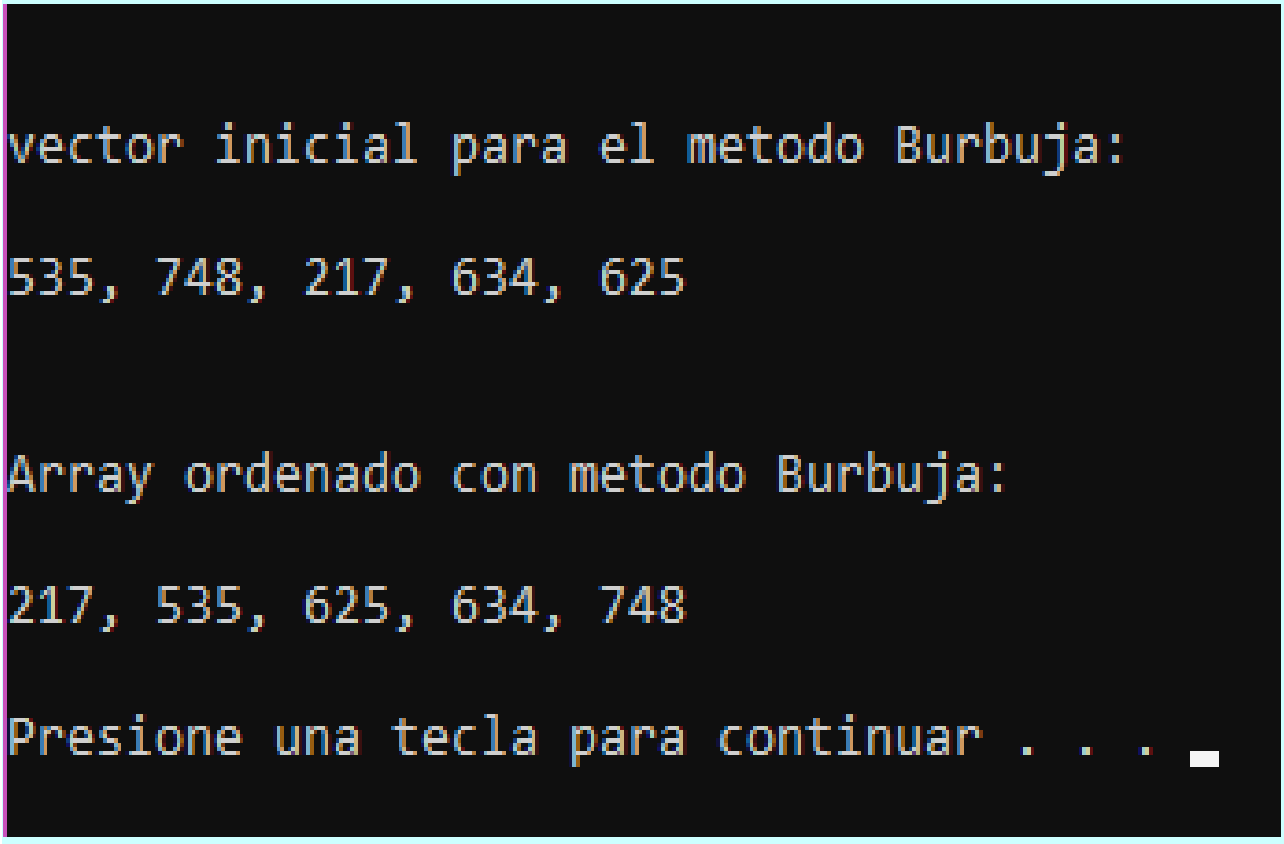
ANÁLISIS DE ALGORITMOS DE ORDENACIÓN



David Prieto Araujo david.prieto707@alu.uhu.es

Índice

Introducción

Los algoritmos de búsqueda consisten en localizar un elemento en una lista o vector, este puede estar ordenado o desordenado.

En esta practica vamos a estudiar el caso medio de los métodos de búsqueda Secuencial Iterativo, Binario Iterativo e Interpolación Iterativa. Estos métodos requieren que la lista este obligatoriamente ordenada.

```
vector inicial para el metodo SecuencialIt:

41, 334, 467
        Introduce la clave a buscar: 467


->   posicion de 467 buscado con el metodo SecuencialIt: 2

Presione una tecla para continuar . . .


vector inicial para el metodo BinariaIt:

169, 500, 724
        Introduce la clave a buscar: 500


->   posicion de 500 buscado con el metodo BinariaIt: 1

Presione una tecla para continuar . . .


vector inicial para el metodo InterpolacionIt:

358, 478, 962
        Introduce la clave a buscar: 358


->   posicion de 358 buscado con el metodo InterpolacionIt: 0
```

Cálculo del tiempo teórico y experimental

Código de los métodos de ordenacion(c++):

