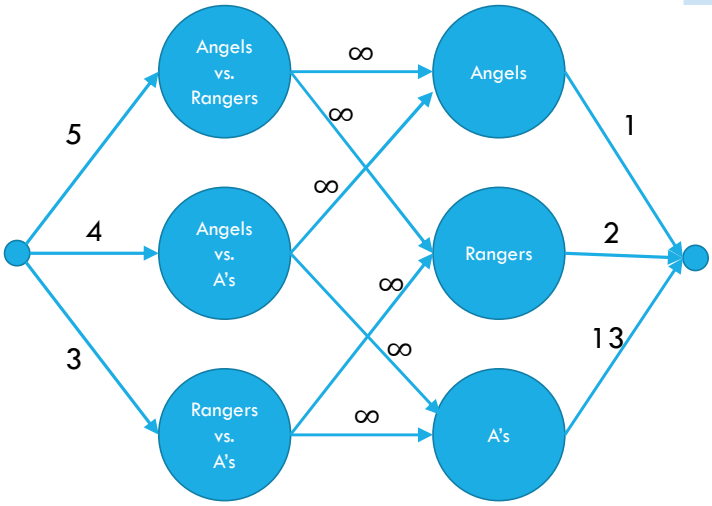
Making a Network



	Angels	Rangers	Mariners	A's
Angels	-	5	3	4
Rangers	5	-	4	3
Mariners	3	4	-	5
A's	4	3	5	-

Team	Wins (w)	Possible Wins (P)
Angels	81	93
Rangers	80	92
Mariners	70	82
A's	69	81

We're done!

"Ranking" difficulty of problems

We'll use "reductions" to tell whether one problem is harder than another.

Reduction (informally)

Using an algorithm for Problem B to solve Problem A.

In that case, we'll say "A reduces to B"

In difficulty (for us, as algorithm designers), $A \leq B$

ANY algorithm for B solves A. A is no harder to solve than B.

A might be easier (maybe there's another way to solve A without B) or they might be about the same (maybe $B \leq A$ too!)

Correctness?

```
2ColorCheck(Graph G)
     Let H be a copy of G
     Add a vertex to H, attach it to all
       other vertices.
     Bool answer = 3ColorCheck(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

NP

Our second set of problems have the property that "I'll know it when I see it"
We're looking for **something**, and if someone shows it to me, we can recognize it quickly (it just might be hard to find)

NP (stands for "nondeterministic polynomial")

The set of all decision problems such that for every YES-instance, there is a certificate for that instance which can be verified in polynomial time.

If you have a "YES" instance, a little birdy can magically find you this certificate-thing, and you'll say "Oh yeah, that's totally a yes instance!"

What if it's a NO instance? No guarantee.