1. Resultados experimentales

1.1. Inicialización

Para realizar los experimentos, se utilizaron tres algoritmos de búsqueda: búsqueda secuencial, búsqueda binaria y búsqueda galopante. Se creó un `std::vector` poblado con enteros consecutivos de 32 bits desde 0 hasta $n-1$, donde n varía según el experimento. Estos datos fueron generados internamente y no se utilizaron datasets externos.

Se realizaron pruebas sobre cada algoritmo ejecutándolo en 4000 repeticiones para cada tamaño de entrada. Se midió tanto el tiempo promedio por operación como la desviación estándar, con el fin de obtener resultados estadísticamente significativos y mitigar el efecto de fluctuaciones de rendimiento aleatorias.

1.2. Primer experimento

El primero `<exp1>` considera diferentes configuraciones en los tamaños de entrada:

```
for (int i = 0; i < n; i++)
        data[i] = i;
```

donde n es el largo del vector y varía según parámetros que serán definidos y discutidos posteriormente. El objetivo a buscar se define como `int target = n - 1` para todos los algoritmos.

1.3. Segundo experimento

El segundo experimento varía la posición del objetivo a buscar. Se define una variable `int64_t fixed_n = 3000000` que representa un vector de largo fijo, y se define `int target = n - 1` como la posición a buscar.

1.4. Especificaciones

Los experimentos se realizaron en una MacBook Air con CPU Apple M1 que posee 8 núcleos a 3.2 GHz y 8 GB LPDDR4X unificada a 4266 MHz. Dado que los algoritmos no involucran memoria secundaria, todas las operaciones se realizaron en memoria principal (RAM).

1.5. Compilación

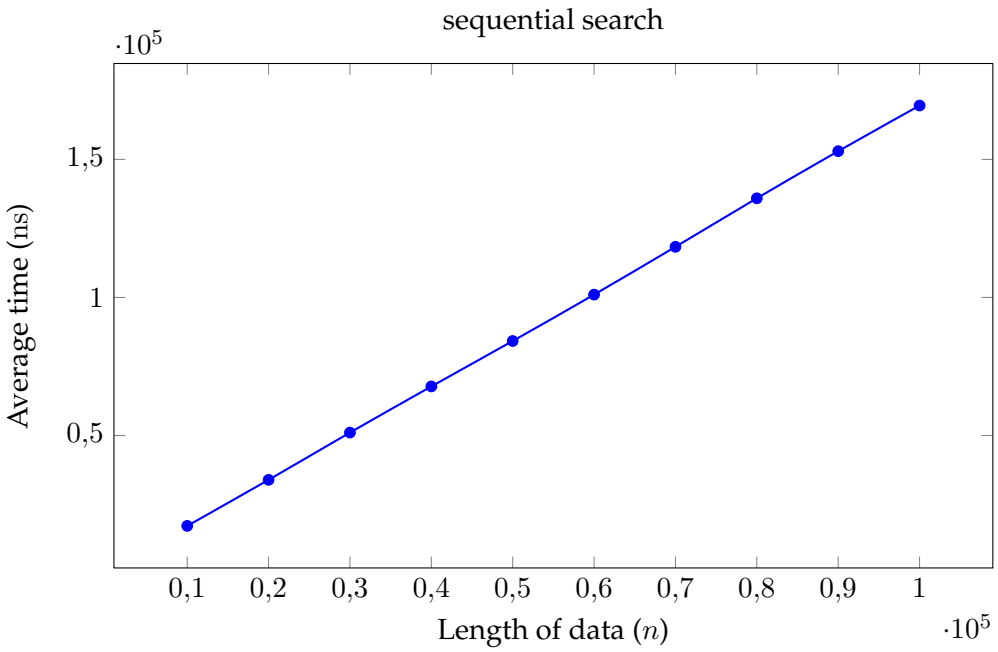
Para automatizar el análisis experimental se utilizó la herramienta `uhr`, disponible en el repositorio del curso. Esta se compiló usando el compilador GNU G++ como:

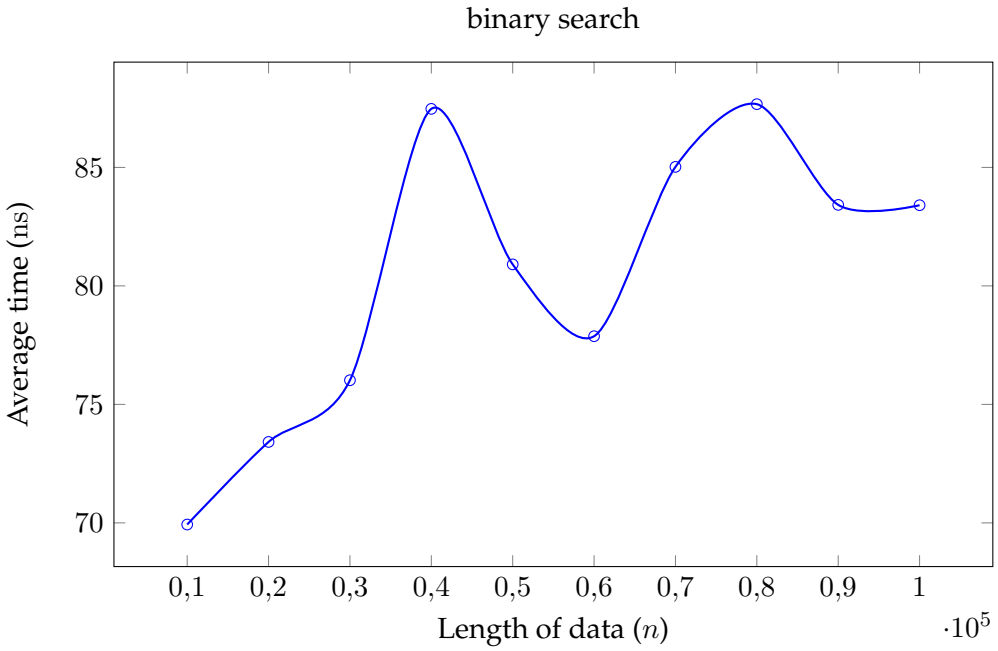
```
g++ -O0 -std=c++11 -Wall -Wpedantic -o <algorithm>_<exp> uhr_<exp>.cpp
```

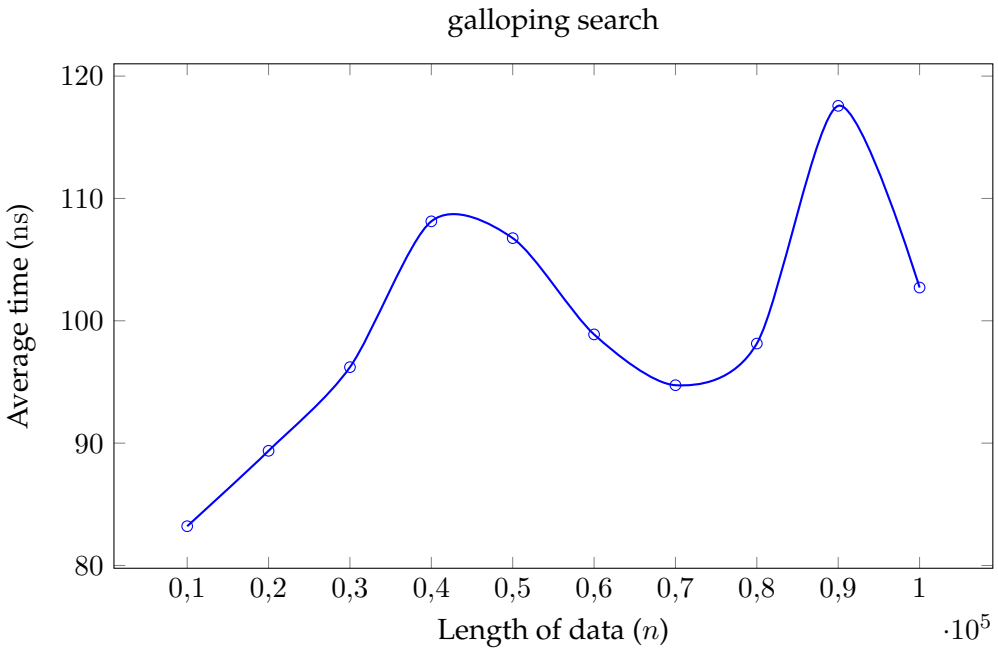
donde `<algorithm>` corresponde al algoritmo utilizado y `<exp>` al experimento. Cada experimento generó 3 ejecutables (uno por cada algoritmo) que luego fueron automatizados mediante un script de bash para probar distintos valores de LOWER, UPPER y STEP. El detalle se encuentra en el archivo `run_experiments.sh`.

2. Resultados experimento 1

Se utilizaron los valores de ...


sequential search
$\cdot 10^5$
1,5
1
0,5
Average time (ns)
0,1
0,2
0,3
0,4
0,5
0,6
0,7
0,8
0,9
1
Length of data (n)
$\cdot 10^5$


binary search
85
80
75
70
Average time (ns)
0,1
0,2
0,3
0,4
0,5
0,6
0,7
0,8
0,9
1
Length of data (n)
$\cdot 10^5$


galloping search
Average time (ns)
120
110
100
90
80
0,1
0,2
0,3
0,4
0,5
0,6
0,7
0,8
0,9
1
Length of data (n)
$\cdot 10^5$