Universidad Autónoma de Baja California
Facultad de Ciencias Químicas e Ingeniería

Algoritmos y Estructuras de Datos

Ciclo 2019-2

Práctica 10

"Análisis de Búsqueda Lineal, Binaria e Interpolada"

Profesor(a): M.I Alma Leticia Palacios Guerrero

Grupo: 551

Alumno:	Matrícula:
• Rodríguez Muñoz José Luis	1260368

Tijuana, Baja California, a 06 de noviembre de 2019

P10 Análisis de Búsqueda Lineal, Binaria e Interpolada

Competencia: Clasificar los algoritmos de búsqueda por su desempeño, mediante la implementación y prueba de los algoritmos de búsquedas en diversos escenarios, para poder seleccionar la más adecuada para la solución de problemas de manejo de información.

Búsqueda Lineal

El método de búsqueda más sencillo es la búsqueda lineal, este consiste en recorrer todo el vector comparando con la llave de búsqueda; es decir, el dato que se está buscando, después de encontrar el elemento que coincida regresa la posición donde lo encontró. Si la llave no se encuentra entonces regresa el método regresa el valor –1.

Búsqueda Binaria

Este método requiere que el arreglo este previamente ordenado. El algoritmo de la búsqueda binaria es:

1. Calcular el centro de la lista, con la fórmula (izquierdo + derecha) / 2. Izquierdo y derecho son las posiciones del elemento menor y mayor del vector.
2. Encontrar el elemento central del arreglo, la llave se compara con el centro si es igual aquí termina la búsqueda.
3. Si no es igual determinar si la llave se encuentra en el lado izquierdo o derecho de la lista.
4. Redefinir el inicio o el final según donde ese haya ubicado la llave. Si la llave es mayor que el centro entonces izquierdo = centro + 1. Si la llave es menor que el centro entonces derecho = derecho - 1.
5. Repetir desde el primer paso hasta encontrar el dato o hasta que ya no sea posible dividir más.
6. Si la llave no fue encontrada regresar -1.

Búsqueda Interpolada

Si los datos están distribuidos de forma uniforme, este método puede ser más eficiente que la búsqueda binaria. Básicamente el algoritmo es el mismo que el de la búsqueda binaria. La diferencia radica en que en la búsqueda interpolada no se busca el dato en el centro del arreglo, sino que se calcula su posición aproximada con la siguiente fórmula:

Búsqueda por interpolación que usa el valor de x, y los valores a los extremos $(i < j)$ del subarreglo actual, para *interpolar* la próxima posicion $g = i + \frac{(j-i)(x-A[i])}{A[j]-A[i]}$ a comparar con x.

DESARROLLO

Implemente los algoritmos de las búsquedas lineal, binaria e interpolada para un arreglo de tamaño N, utilizando inicialmente un arreglo tipo int de 20 posiciones.

RESULTADOS OBTENIDOS

	Tiempo (segundos)			**Cantidad de Iteraciones**		
	Búsqueda Lineal		**Búsqueda Binaria**		**Búsqueda Interpolada**	
Mejor de los casos	**Int** 0.000417 **Float** 0.000517	**Iteraciones** 2	**Int** 0.000417 **Float** 0.001752	**Iteraciones** 2	**Int** 0.000494 **Float** 0.001932	**Iteraciones** 2
Peor de los casos	Int 0.000838 **Float** 0.00173	3	**Int** 0.000440 **Float** 0.001612	4	**Int** 0.000361 **Float** 0.001022	4
Cualquier otro caso	**Int** 0.001102 **Float** 0.001174	3	**Int** 0.000701 **Float** 0.001427	4	**Int** 0.000345 **Float** 0.001132	4

