Lecture 12

CSE 331

Sep 24, 2018

Mini Project group due TODAY!

note

0 views

Actions

Clarifications on mini project

Two comments as you finalize your mini project choices:

- You are responsible for forming a group of size EXACTLY 3. I will not be forming groups from students who could not find a group at the end. **If you have not formed a group by Monday, then you get a zero on the mini project**.
- Once you submit a case study and it is not flagged as not having a conflict, your choice is considered to be final.
    - I'm open to considering requests for change but you need a good reason and this has to be done by email. I.e. do not fill in the form again-- if you do, I will simply delete your later choices.

mini_project

edit good note | 0

Updated Just now by Atri Rudra

You and 331

https://buffalonews.com/2018/09/23/twitter-turns-air-josh-allen-hurdle-photo-from-bills-vikings-game-into-meme/

Connectivity Problem

Input: Graph $G = (V,E)$ and s in V

Output: All t connected to s in G

Breadth First Search (BFS)

BFS via examples

In which we derive the breadth first search (BFS) algorithm via a sequence of examples.

Expected background

These notes assume that you are familiar with the following:

- Graphs and their representation. In particular,
  - Notion of connectivity of nodes and connected components of graphs
  - Adjacency list representation of graphs
  - Notation:
    - $G = (V, E)$
    - $n = |V|$ and $m = |E|$
    - $CC(s)$ denotes the connected component of s
- Trees and their basic properties

The problem

In these notes we will solve the following problem:

Connectivity Problem

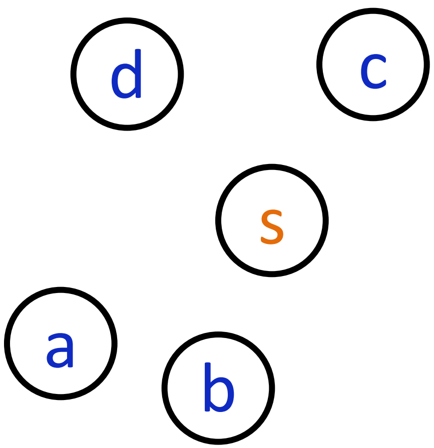
Input: Graph G = (V,E) and s in V

Output: All t connected to s in G

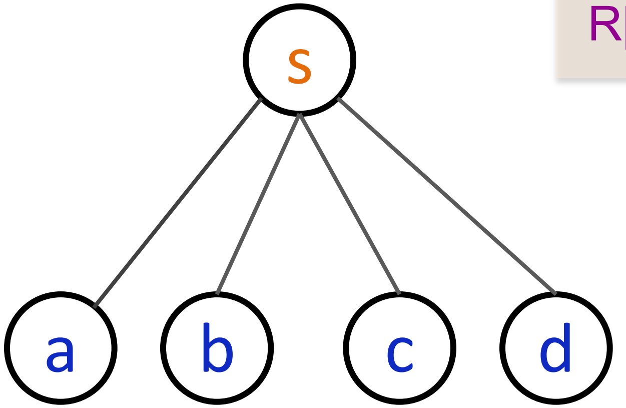
$\boxed{\text{Boolean array R such that R[t] = true iff t is connected to s}}$

Example 0: Know nothing about E

R[s] = T and R[u] = F for every u != s



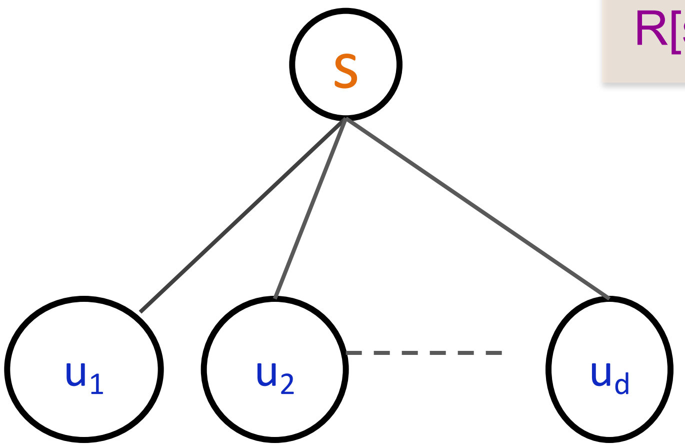
Example 1



R[s] = T and R[u] = F for every u != s

R[s] = R[a] = R[b] = R[c] = R[d] = T

Example 1: G is a “star”



Current algo:

R[s] = T and R[u] = F for every u != s

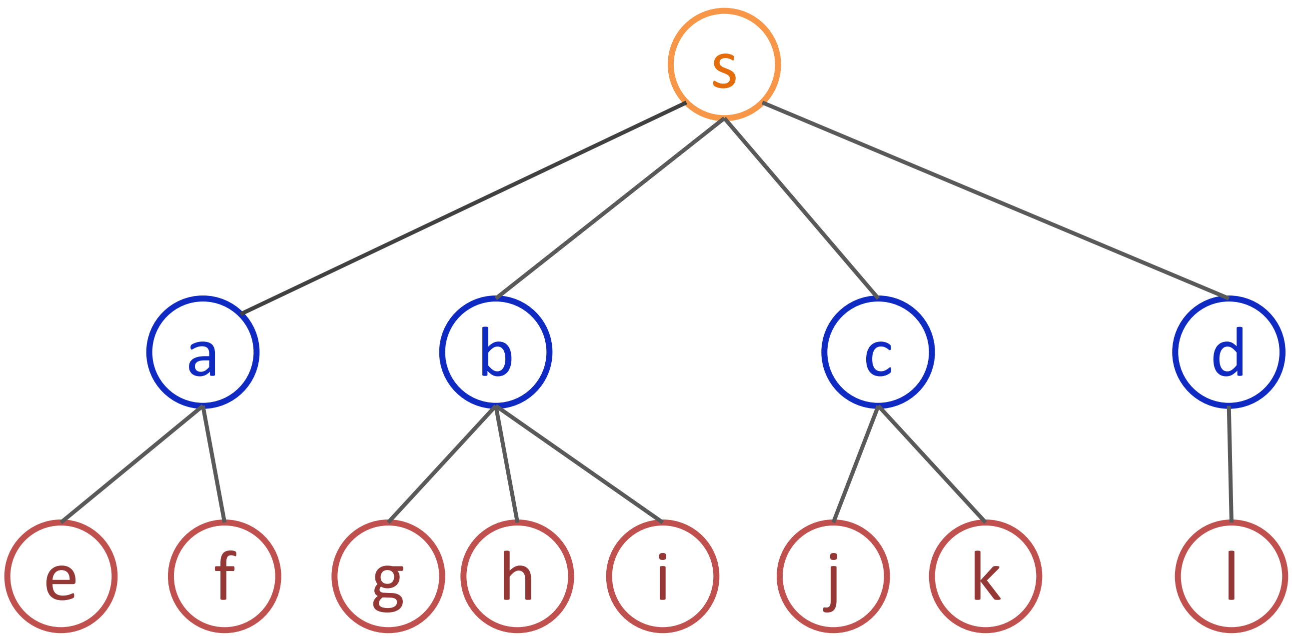
Updated algo:

R[s] = T and R[u] = F for every u != s

For every (s, u) in E

R[u] = T

Example 2’



