $n^3 \in \omega(3n)$ but $n^3 \notin o(3n)$

b) $F(n) = 7n^{0.7} + 2n^{0.2} + 13\log n$. $g(n) = \sqrt{n}$

Lets $h(n) = \frac{f(n)}{g(n)}$ and $I(n) = \frac{g(n)}{f(n)}$

$$\lim_{n\to\infty} h(n) = \lim_{n\to\infty} \frac{7n^{0.7} + 2n^{0.2} + 13\log n}{\sqrt{n}}$$

$$= \lim_{n\to\infty} 7n^{0.7-0.5} + 2n^{0.2-0.5} + 13\log n \cdot n^{-0.5}$$

$$= \lim_{n\to\infty} 7n^{0.2} + 2n^{-0.3} + \frac{13\log n}{n^{0.5}}$$

$$= \lim_{n\to\infty} 7n^{0.2} + \frac{2}{n^{0.3}} + \frac{13\log n}{n^{0.5}}$$

$7n^{0.2}$ growth to infinitive.

$\frac{2}{n^{0.3}}$ goes to 0

$\frac{13\log n}{n^{0.5}}$, we know that $n^{0.5}$ growth faster than $13\log n$ so it goes to 0

$$\lim_{n\to\infty} h(n) = \infty$$

$$\lim_{n\to\infty} I(n) = \frac{\sqrt{n}}{7n^{0.7} + 2n^{0.2} + 13\log n} = 0$$

in this case we know that $F(n)$ is growing faster than $\sqrt{n}$ because $g(n)$ is 0.5 order and $f(n)$ is 0.7 order so it growth faster. So it goes to 0

From thoes analysis we can know that.
$h(n)$
gives us that. $f(n) \in \omega(g(n))$
$f(n) \in \Omega(g(n))$
$f(n) \notin o(g(n))$
$f(n) \notin O(g(n))$
because $f(n) \notin O(g(n)) \Rightarrow f(n) \notin \Theta(g(n))$

From $I(n)$

gives us that $g(n) \in O(f(n))$
$g(n) \in o(f(n))$
$g(n) \notin \omega(f(n))$
$g(n) \notin \Omega(f(n))$
because $g(n) \notin \Omega(f(n)) \Rightarrow g(n) \notin \Theta(f(n))$

c) $f(n) = \frac{n^2}{\log n}$ $g(n) = n \log n$

lets $h(n) = \frac{f(n)}{g(n)}$ and $I(n) = \frac{g(n)}{f(n)}$

$$\lim_{n\to\infty} h(n) = \frac{\frac{n^2}{\log n}}{n \log n} = \frac{n^{\cancel{2}}}{\cancel{n}\log^2 n} = \frac{n}{\log^2 n} = \infty$$

$\log^2 n$ growith slower than n therefore it goes to ~~positive~~

$$\lim_{n\to\infty} I(n) = \frac{n \log n}{\frac{n^2}{\log n}} = \frac{\log^2 n}{n} = 0$$

from $h(n)$ and $I(n)$ we conclude

$f(n) \in \omega\, g(n)$
$f(n) \in \Omega\, g(n)$
$f(n) \notin o\, g(n)$
$f(n) \notin O\, g(n)$
$g(n) \in O\, f(n)$
$g(n) \in o\, f(n)$
$g(n) \notin \omega\, f(n)$
$g(n) \notin \Omega\, f(n)$

because $f(n) \notin (g(n)) \Rightarrow f(n) \notin \Theta(g(n))$

because $g(n) \notin (g(n)) \Rightarrow g(n) \notin \Theta(f(n))$

d) $f(n) = (\log(3n))^3 \cdot g(n) = 9 \cdot \log n$

lets $h(n) = \frac{f(n)}{g(n)}$ and $I(n) = \frac{g(n)}{f(n)}$

$$\lim_{n \to \infty} h(n) = \frac{\log^3(3n)}{9 \log n} = \frac{1}{9} \lim_{n \to \infty} \frac{\log^3(3n)}{\log(n)} = \frac{1}{9} \lim_{n \to \infty} 3\log^2(3n)$$

$$\frac{1}{9} \infty = \infty$$

$$\lim_{n \to \infty} I(n) = \frac{9 \log n}{\log^3(3n)} = 9 \lim_{n \to \infty} \frac{\log n}{\log^3(3n)} = 9 \lim_{n \to \infty} \frac{1}{3\log^2(3n)} =$$

$$9 \cdot 0 = 0$$

From $h(n)$ and $I(n)$ we can say the following.

$f(n) \notin O(g(n))$
$f(n) \notin o(g(n))$
$f(n) \in \Omega(g(n))$
$f(n) \in \omega(g(n))$
$g(n) \in O(f(n))$
$g(n) \in o(f(n))$
$g(n) \notin \Omega(f(n))$
$g(n) \notin \omega(f(n))$

because $f(n) \notin O(g(n)) \Rightarrow f(n) \notin \Theta(g(n))$

because $g(n) \notin \Omega(f(n)) \Rightarrow g(n) \notin \Theta(f(n))$

2 Selection Sort

2.1 Implement Selection Sort

```
void selection_sort(int Arr[], int n){
          //define variables
          int i , j , min_indx;
          i=0;
          int temp;
          //begining of the algorithm
          for(i; i < n - 1 ; i++){
                    // I assume that the smaller value is i
                    min_indx = i;
                    for(j = i + 1 ; j < n ; j++){
                              if(Arr[j] < Arr[min_indx]){
                                        min_indx = j;
                              }
                    }
                    temp=Arr[i];
                    Arr[i]=Arr[min_indx];
                    Arr[min_indx]=temp;
          }
          return;
}
```

2.2 Show that Selection Sort is correct

The selection sort is correct because of the following loop:

