Lecture 14

CSE 331

Sep 29, 2017

Mini Project Pitch due WED

note

stop following 172 views

Actions

You can submit mini project reports now

You can now submit your mini project reports now. It is due in a bit over 2 weeks: by **11:59pm on Wed, Oct 4**.

The mini-project page has all the details on what is needed in the report.

Some important points:

- If you do not register your group by 11:59pm on Monday, you will get an automatic 0 on the entire mini-project.
- The case-studies will be assigned in the order in which I grade your reports.
  - If while grading it turns out another group has already taken your case study I will ask you to choose another case study.
  - If you want to "book" your topic sooner, I would recommend that you submit your report as soon as it is ready and send me email saying it is ready to be graded.
  - By default I will start grading on Oct 5.
- This is a **group submission**. Please see the instructions at the end of this post.
  - Main thing: do **NOT** submit your report till your group is formed.

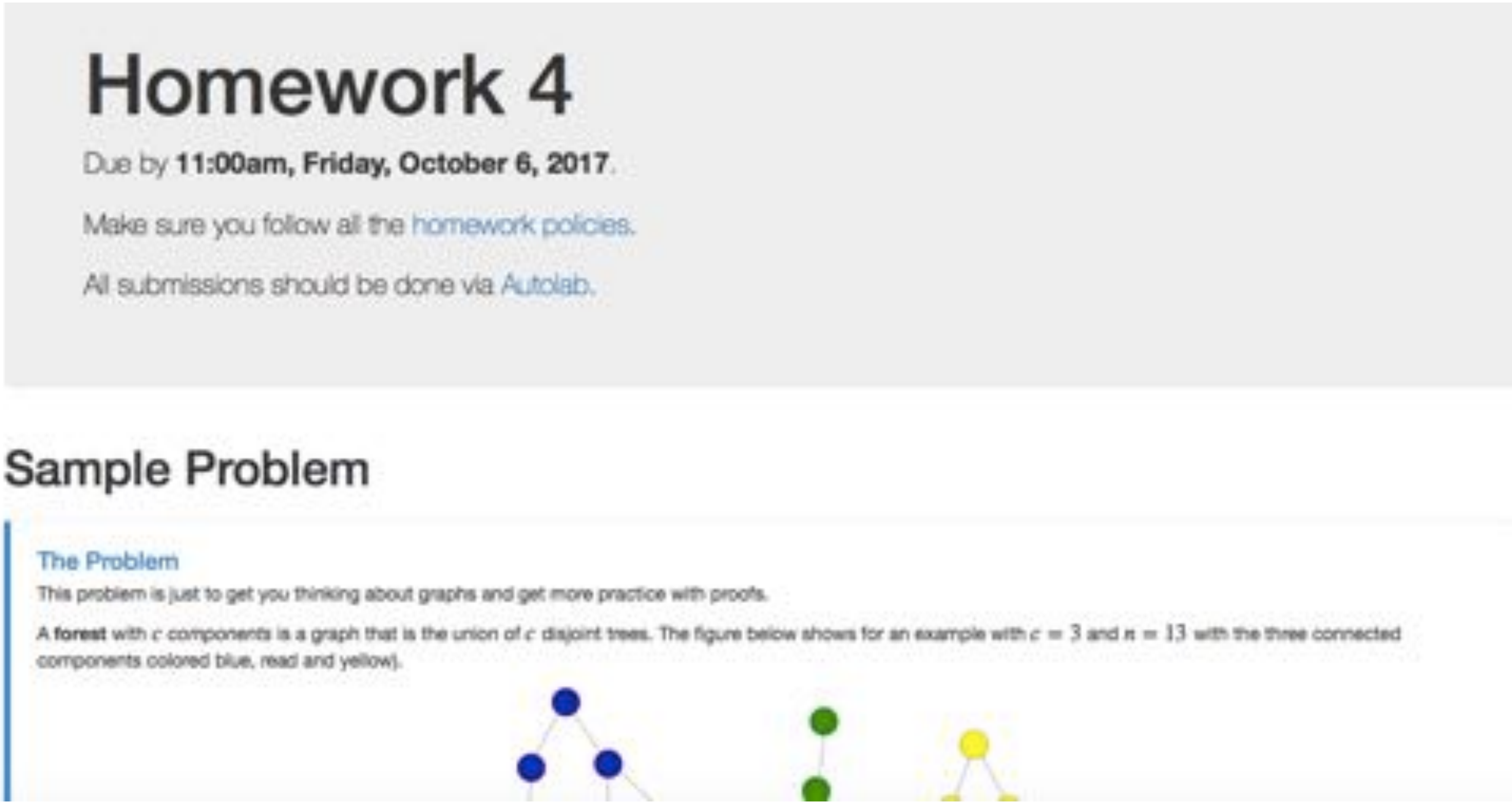
Form your group on Autolab BEFORE submitting your pitch