Re-stating the algo

Current algo:

R[s] = T and R[u] = F for every u != s

For every (s, u) in E

    R[u] = T

Re-written algo:

R[s] = T and R[u] = F for every u != s

L_0 = {s}

L_1 = null

For every u in L_0

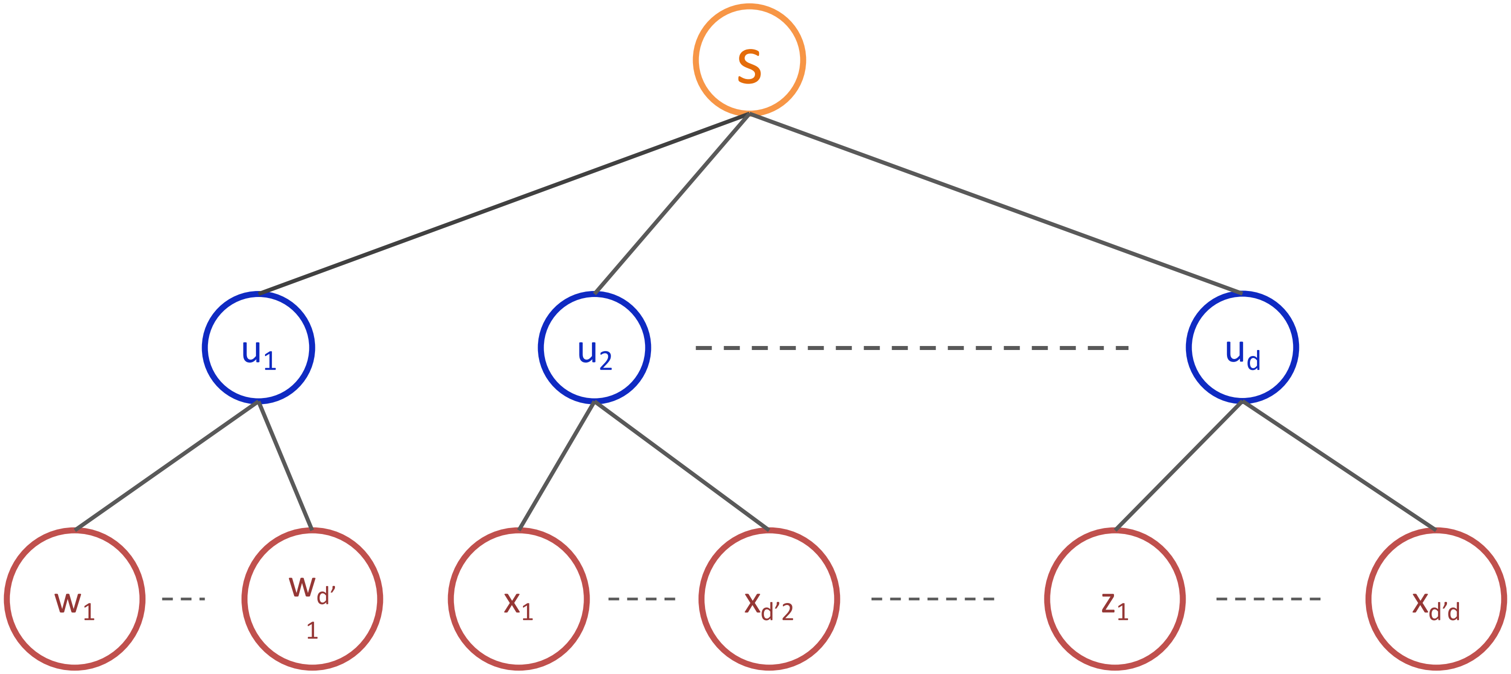
    For every (u,w) in E

        Add w to L_1

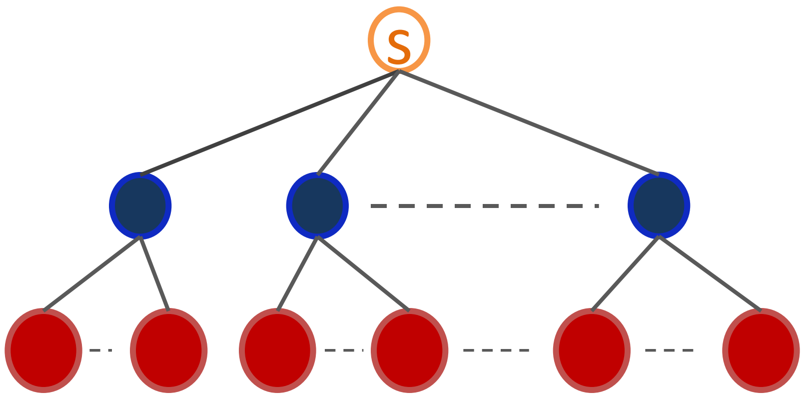
For every w in L_1

    R[w] = T

Example 2



Example 2 algo



$R[s] = T$ and $R[u] = F$ for every $u \neq s$

$L_0 = \{s\}$

L_1 , L_2 = null

For every u in L_0

    For every (u,w) in E

        Add w to L_1

For every w in L_1

    $R[w] = T$

For every u in L_1

    For every (u,w) in E

        Add w to L_2

For every w in L_2

    $R[w] = T$

This is an unwieldy algo

Identify two inefficiencies: one is trivial and another is a bit more subtle

$R[s] = T$ and $R[u] = F$ for every $u \neq s$

$L_0 = \{s\}$

L_1 , L_2 = null

```
For every u in L0
    For every (u,w) in E
        Add w to L1

For every w in L1
    R[w] = T

For every u in L1
    For every (u,w) in E
        Add w to L2

For every w in L2
    R[w] = T
```

The trivial inefficiency (ala 115)

