

Lecture 14

CSE 331

Please have a face mask on

Masking requirement



UB requires all students, employees and visitors – regardless of their vaccination status – to wear face coverings while inside campus buildings.

<https://www.buffalo.edu/coronavirus/health-and-safety/health-safety-guidelines.html>

Quiz 1 on March 11



note @261



[stop following](#)

59 views

Actions ▼

Quiz 1 on Friday, March 11

The first quiz will be from **3:00-3:15pm in class** (Cooke 121) on **Friday, March 11**. We will have a 5 mins break after the quiz and the lecture will start at 3:20pm.

We will hand out the quiz paper at 2:55pm but you will **NOT** be allowed to open the quiz to see the actual questions till 3:00pm. 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: [@244](#).) Also quiz 1 will cover all topics that we cover in class till Wednesday, March 2.

Also, like the mid-term you can bring in one letter sized cheat-sheet (you can use both sides). But other than a cheatsheet and writing implements, nothing else is allowed.

$O(m+n)$ BFS Implementation

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}

$i++$

return cc

$O(m+n)$ BFS Implementation

BFS(s)

Array

Input graph as
Adjacency list

$CC[s] = T$ 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}

$i++$

return cc

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 $w \neq s$

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}

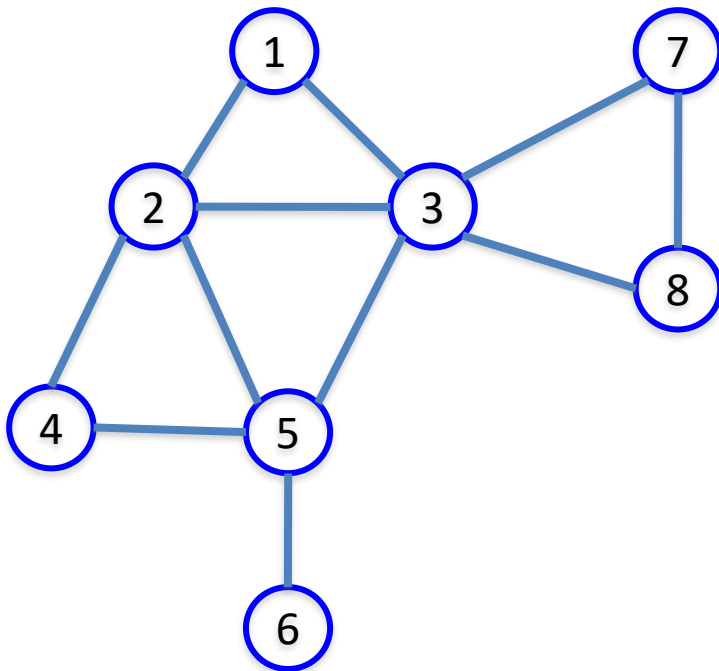
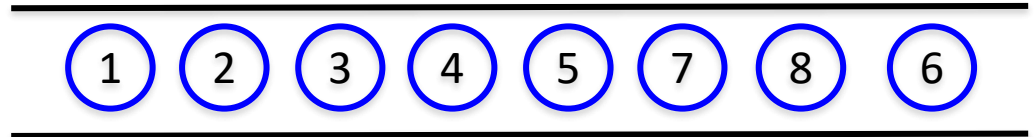
$i++$