CÓDIGO EN C

```c
#include <stdio.h>
#include <windows.h> // Para mostrar tiempo

// Funciones prototipo
void busquedaLineal(int arreglo[20]);
void busquedaBinaria(int arreglo[20]);
void busquedaInterpolada(int arreglo[20]);
void clearBuffer();
double performancecounter_diff(LARGE_INTEGER *a, LARGE_INTEGER *b);

int main()
{
    return menu();
}

// Implementación búsqueda lineal
void busquedaLineal(int arreglo[20])
{
    int busqueda, encontrado, i;
    printf("\n\t B%csqueda Lineal", 163);
    printf("\n\n Ingrese el n%cmero a buscar: ", 163);
    scanf("%d", &busqueda);
    encontrado = 0;

    for(i = 0; i < 20; i++)
    {
        if(arreglo[i] == busqueda) // Si lo encuentra la variable es igual a verdadero
            encontrado = 1;
    }
    if(encontrado == 1) // Si lo encuentra lo imprime
    {
        printf("\n %d se encuentra en el arreglo", busqueda, 162);
        printf("\n\n Lista de n%cmeros: ", 163);
        for(i = 0; i < 20; i++)
        {
            (i < 19)?printf(" %d ", arreglo[i]):printf(" %d ", arreglo[i]); // Se imprime el arreglo por monitoreo
        }
        getch();
    }
    else // En caso contrario no lo encuentra
```

```c
    else // En caso contrario no lo encuentra
    {
        printf("\n No se encontr%c ...", 162);
        printf("\n\n Lista de n%cmeros: ", 163);
        for(i = 0; i < 20; i++)
        {
            (i < 19)?printf(" %d ", arreglo[i]):printf(" %d ", arreglo[i]); // Se imprime el arreglo por monitoreo
        }
        getch();
    }
}

// Implementación búsqueda binaria
void busquedaBinaria(int arreglo[20])
{
    int i, busqueda, centro, izquierda = 0, derecha = 19, encontrado = 0;
    printf("\n\t B%csqueda Binaria", 163);
    printf("\n\n Ingresa el n%cmero a buscar: ", 163);
    scanf("%d", &busqueda);

    while(!encontrado && izquierda <= derecha)
    {
        centro = ((izquierda + derecha) / 2); // Fórmula para el centro del arreglo

        if(busqueda == arreglo[centro])
        {
            encontrado = 1; // Encuentra el elemento
        }
        else if(busqueda < arreglo[centro])
        {
            derecha = centro - 1; // Busca hacía la izquierda
        }
        else
        {
            izquierda = centro + 1; // Busca hacía la derecha
        }
    }
```

```
    if(encontrado) // Si se encuentra lo imprime
    {
        printf("\n %d se encuentra en la posici%cn %d", busqueda, 162, centro + 1);
        printf("\n\n Lista de n%cmeros: ", 163);
        for(i = 0; i < 20; i++)
        {
            (i < 19)?printf(" %d ", arreglo[i]):printf(" %d ", arreglo[i]); // Se imprime el arreglo por monitoreo
        }
        getch();
    }
    else // En caso contrario no lo encuentra
    {
        printf("\n No se encontr%c ...", 162);
        printf("\n\n Lista de n%cmeros: ", 163);
        for(i = 0; i < 20; i++)
        {
            (i < 19)?printf(" %d ", arreglo[i]):printf(" %d ", arreglo[i]); // Se imprime el arreglo por monitoreo
        }
        getch();
    }
}

// Implementación búsqueda interpolada Complejidad: (log log n)
void busquedaInterpolada(int arreglo[20])
{
    int i, g, busqueda, izquierda = 0, derecha = 19, encontrado = 0;
    printf("\n\t B%csqueda Interpolada", 163);
    printf("\n\n Ingresa el n%cmero a buscar: ", 163);
    scanf("%d", &busqueda);

    while(!encontrado && izquierda <= derecha)
    {
        g = (izquierda + ((busqueda - arreglo[izquierda]) * (derecha - izquierda)) / (arreglo[derecha] - arreglo[izquierda])); // Fórmula posición aproximada

        if(busqueda == arreglo[g])
        {
            encontrado = 1; // Si es igual se encuentra
        }
        else if(busqueda < arreglo[g])
        {
```

```
        else if(busqueda < arreglo[g])
        {
            derecha = g - 1; // Busca hacía la izquierda
        }
        else
        {
            izquierda = g + 1; // Busca hacía la derecha
        }
    }

    if(encontrado) // Si lo encuentra lo imprime
    {
        printf("\n %d se encuentra en la posici%cn %d", busqueda, 162, g + 1);
