Union-Find, union de conjuntos disjuntos

Miguel Ortiz

Programación competitiva para ICPC

Mayo 2023 - Cochabamba, Bolivia

Union-Find

- Tenemos n elementos
- Mantenemos una colección de conjuntos disjuntos
- Cada uno de los n elementos está en exactamente un conjunto

- $elementos = \{1, 2, 3, 4, 5, 6\}$
- $colecciones = \{1, 4\}, \{3, 5, 6\}, \{2\}$
- $colecciones = \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}$

- Soporta dos operaciones eficientemente: `find(x)` y `union(x,y)`

Union-Find

- *elementos* $= \{1, 2, 3, 4, 5, 6\}$
- *colecciones* $= \{1, 4\}, \{3, 5, 6\}, \{2\}$

- `find(x)` retorna un elemento representativo del conjunto al que pertenece x
  - `find(1) = 1`
  - `find(4) = 1`
  - `find(3) = 5`
  - `find(5) = 5`
  - `find(6) = 5`
  - `find(2) = 2`

- a y b están en el mismo conjunto si y solo si `find(a) == find(b)`

Union-Find

- *elementos* $= \{1, 2, 3, 4, 5, 6\}$
- *colecciones* $= \{1, 4\}, \{3, 5, 6\}, \{2\}$

- `union(x, y)` une los conjuntos que contienen a x y y.

  - `union(4, 2)`
  - *colecciones* $= \{1, 2, 4\}, \{3, 5, 6\}$
  - `union(3, 6)`
  - *colecciones* $= \{1, 2, 4\}, \{3, 5, 6\}$
  - `union(2, 6)`
  - *colecciones* $= \{1, 2, 3, 4, 5, 6\}$

Implementación

- Unión rápida con compresión de caminos
- Implementación muy simple
- Muy eficiente

```
struct union_find {
    vector<int> parent;
    union_find(int n) {
        parent = vector<int>(n);
        for (int i = 0; i < n; i++) {
            parent[i] = i;
        }
    }

    // find, union
};
```

Implementación

```
// find, union

int find(int x) {
    if (parent[x] == x) {
        return x;
    } else {
        parent[x] = find(parent[x]);
        return parent[x];
    }
}

void unite(int x, int y) {
    parent[find(x)] = find(y);
}
```

Implementación (corta)

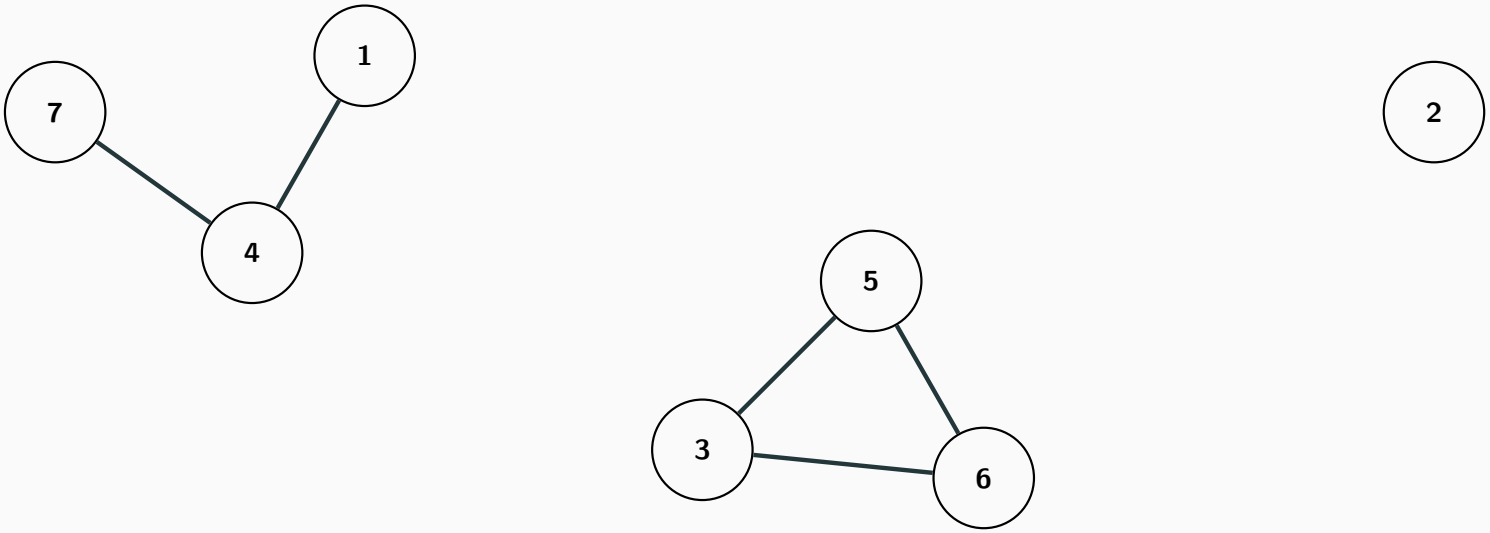
- Si están apurados...

```
#define MAXN 1000 // fijense en los limites del problema
int p[MAXN];

int find(int x) {
  return p[x] == x ? x : p[x] = find(p[x]);
}
void unite(int x, int y) {
  p[find(x)] = find(y);
}

// inicializar
for (int i = 0; i < MAXN; i++) p[i] = i;
```

