| | |
|---|---|
| | **Carátula para entrega de prácticas** |
| Facultad de Ingeniería | Laboratorio de docencia |

# Laboratorios de computación
# salas A y B

*Profesor:* Ing. Jonathan Roberto Torres Castillo

*Asignatura:* Estructura De Datos Y Algoritmos II

*Grupo:* 5

*No de Práctica(s):* 4

*Integrante(s):* Rosas Martínez Jesús Adrián

*Semestre:* 2018-II

*Fecha de entrega:* 15 de marzo de 2018

*Observaciones:*

CALIFICACIÓN: _____

**PRACTICA 4**
**ALGORITMOS DE BUSQUEDA PARTE I**
**ESTRUCTURA DE DATOS Y ALGORITMOS II**
**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**

**Profesor: Ing. Jonathan Roberto Torres Castillo**

**Alumno: Rosas Martínez Jesús Adrián**
**Correo: jesus_olimpo@outlook.com**

## 1. Reglas de las prácticas de laboratorio

- Deberá respetar la estructura general del documento proporcionado para la respectiva práctica y llenar cada una de las secciones que aparecen en él.
- El desarrollo de la práctica deberá ser autentico. Aquellos alumnos que presenten los mismos cálculos, código fuente, etcétera, se les será anulada inapelablemente la práctica correspondiente con una calificación de 0.
- Deberá subir el informe y los scripts al link de DropBox correspondiente a la práctica que se encuentra en la carpeta del curso. Los archivos deberán tener las características descritas en el archivo "Reglas_Laboratorio.pdf" o se anularan son calificación de 0.
- El día de entrega establecido deberá ser respetado por todos los alumnos, la práctica debe ser terminada parcialmente al finalizar la clase, completada máximo a los 7 días siguientes a la sesión de laboratorio tiempo en el cual debe subir la practica al link correspondiente y sustentada en la primera hora de clase de la siguiente sesión de laboratorio.
- No se recibirán informes ni actividades si no asiste a la sesión de laboratorio y no presenta justificación médica, por lo cual no tendrá derecho a la calificación de dicha práctica y la nota será de 0.
- La inasistencia a 3 sesiones de laboratorio en el semestre será causal de reprobación de las prácticas y del curso.

## 2. Objetivos

- El estudiante identificará el comportamiento y características de algunos algoritmos de búsqueda por comparación de llaves.
- El estudiante implementará algoritmos de búsqueda por comparación de llaves.

## 3. Introducción

Recuperar información de una computadora es una de las actividades más útiles e importantes; al proceso de encontrar un dato específico llamado llave en alguna estructura de datos tales como una lista o arreglo, se denomina búsqueda. Este proceso termina

exitosamente cuando se localiza el elemento que contienen la llave buscada, o termina sin éxito, cuando no aparece ningún elemento con esa llave.

Los algoritmos de búsqueda se pueden clasificar en:

- Búsqueda por comparación (búsqueda lineal o secuencial y búsqueda binaria)
- Búsqueda por transformación de llaves

**Búsqueda Lineal**

En este algoritmo se empieza a buscar desde la primera posición de la lista, comparando el elemento con la llave y si no se encuentra continúa verificando las siguientes posiciones hasta encontrarlo, entonces el algoritmo termina. Así el elemento a buscar puede estar en el primer elemento, el último elemento o entre ellos.

El número de iteraciones siempre es igual al tamaño del arreglo, independientemente de donde se encuentre el elemento a buscar. Es decir, considerando la cantidad de comparaciones:

- Mejor caso: O(1) El elemento buscado está en la primera posición. Es decir, se hace una sola comparación.
- Peor caso: O(n): El elemento buscado está en la última posición. Necesitando igual cantidad de comparaciones que de elementos del arreglo.

Por lo tanto, la velocidad de ejecución depende linealmente del tamaño del arreglo.

**Búsqueda Binaria**

El algoritmo de búsqueda binaria o también llamado dicotómica es eficiente para encontrar un elemento en una lista ya ordenada y es un ejemplo de la técnica divide y conquista.

En el algoritmo se divide repetidamente a la mitad la porción de la lista que podría contener al elemento, hasta reducir las ubicaciones posibles a solo una. La estrategia consiste en comparar una llave con el elemento de en medio de la lista, si es igual entonces se encontró el elemento, sino cuando x es menor que el elemento del medio se aplica la misma estrategia a la lista de la izquierda y si x es mayor, a la lista de la derecha. Si en algún momento la sub-lista de búsqueda tiene longitud 0 o negativa significa que el valor buscado no se encuentra en la lista.

Su tiempo de ejecución es O(log n).

