ISIS 1105 DISEÑO Y ANÁLISIS DE ALGORITMOS
PROYECTO
Jorge Andrés de los Rios (201011164)
Laura Camila Mojica López (201212513)

PROBLEMA A

El problema consiste en encontrar un subconjunto del conjunto de canciones $d_1 \ldots d_N$ con el cual se pueda lograr el máximo tiempo de grabación de un disco compacto que posee capacidad S. Para la implementación de la solución, se nos otorgó una solución ingenua y se proponía a mejorar su complejidad mediante dos métodos: Memoización y Programación dinámica. Estos dos métodos fueron implementados con complejidades temporales y espaciales menores.

Algoritmo de solución:

Para lograrlo se ha desarrollado un algoritmo de memoización y un algoritmo basado en programación dinámica. Estos se comportan de manera similar ya que van a ir almacenando información en una estructura de datos (matriz de tamaño S*N), aunque el algoritmo de memoización va consultando la estructura a medida que se requiere consultar un valor.

El algoritmo de programación dinámica va asignando máximos dentro de la estructura, desde la primera posición hasta a posición final, dejando en la posición S,N la máxima capacidad que se puede almacenar.

Para el algoritmo de memoización se van a tener las siguientes especificaciones:

$$\ulcorner ctx: S: int, N: int, duraciones[N-1]: array\ of\ int, matriz[N+1][S+1]: array\ of\ int$$
$$\{Q: S > 0 \wedge N > 0\}$$
$$\{R: \max(\ mem(s, i-1, duraciones, matriz), mem(s - duraciones[i-1], i - 1, duraciones, matriz) + duraciones[i-1])\}$$

Para el algoritmo de programación dinámica se van a tener las siguientes especificaciones:

$$\ulcorner ctx: duraciones[N-1]: array\ of\ int, S: int, N: int$$
$$\{Q: S > 0 \wedge N > 0\}$$
$$\{R: \text{matriz[N][S]} = \max(matriz[i-1][w], matriz[i-1]\left[w - duraciones[i-1]\right] + duraciones[i-1])\}$$

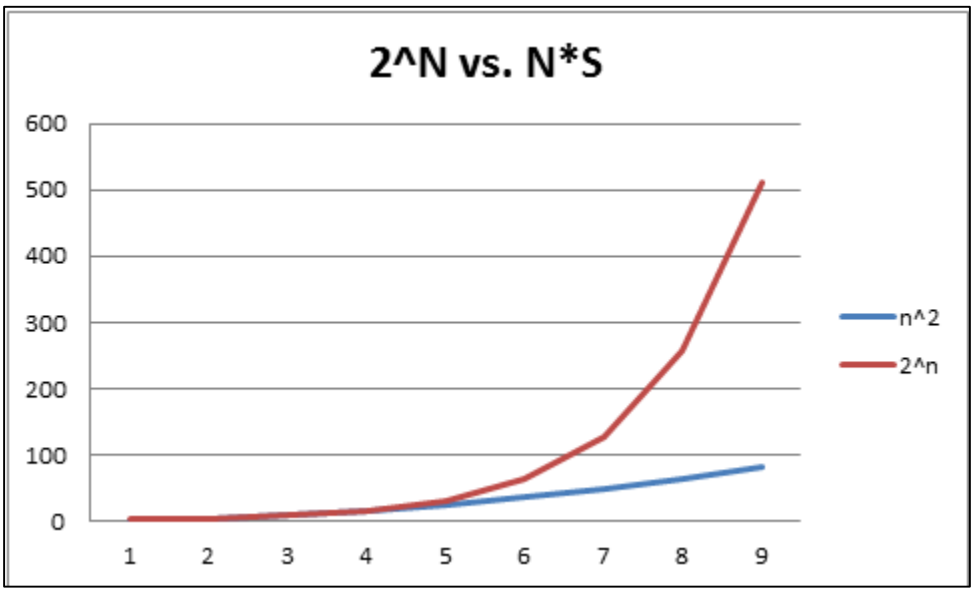
Se escogió implementar estos algoritmos ya que hacía parte de las restricciones del problema, así como lograr una reducción de la complejidad con estos, lo cual se logra pasando a una complejidad pseudopolinomial.

Análisis de complejidad:

En el momento de analizar la complejidad de la solución ingenua otorgada con el problema, se puede observas que lo que se pretende es realizar un árbol de recurrencias, donde en se van a tener N niveles. Esto nos da una complejidad temporal de la solución ingenua de orden $O(2^n)$. Para los métodos de solución planteados, se desarrollaron soluciones las cuales tuvieran como propósito mejorar la complejidad de la solución ingenua:

- Algoritmo de memoización, en el cual se va calculando un máximo a la vez y se agrega en una matriz, en la cual van a quedar estos datos para ser consultados en el futuro, así como ir almacenando un máximo general. Estas operaciones van a tener una complejidad temporal de O(N*S).

- Programación dinámica, lo que se realiza es inicializar una matriz y rellenarla de ceros para posteriormente ir asignando a cala posición de la misma un valor máximo hasta el momento. Al final del algoritmo, el máximo general se encontrará en la posición N, S. Esta solución tiene una complejidad temporal y espacial de O(N*S).



Gráfica 1. Comparación complejidades

Como se puede observar en la anterior gráfica, y si consideramos que la complejidad O(N*S) se puede comportar como $O(n^2)$, en la gráfica se muestra que los algoritmos implementados son mejores para n > 4. Como no siempre va a pasar que S = N, el límite para que estos sean mejor aumentará, pero como la complejidad de la solución ingenia es exponencial, siempre llegará a ser peor que una complejidad pseudopolinomial.

Comentarios finales:

Para finalizar, se puede decir que el problema se comporta de igual manera que un problema del morral (knapsack problema), el cual es un problema que se resuelve en tiempo pseudopolinomial con complejidad O(NS) y que es un problema NP-Completo.

Con programación dinámica y con memoización, se decide usar la matriz como estructura de almacenamiento de los resultados parciales del avance hacia la solución del problema. El algoritmo desarrollado con programación dinámica tiene como fuerte que tiene una complejidad espacial menor al algoritmo memoizado, lo cual hace que su desempeño sea mejor cuando se corre en consola. Los puntos débiles de los algoritmos es que van a depender del tamaño de la matriz, siendo este N*S. Igual, siempre y cuando se alcance el límite, este va a ser un mejor algoritmo al propuesto por la solución ingenua.