PROGRAMACIÓN DINÁMICA [10 puntos] El próximo día 5 de diciembre se va a disputar el partido Chelsea vs Atlético de Madrid correspondiente a la jornada 6 de la Champions League. Los organizadores de dicho partido han concedido al equipo español un cupo de E entradas. El Atlético de Madrid nos ha encomendado gestionar sus entradas entre sus peñas. Se sabe que cada peña del equipo, hay P, ha solicitado un número de entradas, siendo este número S_k con $1 \leq k \leq P$, donde S_k indica las entradas solicitadas por la peña k. El número total de solicitudes que llegaron al club excede con creces las E entradas de las que se dispone, por lo que se ha tomado la siguiente decisión: se concederán las entradas solicitadas por una peña siempre y cuando todos sus solicitantes tengan cabida en el estadio. Teniendo en cuenta que el objetivo del Atlético de Madrid es maximizar la venta de sus E entradas, se trata de resolver el problema mediante Programación Dinámica indicando a qué peña se ha de vender entradas con objeto de vender el mayor número de ellas.

Se pide responder con claridad y concisión a las siguientes cuestiones:

- Secuencia de decisiones: tamaño y significado de la decisión i-ésima (10%)
- Función Objetivo (10%)
- Restricciones (10%)
- Demostración del principio de optimalidad (15%)
- Función recursiva y cómo se realiza la primera llamada a la función (25%)
- Árbol de llamadas completo para el siguiente ejemplo (10%)

$$E = 10, P = 4, S[1..4] = \{ 3, 5, 6, 3 \}$$

- Estructuras de almacenamiento necesarias y rellenado de las mismas para el ejemplo anterior mostrando únicamente los cálculos necesarios para resolver los subproblemas que aparecen en el árbol del apartado anterior. Concluir con el valor que optimiza la función objetivo, así como la forma de alcanzarlo (20%)

Antes de mostrar la solución del ejercicio conviene indicar que es igual que el problema de la mochila visto en clase:

- Disponemos de P peñas y se trata de decidir a qué peñas se venderán las entradas. Del mismo modo que disponíamos de N objetos y se trataba de decidir qué objetos se introducían en la mochila. La secuencia de decisiones será de tamaño P (tantas decisiones como peñas) y la decisión con respecto a cada peña consiste en indicar si a dicha peña se le dan las entradas solicitadas o no => Decisiones.
- No todas las peñas recibirán entradas ya que no hay suficientes entradas para todas, del mismo modo que no todos los objetos se podían introducir en la mochila ya que no todos cabían en ella => Restricciones.
- Se trata de vender/distribuir el mayor número de entradas igual que en el problema de la mochila se trataba de obtener el máximo beneficio => Función objetivo.

SOLUCIÓN.-

Secuencia de decisiones: tamaño y significado de la decisión i-ésima (10%)

Número fijo de decisiones = P

$<d_1, d_2, \ldots, d_P>$

d_i representa si se vende entradas a la peña i o no, representaremos con 0 el hecho de que no y 1 el hecho de que sí.

Función Objetivo (10%)

$$\text{maximizar} \sum_{i=1}^{P} S[i] * di$$

Restricciones (10%)

$$\sum_{i=1}^{P} S[i] * d_i \leq E$$

Demostración del principio de optimalidad (15%)

Sea < d_1, d_2, …, d_P > la solución óptima del problema de distribuir E entradas entre P peñas (de la 1 a la P) obteniendo la mayor venta. Denominaremos a dicho problema Venta(P,E). El valor asociado a dicha secuencia de decisiones es

$$\sum_{i=1}^{P} S[i] * di$$

cumpliéndose además que

$$\sum_{i=1}^{P} S[i] * d_i \leq E$$

Supongamos que prescindimos de la última decisión d_P. La cual indica si a la peña P se le venden entradas o no. La subsecuencia que queda, esto es, < d_1, d_2, …, d_{P-1} > es la solución óptima para el problema asociado, es decir, para el subproblema

Venta(P-1, E - S[P]*d_P),

que es el subproblema de distribuir E - S[P]*d_P entradas entre P-1 peñas (de la 1 a la P-1) obteniendo la mayor venta.

El valor asociado a dicha secuencia de decisiones es

$$\sum_{i=1}^{P-1} S[i] * di$$

cumpliéndose además que

$$\sum_{i=1}^{P-1} S[i] * d_i \leq E - S[P] * d_P$$

Supongamos que < d_1, d_2, …, d_{P-1} > **NO** es la solución óptima para el problema asociado sino que existe otra solución < d^*_1, d^*_2, …, d^*_{P-1} > que mejora su valor. Esto querrá decir que

$$\sum_{i=1}^{P-1} S[i] * di \; < \; \sum_{i=1}^{P-1} S[i] * d_i^* \qquad (1)$$

cumpliéndose además que

$$\sum_{i=1}^{P-1} S[i] * d_i^* \leq E - S[P] * d_P$$

Pero entonces sumando S[P]*d_P a ambos lados de la desigualdad recogida en (1), se cumpliría que:

$$\sum_{i=1}^{P-1} S[i] * di + S[P] * d_P < \sum_{i=1}^{P-1} S[i] * d_i^* + S[P] * d_P$$

lo que significa que la secuencia < d^*_1, d^*_2, …, d^*_{P-1}, d_P > mejoraría el resultado de la secuencia <d_1, d_2, …, d_P > para el problema Venta(P,E), lo cual contradice la hipótesis de partida, pues < d_1, d_2, …, d_P > era la solución óptima de dicho problema. En conclusión, se cumple el principio de optimalidad.