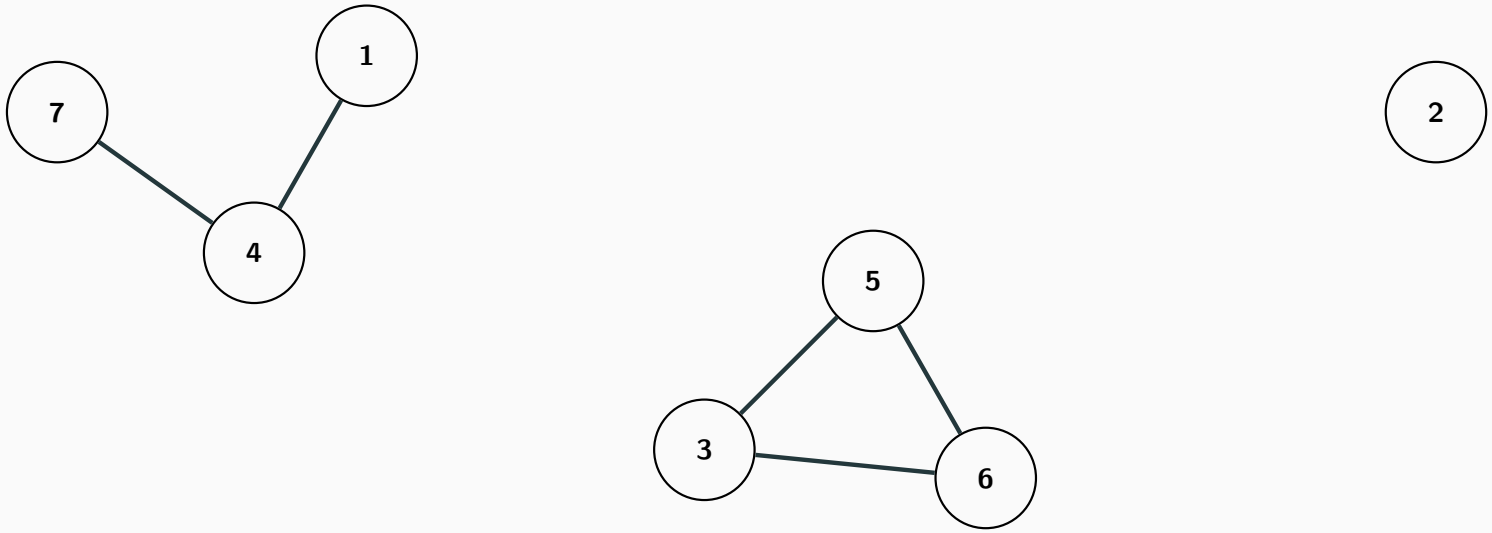
Aplicaciones

- Union-Find mantiene una colección de conjuntos disjuntos
- ¿Cuándo nos importa saber si dos elementos están en el mismo conjunto?
- El ejemplo más común es en grafos

