Práctica 2 – Divide y vencerás

El elemento en su posición – Grupo 5

Juan Carlos Ruiz García, Guillermo Gómez Trenado, Miguel Ángel Rispal Martínez, Ignacio Irurita Contreras, Joaquín FernándezLeón, Daniel García Martos.

Contenido

1. Tarea a realizar

Dado un vector ordenado (de forma no decreciente) de números enteros* v*, todos distintos, el objetivo es determinar si existe un índice* i *tal que* v[i] = i *y encontrarlo en ese caso. Diseñar e implementar un algoritmo "divide y vencerás" que permita resolver el problema. ¿Cuál es la complejidad de ese algoritmo y la del algoritmo "obvio" para realizar esta tarea? Realizar también un estudio empírico e híbrido de la eficiencia de ambos algoritmos.

Supóngase ahora que los enteros no tienen por qué ser todos distintos (pueden repetirse). Determinar si el algoritmo anterior sigue siendo válido, y en caso negativo proponer uno que sí lo sea. ¿Sigue siendo preferible al algoritmo obvio?

2. Introducción

Para la realización de esta práctica hemos realizado **3 algoritmos**:

- *Algoritmo ''obvio''.*
  - *Vector_1 inicializado sin repetidos y sin éxito.*
  - *Vector_2 inicializado sin repetidos y con éxito.*
  - Vector_3 inicializado con repetidos y éxito indeterminado.
- *Algoritmo ''Divide y vencerás sin repetidos''.*
  - *Vector_1 inicializado sin repetidos y sin éxito.*
  - *Vector_2 inicializado sin repetidos y con éxito.*
  - Vector_3 inicializado con repetidos y éxito indeterminado.
- *Algoritmos ''Divide y vencerás con repetidos''.*
  - *Vector_1 inicializado sin repetidos y sin éxito.*
  - *Vector_2 inicializado sin repetidos y con éxito.*

Además de inicializar cada vector de la forma descrita anteriormente, los tamaños de los vectores han ido variando desde **10** hasta **55010**, con un incremento de **25000**, consiguiendo un total de **20** tomas de tiempo por vector y algoritmo. Por consecuente, para que los tiempos sean más verídicos, cada toma ha sido obtenida de la **media** de **15 iteraciones** del mismo vector con el mismo tamaño sobre el mismo algoritmo.

3. Algoritmo obvio

El funcionamiento de este algoritmo es muy básico. Recibe como parámetros el **vector**, la posición de **inicio** y la posición de **fin** y simplemente realiza una **búsqueda secuencial** en dicho vector **desde ini hasta fin** para ver si en algún caso **mi_vector[i] == i**. En caso de acierto devuelve el valor de **i** y en otro caso -**1**.

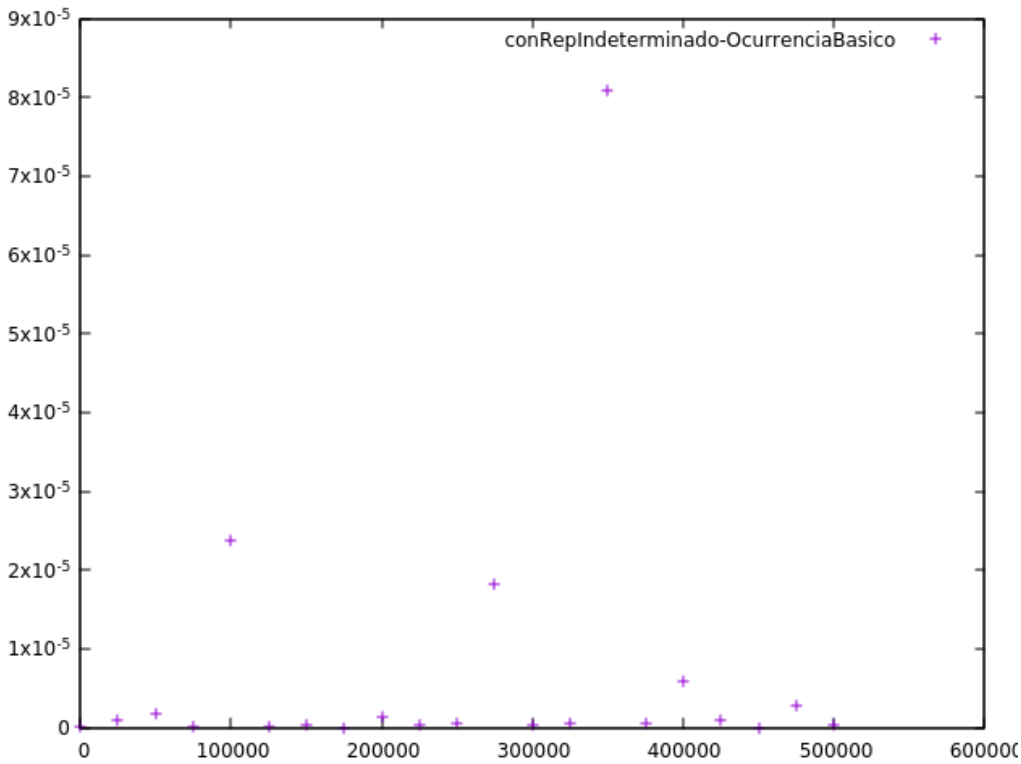
La eficiencia de este algoritmo es **O(n)**.

```
int encontrarOcurrenciaBasico(vector<int>&v, int ini, int fin){
      bool encontrado = false;
    int pos3 = -1;
      for (int i = ini; i < fin && encontrado == false; i++)
    {
     if (v[i] == i)
     {
       encontrado = true;
       pos3 = i;
     }
    }
      return pos3;
}
```

