Clase 7. Algoritmos de Búsqueda y Ordenamiento

TRABAJADORES INFORMÁTICOS
organizados en AGC

Ministerio de Trabajo, Empleo y Seguridad Social
Argentina

¿Qué era un

Algoritmo?

REPASO

Un **algoritmo** es una secuencia detallada de pasos que nos permite resolver un **problema**

Problema de

Búsqueda

Dada una lista de elementos, se quiere obtener la posición en la que se encuentra un determinado elemento.

Por ejemplo, dada la siguiente lista: **[1, 5, 3, 4, 2, 7, 9]**

Se quiere poder recibir un elemento, por ejemplo el **7**, y buscar en qué posición se encuentra en la lista, en este caso en la **posición 5** (empezando a contar desde 0).

Algoritmo de

Búsqueda Lineal (o secuencial)

1) Comienzo por el elemento en la primera posición de la lista
2) Me fijo si el elemento actual es el que estoy buscando
   a) Si lo es, devuelvo su posición
   b) Si no lo es, avanzo al siguiente elemento de la lista, y vuelvo al paso 2)
3) Si recorrí todos los elementos, y ninguno es el que buscaba entonces no se encuentra en la lista.

Algoritmo de

Búsqueda Lineal (o secuencial)

Ejemplo

n = 7

1	5	3	4	2	7	9

Algoritmo de

Búsqueda Lineal (o secuencial)

Ejemplo

n = 7

i = 0

Algoritmo de

Búsqueda Lineal (o secuencial)

Ejemplo

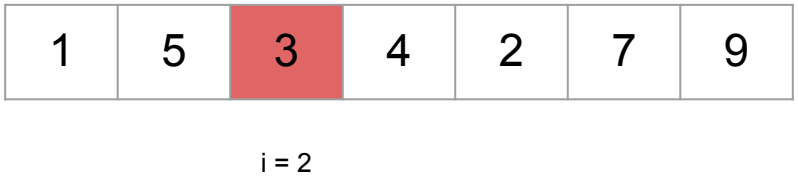
n = 7

Algoritmo de

Búsqueda Lineal (o secuencial)

Ejemplo

n = 7

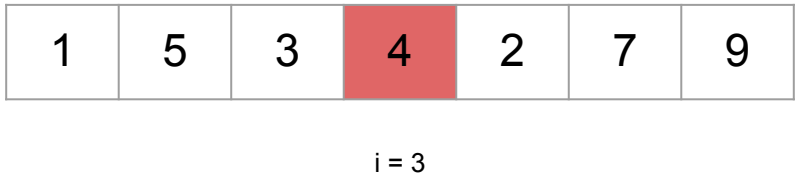


Algoritmo de

Búsqueda Lineal (o secuencial)

Ejemplo

n = 7

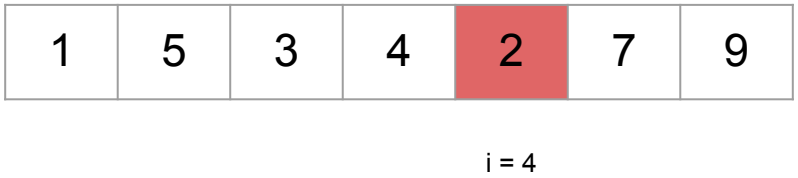


Algoritmo de

Búsqueda Lineal (o secuencial)

Ejemplo

n = 7

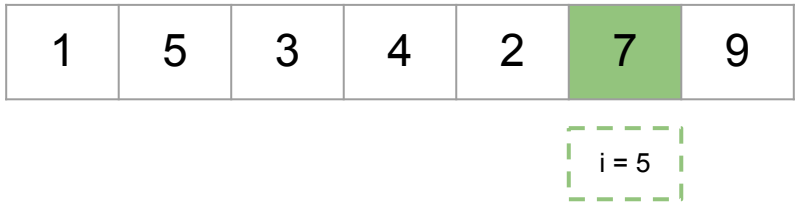


Algoritmo de

Búsqueda Lineal (o secuencial)

Ejemplo

n = 7



¿Qué tan costosa es la

Búsqueda Lineal (o secuencial)?

- En el peor de los casos vamos a tener que recorrer cada uno de los elementos una vez.

