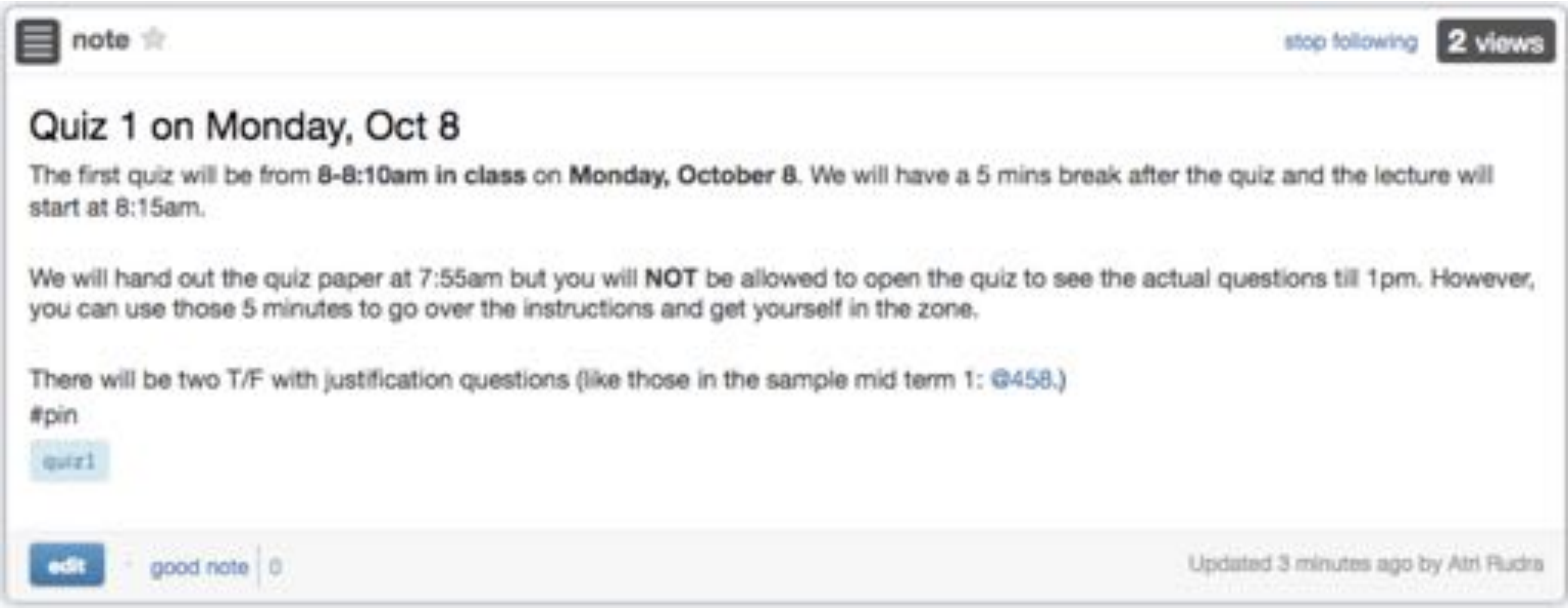
Lecture 15

CSE 331

Oct 1, 2018

Quiz 1 next Monday



Quiz 1 on Monday, Oct 8

The first quiz will be from **8-8:10am in class** on **Monday, October 8**. We will have a 5 mins break after the quiz and the lecture will start at 8:15am.

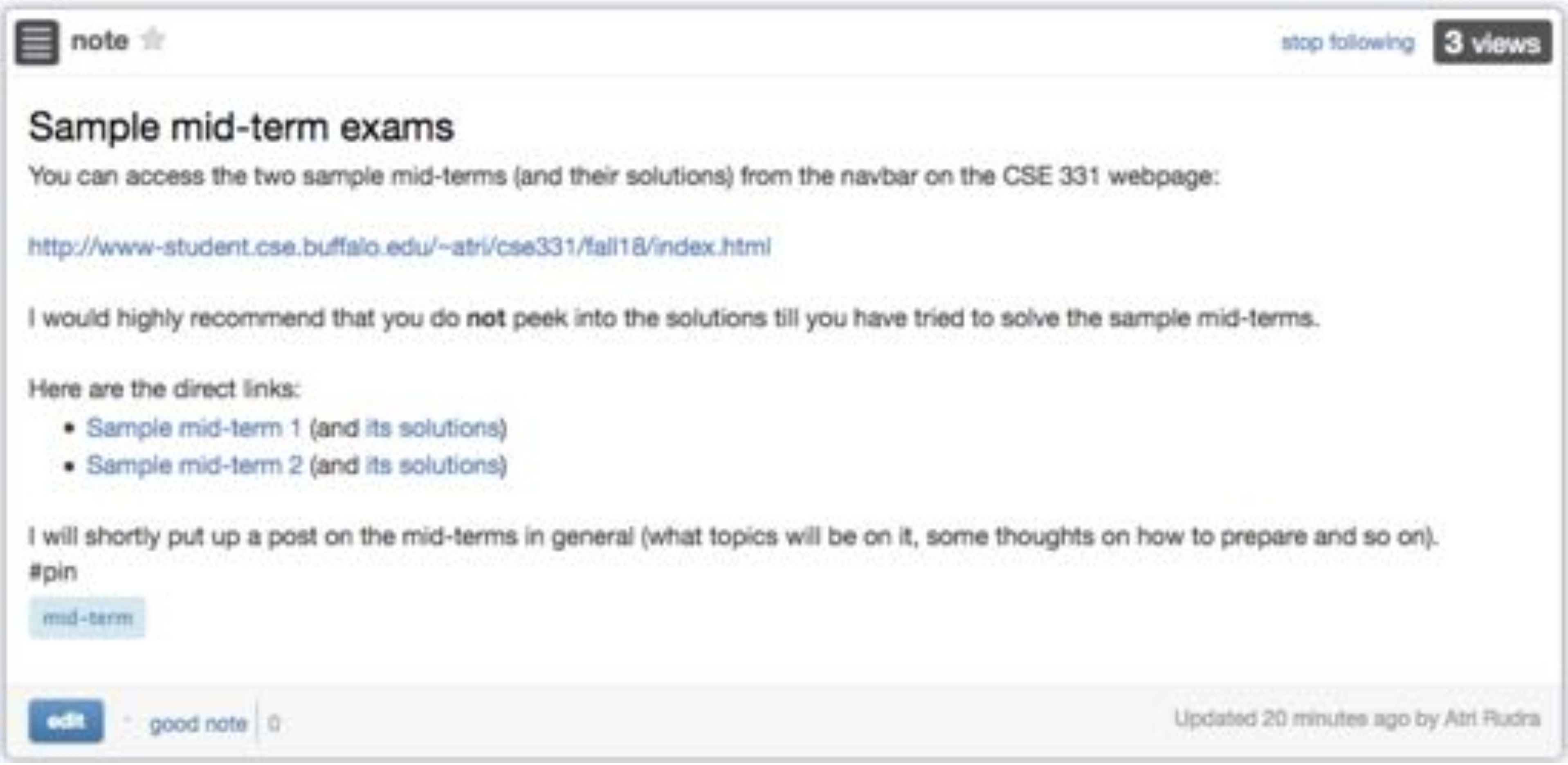
We will hand out the quiz paper at 7:55am but you will **NOT** be allowed to open the quiz to see the actual questions till 1pm. However, you can use those 5 minutes to go over the instructions and get yourself in the zone.