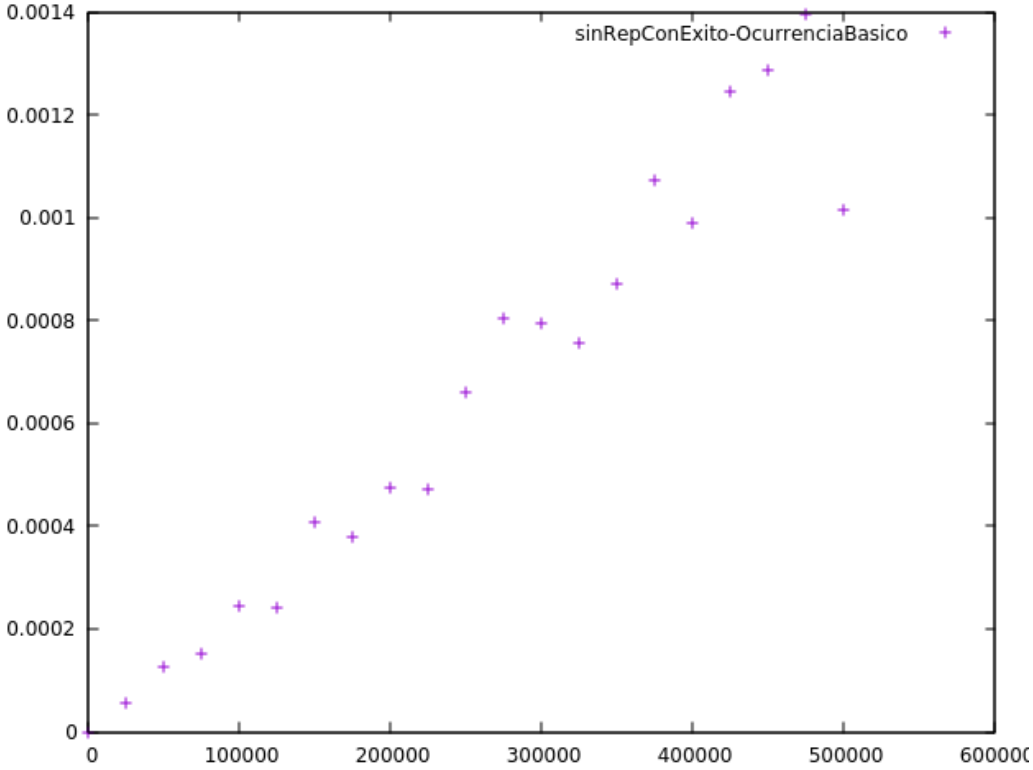
A continuación, mostramos las distintas gráficas obtenidas de las diferentes ejecuciones de los 3 vectores en este algoritmo.

3.1. Eficiencia Empírica

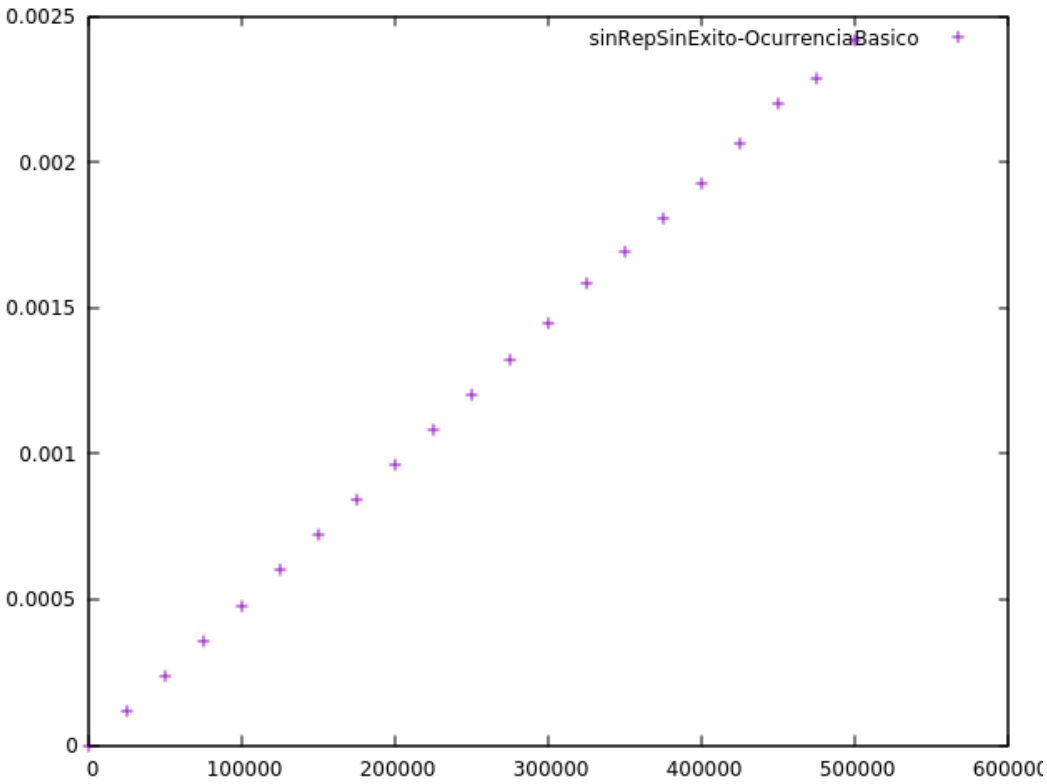
Con elementos repetidos y ocurrencia indeterminada