Ecuación recursiva y primera llamada a la función (25%)

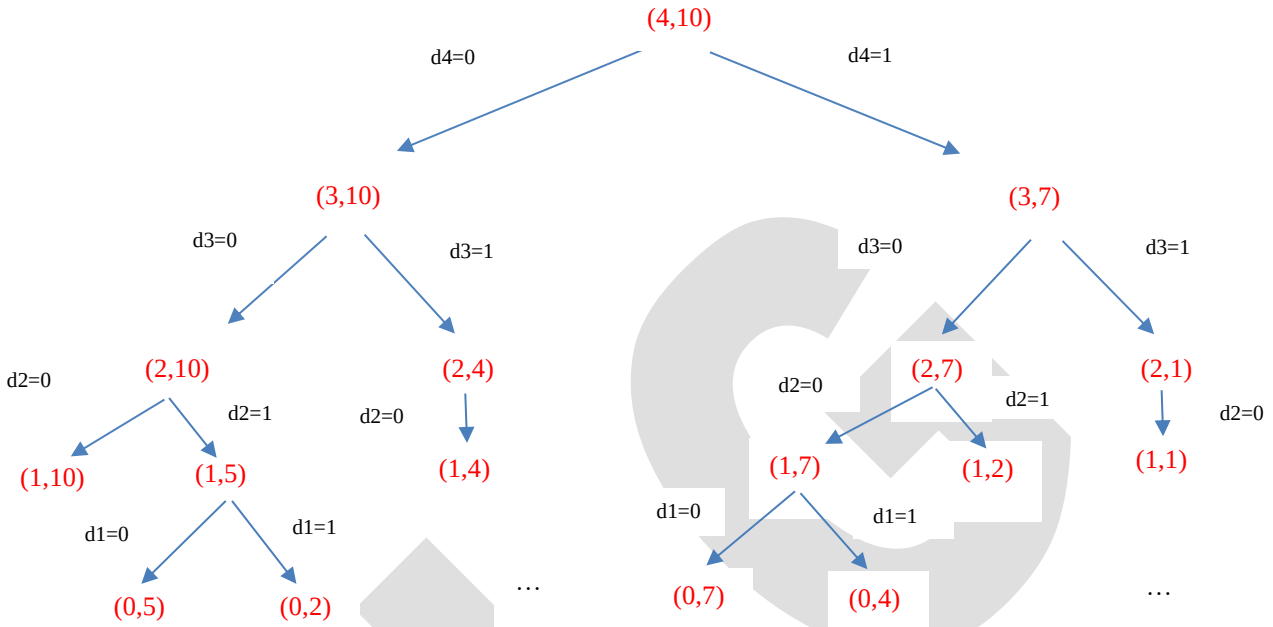
$$Venta\ (p,e) = \begin{cases} 0 & si\ p = 0 \\ Venta(p-1,e) & si\ p > 0\ y\ e < S[p] \\ \max_{d_p \in \{0,1\}} \{Venta(p-1, e - d_p * S[p]) + S[p] * d_p\} & si\ p > 0\ y\ e \geq S[p] \end{cases}$$

donde Venta(p,e) devuelve el máximo número de entradas que se pueden distribuir entre las p peñas (del 1 al p) de las e entradas posibles.

La primera llamada a la función se producirá como Venta(P,E), siendo P el número de peñas totales y E el número de entradas disponibles al comienzo.

Árbol de llamadas para el ejemplo dado (10%)

E = 10, P = 4, S[1..4] = { 3, 5, 6, 3 }



Nota.- Los enunciados facilitados a los alumnos se diferenciaban en que tenían ejemplos diferentes. Esta solución resuelve sólo uno de ellos, el otro era muy similar.

Estructuras de almacenamiento necesarias y rellenado de las mismas para el ejemplo anterior, mostrando únicamente los cálculos necesarios para resolver los subproblemas que aparecen en el árbol. Concluir con el valor que optimiza la función objetivo, así como la forma de alcanzarlo (20%)

Dado que nuestra función recursiva tiene dos parámetros, se precisan dos estructuras bidimensionales. Estas tendrán P+1 filas y E+1 columnas, esto es debido a que $0 \leq p \leq P$ y $0 \leq e \leq E$. En una de las estructuras, Vmax, se guardará el valor asociado a cada subproblema, esto es, el máximo número de entradas distribuidas. En la otra estructura, Dec, se almacenará la alternativa (0 o 1) que proporcione el máximo para cada subproblema.