There will be two T/F with justification questions (like those in the sample mid term 1: @458.)
#pin

quiz1

Updated 3 minutes ago by Atri Rudra

Sample mid-terms


note

stop following

3 views

Sample mid-term exams

You can access the two sample mid-terms (and their solutions) from the navbar on the CSE 331 webpage:

http://www-student.cse.buffalo.edu/~atri/cse331/fall18/index.html

I would highly recommend that you do **not** peek into the solutions till you have tried to solve the sample mid-terms.

Here are the direct links:

- Sample mid-term 1 (and its solutions)
- Sample mid-term 2 (and its solutions)

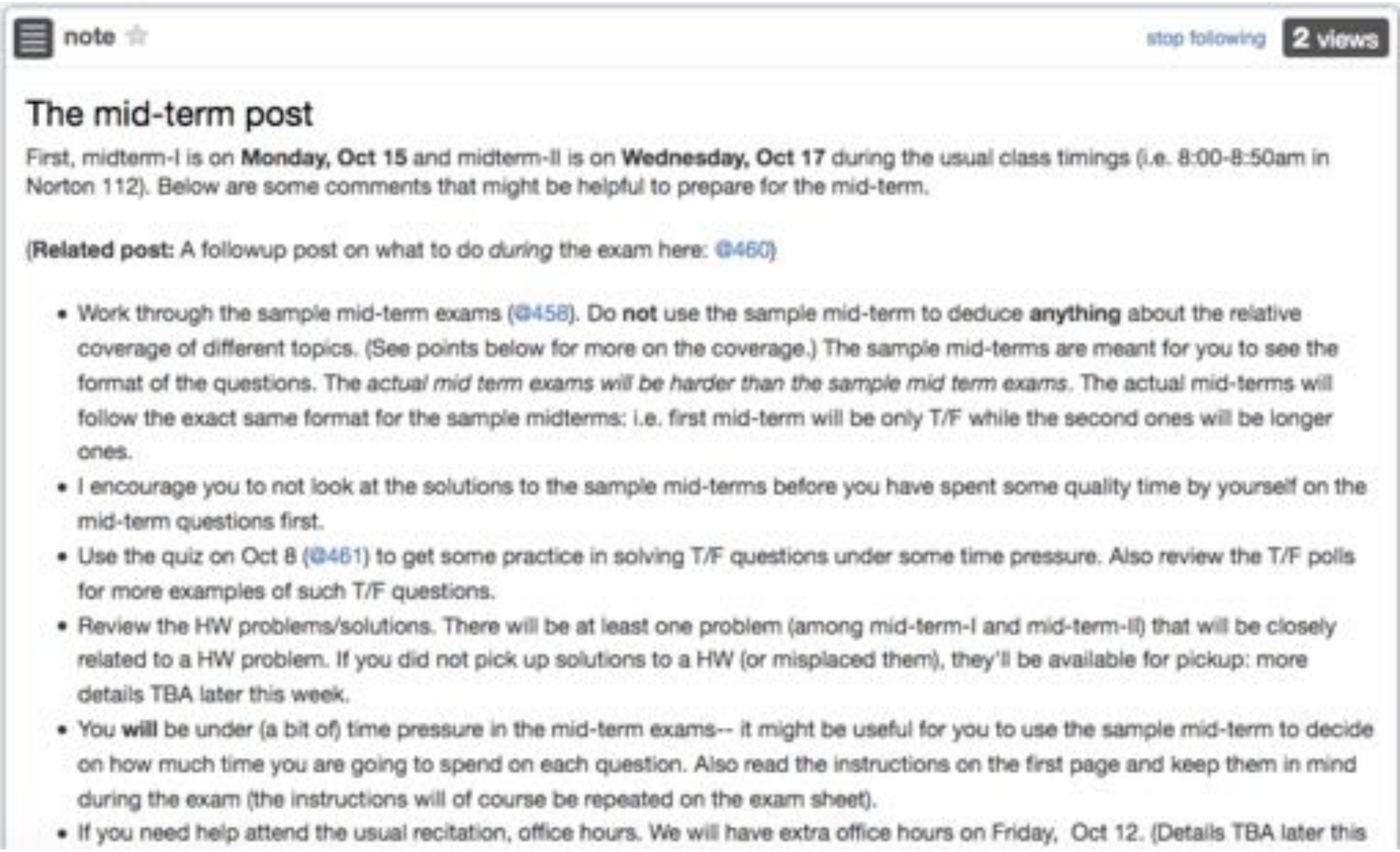
I will shortly put up a post on the mid-terms in general (what topics will be on it, some thoughts on how to prepare and so on).
#pin