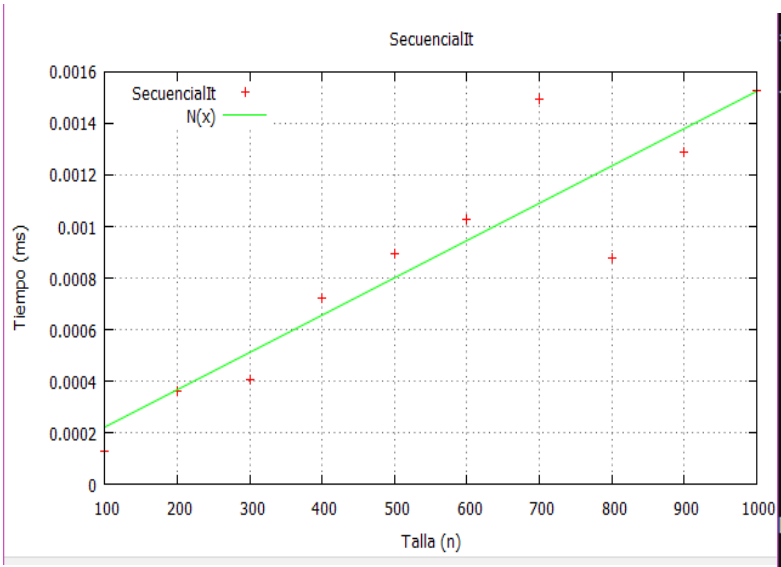
```
int AlgoritmosBusqueda::busquedaBinariaIt(int v[], int size,int key
{
    //** ESCRIBIR PARA COMPLETAR LA PRACTICA **//
    bool encontrado;
    int primero,ultimo,mitad;
    primero=0;
    ultimo=size-1;
    encontrado=false;
    while(primero<=ultimo&&!encontrado)
    {
        mitad=((primero+ultimo)/2);
        if(key==v[mitad])
            encontrado=true;
        else
        {
            if(key<v[mitad]){
                ultimo=mitad-1;}
            else if(key>v[mitad]){
                primero=mitad+1;}
        }
    }
    if(encontrado==true)
        return mitad;
    else
        return -1;
```

```
int AlgoritmosBusqueda::busquedaInterpolacionIt(int v[], int size,int key)
{
    //** ESCRIBIR PARA COMPLETAR LA PRACTICA **//
    int p,primero,ultimo;
    primero=0;
    ultimo=size-1;
    while(v[ultimo]>=key&&v[primero]<key)
    {
        p=primero+((ultimo-primero)*(key-v[primero]))/(v[ultimo]-v[primero]));
        if(key>v[p])
        {
            primero=p+1;
        }
        else
        {
            if(key<v[p]){
                ultimo=p-1;}
            else
                primero=p;
        }
    }
    if(v[primero]==key)
        return primero;
    else
        return -1;
}
```

David Prieto Araujo david.prieto707@alu.uhu.es

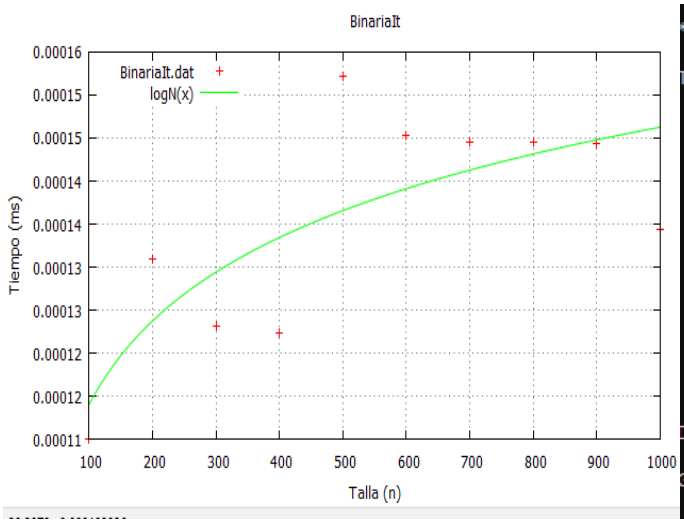
```
int AlgoritmosBusqueda::busquedaSecuencialIt(int v[], int size,int key)
{
   //** ESCRIBIR PARA COMPLETAR LA PRACTICA **//
    int i=0;
    while(v[i]!=key&&i<=size)
    {
        i++;
    }
    if(v[i]==key)
    {
        return i;
    }
    else
        return -1;
}
```

Método Secuencial Iterativo: Su caso medio es de orden N.