return cc

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 $w \neq s$

$O(n)$

Initialize $Q = \{s\}$

$O(1)$

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

Repeat
once

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

$O(1)$

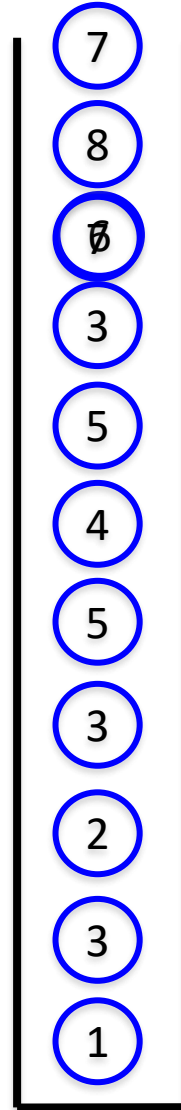
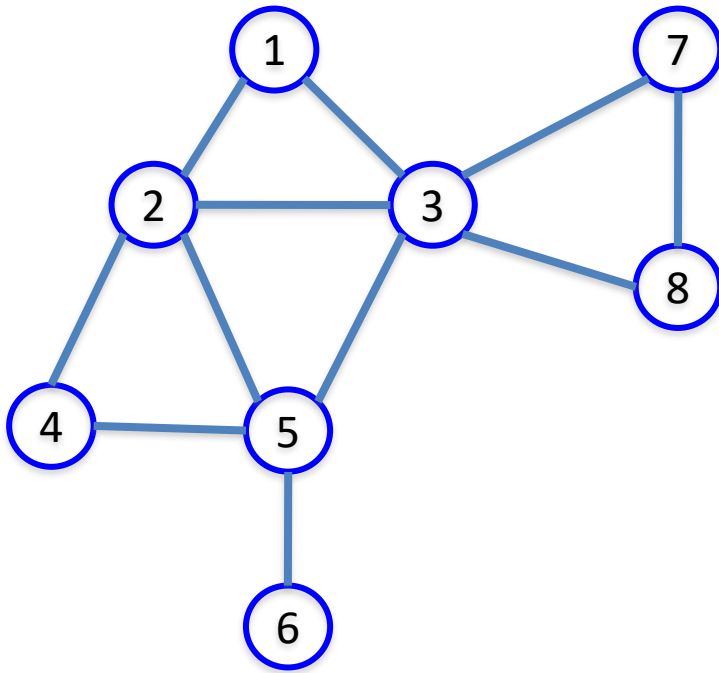
$O(n_u)$

$O(1)$

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 $w \neq s$

Initialize $\hat{S} = \{s\}$

While \hat{S} is not empty


Pop the top element u in \hat{S}

For every edge (u, w)