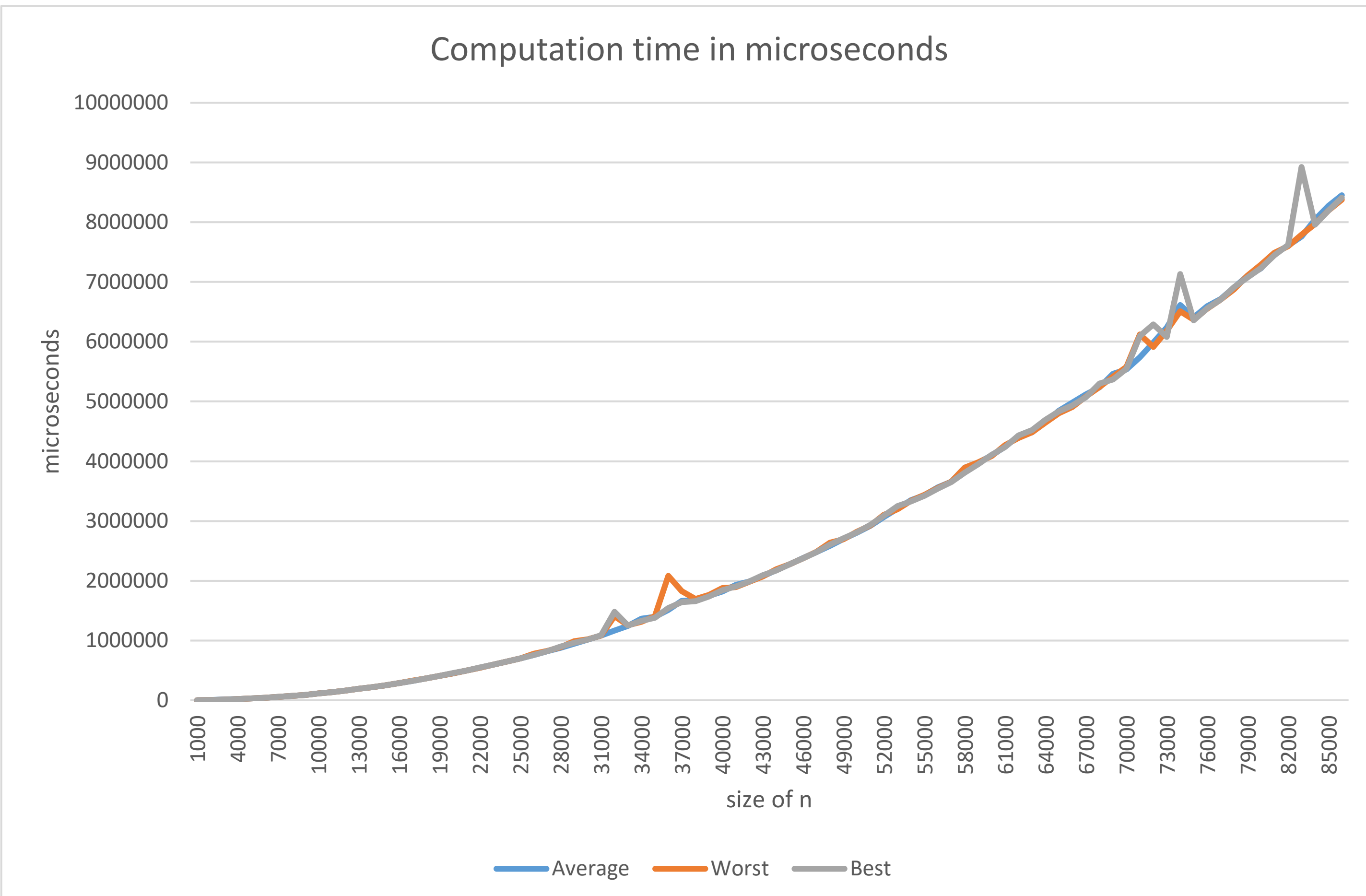
```
for(i; i < n - 1 ; i++){
                    min_indx = i;
                    for(j = i + 1 ; j < n ; j++){
                              if(Arr[j] < Arr[min_indx]){
                                        min_indx = j;
                              }
                    }
                    temp=Arr[i];
                    Arr[i]=Arr[min_indx];
                    Arr[min_indx]=temp;
          }
```

We see that the loop will check all the elements and will look for the smaller element and will place it in the last position of the already sorted part of the array , for example if is the first iteration will place the smaller one at the beginning and the second smaller will be place it in the next position making the loop invariant.

2.3 Random number generator

```
//     Random array generator
//     being length the size of the array
for(k;k < length ;k++){
                array[k]= rand()%100;
        }
//Generate the worst case array inverting the position
//of  the array already sorted
void worst_case(int array[], int size, int worst[] ){
    int i , j;
    j=0;
    for (i=size-1; i >= 0; i--) {
        worst[i] = arr[j];
        j++;
    }
    return;
}
//Generate the best case array coping the position
// of  the array already sorted
void best_case(int arr[], int size, int best[]){
         int i;
         for (i=0; i < size; i++) {
                best[i]=arr[i];
         }
         return;
}
```

2.4 computation time


Computation time in microseconds
microseconds
10000000
9000000
8000000
7000000
6000000
5000000
4000000
3000000
2000000
1000000
0
1000
4000
7000
10000
13000
16000
19000
22000
25000
28000
31000
34000
37000
40000
43000
46000
49000
52000
55000
58000
61000
64000
67000
70000
73000
76000
79000
82000
85000
size of n
Average
Worst
Best

2.5 analysis

From the graph we can conclude that the running time of the algorithm is the almost the same in the best, average and worst case doing the mathematical analysis of the algorithm indeed the running time is the same in the 3 cases being

$$(n(n+1))/2$$