=========== Instructions on forming the group ========

- Under "Options" click on "Group Options"
- Name your group if you want (not required)
- Enter the name of the 2nd person in your group and then click on "Create Group". (Unless things have changed, Autolab does

Do not forget to add URL to your references

HW 4 is now posted



Homework 4

Due by **11:00am, Friday, October 6, 2017**.

Make sure you follow all the homework policies.

All submissions should be done via Autolab.

Sample Problem

The Problem

This problem is just to get you thinking about graphs and get more practice with proofs.

A **forest** with c components is a graph that is the union of c disjoint trees. The figure below shows for an example with $c = 3$ and $n = 13$ with the three connected components colored blue, read and yellow).

Note: Bonus points for the fastest submissions. See **WARNING** though.

Today's agenda

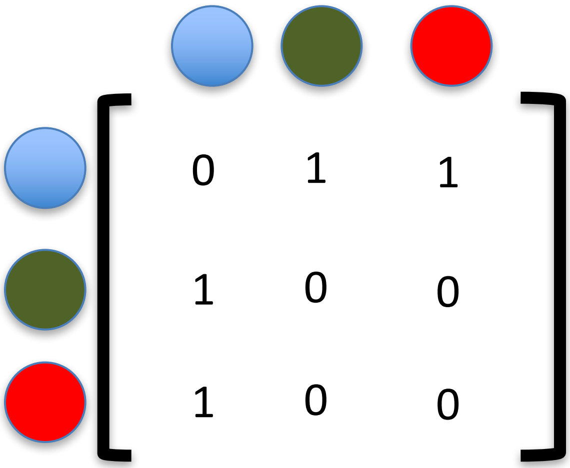
Run-time analysis of BFS (DFS)

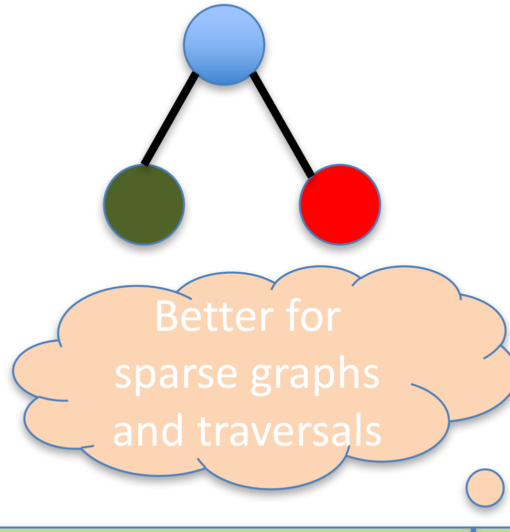
Stacks and Queues

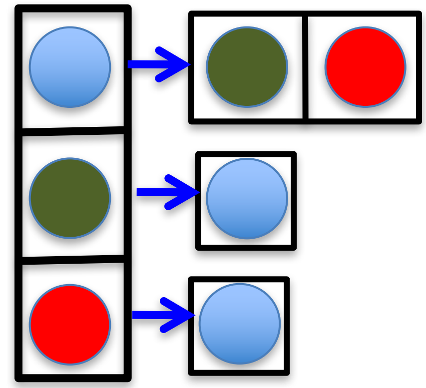
Last in First out

First in First out

Graph representations







Adjacency matrix		Adjacency List
$O(1)$	(u,v) in E?	$O(n)$ [$O(n_v)$]
$O(n)$	All neighbors of u?	$O(n_u)$
$O(n^2)$	Space?	$O(m+n)$

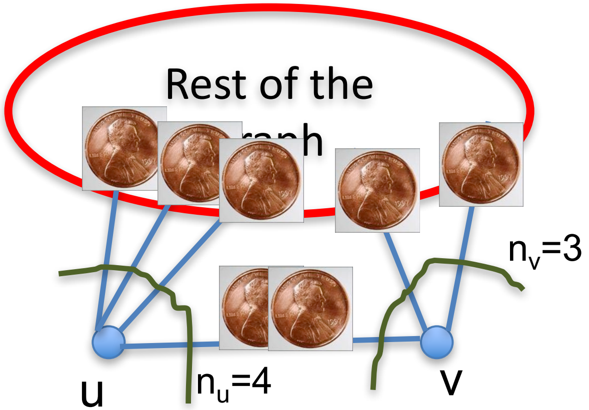
Questions?