## 4. Planteamiento del Problema y Desarrollo

### Actividad 1

A continuación, se proporcionan los pseudocódigos de las funciones para búsqueda lineal explicados en clase.

| | | |
|---|---|---|
| BúsquedaLineal (A,x )<br>Inicio<br>n=longitud de A<br>encontrado=-1<br>Para k=0 hasta n-1<br>Si x==A[k]<br>encontrado = k<br>Fin Si<br>Fin Para<br>Retorna encontrado<br>Fin | BúsquedaLinealMejorada(A,x)<br>Inicio<br>n=longitud de A<br>encontrado=-1<br>Para k=0 hasta n-1<br>Si A[k]==x<br>encontrado= k<br>Salir de la estructura de repetición<br>Fin Si<br>Fin Para<br>retorna encontrado<br>Fin | BusquedaLinealCentinela(A,x)<br>Inicio<br>n=longitud de A<br>tmp=A[n-1]<br>A[n-1]=x<br>k=0<br>Mientras A[k] sea diferente de x<br>k=k+1<br>Fin Mientras<br>A[n-1]=tmp<br><br>Si k < n-1 o A[n-1]==x<br>retorna k<br>En otro caso<br>retorna -1<br>Fin Si<br>Fin |

1. Implemente cada función en algún lenguaje de programación y realice un programa que utilice las tres funciones anteriores para buscar en una lista 'A', un valor o llave 'x'. En este apartado se debe mostrar únicamente el número TOTAL de iteraciones que realiza cada función, así como el índice donde se localizó la llave.
   A=[33,25,16,38,56,1,5,9,18,96,25,14,76,32,25,17,1,4,8]
   x=1

Búsqueda lineal:

```
C:\Users\jesus\Desktop>py bus1.py
Numero de iteraciones: 19
Indice de la llave: 16
```

Búsqueda lineal mejorada:

```
C:\Users\jesus\Desktop>py bus2.py
Numero de iteraciones: 6
Indice de la llave: 5
```

Búsqueda lineal Centinela:

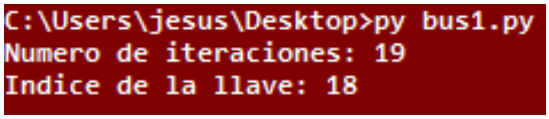
```
C:\Users\jesus\Desktop>py bus3.py
Numero de iteraciones: 6
Indice de la llave: 5
```

2. ¿Los resultados anteriores son iguales?, ¿Por qué?

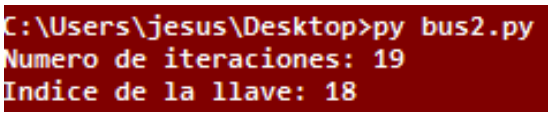
El primer resultado es diferente a los dos siguientes debido a que el algoritmo de búsqueda lineal recorre todo el arreglo, por lo tanto, el número de interaciones es igual al número de elementos que este tiene; además retorna la última posición donde se encontró la llave. El algoritmo de búsqueda lineal mejorado y el de búsqueda lineal Centinela son iguales debido a que solo hace 6 comparaciones y retorna la primera posición donde se encuentre la llave.

3. Ahora busque la llave x=8 en 'A' usando las mismas 3 funciones anteriores. Muestre los resultados y responda si los tres resultados son iguales, ¿por qué?

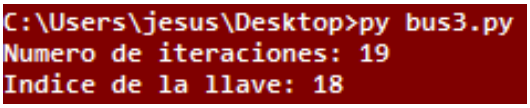
Búsqueda lineal:




Búsqueda lineal mejorada:




Búsqueda lineal Centinela:




Los tres resultados son iguales ya que la llave que se busca (x=8) se encuentra al final de la lista, y por lo tanto los tres algoritmos recorren todo el arreglo (peor caso).

## Actividad 2

A continuación, se proporcionan los pseudocódigos de las funciones para búsqueda binaria (solución iterativa y recursiva) explicados en clase.

| **# Búsqueda Binaria Iterativa** | **# Búsqueda Binaria Recursiva** |
|---|---|
| BusquedaBinariaIterativa(A,x)<br>Inicio<br>  Iteración=0<br>  indiceIzq=0<br>  indiceDer=Longitud de A -1<br>  Encontrado= -1<br>  Mientras indiceIzq <= indiceDer<br>  Iteración= Iteración+1;<br>  Medio=floor[(indiceIzq+indiceDer)/2]<br>  Si x == A[Medio] entonces | Iteración2 =0<br><br>BBRecursiva(A,x,indiceIzq,indiceDer)<br>Inicio<br>  global Iteración2<br>  Si indiceIzq > indiceDer<br>  Retorna -1, Iteración2<br>  Fin Si<br>  medio=floor[(indiceIzq+indiceDer)/2]<br>  Iteración2 = Iteración2+1 |

```
            Retorna Medio, Iteración
        Si no x > A[medio]
            indiceIzq=medio +1
        Si no
            indiceDer=medio-1
        Fin Si no
    Fin Mientras
    Retorna -1, Iteración
Fin
```

```
    Si x==A[medio]
        Retorna medio, Iteración2
    Si no x < A[medio]
        Retorna BBRecursiva (A,x,indiceIzq,medio-1)
    En otro caso
        Retorna BBRecursiva (A,x,medio+1, indiceDer)
    Fin Si
Fin
```