mid-term

edit · good note | 0

Updated 20 minutes ago by Atri Rudra

The mid-term post


note

stop following 2 views

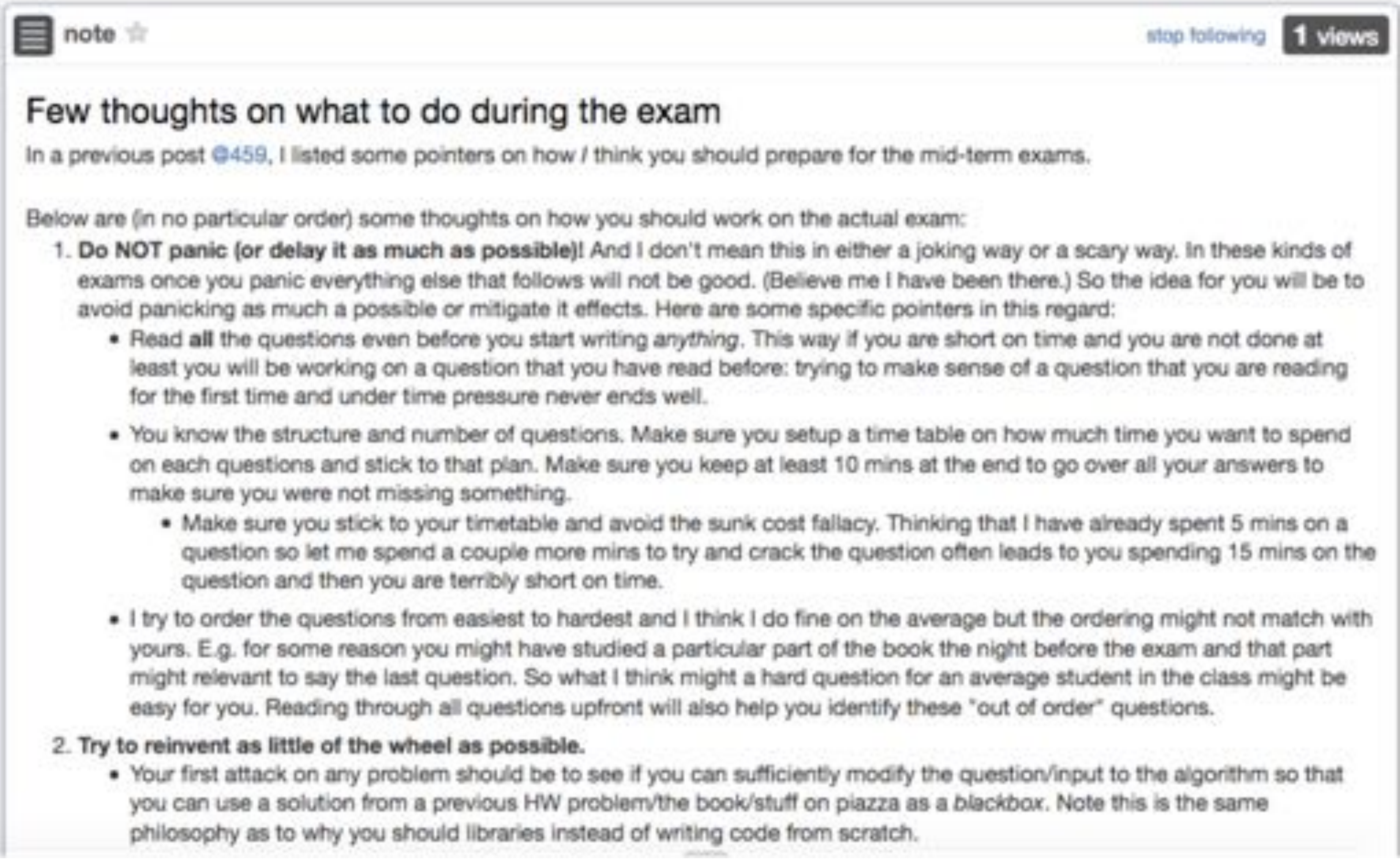
The mid-term post

First, midterm-I is on **Monday, Oct 15** and midterm-II is on **Wednesday, Oct 17** during the usual class timings (i.e. 8:00-8:50am in Norton 112). Below are some comments that might be helpful to prepare for the mid-term.

(**Related post:** A followup post on what to do *during* the exam here: @460)

- Work through the sample mid-term exams (@458). Do **not** use the sample mid-term to deduce **anything** about the relative coverage of different topics. (See points below for more on the coverage.) The sample mid-terms are meant for you to see the format of the questions. The *actual mid term exams will be harder than the sample mid term exams*. The actual mid-terms will follow the exact same format for the sample midterms: i.e. first mid-term will be only T/F while the second ones will be longer ones.
- I encourage you to not look at the solutions to the sample mid-terms before you have spent some quality time by yourself on the mid-term questions first.
- Use the quiz on Oct 8 (@461) to get some practice in solving T/F questions under some time pressure. Also review the T/F polls for more examples of such T/F questions.
- Review the HW problems/solutions. There will be at least one problem (among mid-term-I and mid-term-II) that will be closely related to a HW problem. If you did not pick up solutions to a HW (or misplaced them), they'll be available for pickup: more details TBA later this week.
- You **will** be under (a bit of) time pressure in the mid-term exams-- it might be useful for you to use the sample mid-term to decide on how much time you are going to spend on each question. Also read the instructions on the first page and keep them in mind during the exam (the instructions will of course be repeated on the exam sheet).
- If you need help attend the usual recitation, office hours. We will have extra office hours on Friday, Oct 12. (Details TBA later this

During the exam…



note ☆ stop following 1 views

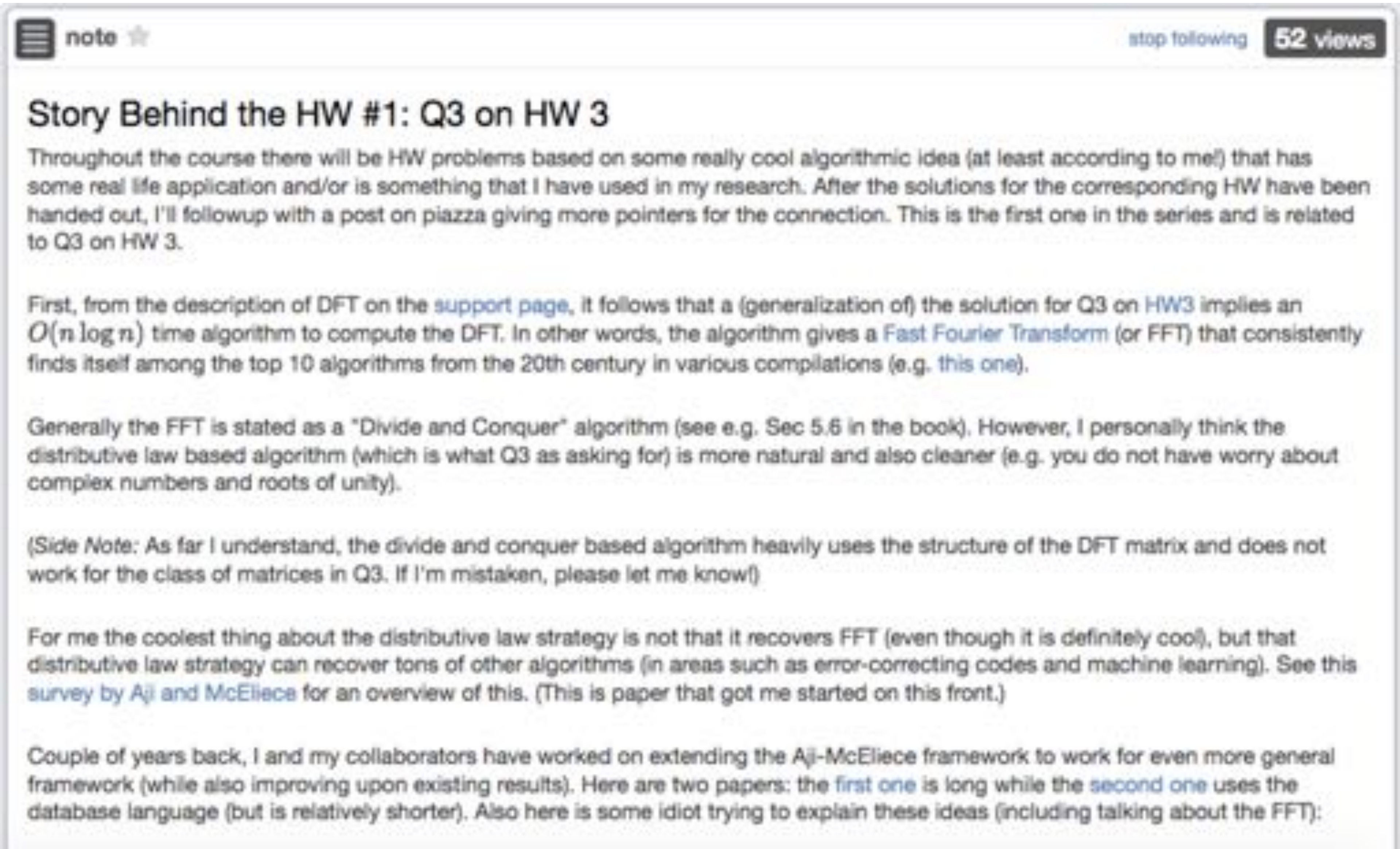
Few thoughts on what to do during the exam

