La sucesión de Fibonacci

Eficiencia empírica

Álvaro Fernández García

Para realizar este estudio se han diseñado tres algoritmos que calculan el término enésimo de la sucesión de Fibonacci.

- El primero de ellos lo calcula de forma recursiva, llamando nuevamente a la función para el término inmediatamente anterior y el anterior a este último. Aquí se presenta el código empleado:

```
int fiborec(int n){
    if(n <= 1)
        return n;
    else
        return (fiborec(n-1) + fiborec(n-2));
}
```

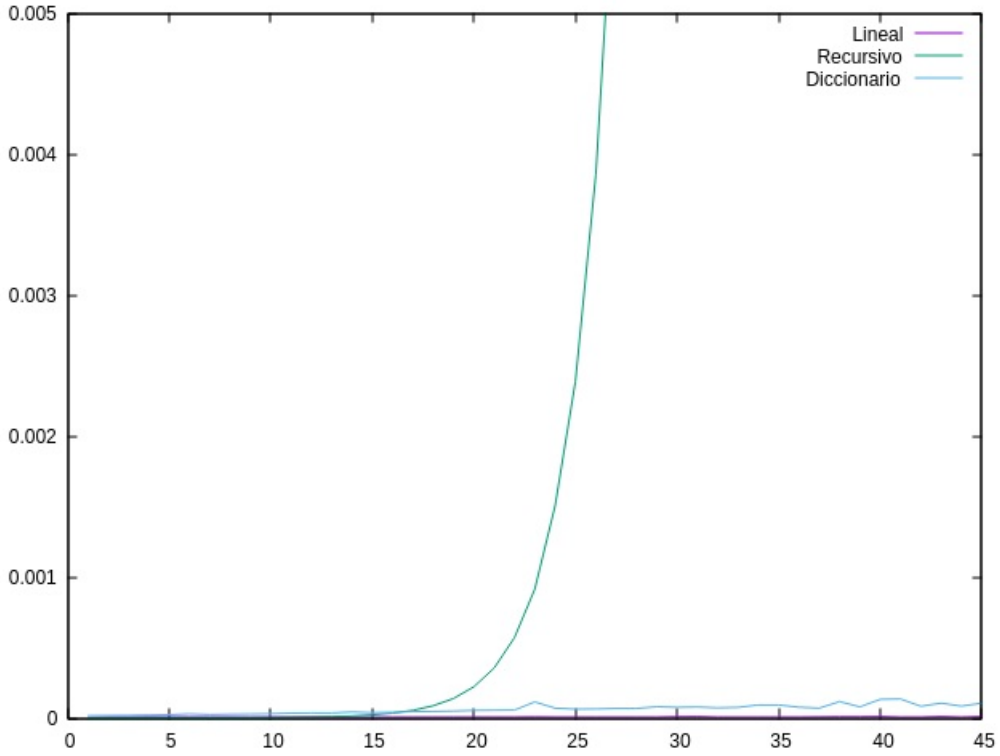
- El segundo, utiliza un map en el que va almacenando los valores de la sucesión ya calculados. Si un valor no está calculado, no hay mas remedio que obtenerlo de forma recursiva. El código es un poco más complejo y es el siguiente:

```
int fibomap_rec(map<int,int> & sol, int k){
    int fib, f1, f2;
    if(k < 2)
        fib = k;
    else{
        auto it1 = sol.find(k-1);
        if(it1 == sol.end())
            f1 = fibomap_rec(sol, k-1);
        else
            f1 = it1->second;

        auto it2 = sol.find(k-2);
        if(it2 == sol.end())
            f2 = fibomap_rec(sol, k-2);
        else
            f2 = it2->second;

        fib = f1 + f2;
    }

    sol[k] = fib;
    return fib;
}

int fibomap(int n){
    map<int,int> soluciones;
    return fibomap_rec(soluciones, n);
}
```

- Por último se ha elaborado una versión del anterior en la que no se utiliza un map para almacenar los resultados, si no un vector, lo que nos permite calcular el término enésimo de forma lineal y además simplifica el código. Tampoco tiene llamadas recursivas.