Conjuntos disjuntos en grafos

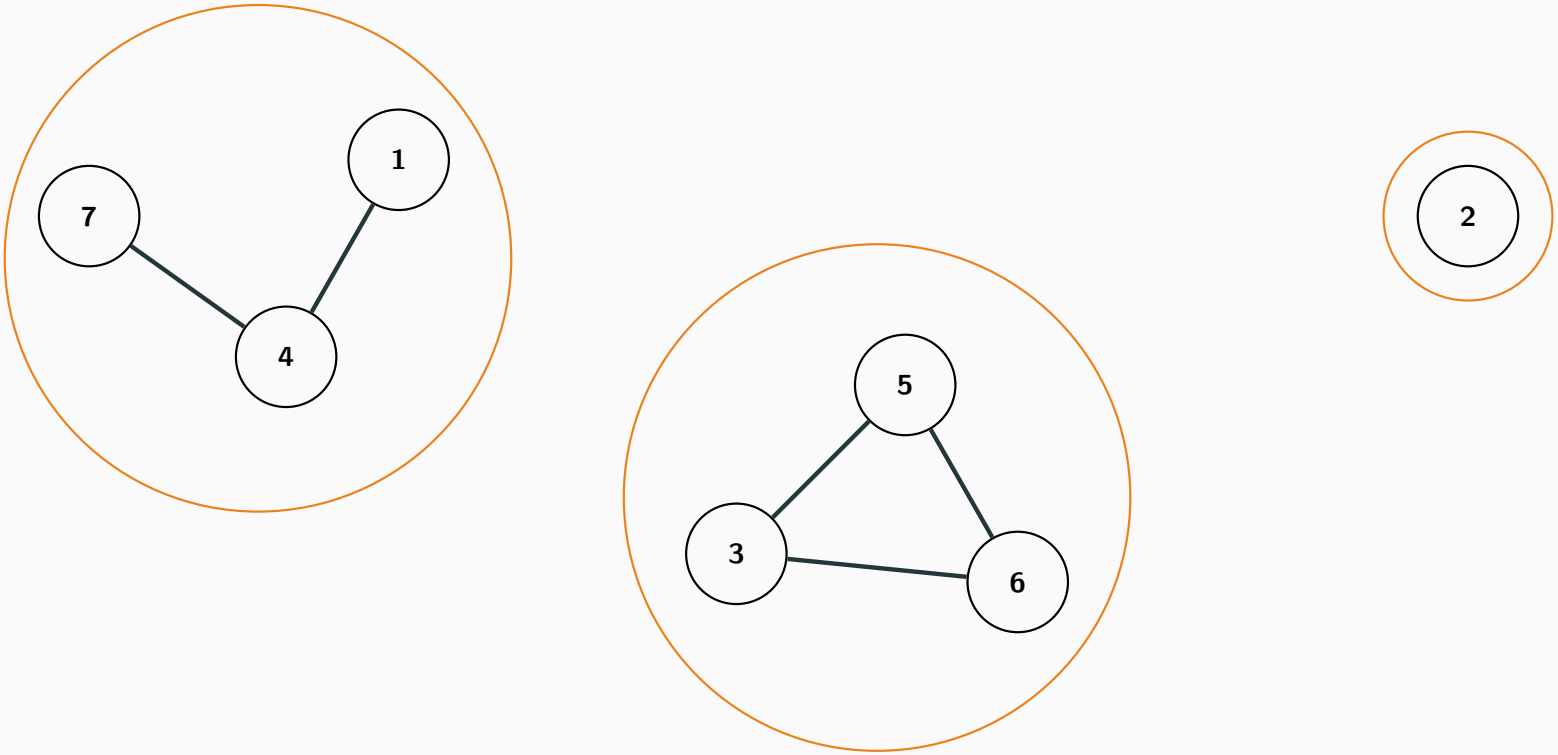


Conjuntos disjuntos en grafos



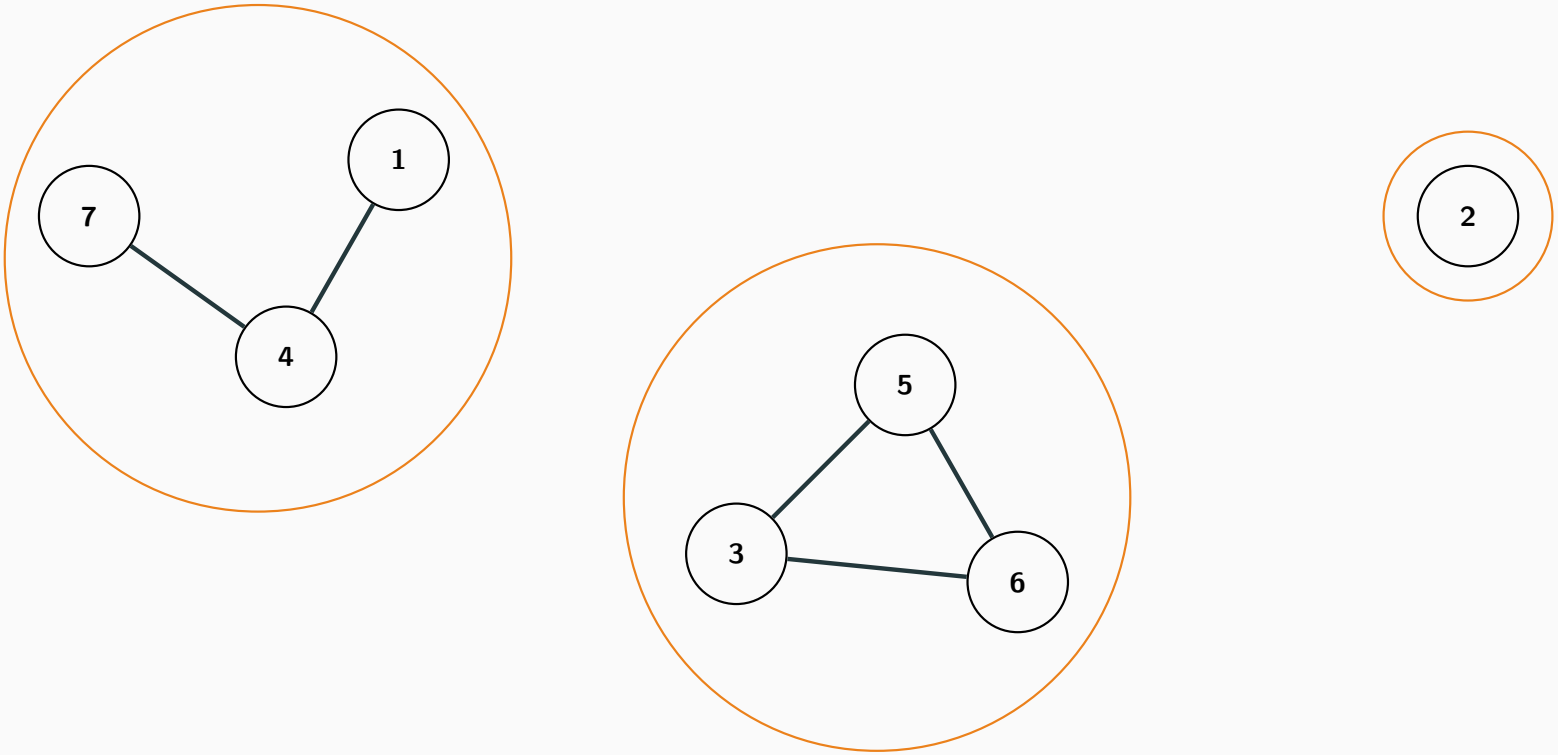
- *elementos* $= \{1, 2, 3, 4, 5, 6, 7\}$