In a previous post @459, I listed some pointers on how *I* think you should prepare for the mid-term exams.

Below are (in no particular order) some thoughts on how you should work on the actual exam:

1. **Do NOT panic (or delay it as much as possible)!** And I don't mean this in either a joking way or a scary way. In these kinds of exams once you panic everything else that follows will not be good. (Believe me I have been there.) So the idea for you will be to avoid panicking as much a possible or mitigate it effects. Here are some specific pointers in this regard:
   - Read **all** the questions even before you start writing *anything*. This way if you are short on time and you are not done at least you will be working on a question that you have read before: trying to make sense of a question that you are reading for the first time and under time pressure never ends well.
   - You know the structure and number of questions. Make sure you setup a time table on how much time you want to spend on each questions and stick to that plan. Make sure you keep at least 10 mins at the end to go over all your answers to make sure you were not missing something.
     - Make sure you stick to your timetable and avoid the sunk cost fallacy. Thinking that I have already spent 5 mins on a question so let me spend a couple more mins to try and crack the question often leads to you spending 15 mins on the question and then you are terribly short on time.
   - I try to order the questions from easiest to hardest and I think I do fine on the average but the ordering might not match with yours. E.g. for some reason you might have studied a particular part of the book the night before the exam and that part might relevant to say the last question. So what I think might a hard question for an average student in the class might be easy for you. Reading through all questions upfront will also help you identify these "out of order" questions.
2. **Try to reinvent as little of the wheel as possible.**
   - Your first attack on any problem should be to see if you can sufficiently modify the question/input to the algorithm so that you can use a solution from a previous HW problem/the book/stuff on piazza as a *blackbox*. Note this is the same philosophy as to why you should libraries instead of writing code from scratch.

Story Behind the HW



note ☆ stop following 52 views

Story Behind the HW #1: Q3 on HW 3

Throughout the course there will be HW problems based on some really cool algorithmic idea (at least according to me!) that has some real life application and/or is something that I have used in my research. After the solutions for the corresponding HW have been handed out, I'll followup with a post on piazza giving more pointers for the connection. This is the first one in the series and is related to Q3 on HW 3.

First, from the description of DFT on the support page, it follows that a (generalization of) the solution for Q3 on HW3 implies an $O(n \log n)$ time algorithm to compute the DFT. In other words, the algorithm gives a Fast Fourier Transform (or FFT) that consistently finds itself among the top 10 algorithms from the 20th century in various compilations (e.g. this one).

Generally the FFT is stated as a "Divide and Conquer" algorithm (see e.g. Sec 5.6 in the book). However, I personally think the distributive law based algorithm (which is what Q3 as asking for) is more natural and also cleaner (e.g. you do not have worry about complex numbers and roots of unity).

(*Side Note:* As far I understand, the divide and conquer based algorithm heavily uses the structure of the DFT matrix and does not work for the class of matrices in Q3. If I'm mistaken, please let me know!)

For me the coolest thing about the distributive law strategy is not that it recovers FFT (even though it is definitely cool), but that distributive law strategy can recover tons of other algorithms (in areas such as error-correcting codes and machine learning). See this survey by Aji and McEliece for an overview of this. (This is paper that got me started on this front.)

Couple of years back, I and my collaborators have worked on extending the Aji-McEliece framework to work for even more general framework (while also improving upon existing results). Here are two papers: the first one is long while the second one uses the database language (but is relatively shorter). Also here is some idiot trying to explain these ideas (including talking about the FFT):

Questions?