Sin elementos repetidos y éxito en ocurrencia

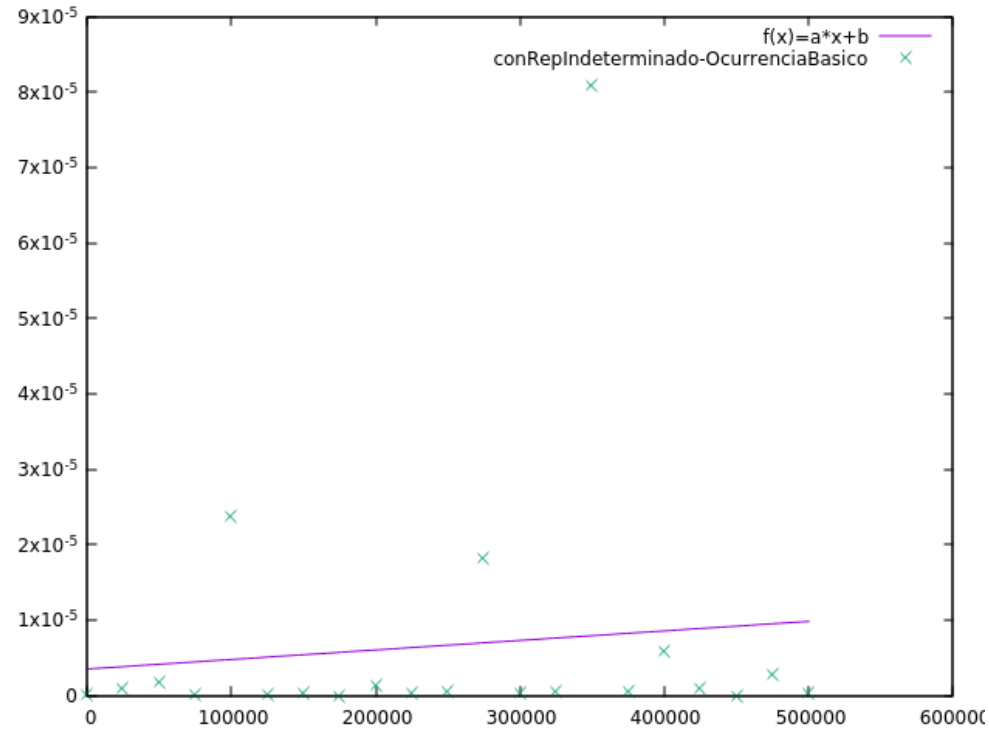


Sin elementos repetidos y sin éxito en ocurrencia

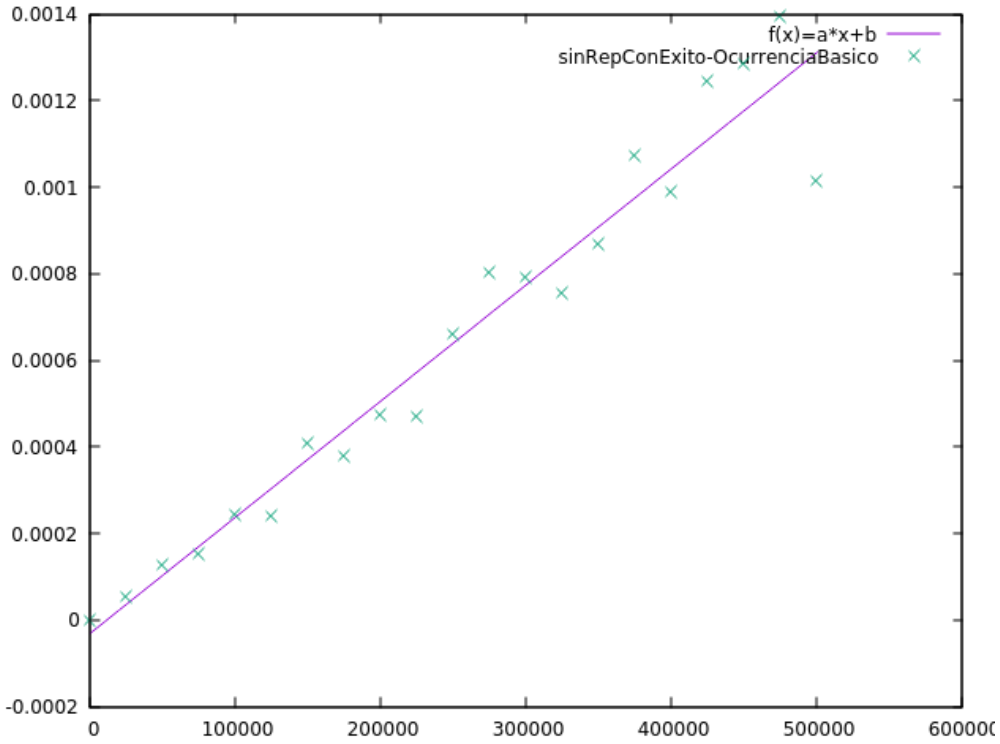


3.2. Eficiencia Híbrida (ajuste lineal)

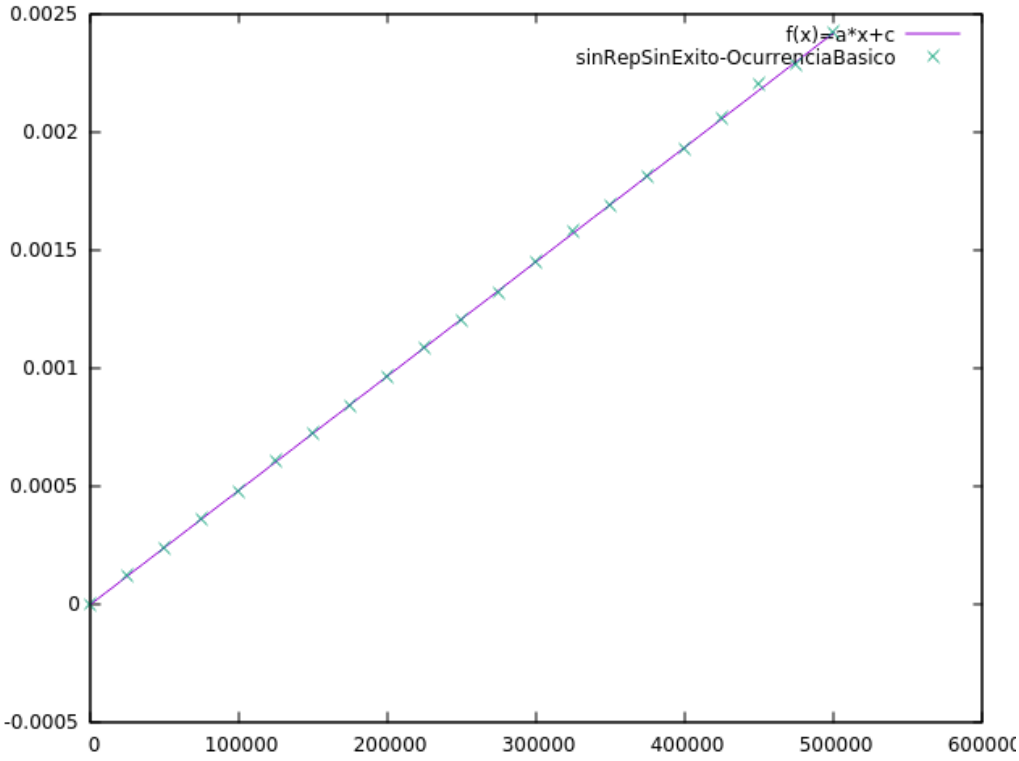
Con elementos repetidos y ocurrencia indeterminada



