Algoritmos y Estructuras de Datos (CB100, 95.15) – Curso Buchwald

2.do parcialito – 20/10/2025

Resolvé los siguientes problemas en forma clara y legible. Podés incluir tantas funciones auxiliares como creas necesarias. **Los algoritmos a implementar deben ser de la menor complejidad posible dadas las características del problema planteado.**

1. A lo largo de su trayectoria, la empresa ha tenido varias rotaciones de equipos. Próximamente habrá una nueva, y se busca que los nuevos equipos incluyan personas que hayan compartido equipos antes, para facilitar la transición.

   Se dispone de una base de datos que registra, para `N` personas, en qué `M` equipos ha participado cada una. La base usa un hash donde la persona es la clave y el valor, una lista de equipos:

   ```
   {
       'Ana':   ['Frontend-Platform', 'Growth-Squad'],
       'Beto':  ['Backend-Services', 'Frontend-Platform', 'Mobile-Core'],
       'Carla': ['Mobile-Core'],
   }
   ```

   Realizar una función `personasEnComun` que reciba el hash, y el nombre de una persona , y devuelva una lista con todas las personas que hayan trabajado en al menos uno de sus equipos listados. Indicar y justificar la complejidad del algoritmo implementado, expresada con las variables `N` y `M` del problema. Por ejemplo, si se consulta por 'Beto', la respuesta incluye a 'Ana' y 'Carla'. Si se pregunta por 'Beto', la respuesta incluye a 'Beto'.

2. En nuestro juego de rol táctico, Final Fantasy Algorithms, el orden de ataque se decide por el atributo *iniciativa*: los de mayor iniciativa atacan primero. En esta batalla, un hechizo global está a punto de activarse, por lo que solo quedan `T` turnos en los que se pueda realizar un ataque. Se desea saber, de los `N` personajes, cuáles lograrán atacar antes de que se active el hechizo Final Fantástico Algorítmico. Se tiene una lista con los personajes que participarán en el combate como una estructura de formato `(nombre string, iniciativa int)`:

   ```
   [ ('Ma-Go Lang', 95), ('Bárbara', 75), ('Cléri-Go Lang', 60), ('Arquera de bugs', 90) ]
   ```

   Se pide realizar una función `determinarOrdenDeAtaque` que reciba la lista de combatientes, y la cantidad `T` turnos de turnos restantes. La función debe devolver una lista con los nombres de los personajes que logran actuar en esa ventana de tiempo, ordenados por turno en el que actúan. Indicar y justificar la complejidad del algoritmo implementado, expresada con las variables `N` y `T` del problema.

3. Implementar en Go una primitiva para Árbol Binario `func (ab *Arbol[int]) ArbolEsPlantable() bool` que determine si un árbol es plantable, o no. Para que lo sea, todo nodo debe cumplir: el dato del nodo debe ser mayor al dato de sus hijos (si los tiene), y además, el dato del nodo no puede superar la altura de dicho nodo. Implementar la primitiva en $\mathcal{O}(n)$, y justificar su complejidad. A fines del ejercicio considerar la estructura del árbol como la definida al dorso.

Algoritmos y Estructuras de Datos (CB100, 95.15) – Curso Buchwald

2.do parcialito – 20/10/2025

Resolvé los siguientes problemas en forma clara y legible. Podés incluir tantas funciones auxiliares como creas necesarias. **Los algoritmos a implementar deben ser de la menor complejidad posible dadas las características del problema planteado.**

1. A lo largo de su trayectoria, la empresa ha tenido varias rotaciones de equipos. Próximamente habrá una nueva, y se busca que los nuevos equipos incluyan personas que hayan compartido equipos antes, para facilitar la transición.

   Se dispone de una base de datos que registra, para `N` personas, en qué `M` equipos ha participado cada una. La base usa un hash donde la persona es la clave y el valor, una lista de equipos:

   ```
   {
       'Ana':   ['Frontend-Platform', 'Growth-Squad'],
       'Beto':  ['Backend-Services', 'Frontend-Platform', 'Mobile-Core'],
       'Carla': ['Mobile-Core'],
   }
   ```

   Realizar una función `personasEnComun` que reciba el hash, y el nombre de una persona , y devuelva una lista con todas las personas que hayan trabajado en al menos uno de sus equipos listados. Indicar y justificar la complejidad del algoritmo implementado, expresada con las variables `N` y `M` del problema. Por ejemplo, si se consulta por 'Beto', la respuesta incluye a 'Ana' y 'Carla'. Si se pregunta por 'Beto', la respuesta incluye a 'Beto'.

2. En nuestro juego de rol táctico, Final Fantasy Algorithms, el orden de ataque se decide por el atributo *iniciativa*: los de mayor iniciativa atacan primero. En esta batalla, un hechizo global está a punto de activarse, por lo que solo quedan `T` turnos en los que se pueda realizar un ataque. Se desea saber, de los `N` personajes, cuáles lograrán atacar antes de que se active el hechizo Final Fantástico Algorítmico. Se tiene una lista con los personajes que participarán en el combate como una estructura de formato `(nombre string, iniciativa int)`:

   ```
   [ ('Ma-Go Lang', 95), ('Bárbara', 75), ('Cléri-Go Lang', 60), ('Arquera de bugs', 90) ]
   ```

   Se pide realizar una función `determinarOrdenDeAtaque` que reciba la lista de combatientes, y la cantidad `T` turnos de turnos restantes. La función debe devolver una lista con los nombres de los personajes que logran actuar en esa ventana de tiempo, ordenados por turno en el que actúan. Indicar y justificar la complejidad del algoritmo implementado, expresada con las variables `N` y `T` del problema.

3. Implementar en Go una primitiva para Árbol Binario `func (ab *Arbol[int]) ArbolEsPlantable() bool` que determine si un árbol es plantable, o no. Para que lo sea, todo nodo debe cumplir: el dato del nodo debe ser mayor al dato de sus hijos (si los tiene), y además, el dato del nodo no puede superar la altura de dicho nodo. Implementar la primitiva en $\mathcal{O}(n)$, y justificar su complejidad. A fines del ejercicio considerar la estructura del árbol como la definida al dorso.

```go
type Arbol struct {
    dato int
    izq *Arbol
    der *Arbol
}
```


```go
type Arbol struct {
    dato int
    izq *Arbol
    der *Arbol
}
```