

COL 351: Analysis and Design of Algorithms

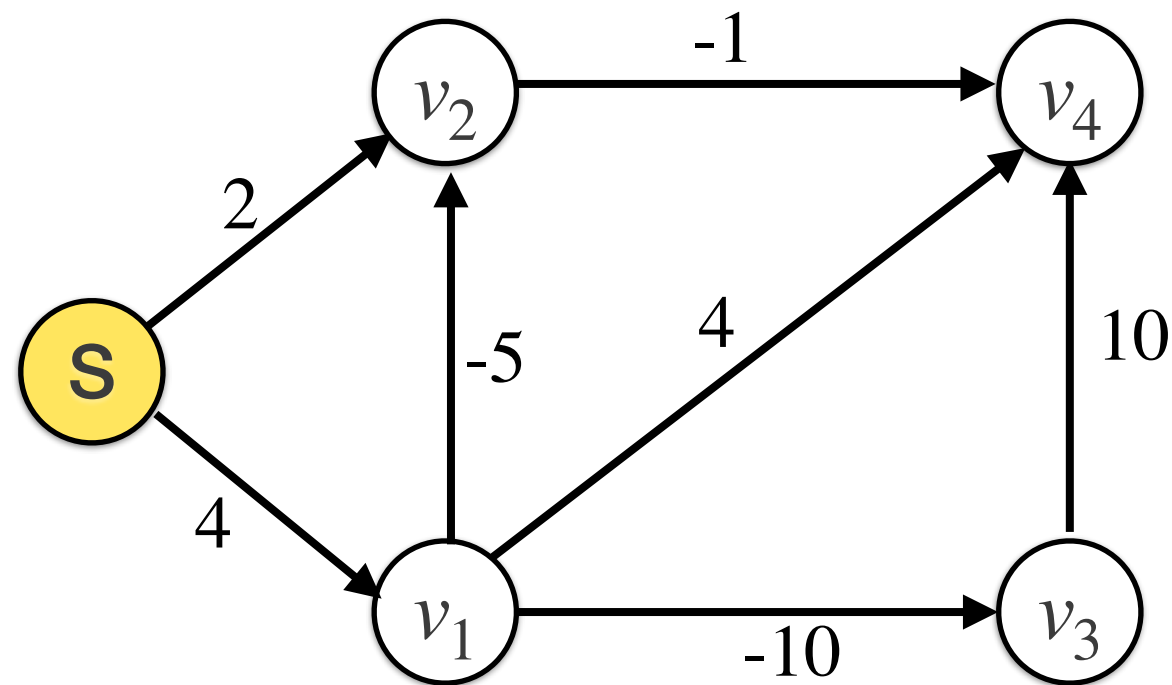
Lecture 15

Single Source Distance Problem

Given: A directed weighted graph $G = (V, E)$ with possibly *negative* edge weights, and a source vertex s .

Output: Find either a **shortest-path-tree** rooted at s , or prove that such a tree doesn't exist.

Example:



*What is
shortest path
tree from s ?*

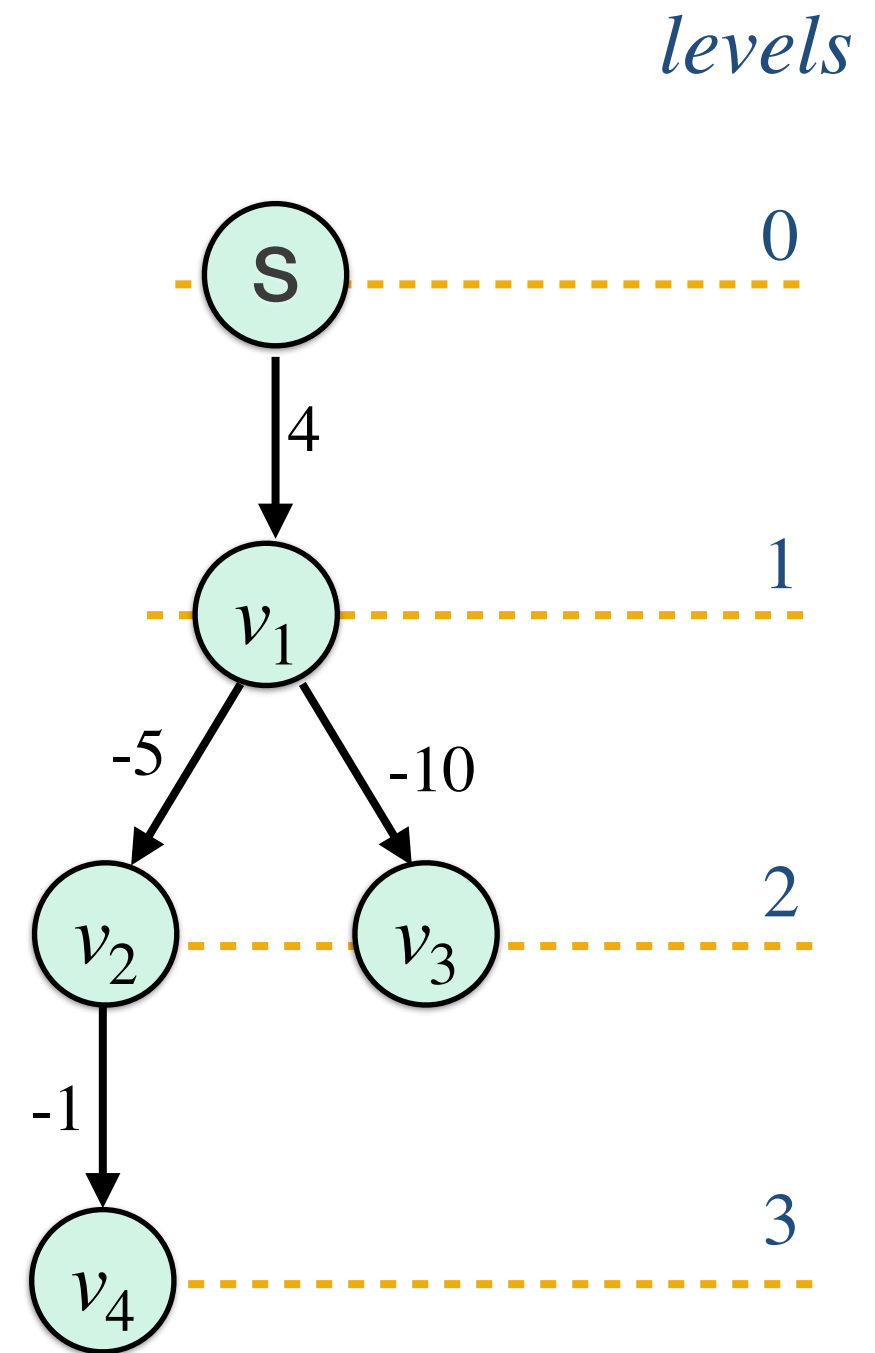


Notation: “Level” of a vertex

Level (v):

