Tema 10: Programación Dinámica

Definición
Método de diseño de algoritmos que se puede usar cuando la solución del problema se puede ver como resultado de una sucesión de decisiones.
Se aplica en **problemas de optimización** y secuencias de ADN.

Cuando dividimos en subproblemas de tamaño $n - 1$ no se utiliza Divide y Vencerás, sino Programación Dinámica. En particular, se aplica a la resolución de problemas n-etápicos usando ecuaciones de recurrencia.

Se aplica en 4 fases:

- **Naturaleza n-etápica**: la función F_N se obtiene de la decisión óptima de F_{N-1}
- **Verificación del POB**: cualquier subsecuencia de decisiones de una secuencia óptima de decisiones que resuelve un problema también debe ser óptima respecto al subproblema que resuelve.
    - Una política es óptima si en un periodo dado, cualesquiera que sean las decisiones precedentes, las decisiones que quedan por tomar constituyen una política óptima teniendo en cuenta los resultados de las decisiones precedentes.
    - Una política óptima sólo puede estar formada por subpolíticas óptimas
- **Planteamiento de una recurrencia**: que represente la forma de ir logrando etapa por etapa la solución optimal
- **Cálculo de la solución**: resolviendo problemas encajados para construir la solución
    - Se puede construir desde el primer estado al último (enfoque adelantado)
    - Desde el último estado al primero (enfoque atrasado)

	P.DINÁMICA	DYV	GREEDY
División del problema	Subproblemas que comparten subproblemas (solapados)	Subproblemas del mismo tipo pero más sencillos (independientes)	Subproblema resultante de la última solución encontrada
Tamaño subproblemas	Se diferencian en una unidad	No se sabe su tamaño	No se sabe su tamaño
Resolución	Se generan resultados a partir de los resultados de anteriores subproblemas	Combinación de las soluciones de los subproblemas (recursivamente)	Se elige la solución óptima para el subproblema resultante
Solución	Solución optimal	Soluciones óptimas	No necesariamente óptima

Los algoritmos PD suelen tener eficiencia polinomial

Tema 11: Algoritmos basados en PD

Problema de la Mochila

Se puede ver como el resultado de una **sucesión de decisiones**: los valores x_i $1 \leq i \leq n$.

Notamos $M(1, j, Y)$ al problema siguiente:

- Maximizar $\sum_{i=1}^{j} p_i x_i$
- Limitando $\sum_{i=1}^{j} w_i x_i \leq Y$

Sea $y_1, y_2, \ldots, y_n$ una sucesión optimal de $M(1, n, M)$.

- Si $y_1 = 0$ (es decir, el primer elemento no se coge), entonces $y_2 \ldots, y_n$ es una sucesión optimal para el problema $M(2, n, M)$. Si no lo es, entonces $y_1, \ldots, y_n$ no es solución optimal de $M(1, n, M)$
- Si $y_1 = 1$ (es decir, se coge el primer elemento completo), entonces $y_2, \ldots, y_n$ es una sucesión optimal para $M(2, n, M - w_1)$. Si no lo fuera, habría otra sucesión $z_2, \ldots, z_n$ tal que $\sum_{i=2}^{n} w_i z_i \leq M - w_1$ y $\sum_{i=2}^{n} p_i z_i > \sum_{i=2}^{n} p_i y_i$. Luego, la sucesión $y_1, z_2, \ldots, z_n$ es una sucesión para el problema de partida con mayor valor

Ya hemos visto que se verifica el principio de optimalidad de Bellman. Veamos ahora que se puede **construir una ecuación recurrente** que resuelva el problema.

Sea $f_j(y)$ el valor de una solución optimal de $M(j + 1, n, y)$, entonces $f_o(M)$ es el valor de una solución optimal para $M(1, n, M)$.

A partir del POB se sigue que: $f_0(M) = Max\{f_1(M), f_1(M - w_1) + p_1\}$ si suponemos que las posibles soluciones de x_1 son 0 ó 1.

Un enfoque de **solución** para el problema: una solución sería la sucesión $x_1, \ldots, x_n$ donde tenemos que decidir para cada i si x_i es 0 ó 1. Supongamos que las decisiones se toman desde n hasta 1.

