Algoritmos Divide y Vencerás

Ana García Muñoz
Manuel Sánchez Pérez
Víctor Gutiérrez Rubiño
Jose María Ramírez González

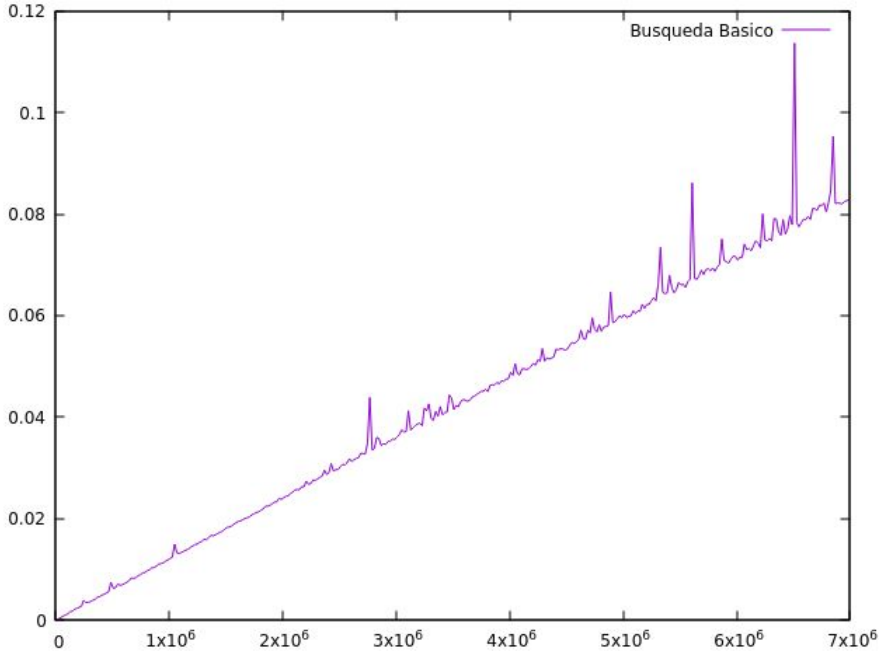
Estudio del algoritmo básico

Eficiencia Teórica

```
for (it = myvector.begin(); it != myvector.end(); ++it)
{
  if (cont == *it)
  {
    encontrado = true;
    pos = cont;
  }
  cont++;
}
```

En el peor de los casos, recorre el vector entero, luego es **O(N)**

Eficiencia empírica




```
6470000 0.079648
6490000 0.077883
6510000 0.113627
6530000 0.078314
6550000 0.077481
6570000 0.078256
6590000 0.078972
6610000 0.078858
6630000 0.079562
6650000 0.078907
6670000 0.081097
6690000 0.081023
6710000 0.080717
6730000 0.081759
6750000 0.081644
6770000 0.082143
6790000 0.080381
6810000 0.082195
6830000 0.084548
6850000 0.095299
6870000 0.082068
6890000 0.082172
6910000 0.08208
6930000 0.082004
6950000 0.082432
6970000 0.08264
```

Saltos de 20.000 en 20.000, desde 10.000 hasta los 7 millones de elementos.

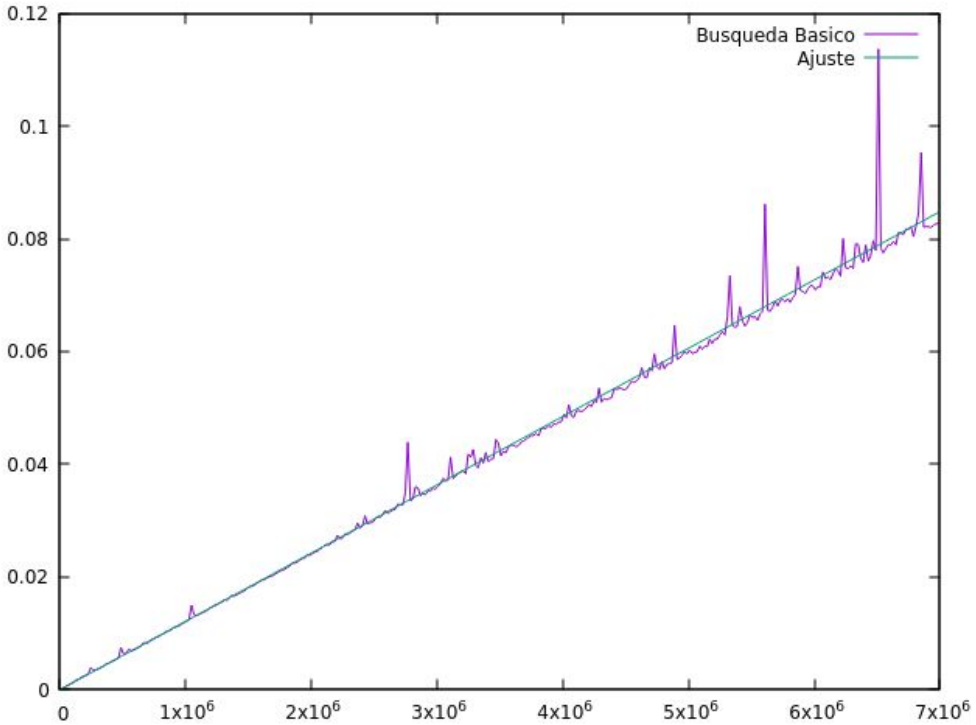
Eficiencia híbrida

Ecuación de ajuste:

T(n) = a0 * n

Constantes ocultas:

```
Final set of parameters            Asymptotic Standard Error
======================             ==========================
a0              = 1.21135e-08      +/- 3.319e-11    (0.274%)
```



Estudio del algoritmo divide y vencerás

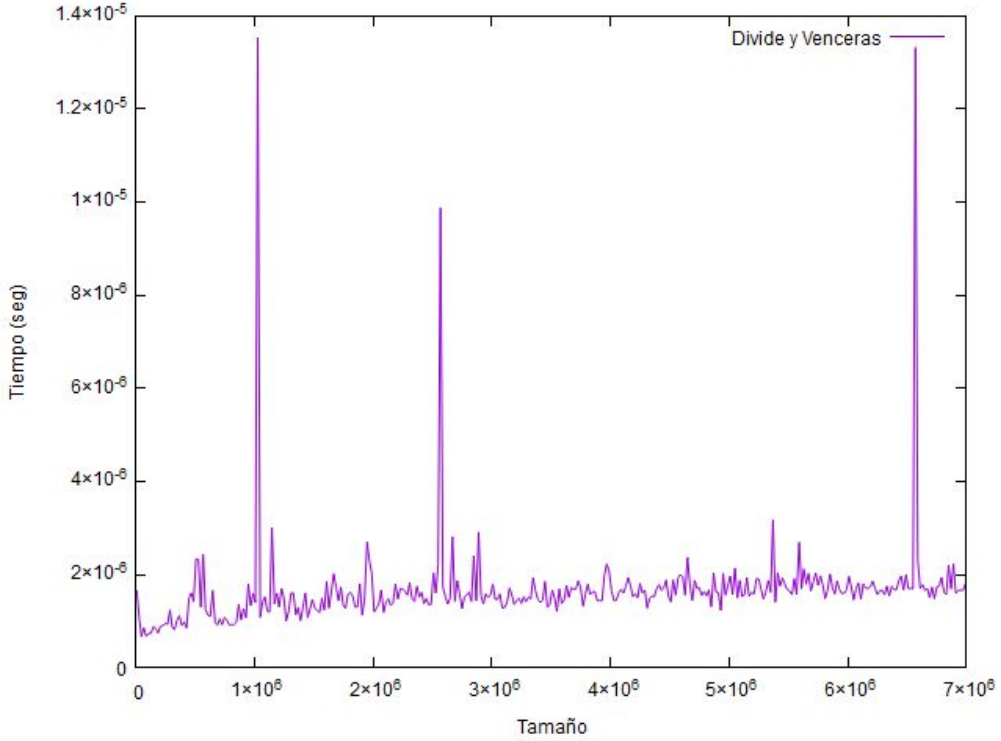
Eficiencia teórica

```
void divide_venceras(vector<int> &vect, int inicio, int fin) {
    int tam = fin - inicio + 1;
    if (tam > 10 && !encontrado) {
        int half = tam/2;
        half += inicio;
        if (vect.at(half) == half) {
            encontrado = true;
            posicion = half;
        } else if(vect.at(half) > half){
            divide_venceras(vect, inicio, half);
        } else{
            divide_venceras(vect, half, fin);
        }

    } else if(!encontrado){
        lineal(vect, inicio, fin);
    }
}
```

En el peor de los casos, recorre el vector de tal forma que 2^n = tam , luego es **O(log(n))**

Eficiencia empírica




```
6130000 1.801e-06
6150000 1.713e-06
6170000 1.699e-06
6190000 1.735e-06
6210000 1.864e-06
6230000 1.698e-06
6250000 1.57e-06
6270000 1.657e-06
6290000 1.668e-06
6310000 1.568e-06
6330000 1.756e-06
6350000 1.543e-06
6370000 1.752e-06
6390000 1.679e-06
6410000 1.676e-06
6430000 1.869e-06
6450000 1.972e-06
6470000 1.669e-06
6490000 2e-06
6510000 1.699e-06
6530000 1.708e-06
6550000 1.695e-06
6570000 1.331e-05
```

Saltos de 20.000 en 20.000, desde 10.000 hasta los 7 millones de elementos.