The highlighted pieces of code can be combined

$R[s] = T$ and $R[u] = F$ for every u != s

$L_0 = \{s\}$

L_1 , L_2 = null

For every u in L_0

For every (u,w) in E

Add w to L_1

For every w in L_1

$R[w] = T$

For every u in L_1

For every (u,w) in E

Add w to L_2

For every w in L_2

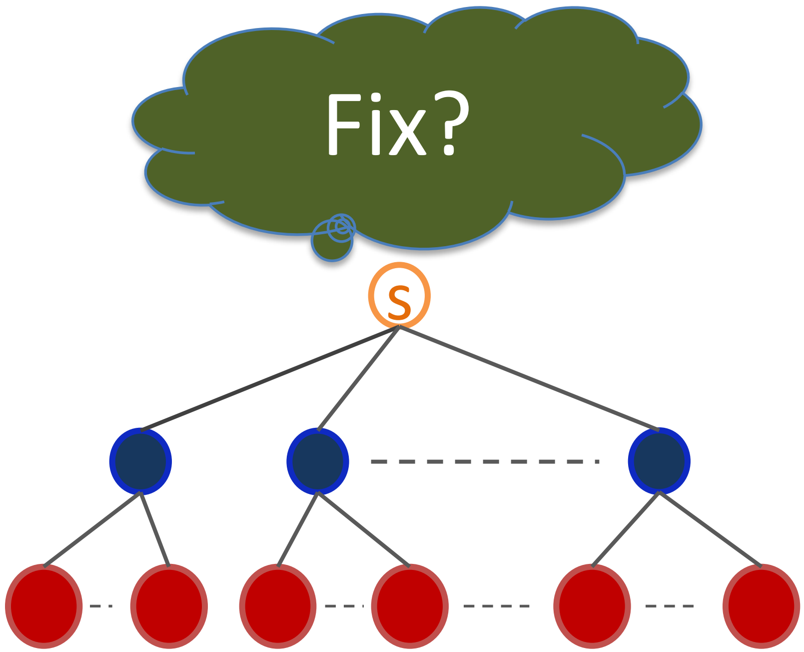
$R[w] = T$

This thing called nested loops

```
R[s] = T and R[u] = F for every u != s

L0 = {s}

L1 , L2 = null

For every u in L0
    For every (u,w) in E
        Add w to L1

For every w in L1
    R[w] = T

For every u in L1
    For every (u,w) in E
        Add w to L2

For every w in L2
    R[w] = T
```

```
R[s] = T and R[u] = F for every u != s

i = 0

L[i] = {s}

While i < 2

    L[i+1] = null
    For every u in L[i]
        For every (u,w) in E
            Add w to L[i+1]

            R[w] = T
    i++
```

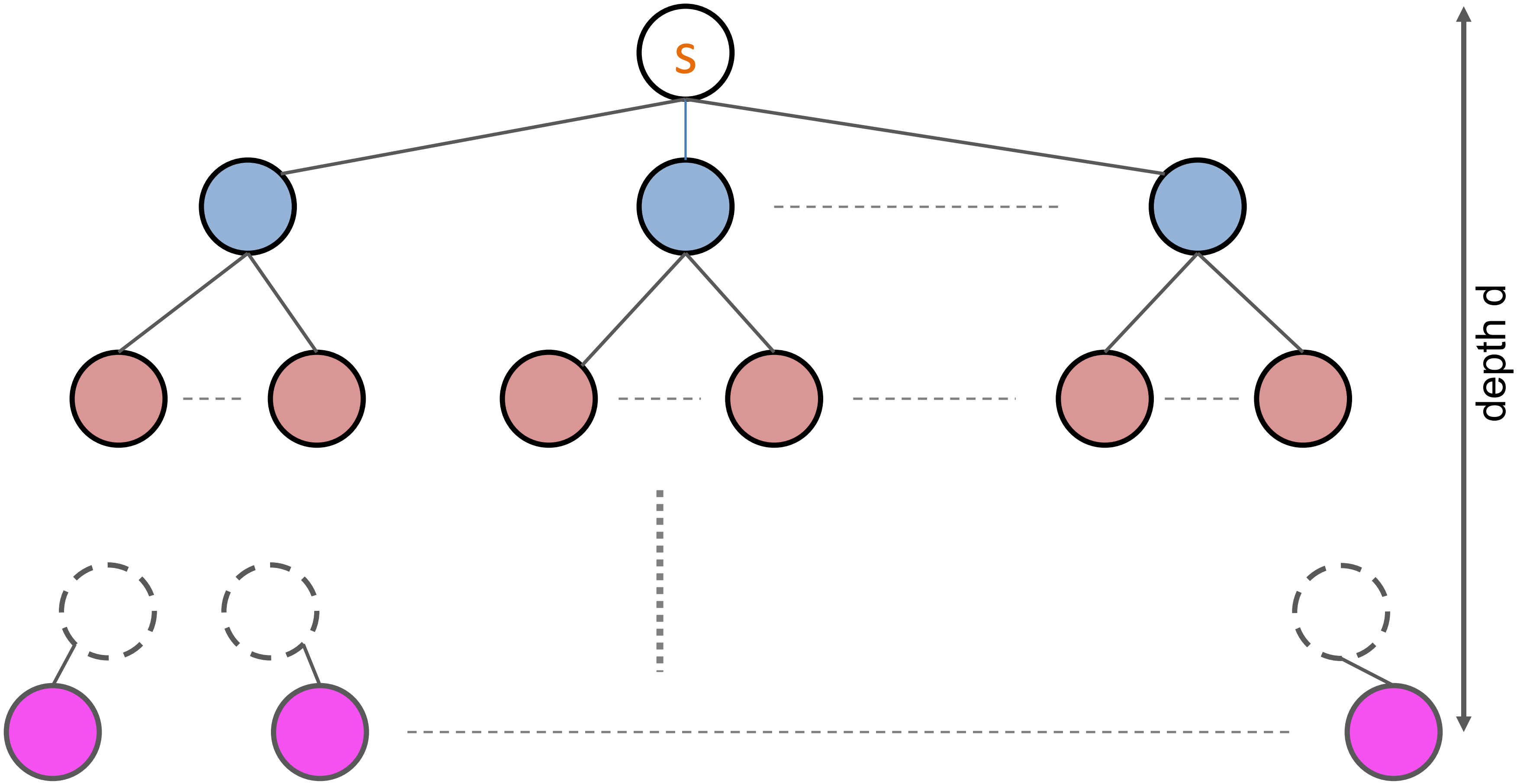
What other extra work is going on?



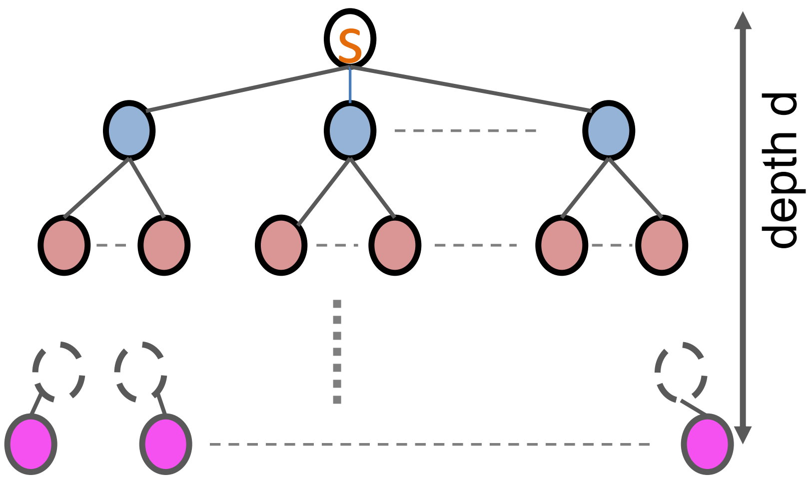
R [blue nodes] is set multiple times

```
R[s] = T and R[u] = F for every u != s

i = 0

L[i] = {s}

While i < 2

    L[i+1] = null
    For every u in L[i]
        For every (u,w) in E
            Add w to L[i+1]

            R[w] = T
    i++
```

