Recurrence

$OPT(v)$ – the weight of the minimum weight vertex cover for the tree rooted at v (whether or not v is included).

$INCLUDE(v)$ – the weight of the minimum weight vertex cover for the tree rooted at v where v is included in the vertex cover.

$$OPT(v) = \begin{cases} \min\{\sum_{u:u \text{ is a child of } v} INCLUDE(u), weight(v) + \sum_{u:u \text{ is a child of } v} OPT(u)\} & \text{if } v \text{ is not a leaf} \\ 0 \text{ if } v \text{ is a leaf} \end{cases}$$

$$INCLUDE(v) = weight(v) + \sum_{u:u \text{ is a child of } v} OPT(u)$$

Vertex Cover Dynamic Program

What memoization structure should we use?

What code should we write?

What's the running time?

A recurrence

$$dist(v) = \begin{cases} 0 & \text{if } v \text{ is the source} \\ \min\limits_{u:(u,v)\in E} \{dist(u) + weight(u,v)\} & \text{otherwise} \end{cases}$$

Our memoization structure can be the graph itself.

What's an evaluation order? (Remember we're in a DAG!)

Ordering

Instead of $dist(v)$, (the true distance) right from the start, we'll let

$dist(v, i)$ to be the length of the shortest path from the source to v that uses at most i edges.

That breaks ties – counting the number of edges required!

$dist(v, i) =$