Sin elementos repetidos y éxito en ocurrencia



Sin elementos repetidos y sin éxito en ocurrencia



4. Algoritmo “Divide y vencerás sin repetidos”

Este algoritmo basa su funcionamiento en la *técnica divide y vencerás*. En nuestro caso, dividiremos el vector en **dos mitades** de forma **recursiva**, buscando en todo momento que la posición del **medio** de nuestro vector sea igual a el valor de **v[medio]** devolviendo el valor **medio** en este caso o, en caso contrario pudiendo llegar al **caso base** en el que **ini=fin**, momento en el que sabremos que no existe ninguna ocurrencia.

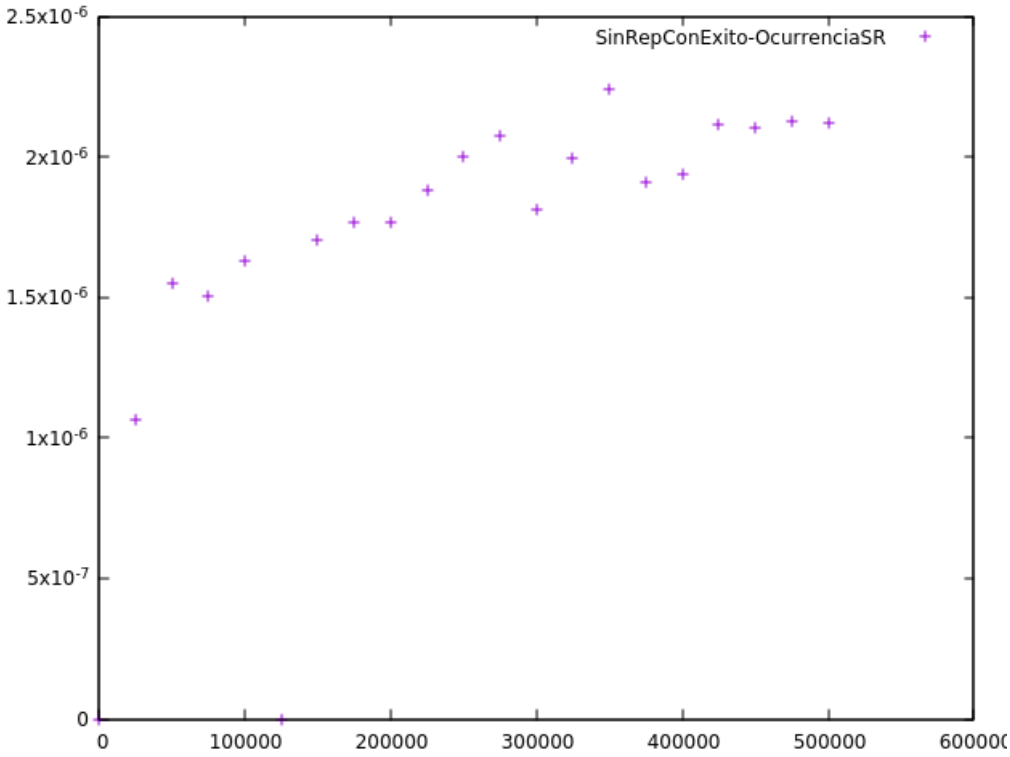
La eficiencia de este algoritmo es **O(logn)**.

```
int encontrarOcurrenciaSR(const vector<int> & v, int ini, int fin){
    // cout << "##" << ini << ' '  << fin << ' '  <<(ini + fin)/2 <<
endl;
    if(ini == fin){
      return -1;
    }
    else{
        int medio = (ini + fin)/2;
        int res;
        if(v[medio] == medio){
            return medio;
        }
        else{
            if( v[medio] < medio){
                res = encontrarOcurrenciaSR(v,medio+1,fin);
                return res;
            }
            else{
                res = encontrarOcurrenciaSR(v,ini,medio);
                return res;
            }
        }
    }
}
```