- Si duplicamos el tamaño de la lista, en promedio vamos a duplicar la cantidad de comparaciones que hacemos hasta encontrar el elemento. Entonces decimos que el tiempo del algoritmo T(N) es proporcional a N (donde N es el tamaño de la lista).

Ejercicio

10 mins ⏱️

Búsqueda Lineal

Implementar una función **buscar** que reciba una lista de enteros y un entero **n**, y devuelva la posición de n en la lista (o None si no se encuentra).

Ejercicio

Búsqueda Lineal

Python

```
def buscar(lista, n):
    i = 0
    for elemento in lista:
        if (elemento == n):
            return i
        i += 1
     return None
```

Ejercicio

Búsqueda Lineal

Python 

```
def buscar(lista, n):
    for i, elemento in enumerate(lista):
        if (elemento == n):
            return i
     return None
```

La función **enumerate** en Python permite obtener el índice de cada elemento

Problema de

Ordenamiento

Dada una lista de enteros desordenados, se quiere obtener una lista con los mismos enteros pero ordenados.

Por ejemplo, dada la siguiente lista: **[2, 5, 3, 4, 1, 7, 9]**

Se quiere poder obtener la lista ordenada **[1, 2, 3, 4, 5, 7, 9]**

Algoritmo de

Ordenamiento por Selección

1) Comienzo por el elemento en la primera posición de la lista
2) Busco el mínimo elemento entre la posición actual y el final de la lista
3) Intercambio posiciones entre el actual y el elemento encontrado en el paso 2)
4) Avanzo al siguiente elemento de la lista, y vuelvo al paso 2)

Algoritmo de

Ordenamiento por Selección

Ejemplo

2	5	4	3	1	9	7

Algoritmo de

Ordenamiento por Selección

Ejemplo

Algoritmo de

Ordenamiento por Selección

Ejemplo

Algoritmo de

Ordenamiento por Selección

Ejemplo

Algoritmo de

Ordenamiento por Selección

Ejemplo

Algoritmo de

Ordenamiento por Selección

Ejemplo

Algoritmo de

Ordenamiento por Selección

Ejemplo

Algoritmo de

Ordenamiento por Selección

Ejemplo

1	2	3	4	5	7	9

Algoritmo de

Ordenamiento por Selección

Ejemplo

1	2	3	4	5	7	9

¿Qué tan costosa es el

Ordenamiento por Selección?

- En el peor de los casos (siempre el mínimo es el último), vamos a tener que por cada elemento que recorremos, recorrer desde esa posición hasta el final para encontrar el mínimo.
- Es decir, para la primera posición vamos a tener que recorrer N elementos hasta encontrar el mínimo, para la segunda N-1 elementos, para la tercera N-2, …, para la última vamos a tener que recorrer 0 elementos. En total, terminamos haciendo (N + N-1 + … + 1) comparaciones, lo que equivale a (N+1)*N/2. En promedio podemos decir que T(N) va a ser proporcional a N^2 + N, en la práctica decimos que T(N) es proporcional a N^2.

Ejercicio

20 mins ⏱️

Selección

Implementar una función **ordenar** que reciba una lista de enteros y ordene la lista usando el algoritmo de **selección**

Algoritmo de

Ordenamiento por Inserción

1) Comienzo por el elemento en la primera posición de la lista
2) Inserto ordenadamente el elemento actual en la sublista que va desde 0 a la posición actual
3) Avanzo al siguiente elemento de la lista, y vuelvo al paso 2)