2 # edges = sum of # neighbors

$$2m = \Sigma_{u \text{ in } V}\, n_u$$

Give 2 pennies to each edge

Total # of pennies = 2m



Each edges gives one penny to its end points

# of pennies u receives = n_u

Breadth First Search (BFS)

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

Use linked lists

Use CC[v] array

Rest of Today’s agenda

Quick run time analysis for BFS

Quick run time analysis for DFS (and Queue version of BFS)

Helping you schedule your activities for the day

O(m+n) BFS Implementation

BFS(s)

Array

Input graph as Adjacency list

CC[s] = T and CC[w] = F for every w≠ s

Set i = 0

Set L_0= {s}

While L_i is not empty

Linked List

$L_{i+1} = Ø$

For every u in L_i

For every edge (u,w)

If CC[w] = F then

CC[w] = T

Add w to L_{i+1}

i++

Version in KT also computes a BFS tree

All the layers as one

BFS(s)

$CC[s] = T$ and $CC[w] = F$ for every $w \neq s$

Set $i = 0$

Set $L_0 = \{s\}$

While L_i is not empty

    $L_{i+1} = \emptyset$

    For every u in L_i

        For every edge (u,w)

        If $CC[w] = F$ then

            $CC[w] = T$

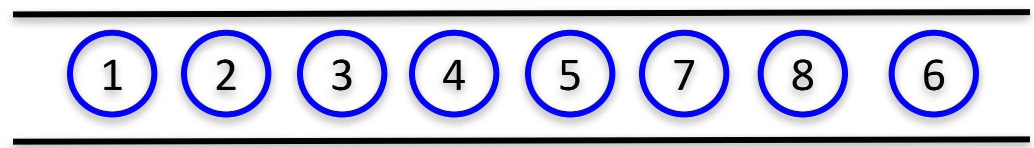
            Add w to L_{i+1}

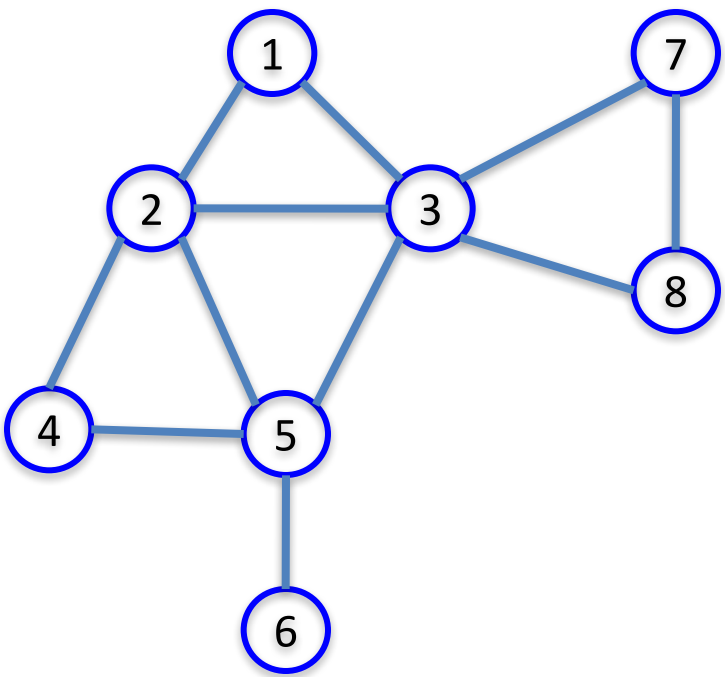
    i++

All layers are considered in first-in-first-out order

Can combine all layers into one queue: all the children of a node are added to the end of the queue

An illustration





Queue O(m+n) implementation

BFS(s)

```
CC[s] = T and CC[w] = F for every w≠ s
Intitialize Q= {s}
While Q is not empty
    Delete the front element u in Q
        For every edge (u,w)
            If CC[w] = F then
                CC[w] = T
                Add w to the back of Q
```