Inicializar las estructuras de almacenamiento con los resultados de los problemas triviales, corresponde a rellenar la fila 0 de cada matriz con 0, dado que los problema triviales son del tipo Venta(0,e), esto es, Vmax[0][e]=0 con $0 \leq x \leq 10$ y Dec[0][e]=0 con $0 \leq x \leq 10$

Completar el rellenado de la matrices: dado que para solucionar el subproblema Venta(p,e) se precisa conocer la solución de los subproblemas del tipo Venta(p-1,x) donde $x \leq e$, ambas matrices se rellenan por filas en sentido creciente, desde la fila 1 hasta la fila P. (=> Dependencia de los subproblemas).

Vmax[1][1]=Vmax[0][1]=0 Dec[1][1]=0
Vmax[1][2]=Vmax[0][2]=0 Dec[1][2]=0
Vmax[1][4]=max { Vmax[0][4], Vmax[0][1]+3 }=max{0, 0+3}=3 Dec[1][4]=1
Vmax[1][5]=max { Vmax[0][5], Vmax[0][2]+3 }=max{0, 0+3}=3 Dec[1][5]=1
Vmax[1][7]=max { Vmax[0][7], Vmax[0][4]+3 }=max{0, 0+3}=3 Dec[1][7]=1
Vmax[1][10]=max { Vmax[0][10], Vmax[0][7]+3 }=max{0, 0+3}=3 Dec[1][10]=1

Vmax[2][1]=Vmax[1][1]=0 Dec[2][1]=0
Vmax[2][4]=Vmax[1][4]=3 Dec[2][4]=0
Vmax[2][7]=max{Vmax[1][7],Vmax[1][2]+5}=max{3,0+5}=5 Dec[2][7]=1
Vmax[2][10]= max{Vmax[1][10],Vmax[1][5]+5}=max{3,3+5}=8 Dec[2][10]=1

Vmax[3][7]= max{Vmax[2][7],Vmax[2][1]+6}=max{5,0+6}=6 Dec[3][7]=1
Vmax[3][10]= max{Vmax[2][10],Vmax[2][4]+6}=max{8,3+6}=9 Dec[3][10]=1

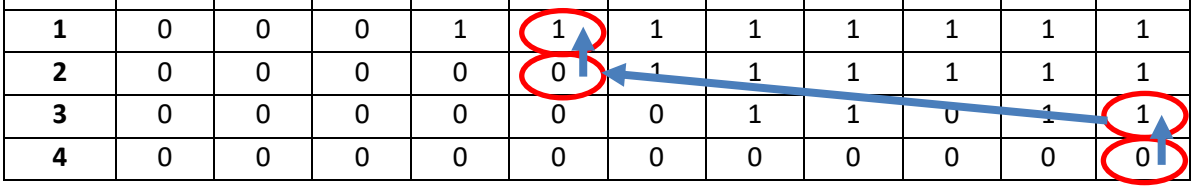
Vmax[4][10]= max{Vmax[3][10],Vmax[3][7]+3}=max{9,6+3}=9 Dec[4][10]=0

Las matrices Vmax y Dec tras el proceso de rellenado quedarían del siguiente modo:

Vmax	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	3	3	3	3	3	3	3	3
2	0	0	0	3	3	5	5	5	8	8	8
3	0	0	0	3	3	5	6	6	8	9	9
4	0	0	0	3	3	5	6	6	8	9	9

El mayor número de entradas distribuido estará en la posición [P][E] de la matriz Vmax, esto es, 9.

Dec	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1	1
2	0	0	0	0	0	1	1	1	1	1	1
3	0	0	0	0	0	0	1	1	0	1	1
4	0	0	0	0	0	0	0	0	0	0	0



La secuencia óptima de decisiones se obtiene recorriendo determinadas posiciones de la matriz Dec. Comenzaríamos por la posición [4][10], el valor ahí almacenado correspondería a d_4, es decir, d_4 será 0. Seguidamente iríamos a Dec[4-1][10-d_4*S[4]]=Dec[3][10], ahí se encontrará el valor correspondiente a d_3, d_3 será 1 por tanto. Seguidamente iríamos a Dec[3-1][10-d_3*S[3]]=Dec[2][4], ahí se encontrará el valor correspondiente a d_2, d_2 será 0 por tanto. Por último, iríamos a Dec[2-1][4-d_2*S[2]]=Dec[1][4], ahí se encontrará el valor correspondiente a d_1, d_1 será 1. De tal modo que la secuencia de decisiones óptima corresponde a: $< d_1, d_2, d_3, d_4 > = < 1, 0, 1, 0 >$