CSCI-UA 480.4: APS
Algorithmic Problem Solving

Disjoint Set / Union-Find Data Structures

Instructor: Joanna Klukowska

created based on materials for this class by
Bowen Yu and materials shared by the authors of
the textbook Steven and Felix Halim

Disjoint Set / Union-Find

- tracks a set of elements partitioned into disjoint subsets
    - disjoint?

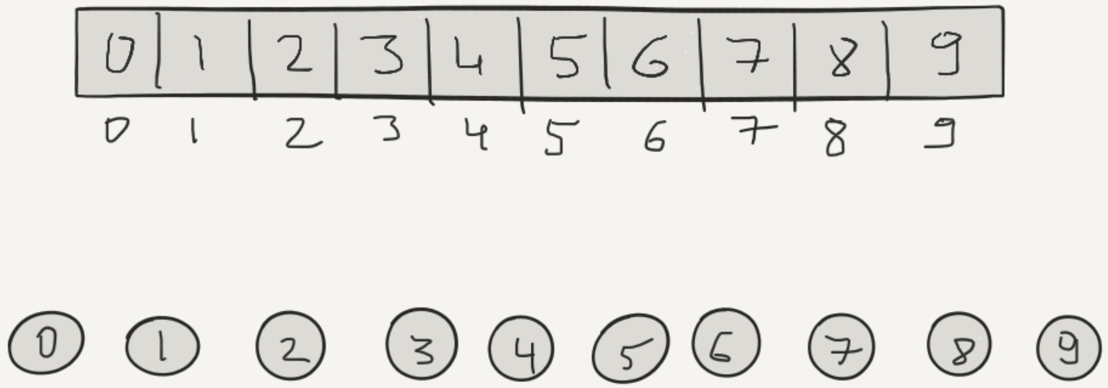
Disjoint Set / Union-Find

- tracks a set of elements partitioned into disjoint subsets
    - disjoint? - non overlapping, no elements in common

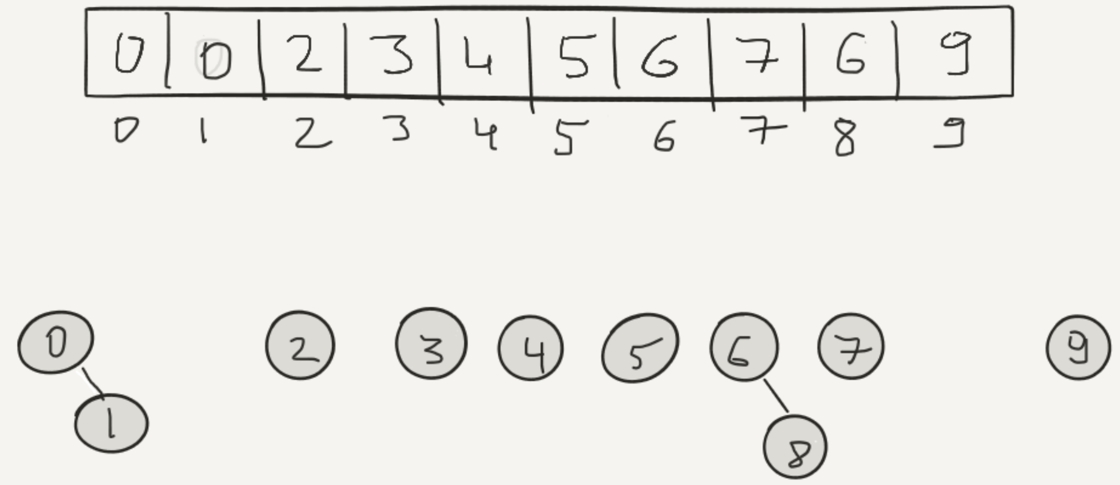
Disjoint Set / Union-Find

- tracks a set of elements partitioned into disjoint subsets
  - disjoint? - non overlapping, no elements in common
- performance: near constant time (bound by inverse Ackerman function) for
  - - determine which set an element belongs to
  - - determine if x and y belong to the same set
  - - merge two sets of which x and y are members
- the above performance assumes
  - path compression
  - merging by rank

