Introducción a la conectividad en grafos

Dr. Alonso Ramirez Manzanares
Depto. de Matemáticas
Univ. de Guanajuato

Problema de conectividad (1)

Problema de conectividad (1)

- Nos dan una secuencia de pares de enteros, donde cada entero representa un objeto de algún tipo y tenemos que interpretar el par *p*-*q* como "*p* está conectado con *q*".

Problema de conectividad (1)

- Nos dan una secuencia de pares de enteros, donde cada entero representa un objeto de algún tipo y tenemos que interpretar el par *p*-*q* como "*p* está conectado con *q*".

- Relación transitiva: Si "*p* está conectado con *q*" y "*q* está con *r*", entonces "*p* está conectado con *r*".

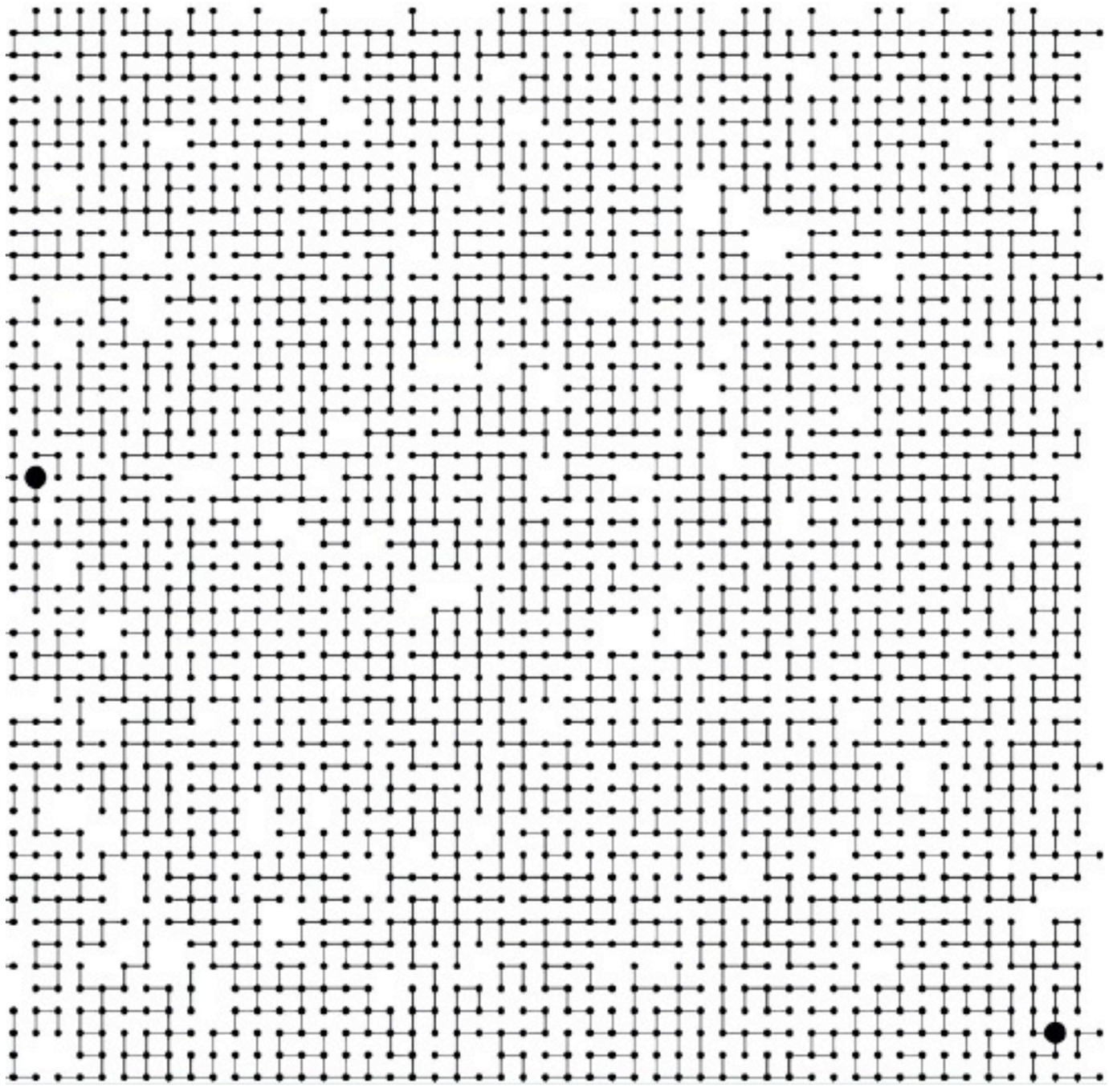
Problema de conectividad (1)

- Nos dan una secuencia de pares de enteros, donde cada entero representa un objeto de algún tipo y tenemos que interpretar el par *p*-*q* como "*p* está conectado con *q*".

- Relación transitiva: Si "*p* está conectado con *q*" y "*q* está con *r*", entonces "*p* está conectado con *r*".

- La meta es escribir un programa que filtre los pares externos al conjunto (guardar solo las nuevas conexiones mínimas): Para un par entrada *p*-*q*, la salida deberá ser el par solo si los pares revisados hasta entonces no implican que *p* está conectado a *q*. Si los pares visitados implican que *p* esta conectado a *q*, se debe ignorar el par *p*-*q* y continuaremos al siguiente par.

Esto es análogo saber si los 2 puntos están conectados:



Ejemplo 2 : Problema de conectividad (2)

Dados N nodos de índices 0 a N-1

Ojo, la salida nunca tendrá más de N-1 pares

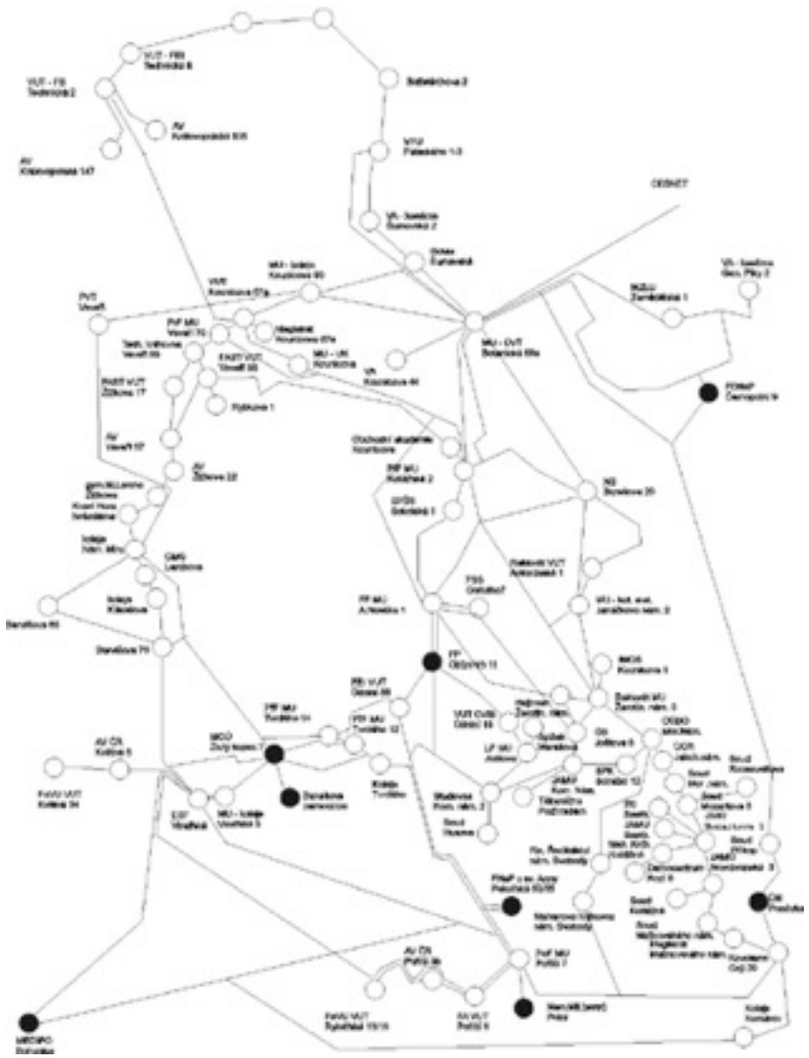
Ejemplo 2 : Problema de conectividad (2)