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
        $L_{i+1}$ = 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++
```

Linked List

Version in KT also computes a BFS tree

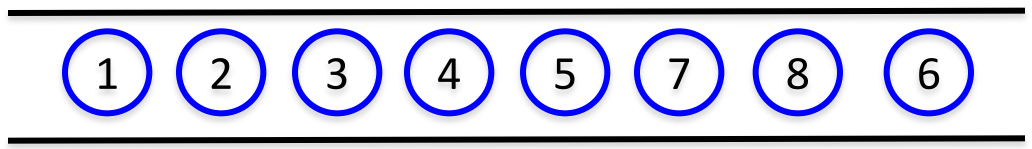
All the layers as one

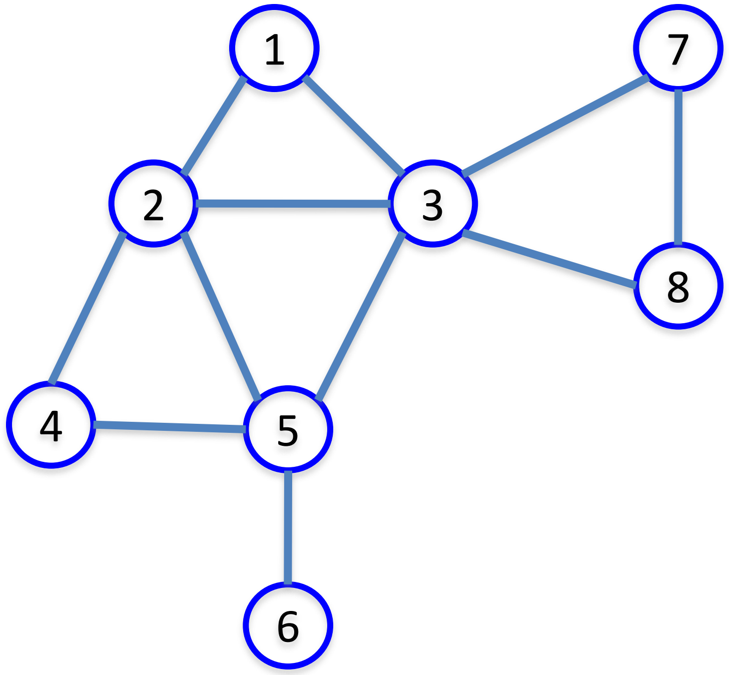
```
BFS(s)
    CC[s] = T and CC[w] = F for every w≠ s
    Set i = 0
    Set $L_0$= {s}
    While $L_i$ is not empty
        $L_{i+1} = \emptyset$
        For every u in $L_i$
            For every edge (u,w)
            If CC[w] = F then
                CC[w] = T
                Add w to $L_{i+1}$
        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
```

$O(n)$

$O(1)$

Repea

onc

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

$O(1)$

$O(n_u)$

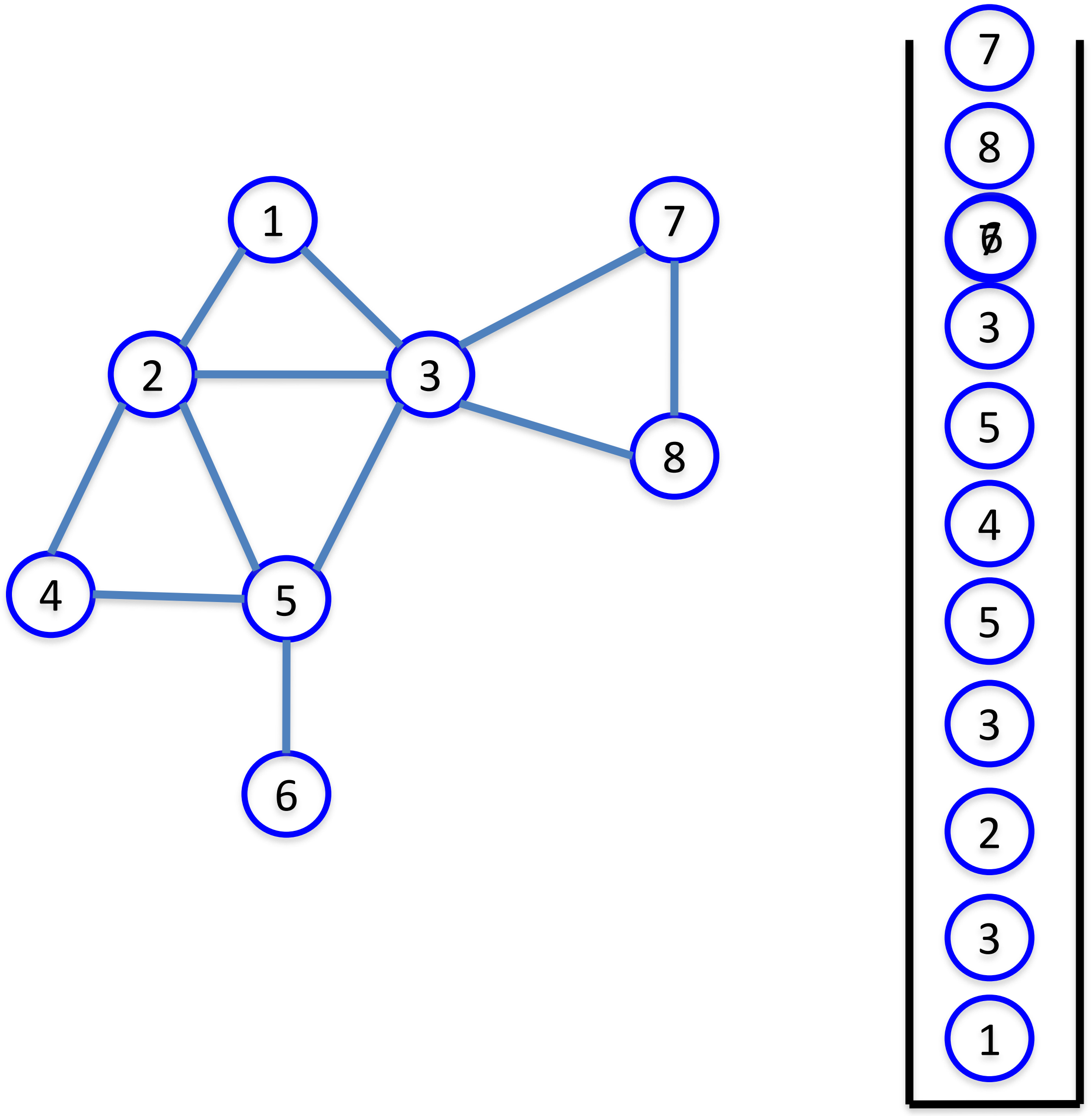
$O(1)$

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)

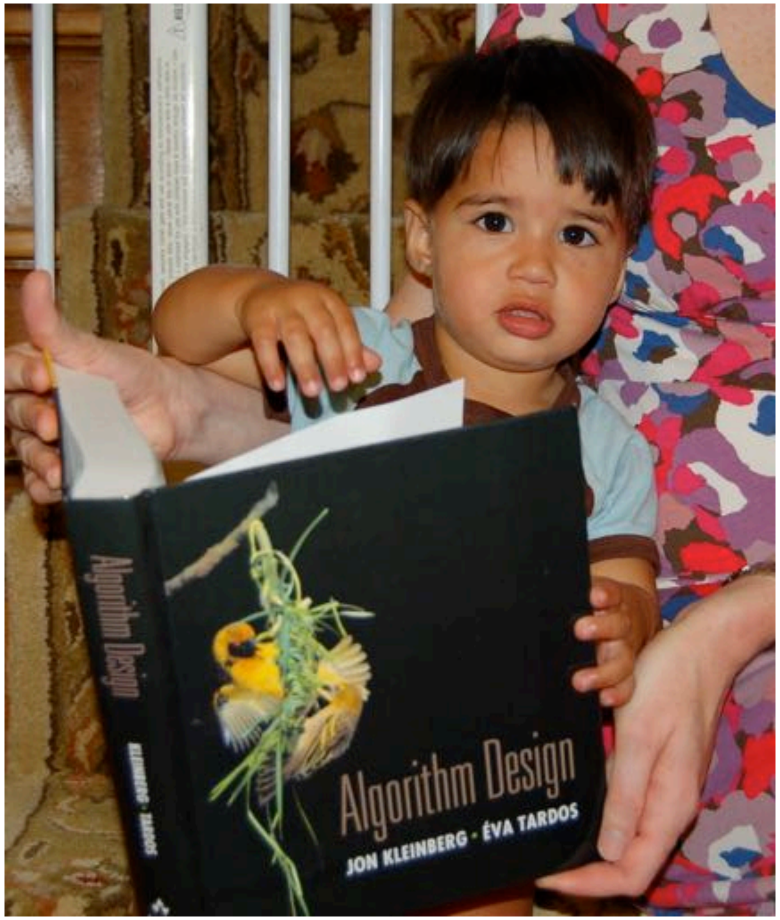
```
CC[s] = T and CC[w] = F for every w≠ s