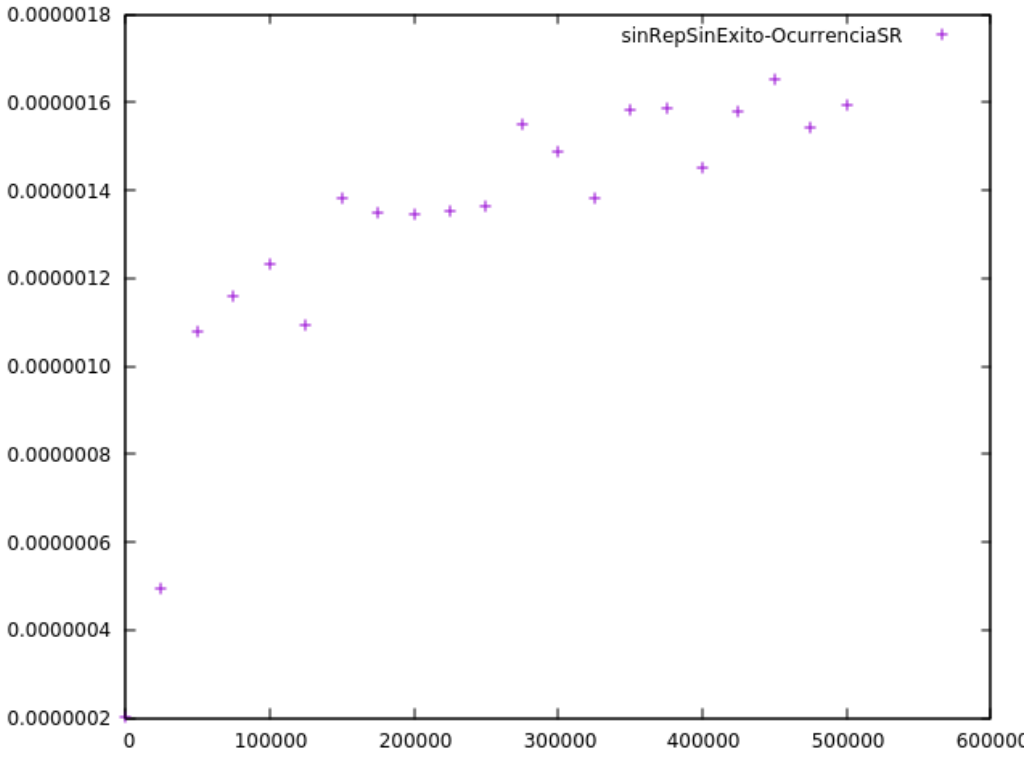
A continuación, mostramos las distintas gráficas obtenidas de las diferentes ejecuciones de los 2 vectores en este algoritmo (solo 2 vectores debido a que este algoritmo no funciona con un vector con repetidos).