Dados N nodos de índices 0 a N-1

par entrada	salida	conexión
3-4	3-4	
4-9	4-9	
8-0	8-0	
2-3	2-3	
5-6	5-6	
2-9		2-3-4-9
5-9	5-9	
7-3	7-3	
4-8	4-8	
5-6		5-6
0-2		0-8-4-3-2
6-1	6-1	

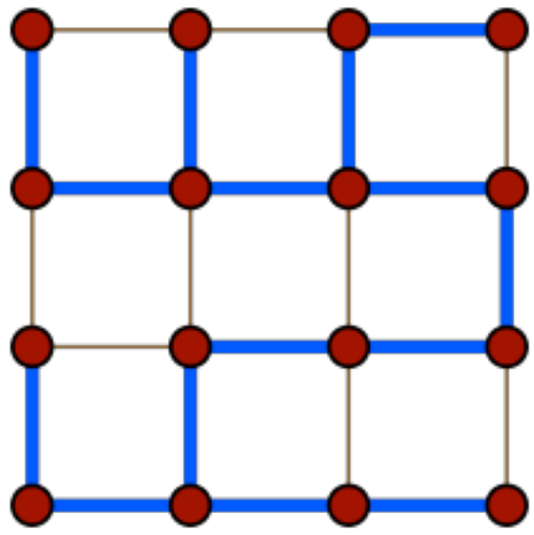
Ojo, la salida nunca tendrá más de N-1 pares

Ejemplo 2:Problema de conectividad (aplicaciones)



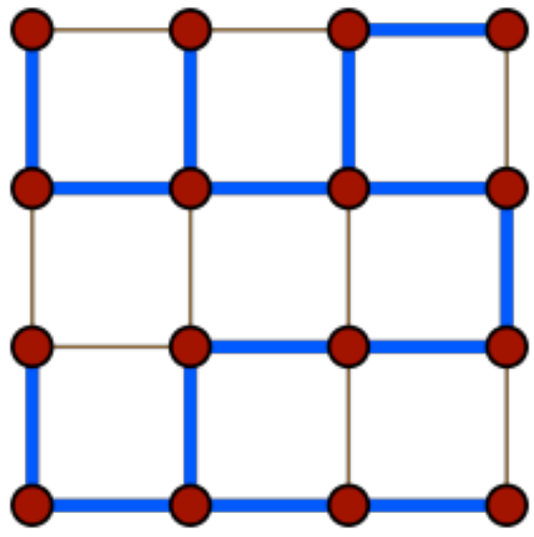
- redes de computadoras: se pueden usar las conexiones existentes o hay que hacer una nueva conexión?
- redes eléctricas, búsqueda de caminos
- si el número de entradas es pequeño cualquier algoritmo es razonable, si el número de entradas son millones de enteros...
- ¿Cómo podemos arreglarnos para determinar si dos pares están conectados?

Ejemplo 2: Problema de conectividad (4)



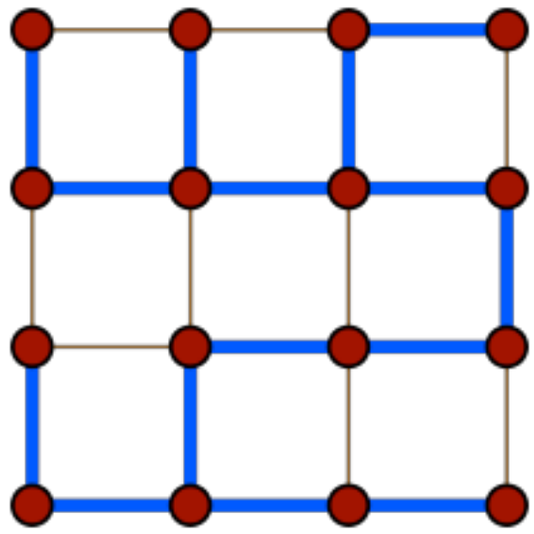
Ejemplo 2: Problema de conectividad (4)

- mientras más requiramos de un algoritmo, más tiempo y espacio se requieren para completar la tarea.



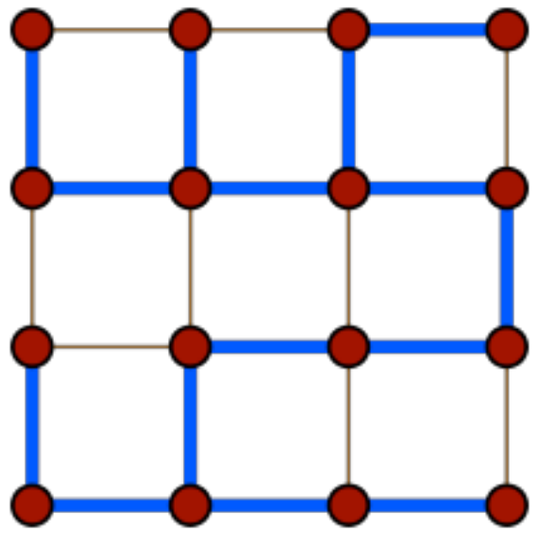
Ejemplo 2: Problema de conectividad (4)

- mientras más requiramos de un algoritmo, más tiempo y espacio se requieren para completar la tarea.
  - ¿están conectados p y q?



