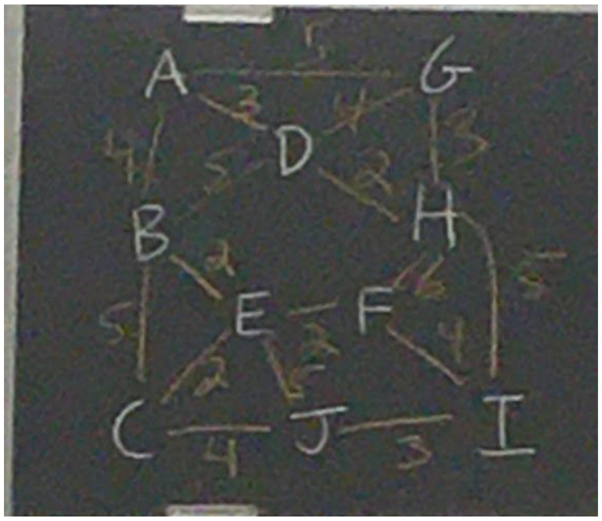
# Properties and Oracles

Weaknesses of tests:

- Can gain confidence in correctness of code, but can't know for sure
- Can't test every single possible input
- Can't deal with non-deterministic programs



Source: G, Destination: E

Find shortest path in above graph:

- E.g. `[G -> D -> B -> E]` is a shortest path
- Problem: there are multiple shortest paths: **multiple correct answers!**
    - `[G -> H -> F -> E]`, `[G -> A -> B -> E]`
    - How do we write test cases for inputs with many possible correct outputs such that we don't want to figure out each possible correct answer by hand?
- The problem domain that we're testing is a relation (input has many valid outputs) instead of a function
    - Other problems that face similar testing difficulties:
        - Minimum spanning tree
        - Sorting algorithms
- What is the type signature of a test suite?
    - Consumes a function and produces a boolean which denotes, very generally speaking, whether the input function is correct
    - What's another tool that we could design with the same type signature that solves some of the above challenges?
- We can construct a function for our problem domain, `isValid`, which consumes an input and an output and determines whether the given output is a valid solution on the given input.
    - What should `isValid` do for the shortest path problem?

        - Consumes a graph and a sequence of vertices.
        - Checks the following properties:
            - Is the sequence of vertices actually a path in the input graph?
            - Does the path start with `src` and end with `dst`?
            - Is the path actually the shortest path in the graph?
    - **Testing against properties** instead of testing against values
- Another approach: think about how we come up with inputs to `isValid` or to normal I/O tests.
    - Usually, we make them up using out intuitions about what sorts of cases are likely to catch bugs (e.g. positive and negative numbers, long lists, empty lists, etc.)
    - Alternatively, we can write a program to generate inputs!
        - This is a nontrivial problem - we want to make sure we generate a good distribution of inputs.  Consider a graph generator - we want to make sure we test sparse and dense graphs, as well as connected and disjoint graphs.  A naive input generator likely won't be good at generating all of these sorts of inputs.
        - There are good QuickCheck libraries in many languages with specifiable distributions for generating test inputs.
    - What do we do with inputs that we generate?
        - We use `isValid` to check them!
        - Call this combination of input generation/property checking an Oracle.
- This isn't perfect - Oracles don't prove correctness, but they help a lot with expert blindspot in generating test cases and they make it easier to test nondeterministic solutions or problems with multiple valid answers.