$O(n)$

$O(1)$

$\Sigma_u O(n_u) = O(\Sigma_u n_u) = O(m)$

$O(1)$

$O(n_u)$

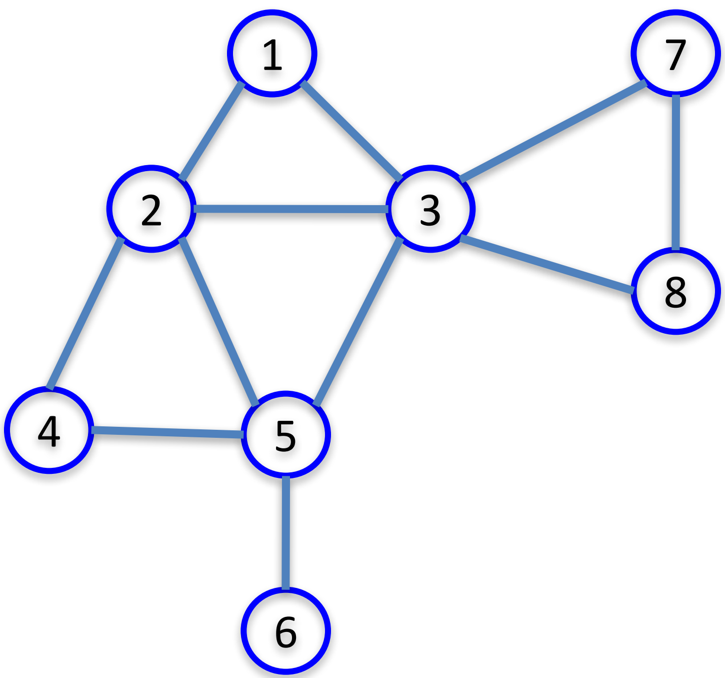
$O(1)$

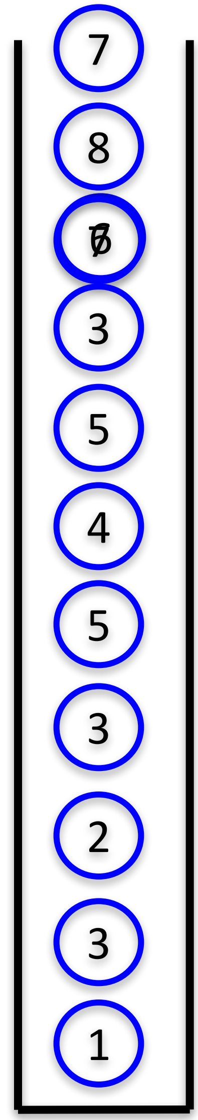
Questions?

Implementing DFS in O(m+n) time

Same as BFS except stack instead of a queue

A DFS run using an explicit stack





DFS stack implementation

DFS(s)

CC[s] = T and CC[w] = F for every w≠ s

Intitialize Ŝ = {s}

While Ŝ is not empty

Pop the top element u in Ŝ

For every edge (u,w)