Tras una decisión sobre x_n puede ocurrir que:

- sea 0, entonces no ha habido incremento de beneficio
- sea 1, entonces la capacidad restante es $M - w_n$ y aumenta el beneficio p_n

Entonces $f_i(M) = Max\{f_{i-1}(M), f_{i-1}(M - w_i) + p_i\}$ es una solución optimal a $M(1, i, M)$, y podemos calcular sucesivamente $f_1, f_2, \ldots, f_n$

Problema del Camino Mínimo (Floyd)

Comprobemos el **POB**:

Sea $i, i_1, i_2,..., i_k, j$ el camino mínimo desde i hasta j. Desde i se ha decidido continuar por i_1 para obtener la solución óptima. Luego, $i_1,..., j$ debe constituir un camino mínimo entre i_1 y j. En caso contrario, $i_1, r_1,..., r_q, j$ sería un camino entre i y j más corto.

Construyamos la **ecuación recurrente**:

$D_k(i,j) = Min\{D_{k-1}(i,j), D_{k-1}(i,k) + D_{k-1}(k,j)\}$ donde $k \in A_i$ y A_i es el conjunto de los vértices adyacentes a i

Algoritmo de Floyd

Resuelve el problema del Camino Mínimo para un grafo donde entre cada par de nodos puede haber una distancia 0 (de un nodo a sí mismo), distancia mayor que 0 (distancia a otro nodo conectado) o distancia infinita (si no existe conexión entre los dos nodos).

La forma de **aplicar el POB** es suponiendo que k es un nodo en el camino mínimo que une i y j, entonces el camino de i hasta k y el camino de k hasta j son optimales.

Para **realizar el algoritmo** se crea una matriz D con la longitud del camino mínimo entre cada par de nodos. Tras la iteración k, D nos da la longitud de los caminos mínimos usando los nodos $\{1, 2,..., k\}$ y después de n iteraciones tendremos la solución optimal.
Por tanto, en la iteración k se comprueba para cada par de nodos si existe o no un camino que pase a través de k que sea mejor que el que estaba.

El tiempo del algoritmo es $O(n^3)$. También se puede usar el algoritmo Dijkstra eligiendo un nodo raíz cada vez, el tiempo es el mismo teóricamente pero en la práctica es más corto.

Viajante de Comercio

Dado un grafo con distancias entre los vértices, buscamos el circuito más corto que empiece y termine en el mismo nodo (pasa por cad vértice una sola vez).
Los valores de las distancias son los mismos que en algoritmo Floyd (0,L,infinito).

Por el POB vemos que: $g(i,S) = Min_{j \in S}[L_{ij} + g(j, S - \{j\})]$

//ni lo entiendo ni tengo ganas de entenderlo :(

Tema 12: Aplicaciones

Multiplicación encadenada de matrices

Dadas las matrices A_1,..., A_n con A_i de dimensión $d_{i-1} \times d_i$. Queremos determinar el orden de multiplicación para minimizar el número de multiplicaciones escalares.

- Para multiplicar una matriz $p \times q$ por otra $q \times r$ se necesitan pqr multiplicaciones escalares
- El producto de matrices está parentizado si está constituido por una matriz o por la multiplicación de dos matrices

Sea $m[i,j]$ el mínimo número de operaciones escalares para $A_i ... A_j$, entonces buscamos la solución $M[1,n]$.

Por ejemplo, $min[1,6] = min\ de$ $\quad m[1,1] + m[2,6] + d_0 d_1 d_6$

$m[1,2] + m[3,6] + d_0 d_2 d_6$

$\dots$

$m[1,5] + m[6,6] + d_0 d_5 d_6$

La fórmula recursiva será: $m[i,j] = \begin{cases} 0\ si\ i = j \\ min\{m[i,k] + m[k+1,j] + p_{i-1} p_k p_j\}\ si\ i \leq j \end{cases}$

//de repente me saca no se que de PD yo que se paso

El problema del play-off

Supongamos que dos equipos A y B juegan una final en la que se quiere saber cual será el primero en ganar n partidos.

Sea $P(i,j)$ la probabilidad de que sea A el ganado si A necesita i partidos para ganar y B necesita j.

Antes del primer partido, la probabilidad de que gane A es $P(n,n)$.