1. Implemente cada función en algún lenguaje de programación y realice un programa que utilice las 2 funciones anteriores para buscar en una lista 'A', un valor o llave 'x'. En este apartado se debe mostrar únicamente el número TOTAL de iteraciones que realiza cada función, así como el índice donde se localizó la llave.
   A=[1, 1, 1, 5, 8, 9, 14, 16, 17, 18, 25, 25, 25, 32, 33, 38, 56, 76, 96]
   x=1

Búsqueda binaria iterativa:

```
C:\Users\jesus\Desktop>py bin11.py
Numero de iteraciones: 3
Indice de la llave: 1
```

Búsqueda binaria recursiva:

```
C:\Users\jesus\Desktop>py bin22.py
Numero de iteraciones: 3
Indice de la llave: 1
```

2. ¿Los resultados anteriores son iguales?, ¿Por qué?

Sí, ya que en ambos algoritmos se va dividiendo el arreglo y se van haciendo una serie de pasos para encontrar la llave. Ambos dividen el arreglo de la misma forma, solo que uno es recursivo; como la complejidad es la misma para ambos algoritmos, el número de iteraciones y la posición de la llave es el mismo.

3. Qué pasa si usa los algoritmos de búsqueda binaria con el arreglo. Muestre y discuta los resultados.

   A=[33,25,16,38,56,1,5,9,18,96,25,14,76,32,25,17,1,4,8]
   x=1

Búsqueda binaria iterativa:

```
C:\Users\jesus\Desktop>py bin11.py
Numero de iteraciones: 4
Indice de la llave: -1
```

Búsqueda binaria recursiva:

```
C:\Users\jesus\Desktop>py bin22.py
Numero de iteraciones: 4
Indice de la llave: -1
```

En ambos casos el número de iteraciones es el mismo, y en ambos algoritmos no se encuentra la llave (x=1); esto debido a que la lista o arreglo está desordenado, por ello al dividir la lista no se asegura que en la parte izquierda se encuentren números menores al elemento medio, y tampoco se asegura que en la parte derecha solo se encuentren números mayores a este. Como solo se compara el número que está en medio de la lista al dividirla, si no está ordenada hay posibilidad de que no lo encuentre, aunque si esté en el arreglo.

## Actividad 3

1. Compare en términos de tiempo de procesamiento y de iteraciones el algoritmo de Busqueda Lineal Centinela y el algoritmo de Búsqueda Binaria Recursiva usando los arreglos de datos:

   A=[1, 1, 1, 5, 8, 9, 14, 16, 17, 18, 25, 25, 25, 32, 33, 38, 56, 76, 96]
   x=1

| | **Búsq. Lineal Centinela** | | **Búsq. Binaria Recursiva** | |
|---|---|---|---|---|
| | 0.0000082106 | 6 | 0.0001929495 | 4 |
| | 0.0000088948 | 6 | 0.0001635281 | 4 |
| | 0.0000082106 | 6 | 0.0001751598 | 4 |
| **Promedio** | **0.0000084386** | **6** | **0.0001772124** | **4** |

   A=[1, 1, 1, 5, 8, 9, 14, 16, 17, 18, 25, 25, 25, 32, 33, 38, 56, 76, 96]
   x=96

| | **Búsq. Lineal Centinela** | | **Búsq. Binaria Recursiva** | |
|---|---|---|---|---|
| | 0.0000068421 | 10 | 0.0000082106 | 1 |
| | 0.0000068421 | 10 | 0.0000088948 | 1 |
| | 0.0000075264 | 10 | 0.0000075264 | 1 |
| **Promedio** | **0.0000070702** | **10** | **0.0000082106** | **1** |

A=[1, 1, 1, 5, 8, 9, 14, 16, 17, 18, 25, 25, 25, 32, 33, 38, 56, 76, 96]
x=30

| | Búsq. Lineal Centinela | | Búsq. Binaria Recursiva | |
|---|---|---|---|---|
| | 0.0000088948 | 19 | 0.0001642123 | 5 |
| | 0.0000095790 | 19 | 0.0001477911 | 5 |
| | 0.0000088948 | 19 | 0.0001690019 | 5 |
| **Promedio** | **0.0000091228** | **19** | **0.0001603351** | **5** |

