The Doors

Borrowed from 2017 Maryland contest

Rey and her friends find themselves in a long corridor with many (say, *n)* doors heading in different directions, and a switchboard with a number of switches (*m)* on them. Both the doors and the switches are numbered from *0* to *n-1* and *0* to *m-1* respectively. A switch however, does not open or close a single door; Instead, each switch toggles the state of a subset of the doors. In the example below, the first switch (Switch 0) will toggle the states of the doors 0, 1, 3, and 4; If one of those doors is open, it will be closed, and if one of them is closed, it will be opened. Similarly, the second switch will toggle the states of the doors 1, 2, and 4. The doors are all initially closed, and Rey needs to open a specific subset of the doors (they are going to split up to do the different tasks), and no others (to ensure no storm troopers can intercept them). They only have one shot at going through the corridor, so she can't open the doors one after the other; Instead she must toggle a subset of the switches so the desired configuration is reached, and then they will all start running towards their respective doors.

Switch 0	Toggles doors 0, 1, 3, and 4
Switch 1	Toggles doors 1, 2, and 4
Switch 2	Toggles doors 2 and 3
Switch 3	Toggles doors 0 and 4

For example, assuming a corridor with 5 doors (0 thru 4):

- if the goal was to have exactly doors 2 and 3 open, then we would use switch 2 to open doors 2 and 3.
- If the goal was to have only door 0 open, then we would use switch 0 to open doors 0, 1, 3, and 4, switch 1 will close door 1, open door 2 and close door 4, and then Switch 2 will close door 2 and close door 3.
- On the other hand, there is no way to get the doors 0 and 1 open, without keeping other doors open.

You have to help Rey decide on the set of switches to toggle to achieve the desired configuration of open and closed doors. If more than one solution exists, return the solution which requires using the fewest switches.

This problem uses the completed `Switches` helper class representing a switch which toggles the state of a subset of the doors. The `Switches` class has accessor methods `getSwitch()` (returns an `int[]` containing the door numbers to be toggled by this switch), `size()` (returns `getSwitch().length()`, that is, the number of doors this switch toggles), `get(int numDoor)` (returns `getSwitch()[numDoor]`, that is door number at index `numDoor`) and a correctly implemented `equals` method. For example:

The following code creates a switch that toggles the state of doors 0, 1, 3, and 4.
`Switch sw0 = new Switch(new int[] {0, 1, 3, 4});`
`sw0.getSwitch()` returns an int array containing the number 0, 1, 3 and 4
`sw0.size()` returns 4
`Sw0.get(2)` returns 3

The Doors

Borrowed from 2017 Maryland contest

In this problem you will complete one method in the `TheDoors` class. The `TheDoors` class has `List<Switches> mySwitches` and an `int numDoors` as two instance variables and one constructor which initializes both instance variables. You should assume `2 <= numDoors <= 10` and `1 <= mySwitches.length() <= 10`.

The one method to complete is: `ArrayList<Switches> findSwitchCombination(int[] goal)`. `findSwitchCombination` has a single parameter, `int[] goal`, which contains the door numbers to open. `findSwitchCombination` returns a `List<Switches>` to toggle to achieve the desired configuration of open and closed doors. If more than one List exists, return any List with the fewest switches.

The following code shows the results of the `findSwitchCombination` method.

The following code	Returns

`Switches switch0 = new Switches(new int[] {0, 1, 3, 4});`
`Switches switch1 = new Switches(new int[] {1, 2, 4});`
`Switches switch2 = new Switches(new int[] {2, 3});`
`Switches switch3 = new Switches(new int[] {0, 4});`
`List<Switches> switches = new ArrayList<Switches>();`
`switches.add(switch0);`
`switches.add(switch1);`
`switches.add(switch2);`
`switches.add(switch3);`
 	
`TheDoors tDoor = new TheDoors( switches, 5);`	

`//  open doors 2 and 3 only`
 	
`ArrayList<Switches> ans =`
`tDoor.findSwitchCombination(new int[] {2, 3});`	

`ans.size();`	`1`
`ans.contains(switch0);`	`false`
`ans.contains(switch1);`	`false`
`ans.contains(switch2);`	`true`
`ans.contains(switch3);`	`false`
Another example using the same switches On the next page	

The Doors

Borrowed from 2017 Maryland contest

`//  open door 0 only`
 	
`ArrayList<Switches> ans =`
`tDoor.findSwitchCombination(new int[] {0});`	

`ans.size();`	`3`
`ans.contains(switch0);`	`true`
`ans.contains(switch1);`	`true`
`ans.contains(switch2);`	`true`
`ans.contains(switch3);`	`false`
Another example using the same switches	

`//  open doors 0 and 1 only  - remember, no combination exsist`
 	
`ArrayList<Switches> ans =`
`tDoor.findSwitchCombination(new int[] {0, 1});`	

`ans.size();`	`0`
`ans.contains(switch0);`	`false`
`ans.contains(switch1);`	`false`
`ans.contains(switch2);`	`false`
`ans.contains(switch3);`	`false`