Conjuntos disjuntos en grafos



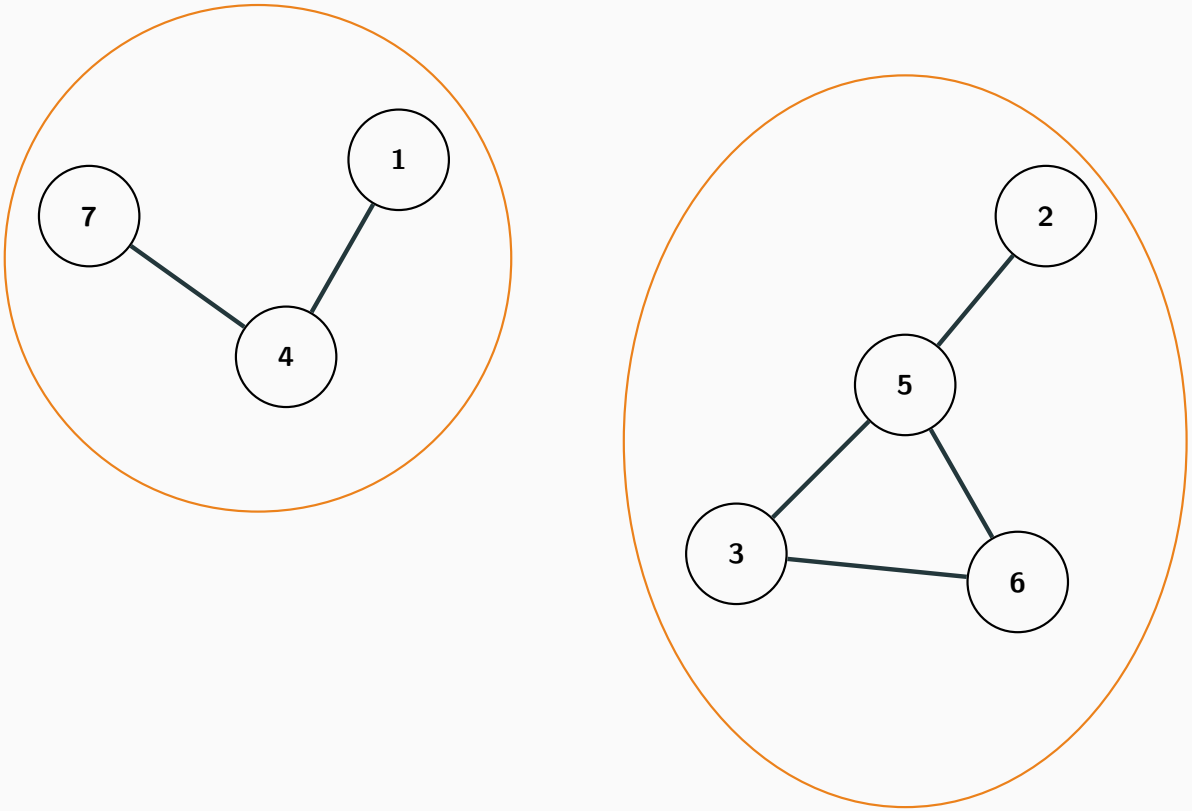
- *elementos* $= \{1, 2, 3, 4, 5, 6, 7\}$
- *colecciones* $= \{1, 4, 7\}, \{2\}, \{3, 5, 6\}$