Example 3



Algo for trees



R[s] = T and R[u] = F for every u != s

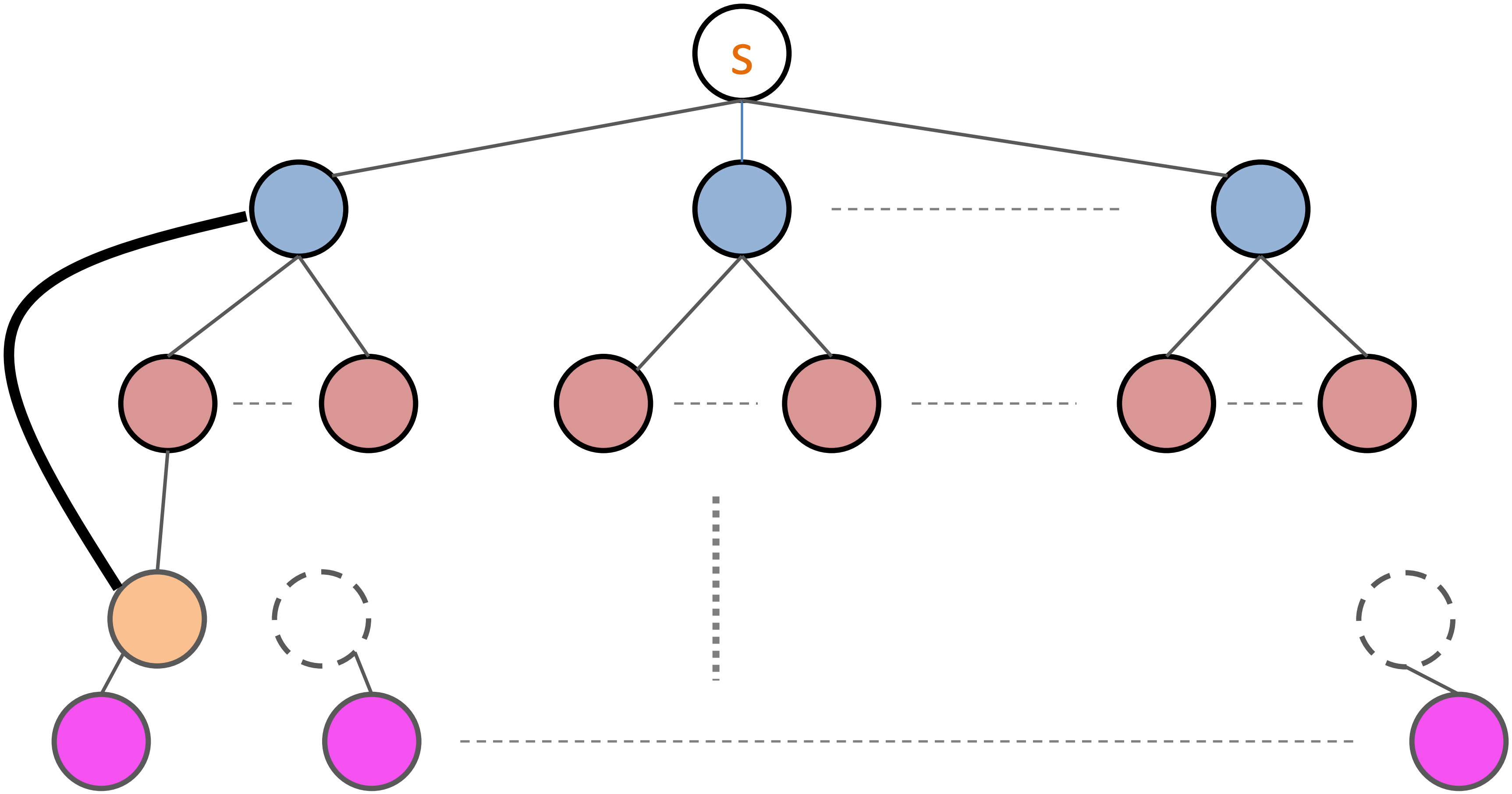
i = 0

L[i] = {s}

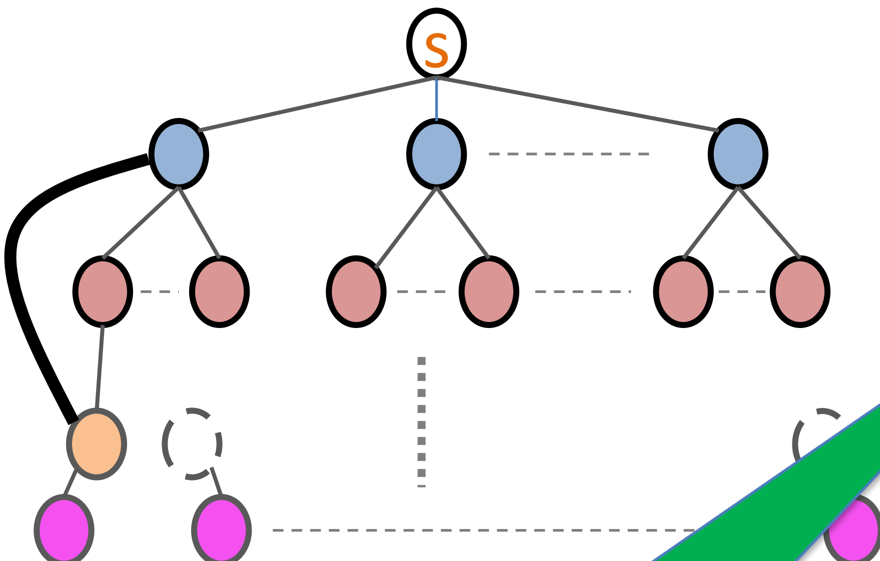
While i < d

    L[i+1] = null
    For every u in L[i]
        For every (u,w) in E
            Add w to L[i+1]
            R[w] = T
    i++

Example 4



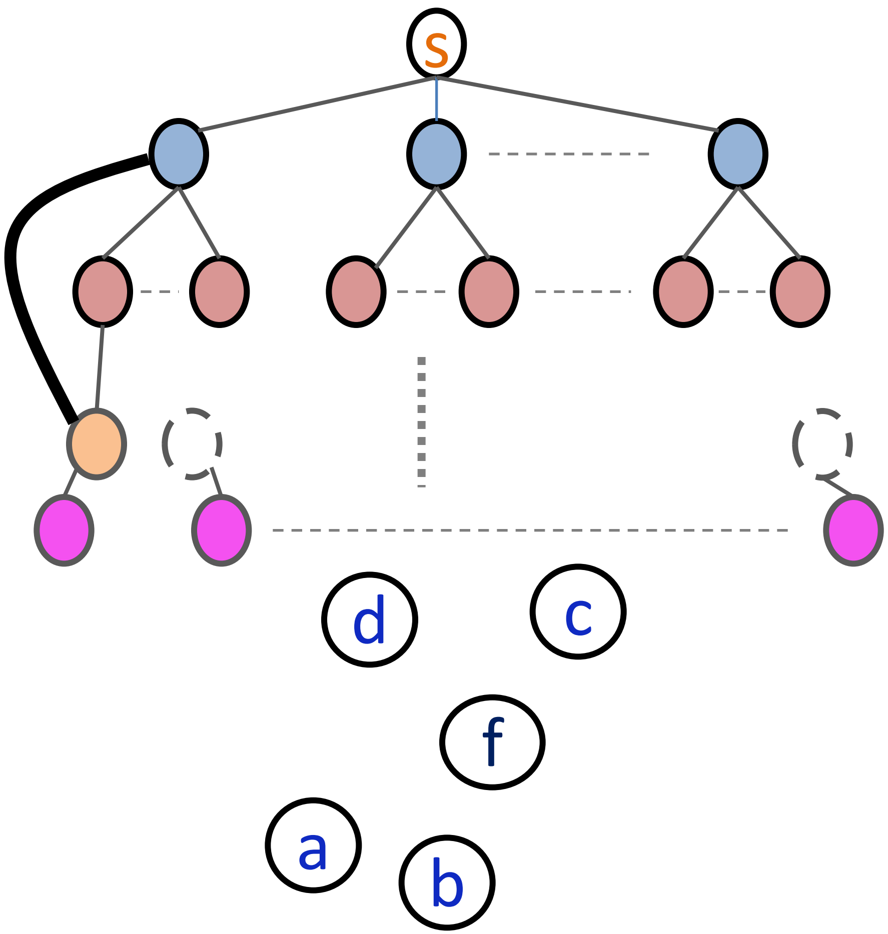
Algo for tree + one edge




```
R[s] = T and R[u] = F for every u != s

i = 0

L[i] = {s}

While i < d

    L[i+1] = null
    For every u in L[i]
        For every (u,w) in E
            Add w to L[i+1]

            R[w] = T
    i++
```

How would you change the loop condition?