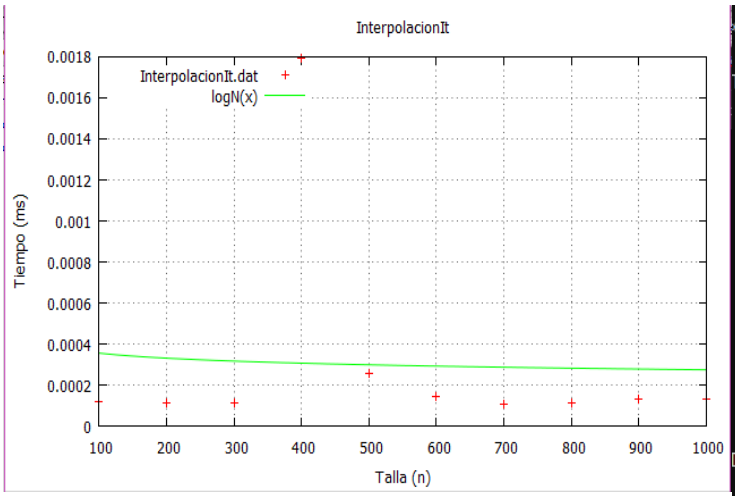

```
***Ordenacion por SecuencialIt ***

Tiempos de ejecucion promedio

        Talla           Tiempo (mseg)

        100               0.00014
        200                0.0017
        300               0.00055
        400               0.00076
        500               0.00089
        600                0.0011
        700                0.0011
        800                0.0008
        900                0.0025
        1000               0.0033

Datos guardados en el fichero SecuencialIt.dat

Generar grafica de resultados? (s/n):
```

Método Binario Iterativo: Su caso medio es de orden Log N.



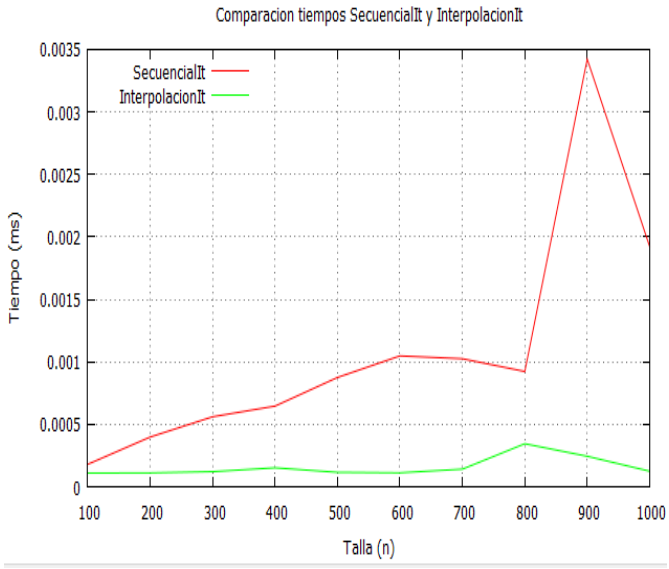

```
***Ordenacion por BinariaIt ***

Tiempos de ejecucion promedio

        Talla           Tiempo (mseg)

        100               0.00027
        200               0.00016
        300               0.00015
        400               0.00011
        500               0.00013
        600               0.00015
        700               0.00012
        800                0.0002
        900               0.00059
        1000              0.00058

Datos guardados en el fichero BinariaIt.dat

Generar grafica de resultados? (s/n):
```

Método Interpolación Iterativa: Su caso medio es de orden Log N.



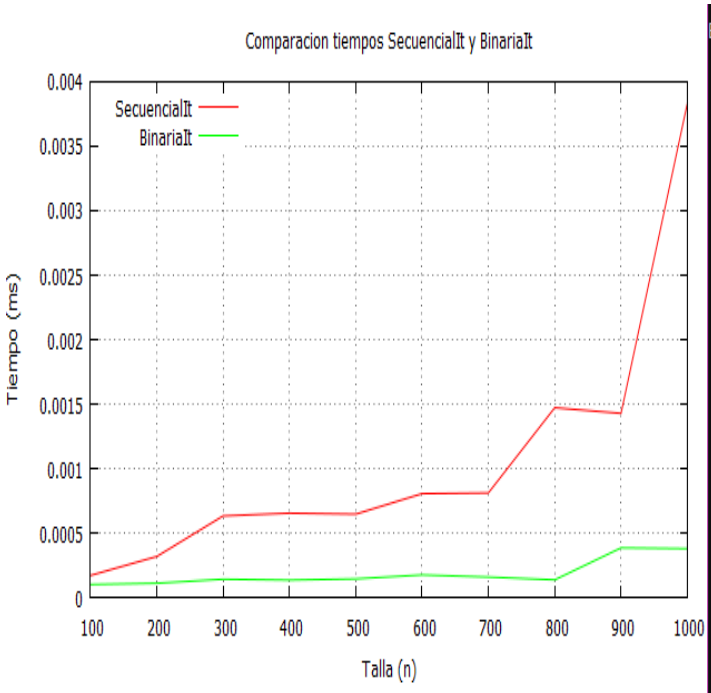

```
***Ordenacion por InterpolacionIt ***

Tiempos de ejecucion promedio

        Talla           Tiempo (mseg)

        100                0.00014
        200                0.00021
        300                0.00016
        400                0.00012
        500                0.00012
        600                0.00011
        700                0.00013
        800                0.00013
        900                0.00036
        1000               0.00048

Datos guardados en el fichero InterpolacionIt.dat

Generar grafica de resultados? (s/n):
```

