Editorial Tarea 2

Profesor: Andrés Abeliuk
Auxiliares: Daniel Báez
Julieta Coloma
Máximo Flores Valenzuela
Blaz Korecic
Diego Salas

A. Florencia y los laberintos

Prerequisito: Grafos Implícitos: En este problema se nos presenta un grafo implícito, en el sentido de que no necesitamos guardar una lista de adyacencia si no que con solo mirar la grilla tenemos toda la información necesaria.

No necesitamos lista de adyacencia porque si nuestro nodo es una coordenada (x, y), sus vecinos serán los nodos $(x+1, y), (x-1, y), (x, y+1)$ y $(x, y-1)$ (siempre y cuando estén dentro de la grilla). Es decir, desde el nodo, representado por un par ordenado, podemos obtener sus vecinos.

Un ejemplo en código de un BFS en una grilla se encuentra en material docente.

Digamos que tenemos A casillas vacías inicialmente. La observación clave es que convertir k casillas vacías a murallas es lo mismo que escoger $A-k$ casillas vacías y convertir el resto a murallas. Si estas $A-k$ casillas son conexas, entonces la solución será válida.

Ahora, encontrar $A-k$ casillas conexas es un problema mucho más fácil, pues un BFS o DFS siempre visitará nodos en un orden tal que son conexos. Es decir, podemos hacer un BFS o DFS desde cualquier casilla vacía y limitarlo con algún contador para no visitar más de $A-k$ nodos, y finalmente convertir los nodos no visitados en murallas para obtener una respuesta válida.

C. Puzzle de hielo

La dificultad de este problema recae en modelarlo como un grafo. Para esto, es importante la siguiente observación: si estamos en un zona de tierra, podemos visitar cualquier otra que esté en la misma horizontal (coordenada y) o vertical (coordenada x). Al ser la intersección entre ambas coordenadas, una zona de tierra "conecta" una de estas horizontales con una vertical.

Con la información anterior, podemos crear un grafo: los nodos serán cada horizontal y vertical (cada coordenada x y cada coordenada y, pero solo las de las zonas iniciales), y una zona de tierra conecta el nodo de su coordenada x con el de la y.

Agregar una zona de tierra equivale a agregar una arista en este grafo. Ahora, podemos abstraer el problema y solo fijarnos en nuestro grafo. Si tenemos solo una componente conexa,

entonces se puede llegar de cualquier zona a cualquier otra y la respuesta será cero. Si tenemos x componentes conexas, ¿cuál es la forma más eficiente de conectarlas todas?

Cada arista agregada conecta como máximo dos componentes conexas, así que estaremos forzados a agregar como mínimo $x - 1$ aristas. Así, la respuesta será la cantidad de componentes menos uno luego de armar el grafo y contarlas.

E. Caos en U-Cursos

La intuición es crear un grafo donde los nodos sean los usuarios, y la aristas indican que comparten al menos una comunidad. La respuesta para un usuario x es la cantidad de nodos que puedo visitar en el grafo empezando desde el nodo que representa a x, que es lo mismo que el tamaño de su componente conexa.

Entonces, tenemos que computar dos cosas:

- A qué componente conexa pertenece cada nodo, y
- el tamaño de cada componente conexa.

Lo anterior se puede realizar con DFS o BFS. La idea es:

i) Iteramos por todos los nodos. Cada vez que encontramos uno no visitado, realizamos un DFS/BFS desde él. Esto dejará toda su componente conexa como visitada.

ii) En este DFS/BFS agregamos un parámetro adicional para indicar el representante de la componente conexa. Éste será el nodo inicial (en el DFS habrá que agregar un parámetro, en el BFS probablemente ya esté).

iii) Cada vez que visitamos un nodo nuevo sumamos uno a `tam[representante]`, donde `tam` será un vector o mapa que guarde el tamaño de cada componente conexa indexada por su representante.

iv) Cada vez que visitamos un nodo nuevo u asignamos `rep[u]` $=$ *representante* para guardar a qué componente conexa pertenece cada nodo.

v) Finalmente, la respuesta de cada nodo u será `tam[rep[u]]`.

La complejidad de DFS y BFS es $O(|V| + |E|)$, donde V es el conjunto de nodos y E el de aristas. Esto sucede porque se visita cada vez un nodo y se chequea dos veces cada arista (una vez en cada dirección en un grafo no dirigido).

El problema acá es que el grafo es muy grande. En el peor caso hay una comunidad con todas las personas, lo cual genera $n(n-1)/2 = O(n^2)$ aristas, y nuestra solución será $O(n^2)$ lo cual es muy lento para el time limit.

La idea clave es simplificar el grafo de una forma que las componentes conexas se mantengan, pues lo único que nos importa preservar son las componentes conexas, y para eso hay muchas aristas redundantes en el peor caso.

Una forma de hacer lo anterior es esta: si tenemos una comunidad con miembros $a_1, a_2, \ldots, a_k$, en vez de agregar todos los pares de nodos como aristas (cuadrático), podemos agregar las aristas $(a_1, a_2), (a_2, a_3), (a_3, a_4), \ldots, (a_{k-1}, a_k)$. Esto mantiene la conectividad de cada componente, y crea una cantidad lineal de aristas. Así, en total tendríamos $O(n)$ aristas y $O(|V| + |E|) = O(n)$, con lo cual nuestra solución pasa el tiempo límite.