Algorítmica

Grado en Ingeniería Informática

# Ejercicio de clase

## Búsqueda ternaria

**Autores**
Carlos Sánchez Páez



Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2017-2018

# Índice



# Índice de cuadros

# 1. Enunciado

Realizar un estudio empírico para determinar si es preferible utilizar la búsqueda binaria o la búsqueda ternaria comentada en clase (ambos algoritmos son de orden logarítmico, pero sus constantes ocultas son diferentes).

# 2. Resolución

## 2.1. Metodología

Para resolver el ejercicio, ejecutaremos 25 veces cada código con tamaños de problema ascendentes mediante un script. Después, estudiaremos empíricamente su eficiencia y hallaremos el valor de sus constantes ocultas (eficiencia híbrida).

| **Algoritmo** | **Tamaño inicial** | **Tamaño final** | **Incremento** |
|---|---|---|---|
| Búsqueda Binaria<br><br>Búsqueda Ternaria | 100.000.000 | 340.000.000 | 10.000.000 |

Cuadro 1: Tamaños para la ejecución

## 2.2. Resultados obtenidos

| Tamaño del problema | Búsqueda Binaria | Búsqueda Ternaria |
|---|---|---|
| 100.000.000 | | |
| 110.000.000 | | |
| 120.000.000 | | |
| 130.000.000 | | |
| 140.000.000 | | |
| 150.000.000 | | |
| 160.000.000 | | |
| 170.000.000 | | |
| 180.000.000 | | |
| 190.000.000 | | |
| 200.000.000 | | |
| 210.000.000 | | |
| 220.000.000 | | |
| 230.000.000 | | |
| 240.000.000 | | |
| 250.000.000 | | |
| 260.000.000 | | |
| 270.000.000 | | |
| 280.000.000 | | |
| 290.000.000 | | |
| 300.000.000 | | |
| 310.000.000 | | |
| 320.000.000 | | |
| 330.000.000 | | |
| 340.000.000 | | |

Cuadro 2: Tiempos obtenidos (seg)

## 2.3. Anexo:Algoritmos desarrollados

### 2.3.1. Búsqueda binaria

```
#define TEST 0        //Imprimir o no el resultado
#include <iostream>
#include <chrono>
#include <ctime>
#include <ratio>
#include <chrono>
using namespace std;
using namespace std::chrono;

int busquedaBinaria(const int *v, const int i, const int j, const int buscado) {
        //Caso base: nos cruzamos
        if (i > j) {
                return -1;
        }
        else {
                int mitad = (i + j) / 2;
                if (v[mitad] == buscado)                //Acertamos
                        return mitad;
                else if (buscado < v[mitad])        //Buscamos hacia la izquierda
                        return busquedaBinaria(v, i, mitad - 1, buscado);
                else                                                    //Buscamos hacia la derecha
                        return busquedaBinaria(v, mitad + 1, j, buscado);
        }
}
int main(int argc, char **argv) {
        if (argc != 2) {
                cerr << "Uso del programa: " << argv[0] << " <tamaño>" << endl;
                exit(-1);
        }
        int tam = atoi(argv[1]);
        int *v = new int[tam];
        high_resolution_clock::time_point tantes;
        high_resolution_clock::time_point tdespues;
        duration<double> tiempo;

        //Inicializamos vector
        for (int i = 0; i < tam; i++)
                v[i] = i;

        //Peor caso: el elemento no existe
        tantes = high_resolution_clock::now();
        int pos = busquedaBinaria(v, 0, tam, tam);
        tdespues = high_resolution_clock::now();
        tiempo = duration_cast<duration<double>>(tdespues - tantes);
#if TEST
        cout << "Posición: " << pos << endl;
#endif
        cout << tam << "\t\t" << tiempo.count() << endl;
}
```

### 2.3.2. Búsqueda ternaria

```cpp
#define TEST 0        //Imprimir o no el resultado
#include <iostream>
#include <chrono>
#include <ctime>
#include <ratio>
#include <chrono>
using namespace std;
using namespace std::chrono;

int busquedaTernaria(const int *v, const int i, const int j, const int buscado){
        //Caso base: nos cruzamos
        if (i > j)
                return -1;
        else {
                int tam = j - i;
                int tercio = i + (tam / 3);
                int dostercio = i + (tam * 2) / 3;
                if (v[tercio] == buscado) //Acertamos
                        return tercio;
                else if (buscado < v[tercio]) //Primer tercio
                        busquedaTernaria(v,i, tercio - 1, buscado);
                else {
                        if (v[dostercio] == buscado)        //Acertamos
                                return dostercio;
                        else if (buscado > v[dostercio]) //Tercer tercio
                                busquedaTernaria(v,dostercio + 1, j, buscado);
                        else //Segundo tercio
                                busquedaTernaria(v,tercio + 1, dostercio - 1, buscado);
                }
        }
}

int main(int argc, char **argv) {
        if (argc != 2) {
                cerr << "Uso del programa: " << argv[0] << " <tamaño>" << endl;
                exit(-1);
        }
        int tam = atoi(argv[1]);
        int *v = new int[tam];
        high_resolution_clock::time_point tantes;
        high_resolution_clock::time_point tdespues;
        duration<double> tiempo;

        //Inicializamos vector
        for (int i = 0; i < tam; i++)
                v[i] = i;

        //Peor caso: el buscado no eviste
        tantes = high_resolution_clock::now();
        int pos = busquedaTernaria(v, 0, tam, tam);
        tdespues = high_resolution_clock::now();
        tiempo = duration_cast<duration<double>>(tdespues - tantes);

#if TEST
        cout << "Posición: " << pos << endl;
#endif
```

```
57          cout << tam << "\t\t" << tiempo.count() << endl;
58  }
```

### 2.3.3. Script para múltiples ejecuciones

```
1   #!/bin/bash
2   if [ $# -eq 3 ]
3   then
4           i="0"
5           tam=$2
6           #Primer argumento: programa a ejecutar
7           #Segundo argumento: tamaño inicial
8           #Tercer argumento : incremento
9           while [ $i -lt 25 ]
10          do
11                  ./$1 $tam >> ./$1.dat
12                  i=$[$i+1]
13                  tam=$[$tam+$3]
14          done
15  else
16          echo "Error de argumentos"
17  fi
```

## 2.4. Script de GNUPlot

```
1   #!/usr/bin/gnuplot
2
3   set xlabel "Tamanio del problema"
4   set ylabel "Tiempo (seg)"
5   set terminal png size 640,480
6   set output 'busqueda_binaria.png'
7   plot 'busqueda_binaria.dat' with lines
8   set output 'busqueda_ternaria.png'
9   plot 'busqueda_ternaria.dat' with lines
```

## 2.5. Script automatizado

```
1   #!/bin/bash
2           echo "Compilando..."
3           g++ -o busqueda_binaria busqueda_binaria.cpp
4           g++ -o busqueda_ternaria busqueda_ternaria.cpp
5           rm -f ./busqueda_binaria.dat ;
6           rm -f ./busqueda_ternaria.dat ;
7           echo "Ejecutando búsqueda binaria..."
8           ./individual.sh busqueda_ternaria 100000000 10000000;
9           echo "Ejecutando búsqueda ternaria..."
10          ./individual.sh busqueda_binaria 100000000 10000000;
11          echo "Generando gráficas..."
12          ./gnuplot.sh
```