CS2023 - Aula de Ejercicios Nº 10

Brenner H. Ojeda Rios

Semestre 2024-1

Se sugiere que cada estudiante trate de resolver los ejercicios de forma **individual** y luego los discuta en grupo. Tópico: **Trie y Disjoint set.**

Ejercicios

1. (6 pts) Escriba una función para encontrar la cadena de prefijo común más larga entre un arreglo de strings.

   Si no hay un prefijo común, devuelve una cadena vacía. **Sólo será considerada soluciones que usen Tries**.

   - **Ejemplo 1:**
     `Input: strs = ['flower','flow','flight']`
     `Output: 'fl'`
   - **Ejemplo 2:**
     `Input: strs = ['dog','racecar','car']`
     `Output: ''`
     Explicación: No existe un prefijo común entre las cadenas de entrada.

   Restricciones:

   - $1 \leq$ `strs.length` ≤ 200
   - $0 \leq$ `strs[i].length` ≤ 200
   - `strs[i]` consta únicamente de letras minúsculas en inglés.

2. (7 pts) Queremos dividir un grupo de n personas (etiquetadas del 1 al n) en dos grupos de cualquier tamaño. Es posible que a cada persona no le gusten otras personas y no deben pertenecer al mismo grupo.

   Dado el número entero n y un arreglo `disgustos` donde `disgustos[i] = [`a_i`,` b_i`]` indica que a la persona etiquetada a_i no le gusta la persona etiquetada b_i, devuelve verdadero si es posible dividir a todos en dos grupos de esta manera. **Sólo será considerada soluciones que usen Disjoint set**.

   Ejemplo 1

   Input:

   `n = 4, dislikes = [[1,2],[1,3],[2,4]]`

   Output: `true`

   Explicación: El primer grupo tiene `[1,4]` y el segundo grupo tiene [2,3].

   Ejemplo 2

   Input:

   `n = 3, dislikes = [[1,2],[1,3],[2,3]]`

   Output: `false`

   Explicación: Necesitamos al menos 3 grupos para dividirlos. No podemos ponerlos en dos grupos.

Restricciones:

- $1 \leq n \leq 2000$
- $0 \leq$ `dislikes.length` $\leq 10^4$
- `dislikes[i].length` $= 2$
- $1 \leq a_i < b_i \leq n$
- Todos los pares de disgustos son únicos.

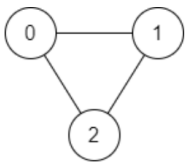
3. (7 pts) Hay un grafo bidireccional con n vértices, donde cada vértice está etiquetado del 0 al $n - 1$ (inclusive). Las aristas en el grafo están representadas como un arreglo bidimensional de enteros `edges`, donde cada `edges[i] = [`u_i`, `v_i`]` denota una arista bidireccional entre el vértice u_i y el vértice v_i. Cada par de vértices está conectado por como máximo una arista, y ningún vértice tiene una arista hacia sí mismo.

   Se quiere determinar si existe un camino válido desde el vértice `source` hasta el vértice `destination`.

   Dados `edges` y los enteros n, `source` y `destination`, devuelve `true` si existe un camino válido desde `source` hasta `destination`, o `false` en caso contrario. **Sólo será considerada soluciones que usen Disjoint set**.

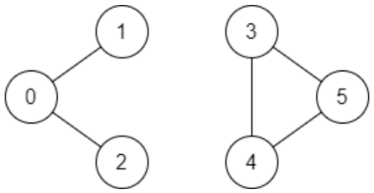
   Ejemplo 1:

```
Input: n = 3, edges = [[0,1],[1,2],[2,0]], source = 0, destination = 2
Output: true
Explanation: Hay dos caminos desde el vértice 0 al vértice 2:
- 0 → 1 → 2
- 0 → 2
```



   Ejemplo 2:

```
Input: n = 6, edges = [[0,1],[0,2],[3,5],[5,4],[4,3]], source = 0, destination = 5
Output: false
Explanation: No hay un camino desde el vértice 0 al vértice 5.
```



   Restricciones:

- $1 \leq n \leq 2 \times 10^5$
- $0 \leq$ `edges.length` $\leq 2 \times 10^5$
- `edges[i].length` $= 2$

- $0 \leq u_i, v_i \leq n - 1$
- $u_i \neq v_i$
- $0 \leq \texttt{source}, \texttt{destination} \leq n - 1$
- No hay aristas duplicadas.
- No hay aristas hacia sí mismas.