How about this?



R[s] = T and R[u] = F for every u != s

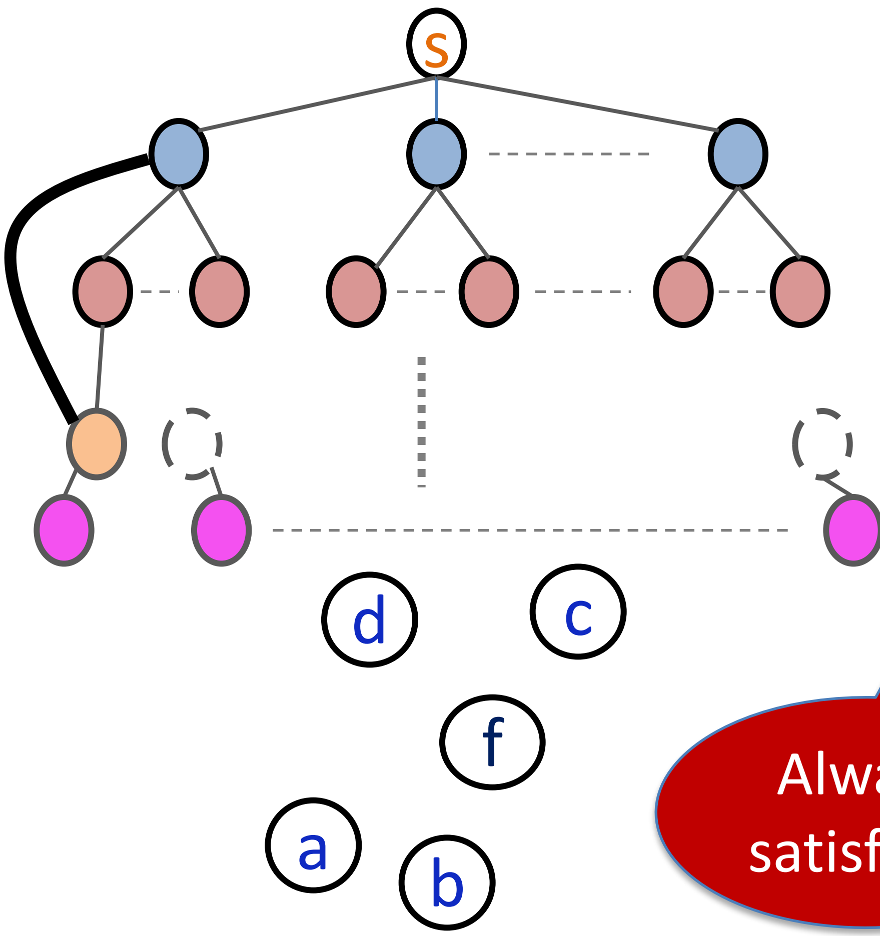
i = 0

L[i] = {s}

While i < d there is an u s.t. R[u] == F

    L[i+1] = null
    For every u in L[i]
        For every (u,w) in E
            Add w to L[i+1]
            R[w] = T
    i++

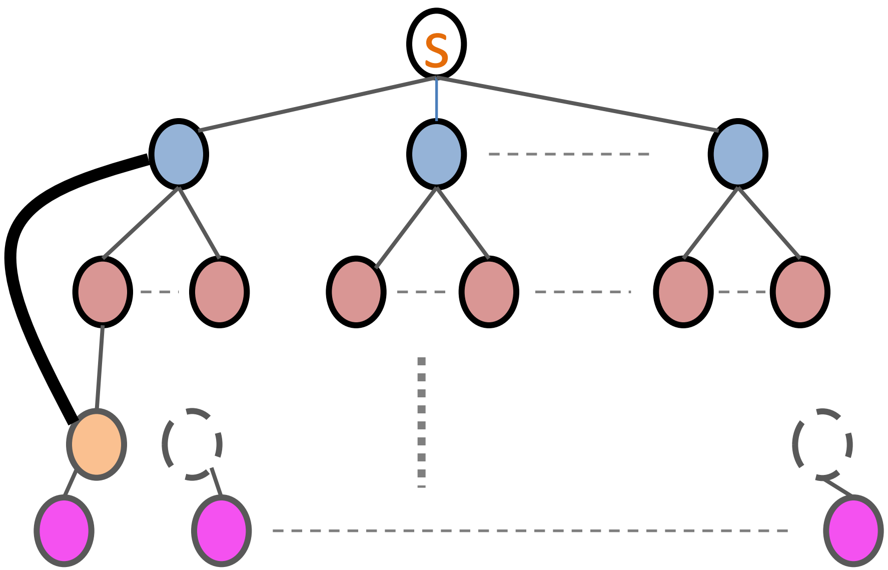
OK, fine-- how about this?




```
R[s] = T and R[u] = F for every u != s

i = 0

L[i] = {s}

While L[i] != null

    L[i+1] = null
    For every u in L[i]
        For every (u,w) in E
            Add w to L[i+1]

            R[w] = T
    i++
```