4.1. Eficiencia Empírica

Sin elementos repetidos y con éxito en ocurrencia

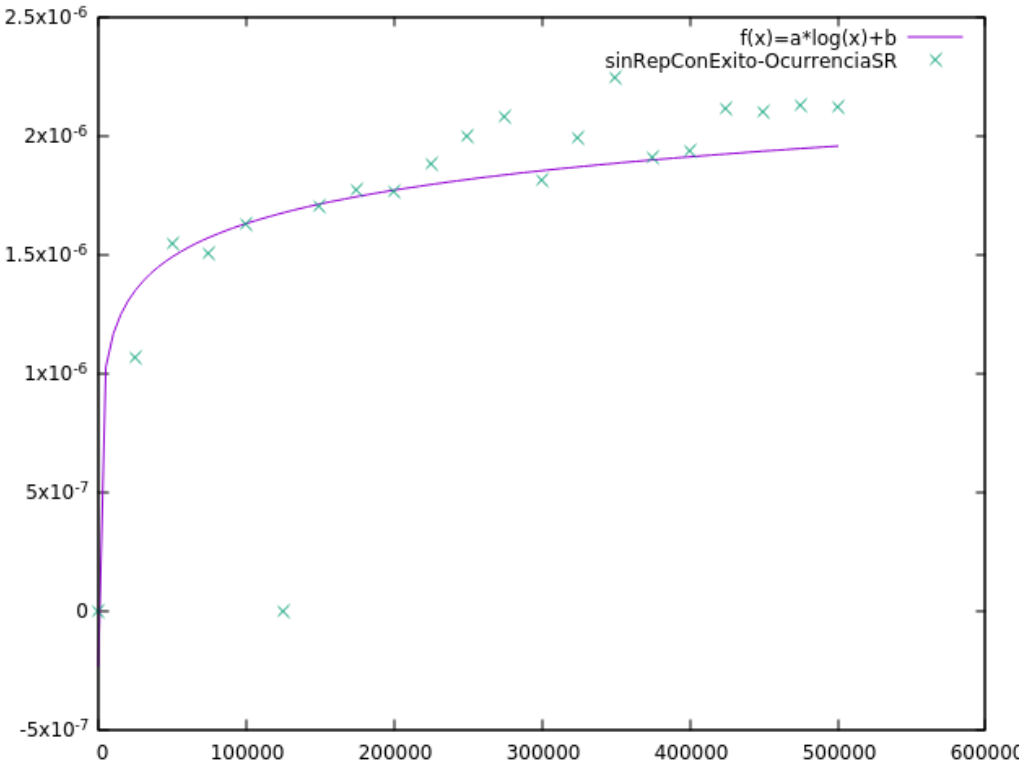


Sin elementos repetidos y sin éxito en ocurrencia

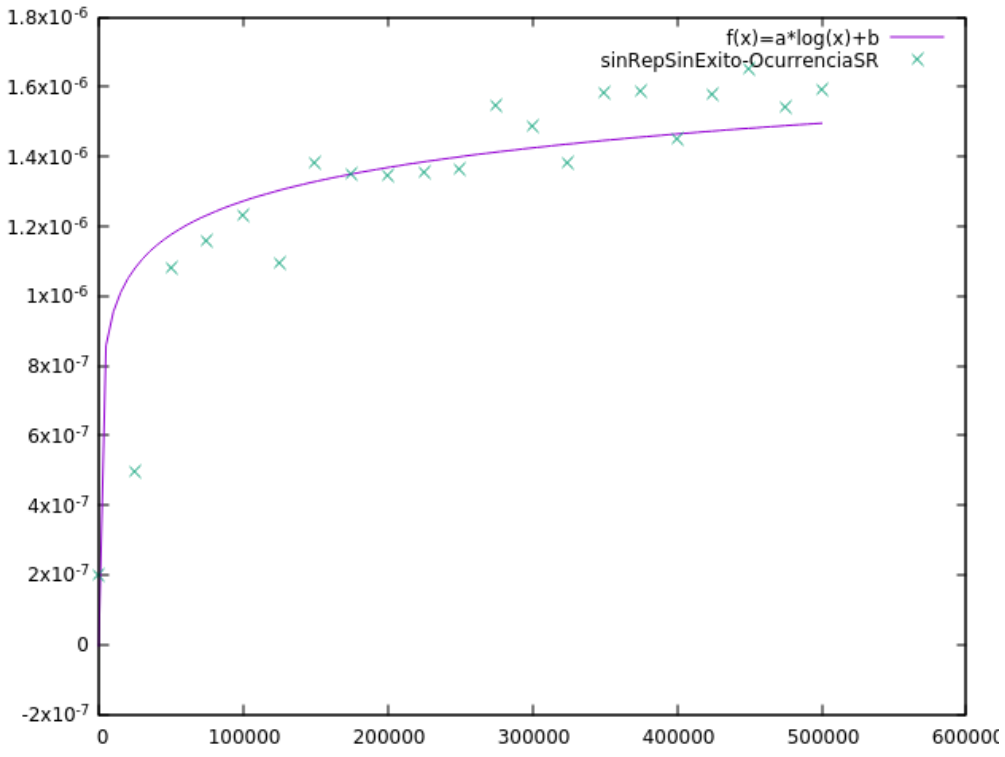


4.2. Eficiencia Híbrida (ajuste logarítmico)

Sin elementos repetidos y con éxito en ocurrencia



Sin elementos repetidos y sin éxito en ocurrencia



5. Algoritmo “Divide y vencerás con repetidos”

Este algoritmo al igual que el anterior también se basa en la *técnica divide y vencerás*. Su funcionamiento es muy similar al anterior, salvando la excepción de que ahora podemos **tener elementos repetidos** en el vector. Ahora lo que hacemos es **no descartar** una de las dos mitades y buscar en ambas, ya que al haber elementos repetidos **no podemos asegurarnos de en qué lado es posible que exista una ocurrencia**.

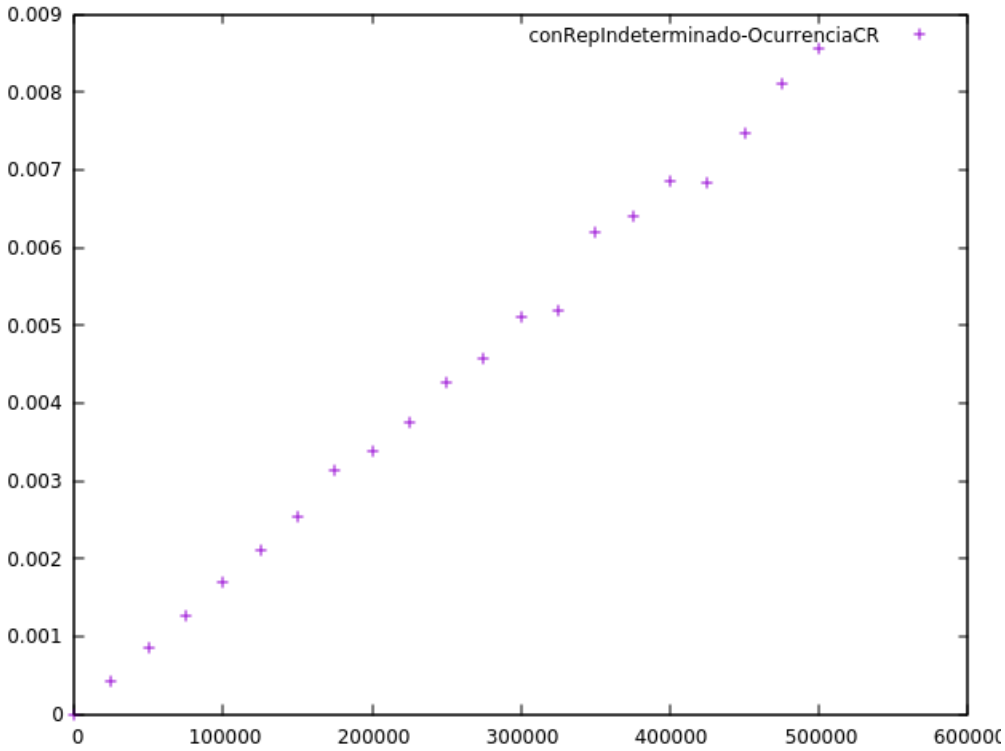
Como dividimos el vector en **dos mitades** de forma **recursiva** pero **no descartamos ninguna mitad** al final acabamos recorriendo el vector completamente, por lo que la eficiencia de este algoritmo es **O(n)**.

```
int encontrarOcurrenciaCR(vector<int>&v, int ini, int fin)
{
    int pos = -1;

    if(ini==fin){
        if(v[ini]==ini)
            pos = ini;
    }
    else{
        int izq,dch,mitad;
        mitad=(ini+fin)/2;

        izq=encontrarOcurrenciaCR(v,ini,mitad);
        dch=encontrarOcurrenciaCR(v,mitad+1,fin);

        if(izq>=0)
            pos=izq;
        else if(dch>=0)
            pos=dch;
    }

    return pos;
}
```

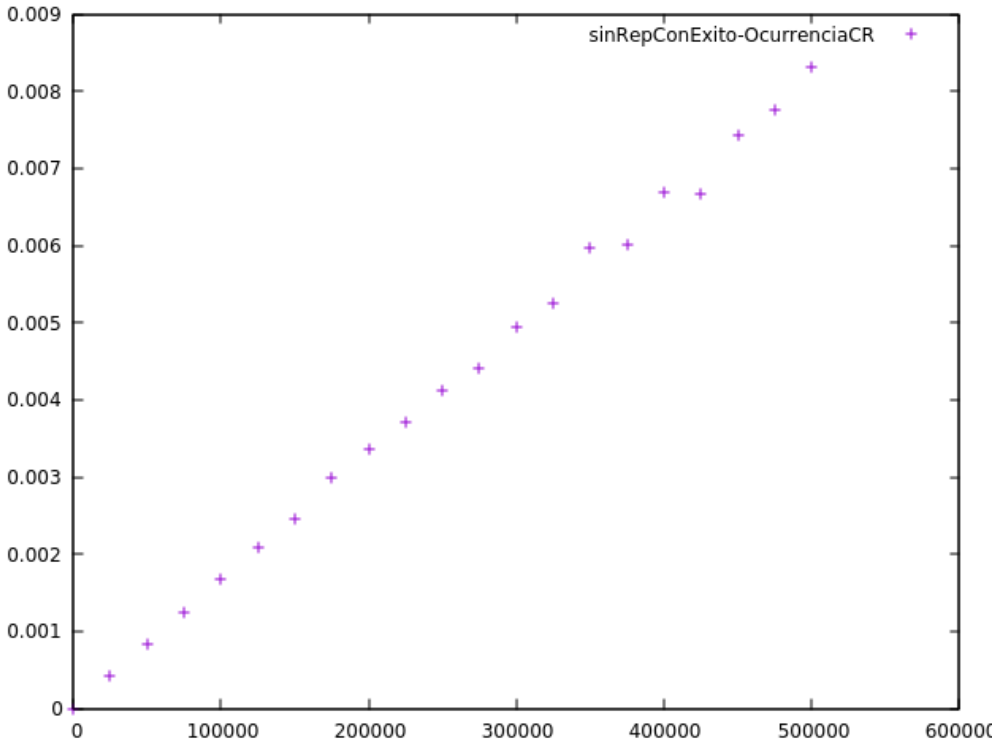
A continuación, mostramos las distintas gráficas obtenidas de las diferentes ejecuciones de los 3 vectores en este algoritmo.