If CC[w] = F then

CC[w] = T

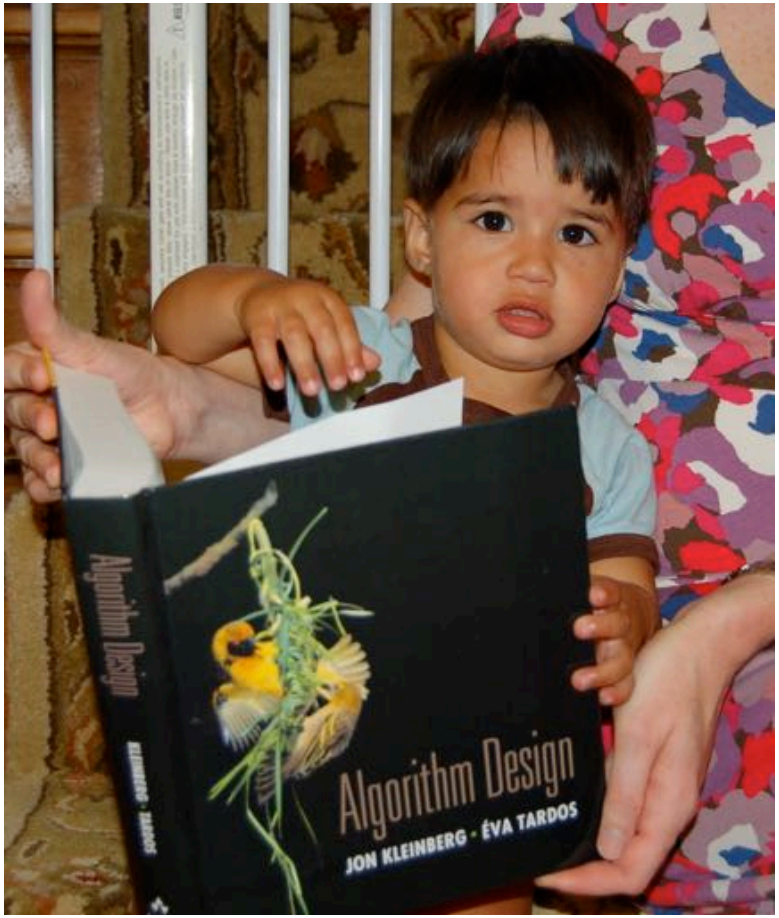
Push w to the top of Ŝ

Same O(m+n) run time analysis as for BFS

Questions?

Reading Assignment

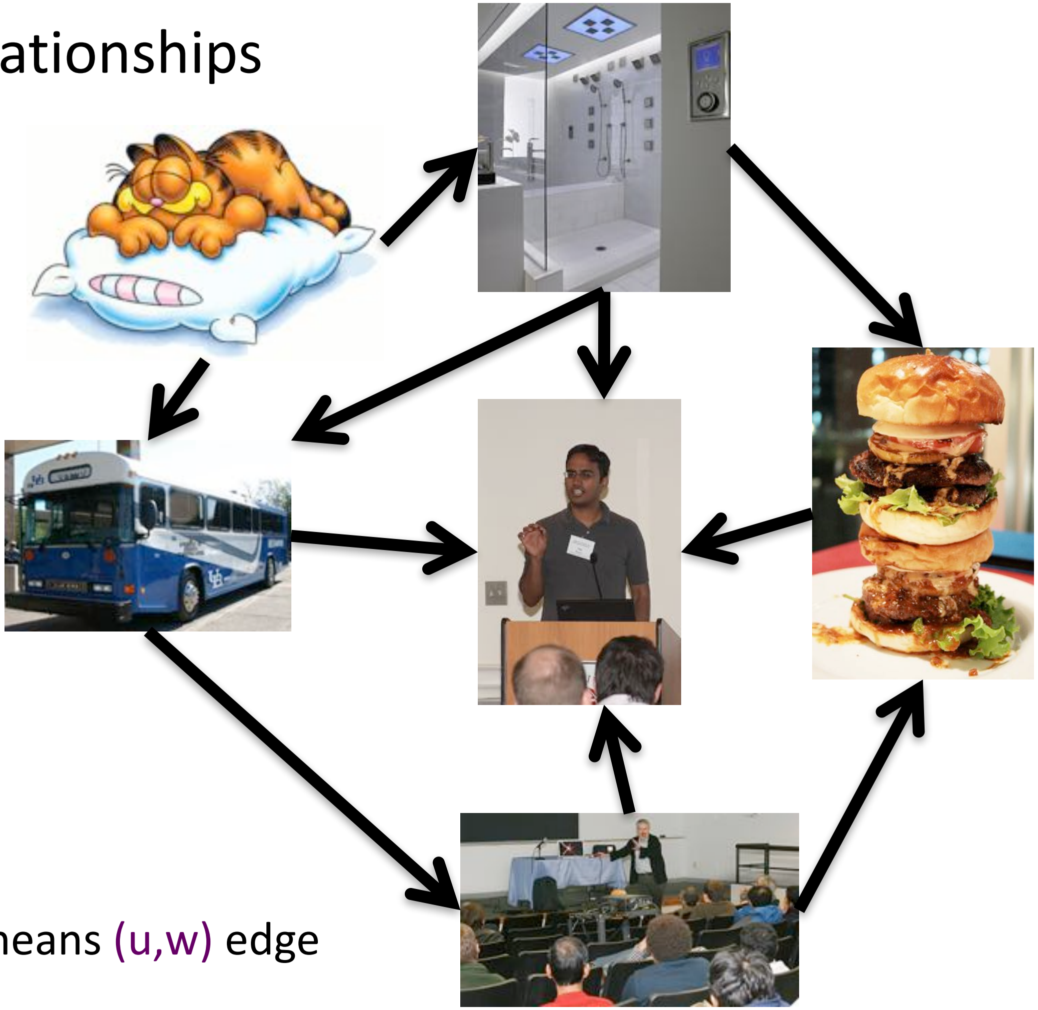
Sec 3.3, 3.4 and 3.5 of [KT]