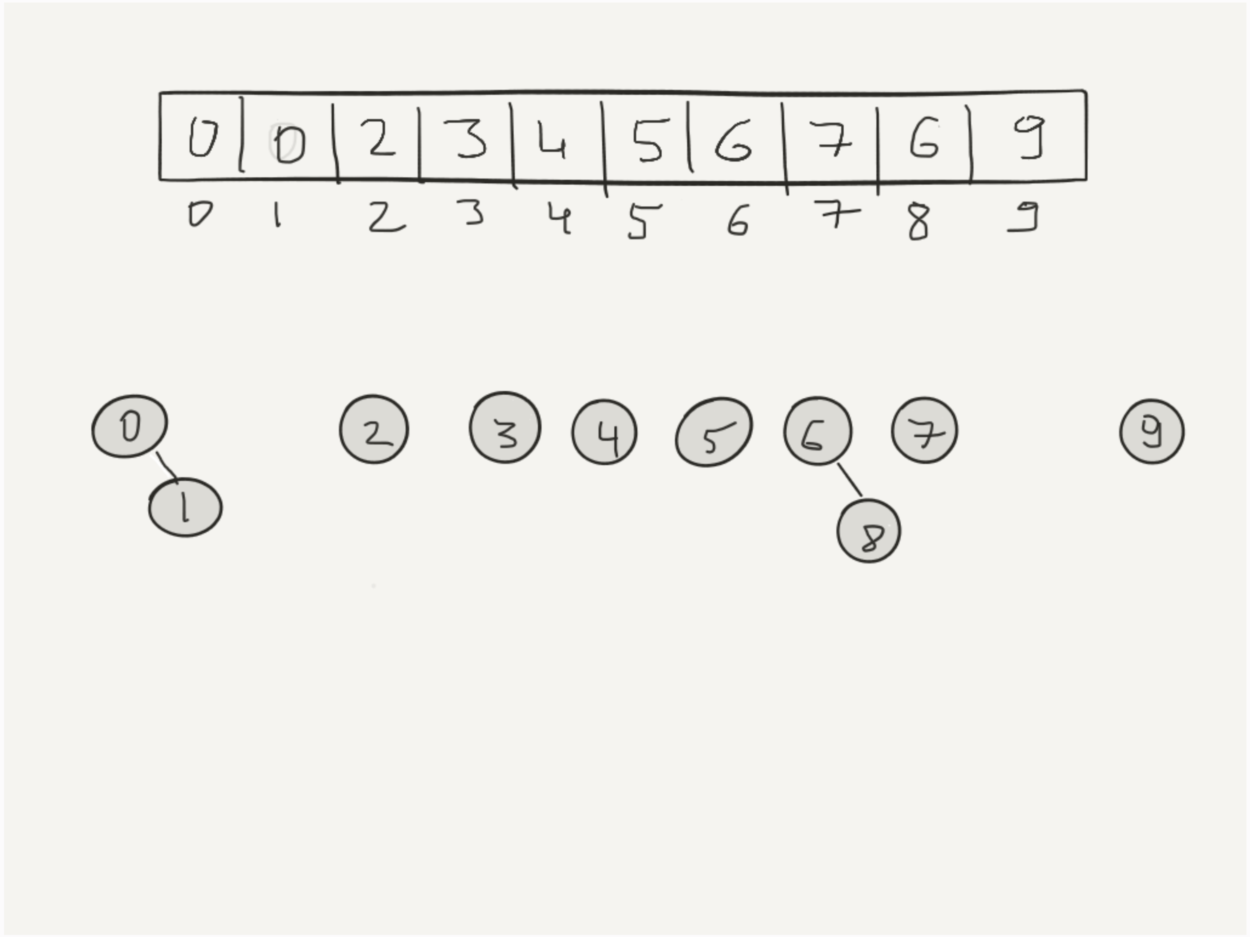
Example



Example

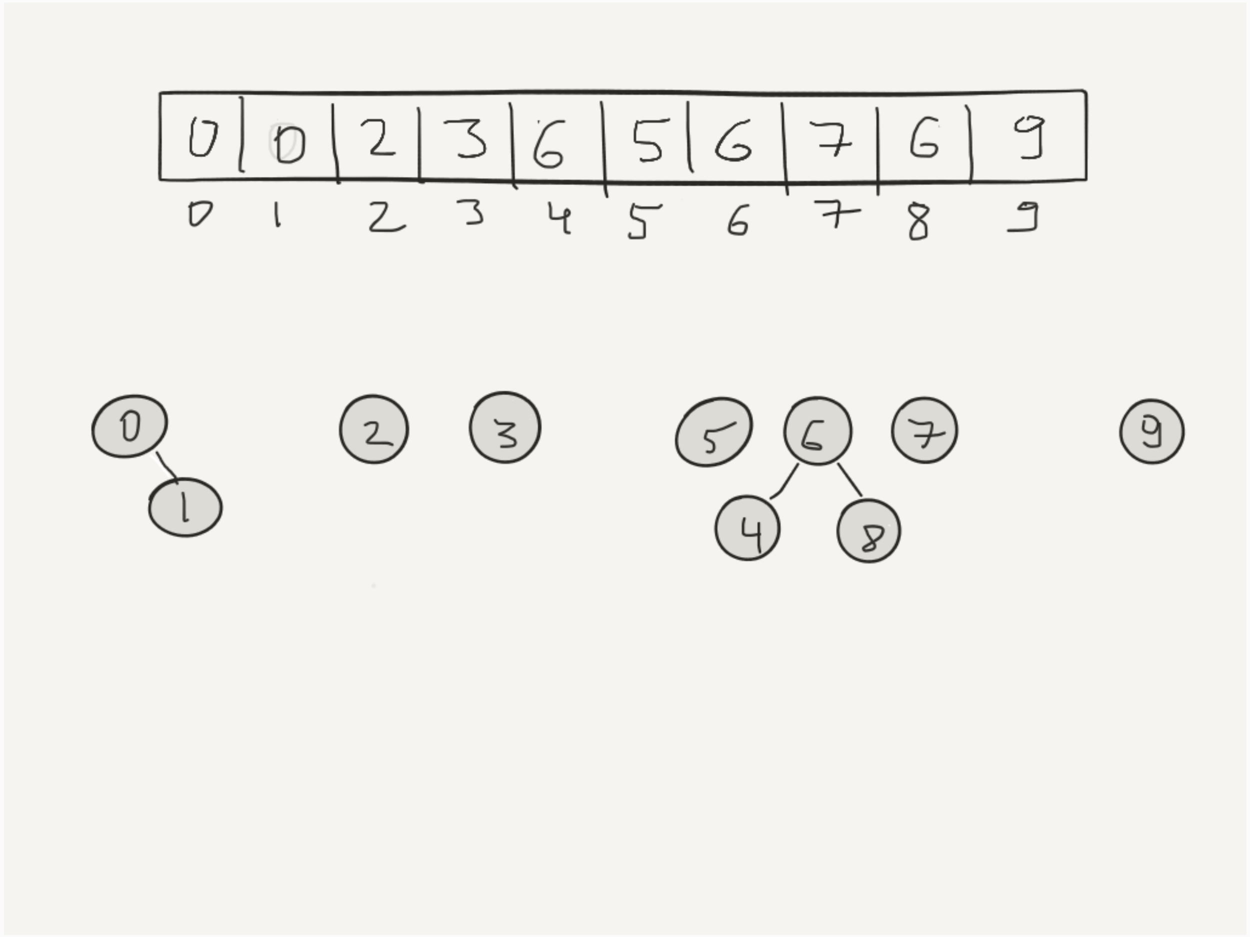


