Tarea Diseño de Algoritmos

Estudiante: Ignacio Lara Vidal

Sección B-2

Idea:

Utilizando el método goloso, ordenaremos las canciones a partir de su "valor". El valor de cada canción será su puntuación dividida por su duración, así, entregando el "valor por minuto" de cada canción. Finalmente, agregaremos canciones a la cinta, dando prioridad a las que tengan mayor valor.

Algoritmo:

Entrada: Casos de prueba
Salida: Puntuación total máxima

```
Leer casos de prueba; Se genera una lista de listas del formato
[[duración, puntuación], [duración, puntuación], …] y se genera
largoCinta, con el largo de cada lado de la cinta.
Para i = 1 hasta largo(canciones):
    canciones[i] = [canciones[i][2] / canciones[i][1], canciones[i][1],
canciones[i][2]]
ordenando = True
mientras ordenando:
    para i = 1 hasta largo(canciones) - 1:
        si canciones[i + 1][1] > canciones[i][1]:
            aux = canciones[i]
            canciones[i] = canciones[i+1]
            canciones[i+1] = aux

    ordenando = False
    para i = 1 hasta largo(canciones) – 1:
        si canciones[i + 1] < canciones[i]:
            ordenando = True

total = 0

para i = 1 hasta 2:
    cintaNueva = largoCinta
    Llena = False
    mientras Llena == False:
        j = 1
        si canciones[j][2] <= cintaNueva:
            total += canciones[j][3]
            cintaNueva -= canciones[j][2]
```

```
            canciones.eliminar(j) #Elimina posición j de canciones
        si no:
            j += 1
            si j > largo(canciones):
                Llena = True
mostrar total
```

Tiempos del algoritmo:

$T(n) = n + n(n-1)(4) + n(2) + (2 + J(4)) + (2 + K(4))$
$T(n) = 4n^2 - n + 4J + 4K + 4$
Siendo n el número de canciones, J el número de canciones que entran en el lado a de la cinta, y K el número de canciones que entran en el lado b de la cinta.

$O(n) = (n^2)$

Análisis del algoritmo:

Se observa a simple vista que la sección más densa del pseudocódigo corresponde a la parte de ordenar las canciones por su valor por minuto. En este aspecto, notamos que anteriormente ya guardamos el valor por minuto asociado a cada canción, por lo que (de manera muy ligera) se utilizó la memoización, en lugar de tener que calcular el valor para cada iteración, sin embargo, no estoy muy seguro de poder considerarlo programación dinámica, al no verse envuelto en procesos recursivos. Respecto a paralelismo, se podría utilizar al momento de calcular la puntuación por minuto de cada canción, además de poder reemplazar el algoritmo de orden por un merge Sort paralelo, el cual, con suficientes núcleos, llegaría a ser de orden log(n), reduciendo el O(n) a log(n).

Pese a que el algoritmo sí puede entregar valores, en su mayoría correctos, sí es posible que se presenten combinaciones de canciones que no funcionen correctamente con el algoritmo entregado.