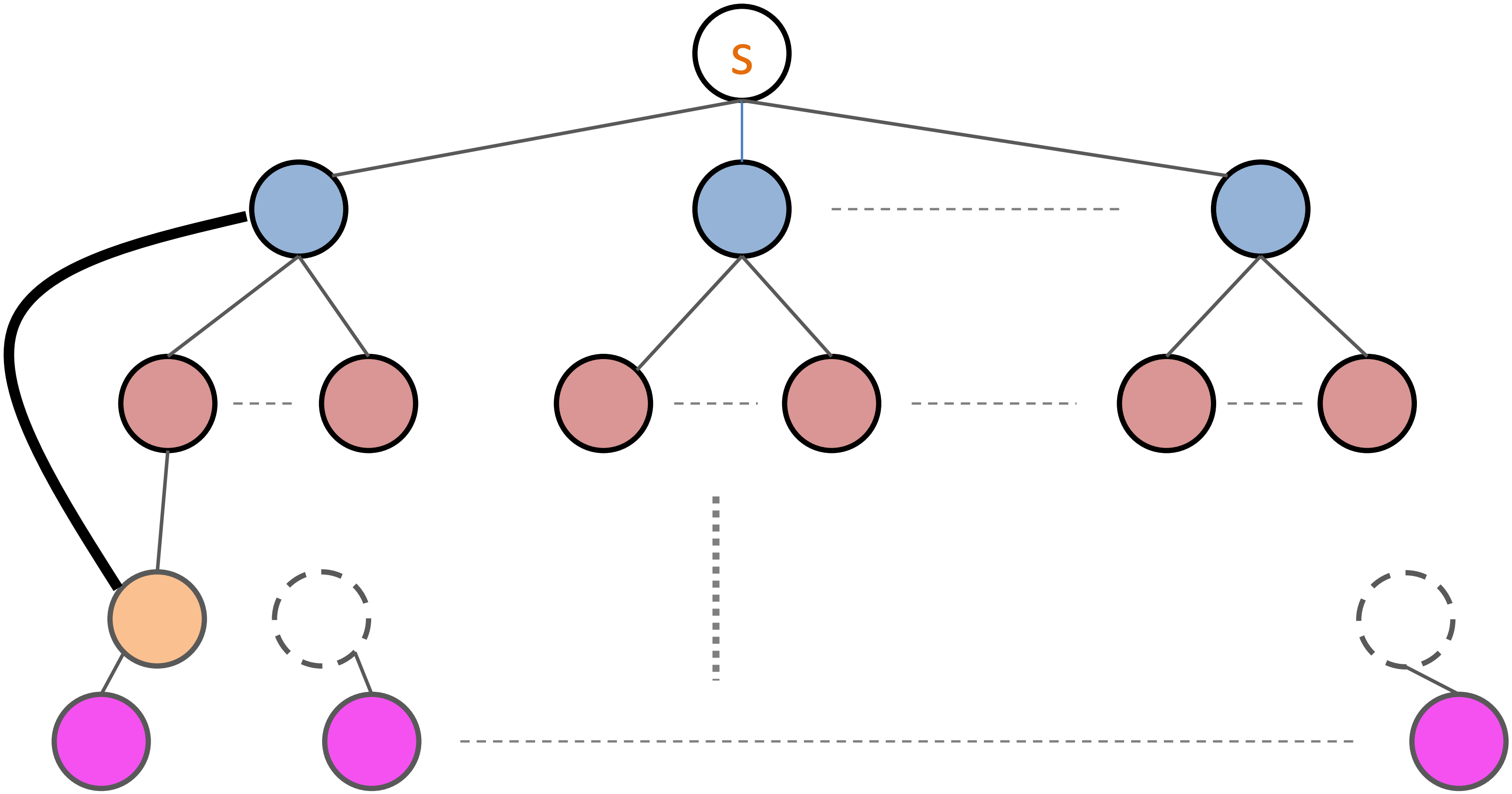
Always satisfied!

A simple fix

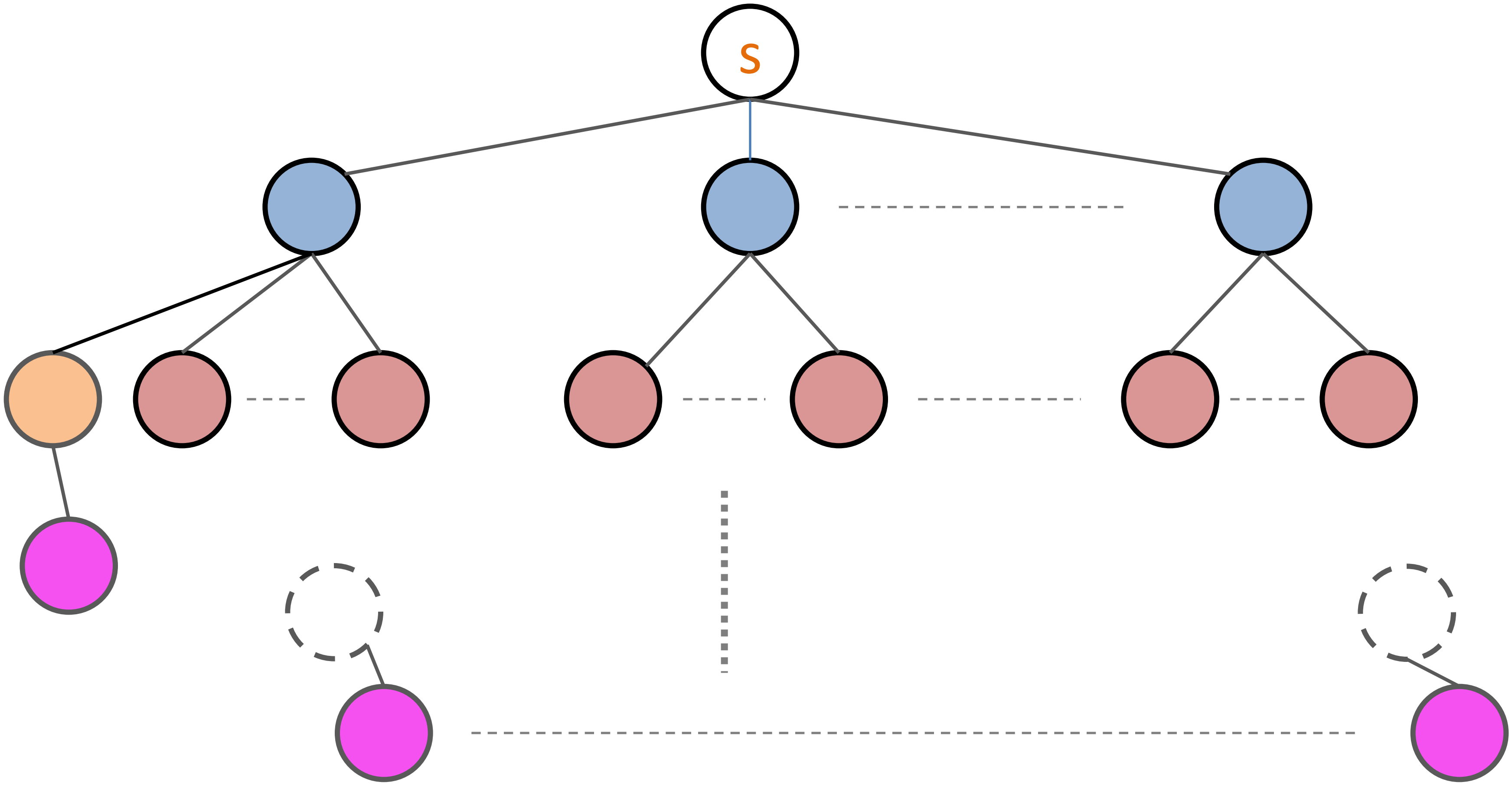



```
R[s] = T and R[u] = F for every u != s

i = 0

L[i] = {s}

While L[i] != null

    L[i+1] = null
    For every u in L[i]
        For every (u,w) in E
            If R[w] == F
                Add w to L[i+1]
                R[w] = T
    i++
```