Example

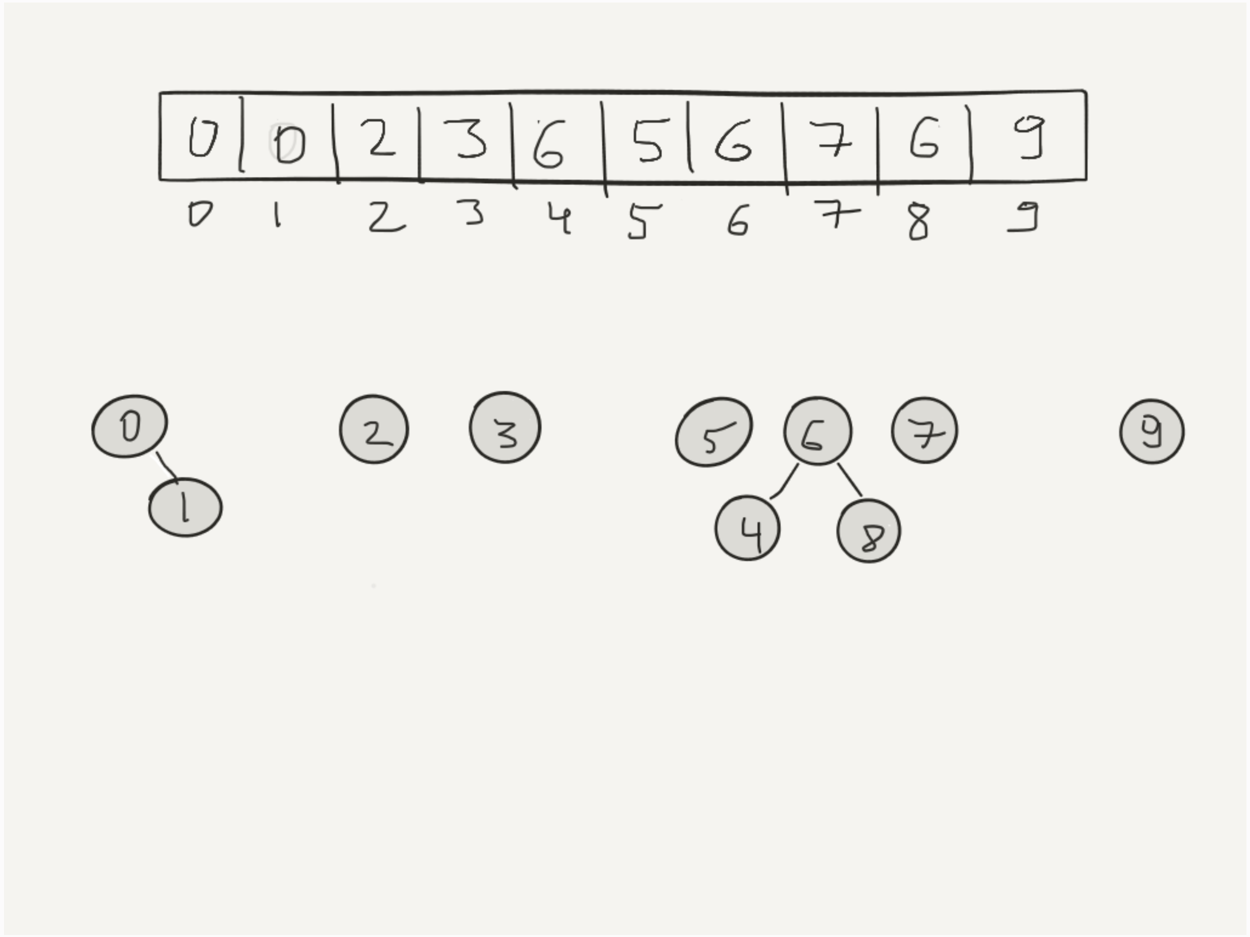


union(0,1)
union(6,8)

