Ricardo Pineda
20160164

Laboratorio Alogoritmia 1

Ejercicio 1

Input: Secuencia de n Numeros A = [a1, a2, a3... an]
Output: A[i] o nulo
Asumiendo que el valor que buscamos esta dado por T:

.

Algorithm 1 Busqueda Lineal (Pseudo-Codigo)

```
1: i = 0;
2: for i <= A.length do
3:     if i == T then
4:         Return i;
5:     else
6:         i = i + 1;
7:         return;
8: if i > A.length then
9:     return null;
```

Entonces, el *loop invariant* en este caso es que i recorre la lista entera buscando T y no se detiene hasta encontrarlo o hasta terminar de recorrer la lista sin exito.

Ejercicio 2

Input: matriz A (n x m) y matriz B (m x p)
Output: matriz C (n x p)

For i from 1 to n: —— **n**
For j from 1 to p: —— **n**
Let sum = 0 —— **c**
For k from 1 to m: —— **n**
Set sum ¡- sum + A[i][k] * B[k][j] —— **c**
Set C[i][j] ¡- sum —— **c**
return C —— **c**
Running Time: —— **O**(n^3)

Ejercicio 3

Bubble Sort
El *Worst-Case* de este algoritmo es $\mathbf{O}(n^2)$

Como se comparan *Best-Case* y *Worst-Case* de Bubble Sort e Insertion Sort:

Best-Case

En ambos casos, el Best-Case sucede cuando la lista ya esta ordenada, por lo que ambos algoritmos recorreran la lista entera sin tener que efectuar ningun cambio y por consiguiente, tardaran **O(n)** en hacerlo.

Worst-Case

En ambos algoritmos se tendra el Worst-Case en el momento en el que querramos ordenar una lista que esta ordenada inversamente a como lo necesitamos ordenar.

Por consiguiente, ambos algoritmos tardaran $\mathbf{O}(n^2)$ en ordenarlo correctamente puesto que deben de recorrer toda la lista por su manera de funcionar.

El **Bubble Sort** agarra los primeros dos y los ordena, luego pasa al siguiente elemento y lo compara solo con el elemento previo, y los cambia de posicion si es necesario, luego pasa al siguiente y hace lo mismo hasta eventualmente recorrer la lista entera.
Luego de esto, tiene que volver a hacer el mismo proceso ordenando los elementos de la manera correcta hasta que todos queden ordenados.
Cabe mencionar que Bubble Sort no tiene la capacidad de *saber* cuando la lista este ordenada, por lo que tendra que recorrer la lista entera al menos una vez incluso ya estando ordenada.

El **Insertion Sort** al igual que el Bubble Sort tiene que recorrer toda la lista, solo que en vez de ir por parejas, compara el siguiente elemento con los elementos que ya ha ordenado, poniendolo asi en la posicion correcta. Debe de hacer eso con todos los elementos por lo que recorre la lista entera.

Diferencias:

1. El Insertion Sort es mas rapido al detectar que la lista ya esta ordenada.
2. El Bubble Sort recorre la lista varias veces cuando no esta ordenada, mientras que Insertion Sort solo necesita hacerlo una vez.