Directed graphs

Model asymmetric relationships

Precedence relationships

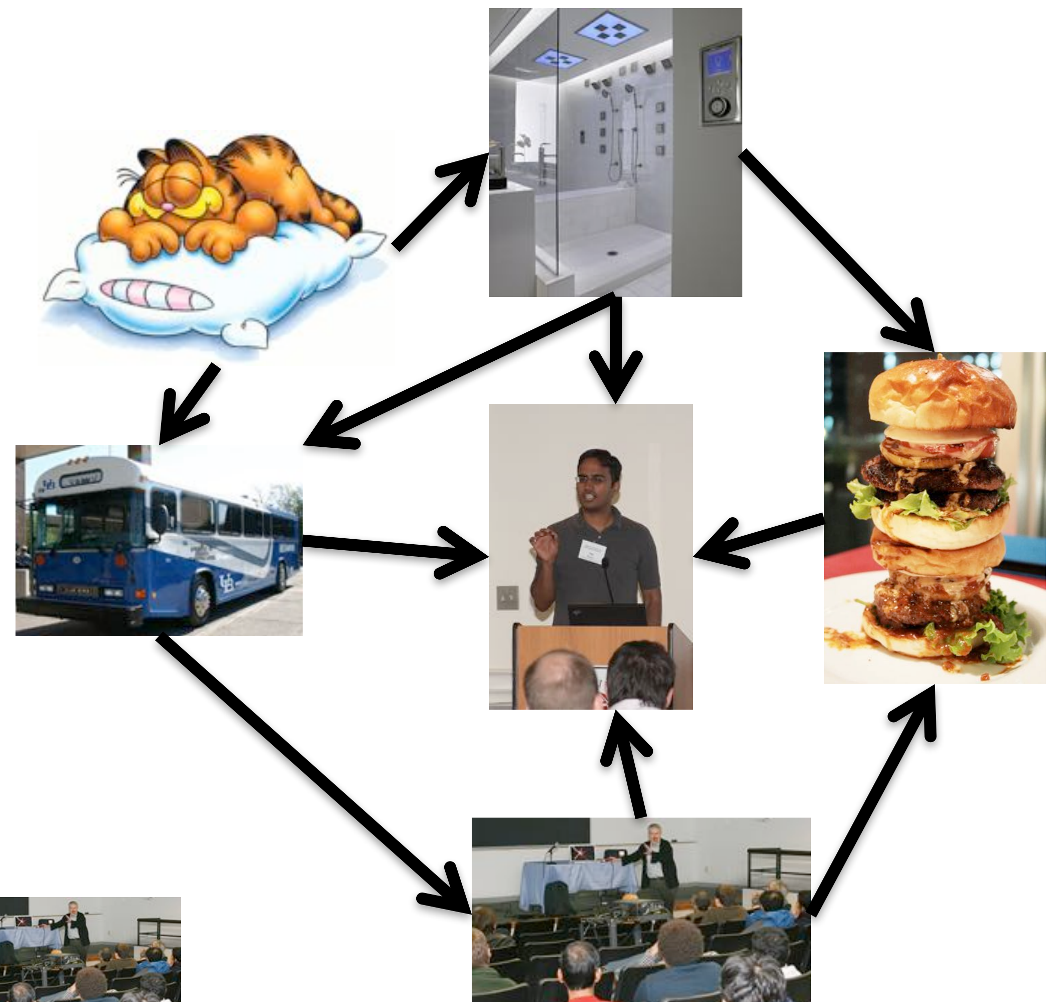


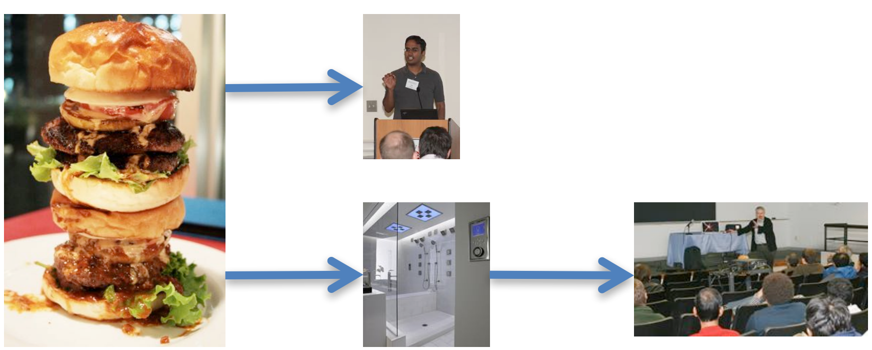
u needs to be done before w means (u,w) edge

