Universidad Simón Bolívar
División de Ciencias Físicas y Matemáticas
Departamento de Computación y Tecnología de la Información
CI5651: Diseño de Algoritmos I

Profesor: Ricardo Monascal. **Estudiante**: Astrid Alvarado, 18-10938.

Tarea 2 (9 %):

1. Pregunta 1

La idea consiste en calcular, para cada canción, el **tiempo en el que finaliza** las mismas. Esto es:

$$t_{final_i} = t_i + d_i,\ 1 \leq i \leq n$$

Con esto, se ordenarán las canciones de menor a mayor en función del tiempo en que terminan. Esto asegurará que se puedan escoger las canciones que terminan más temprano primero para así maximizar la cantidad de canciones que se quieren escuchar. Note que al hacer esto, implícitamente, se está escogiendo las canciones que tienen menor duración para así poder escuchar una mayor cantidad de canciones.

A continuación se presenta el siguiente pseudo-código con lo antes expresado. Se supone que existe una estructura especial *Canciones* con atributos `init` (tiempo en el que inicia la canción), `duration` (duración de la canción) y `finish` (tiempo en el que termina la canción)

```
function selecteSongs(canciones: array [1..n] de Canciones)
    selectedSongs ← ∅

    for c in canciones do
        c.finish ← c.init + c.duration
    end for

    Ordenar canciones de menor a mayor en función de c.finish

    selectedSongs ← selectedSongs ∪ {canciones[1]}
    lastSong ← canciones[1]

    for c in canciones[2:] do                ▷ Se empieza desde la segunda canción
        if c.init ≥ lastSong.finish then
            selectedSongs ← selectedSongs ∪ {c}
            lastSong ← c
        end if
    end for

    return selectedSongs
end function
```

El escoger las canciones en función del tiempo en que terminan permite "liberar" el tiempo más rápido, asegurando que el siguiente espacio disponible para la otra canción sea lo más temprano posible. Ahora, note que:

- Calcular el tiempo de finalización toma $O(n)$ dado que solo es necesario iterar una vez por el arreglo de canciones para calcularlo.
- Ordenar el arreglo toma $O(n\ log\ n)$
- Recorrer el arreglo para seleccionar las canciones toma $O(n)$ pues solo se itera una vez por la misma.

Dado que el paso más costoso del algoritmo es el ordenamiento, se tendrá que la complejidad de tiempo es $O(n\ log\ n)$. Por otro lado:

- Almacenar la información de las n canciones requiere de $O(n)$
- El conjunto maximal, en el peor de los casos, requiere de $O(n)$

Por lo tanto, el uso de memoria del algoritmo en general es de $O(n)$.

Este programa fue implementado en `C++` y se encuentra disponible en el siguente enlace.

2. Pregunta 2

a) Para demostrar que M_T es una matroide, se debe demostrar que:

- F es un conjunto finito:
  Se sabe que F se define como todas las firmas $X \to Y$ posibles entre los tipos de T. Suponiendo que T es finito, entonces este conjunto tiene una cardinalidad $|T| = n$, por lo que la cantidad de firmas que se pueden generar a partir de este conjunto equivale a la cantidad de formas de elegir un tipo en T para X y uno para Y, lo cual se puede hacer de $n \times n = n^2$ formas.

  Con esto, se tiene entonces que $|F| = n^2$, lo cual lo hace un conjunto finito.

- Se cumple la propiedad hereditaria:
  Para esto se debe probar que cualquier subconjunto A de B, siendo B definitivo (es decir, $B \in \mathcal{I}$), entonces A también será definitivo (es decir, $A \in \mathcal{I}$).

  Dado que B es definitivo, quiere decir que cada tipo aparece como máximo una vez como imagen en la firma. Al tomar un subconjunto A de B, no se están introduciendo nuevos elementos que puedan generar un conflicto. En cambio, se está retirando firmas, lo que garantiza que los conflictos existentes no aumenten ni aparezcan nuevos. Por lo que, como A no tiene conflictos, cumple con la defínción de *definitivo* y, por ende, $A \in \mathcal{I}$.

- Se cumple la propiedad de intercambio:
  Para esto se debe probar que si para dos conjuntos definitivos A y B, donde A tiene menos firmas que B (es decir, $|A| < |B|$), entonces debe existir al menos una firma en B que no esté en A (llámese f) que se pueda agregar a A para formar un nuevo conjunto $A \cup \{f\}$ que también sea definitivo.

  Dado que A y B son definitivos, se tiene que todas las imágenes de A y B son distintas, por lo que se puede decir que hay $|A|$ imágenes en A y $|B|$ imágenes

en B. Por otro lado, como $|A| < |B|$ se infiere que existe al menos una firma f que está en B pero que no está en A, y por lo tanto, debe existir al menos una imagen Y que no forma parte de ninguna de las imágenes de las firmas de A.

Con esta información, al realizar $A \cup \{f\}$ se puede asegurar que dicho conjunto será definitivo, pues se garantiza que no hay conflictos al hacer esta extensión dado que f incluye la imagen Y que, como se mencionó, no formaba parte de ninguna firma en A.

Por lo tanto, M_T es una matroide. ■

b) Se define w como:

$$w : F \to \mathbb{R}$$
$$w(X \to Y) = |Y|^{|X|}$$

Con esta función, y gracias al uso de matroides, el algoritmo voraz generalizado puede encontrar la solución óptima para este problema pues el algoritmo ordena todas las firmas disponibles de forma decreciente según su potencial, asegurando que se considere la firma más con mayor potencial. Luego, al recorrer esta lista ordenada, en cada iteración, la firma se añade al conjunto de la solución solo si no viola la condición de "ser definitivo". Si agregar la firma genera un conflicto, se descarta y se continúa con la siguiente.

Por lo tanto, la función w es una función de costo acorde al problema.