Lecture 21: Disjoint Sets

Agenda:

- Introduction
- Application — graph (connected) components
- 1^{st} implementation — an array of representatives

Reading:

- Textbook pages 498 – 505

Overview:

- An abstract data type (ADT)
- Analysis: via sequences of operations
- Implementation: array of representatives
- Application: graph components
- Implementation: forest of rooted trees (next lecture)
  - basic implementation
  - improvement: union by rank
  - improvement: compressed find

Disjoint sets:

- An abstract data type
- It maintains: pairwise disjoint sets
- One element of each set is the representative
- Operations:
  - `MakeSet`(x) — make x itself into a set and use x to be the representative: $S_x \leftarrow \{x\}$
  - `Find`(x) — return the representative of the set $S_{f(x)}$ containing x, which is $f(x)$
  - `Union`(x, y) — find the sets containing x and y, respectively, and union them into a new set with representative z: $S_z \leftarrow S_{f(x)} \cup S_{f(y)}$
- Analysis over a sequence of operations on n elements
  - `MakeSet`(1), `MakeSet`(2), ..., `MakeSet`(n)
  - all the `Union` and `Find` operations
  - m — number of operations
  - running time for all $m = n + |U| + |F|$ operations?

Simplest DS implementation

— array R of representatives:

- $R(x)$ — the representative of the set containing x
  - $\Theta(n)$ space
  - how to describe a set now:
    elements with the same representative are in a same set
  - report a set in $\Theta(n)$ time

- `procedure MakeSet(`x`) **initialize representative for` x

  $R(x) \leftarrow x$

- `procedure Find(`x`)    **representative of` x

  `return` $R(x)$

- `procedure Union(`x, y`)`

```
rx ← R(x)
ry ← R(y)
for j ← 1 to n do
    if R(j) = ry then            if R(j) = rx then
        R(j) ← rx                    R(j) ← ry
```

An example:

sets at start	{1}	{2}	{3}	{4}	{5}	{6}	{7}	{8}
index	1	2	3	4	5	6	7	8
representative	1	2	3	4	5	6	7	8
`Union`(1, 4)	1	2	3	**1**	5	6	7	8
`Find`(1)	1							
`Find`(4)				1				
`Union`(2, 3)	1	2	**2**	1	5	6	7	8
`Union`(5, 1)	**5**	2	2	**5**	5	6	7	8
`Union`(1, 8)	5	2	2	5	5	6	7	5
`Union`(6, 5)	**6**	2	2	**6**	**6**	6	7	**6**
`Find`(6)						6		
`Find`(3)			2					

sets at finish {1, 4, 5, 6, 8}, {2, 3}, {7}

A DSUF application

— finding connected components of a graph:

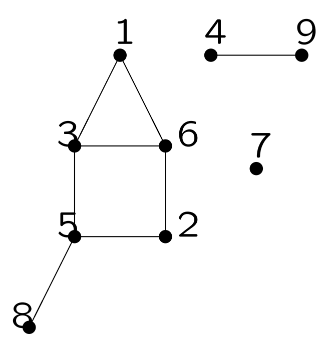
- procedure ConnectedComponents(G)

```
for each vertex v ∈ V(G) do
    MakeSet(v)
for each edge (x,y) ∈ E(G) do
    if not SameComponent(x,y,G) then
        Union(x,y)
```

- procedure SameComponent(x, y, G)

```
return Find(x) = Find(y)
```

- An example:



$V = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$E = \{\{1, 3\}, \{1, 6\}, \{2, 5\}, \{2, 6\}, \{3, 5\}, \{3, 6\}, \{4, 9\}, \{5, 8\}\}$

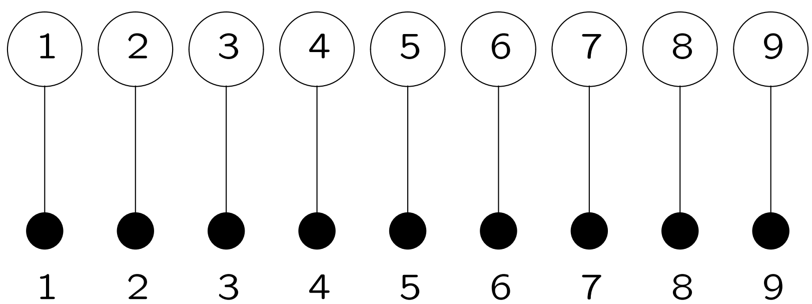
Finding connected components of a graph:

- Graph $G = (V, E)$:

  $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

  $E = \{\{1, 3\}, \{1, 6\}, \{2, 5\}, \{2, 6\}, \{3, 5\}, \{3, 6\}, \{4, 9\}, \{5, 8\}\}$

- After `MakeSets`:



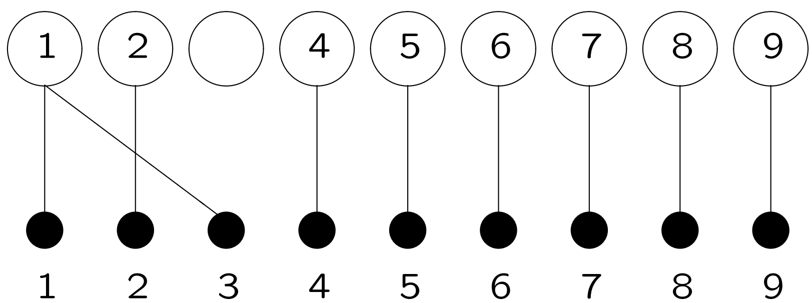
Finding connected components of a graph:

- Graph $G = (V, E)$:

  $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

  $E = \{\{1, 3\}, \{1, 6\}, \{2, 5\}, \{2, 6\}, \{3, 5\}, \{3, 6\}, \{4, 9\}, \{5, 8\}\}$

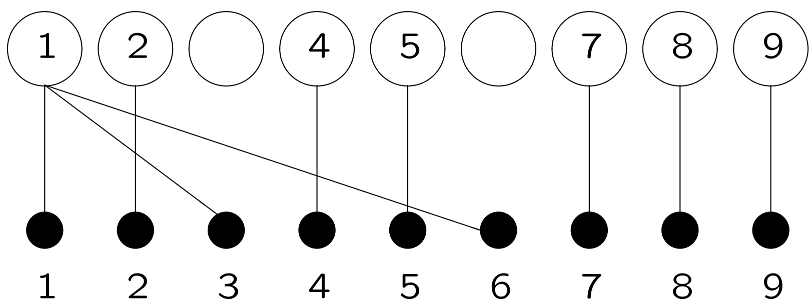
- After considering edge $(1, 3)$:



Finding connected components of a graph:

- Graph $G = (V, E)$:
  $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
  $E = \{\{1, 3\}, \{1, 6\}, \{2, 5\}, \{2, 6\}, \{3, 5\}, \{3, 6\}, \{4, 9\}, \{5, 8\}\}$

- After considering edge $(1, 6)$:



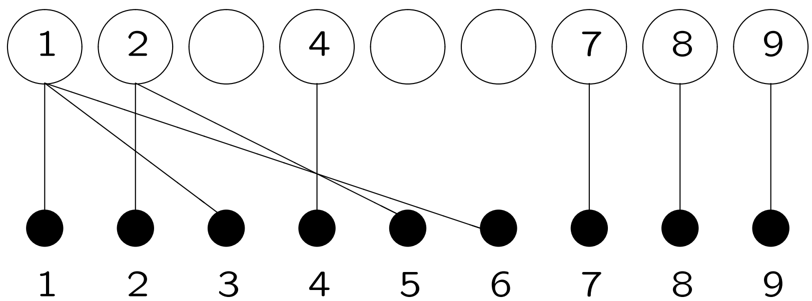
Finding connected components of a graph:

- Graph $G = (V, E)$:

  $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

  $E = \{\{1, 3\}, \{1, 6\}, \{2, 5\}, \{2, 6\}, \{3, 5\}, \{3, 6\}, \{4, 9\}, \{5, 8\}\}$

- After considering edge $(2, 5)$:



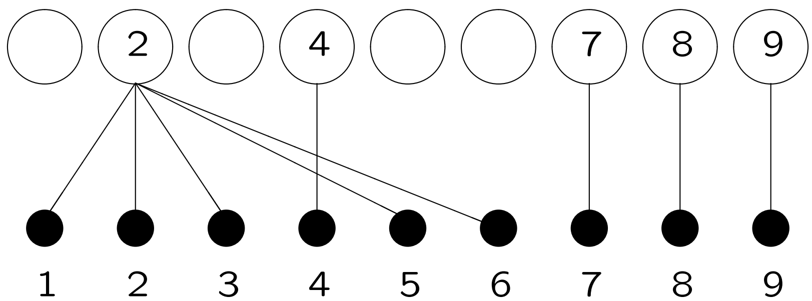
Finding connected components of a graph:

- Graph $G = (V, E)$:

  $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

  $E = \{\{1, 3\}, \{1, 6\}, \{2, 5\}, \{2, 6\}, \{3, 5\}, \{3, 6\}, \{4, 9\}, \{5, 8\}\}$

- After considering edge $(2, 6)$:



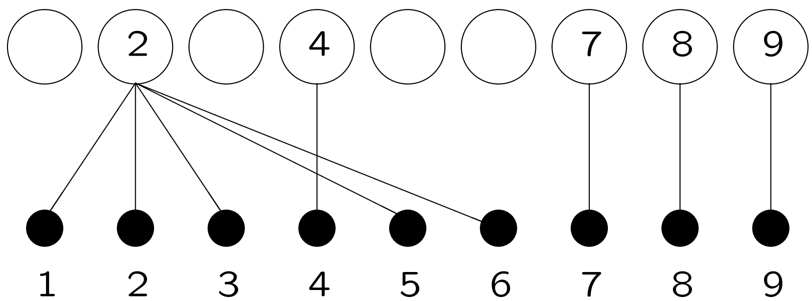
Finding connected components of a graph:

- Graph $G = (V, E)$:

  $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

  $E = \{\{1, 3\}, \{1, 6\}, \{2, 5\}, \{2, 6\}, \{3, 5\}, \{3, 6\}, \{4, 9\}, \{5, 8\}\}$

- After considering edges (3, 5) and (3, 6)



(no change since 3, 5, 6 are already in a same component):

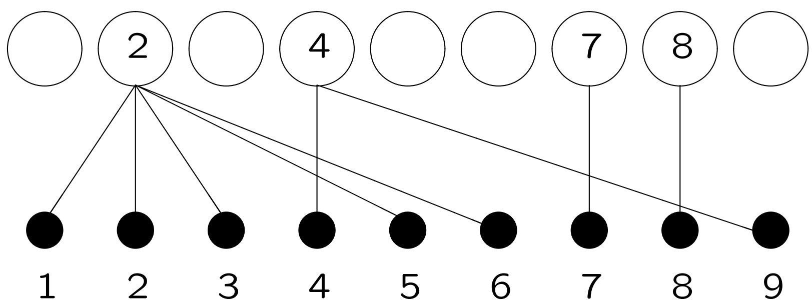
Finding connected components of a graph:

- Graph $G = (V, E)$:

  $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

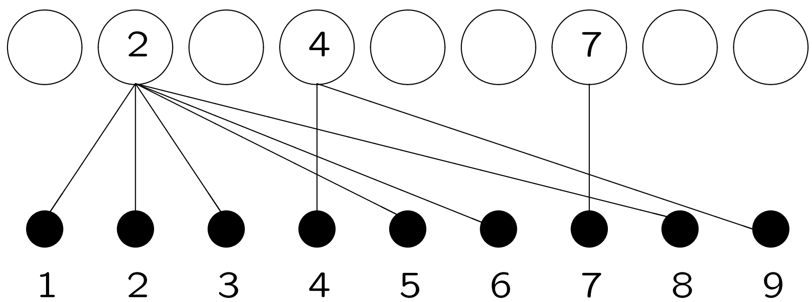
  $E = \{\{1, 3\}, \{1, 6\}, \{2, 5\}, \{2, 6\}, \{3, 5\}, \{3, 6\}, \{4, 9\}, \{5, 8\}\}$

- After considering edge $(4, 9)$:

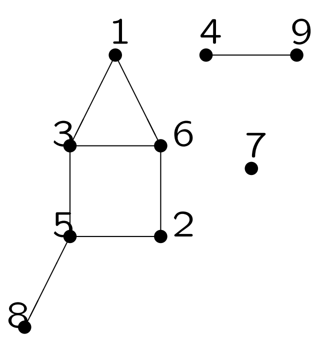


Finding connected components of a graph:

- Graph $G = (V, E)$:
  $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
  $E = \{\{1, 3\}, \{1, 6\}, \{2, 5\}, \{2, 6\}, \{3, 5\}, \{3, 6\}, \{4, 9\}, \{5, 8\}\}$
- After considering edge $(5, 8)$:



- Sets at finish:
  {1, 2, 3, 5, 6, 8}, {4, 9}, and {7}
- Therefore, there are 3 connected components



Analysis of the simplest implementation:

- n `MakeSet` and $(m-n)$ `Union/Find`
- Each `MakeSet` — $\Theta(1)$ time
- Each `Find` — $\Theta(1)$ time

  since we record for every element x its representative as $R(x)$
- Each `Union` — $\Theta(n)$ time

  since we need to check for every element in order to update its representative and for every element it takes $\Theta(1)$ time
- Worst case:

  $\Theta(n + (m-n)n) = \Theta(mn)$ time (assuming $m >> n$)
- On average, $\Theta(n)$ per operation

  amortized running time analysis

Have you understood the lecture contents?

well	ok	not-at-all	topic
☐	☐	☐	disjoint sets?
☐	☐	☐	3 operations
☐	☐	☐	array of representatives
☐	☐	☐	finding connected components
☐	☐	☐	running time analysis