Análisis de métodos de buscar

Diego Quirós Artiñano

Universidad Nacional de Costa Rica

EIF-203: Estructuras Discretas (10 A.M.)

Carlos Loria-Saenz

24 de marzo, 2022

Índice

Índice de fíguras

Índice de tablas

Análisis de buscar_while

0. El algoritmo original.

```
def buscar_while(x:int, a:list[int])-> int:
    p:int = 0
    while p < len(a):
        if x == a[p]:
            return p
        else:
            p += 1
    return -1
```

1. Tamaño de los datos

   El tamaño esta dado por la cantidad de elementos en a, n = len(a)

2. Operaciones de Interés

Notación [en función de tiempo (T)]	Operación	Tipo de Operación
$T_{=}$	p:int = 0	constante
$T_{<}$	p <len(a)	constante
$T_{==}$	x == a[p]	constante
$T_{[]}$	a[p]	constante
$T_{+=}$	p += 1	constante
$T_{len(a)}$	len(a)	constante

Cuadro 1

Operaciones buscar_while()

Peor caso: toas son iguales a la mas grande de todas, como supuesto

Asumimos que vale 1 (unidad de tiempo), según el más alto

3. Ecuación

```
# Separando por partes
# Parte 1
p:int = 0
# Parte 2
while p < len(a):
    if x == a[p]:
        return p
    else:
        p += 1
# Parte 3
return -1
```

$$T_{buscar\_while()}(n) = T_{parte1} + T_{parte2} + T_{parte3}$$

$$T_{buscar\_while()}(n) = 1 + T_{while()}(n) + 0$$

$$T_{buscar\_while()}(n) = T_{while()}(n) + 1$$

Evaluando $T_{while()}(n)$:

2 operaciones: <y len(a)

2 operaciones: == y []

1 operación: +=

Entonces:

$$T_{while()}(n) = 2 + 2 + 1 + T_{while()}(n-1)$$

$$T_{while()}(n) = 5 + 5 + 5 + ...T_{while()}(0)$$

Como el while va a verificar si sigue siendo parte del while entonces se cuentan las primeras 2 operaciones otra vez:

$$T_{while()}(N) = 5n + 2$$

4. Volviendo a meter en la ecuación original:

$$T_{buscar\_while()}(n) = 5n + 2 + 1$$

$$T_{buscar\_while()}(n) = 5n + 3$$

5. Orden de crecimiento

   Como la ecuación es una función lineal entonces el orden de crecimiento también lo cual significa que: $O(n) \sim T_{buscar_while()}(n)$

6. Código

```
def buscar_while_instrumentado(x:int, a:list[int])-> int:
    contador_operaciones = 0
    p:int = 0
    contador_operaciones += 1
    no_salir = p < len(a)
    contador_operaciones += 2
    while no_salir:
        if x == a[p]:
            contador_operaciones += 2
            return contador_operaciones
        else:
            contador_operaciones += 2
            p += 1
```

```
            contador_operaciones += 1
        no_salir = p < len(a)
        contador_operaciones += 2
    return contador_operaciones
```

Análisis de buscar_for

0. El algoritmo original.

```
def buscar_for(x:int, a:list[int]) -> int:
    for p in range(len(a)):
        if x == a[p]:
            return p
    return -1
```

1. Tamaño de los datos

   El tamaño esta dado por la cantidad de elementos en a, n = len(a)

2. Operaciones de Interés

Notación [en función de tiempo (T)]	Operación	Tipo de Operación
$T_{range()}$	range(len(a))	constante
$T_{==}$	x == a[p]	constante
$T_{[]}$	a[p]	constante

Cuadro 2

Operaciones buscar_for()

   Peor caso: toas son iguales a la mas grande de todas, como supuesto

   Asumimos que vale 1 (unidad de tiempo), según el más alto

3. Ecuación

```
# Separando por partes
# Parte 1 (range) y Parte 2 (for)
for p in range(len(a)):
    if x == a[p]:
        return p
# Parte 3
return -1
```

$$T_{buscar\_for()}(n) = T_{parte1} + T_{parte2} + T_{parte3}$$

$$T_{buscar\_for()}(n) = 1 + T_{for()}(n) + 0$$

$$T_{buscar\_for()}(n) = T_{for()}(n) + 1$$

Evaluando $T_{for()}(n)$:

