ALGORITMOS DE ORDENACIÓN

Algoritmo	Complejidad temporal			Complejidad espacial
	Mejor	Promedio	Peor	Peor
Quicksort	O(n log(n))	O(n log(n))	O(n^2)	O(log(n))
Mergesort	O(n log(n))	O(n log(n))	O(n log(n))	O(n)
Bubble Sort	O(n)	O(n^2)	O(n^2)	O(1)
Insertion Sort	O(n)	O(n^2)	O(n^2)	O(1)
Selection Sort	O(n^2)	O(n^2)	O(n^2)	O(1)

QuickSort pivote mediana = no influye que esté ordenado en la complejidad temporal
QuickSort pivote central = mejor cuando está ordenado
QuickSort pivote primero = peor cuando está ordenado

MergeSort y QuickSort aplican **Divide y Vencerás**

Si se modifica **MergeSort** para dividir en **tres** partes, posteriormente se combinan las soluciones parciales la nueva **complejidad temporal** es: **n log(n)**. Y además, la complejidad de la **combinación** de las soluciones parciales sería **Theta(n)**

La complejidad **temporal asintótica** del **MergeSort** de **dividir** el problema en dos subproblemas es O(1)

La complejidad **temporal asintótica** del **QuickSort** de **dividir** el problema en dos subproblemas es O(n)

MergeSort tiene menor complejidad que Inserción Directa y que Burbuja.

Los algoritmos de ordenación **directos** tienen mayor complejidad temporal y ejecuciones mayores que los indirectos.

El mergeSort es **asintóticamente** más rápido o igual que el QuickSort

RAMIFICACIÓN Y PODA

Cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima.

El orden escogido para priorizar los nodos en la lista de nodos vivos puede influir en el número de nodos que se descartan sin llegar a expandirlos.

La cota pesimista puede servir para actualizar el valor de la mejor solución hasta el momento.

El uso de funciones cota puede reducir el número de instancias del problema que pertenecen al caso peor.

La complejidad en el mejor de los casos puede ser polinómica con el número de decisiones a tomar.

En general el valor de una cota pesimista es **mayor** que el valor de una cota optimista si se trata de un problema de **minimización**, aunque en ocasiones ambos valores pueden coincidir.

En general el valor de una cota pesimista es **menor** que el valor de una cota optimista si se trata de un problema de **maximización**, aunque en ocasiones ambos valores pueden coincidir.

La diferencia principal entre vuelta atrás y ramificación y poda en el problema de la mochila es el orden de exploración de las soluciones

VUELTA ATRÁS

Es condición necesaria (aunque no suficiente) que el dominio de las decisiones sea discreto o discretizable y que el número de decisiones a tomar esté acotado.

Los mecanismos de poda basados en la mejor solución hasta el momento pueden eliminar vectores que representan posibles soluciones factibles.

Los mecanismos de poda basados en la mejor solución hasta el momento pueden eliminar soluciones parciales que son factibles

En ausencia de cotas optimistas y pesimistas, no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.

El orden en el que se van asignando los distintos valores a las componentes del vector que contendrá la solución:

- Es irrelevante si no se utilizan mecanismos de poda basados en la mejor solución hasta el momento.
- Puede ser relevante si se utilizan mecanismos de poda basados en estimaciones optimistas.

Al convertir un algoritmo vuelta atrás en ramificación y poda, podemos aprovechar mejor las cotas optimistas.

Puede resolver el problema de la mochila discreta más rápido si se prueba primero a meter cada objeto antes de no meterlo, pero sólo si se usan cotas optimistas para podar el árbol.

Si hacen uso de cotas optimistas generan las soluciones posibles al problema mediante un recorrido en profundidad del árbol que representa el espacio de soluciones.

En el peor caso la complejidad temporal de la solución de la mochila discreta es exponencial.

Puede ocurrir que el uso de las cotas pesimistas y optimistas sea inútil, incluso perjudicial, puesto que es posible que a pesar de utilizar dichas cotas no se descarte ningún nodo.

DIVIDE Y VENCERÁS

Un algoritmo recursivo alcanza su máxima eficiencia cuando el problema de tamaño n se divide en a problemas de tamaño n/a.

No garantiza la existencia de una solución de complejidad temporal polinómica.

Será más eficiente cuanto más equitativa sea la división en subproblemas.

Es el esquema con la mejor solución para el problema de la mochila continua.

VORAZ

La eficiencia se basa en el hecho de que las decisiones tomadas no se reconsideran.

Puede ser la única alternativa si el dominio de las decisiones es un conjunto infinito.

Cuando se usa para abordar un problema de optimización por selección discreta (encontrar un subconjunto del conjunto de elementos):

- Puede que no encuentre solución.
- Puede que la solución no sea óptima.
- Imposible que reconsidere la decisión tomada anteriormente.

El valor que obtiene para el problema de la mochila discreta es una cota inferior par ale valor óptimo que a veces puede ser igual a este.

Garantiza la solución óptima SÓLO para determinados problemas.

Hay problemas para los cuales se puede obtener siempre la solución óptima.

Hay problemas para los cuales solo obtenía la solución óptima para algunas instancias y un subóptimo para muchas instancias.

Se basan en la idea de que la solución óptima se puede construir añadiendo repetidamente le mejor elemento disponible.

PROGRAMACIÓN DINÁMICA

Hace uso de que se puede ahorrar esfuerzo guardando los resultados de esfuerzos anteriores.

Solución alternativa a una solución recursiva ingenua que por un lado se basa en obtener soluciones óptimas a problemas parciales más pequeños y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo.

Esquema más apropiado cuando la descomposición de un problema da lugar a subproblemas de tamaño similar al original, muchos de los cuales se repiten.

Para valores continuos la programación dinámica recursiva puede resultar mucho más eficiente que la programación dinámica iterativa en cuanto al uso de memoria.

La mejora de la programación dinámica frente a la solución ingenua es que en la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos.

EFICIENCIA DE PROBLEMAS

Verde = SÍ - Rojo = NO

Solución voraz trivial y eficiente

- **Mochila continua (Pero es mejor implementarla en divide y vencerás)**
- **Mochila discreta sin limitación en la carga máxima**
- **Árbol de recubrimiento mínimo para un grado no dirigido con pesos**
- **Mochila continua o con fraccionamiento**
- **Árbol de cobertura de coste mínimo de un grado conexo**
- **El problema del cambio**
- **Mochila discreta (a secas)**
- **El problema de la asignación de tareas de coste mínimo en una matriz**

Solución dinámica eficiente

- **El problema del cambio (eficiente)**
- **n-ésimo elemento de la serie de Fibonacci (eficiente)**
- **Mochila discreta (a secas)**
- **Cortar un tubo de longitud n en segmentos de longitud entera entre 1 y n**
- **La mochila discreta sin restricciones adicionales**
- **El problema de la asignación de tareas**
- **Torres de Hanoi**

Ramificación y poda eficiente

- **Encontrar subconjunto de tamaño m de suma mínima en un conjunto de n enteros positivos**
- **Problema del viajante comercio: listar todas las soluciones factibles**

Vuelta atrás eficiente

- **Obtener todas las permutaciones de una lista compuesta por n elementos**