ALGORÍTMOS

El método "**divide y vencerás**" está basado en la resolución recursiva de un problema dividiéndolo en dos o más subproblemas de igual tipo o similar. El proceso continúa hasta que estos llegan a ser lo suficientemente sencillos como para que se resuelvan directamente. Al final, las soluciones a cada uno de los subproblemas se combinan para dar una solución al problema original.

La **programación dinámica** se usa cuando se trata de hallar una solución óptima cuando la subdivisión del problema conduce a:

- Una enorme cantidad de problemas.
- *Problemas cuyas soluciones parciales se solapan.*
- Grupos de problemas de muy distinta complejidad.

Un **algoritmo voraz** es una estrategia de búsqueda por la cual se sigue una heurística consistente en elegir la opción óptima en cada paso local con la esperanza de llegar a una solución general óptima.

- Programación dinámica: objetos no fragmentables y pesos discretos
- Algoritmos voraces: objetos fragmentables

Backtracking se usa cuando se quieren obtener todas las posibles combinaciones para obtener una solución.

Ramificación y poda es una variante del Backtracking mejorado sustancialmente. La técnica de Ramificación y poda se suele interpretar como un árbol de soluciones, donde cada rama nos lleva a una posible solución posterior a la actual. La característica de esta técnica con respecto a otras anteriores (y a la que debe su nombre) es que el algoritmo se encarga de detectar en qué ramificación las soluciones dadas ya no están siendo óptimas, para «podar» esa rama del árbol y no continuar malgastando recursos y procesos en casos que se alejan de la solución óptima.

- ¿Podríamos llegar antes a la solución óptima con otro recorrido?
- ¿Cómo?
  - Adecuando el orden de exploración del árbol de soluciones según nuestros intereses. Se priorizará para su exploración aquellos nodo mas prometedores
- ... sin olvidar las podas

Proceso de poda

- Cota optimista:
  - estima, a mejor, el mejor valor que podría alcanzarse al expandir el nodo
  - puede que no haya ninguna solución factible que alcance ese valor
  - normalmente se obtienen relajando las restricciones del problema
  - si la cota optimista de un nodo es peor que la solución en curso, se puede podar el nodo
- Cota pesimista:
  - estima, a peor, el mejor valor que podría alcanzarse al expandir el nodo
  - ha de asegurar que existe una solución factible con un valor mejor que la cota
  - normalmente se obtienen mediante soluciones voraces del problema
  - se puede eliminar un nodo si su cota optimista es peor que la mejor cota pesimista
  - permite la poda aún antes de haber encontrado una solución factible
- Cuanto mas ajustadas sean las cotas, mas podas se producirán

JERARQUÍAS DE FUNCIONES

$$\underbrace{O(1)}_{\text{constantes}} \subset \underbrace{O(\log\log n)}_{\text{sublogarítmicas}} \subset \underbrace{O(\log n) \subset O(log^{a(>1)}n)}_{\text{logarítmicas}} \subset \underbrace{O(\sqrt{n})}_{\text{sublineales}} \subset \underbrace{O(n)}_{\text{lineales}} \subset \underbrace{O(n\log n)}_{\text{lineal-logarítmicas}}$$

$$\subset \underbrace{O(n^2) \subset O(n^{a(>2)})}_{\text{polinómicas}} \subset \underbrace{O(2^n) \subset O(a(>2)^n)}_{\text{exponenciales}} \subset \underbrace{O(n!) \subset O(n^n)}_{\text{superexponenciales}}$$

PROBLEMA DE LA MOCHILA

Mochila Discreta: Solución óptima con Divide y Vencerás aunque con un coste exponencial que lo hace inviable; por ello se utiliza Programación Dinámica (refinamiento de Divide y vencerás), pero sólo se puede aplicar si los pesos son cantidades discretas. En este caso Programación Dinámica garantiza la solución óptima.

Si los pesos no son discretos (seguimos con mochila discreta) entonces la única técnica que queda es backtraking o ramificación y poda, que garantizan la solución óptima aunque con una complejidad exponencial, pero siempre se puede mitigar con buenos mecanismos de poda.

→ Sin fraccionamiento

Mochila Continua: Solución óptima voraz siempre que se siga el criterio del valor específico. Si no se sigue ese criterio el voraz no garantiza la óptima.

→ Con fraccionamiento

INFORMACIÓN EXTRAÍDA DE LOS TESTS

$f \in \theta(g_1)$ y $f \in \theta(g_2)$

- $f \in \theta(g_1 + g_2)$
- $f \in \theta(\max(g_1, g_2))$
- $f^2 \in \theta(g_1 * g_2)$

$f(n)=2f(n/2)+1 \in \theta(n)$

$f(n)=2f(n/2)+n \in \theta(n \log n)$

$f(n)=2f(n-1)+n \in \Omega(2^n)$

$T(n)=T(n/2)+1 \in O(\log n)$

$T(n)=T(n/2)+n \in O(n \log n)$

$T(n)=T(n-1)+x \in O(n^2)$

En Quicksort, si el vector ya está ordenado y utiliza como pivote:

- El primer elemento → Se comporta peor.
- La mediana → No influye en la complejidad temporal.
- El elemento central → Se comporta mejor.