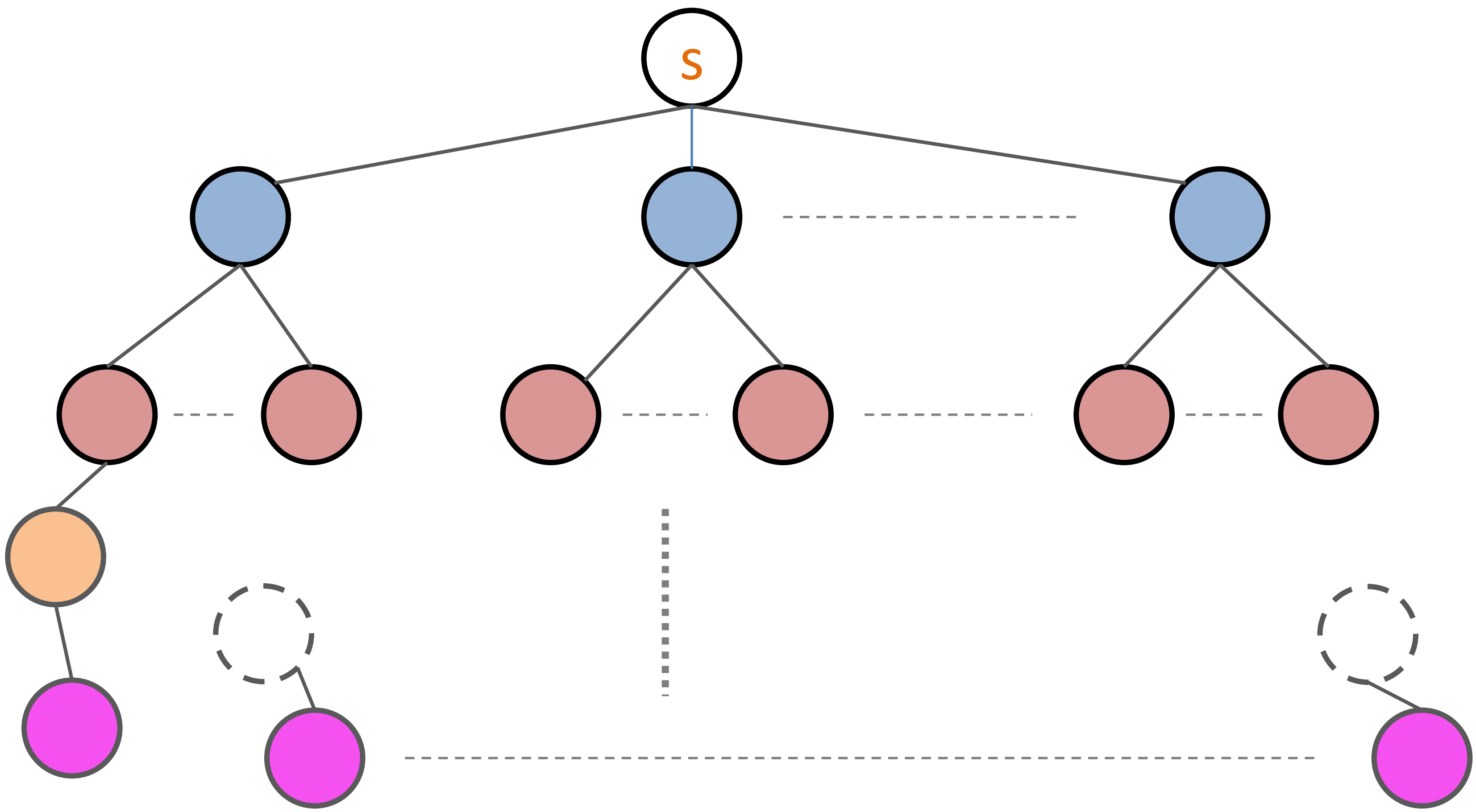
Example 4 (again)



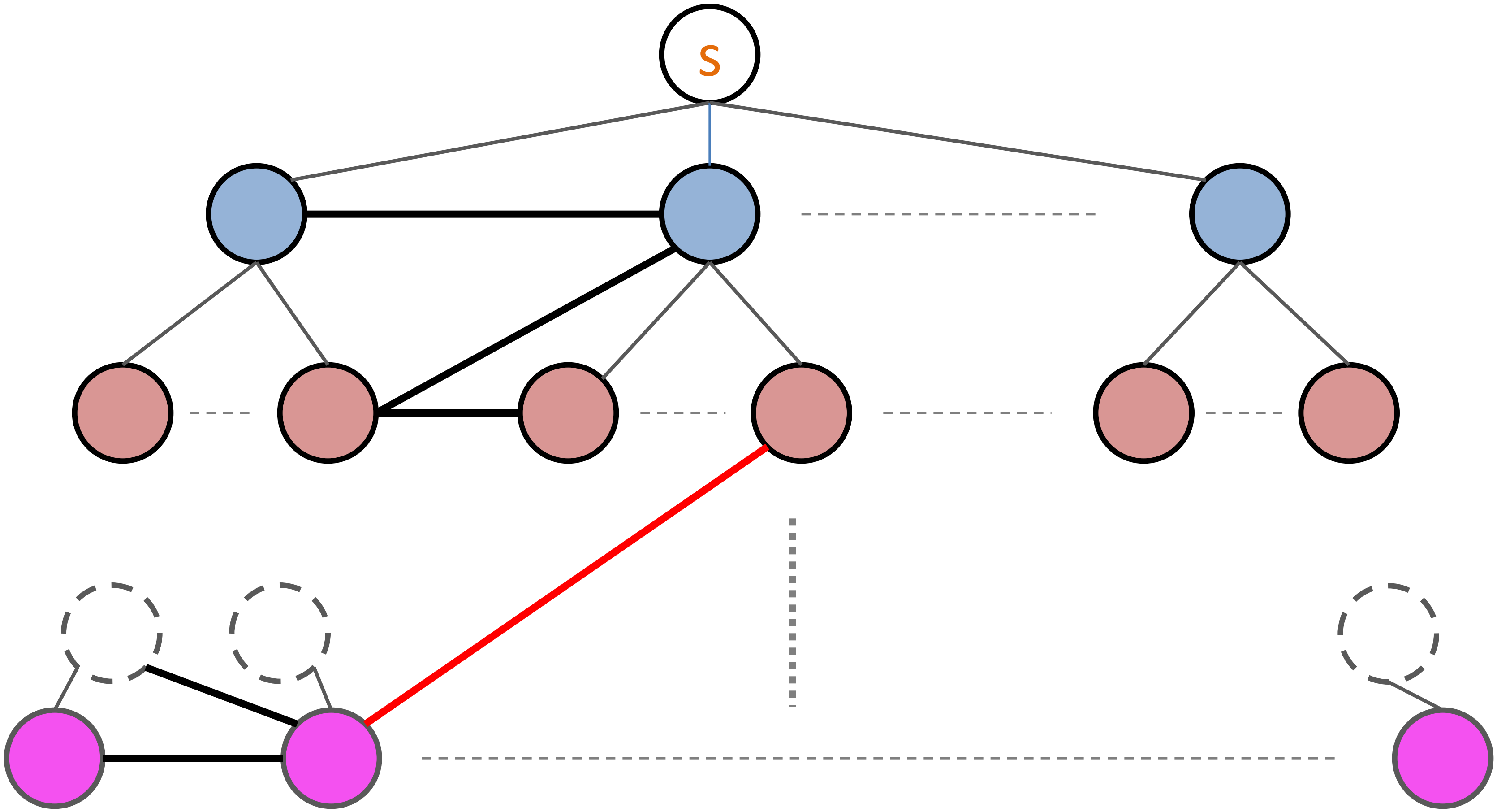
Orange node gets added to L[2]



