A Formal Definition

We need a formal definition of a reduction.

We will say "A reduces to B in polynomial time" (or "A is polynomial time reducible to B" or "A reduces to B" or "$A \leq B$") if:

There is an algorithm to solve problem A, which, if given access to a polynomial-time algorithm for problem B, runs in polynomial time overall (including the library's running time!!!).

Let's Do A Reduction

4 steps for reducing (decision problem) A to problem B.

1. Describe the reduction itself (i.e. the algorithm, with a call to a library for problem B)

2. Make sure the running time would be polynomial (usually skip writing out this step).

3. Argue that if the correct answer (to the instance for A) is YES, then our algorithm answers YES.

4. Argue that if the correct answer (to the instance for A) is NO, then our algorithm answers NO.

Reduce 3-coloring to 4-coloring

What's 3-coloring?

3-coloring
Input: Undirected Graph G
Output: `True` if the vertices of G can be labeled with red,green, and blue so that no edge has both of its endpoints colored the same color. `False` if it cannot.

4-coloring
Input: Undirected Graph G
Output: `True` if the vertices of G can be labeled with red,green, blue, and orange so that no edge has both of its endpoints colored the same color. `False` if it cannot.

Correctness?

```
3ColorCheck(Graph G)
    Let H be a copy of G
    Add a vertex to H, attach it to all
      other vertices.
    Bool answer = 4ColorCheck(H)
    return answer
```

TWO statements to prove: ("two directions")

If the correct answer for G is YES, then we say YES

If the correct answer for G is NO, then we say NO