Example

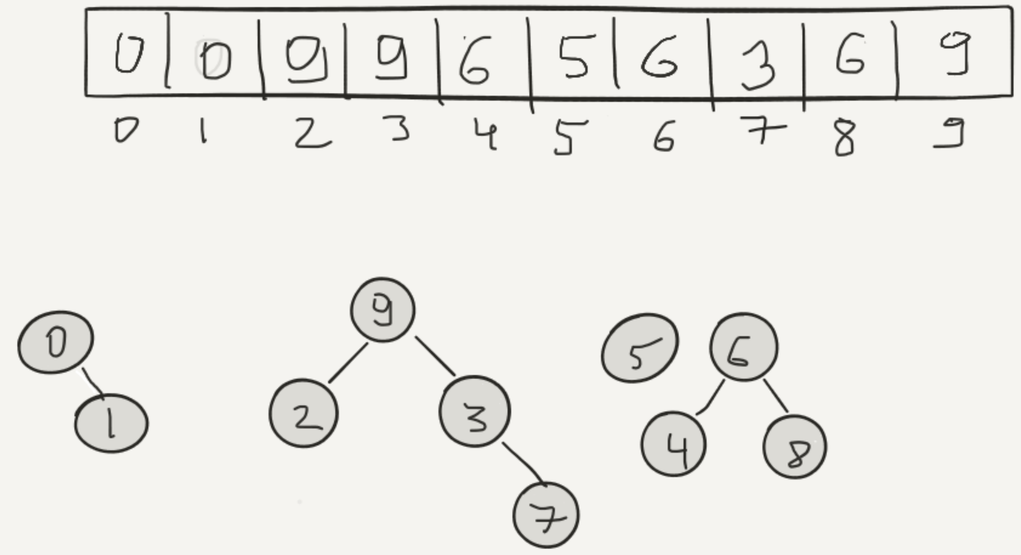


Example

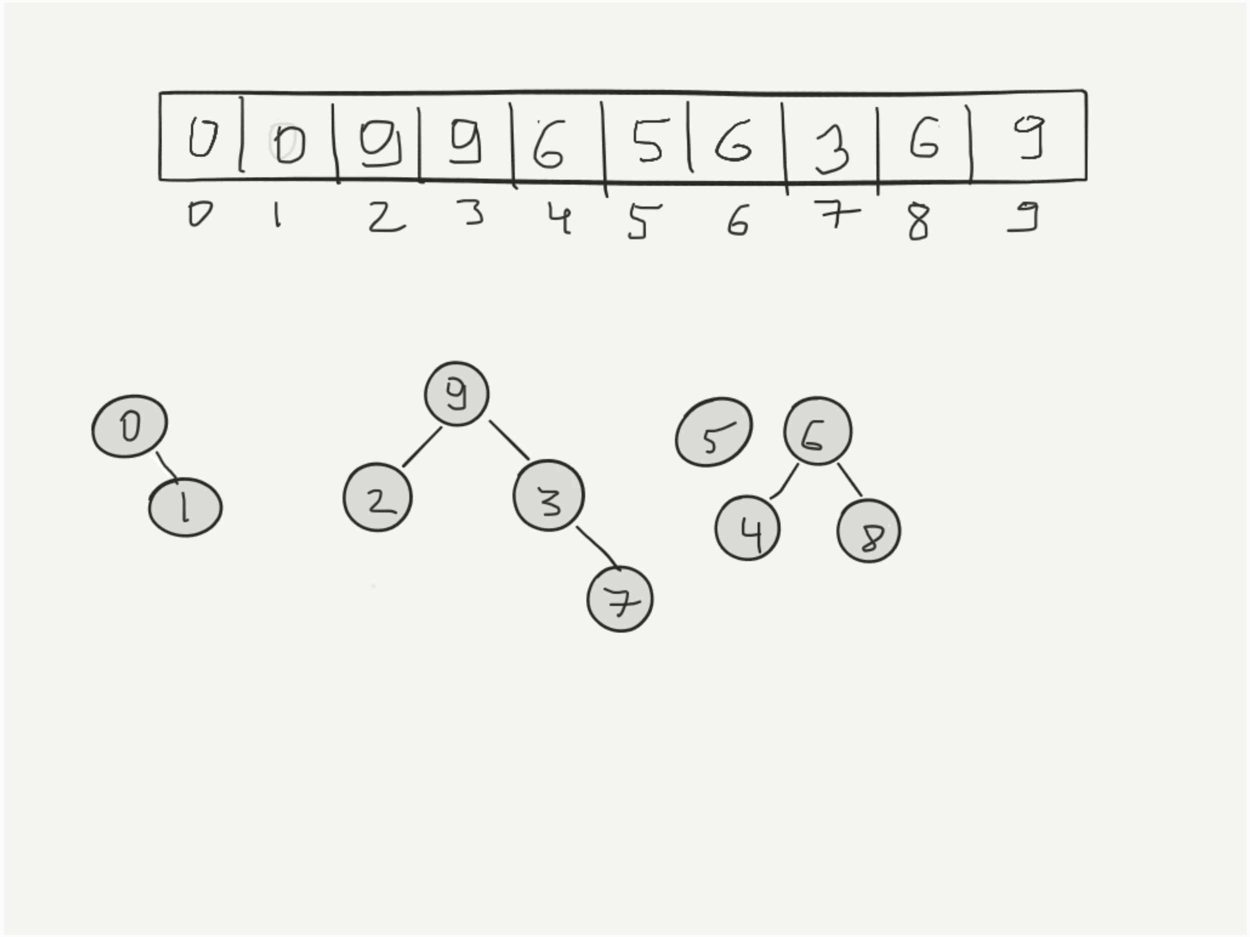


union(4,6)
(uses merging by rank: the *tree* with higher approximate height becomes the root)