## 5. Código

En esta sección se presenta el código fuente del programa que permitió cumplir los objetivos propuestos.

```
#BusquedaLineal

def busquedaLineal(A,n,x):
        encontrado=-1
        for k in range(n):
                if A[k] == x:
                        encontrado=k
        print("Numero de iteraciones: "+str(k+1))
        return encontrado

A=[33,25,16,38,56,1,5,9,18,96,25,14,76,32,25,17,1,4,8]
ind=busquedaLineal(A,len(A),8)
print("Indice de la llave: "+str(ind))
```

```
#BusquedaLinealMejorada

def busquedaLinealMejorada(A,n,x):
        encontrado=-1
        for k in range(n):
                if A[k]==x:
                        encontrado=k
                        break
        print("Numero de iteraciones: "+str(k+1))
        return encontrado

A=[33,25,16,38,56,1,5,9,18,96,25,14,76,32,25,17,1,4,8]
x=busquedaLinealMejorada(A,len(A),8)
print("Indice de la llave: "+str(x))
```

```
#BusquedaLinealCentinela
import time

def busquedaLinealCentinela(A,n,x):
        tmp=A[n-1]
        A[n-1]=x
        while A[k] != x:
                k=k+1
        print("Numero de iteraciones: "+str(k+1))
        A[n-1]=tmp
        if k<n-1 or A[n-1]==x:
                return k
        else:
                return -1

A=[33,25,16,38,56,1,5,9,18,96,25,14,76,32,25,17,1,4,8]
tiempo_in=time.clock()
x=busquedaLinealCentinela(A,len(A),30)
print("Indice de la llave: "+str(x))
tiempo_fin=time.clock()
tiempo_tot=tiempo_fin-tiempo_in
print("Tiempo de procesamiento: "+str(tiempo_tot)+" segundos")
```

```
#BusquedaBinariaIterativa

def BusquedaBinIter(l,x,izquierda,derecha):
	i=0
	while izquierda<=derecha:
		i+=1
		medio = (izquierda+derecha)//2
		if l[medio]==x:
			print("Numero de iteraciones: "+str(i))
			return medio
		elif l[medio]<x:
			izquierda=medio+1
		else:
			derecha=medio-1
	print("Numero de iteraciones: "+str(i))
	return -1

l=[1,1,1,5,8,9,14,16,17,18,25,25,25,32,33,38,56,76,96]
ind=BusquedaBinIter(l,1,0,len(l)-1)
print("Indice de la llave: "+str(ind))
```

```
#BusquedaBinariaRecursiva
import time

def BusquedaBinRecursiva(l,x,izquierda,derecha,i):
	if izquierda > derecha:
		print("Numero de iteraciones: "+str(i))
		return -1
	medio = (izquierda+derecha)//2
	if l[medio]==x:
		i=i+1
		print("Numero de iteraciones: "+str(i))
		return medio
	elif l[medio]<x:
		i=i+1
		return BusquedaBinRecursiva(l,x,medio+1,derecha,i)
	else:
		i=i+1
		return BusquedaBinRecursiva(l,x,izquierda,medio-1,i)
```

```
l=[1,1,1,5,8,9,14,16,17,18,25,25,25,32,33,38,56,76,96]
tiempo_in=time.clock()
ind=BusquedaBinRecursiva(l,30,0,len(l)-1,0)
print("Indice de la llave: "+str(ind))
tiempo_fin=time.clock()
tiempo_tot=tiempo_fin-tiempo_in
print("Tiempo de procesamiento: "+str(tiempo_tot)+" segundos")
```

## 6. Conclusiones

En conclusión, los algoritmos de búsqueda son una herramienta fundamental para obtener datos o llaves e información en general de una lista o grupo de datos de gran tamaño. En esta práctica se trabajó con los algoritmos de búsqueda por comparación que en el mejor de los casos coinciden en su complejidad O(1), al igual que en el peor caso O(n).

En los tres primeros algoritmos (búsqueda lineal, lineal mejorada y centinela) se va comparando la llave que se busca con cada elemento del arreglo mientras que en los de búsqueda binaria se utiliza la técnica divide y conquista por lo que su complejidad es O(log n), es decir, el número de iteraciones es menor a comparación de los algoritmos lineales, por lo tanto es más eficiente. La única desventaja es que en la búsqueda binaria el arreglo debe estar ordenado lo cual conlleva a hacer uso de algún algoritmo de ordenamiento, por lo que su eficiencia sería afectada.

Finalmente, se cumplieron los objetivos ya que se analizaron y programaron los algoritmos de búsqueda por comparación de llaves.

## 7. Referencias


[1] Sáenz García, Elba Karen, Guía práctica de estudio 4, Algoritmos de búsqueda parte 1, Facultad de ingeniería, UNAM.

[2] CORMEN, Thomas, LEISERSON, Charles, et al. Introduction to Algorithms 3rd edition MA, USA The MIT Press, 2009

[3] KNUTH, Donald The Art of Computer Programming New Jersey Addison-Wesley Professional, 2011 Volumen.