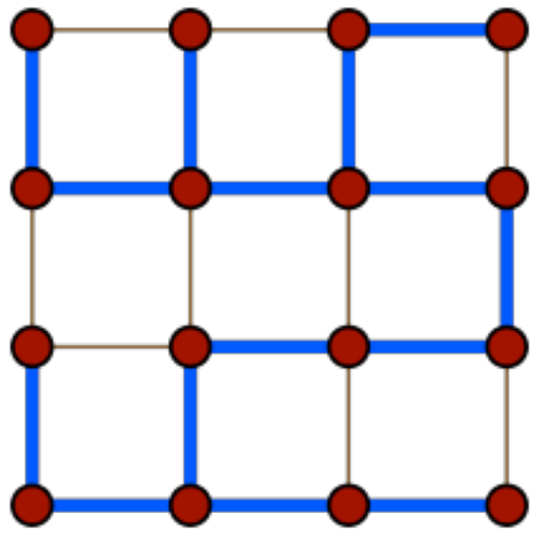
Ejemplo 2: Problema de conectividad (4)

- mientras más requiramos de un algoritmo, más tiempo y espacio se requieren para completar la tarea.
  - ¿están conectados p y q?
  - ¿son suficientes las conexiones existentes para que p y q estén conectados?



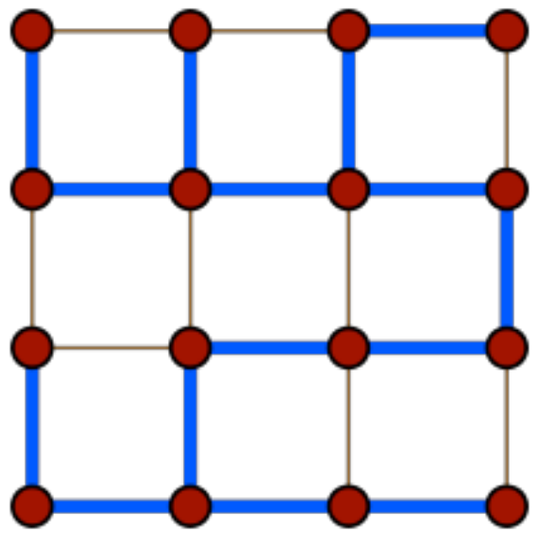
Ejemplo 2: Problema de conectividad (4)

- mientras más requiramos de un algoritmo, más tiempo y espacio se requieren para completar la tarea.
  - ¿están conectados p y q?
  - ¿son suficientes las conexiones existentes para que p y q estén conectados?
  - encuentra el camino para llegar de p a q



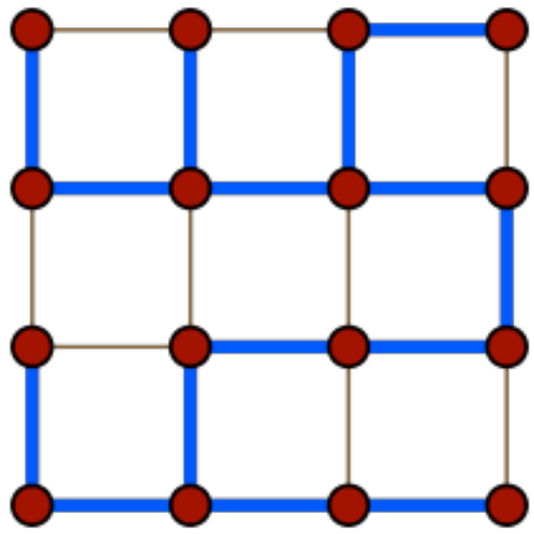
Ejemplo 2: Problema de conectividad (4)

- mientras más requiramos de un algoritmo, más tiempo y espacio se requieren para completar la tarea.
  - ¿están conectados p y q?
  - ¿son suficientes las conexiones existentes para que p y q estén conectados?
  - encuentra el camino para llegar de p a q
- el conjunto de pares en la entrada se llaman **grafo (informalmente es un conjunto de nodos unidos por aristas que determinar relaciones binarias entre elementos de conjuntos),** el conjunto de pares de salida se conoce como **árbol de recubrimiento (*spanning tree)*** de ese grafo.



Ejemplo 2: Problema de conectividad (4)

- mientras más requiramos de un algoritmo, más tiempo y espacio se requieren para completar la tarea.
  - ¿están conectados p y q?
  - ¿son suficientes las conexiones existentes para que p y q estén conectados?
  - encuentra el camino para llegar de p a q
- el conjunto de pares en la entrada se llaman **grafo (informalmente es un conjunto de nodos unidos por aristas que determinar relaciones binarias entre elementos de conjuntos),** el conjunto de pares de salida se conoce como **árbol de recubrimiento (*spanning tree)*** de ese grafo.

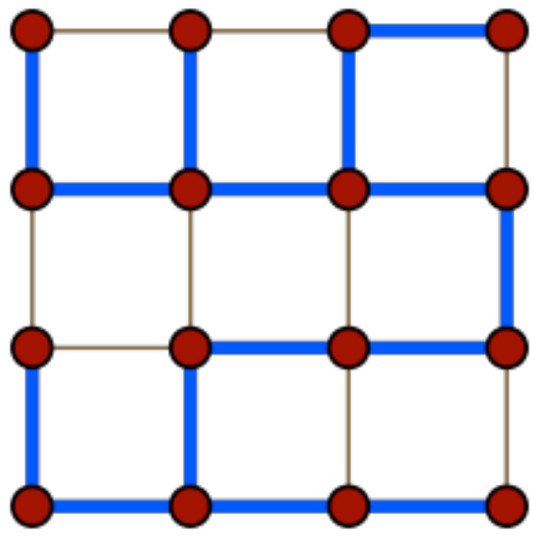


1. Especificar el problema claramente.

Ejemplo 2: Problema de conectividad (4)

- mientras más requiramos de un algoritmo, más tiempo y espacio se requieren para completar la tarea.
  - ¿están conectados p y q?
  - ¿son suficientes las conexiones existentes para que p y q estén conectados?
  - encuentra el camino para llegar de p a q
- el conjunto de pares en la entrada se llaman **grafo (informalmente es un conjunto de nodos unidos por aristas que determinar relaciones binarias entre elementos de conjuntos),** el conjunto de pares de salida se conoce como **árbol de recubrimiento (*spanning tree)*** de ese grafo.

1. Especificar el problema claramente.
2. Encontrar las operaciones necesarias para implementarlo.



Ejemplo 2: Problema de conectividad (5)

Ejemplo 2: Problema de conectividad (5)

- Operaciones fundamentales para un algoritmo de conectividad (útiles para resolver tareas similares sobre la misma estructura del grafo)

Ejemplo 2: Problema de conectividad (5)

- Operaciones fundamentales para un algoritmo de conectividad (útiles para resolver tareas similares sobre la misma estructura del grafo)
  - al tener un par nuevo: determinar si representa una nueva conexión.

Ejemplo 2: Problema de conectividad (5)

- Operaciones fundamentales para un algoritmo de conectividad (útiles para resolver tareas similares sobre la misma estructura del grafo)
  - al tener un par nuevo: determinar si representa una nueva conexión.
  - Si es así, incorporar la información que la conexión ha sido encontrada y actualizar la estructura del grafo.