Comparación de los métodos

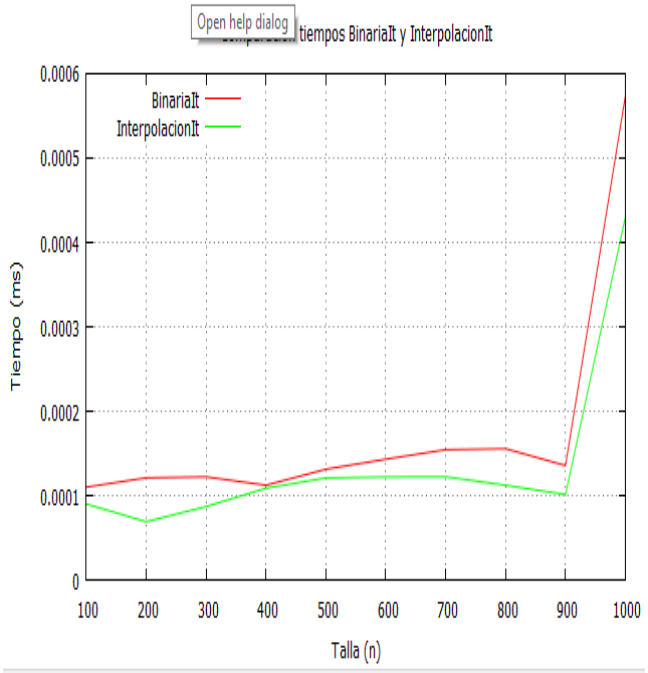



```
Busqueda SecuencialIt y InterpolacionIt. Tiempos de ejecucion promedio

                SecuencialIt          InterpolacionIt
    Talla(n)    Tiempo<mseg>    Tiempo<mseg>

    100             0.00013         9e-005

    200             0.00026        0.00011

    300              0.0004         0.0001

    400             0.00054         9e-005

    500             0.00064        0.00014

    600             0.00074        0.00011

    700             0.00096        0.00011

    800             0.00099        0.00012

    900               0.004        0.00035

    1000              0.0027       0.00038


Grafica guardada en el fichero SecuencialIt.dat y InterpolacionIt.dat
```

David Prieto Araujo david.prieto707@alu.uhu.es




```
Busqueda SecuencialIt y BinariaIt. Tiempos de ejecucion promedio

                SecuencialIt          BinariaIt
     Talla(n)   Tiempo<mseg>    Tiempo<mseg>

     100           0.00015        0.00012

     200           0.00034        0.00012

     300           0.00047        0.00011

     400           0.00058         0.0001

     500           0.00074        0.00012

     600             0.001        0.00017

     700           0.00092        0.00015

     800           0.00082        0.00012

     900            0.0012        0.00014

     1000           0.0027        0.00032


Grafica guardada en el fichero SecuencialIt.dat y BinariaIt.dat
```




```
Busqueda BinariaIt y InterpolacionIt. Tiempos de ejecucion promedio

                BinariaIt             InterpolacionIt
     Talla(n)   Tiempo<mseg>    Tiempo<mseg>

     100           0.00011          8e-005

     200           0.00013        0.00011

     300           0.00012        0.00011

     400           0.00016        0.00011

     500           0.00016        0.00013

     600           0.00014        0.00012

     700           0.00011        0.00011

     800           0.00014         0.0001

     900           0.00045        0.00038

     1000          0.00056        0.00046


Grafica guardada en el fichero BinariaIt.dat y InterpolacionIt.dat
```

Conclusiones:

Como se puede observar en las gráficas y en las tablas de las comparaciones entre los métodos en base al tiempo de eficiencia, el algoritmo más eficiente de estos 3 sería el método de Interpolación Iterativa, a continuacion sería el de Binario Iterativo y por último el Secuencial Iterativo, ya que este es de orden N a diferencia de los otros dos ya que son

de orden logarítmico.

Diseño de la aplicación

Para llevar a cabo la práctica, se ha hecho uso del programa codeblocks para ejecutar el código(del cual se ha implementado el archivo 'Principal.cpp', 'Graficas.cpp', 'AlgoritmosBusqueda.cpp' y 'TestBusqueda.cpp' y en el lenguaje de programación 'c++') y gnplot para crear las gráficas.