If $CC[w] = F$ then

$CC[w] = T$

Push w to the top of \hat{S}



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

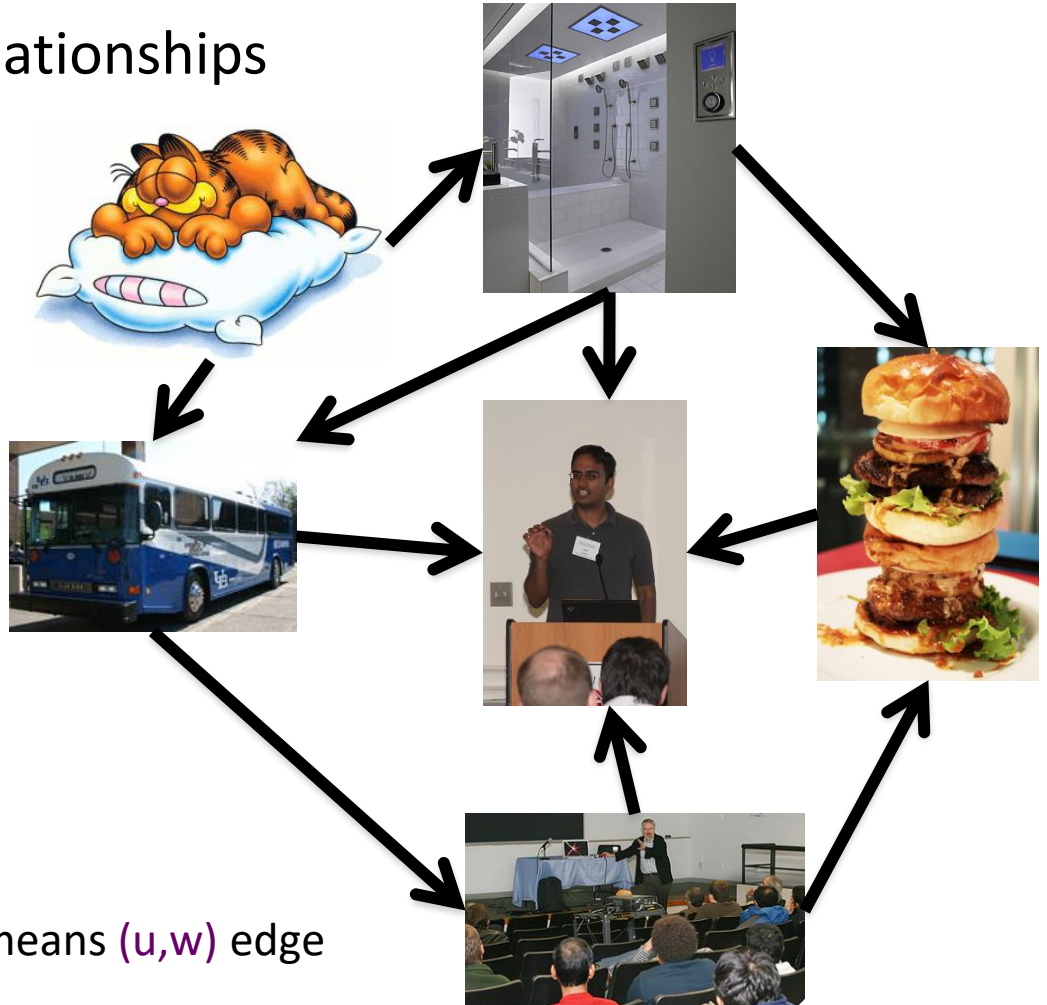
Reading Assignment

Sec 3.3, 3.4, 3.5 and 3.6 of [KT]