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

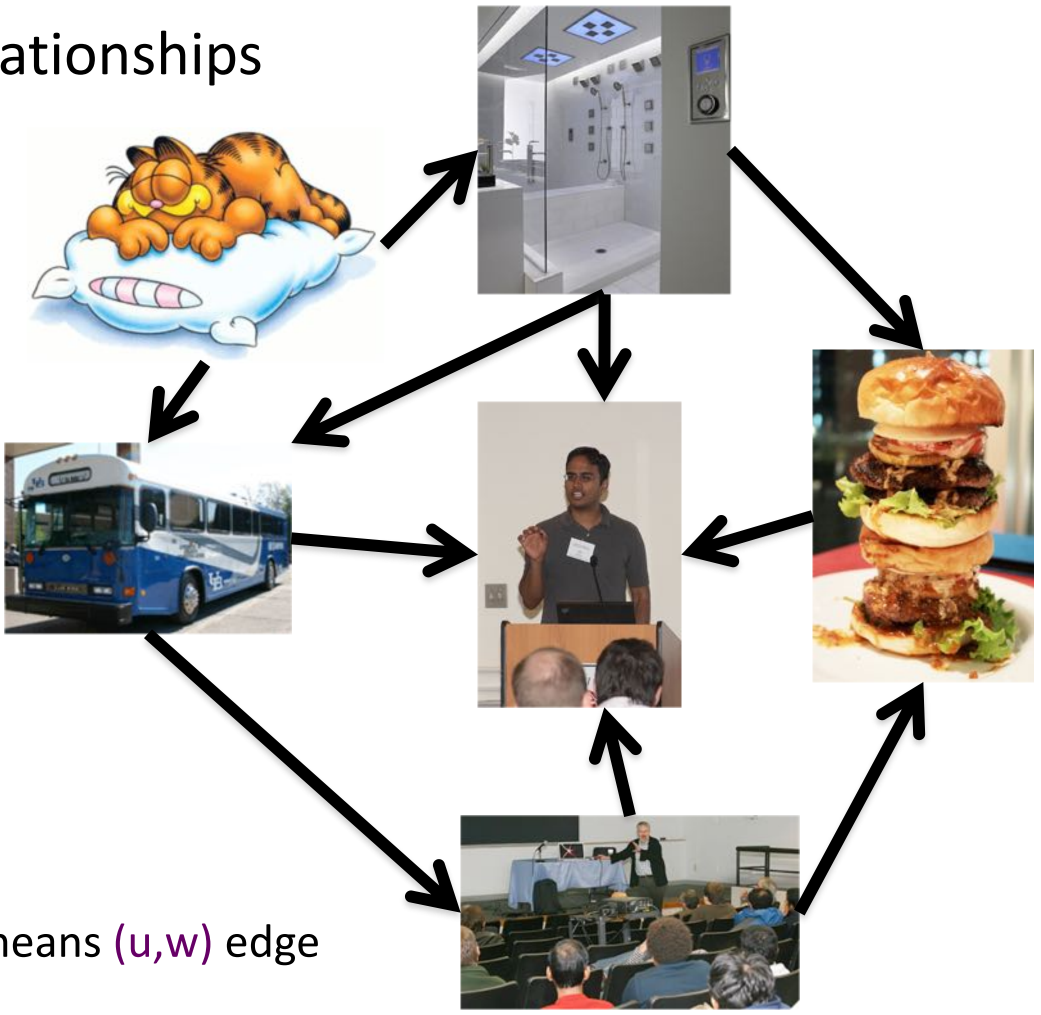
Sec 3.3, 3.4 and 3.5 of [KT]



