## Algoritmo de divide y vencerás

<https://www.youtube.com/watch?v=63uo79Swd3Q>

**Divide y Vencerás** es una técnica de diseño de algoritmos que consiste en resolver un problema a partir de la solución de subproblemas del mismo tipo, pero de menor tamaño. Si los subproblemas son todavía relativamente grandes se aplicará de nuevo esta técnica hasta alcanzar subproblemas lo suficientemente pequeños para ser solucionados directamente. Ello naturalmente sugiere el uso de la recursión en las implementaciones de estos algoritmos.

La resolución de un problema mediante esta técnica consta fundamentalmente de los siguientes pasos:

**1.** En primer lugar ha de plantearse el problema de forma que pueda ser descompuesto en k

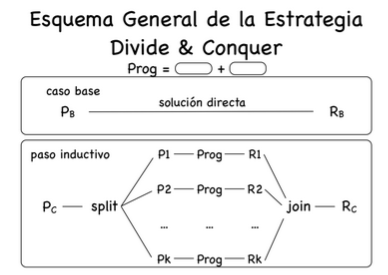
subproblemas del mismo tipo, pero de menor tamaño. Es decir, si el tamaño de la entrada es

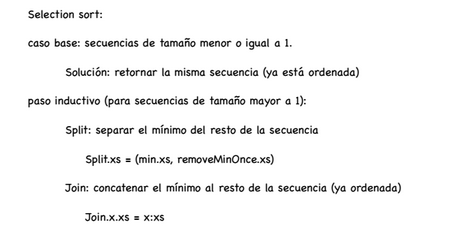
n, hemos de conseguir dividir el problema en k subproblemas (donde 1 ≤ k ≤ n ), cada uno con una entrada de tamaño nk y donde 0 ≤ nk < n. A esta tarea se le conoce como división.

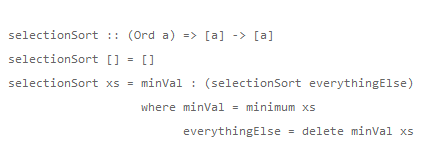
**2.** En segundo lugar han de resolverse independientemente todos los subproblemas, bien directamente si son elementales o bien de forma recursiva.

El hecho de que el tamaño de los subproblemas sea estrictamente menor que el tamaño original del problema nos garantiza la convergencia hacia los casos elementales, también denominados casos base.

**3.** Por último, combinar las soluciones obtenidas en el paso anterior para construir la solución del problema original.







**Los algoritmos diseñados mediante la técnica de Divide y Vencerás van a heredar las ventajas e inconvenientes que la recursión plantea:**

**a)** Por un lado el diseño que se obtiene suele ser simple, claro, robusto y elegante, lo que da lugar a una mayor legibilidad y facilidad de depuración y mantenimiento del código obtenido.

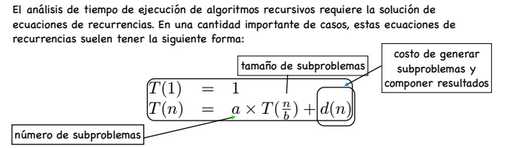
**b)** Sin embargo, los diseños recursivos conlleva normalmente un mayor tiempo de ejecución que los iterativos, además de la complejidad espacial que puede representar el uso de la pila de recursión.

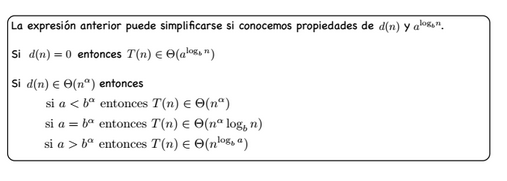
**Otra consideración importante a la hora de diseñar algoritmos Divide y Vencerás es el reparto de la carga entre los subproblemas**, puesto que es importante que la división en subproblemas se haga de la forma más equilibrada posible. En caso contrario nos podemos encontrar con “anomalías de funcionamiento” como le ocurre al algoritmo de ordenación Quicksort. Éste es un representante claro de los algoritmos Divide y Vencerás, y su caso peor aparece cuando existe un desequilibrio total en los subproblemas al descomponer el vector original en dos subvectores de tamaño 0 y n–1. Como vimos en el capítulo anterior, en este caso su orden es O(n^2), frente a la buena complejidad, O(n log n), que consigue cuando descompone el vector en dos vectores de igual tamaño.

**También es interesante tener presente la dificultad y el esfuerzo requerido en cada una de las fases que van a depender del planteamiento del algoritmo concreto.** Por ejemplo, los métodos de ordenación por Mezcla y Quicksort son dos representantes claros de esta técnica pues ambos están diseñados siguiendo el esquema presentado: dividir y combinar.

La división de Quicksort es costosa, pero una vez ordenados los dos subvectores la combinación es inmediata. Sin embargo, la división que realiza el método de ordenación por Mezcla consiste simplemente en considerar la mitad de los elementos, mientras que su proceso de combinación es el que lleva asociado todo el esfuerzo.

En soluciones Divide & Conquer, en general sucede que si la division de problemas en subproblemas es “simple”, entonces el “pegado” de soluciones parciales para conseguir la solucion global es “dificil”. Igualmente, si la division de problemas en subproblemas es “dificil”, el pegado es “simple”.





Las diferencias surgen de los distintos valores que pueden tomar a y b, que en definitiva determinan el número de subproblemas y su tamaño. Lo importante es observar que en todos los casos la complejidad es de orden polinómico o polilogarítmico pero nunca exponencial, frente a los algoritmos recursivos que pueden alcanzar esta complejidad en muchos casos. Esto se debe normalmente a la repetición de los cálculos que se produce al existir solapamiento en los subproblemas en los que se descompone el problema original.

Para aquellos problemas en los que la solución se construya a partir de las soluciones de subproblemas entre los que se produzca necesariamente solapamiento existe otra técnica de diseño más apropiada, y que permite eliminar el problema de la complejidad exponencial

debida a la repetición de cálculos. Estamos hablando de la Programación Dinámica.