Directed graphs

Model asymmetric relationships

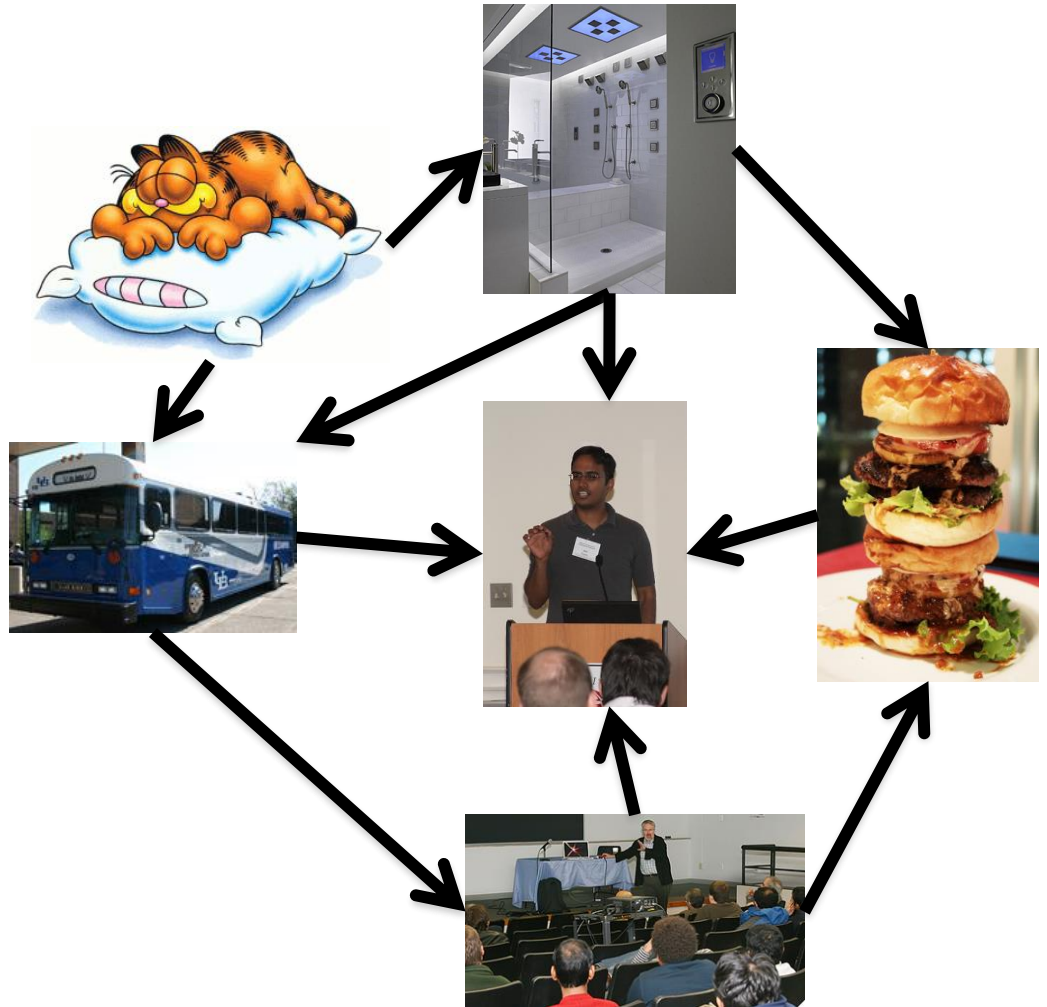
Precedence relationships



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

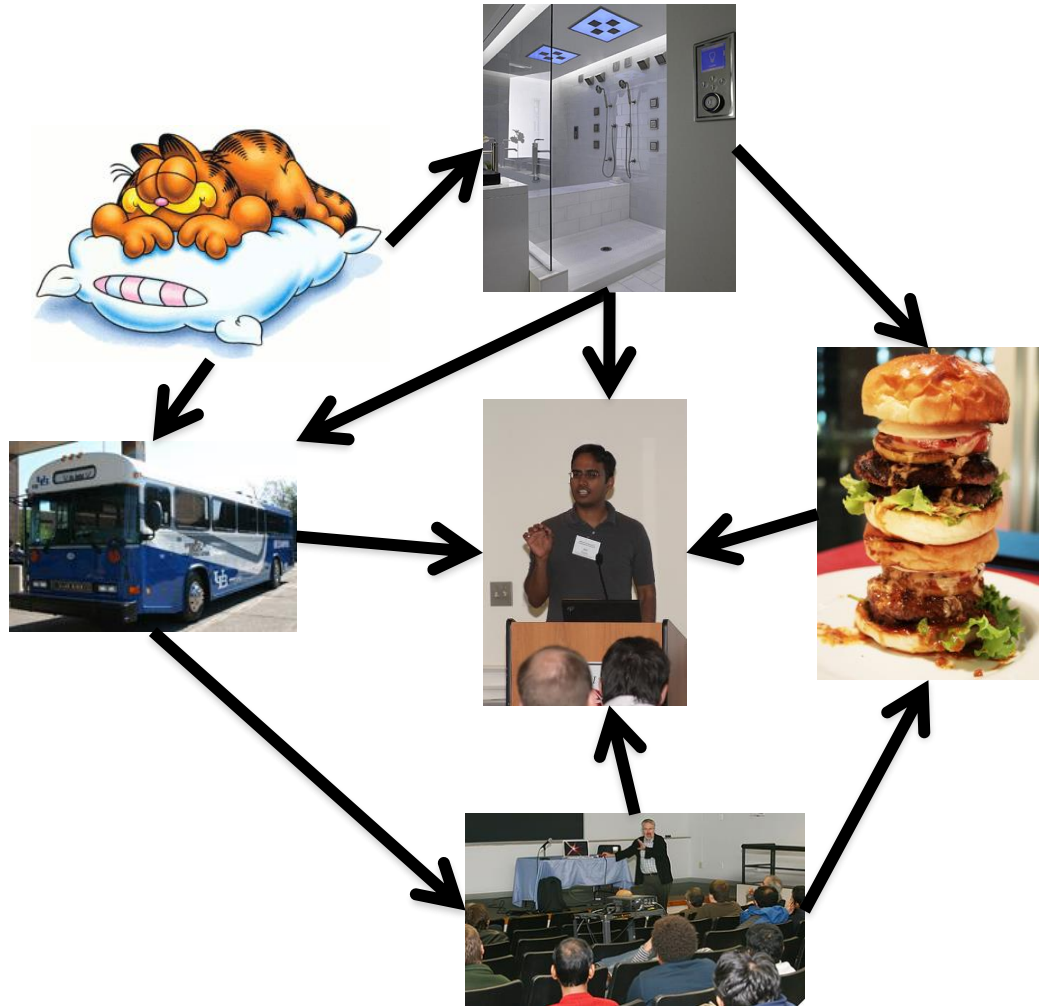
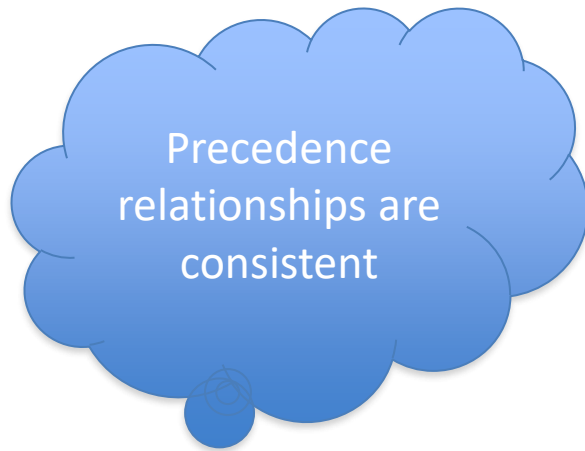
Directed graphs

