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

Índice de figuras

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 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**

Cuadro 2: Tiempos obtenidos (seg)

2.3. Cálculo de la constante oculta

Realizamos una regresión mediante *gnuplot* para averiguar la constante:

Algoritmo	**Valor de la constante oculta**	**Porcentaje de error**
Búsqueda Binaria		
Búsqueda Ternaria		

Cuadro 3: Bondad del ajuste

2.4. Gráficas

Figura 1: Eficiencia empírica. Búsqueda binaria

Figura 2: Eficiencia empírica. Búsqueda ternaria

Figura 3: Eficiencia híbrida. Búsqueda binaria

Figura 4: Eficiencia híbrida. Búsqueda ternaria

2.5. Conclusiones

2.6. Anexo:Algoritmos desarrollados

2.6.1. 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;

        //Inicializar vector con valores aleatorios
        srand (time(NULL));

        for (int i = 0; i < tam; i++)
                v[i] = rand() ;
        // Peor caso: no está
        tantes = high_resolution_clock::now();
        int pos = busquedaBinaria(v, 0, tam, -1);
        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;

        delete []v;
}
```

2.6.2. Búsqueda ternaria

```
#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;

        //Inicializar vector con valores aleatorios
        srand (time(NULL));

        for (int i = 0; i < tam; i++)
                v[i] = rand() ;
        // Peor caso: no está
        tantes = high_resolution_clock::now();
        int pos = busquedaTernaria(v, 0, tam, -1);
        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;

        delete []v;
}
```

2.6.3. Script para múltiples ejecuciones

```
#!/bin/bash
if [ $# -eq 3 ]
then
        i="0"
        tam=$2
        #Primer argumento: programa a ejecutar
        #Segundo argumento: tamaño inicial
        #Tercer argumento : incremento
        while [ $i -lt 25 ]
        do
                ./$1 $tam >> ./$1.dat
                i=$[$i+1]
                tam=$[$tam+$3]
        done
else
        echo "Error de argumentos"
fi
```

2.6.4. Script de *gnuplot*

```
#!/usr/bin/gnuplot

set xlabel "Tamanio del problema"
set ylabel "Tiempo (seg)"
set terminal png size 640,480
set output 'busqueda_binaria.png'
plot 'busqueda_binaria.dat' with lines
set output 'busqueda_ternaria.png'
plot 'busqueda_ternaria.dat' with lines
```

2.6.5. Script automatizado

```
#!/bin/bash
        echo "Compilando..."
        g++ -o busqueda_binaria busqueda_binaria.cpp &&
        g++ -o busqueda_ternaria busqueda_ternaria.cpp &&
        rm -f ./busqueda_binaria.dat ;
        rm -f ./busqueda_ternaria.dat ;
        echo "Ejecutando búsqueda binaria..." ;
        ./individual.sh busqueda_binaria 10000000 10000000;
        echo "Ejecutando búsqueda ternaria..." ;
        ./individual.sh busqueda_ternaria 10000000 10000000;
        echo "Generando gráficas..." ;
        ./gnuplot.sh ;
```