Directed graphs

Adjacency matrix is not symmetric

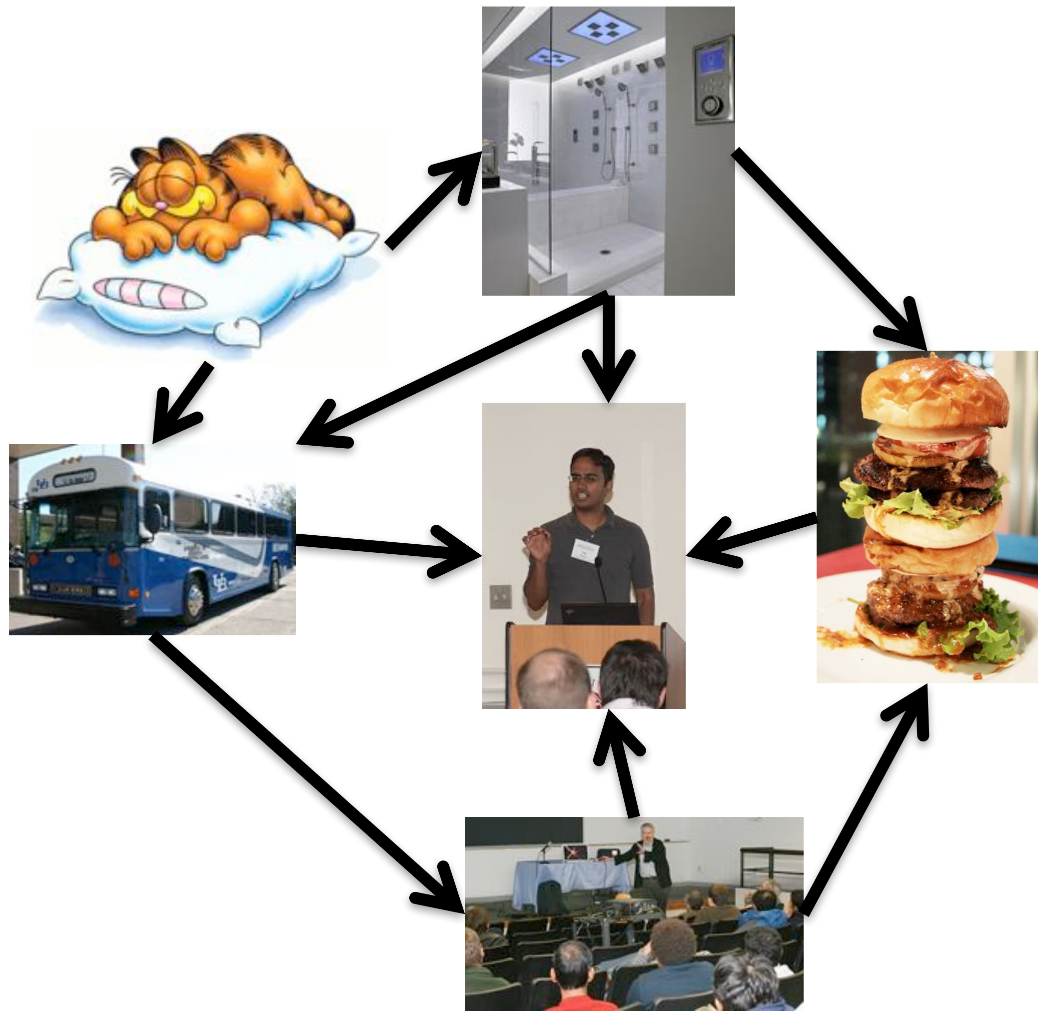
Each vertex has two lists in Adj. list rep.





Directed Acyclic Graph (DAG)

No directed cycles



Precedence relationships are consistent

Topological Sorting of a DAG

Order the vertices so that all edges go "forward"