Adjacency
matrix is not
symmetric



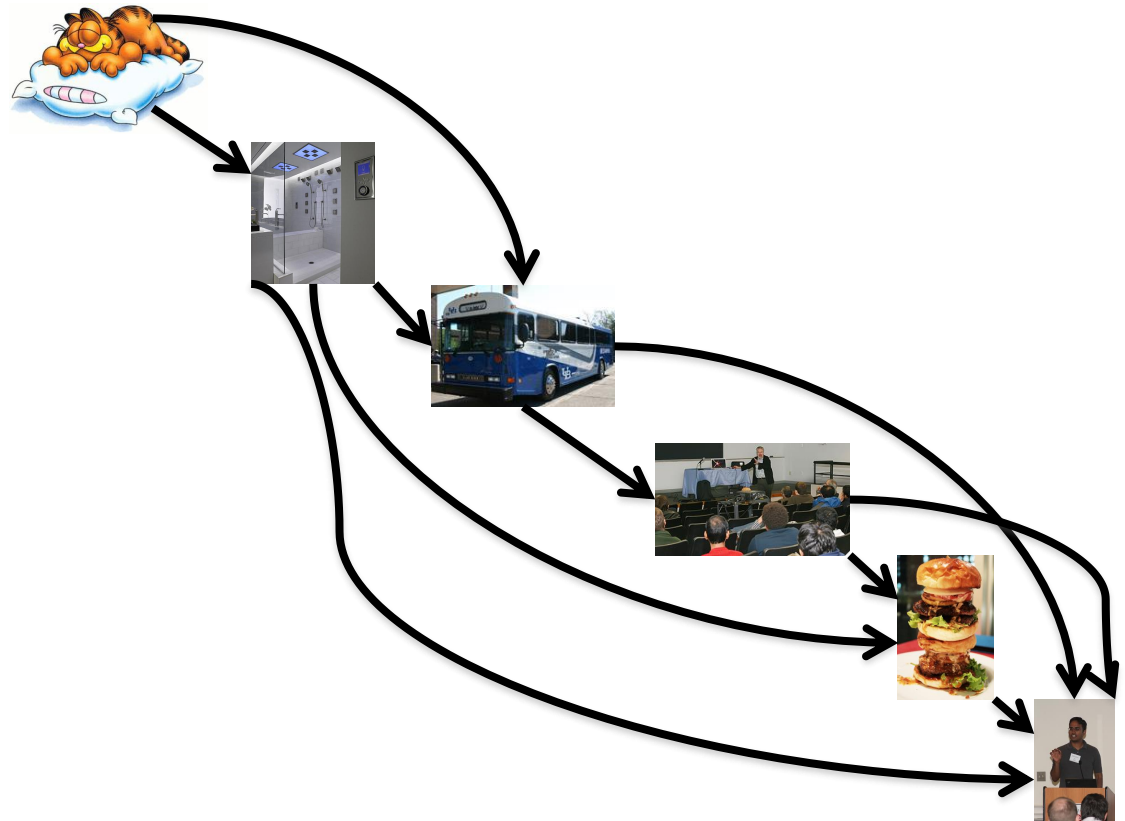
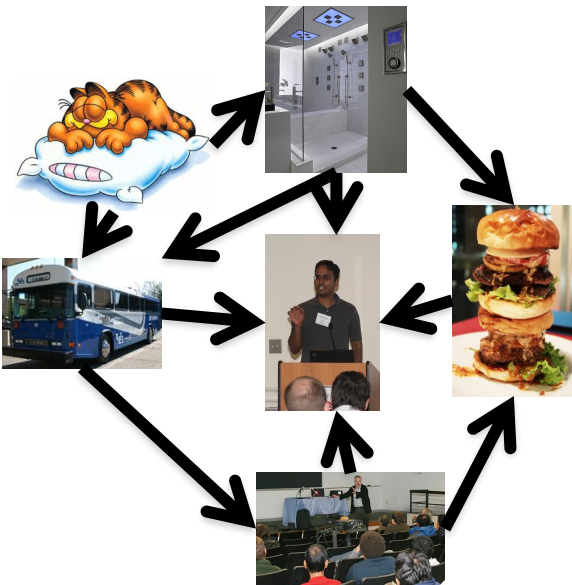
Directed Acyclic Graph (DAG)

No directed cycles



Topological Sorting of a DAG

Order the vertices so that all edges go “forward”



More details on Topological sort

Topological Ordering

This page collects material from previous incarnations of CSE 331 on topological ordering.