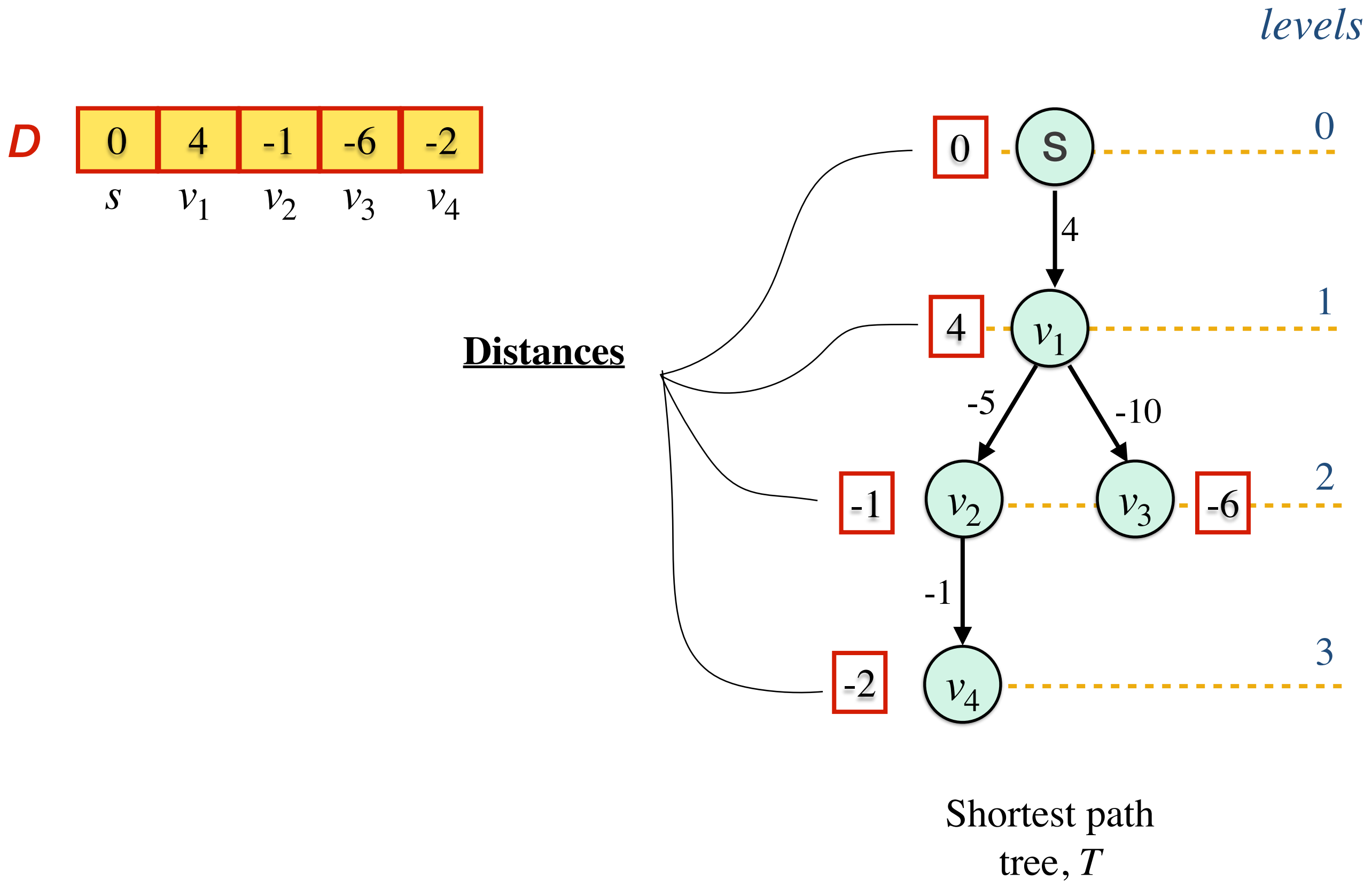
Number of edges on $s - v$
shortest path.

(Break the ties by taking min value).



Shortest path
tree, T

Notation: Distance vector



Simpler Question:

How can one verify if
a given 'D' is correct?