Example

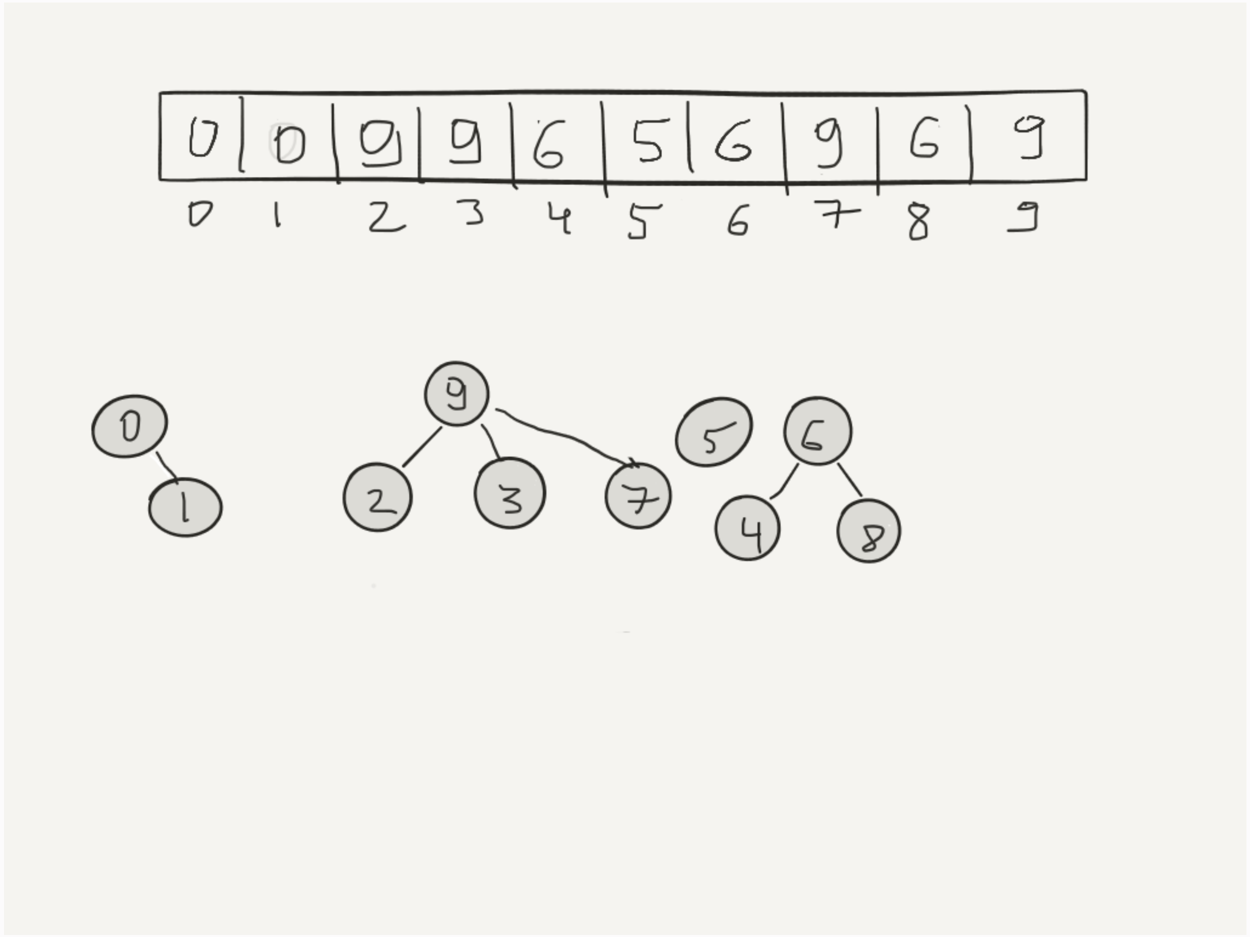


Example

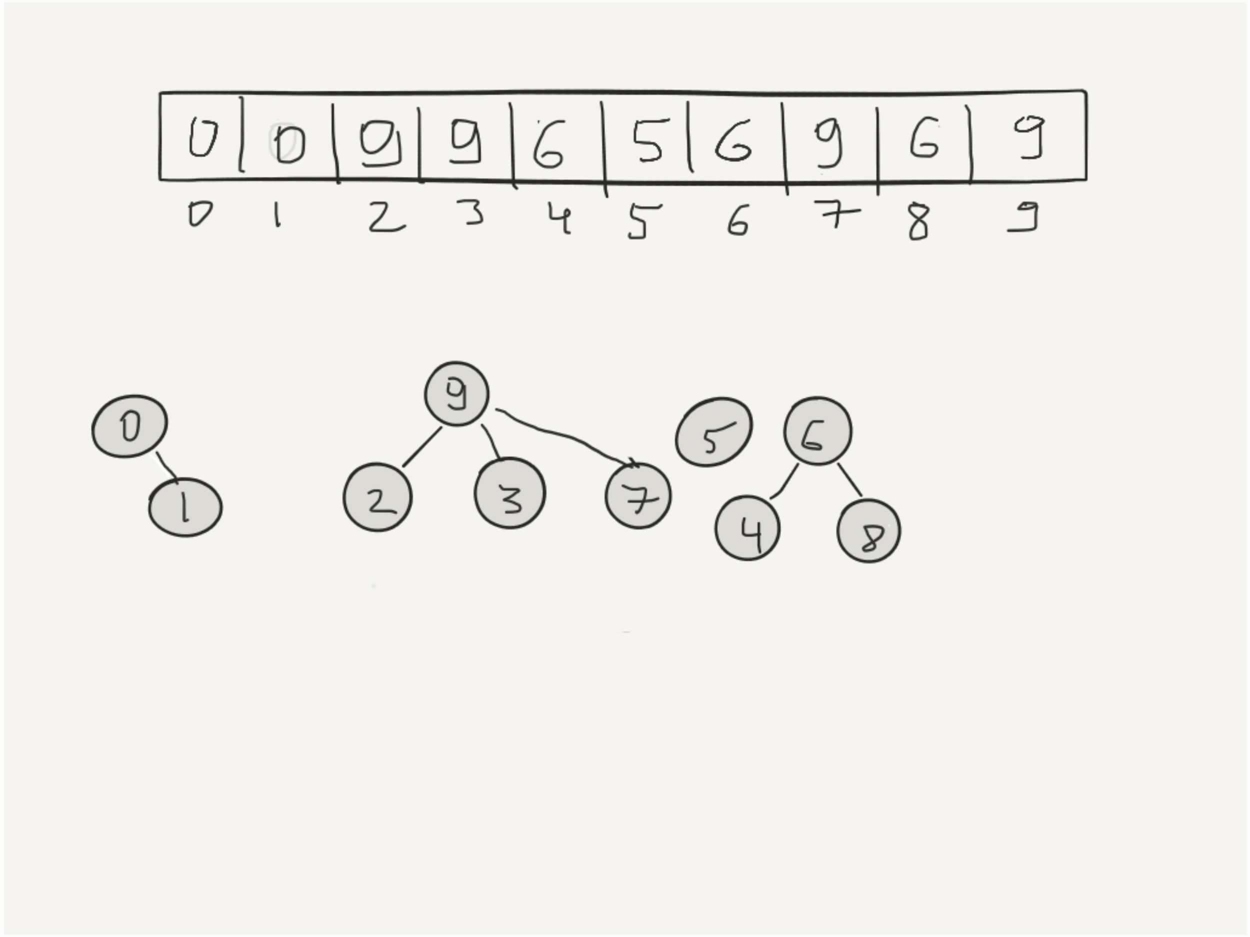


union(3,7)
union(2,9)
union(9,3)

Example



Example



find(7)
(uses path compression)

Visualizations:

- USFCA visualization
- VisuAlgo

Challenge

There are N students (2 <= N <= 10^5). Each student belongs to exactly one student club. We do not know what clubs the students belong to, but we do have information about pairs of students who belong to the same club. This information is presented in the form of pairs: (a,b) - this indicates that the students a and b belong to the same student club. We have P (1 <= P <= 10^5) such pairs. The pairs are not guaranteed to be unique.

Find the number of student clubs on campus.