5.1. Eficiencia Empírica

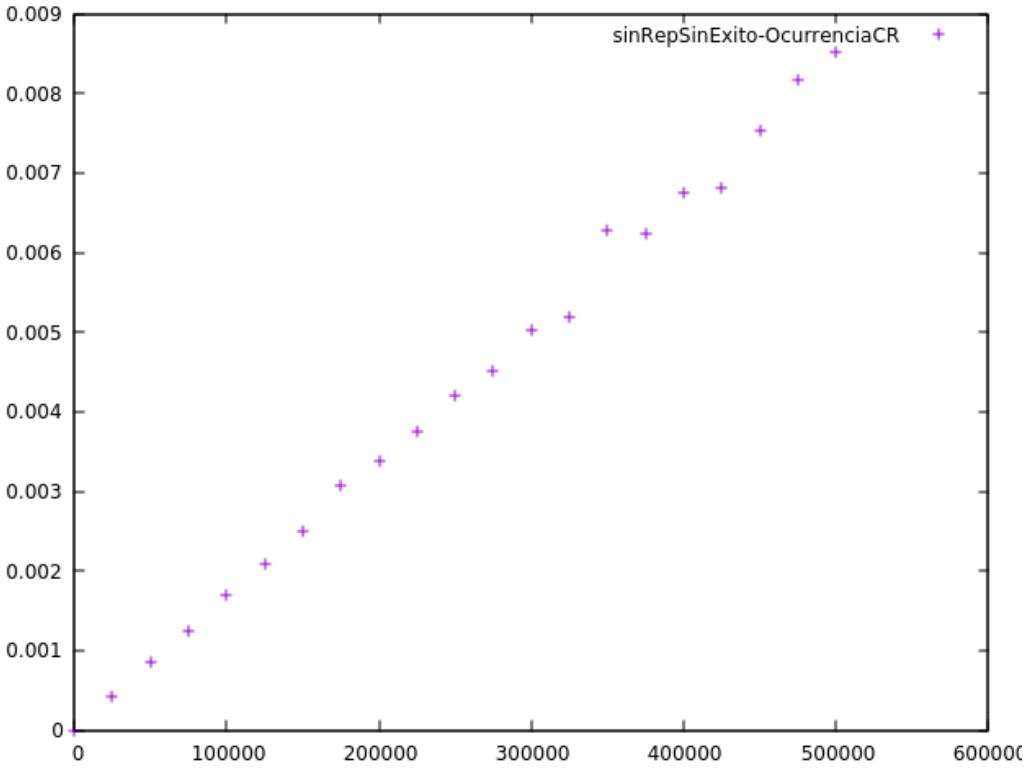
Con elementos repetidos y ocurrencia indeterminada



