Caucasus

Sets and dictionaries in doctests

The elements of a set and the keys of a dictionary do not have a particular order. This means that two sets or two dictionaries are equal, irrespective of the order in which the elements/keys have been added to the set/dictionary.

```
>>> {1, 3, 2, 4} == {4, 3, 1, 2}
True
>>> {'A': 1, 'B': 2, 'C': 3} == {'B': 2, 'A': 1, 'C': 3}
True
```

When working with doctests, however, sets and dictionaries might cause you trouble. The reason for this problem is that in comparing expected and generated results, doctests proceed as follows: the expected result is extracted from the doctest as a string, and the value returned by a function or resulting from the evaluation of an expression is converted to a string. These two strings are then compared with each other (as a string, not as a set or a dictionary). In comparing strings, the order of the characters is important, and that's what's causing the trouble.

```
>>> d1 = {'A': 1, 'B': 2, 'C': 3}
>>> s1 = str(d1)   # executed on computer 1
>>> d1
"{'A': 1, 'C': 3, 'B': 2}"
>>> d2 = {'A': 1, 'B': 2, 'C': 3}
>>> s2 = str(d2)   # executed o computer 2
"{'A': 1, 'B': 2, 'C': 3}"
>>> d1 == d2
True
>>> s1 == s2
False
```

Although dictionaries `d1` and `d2` have the same value, the doctest indicates that the string representations of these two dictionaries are different. The conversion of a set/dictionary to a string may depend on the computer that executes the code, the Python version used on that computer, and might even differ if you do repeat the same conversion on the same computer using the same Python version. The problem can be solved by making sure the doctests do not compare strings, but directly compare sets or dictionaries. For example, if a doctest initially looks like

```
>>> aFunction(parameter1, parameter2)
{'A' : 1, 'B': 2, 'C': 3}
```

you may better rewrite this doctest as

```
>>> aFunction(parameter1, parameter2) == {'A' : 1, 'B': 2, 'C': 3}
True
```

This problem never occurs on Dodona, since its way of testing the source code never converts return values or results of expression evaluations into strings, but directly compares the resulting objects.

Specific information

When associating a value with a dictionary key, you may have to take into account that the dictionary may or may not already associate a value to this key. It is often the case that you have to add a new key/value pair in case the key was not used in the dictionary, and thay you have to update an existing key/value pair in case the key was already used in the dictionary.

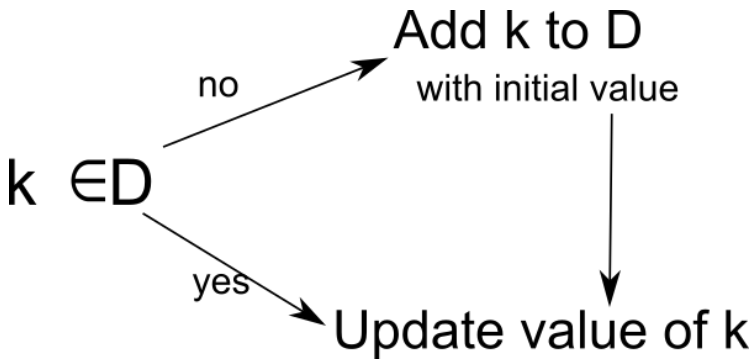

Figure 1: update dictionary