Directed graphs

Model asymmetric relationships

Precedence relationships

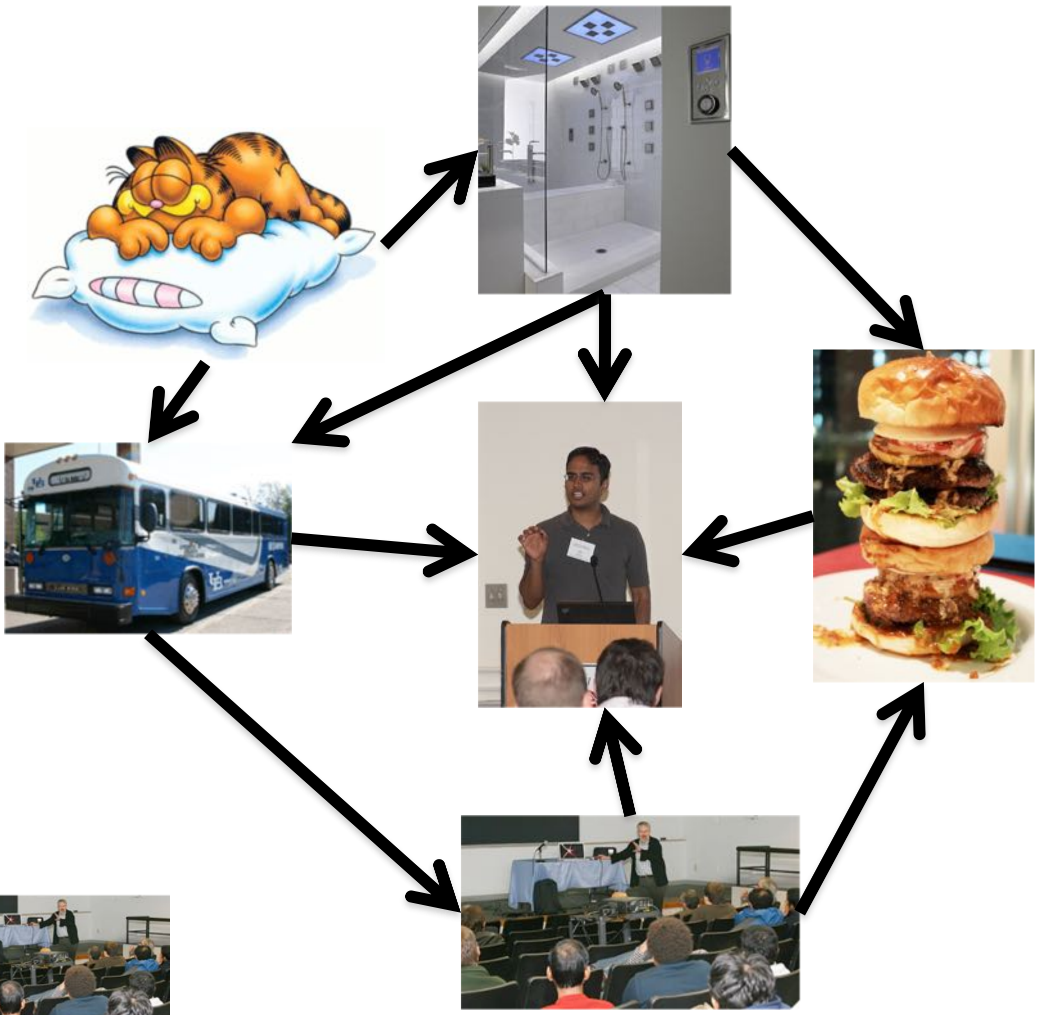


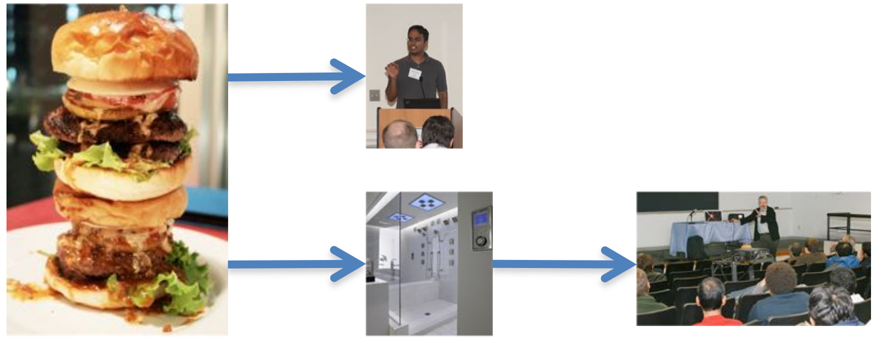
u needs to be done before w means (u,w) edge

Directed graphs

Adjacency matrix is not symmetric

Each vertex has two lists in Adj. list rep.

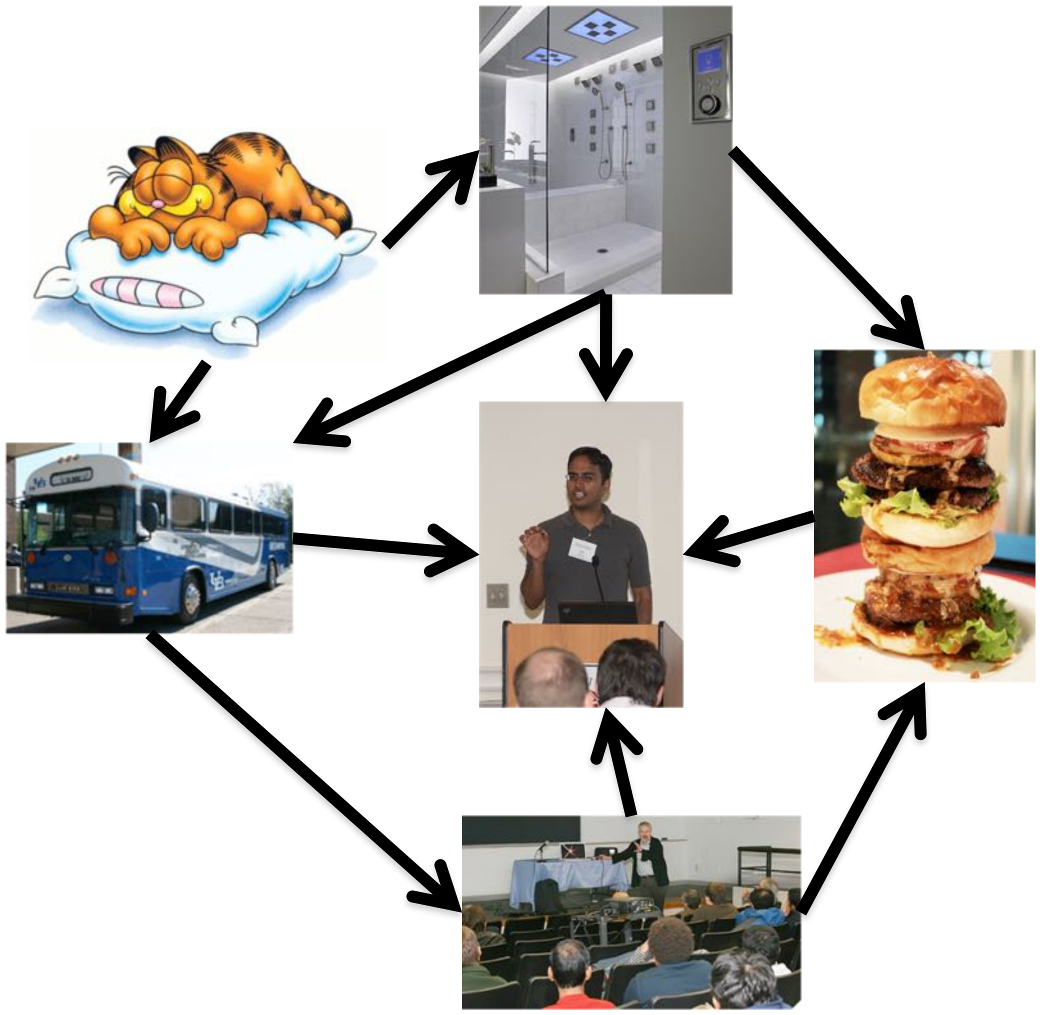




Directed Acyclic Graph (DAG)