Sin elementos repetidos y con éxito en ocurrencia

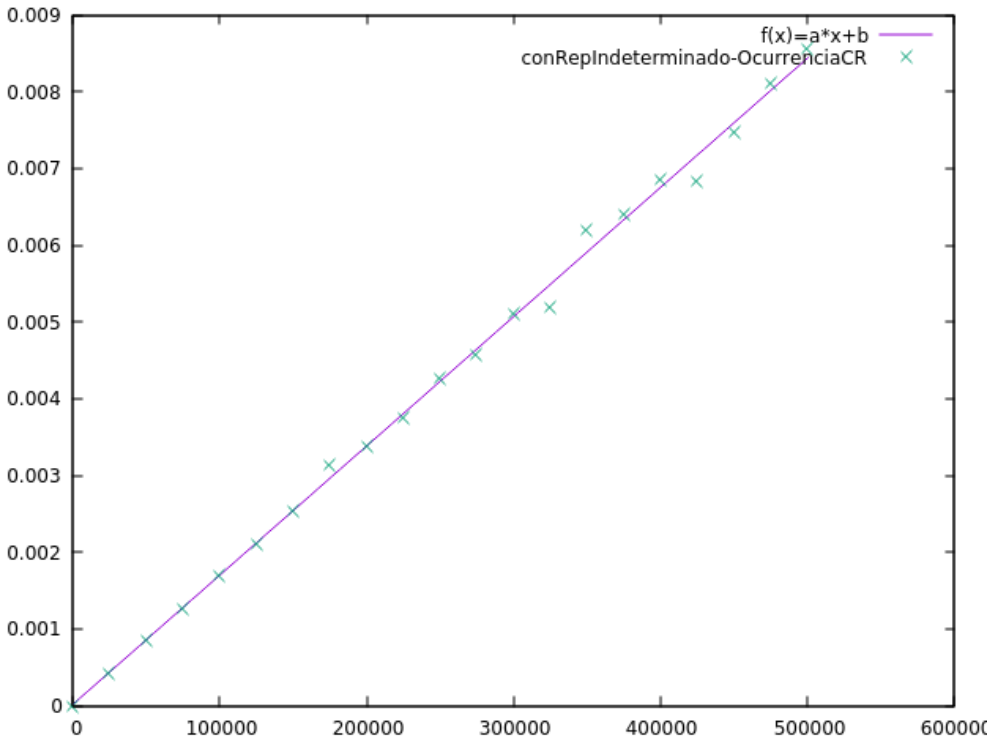


Sin elementos repetidos y sin éxito en ocurrencia

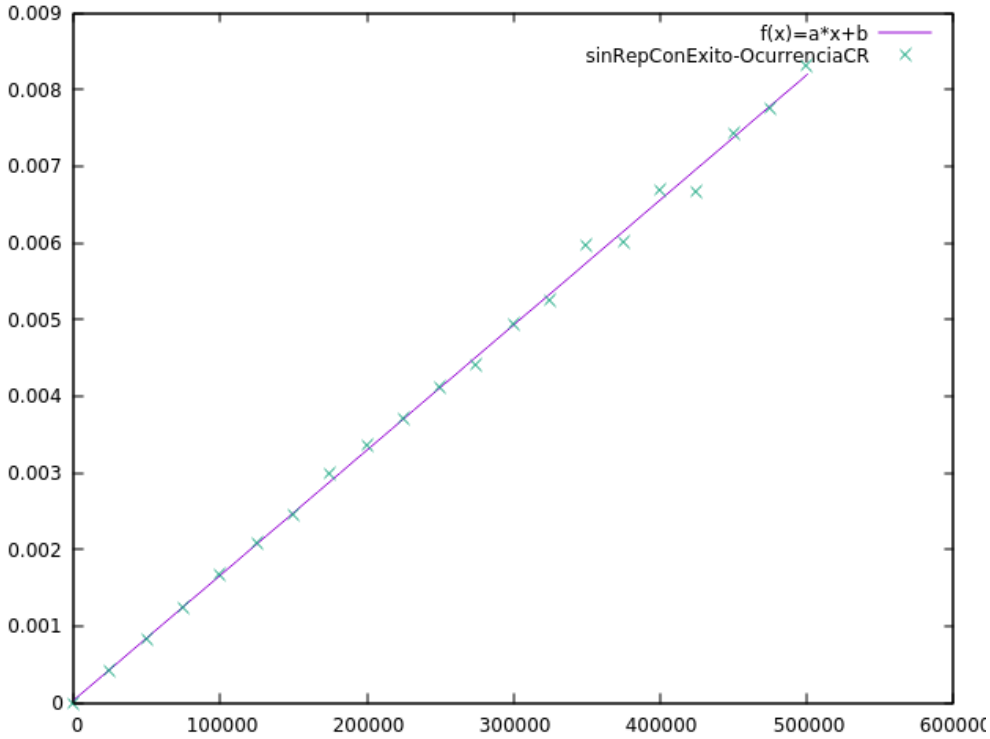


5.2. Eficiencia Híbrida (ajuste lineal)

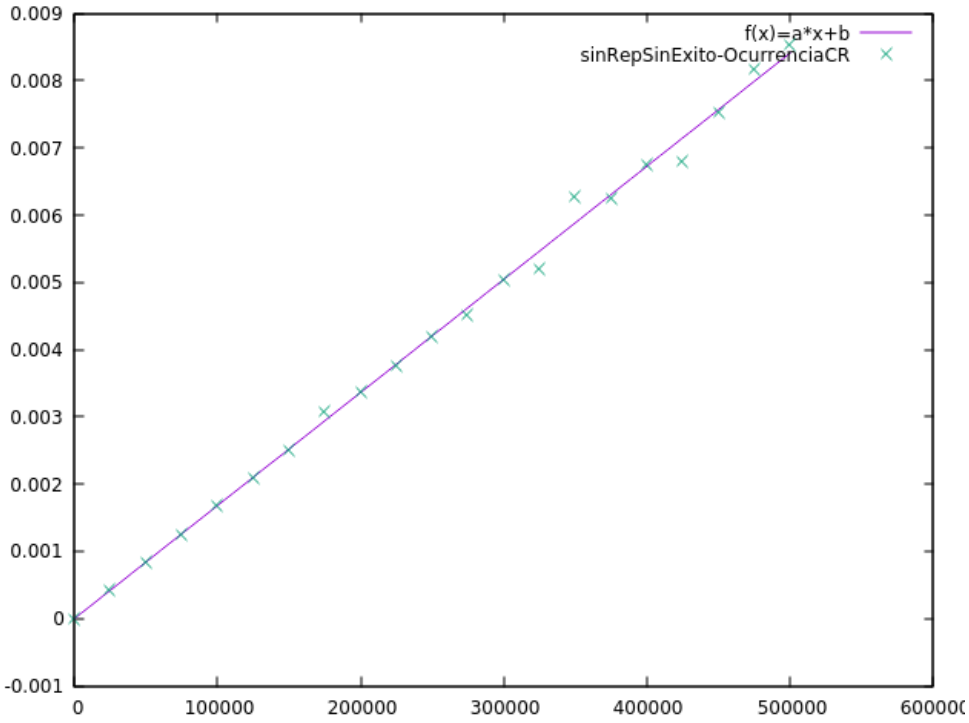
Con elementos repetidos y ocurrencia indeterminada



Sin elementos repetidos y con éxito en ocurrencia



Sin elementos repetidos y sin éxito en ocurrencia



6. Conclusión

Tras realizar la eficiencia empírica e híbrida de los **algoritmos básico y "divide y vencerás con repetidos"**, observando sus gráficas y los tiempos obtenidos hemos llegado a la conclusión de que este es un claro ejemplo en el que **no interesa aplicar la técnica de *divide y vencerás*** ya que los tiempos obtenidos **no mejoran en nada** a los del algoritmo "obvio", y es absurdo molestarse en crear un algoritmo más complejo teniendo uno más simple que resulta **igual de eficiente**.