Ejemplo 2: Problema de conectividad (5)

- Operaciones fundamentales para un algoritmo de conectividad (útiles para resolver tareas similares sobre la misma estructura del grafo)
  - al tener un par nuevo: determinar si representa una nueva conexión.
  - Si es así, incorporar la información que la conexión ha sido encontrada y actualizar la estructura del grafo.
  - Operaciones:

Ejemplo 2: Problema de conectividad (5)

- Operaciones fundamentales para un algoritmo de conectividad (útiles para resolver tareas similares sobre la misma estructura del grafo)
  - al tener un par nuevo: determinar si representa una nueva conexión.
  - Si es así, incorporar la información que la conexión ha sido encontrada y actualizar la estructura del grafo.
- Operaciones:
  - ***Encontrar*** el conjunto que contiene un elemento particular.

Ejemplo 2: Problema de conectividad (5)

- Operaciones fundamentales para un algoritmo de conectividad (útiles para resolver tareas similares sobre la misma estructura del grafo)
  - al tener un par nuevo: determinar si representa una nueva conexión.
  - Si es así, incorporar la información que la conexión ha sido encontrada y actualizar la estructura del grafo.
- Operaciones:
  - ***Encontrar*** el conjunto que contiene un elemento particular.
  - Reemplazar los conjuntos que contienen dos elementos dados por su ***unión***.

Ejemplo 2: Problema de conectividad (5)

- Operaciones fundamentales para un algoritmo de conectividad (útiles para resolver tareas similares sobre la misma estructura del grafo)
  - al tener un par nuevo: determinar si representa una nueva conexión.
  - Si es así, incorporar la información que la conexión ha sido encontrada y actualizar la estructura del grafo.
  - Operaciones:
    - ***Encontrar*** el conjunto que contiene un elemento particular.
    - Reemplazar los conjuntos que contienen dos elementos dados por su ***unión***.
- el problema de conectividad puede resolverse realizando estas dos operaciones. Cada conjunto se conoce como ***componentes conectados.***

Ejemplo 2: Problema de conectividad (6)

Ejemplo 2: Problema de conectividad (6)

- ¡Implementar un algoritmo simple que solucione el problema!

Ejemplo 2: Problema de conectividad (6)

- ¡Implementar un algoritmo simple que solucione el problema!
- **Idea 1:** guardar todos los pares de entrada, escribir una función para recorrerlos y ver si están conectados.

Ejemplo 2: Problema de conectividad (6)

- ¡Implementar un algoritmo simple que solucione el problema!
- **Idea 1:** guardar todos los pares de entrada, escribir una función para recorrerlos y ver si están conectados.
  - si el número de pares entrada es muy grande, la memoria necesaria podría superar los recursos de la máquina.

Ejemplo 2: Problema de conectividad (6)

- ¡Implementar un algoritmo simple que solucione el problema!
- **Idea 1:** guardar todos los pares de entrada, escribir una función para recorrerlos y ver si están conectados.
  - si el número de pares entrada es muy grande, la memoria necesaria podría superar los recursos de la máquina.
  - no conocemos ahora un método inmediato para determinar si dos objetos están conectados, incluso si se pudieran almacenar todos.

Ejemplo 2: Problema de conectividad (6)

- ¡Implementar un algoritmo simple que solucione el problema!
- **Idea 1:** guardar todos los pares de entrada, escribir una función para recorrerlos y ver si están conectados.
  - si el número de pares entrada es muy grande, la memoria necesaria podría superar los recursos de la máquina.
  - no conocemos ahora un método inmediato para determinar si dos objetos están conectados, incluso si se pudieran almacenar todos.
  - utilizaremos **arreglos** de enteros, donde cada objeto a conectar será un elemento del arreglo: si necesitamos 1000 enteros, el arreglo será id[1000]; y nos referiremos al entero i, escribiendo id[i] para 1 <= i <= N

Problema de conectividad: algoritmo *quick-find*

- *p* y *q* están conectados si y solo si los elementos *p* y *q* del arreglo son iguales.
- cuando procesamos cada par *p,q*, cambiamos todos los elementos que valgan id[p] por el valor de id[q].

Problema de conectividad: algoritmo *quick-find*

- *p* y *q* están conectados si y solo si los elementos *p* y *q* del arreglo son iguales.
- cuando procesamos cada par *p,q*, cambiamos todos los elementos que valgan id[p] por el valor de id[q].

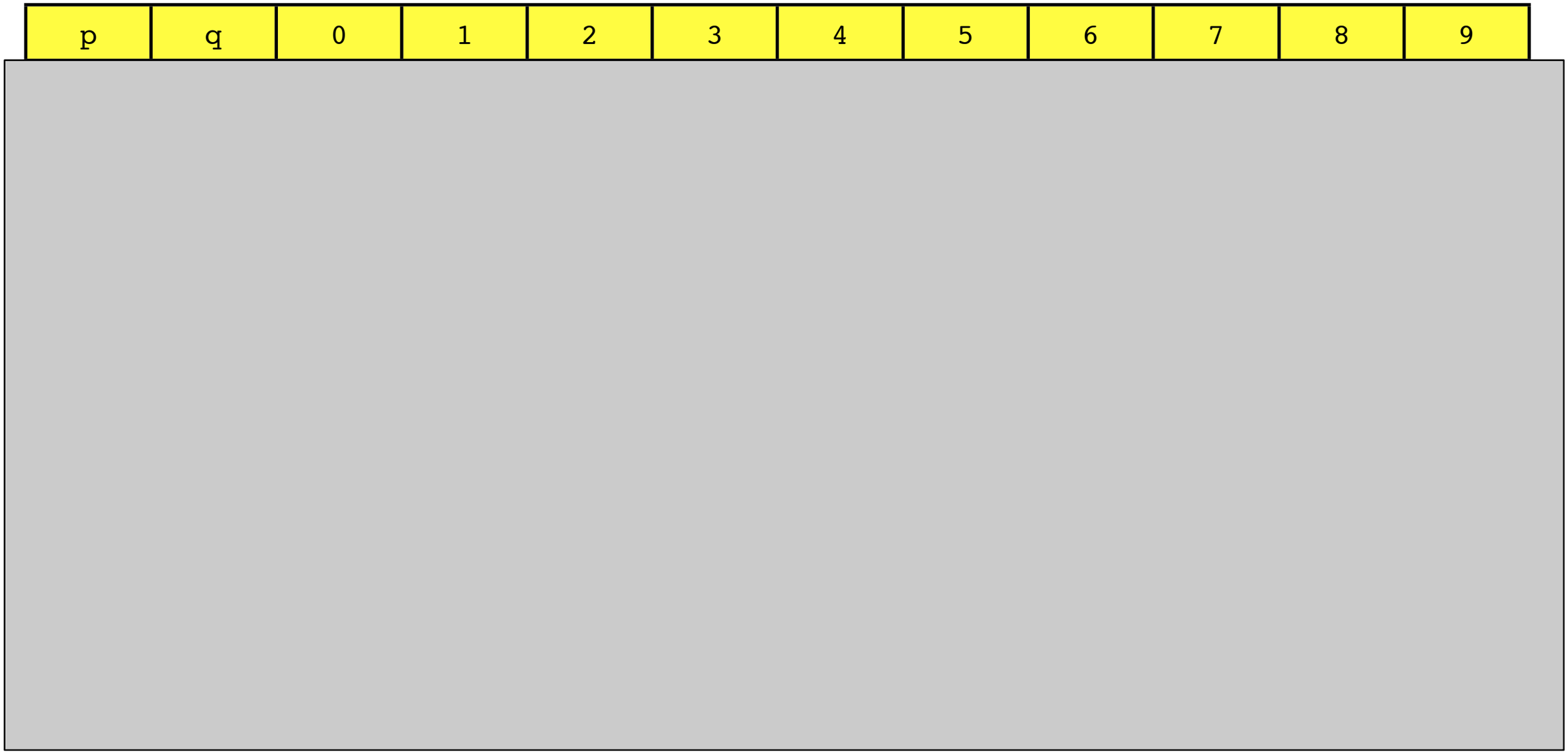
```
#include <iostream.h>
using namespace std;


static const int N = 10000;


int main()
{
        int i, p, q, id[N];
        for (i=0; i<N; i++)  // initialization
                        id[i] = i;
        while ( cin >> p >> q ){
                int t = id[p];
                if (t == id[q]) continue; // quick find
                for ( i=0; i<N; i++) // union
                        if (id[i] == t ) id[i]=id[q];
                cout << " " << p << " " << q << endl;
      }
}
```

