ITESO
Diseño de Algoritmos
Otoño 2024
Prof. Luis Alvarado

ITESO, Universidad Jesuita de Guadalajara

Práctica 8

Instrucciones:

Resuelva **individualmente** los siguientes ejercicios. No se reciben trabajos fuera de la fecha establecida en la plataforma *Canvas*:

- Entregable: Archivo.pdf
  - El nombre del archivo será "P, #Práctica, Primer Nombre, Primer Apellido".
  - Ejemplo: "P1LuisAlvarado.pdf"

La calificación de esta práctica estará distribuida equitativamente entre los ejercicios. Agrega los comentarios necesarios al archivo.

Ejercicios:

1. Simulación de sistema de tuberías

Un sistema de tuberías conecta varios tanques de agua. Cada tanque tiene una capacidad de almacenamiento y está conectado a otros tanques a través de tuberías. Cada minuto, una cantidad de agua fluye del tanque de mayor capacidad a otro con menor capacidad. La cantidad de agua que fluye es igual a la mitad de la diferencia entre las capacidades actuales de ambos tanques.

Crea una simulación que, dada una lista de capacidades iniciales de tanques y una lista de conexiones, determine el estado de cada tanque después de n minutos.

- **Entrada**:
  - Capacidades: [50, 30, 70] (iniciales de cada tanque)
  - Conexiones: [(0, 1), (1, 2)] (índices de los tanques conectados)
  - Minutos: n = 2
- **Salida esperada**: [50.0, 45.0, 55.0] (estado de cada tanque tras 2 minutos)

---

2. Ordenar caravanas de camellos

Un mercader tiene caravanas de camellos que deben viajar por un desierto. Cada camello de una caravana tiene una capacidad máxima de carga, y el mercader necesita ordenar a sus caravanas de acuerdo con la capacidad total (suma de las

capacidades de cada camello) de menor a mayor. Si dos caravanas tienen la misma capacidad total, deben ordenarse alfabéticamente por nombre.

Desarrolla una función que, dado un diccionario con los nombres de las caravanas como claves y las capacidades de cada camello como una lista de valores, devuelva una lista de nombres de caravanas ordenada.

- **Entrada**:
  - Caravanas: {'Alfa': [10, 20, 30], 'Bravo': [15, 10], 'Charlie': [10, 20, 5]}
- **Salida esperada**: ['Bravo', 'Charlie', 'Alfa']

---

3. Verificar la validez de una grilla de juego

Imagina que tienes una grilla de juego de palabras donde cada fila y cada columna deben contener palabras de tres letras diferentes, y ninguna letra debe repetirse en la misma posición dentro de la grilla. Es decir, si hay una “A” en la primera posición de la primera fila, no puede haber otra “A” en la primera posición de ninguna otra fila o columna.

Dada una matriz de letras (lista de listas), crea una función que verifique si la grilla cumple con estas condiciones.

- **Entrada**:
  - Grilla: [['A', 'B', 'C'], ['B', 'C', 'A'], ['C', 'A', 'B']]
- **Salida esperada**: True
- **Caso de prueba**:
  - Grilla: [['A', 'B', 'C'], ['A', 'C', 'B'], ['C', 'A', 'B']]
  - Salida esperada: False (la “A” se repite en la primera posición de dos filas)

Problema 4: Simular ciclos de producción en una fábrica

En una fábrica, cada estación de trabajo requiere de ciertos materiales para producir un artículo. Cada ciclo de producción consume una cantidad específica de cada

material y puede producir uno o más artículos si hay suficientes materiales disponibles. El inventario de materiales se recarga periódicamente.

Dado un diccionario de estaciones y sus materiales requeridos, y una lista de cantidades iniciales, simula el proceso de producción durante n ciclos e indica cuántos artículos se pueden producir en cada estación al final.

- **Entrada**:
  - Materiales: {'estacion1': {'madera': 3, 'metal': 1}, 'estacion2': {'madera': 2, 'piedra': 2}}
  - Inventario inicial: {'madera': 10, 'metal': 4, 'piedra': 6}
  - Ciclos: n = 3
- **Salida esperada**: {'estacion1': 2, 'estacion2': 2}
- **Caso de prueba**:
  - Materiales: {'A': {'madera': 1, 'hierro': 1}, 'B': {'hierro': 2, 'piedra': 1}}
  - Inventario inicial: {'madera': 3, 'hierro': 4, 'piedra': 2}
  - Ciclos: n = 2
  - Salida esperada: {'A': 2, 'B': 1}

Problema 5: Consolidar pedidos en almacenes

Un sistema de distribución tiene una lista de productos solicitados por clientes y varios almacenes. Cada almacén tiene una lista de productos disponibles y la cantidad de cada uno. La tarea es asignar los productos solicitados a los clientes de forma que se optimice el número de almacenes que deben abrirse.

Dado un diccionario con los productos solicitados y un diccionario con la disponibilidad de productos en cada almacén, determina qué almacenes abrir.

- **Entrada**:
    - Pedidos: {'manzanas': 10, 'peras': 5}
    - Almacenes: {'A': {'manzanas': 5, 'peras': 5}, 'B': {'manzanas': 5}, 'C': {'peras': 10}}
- **Salida esperada**: ['A', 'B']
- **Caso de prueba**:
    - Pedidos: {'manzanas': 8, 'uvas': 4}
    - Almacenes: {'X': {'manzanas': 8}, 'Y': {'uvas': 4}, 'Z': {'manzanas': 4, 'uvas': 2}}
    - Salida esperada: ['X', 'Y']

Problema 6: Optimizar rutas de entrega

Una empresa de logística tiene una lista de ciudades a las que debe enviar productos. Cada ciudad tiene una demanda específica de productos y cada camión tiene una capacidad máxima de carga. La tarea es agrupar las ciudades en rutas para minimizar la cantidad de camiones necesarios y, a su vez, cubrir toda la demanda.

- **Entrada**:
    - Ciudades: [("Ciudad1", 10), ("Ciudad2", 15), ("Ciudad3", 5), ("Ciudad4", 8)]
    - Capacidad del camión: 20
- **Salida esperada**: [['Ciudad1', 'Ciudad3'], ['Ciudad2'], ['Ciudad4']] (agrupación de ciudades en rutas)
- **Caso de prueba**:
    - Ciudades: [("A", 10), ("B", 8), ("C", 12), ("D", 5)]
    - Capacidad del camión: 14
    - Salida esperada: [['A'], ['B', 'D'], ['C']]