# Exercise No. 4

28.03.06-11.04.06

The main purpose of this assignment is to practice implement a class given its contract and its abstraction function.

You will implement a class named `DistjointSets` that represents a set S of disjoint sets: S = {S1,S2, ...,Sn}, where each Si is a set of nonnegative integers. The class will support the following public methods:

```
// @pre x is not in any of the sets Si
// @post S = old(S) U {{x}}
public void makeSet(int x);

// @pre x in Si , y in Sj
// @return true iff i == j
public boolean equiv(int x, int y);

// @pre x in Si , y in Sj , i !=j
// @post S = old(S) - Si - Sj U {Sij} where Sij = Si U Sj
public void joinSets(int x, int y);

// @pre nothing
// @return true iff x is in some Si
public boolean inASet(int x);
```

One way to implement the class is by representing each set Si as a tree, where each node (associated with a nonnegative integer x) points to his parent. In this way, the root of a tree uniquely represents a set Si. An array named parent can be used to hold all these pointers. Specifically, for each number x, the value of parent[x] is the number of the parent node in the tree or −1 if the number x does not belong to any set Si. The root of a tree points to itself. The length of the array should be bigger than any number x that currently in one of the sets Si. Thus, there is a need to replace the current array with a bigger one when a new large number is added.

Given an object of the DisjointSets class, its abstract state is a set of disjoint sets of nonnegative integers, {S1,S2, ...,Sn}. To define the abstraction function, we first define a helper function r(x) as follows:

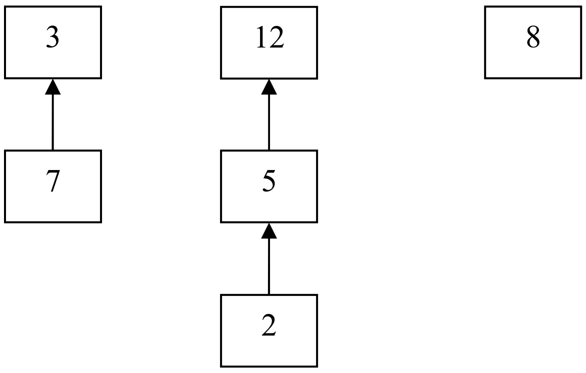
If parent[x] = −1 then r(x) = −1
Else if parent[x] == x then r(x) = x
Else r(x) = r(parent[x])

Now we can define the abstraction function that maps from parent, the field of the object, to a set of sets:
S(this) = $\{S_1, S_2, \ldots S_n\}$ such that
For all x, [for all i, $1 \le i \le n$, $x \notin S_i$ ] iff [$x \ge$ parent.length or parent[x] = −1 ]
For all $0 \le x,y \le$ parent.length, $x,y \in S_i$ (x and y are in the same set) iff r(x) = r(y) ≠ −1

For example, for an object of the `DisjointSets` class with `parent` =

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| -1 | -1 | 5 | 3 | -1 | 12 | -1 | 3 | 8 | -1 | -1 | -1 | 12 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |

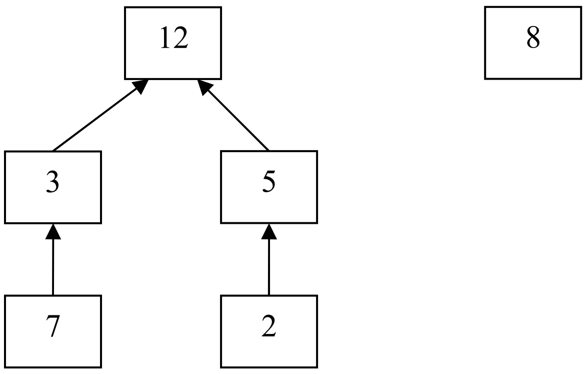
the associated tress are:



and the abstract state is S = { {3, 7} {12, 5, 2} {8} }. Applying `equiv(3,5)` will return `false`. If we apply `joinSets(7,5)` the parent array will be

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| -1 | -1 | 5 | 12 | -1 | 12 | -1 | 3 | 8 | -1 | -1 | -1 | 12 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |

where the associated trees are



and the abstract state is S = { {3, 7, 12, 5, 2} {8} }. If we apply `makeSet(4)` the abstract state will be S = { {3, 7, 12, 5, 2} {8} {4}}.

Your assignment is to:

- Implement the DisjointSets class efficiently using the parent array (a template can be found on the course web site). Note that for your convenient you may need to define helper methods. For example, it may help to define a function that ensures that parents[] is long enough.