

Ideas para la implementación de las funciones:

Supongamos que ya tengo mi matriz llena de 0's y 1's.

Primero: Identificar clusters.

Los diferentes clusters se van a identificar por números, empezando desde el 2. Todos los sitios que pertenezcan al mismo cluster tendrán el mismo número. Ahora, se irá recorriendo la matriz de manera "correcta" y aumentando el número del cluster (No está bien explicado, por eso hagomósb con un ejemplo), usando las matrices que están de ejemplo en el pdf del profe:

///	///		///
			///
///		///	
///			///

⇒

2	2		3
			3
4		5	
4			6

///	///		///
	///		///
///	///	///	
///			///

⇒

2	2		3
	2		3
2	2	2	
2			4

Los sitios en blanco quedan llenos con 0's.

Ahora, ¿cómo se hará esto?

Al principio, la matriz sólo tendrá 0's y 1's. Se empezará recorriendo la matriz de forma "correcta" hasta encontrar un

1. Luego a ese lugar se le asociará el número 2. Ahora se llenará de la siguiente forma:

Mientras find=true

↳ find chequea si se encuentra nuevo cluster.
Debe ir al final del for.

1.

1	1		1
			1
1		1	
1			1

⇒

2.

2	1		1
			1
1		1	
1			1

¿Hay 1's al lado de los 2's? Si hay cambiar los 1's por 3. (Este puede ser un contador de pasos).

3.

2	3		1
			1
1		1	
1			1

¿Hay 1's al lado de los 3's? Si hay cambiar por 4's. No hay en este caso. Se sobra restando el tamaño del cluster antes de este paso y después.

Ahora, el número del cluster es 2, si hay espacios en la matriz con números mayores que 2 se cambian por 2. y si aumentará el número del cluster y se reiniciará la variable step por un número adecuado.

4.

2	2		1
			1
1		1	
1			1

Queda así:

5.

2	2		3
			1
1		1	
1			1

Se recorre la matriz de manera "correcta" y se encuentra un nuevo cluster donde se coloca el 3.

6.

2	2		3
			4
1		1	
1			1

Hay 1's al lado del 3. por tanto, se cambian por 4.

7.

2	2		3
			3
1		1	
1			1

No hay 1's al lado del 3, por tanto se cambian los números mayores de 3, por 3.

8.

2	2		3
			3
4		1	
1			1

Encuentra un cluster donde se coloca el número 4.

9.

2	2		3
			3
4		1	
5			1

Hay 1's al lado del 4. Se cambian por 5.

10.

2	2		3
			3
4		1	
4			1

No hay 1's al lado del 5. Se cambian los números mayores que 4 por 4.

11.

2	2		3
			3
4		5	
4			1

Encuentra nuevo cluster donde se coloca un 5.

12.

2	2		3
			3
4		5	
4			1

Busca 1's al lado del 5. No encuentra, luego busca números mayores que 5 para cambiarlos por 5.

13.

2	2		3
4		5	
4			6

Encuentra nuevo cluster donde está el 6. Busca 1's al lado. No encuentra. Busca números mayores que 6 para cambiarlos por 6.

Así quedan clasificados los clusters.

Para el segundo ejemplo la clasificación se hace así:

1.

1	1		1
	1		1
1	1	1	
1			1

2.

2	1		1
	1		1
1	1	1	
1			1

3.

2	3		1
	1		1
1	1	1	
1		1	1

4.

2	3		1
	4		1
1	1	1	
1			1

5.

2	3		1
	4		1
1	5	1	
1			1

6.

2	3		1
	4		1
6	5	6	
1			1

7.

2	3		1
	4		1
6	5	6	
7			1

8.

2	2		1
	2		1
2	2	2	
2			1

9.

2	2		3
	2		1
2	2	2	
2			1

Aquí, buscar por
vecinos del 7,
no encuentra y
cambia los números
mayores de 2 por
2.

10.

2	2		3
	2		4
2	2	2	
2			1

11.

2	2		3
	2		3
2	2	2	
2			1

12.

2	2		3
	2		3
2	2	2	
2			4

Encontrar clusters percolantes

Luego de clasificar clusters, para encontrar si hay un cluster percolante se hace lo siguiente: Se recorre la primera fila de la matriz. Supongamos que encontramos el cluster número 2. Entonces se busca en la última fila si hay un elemento que pertenece al cluster número 2. Si encuentra este sería un cluster percolante. Si no, se sigue chequeando todos los clusters que tengan elementos en la primera fila. Allí se encontrarían todos los clusters percolantes que van de arriba a abajo (si los hay).

Ahora, para buscar los clusters percolantes que van de izquierda a derecha, lo que se hace es: Se buscan los clusters que tengan elementos en la 1 columna. Digamos que se encuentra el cluster número N . Entonces, se busca en la última columna si hay un elemento que pertenezca al cluster N . Si lo hay el cluster N es percolante. Si no se sigue buscando por la primera columna y así se encuentran todos los clusters percolantes de izquierda a derecha.