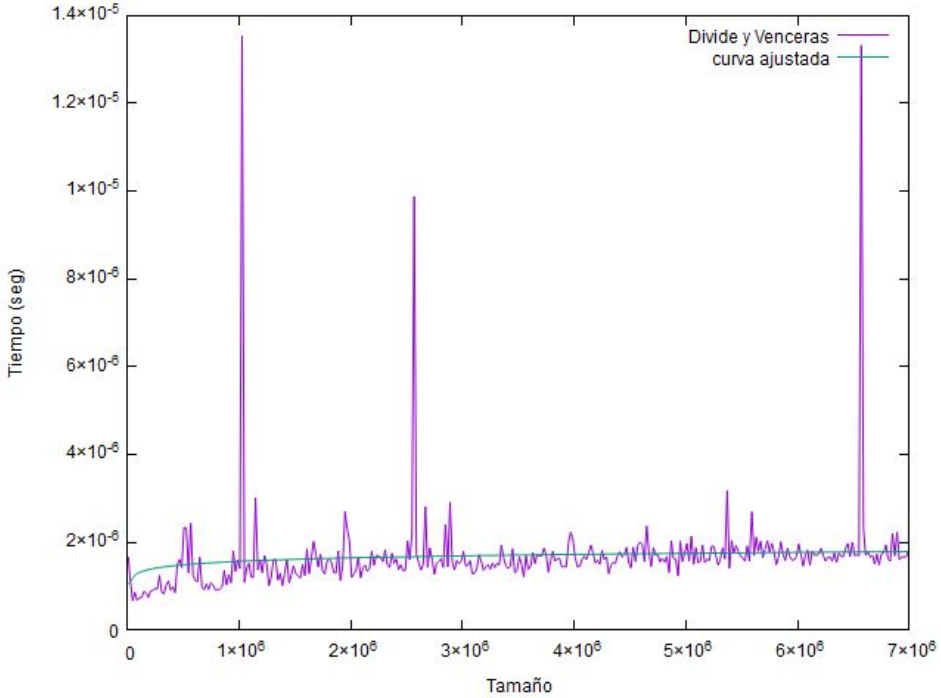
Eficiencia híbrida

Ecuación de ajuste:

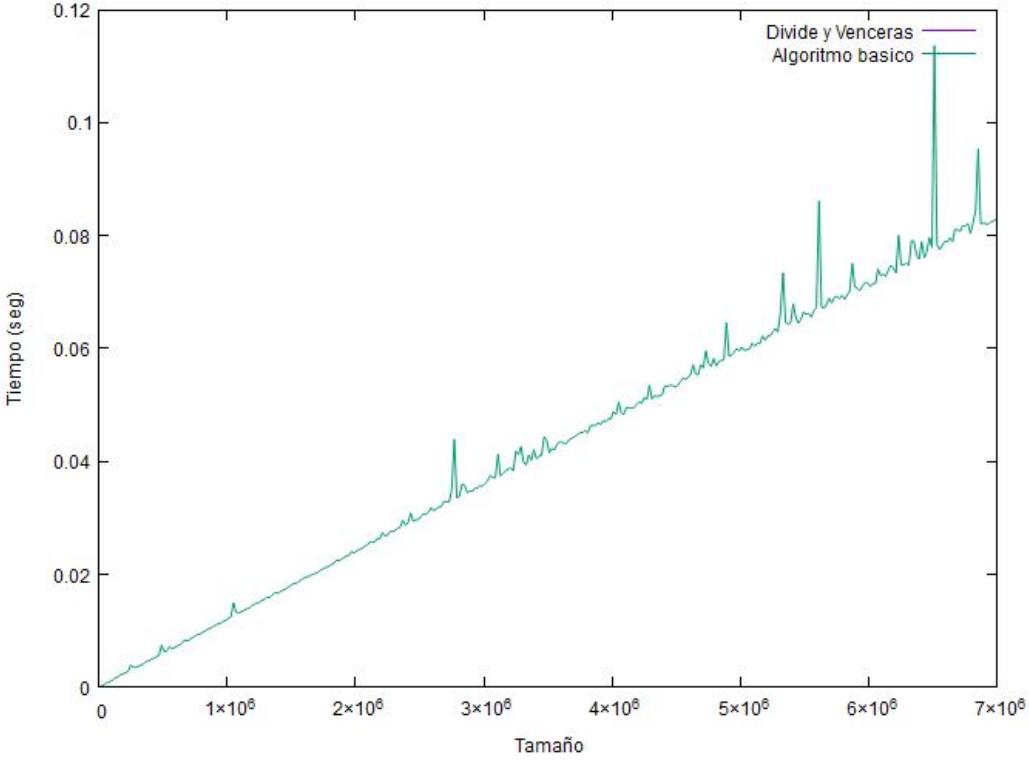
T(n) = a0 * log(n)

Constantes ocultas:

```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a0              = 1.13383e-07      +/- 3.774e-09    (3.329%)
```



Comparativa de ambos algoritmos


Divide y Venceras
Algoritmo basico
Tiempo (seg)
Tamaño
0.12
0.1
0.08
0.06
0.04
0.02
0
0
1×10^6
2×10^6
3×10^6
4×10^6
5×10^6
6×10^6
7×10^6