        printf("\n\n Lista de n%cmeros: ", 163);
        for(i = 0; i < 20; i++)
        {
            (i < 19)?printf(" %d ", arreglo[i]):printf(" %d ", arreglo[i]); // Se imprime el arreglo por monitoreo
        }
        getch();
    }
    else // En caso contrario no lo encuentra
    {
        printf("\n No se encontr%c ...", 162);
        printf("\n\n Lista de n%cmeros: ", 163);
        for(i = 0; i < 20; i++)
        {
            (i < 19)?printf(" %d ", arreglo[i]):printf(" %d ", arreglo[i]); // Se imprime el arreglo por monitoreo
        }
        getch();
    }
}

// Limpiar buffer
void clearBuffer()
{
    while(getchar() != '\n');
}
```

```
// Implementación para mostrar el tiempo de ejecución
double performancecounter_diff(LARGE_INTEGER *a, LARGE_INTEGER *b)
{
    LARGE_INTEGER frequencia;
    QueryPerformanceCounter(&frequencia);
    return (double)(a -> QuadPart - b -> QuadPart) / (double)frequencia.QuadPart;
}

int menu()
{
    int arreglo[20] = {3, 4, 7, 1, 2, 8, 9, 0, 2, 3, 5, 6, 12, 16, 15, 19, 15, 20, 30, 50};
    LARGE_INTEGER tiempoInicial, tiempoFinal;
    double segundos;

    char opcion;
    do
    {
        system("cls");
        printf("\n\t B%csquedas", 163);
        printf("\n\n 1) B%csqueda Lineal", 163);
        printf("\n 2) B%csqueda Binaria", 163);
        printf("\n 3) B%csqueda Interpolada", 163);
        printf("\n 4) Salir");
        printf("\n\n\t Opci%cn: ", 162);
        opcion = getchar();

        switch(opcion)
        {
        case '1':
            system("cls");
            clearBuffer();
            QueryPerformanceCounter(&tiempoInicial);
            busquedaLineal(arreglo);
            QueryPerformanceCounter(&tiempoFinal);
            segundos = performancecounter_diff(&tiempoFinal, &tiempoInicial);
            printf("\n\n\t Tiempo de ejecuci%cn: %f segundos\n", 162, segundos);
            getch();
            break;
```

```
        case '2':
            system("cls");
            clearBuffer();
            QueryPerformanceCounter(&tiempoInicial);
            busquedaBinaria(arreglo);
            QueryPerformanceCounter(&tiempoFinal);
            segundos = performancecounter_diff(&tiempoFinal, &tiempoInicial);
            printf("\n\n\t Tiempo de ejecuci%cn: %f segundos\n", 162, segundos);
            getch();
            break;

        case '3':
            system("cls");
            clearBuffer();
            QueryPerformanceCounter(&tiempoInicial);
            busquedaInterpolada(arreglo);
            QueryPerformanceCounter(&tiempoFinal);
            segundos = performancecounter_diff(&tiempoFinal, &tiempoInicial);
            printf("\n\n\t Tiempo de ejecuci%cn: %f segundos\n", 162, segundos);
            getch();
            break;

        }

    } while(opcion != '4' && opcion != EOF);
     {
        system("cls");
        printf("\n Cierre exitoso ...");
     }
    return 0;
}
```

CONCLUSIONES

- **¿Qué pasa al cambiar el tipo de dato del vector de int a float?**
  Varia un poco el tiempo de ejecución, tarda más. Debido a que el procesamiento del tipo de dato es diferente, en este caso int y float.

- **¿Qué pasa al aumentar el tamaño de N?**
  El tiempo de ejecución varia un poco, debido a que se debe de buscar un elemento en un arreglo más grande.

- **¿Qué pasa si el elemento NO está en el arreglo?**
  Tarda más en ejecutar las instrucciones, debido a que al no ser encontrado, estarías analizando en el peor de los casos, quiere decir, más tiempo de procesamiento.

- **¿Se puede utilizar la búsqueda binaria para cadenas? ¿Por qué?**
  Si se puede, de hecho, ya habíamos hecho una práctica de búsqueda binaria con una cadena previamente inicializada, y todo fluía correctamente.

- **¿Se puede utilizar la búsqueda interpolada para cadenas? ¿Por qué?**
  También se puede. La búsqueda interpolada es muy similar a la búsqueda binaria, de hecho, el algoritmo es el mismo solamente que en la búsqueda interpolada no se calcula el centro, sino la posición aproximada con una fórmula.