Problema de conectividad: algoritmo *quick-find*

Problema de conectividad: algoritmo *quick-find*



Problema de conectividad: algoritmo *quick-find*

p	q	0	1	2	3	4	5	6	7	8	9
3	4	0	1	2	4	4	5	6	7	8	9

Problema de conectividad: algoritmo *quick-find*

p	q	0	1	2	3	4	5	6	7	8	9
3	4	0	1	2	4	4	5	6	7	8	9
4	9	0	1	2	9	9	5	6	7	8	9

Problema de conectividad: algoritmo *quick-find*

p	q	0	1	2	3	4	5	6	7	8	9
3	4	0	1	2	4	4	5	6	7	8	9
4	9	0	1	2	9	9	5	6	7	8	9
8	0	0	1	2	9	9	5	6	7	0	9

Problema de conectividad: algoritmo *quick-find*

p	q	0	1	2	3	4	5	6	7	8	9
3	4	0	1	2	4	4	5	6	7	8	9
4	9	0	1	2	9	9	5	6	7	8	9
8	0	0	1	2	9	9	5	6	7	0	9
2	3	0	1	9	9	9	5	6	7	0	9
5	6	0	1	9	9	9	6	6	7	0	9
2	9	0	1	9	9	9	6	6	7	0	9
5	9	0	1	9	9	9	9	9	7	0	9
7	3	0	1	9	9	9	9	9	9	0	9
4	8	0	1	0	0	0	0	0	0	0	0
5	6	0	1	0	0	0	0	0	0	0	0
0	2	0	1	0	0	0	0	0	0	0	0
6	1	1	1	1	1	1	1	1	1	1	1

Problema de conectividad: algoritmo *quick-find*

- Para implementar la operación *find* solo tenemos que probar si id[p] es igual a id[q].

- El algoritmo *quick-find* ejecuta *MN* instrucciones para resolver un problema de conectividad, donde *N* es el numero de objetos que involucran *M* operaciones de unión.

- Para M operaciones de unión, hay que iterar el ciclo *for* de N veces. Cada iteración requiere al menos una instrucción (verificar si la iteración se terminó)

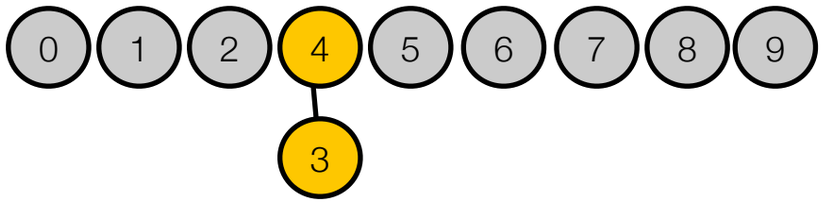
- Si el número de objetos es muy grande el problema no se puede realizar de esta forma en una computadora moderna.

Problema de conectividad: *quick-find* (árbol)

p	q
3	4
4	9
8	0
2	3
5	6
2	9
5	9
7	3
4	8
5	6
0	2
6	1

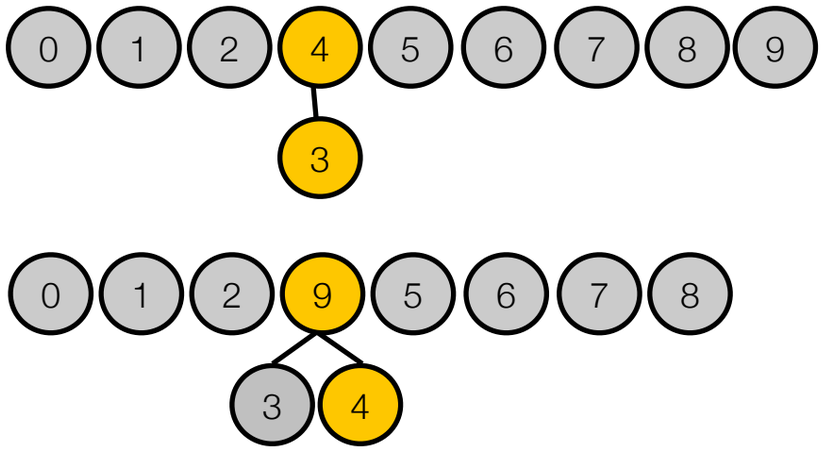
Problema de conectividad: *quick-find* (árbol)