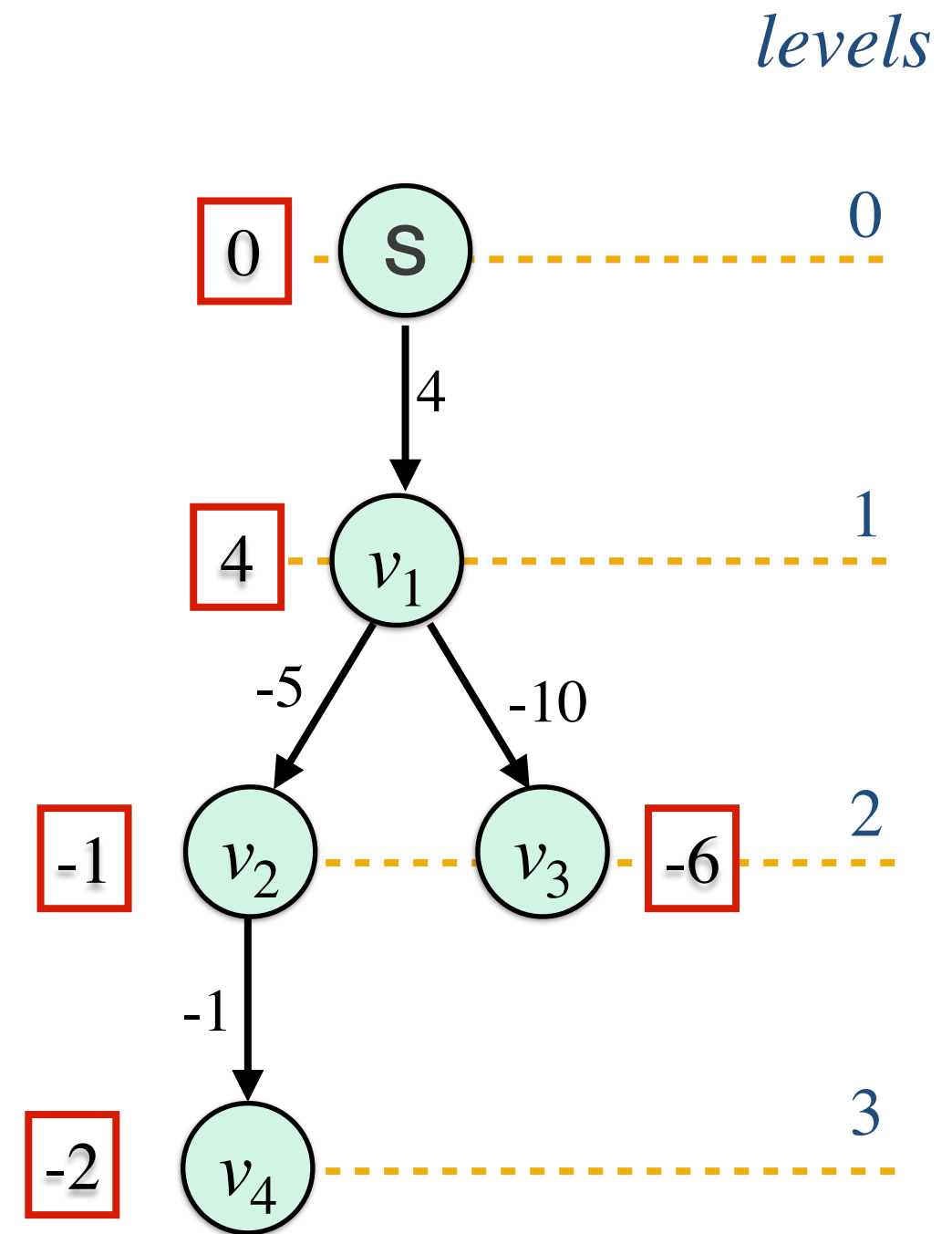
D	0	4	-1	-6	-2
	s	v_1	v_2	v_3	v_4

Claim:

If D is correct then:

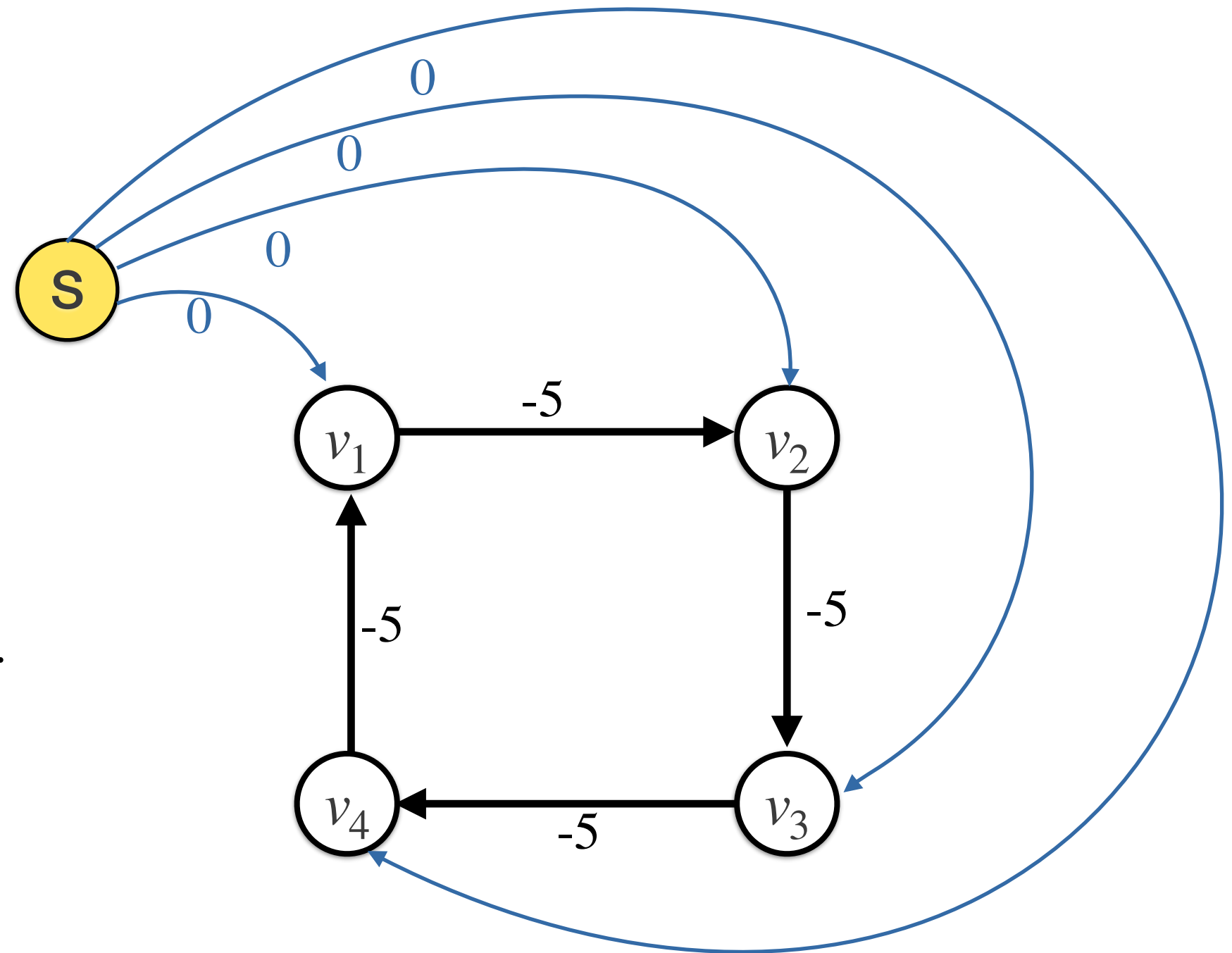
For Each $(x, y) \in E$:

$$D[y] \leq D[x] + \text{weight}(x, y)$$



Question:

Does a shortest path tree always exist?



Shortest path tree cannot be defined for the graph on right. (Why?)

Homework:

Compute shortest path for all vertices in the cycle.

Definition

An n -sized array D is *valid / correct* upto level i if:

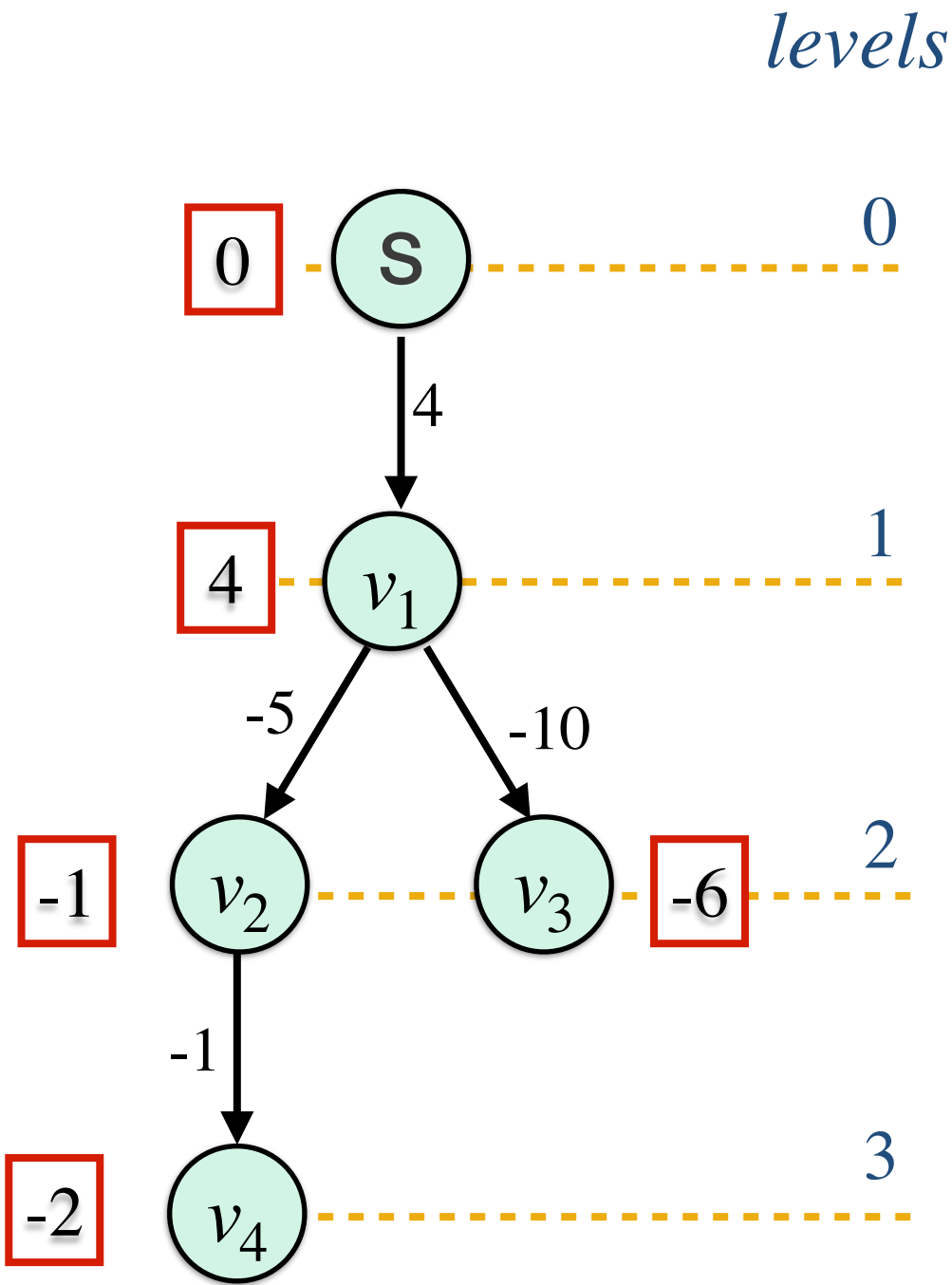
- For each v with level at most i ,
 $D[v] = \text{dist}(s, v, G)$.
- For each v with level larger than i ,
 $D[v] \geq \text{dist}(s, v, G)$.