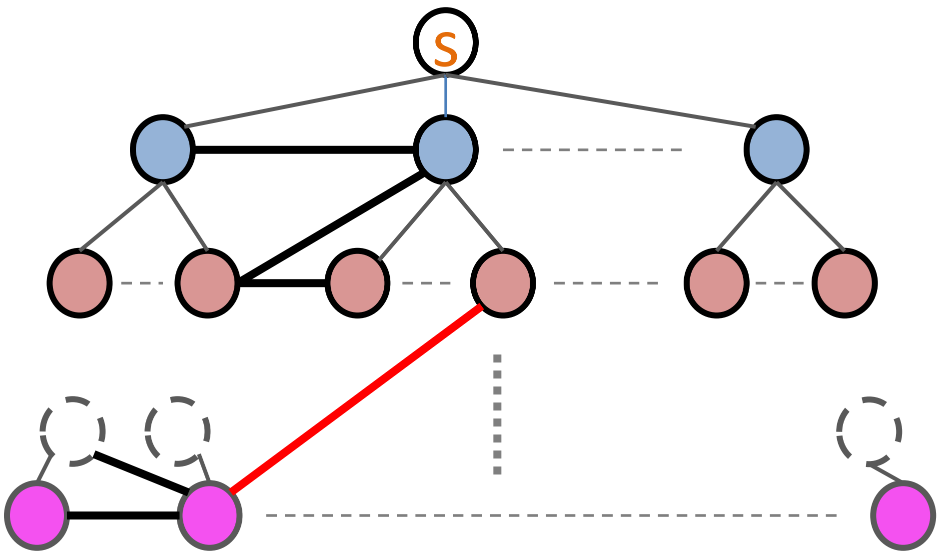
This would have worked too



General Case



Example where this fails?



There is none!

```
R[s] = T and R[u] = F for every u != s

i = 0

L[i] = {s}

While L[i] != null

    L[i+1] = null
    For every u in L[i]
        For every (u,w) in E
            If R[w] == F
                Add w to L[i+1]
                R[w] = T
    i++
```

Questions?

There are notes

Breadth First Search (BFS): Algo Idea

Build layers of vertices connected to s

$L_0 = \{s\}$

Assume $L_0,..,L_j$ have been constructed

L_{j+1} set of vertices not chosen yet but are connected to L_j

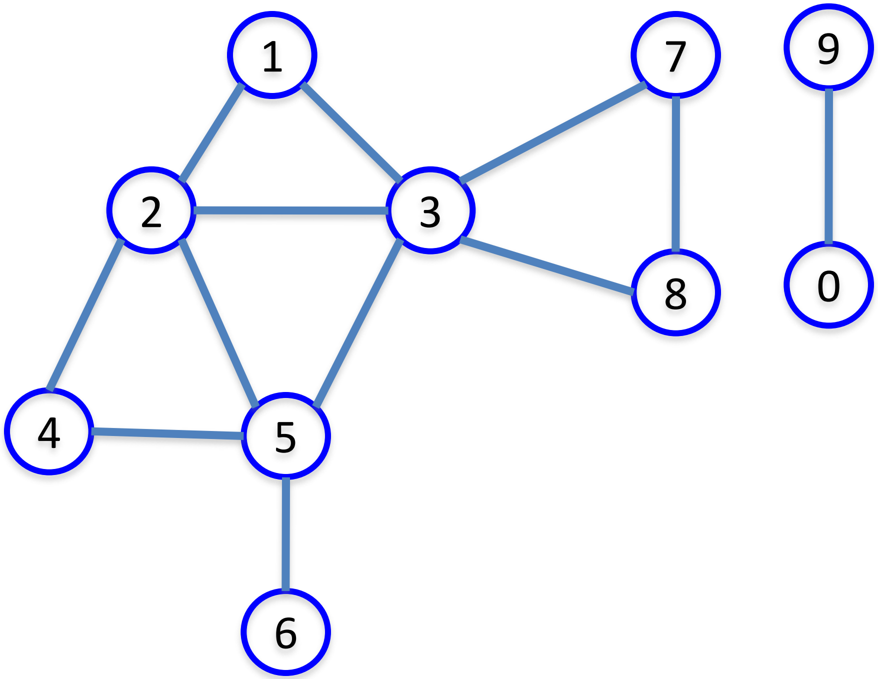
Stop when new layer is empty

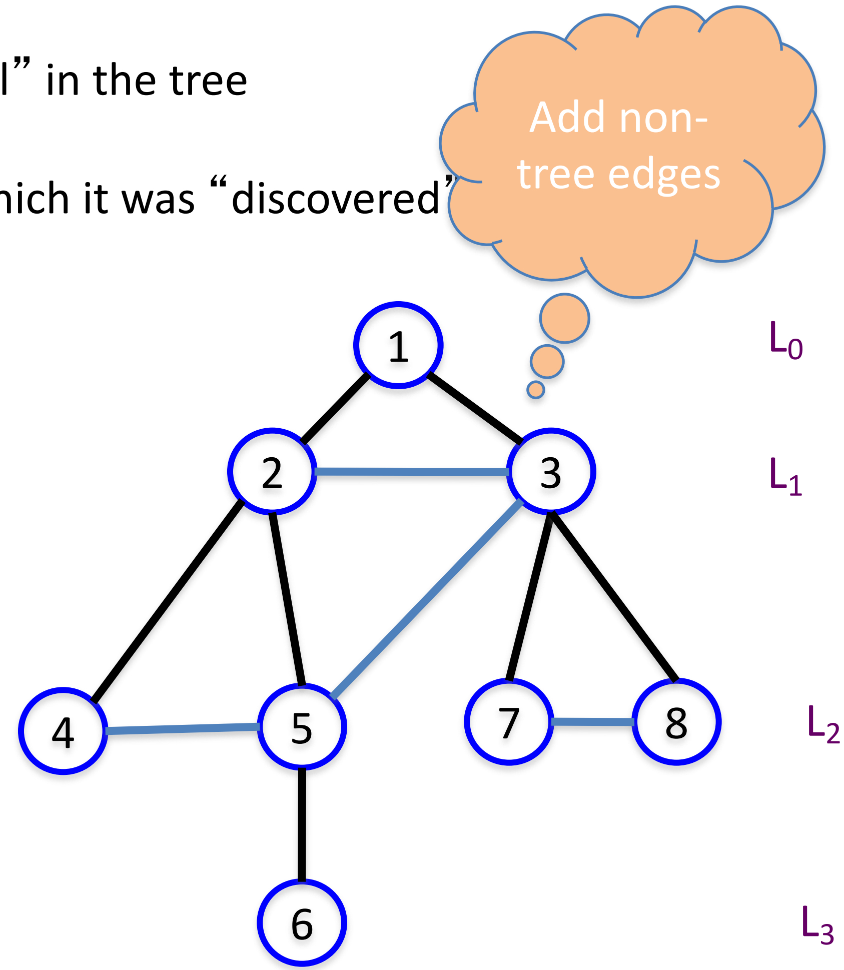
BFS Tree

BFS naturally defines a tree rooted at s

L_j forms the jth "level" in the tree

u in L_{j+1} is child of v in L_j from which it was "discovered"





Rest of today’s agenda

Every edge in is between consecutive layers

Computing Connected component