Example:

n = 5, (so at most 5 clubs)

list of pairs:

Challenge

There are N students (2 <= N <= 10^5). Each student belongs to exactly one student club. We do not know what clubs the students belong to, but we do have information about pairs of students who belong to the same club. This information is presented in the form of pairs: (a,b) - this indicates that the students a and b belong to the same student club. We have P (1 <= P <= 10^5) such pairs. The pairs are not guaranteed to be unique.

Find the number of student clubs on campus.

Example:

n = 5, (so at most 5 clubs)

list of pairs:

Answer There are two student clubs.

Challenge: Allies and Enemies

There are N countries (2 <= N <= 10^5). Any pair of countries are either allies or enemies.

Instructions to be implemented:

- , x and y are allies
- , x and y are enemies
- , queries about being allies
- , queries about being enemies

(if an instruction or conflicts with existing information, it should be ignored and a conflict should be reported)

We have the following relations:

- if and then
- if and then
- if and then
- if and then

Challenge: Allies and Enemies

Example

n = 5, (countries numbered 1 - 5)

instructions	response
	false, no info yet
	false, no info yet
	OK (i.e., no conflict)
	OK (i.e., no conflict)
	conflict
	OK (i.e., no conflict)
	true
	OK (i.e., no conflict)
	false
	true
	conflict

Challenge: Allies and Enemies

Solution

Represent each country with two values: x and x'

The required instructions can be now implemented using disjoint set at follows:

- ally(x,y)
  - if sameSet(x,y'), then conflict
  - otherwise union(x,y) and union(x',y')
- enemy(x,y)
  - if sameSet(x,y), then conflict
  - otherwise union(x,y`) and union(x',y)
- isAlly(x,y)
  - return sameSet(x,y)
- isEnemy(x,y)
  - return sameSet(x,y')

Challenge: Allies and Enemies

Example Solution

n = 5, (countries numbered 1 - 5)

instructions	response	disjoint set
	false, no info yet	
	false, no info yet	
	OK (i.e., no conflict)	
	OK (i.e., no conflict)	
	conflict	
	OK (i.e., no conflict)	
	true	
	OK (i.e., no conflict)	
	false	
	true	*
	conflict	

- path compression

Challenge: Build a Maze

Given an NxN grid generate a random maze.

- we have a fixe start point and end point
- there should be no cycles in the maze
- every cell should be reachable from every other cell

Challenge: Building a Maze

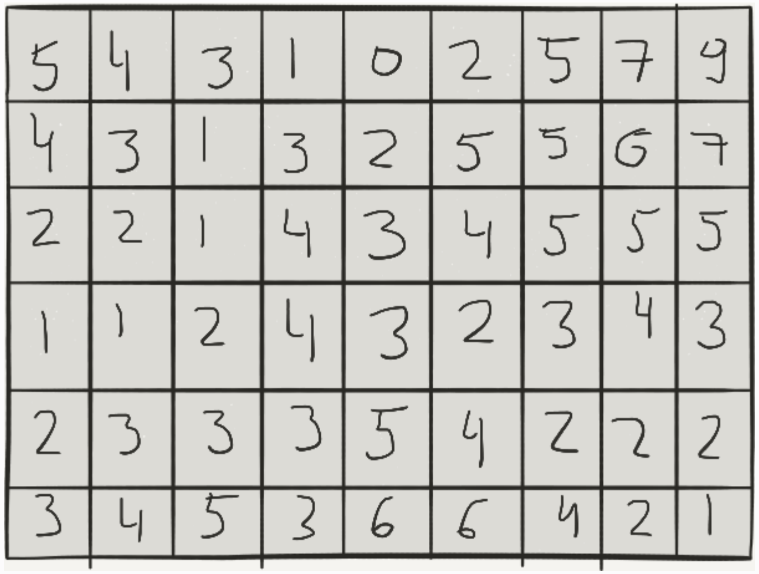
Algorithm:

- create a set of all internal walls
- choose a wall at random
- if that wall on each side of this wall are not in the same set,
  - then erase it (union the two sets in which they are in) -> this avoids cycles
- repeat until all cells are in the same set (each cell is reachable from every other cell)

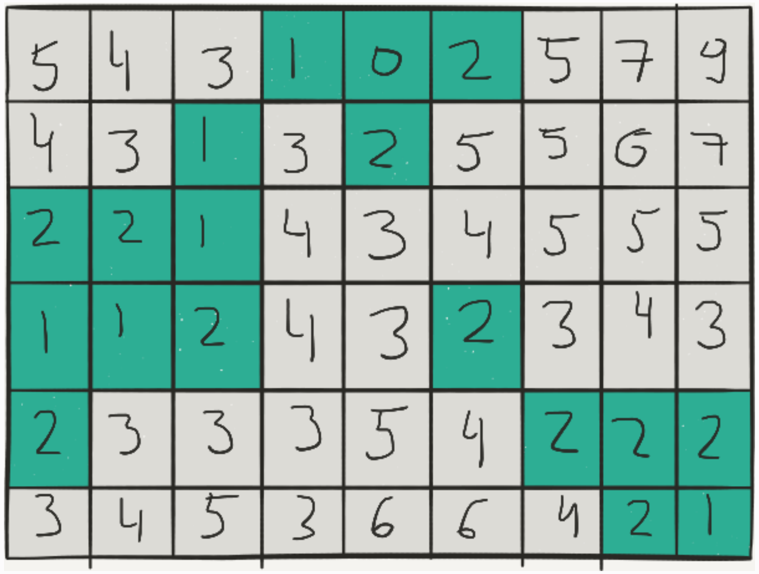
Challenge: Counting Islands

There NxM grid of integers gives terrain elevation. Given a water level L, every cell with the height (=elevation) <= L is below the water. The islands are the cells above the water. An island is a group of 4-connected cells (connected by the sides, not corners).

Determine the number of islands.