Conjuntos disjuntos en grafos



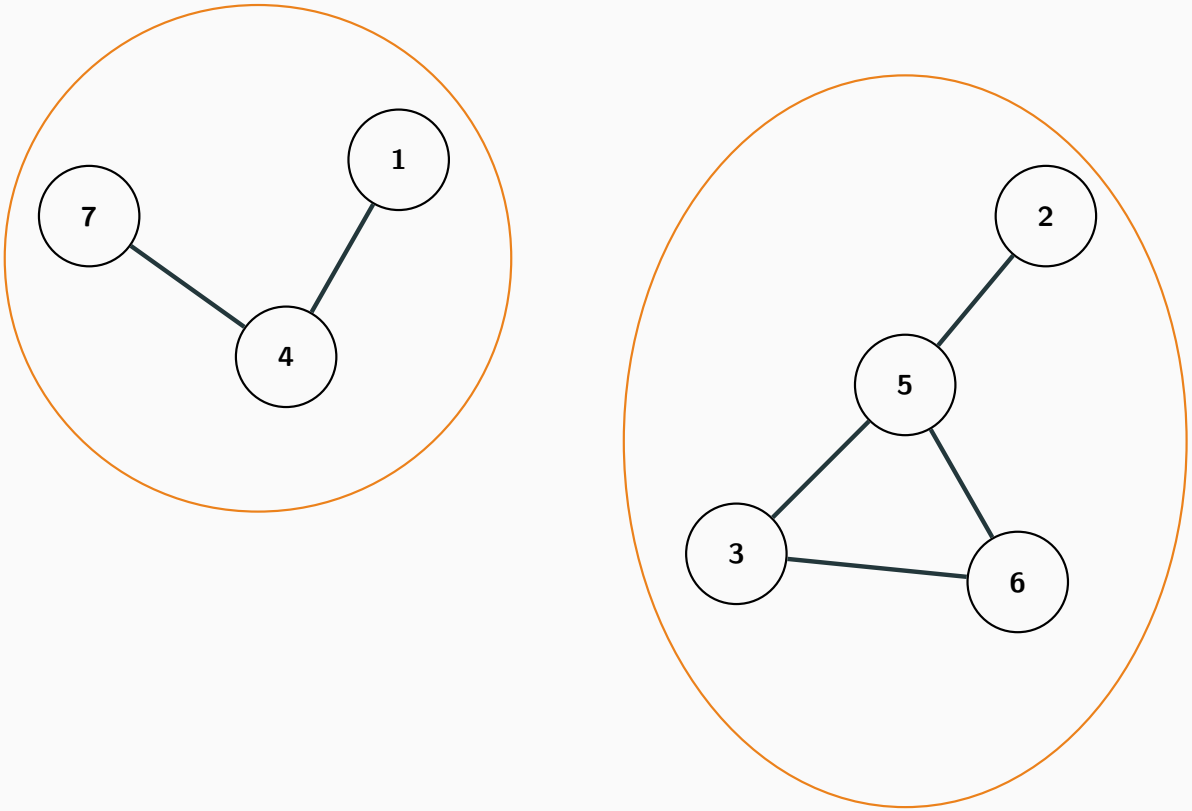
- *elementos* $= \{1, 2, 3, 4, 5, 6, 7\}$
- *colecciones* $= \{1, 4, 7\}, \{2\}, \{3, 5, 6\}$
- `union(2, 5)`

Conjuntos disjuntos en grafos



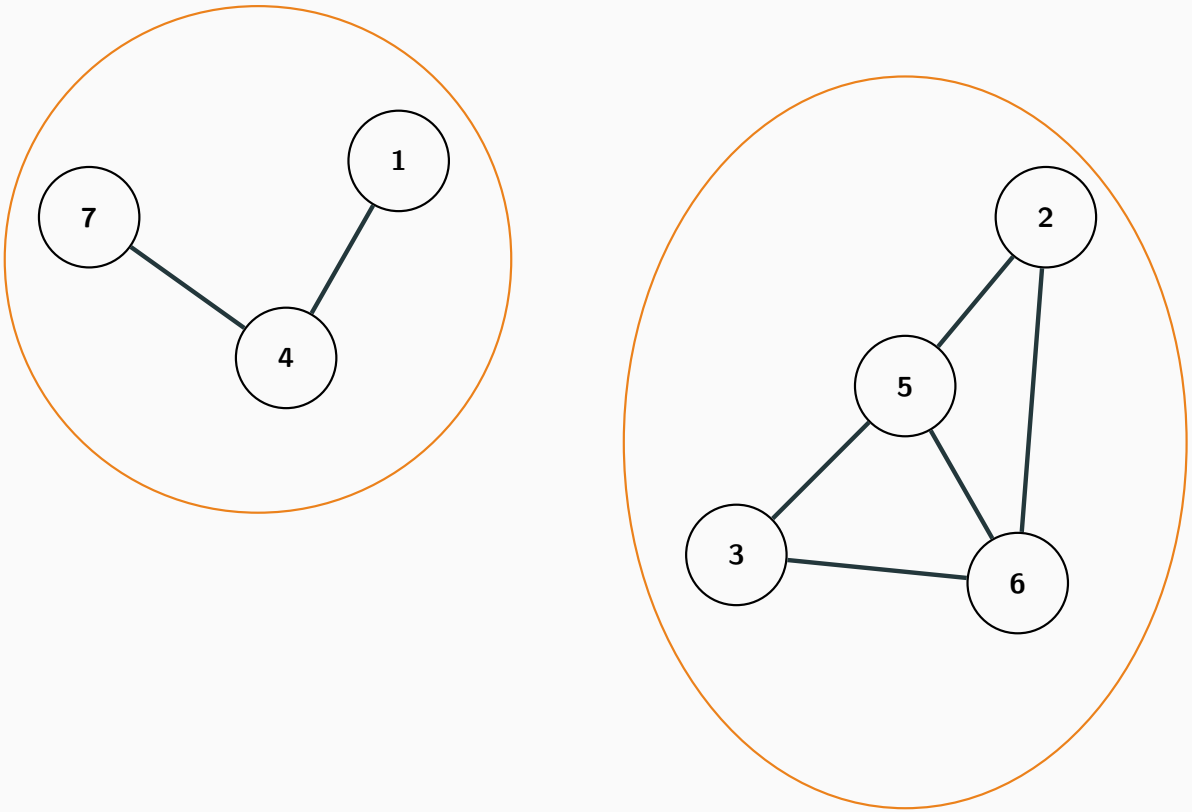
- *elementos* $= \{1, 2, 3, 4, 5, 6, 7\}$
- *colecciones* $= \{1, 4, 7\}, \{2, 3, 5, 6\}$