p	q
3	4
4	9
8	0
2	3
5	6
2	9
5	9
7	3
4	8
5	6
0	2
6	1



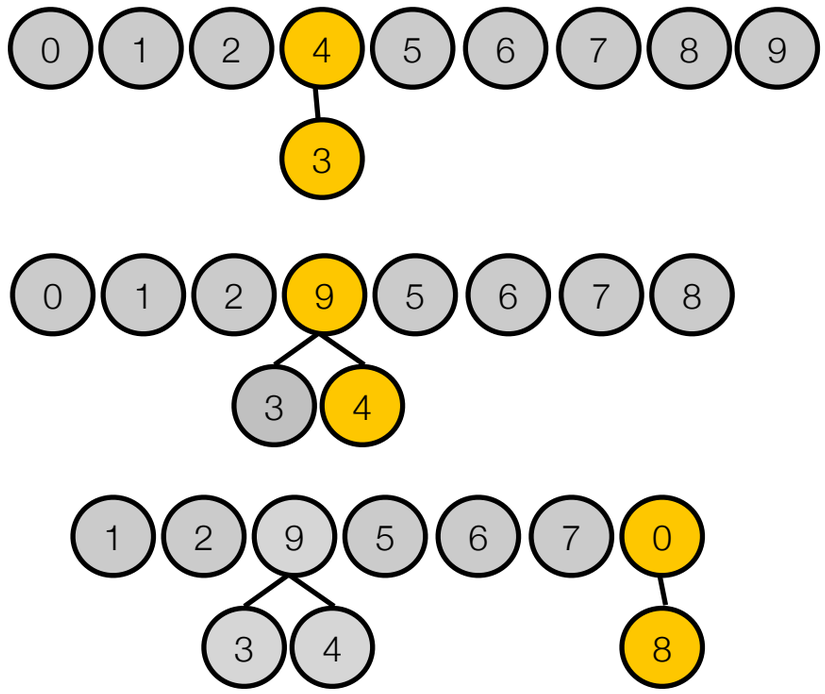
Problema de conectividad: *quick-find* (árbol)

p	q
3	4
4	9
8	0
2	3
5	6
2	9
5	9
7	3
4	8
5	6
0	2
6	1



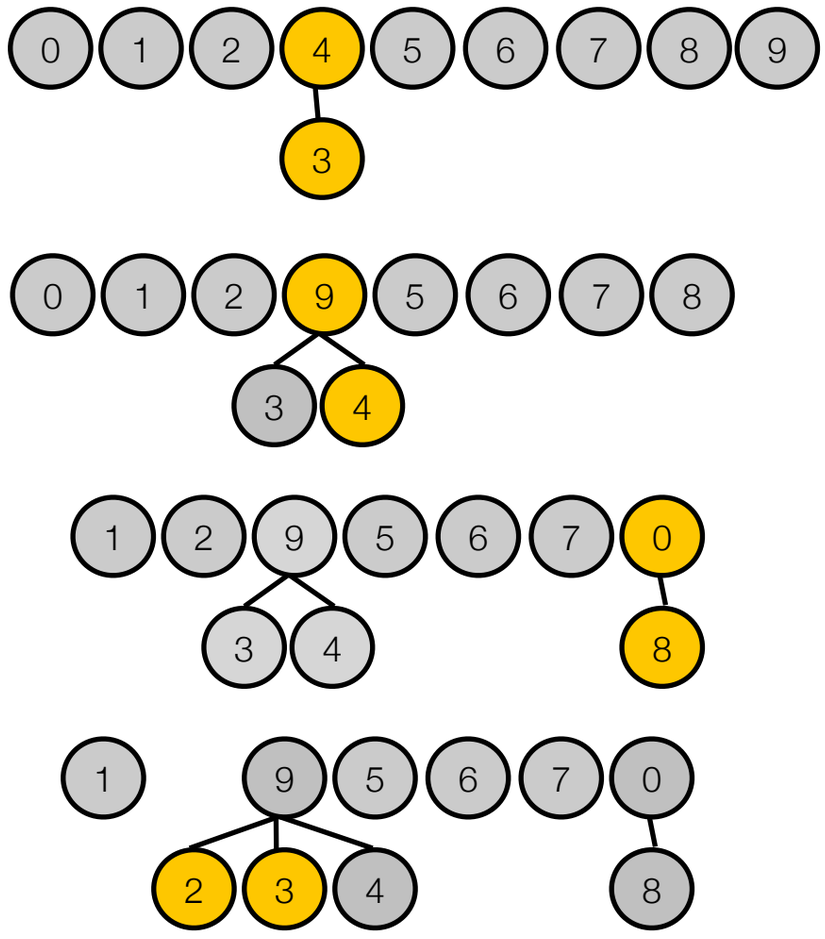
Problema de conectividad: *quick-find* (árbol)

p	q
3	4
4	9
8	0
2	3
5	6
2	9
5	9
7	3
4	8
5	6
0	2
6	1



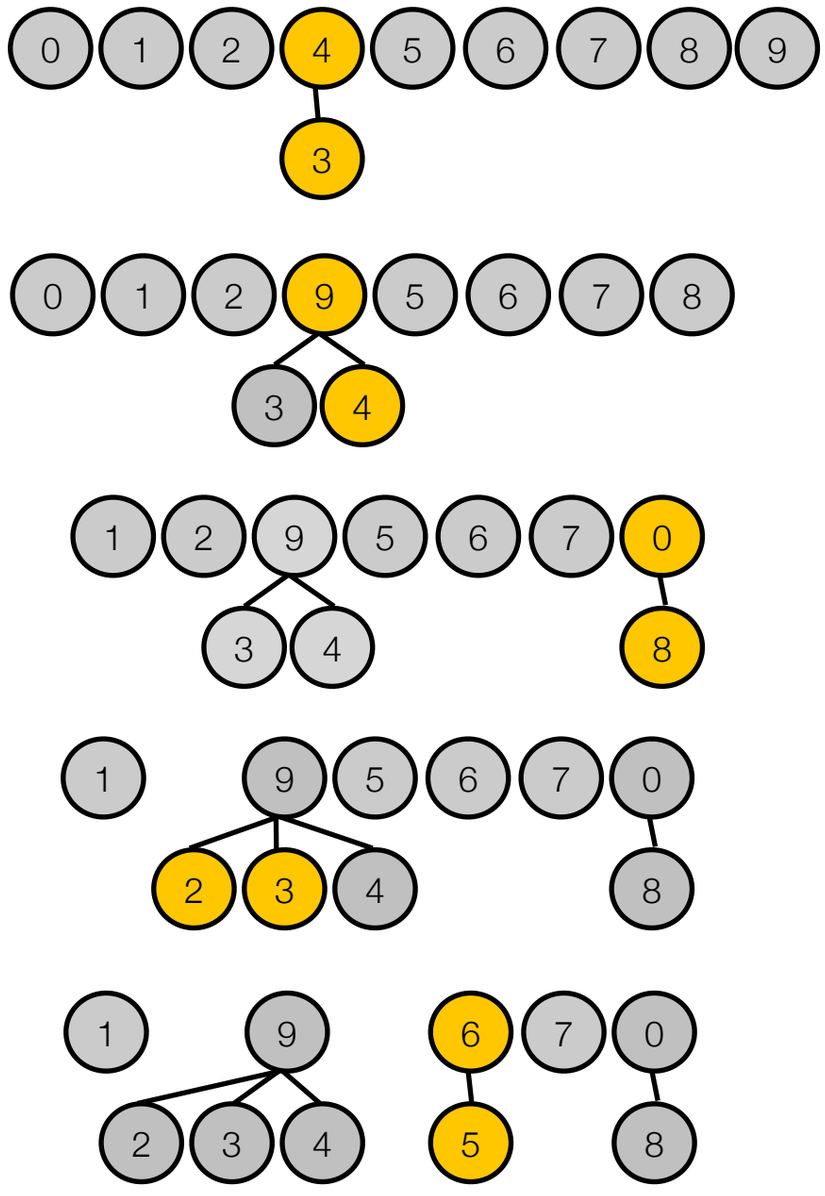
Problema de conectividad: *quick-find* (árbol)

p	q
3	4
4	9
8	0
2	3
5	6
2	9
5	9
7	3
4	8
5	6
0	2
6	1



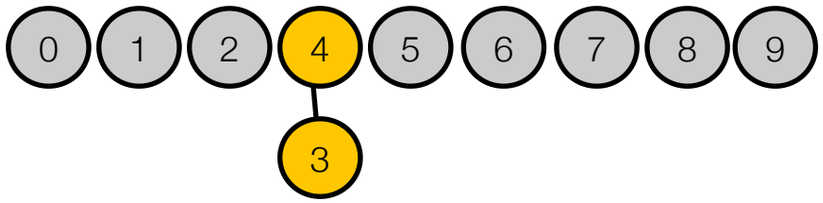
Problema de conectividad: *quick-find* (árbol)