Valid up to level $i = 2$

D	0	4	-1	-6	99
Levels	0	1	2	2	3
	s	v_1	v_2	v_3	v_4

Valid up to level $i = 1$

D	0	4	-1	66	99
Levels	0	1	2	2	3
	s	v_1	v_2	v_3	v_4



Lemma

Lemma: Let D be an n sized array that is valid upto level i . Then executing the following step ensures that D is valid upto level $i + 1$.

For Each $(x, y) \in E$:

If $(D[y] > D[x] + \text{weight}(x, y))$ **then**

$D[y] = D[x] + \text{weight}(x, y)$

Proof Sketch:

Consider a vertex y in level $i + 1$. Let x be parent of y in T . Then, $\text{level}(x) = i$.

Now, $\text{distance}(s, y) = \text{distance}(s, x) + \text{weight}(x, y)$, as x is predecessor of y on a $s - y$ shortest path.

The level of x is i which means $D[x]$ is correct, so executing the above code will ensure $D[y] = \text{distance}(s, y)$.

Home-work: Argue that for vertices upto level i , there will be no change in D on executing the above code.

Algorithm (assuming no negative weight cycle)

For Each $v \in V$:

$D[v] = \infty$ and $\text{parent}[v] = \text{null}$

$D[s] = 0$

For $i = 1$ to $n - 1$:

For Each $(x, y) \in E$:

If $(D[y] > D[x] + \text{weight}(x, y))$ **then**

$D[y] = D[x] + \text{weight}(x, y)$

$\text{parent}[y] = x$

Return D , parent .

Time = $O(mn)$

Question: What if G has a negative weight cycle reachable from s ?

What if G has negative weight cycle reachable from s ?

Lemma: G has ‘negative weight cycle’ reachable from s if and only if we can make improvement in vector D even in n^{th} round by using the following procedure.

For Each $(x, y) \in E$:

If $(D[y] > D[x] + \text{weight}(x, y))$ **then**

$D[y] = D[x] + \text{weight}(x, y)$

Proof:

Home-work

Bellman-Ford Algorithm

For Each $v \in V$:

$D[v] = \infty$ and $\text{parent}[v] = \text{null}$

$D[s] = 0$

For $i = 1$ to $n - 1$:

For Each $(x, y) \in E$:

If $(D[y] > D[x] + \text{weight}(x, y))$ **then**

$D[y] = D[x] + \text{weight}(x, y)$

$\text{parent}[y] = x$

If $(\exists \text{ an edge } (x, y) \text{ satisfying } D[y] > D[x] + \text{weight}(x, y))$ **then**

Return “Negative-weight cycle found.”

Return D , parent .

$O(mn)$ time algorithm for
graphs with negative weights

Challenge Problems

Problem 1: How can you find in $O(mn)$ time a *cycle of negative weight* reachable from s ?

Problem 2: Let G be a directed weighted graph with the property that no cycle in it has negative weight. Let s be a source vertex.

Given a parameter L , design an $O(mL)$ total time algorithm to find for each $v \in V$, an $s - v$ path of minimum weight having at most L edges (If it exists).