Conjuntos disjuntos en grafos



- *elementos* $= \{1, 2, 3, 4, 5, 6, 7\}$
- *colecciones* $= \{1, 4, 7\}, \{2, 3, 5, 6\}$
- `union(6, 2)`

Conjuntos disjuntos en grafos



- *elementos* $= \{1, 2, 3, 4, 5, 6, 7\}$
- *colecciones* $= \{1, 4, 7\}, \{2, 3, 5, 6\}$

Problema de ejemplo: Where's My Internet??

- https://open.kattis.com/problems/wheresmyinternet

Problema de ejemplo: Where's My Internet??

Descripcion del problema
En una ciudad hay n casas y algunas no tienen conexión a internet.

Las casas están numeradas de 1 a n. La casa 1 ya tiene conexión a internet a través de un cable que va a otra ciudad. Una casa tiene conexión a internet si tiene un cable a otra casa que ya está conectada a internet.

Dada una lista de pares de casas que ya están conectadas, indicar los números de las casas que no tienen conexión a internet.

Problema de ejemplo: Where's My Internet??

Input
La primera línea contiene dos enteros n y m, el número de casas y el número de pares de casas que ya están conectadas. Las siguientes m líneas contienen dos enteros a_i, b_i indicando que las casas a_i y b_i ya están conectadas.

Output
Si todas las casas tienen conexión a internet, imprimir "`Connected`".

Si no, imprimir una línea por cada casa que no tiene conexión a internet, en orden ascendente.

Problema de ejemplo: Where's My Internet??

Input de ejemplo	Output de ejemplo
6 4 1 2 2 3 3 4 5 6	5 6

Problema de ejemplo: Where's My Internet??

```
int p[MAXN];
int find(int x) {...}
void unite(int x, int y) {...}

int main() {
  int n, m;
  cin >> n >> m;
  for (int i = 1; i <= n; ++i) p[i] = i;
  for (int i = 0; i < m; ++i) {
    int a, b;
    cin >> a >> b;
    unite(a, b);
  }
  // encontrar casas no conectadas
}
```

Problema de ejemplo: Where's My Internet??

```
// encontrar casas no conectadas
bool connected = true;
for (int i = 1; i <= n; ++i) {
  if (find(1) != find(i)) {
    cout << i << endl;
    connected = false;
  }
}
if (connected) {
  cout << "Connected" << endl;
}
```

Almacenar información sobre el conjunto

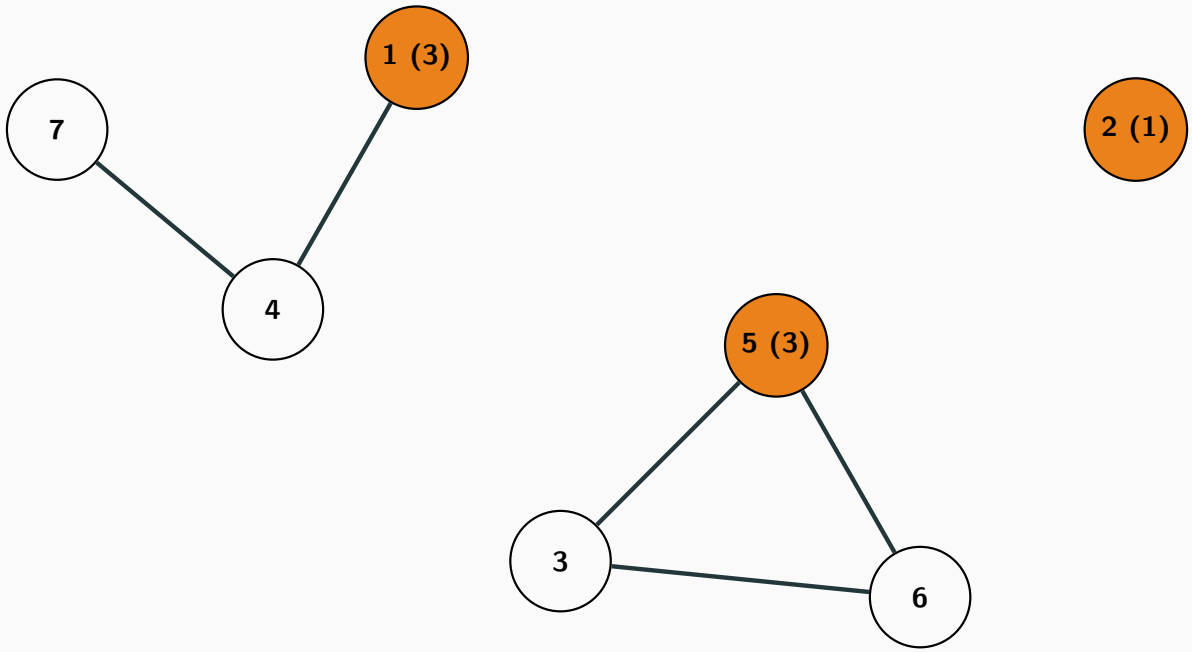
- A veces no solo queremos saber si dos elementos están en el mismo conjunto
- ¿Cuántos elementos hay en el conjunto?
- ¿Cuál es el elemento más pequeño del conjunto?