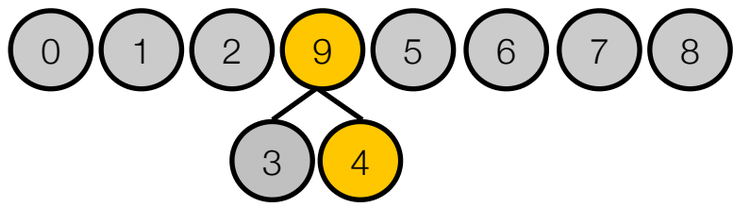
p	q
3	4
4	9
8	0
2	3
5	6
2	9
5	9
7	3
4	8
5	6
0	2
6	1

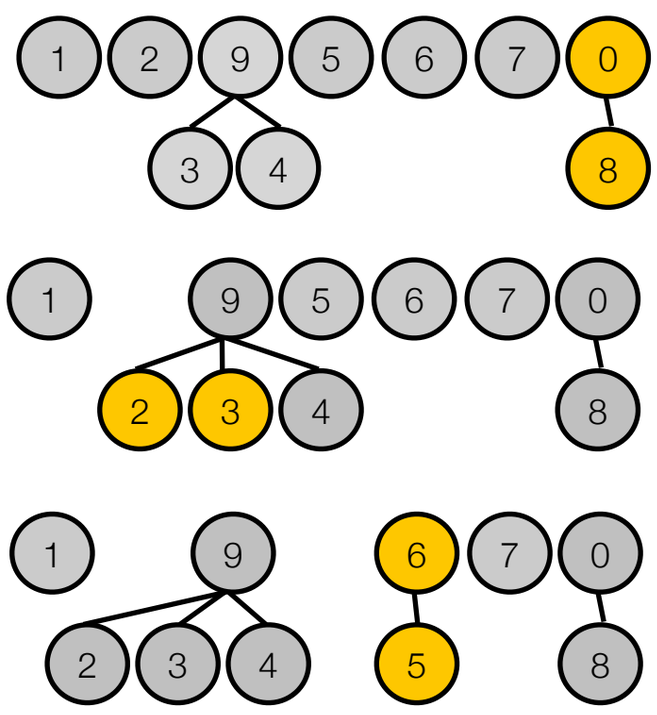


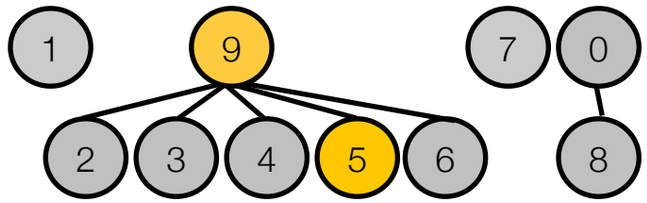
Problema de conectividad: *quick-find* (árbol)