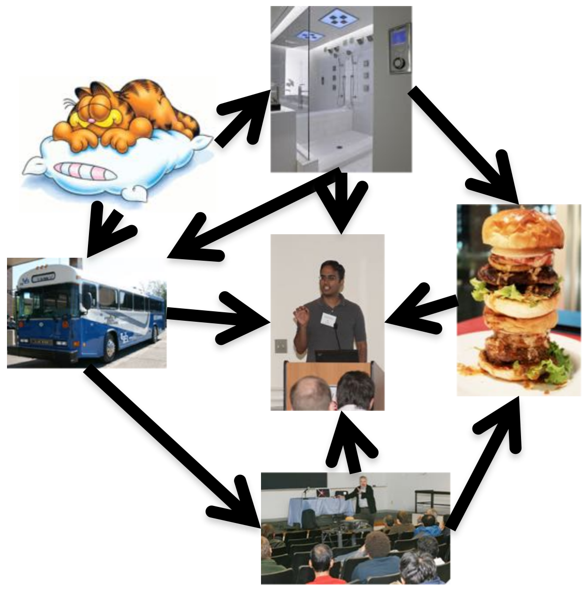
No directed cycles

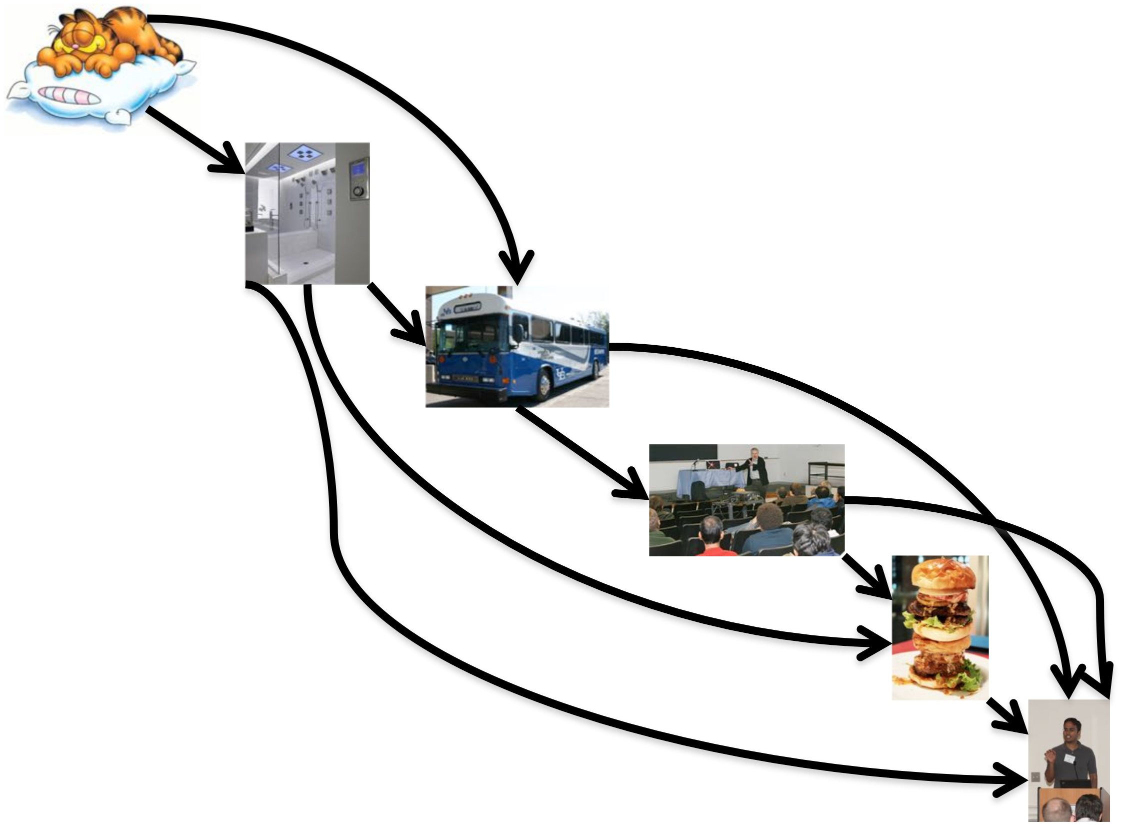
Precedence relationships are consistent



Topological Sorting of a DAG

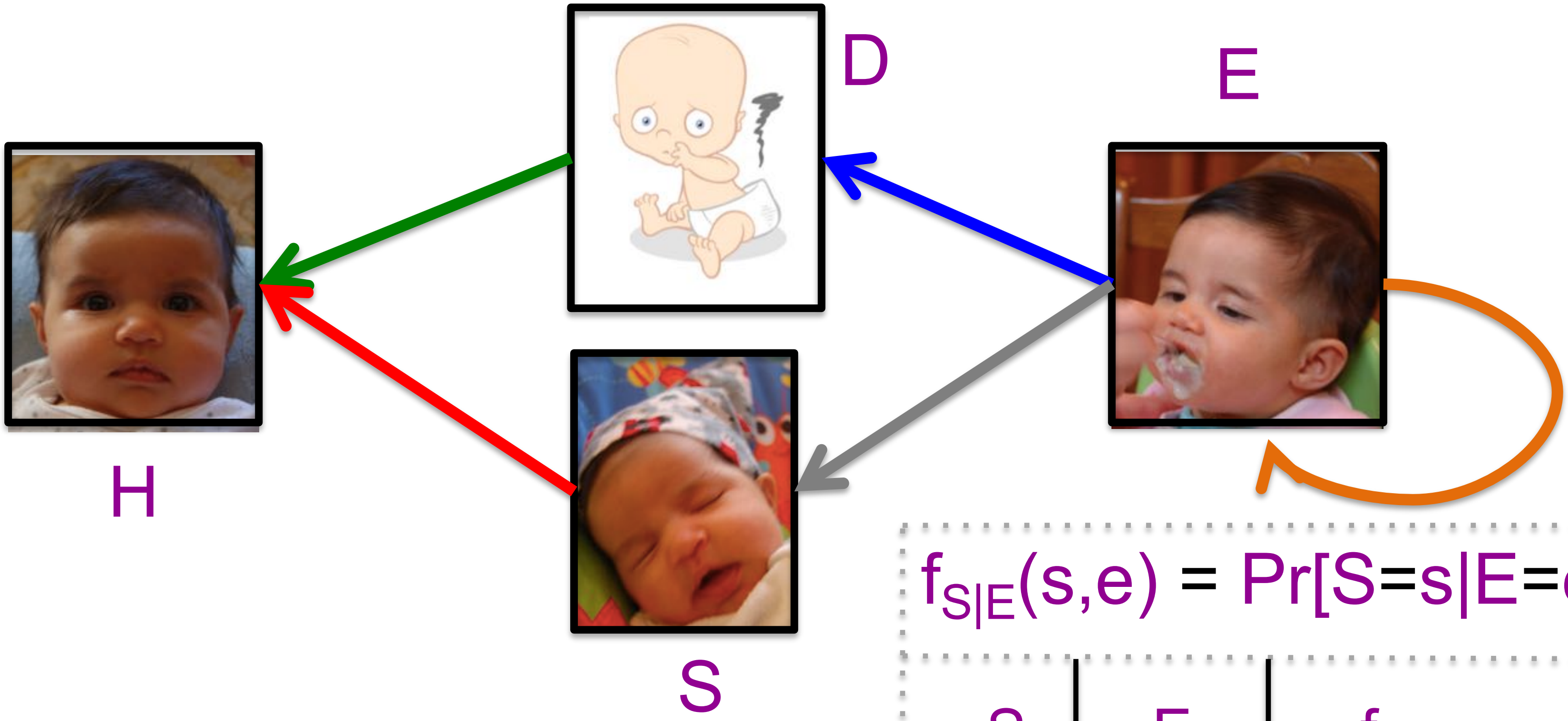
Order the vertices so that all edges go “forward”





Probabilistic Graphical Models (PGMs)

http://ginaskokopelli.com/wp-content/uploads/2013/01/DiaperDealsLogo.jpg



$$f_{S|E}(s,e) = \Pr[S{=}s|E{=}e]$$

S	E	$f_{S|E}$
1	1	0.8
1	0	0.3
0	1	0.2
0	0	0.7

$$\varphi\ (h) = \sum_{d,s,e} f_{H|D,S}(h,d,s) \times f_{S|E}(s,e) \times f_{D|E}(d,e) \times f_E(e)$$