Almacenar información sobre el conjunto

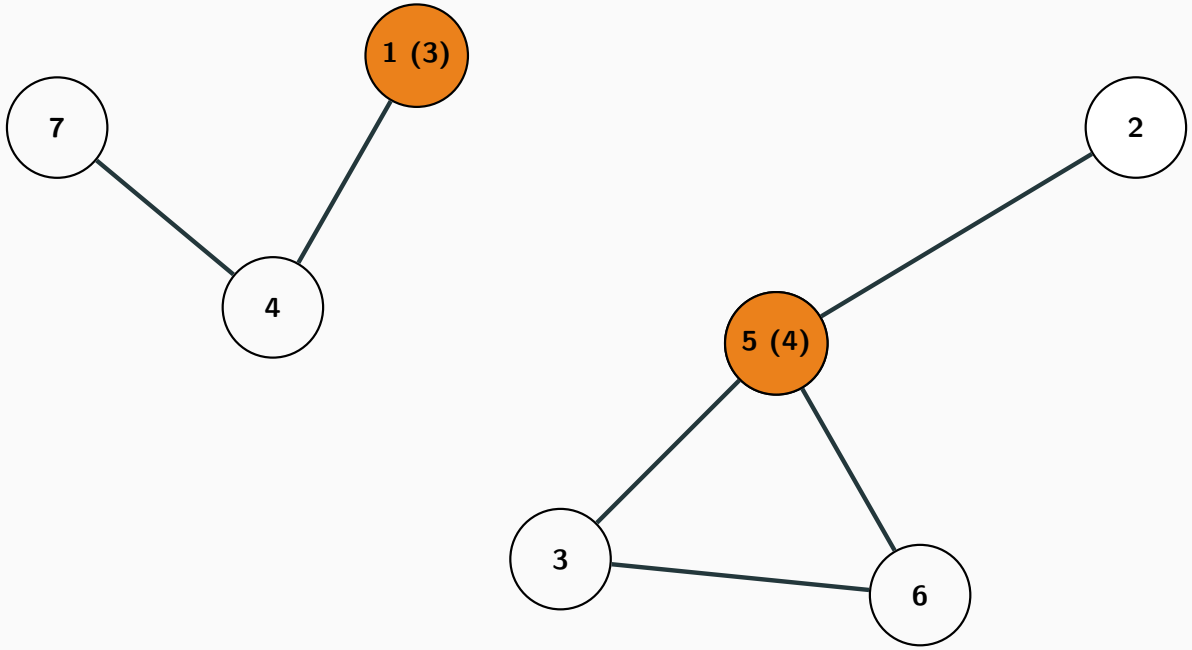
- A veces no solo queremos saber si dos elementos están en el mismo conjunto
- ¿Cuántos elementos hay en el conjunto?
- ¿Cuál es el elemento más pequeño del conjunto?

- Podemos almacenar esta información en el elemento representativo del conjunto
- Debemos tener cuidado al actualizar esta información cuando unimos dos conjuntos