p	q
3	4
4	9
8	0
2	3
5	6
2	9
5	9
7	3
4	8
5	6
0	2
6	1



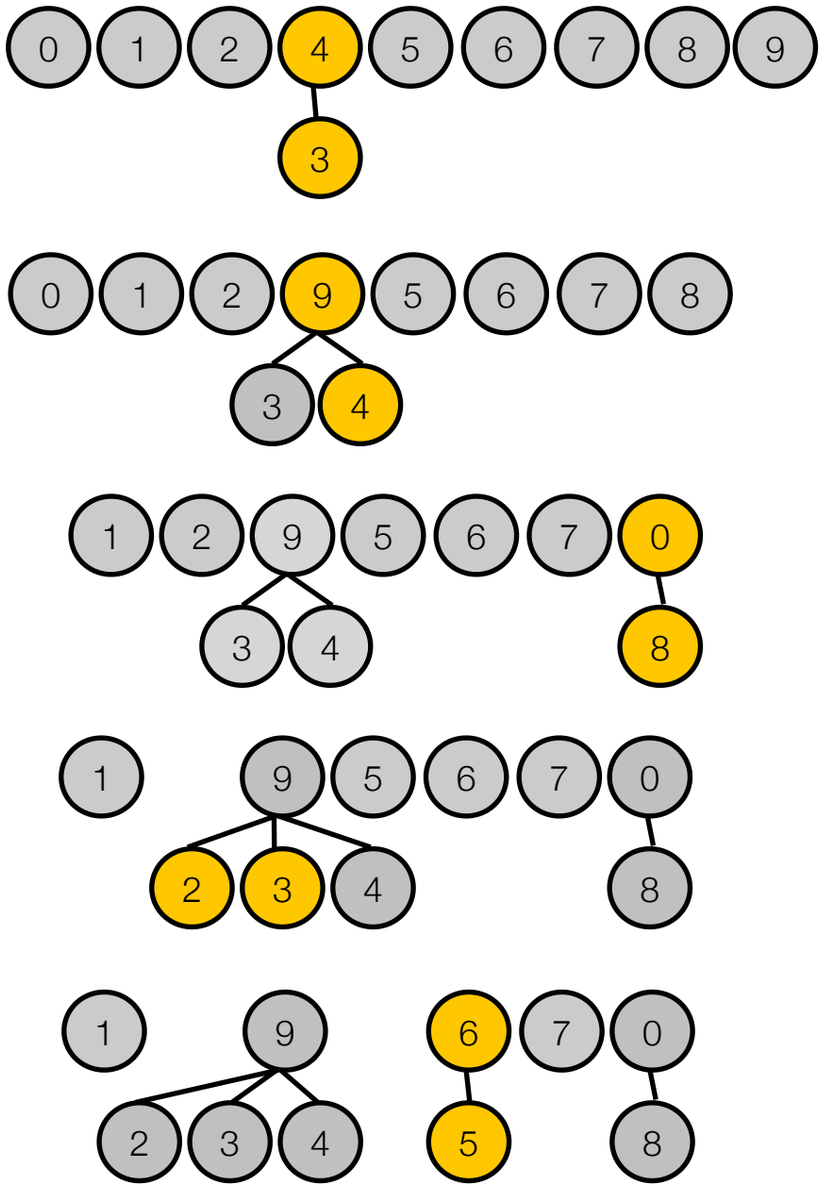


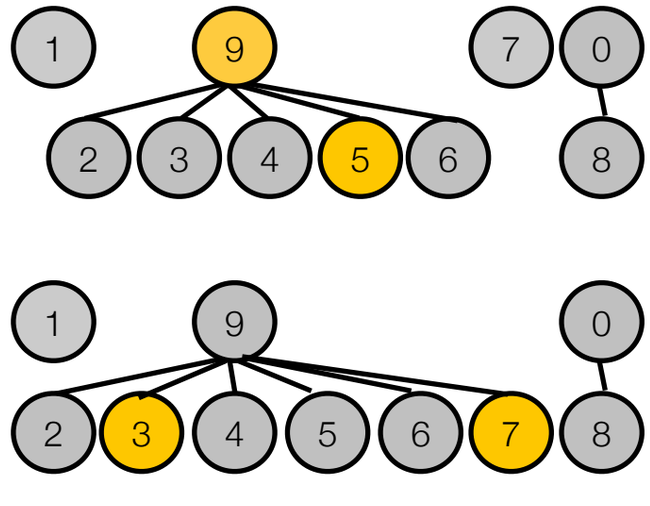




Problema de conectividad: *quick-find* (árbol)

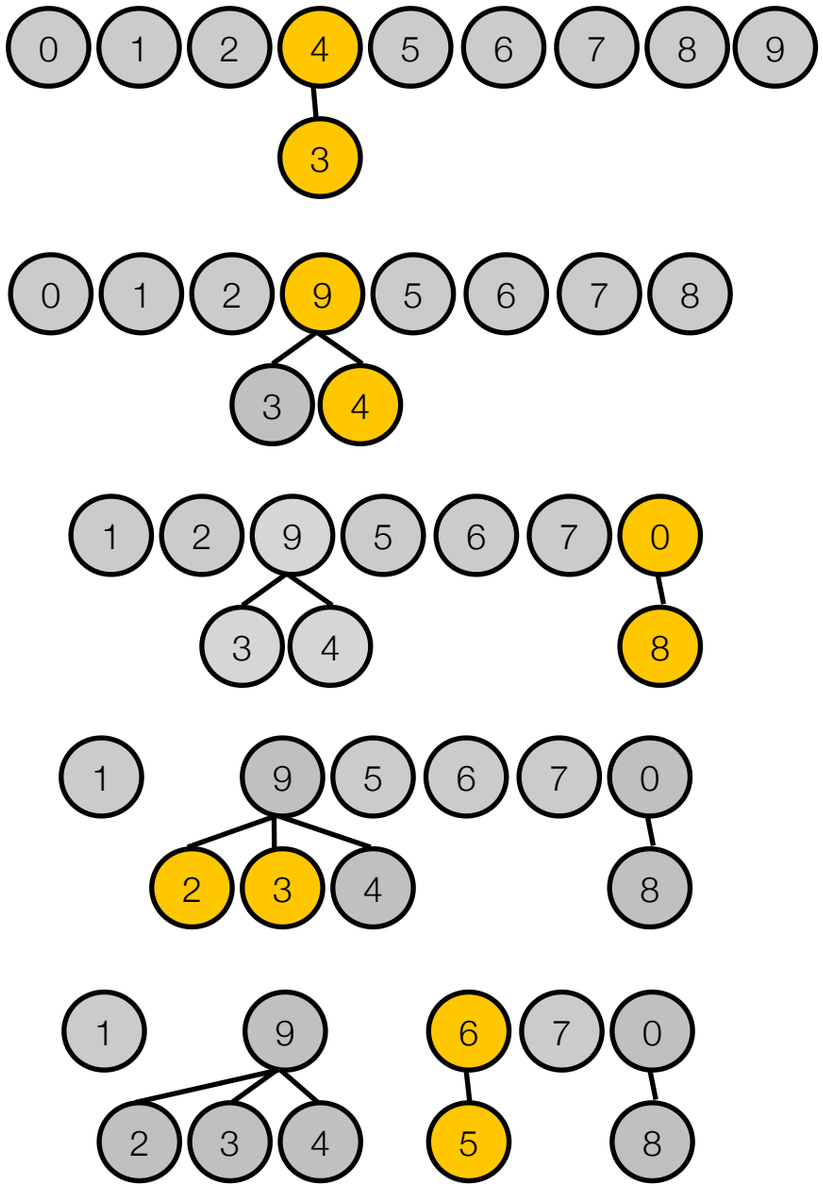
p	q
3	4
4	9
8	0
2	3
5	6
2	9
5	9
7	3
4	8
5	6
0	2
6	1

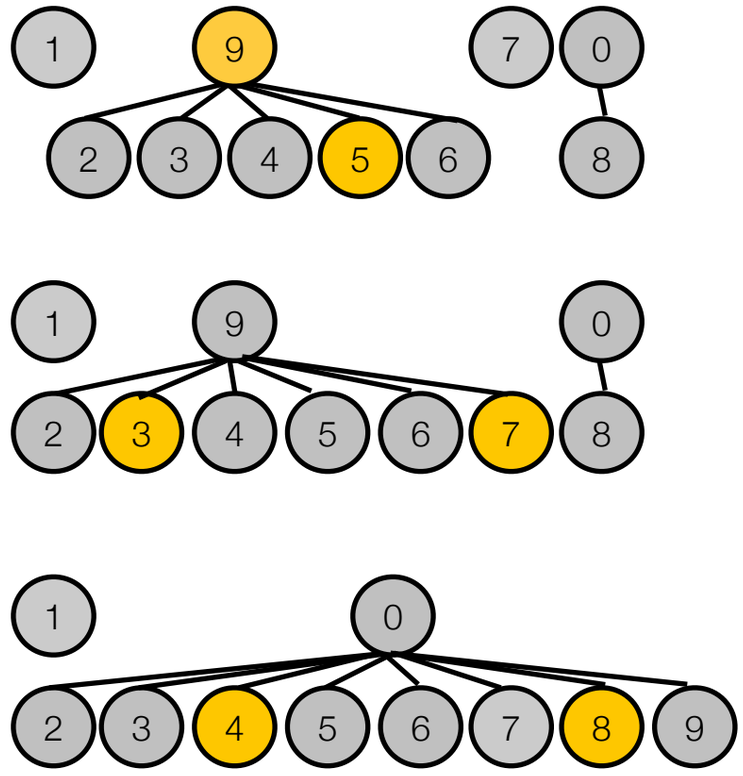




Problema de conectividad: *quick-find* (árbol)

p	q
3	4
4	9
8	0
2	3
5	6
2	9
5	9
7	3
4	8
5	6
0	2
6	1





Problema de conectividad: *quick-find* (árbol)

p	q
3	4
4	9
8	0
2	3
5	6
2	9
5	9
7	3
4	8
5	6
0	2
6	1