5	4	3	1	0	2	5	7	9
4	3	1	3	2	5	5	6	7
2	2	1	4	3	4	5	5	5
1	1	2	4	3	2	3	4	3
2	3	3	3	5	4	2	2	2
3	4	5	3	6	6	4	2	1



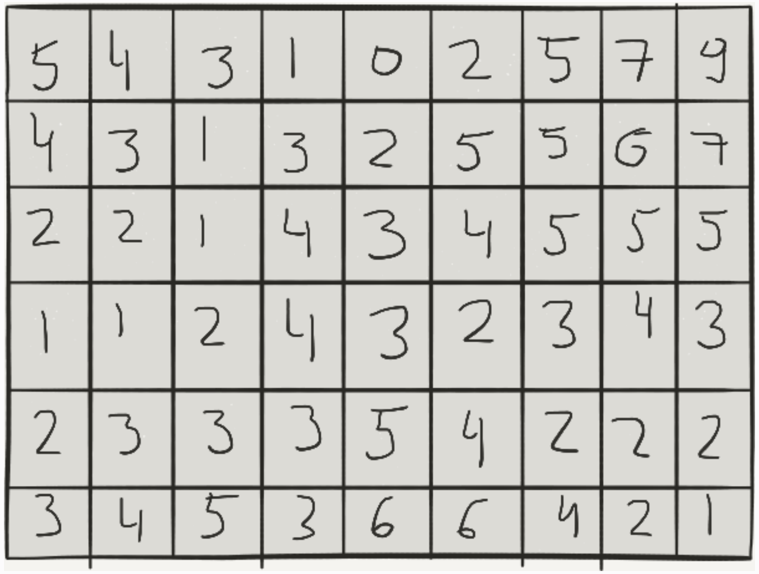
5	4	3	🟩 1	🟩 0	🟩 2	5	7	9
4	3	🟩 1	3	🟩 2	5	5	6	7
🟩 2	🟩 2	🟩 1	4	3	4	5	5	5
🟩 1	🟩 1	🟩 2	4	3	🟩 2	3	4	3
🟩 2	3	3	3	5	4	🟩 2	🟩 2	🟩 2
3	4	5	3	6	6	4	🟩 2	🟩 1

L = 2, there are 2 islands

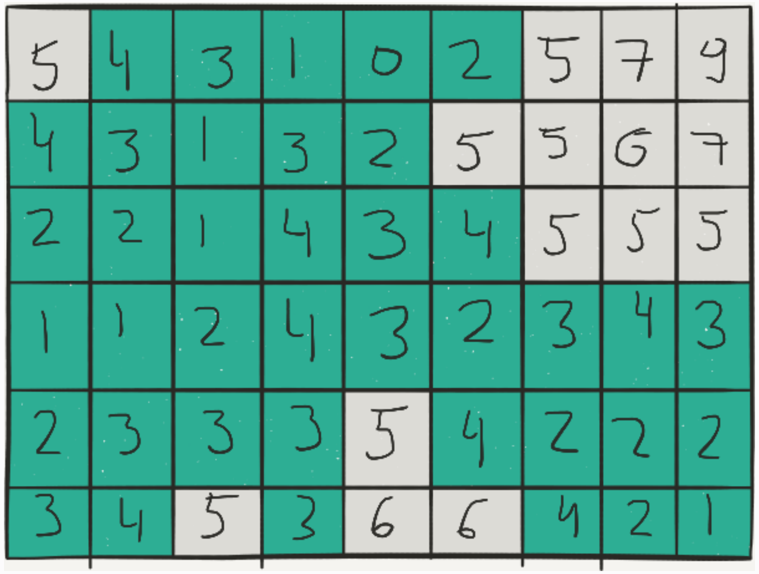
Challenge: Counting Islands

There NxM grid of integers gives terrain elevation. Given a water level L, every cell with the height (=elevation) <= L is below the water. The islands are the cells above the water. An island is a group of 4-connected cells (connected by the sides, not corners).

Determine the number of islands.



5	4	3	1	0	2	5	7	9
4	3	1	3	2	5	5	6	7
2	2	1	4	3	4	5	5	5
1	1	2	4	3	2	3	4	3
2	3	3	3	5	4	2	2	2
3	4	5	3	6	6	4	2	1



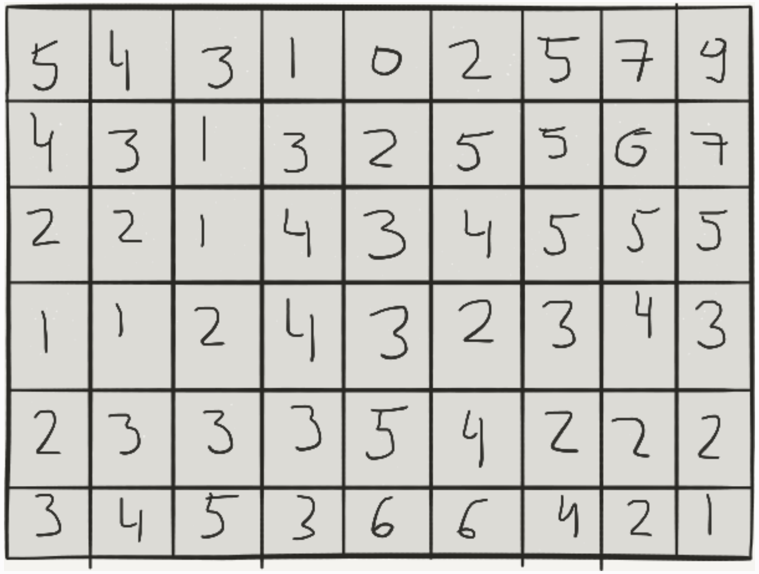
5	4	3	1	0	2	5	7	9
4	3	1	3	2	5	5	6	7
2	2	1	4	3	4	5	5	5
1	1	2	4	3	2	3	4	3
2	3	3	3	5	4	2	2	2
3	4	5	3	6	6	4	2	1