2 operaciones: == y []

Entonces:

$$T_{for()}(n) = 2 + T_{for()}(n-1)$$

$$T_{for()}(n) = 2 + 2 + 2 + ... + T_{for()}(0)$$

$$T_{for()}(n) = 2n + 0$$

4. Volviendo a meter en la ecuación original:

$$T_{buscar_{for()}}(n) = 2n + 1$$

5. Orden de crecimiento

   Como la ecuación es una función lineal entonces el orden de crecimiento también lo cual significa que: $O(n) \sim T_{buscar_{for()}}(n)$

6. Código

```
def buscar_for_instrumentado(x:int, a:list[int]) -> int:
    contador_operaciones = 0
    contador_operaciones += 1 #crear range y
    # todas la operaciones dentro de este
    for p in range(len(a)):
        contador_operaciones += 2 # == y []
        if x == a[p]:
            return contador_operaciones
    return contador_operaciones
```

Excel

Para importar a Excel para hacer la gráfica, se hizo este código:

```
def test_buscar_instrumentado(filename, init, maxi, inc):
    file = open(filename, 'w')
    file.write('n;time_while;time_for\n')
    for n in range(init, maxi, inc):
        a = list(range(n))
        x = n
        file.write(f'{n};{buscar_while_instrumentado(x,a)};{buscar_for_instrumentado(x,a)}\n')
    file.close()
test_buscar_instrumentado('buscar_instrumentado.csv', 10, 200, 10)
```

n	time_while	time_for
10	53	21
20	103	41
30	153	61
40	203	81
50	253	101
60	303	121
70	353	141
80	403	161
90	453	181
100	503	201
110	553	221
120	603	241
130	653	261
140	703	281
150	753	301
160	803	321
170	853	341
180	903	361
190	953	381

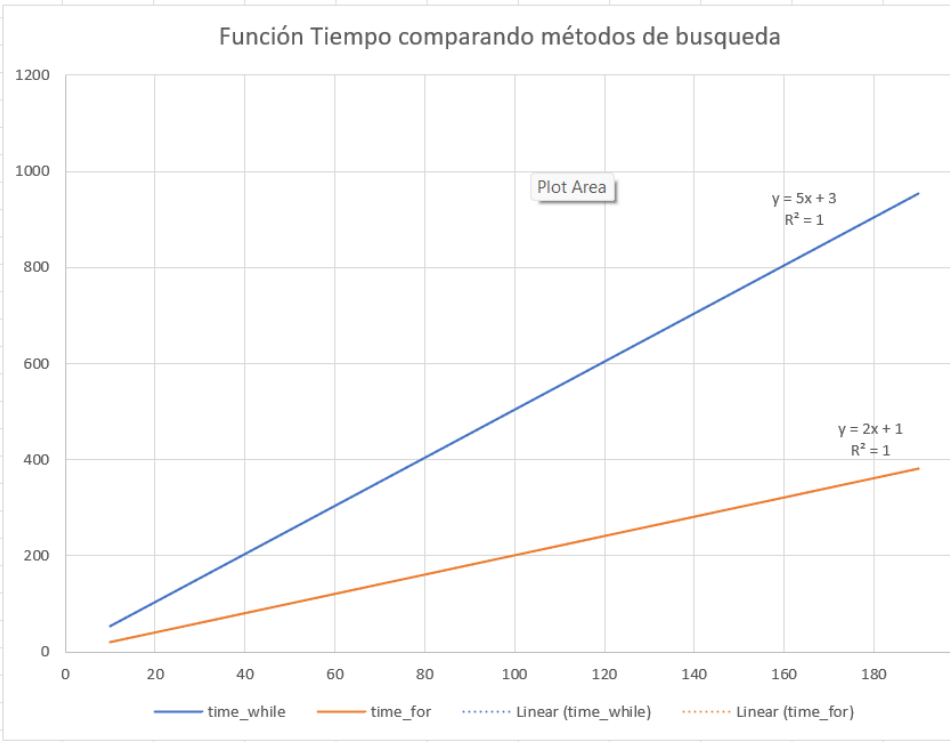

Figura 1

Gráfica de los métodos en Excel

Referencias

Loria-Saenz, C. (2022). *Clase 21 de marzo, 2022. estructuras discretas.* UNA. Descargado de https://sites.google.com/site/unaloriacarlos