I have a problem

My problem C is too difficult to solve (at least for me).

So difficult, it's probably NP-hard. How do I show it?

What does it mean to be NP-hard?

We need to be able to reduce any problem A in NP to C.

Let's choose B to be a known NP-hard problem. Since B is known to be NP-hard, $A \leq B$ for every possible A. So if we show $B \leq C$ too then $A \leq B \leq C \rightarrow A \leq C$ so every NP problem reduces to C!

Approximation Ratio

For a minimization problem (find the shortest/smallest/least/etc.)

If $OPT(\mathrm{G})$ is the value of the best solution for G, and $ALG(G)$ is the value that your algorithm finds, then ALG is an α approximation algorithm if for every G,

$$\alpha \cdot OPT(G) \geq ALG(G)$$

i.e. you're within an α factor of the real best.

Recall: Finding an approximation for VC

For every edge, at least one of u,v is in the minimum vertex cover.

But instead of checking which of u, v a good idea to add, just add them both!

```
While(G still has edges)
     Choose any edge (u,v)
     Add u to VC, and v to VC
     Delete u v and any edges touching them
EndWhile
```

We talked about this before. During greedy algorithms week!

Another Algorithm

Lets try to approximate Travelling Salesperson.

Traveling Salesperson

Given a weighted graph, find a tour (a walk that visits every vertex and returns to its start) of weight at most k.

Some assumptions:

1. The graph is undirected.

2. The graph is complete (every edge is there) – the edges might represent series of roads rather than individual streets. Weight is how much gas you need to travel.

2. The weights satisfy the "triangle inequality" (it's faster to go from x to y directly than it is to go from x to y through x).