Almacenar información sobre el conjunto

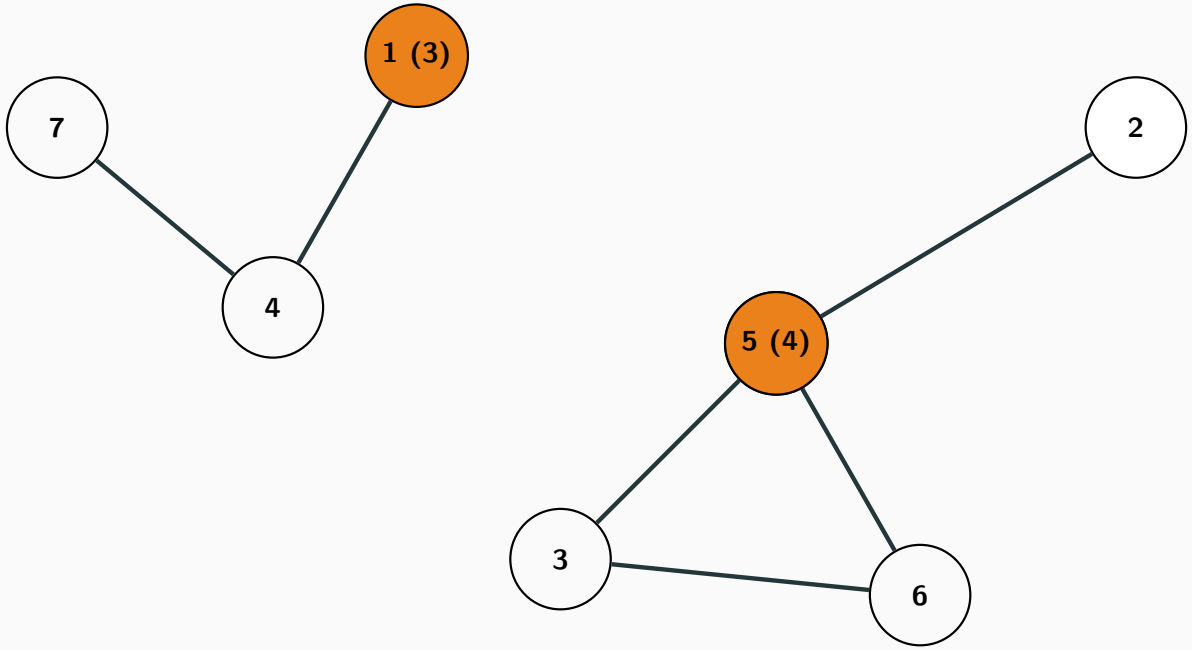


Almacenar información sobre el conjunto



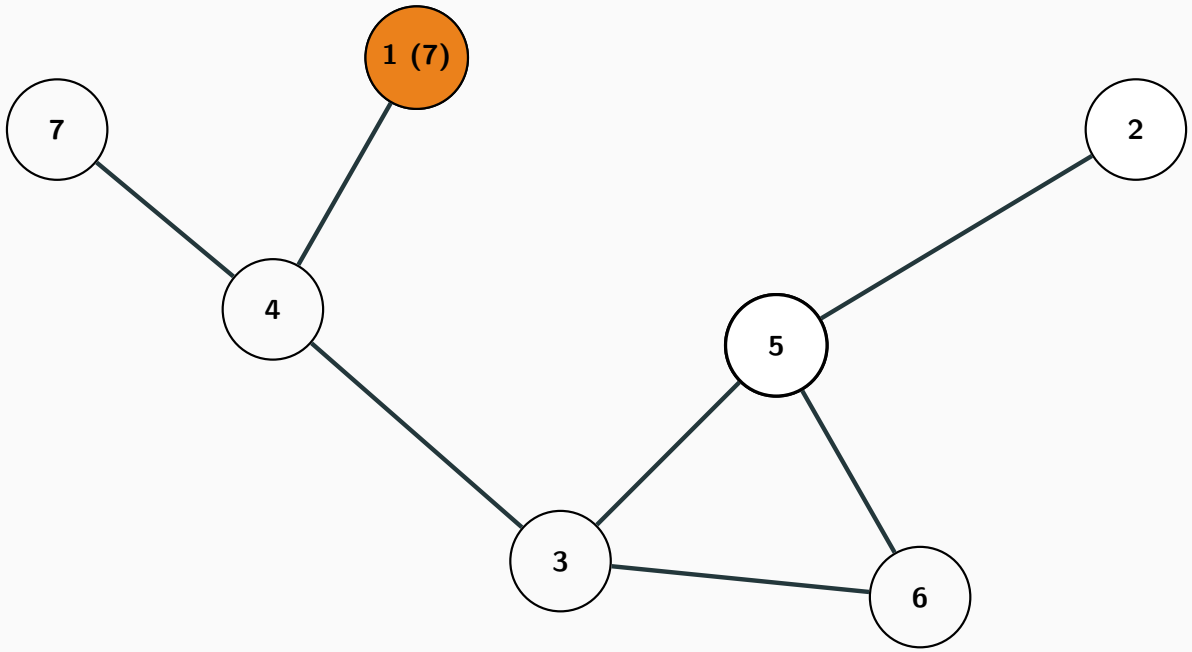
- `union(2, 5)`

Almacenar información sobre el conjunto



- `union(2, 5)`
- `union(3, 4)`

Almacenar información sobre el conjunto



- `union(2, 5)`
- `union(3, 4)`

Almacenar información sobre el conjunto

```
int p[MAXN], sz[MAXN];
int find(int x) {...}
void unite(int x, int y) {
  x = find(x);
  y = find(y);
  // para no actualizar con información duplicada
  if (x == y) return;

  p[y] = x;
  sz[x] += sz[y];
}
// sz[find(x)] <- tamaño del conjunto de x
```

Almacenar información sobre el conjunto

```
int p[MAXN], sz[MAXN];
int find(int x) {...}
void unite(int x, int y) {
  x = find(x);
  y = find(y);
  // para no actualizar con información duplicada
  if (x == y) return;
  if (sz[x] < sz[y]) swap(x, y); // O(1) en practica
  p[y] = x;
  sz[x] += sz[y];
}
// sz[find(x)] <- tamaño del conjunto de x
```