Algoritmo de

Ordenamiento por Inserción

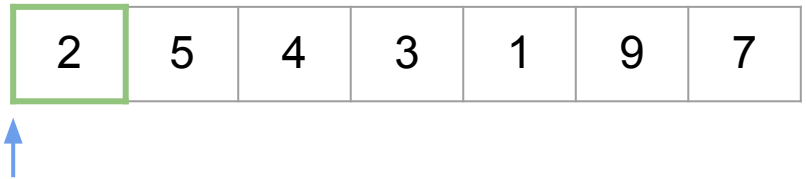
Ejemplo

2	5	4	3	1	9	7

Algoritmo de

Ordenamiento por Inserción

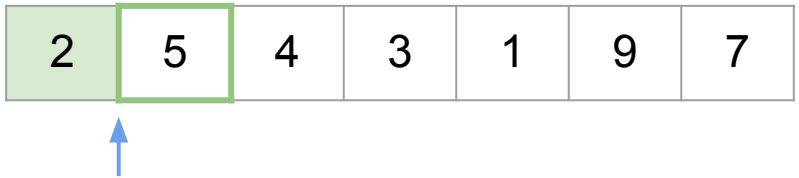
Ejemplo



Algoritmo de

Ordenamiento por Inserción

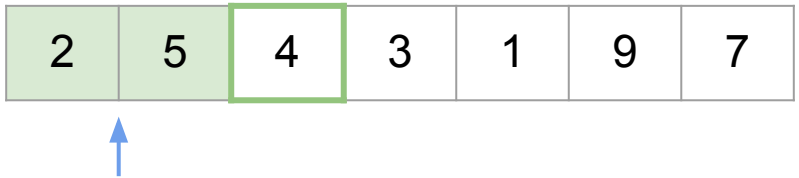
Ejemplo



Algoritmo de

Ordenamiento por Inserción

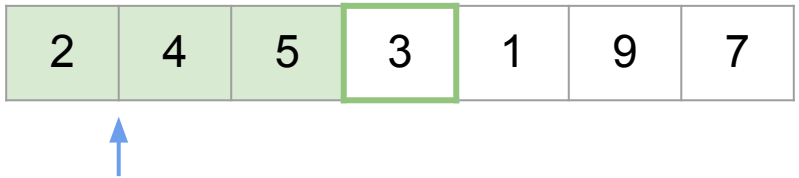
Ejemplo



Algoritmo de

Ordenamiento por Inserción

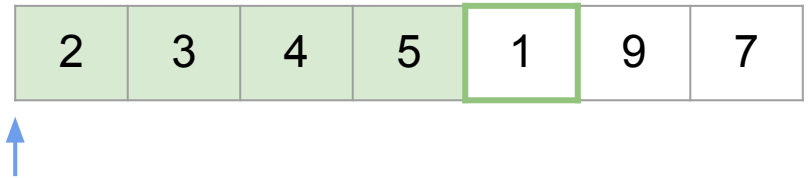
Ejemplo



Algoritmo de

Ordenamiento por Inserción

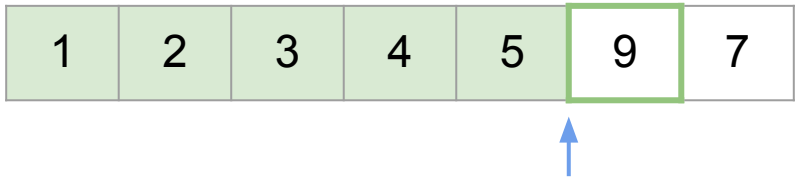
Ejemplo



Algoritmo de

Ordenamiento por Inserción

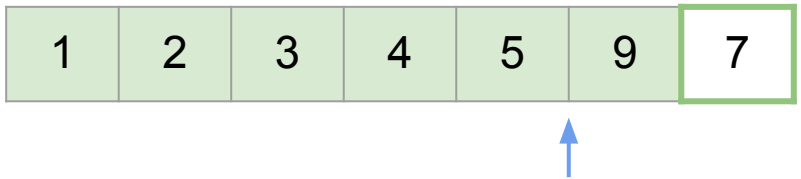
Ejemplo



Algoritmo de

Ordenamiento por Inserción

Ejemplo



Algoritmo de

Ordenamiento por Inserción

Ejemplo

1	2	3	4	5	7	9

¿Qué tan costosa es el

Ordenamiento por Inserción?

- En el peor de los casos (siempre tenemos que insertar al principio), vamos a tener que por cada elemento que recorremos, recorrer desde esa posición hasta el principio para insertarlo.
- Es decir, para la última posición vamos a tener que recorrer N elementos hasta insertar en la primera, para la anteúltima N-1 elementos, para la ante anteúltima N-2, …, para la primera vamos a tener que recorrer 0 elementos. En total, terminamos haciendo (N + N-1 + … + 1) comparaciones, lo que equivale a (N+1)*N/2. En promedio podemos decir que T(N) va a ser proporcional a N^2 + N, en la práctica decimos que T(N) es proporcional a N^2.

Ejercicio

25 mins ⏱️

Inserción

Implementar una función **ordenar** que reciba una lista de enteros y ordene la lista usando el algoritmo de **inserción**