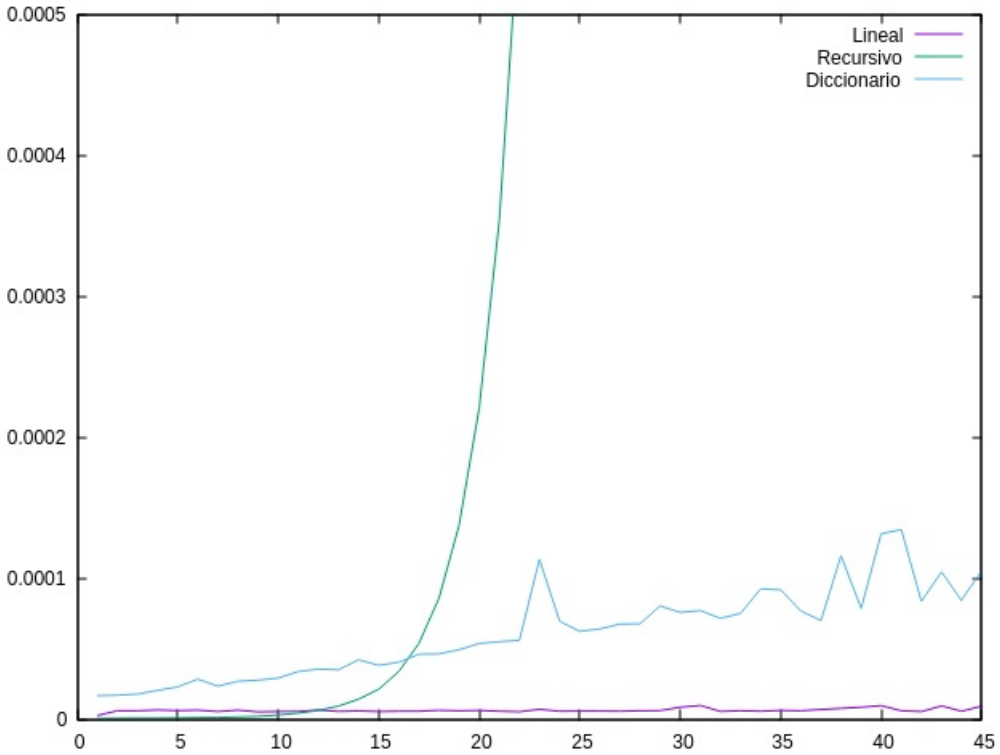
```
int fibolineal(int n){
    int* T = new int[n+1];
    if(n <= 1)
        return n;
    else{
        T[0]=0;
        T[1]=1;
        for(int i = 2; i <= n; ++i)
            T[i]=T[i-1]+T[i-2];
    }
    int res = T[n];
    delete[] T;
    return res;
}
```

Una vez mostrado esto, lo resultados empíricos obtenidos son los siguientes:



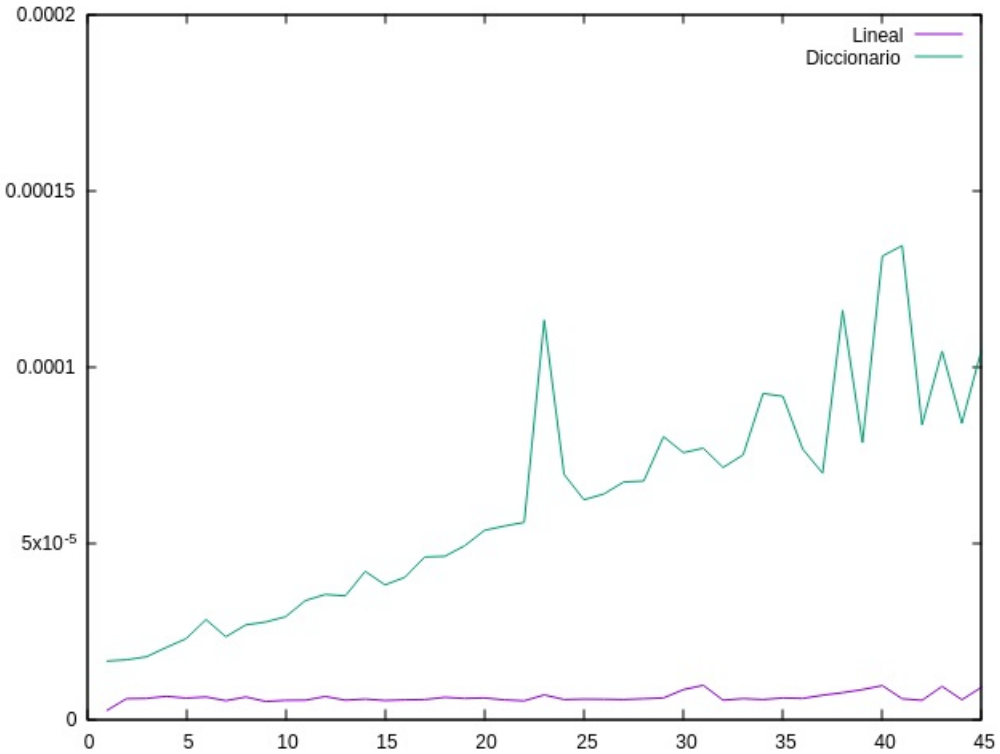
Puede que la versión recursiva sea la más simple y elegante, sin embargo como podemos observar tiene unos pésimos resultados en cuanto a lo que eficiencia se refiere. Esto se debe a la gran cantidad de llamadas recursivas que se realizan. Además la mayoría de ellas se utilizan para calcular valores que ya se habían calculado previamente. Por tanto el simple hecho de almacenar dichos valores hace que la eficiencia mejore brutalmente.

Aquí podemos ver una comparativa más cercana:



A niveles bajos, la recursiva es incluso mejor que la que emplea un map, sin embargo luego lo supera con creces.

Por último se muestra una comparativa entre la versión lineal y el map:



El hecho de que no realice las llamadas recursivas y no utilice métodos para buscar en mapas (menos eficientes que los accesos a un vector), hacen que la version lineal presente unos mejores tiempos.