Lab 13 - Time Complexity

hpotosij@unal.edu.co

Cormen, Leiserson, Rivest and Stein

Exercise 2.1-2

Rewrite the INSERTION-SORT procedure to sort into nonincreasing instead of nondecreasing order.

```
For j = 2 to  A.length
        Key = A[j]
        i = j – 1
        while i > 0 and  A[i] < key
                A[i + 1] = A[i]
                I = i – 1
        A[i + 1] = key
```

Exercise 2.1-3

Consider the searching problem:

Input: A sequence of n numbers A [a_1, a_2, … a_n] and a value *v*.

Output: An index i such that *v* = [i] or the special value NIL if v does not appear in A.

Write pseudocode for linear search, which scans through the sequence, looking for. Using a loop invariant, prove that your algorithm is correct. Make sure that your loop invariant fulfills the three necessary properties.

Pseudocode:

```
Linear_search(A, v)
        for i = 1 to  A.length
                if A[i] = = v
                        return i
        return NIL
```

Propiedades:

Initialization. Initially, the items A[1..i – 1] does not match the value of v. this is trivially because the set of items, in this case, is empty-there is no item at index 0; therefore, all the preceding items does not match the invariant value initially

Maintenance. On each step, we know that A does not contain ν. We compare it with A[i]. If they are the same, we return i, which is a correct result. Otherwise, we continue to the next step. We have already insured that A[A..i−1] does not contain ν and that A[i] is different from ν, so this step preserves the invariant.

Termination. The loop terminates when i>A.length. Since i increases by 1 and i>A.length, we know that all the elements in A have been checked and it has been found that ν is not among them. Thus, we return NIL

Dasgupta, Papadimitriou and Vazirani

Exercise 0.1.

0.1. In each of the following situations, indicate whether $f = O(g)$, or $f = \Omega(g)$, or both (in which case $f = \Theta(g)$).

	$f(n)$	$g(n)$
(a)	$n - 100$	$n - 200$
(b)	$n^{1/2}$	$n^{2/3}$
(c)	$100n + \log n$	$n + (\log n)^2$
(d)	$n \log n$	$10n \log 10n$
(e)	$\log 2n$	$\log 3n$
(f)	$10 \log n$	$\log(n^2)$
(g)	$n^{1.01}$	$n \log^2 n$
(h)	$n^2 / \log n$	$n(\log n)^2$
(i)	$n^{0.1}$	$(\log n)^{10}$
(j)	$(\log n)^{\log n}$	$n / \log n$
(k)	$\sqrt{n}$	$(\log n)^3$
(l)	$n^{1/2}$	$5^{\log_2 n}$
(m)	$n2^n$	3^n
(n)	2^n	2^{n+1}
(o)	$n!$	2^n
(p)	$(\log n)^{\log n}$	$2^{(\log_2 n)^2}$
(q)	$\sum_{i=1}^{n} i^k$	n^{k+1}

a) $f = \Theta(g)$
b) $f = O(g)$
c) $f = \Theta(g)$
d) $f = \Theta(g)$
e) $f = \Theta(g)$
f) $f = \Theta(g)$
g) $f = \Omega(g)$
h) $f = \Omega(g)$
i) $f = \Omega(g)$
j) $f = \Omega(g)$
k) $f = \Omega(g)$
l) $f = O(g)$
m) $f = O(g)$
n) $f = \Theta(g)$
o) $f = \Omega(g)$
p) $f = O(g)$
q) $f = \Theta(g)$

Exercise 0.2 .
Show that, if c is a positive real number, then $g(n) = 1 + c + c^2 + \cdots + c^n$ is:

a) $\Theta(1)$ if $c < 1$.
b) $\Theta(n)$ if $c = 1$
c) $\Theta(c^n)$ if $c > 1$

por series geométricas tenemos

$$g(n) = \frac{c^{n+1} - 1}{c - 1}$$

a) c < 1
aplicando limite a g(n) tenemos

$$\lim_{n\to\infty}\left(\frac{c^{n+1}-1}{c-1}\right) = \frac{1}{c-1}$$

Entonces, $\frac{1}{c-1} > g(n) > 1$
Por lo que $g(n) = \Theta(1)$

b) si c = 1 tenemos

$$g(n) = \sum_{i=0}^{n}(c^i) = n + 1$$

por lo que $g(n) = \Theta(n)$

c) sí c > 1 tenemos

$$\lim_{n\to\infty}\left(\frac{g(n)}{c^n}\right) = \lim_{n\to\infty}\left(\frac{c^{n+1}-1}{c^{n+1}- c^n}\right) = \frac{c}{c-1}$$

entonces $g(n) = O(c^n)$ y $g(n) = \Omega(c^n)$ por lo tanto tenemos
$g(n) = \Theta(c^n)$

Complete the following table

Algorithm	**Worst case time complexity**	**Best case time complexity**	**Average case time complexity**	**Space complexity**
Binary Search	O(Log n)	O(1)	O(Log n)	O(1)
Finding the smallest or largest item in an unsorted array	O(n Log n)	O(n Log n)	O(n Log n)	O(1)
Kadane's algorithm	O(n)	O(1)	O(n)	O(1)
Sieve of Eratosthenes	O(n)	O(1)	O(n log (log n))	O(1)
Merge Sort	O(n Log n)	O(n Log n)	O(n Log n)	O(1)
Heap Sort	O(n Log n)	O(n Log n)	O(n Log n)	O(n)
Quick Sort	$O(n^2)$	O(n Log n)	O(n Log n)	O(n)
Tim Sort	O(n Log n)	O(n)	O(n Log n)	O(n)
Divide and conquer (Convex Hull)	O(n Log n)	O(n Log n)	O(n Log n)	O(n)
Insertion Sort	$O(n^2)$	O(n)	$O(n^2)$	O(1)

Dijkstra's algorithm	O(|E| + |V| Log|V|)	O(|E| Log|V|)	O(|E| + |V| Log|V|)	
Naive Matrix Multiplication	$O(n^3)$	$O(n^3)$	$O(n^3)$	$O(n^3)$
Floyd–Warshall algorithm	$O(n^3)$	$O(n^3)$	$O(n^3)$	$O(n^3)$
Naive Matrix Inversion	$O(n^3)$	$O(n^2 \text{ Log } n)$	$O(n^3)$	$O(n^3)$
Calculate the permutations of n distinct elements without repetitions	O(n! Log(n!)	O(n!)		