L = 4, there are 4 islands

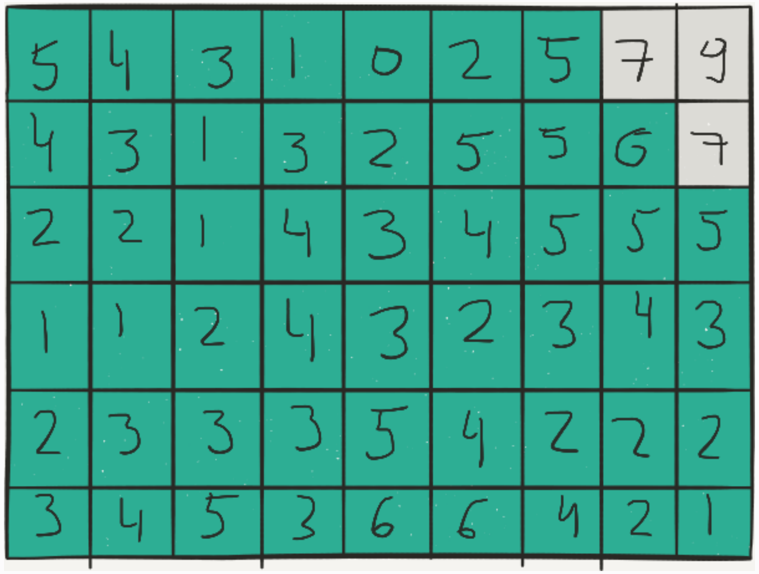
Challenge: Counting Islands

There NxM grid of integers gives terrain elevation. Given a water level L, every cell with the height (=elevation) <= L is below the water. The islands are the cells above the water. An island is a group of 4-connected cells (connected by the sides, not corners).

Determine the number of islands.



5	4	3	1	0	2	5	7	9
4	3	1	3	2	5	5	6	7
2	2	1	4	3	4	5	5	5
1	1	2	4	3	2	3	4	3
2	3	3	3	5	4	2	2	2
3	4	5	3	6	6	4	2	1



5	4	3	1	0	2	5	7	9
4	3	1	3	2	5	5	6	7
2	2	1	4	3	4	5	5	5
1	1	2	4	3	2	3	4	3
2	3	3	3	5	4	2	2	2
3	4	5	3	6	6	4	2	1

L = 6, there is 1 island

Counting Islands

Solution 1 Use O(N*M) algorithm that visits all the cells one by one. For each cell that is above the level of water find all the cells adjacent to it (DFS type search) and mark them as visited. For each discovered island increment the count of islands.

Counting Islands

Solution 1 Use O(N*M) algorithm that visits all the cells one by one. For each cell that is above the level of water find all the cells adjacent to it (DFS type search) and mark them as visited. For each discovered island increment the count of islands.

Solution 2 Use disjoint sets data structure.

Counting Islands

Solution 1 Use O(N*M) algorithm that visits all the cells one by one. For each cell that is above the level of water find all the cells adjacent to it (DFS type search) and mark them as visited. For each discovered island increment the count of islands.

Solution 2 Use disjoint sets data structure.

For each cell, look at the four adjacent neighbors. If they are above L union the two cells. The number of sets is the number of islands.

Counting Islands

Solution 1 Use O(N*M) algorithm that visits all the cells one by one. For each cell that is above the level of water find all the cells adjacent to it (DFS type search) and mark them as visited. For each discovered island increment the count of islands.

Solution 2 Use disjoint sets data structure.

For each cell, look at the four adjacent neighbors. If they are above L union the two cells. The number of sets is the number of islands.

Or is it? What about the "sets" associated with the cells with values <= L?

Counting Islands

Solution 1 Use O(N*M) algorithm that visits all the cells one by one. For each cell that is above the level of water find all the cells adjacent to it (DFS type search) and mark them as visited. For each discovered island increment the count of islands.

Solution 2 Use disjoint sets data structure.

For each cell, look at the four adjacent neighbors. If they are above L union the two cells. The number of sets is the number of islands.

Or is it? What about the "sets" associated with the cells with values <= L?

Problem variation

But, what if we need to provide the answer for "continuous" levels:

- global warming -> waters are rising
- after floding -> waters are slowly dropping

Counting Islands Continuous

This approach counts the islands for levels from largest (everything is under water) to smallest (everything is above water)

- each cell is in its own set
- for processing sort the cells from largest (highest elevation) to lowest
- for level L0 (the highest level), go through all cells with values > L0 and union each of them with its four neighbors if that neighbor is also above L0; keep track of the number of islands
- for level L1 (the next highest level), through through all the cells with values > L1 and <=L0, union each with its four neighbors if that neighbor is between L1 and L0; keep track of the number of islands
- ...

Challange: Weighted N-ary Tree

Each node in a tree has a non-negative integer weight assigned to it.

Task: Find the size of a maximum subtree in which **all** weights are even.

Example:

Number of nodes = 7

Weights of nodes

node index	node weight
1	1
2	2
3	6
4	4
5	2
6	0
7	3

Connectivity (based on node indexes):
(1, 2), (1, 3), (2, 4), (2, 5), (4, 6), (6, 7)

Calvin's Stars

Calvin likes to lie in a field and look at the night sky. Since he doesn't know any real star constellations, he makes them up: if two stars are close to each other, they must belong to the same constellation. He wants to name them all, but fears to run out of names. Can you help him and count how many constellations there are in the sky?

Two stars belong to the same constellation if distance between their projections on a two-dimensional sky plane isn't more than D units.

Input

There is a number of tests T ($T \leq 50$) on the first line. Each test case contains the number of stars N ($0 \leq N \leq 1000$) a real distance D ($0.00 \leq D \leq 1000.00$). Next N lines have a pair of real coordinates X Y ($-1000.00 \leq X, Y \leq 1000.00$) for each star. Real numbers in the input will have at most 2 digits after a decimal point.

Output

For each test case output a single line '`Case` T`:` N'. Where T is the test case number (starting from 1) and N is the number of constellations.

Sample Input

```
2
5 1.5
1.0 0.1
2.0 0.0
5.0 0.2
6.0 0.4
3.0 -0.1
3 4.0
```