Where does the textbook talk about this?

Section 3.6 in the textbook has the lowdown on topological ordering.

Fall 2018 material

First lecture

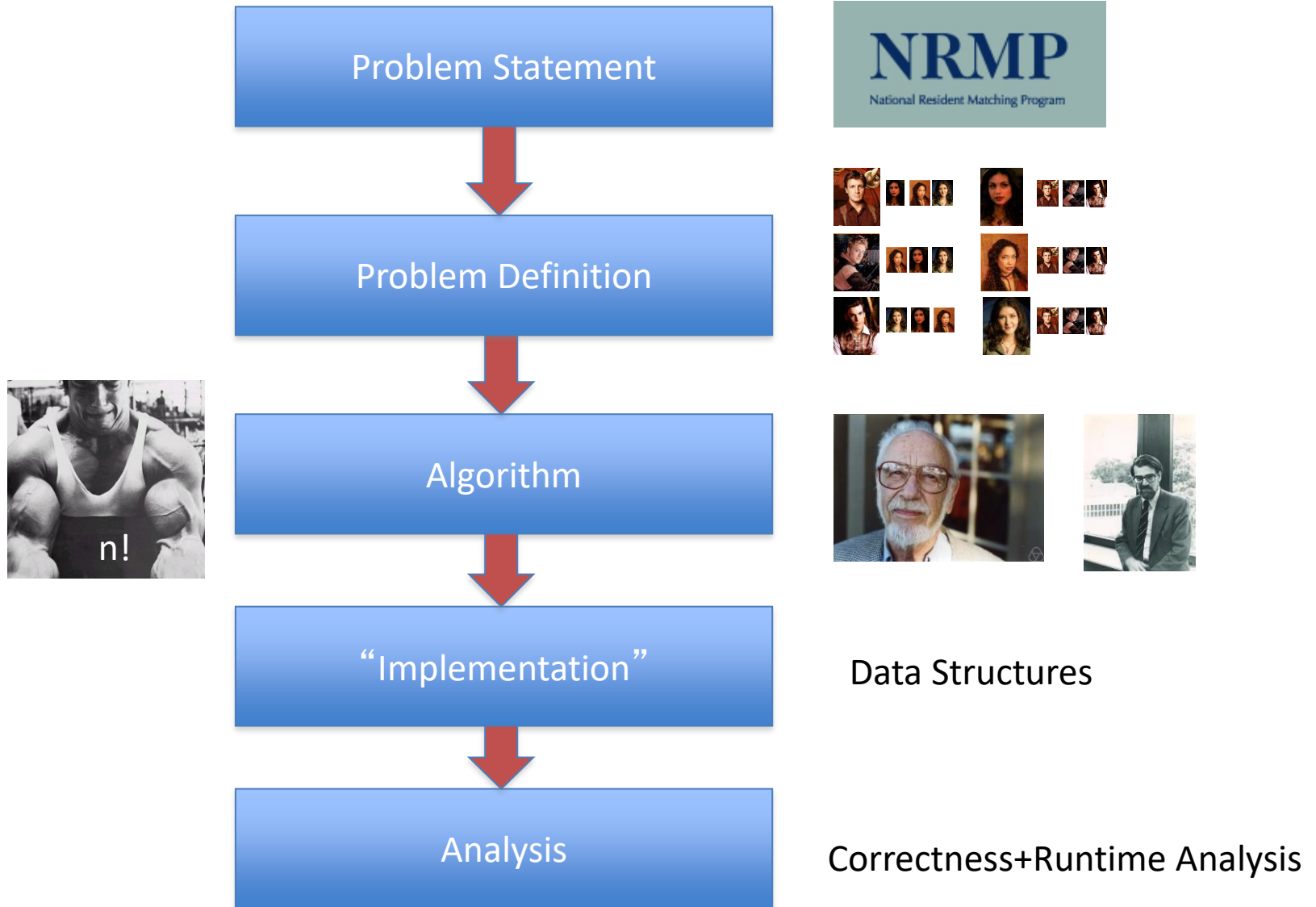
Here is the lecture video:

CSE331 on 10/1/2018 (Mon)

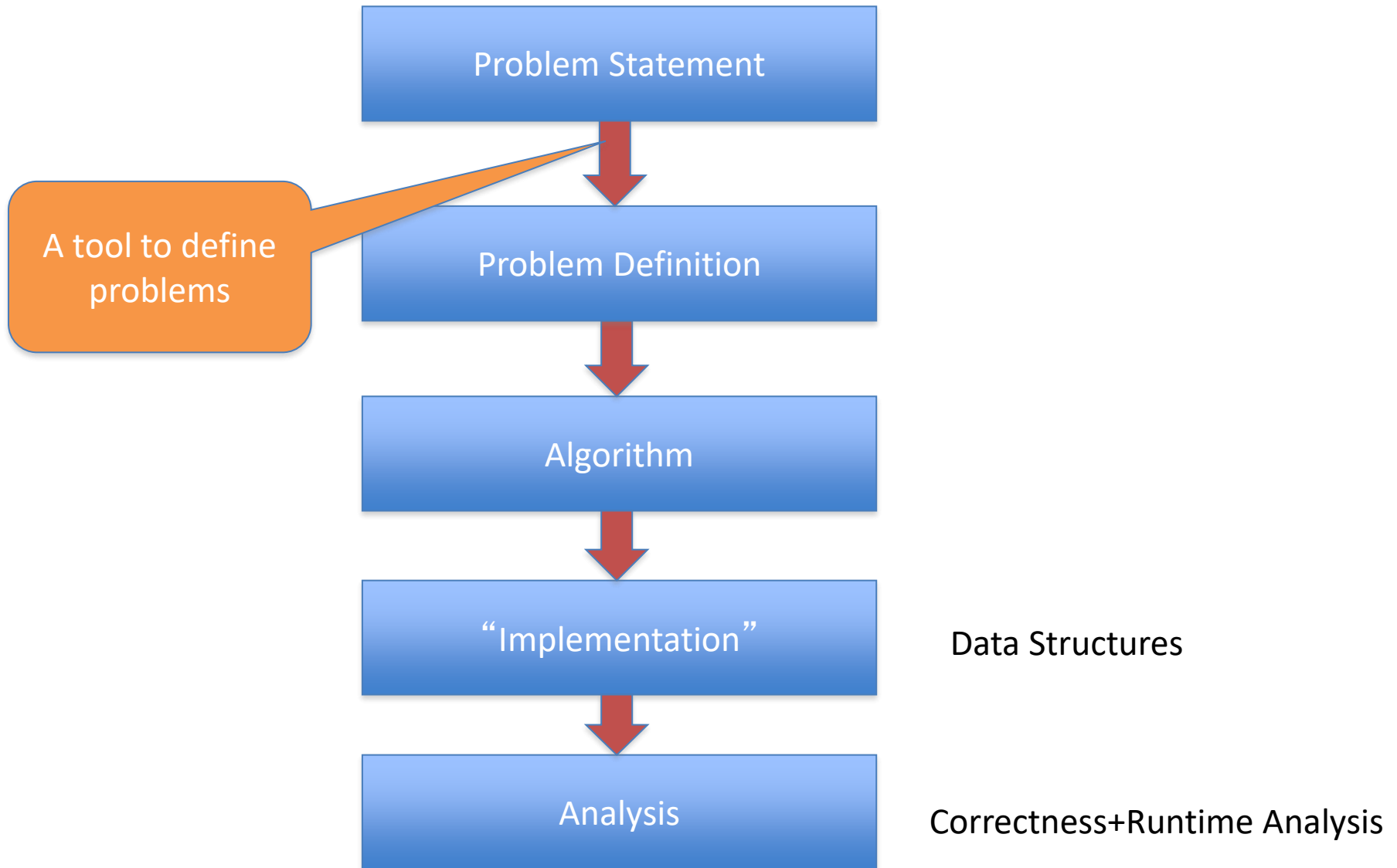


Mid-term material until here

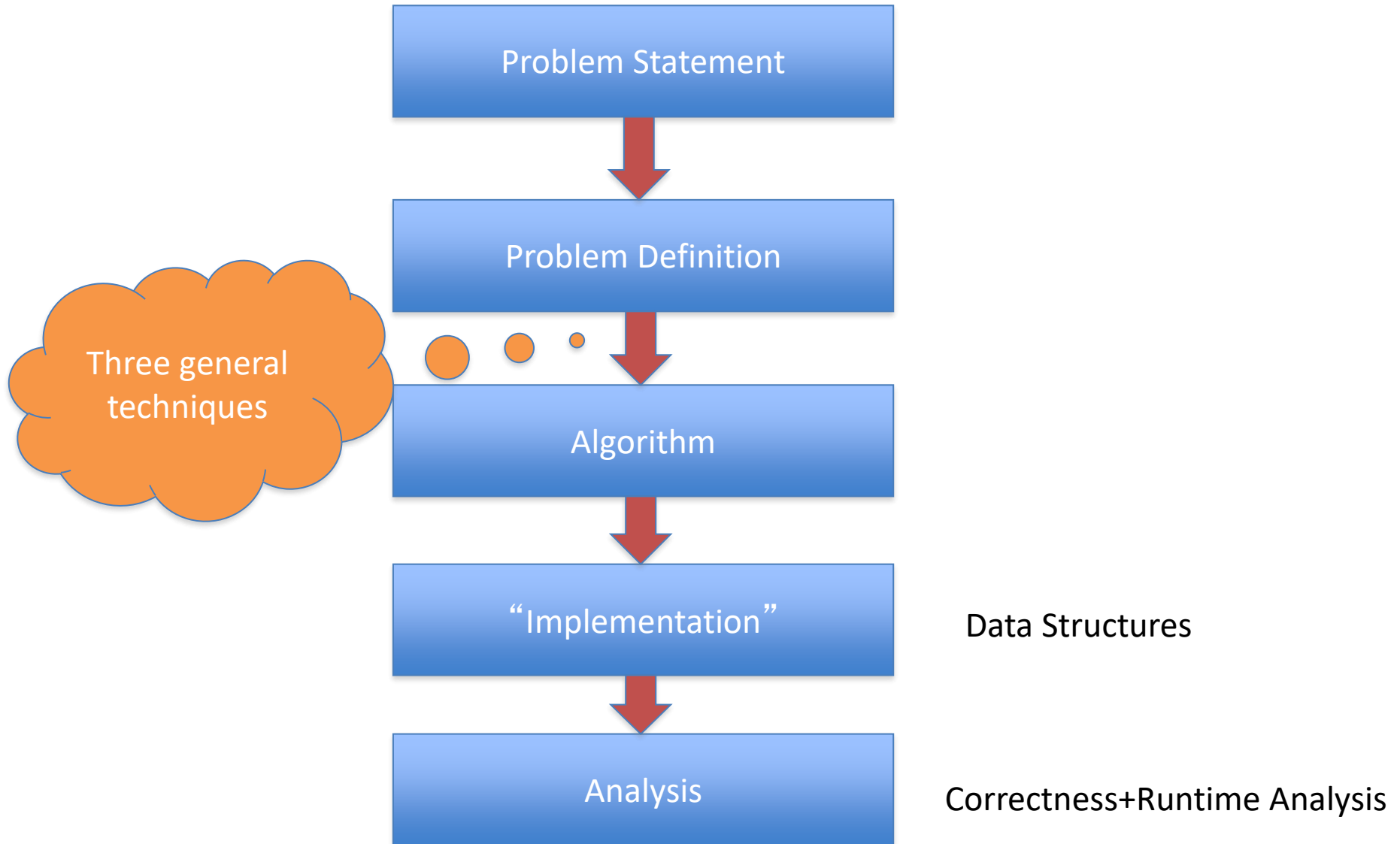
Main Steps in Algorithm Design



Where do graphs fit in?



Rest of the course*



Greedy algorithms

Build the final solution piece by piece

Being short sighted on each piece

Never undo a decision

Know when you see it



End of Semester blues

Can only do one thing at any day: what is the maximum number of tasks that you can do?



Write up a term paper

Party!

Exam study

Homework

331 HW

Project

Saturday

Sunday

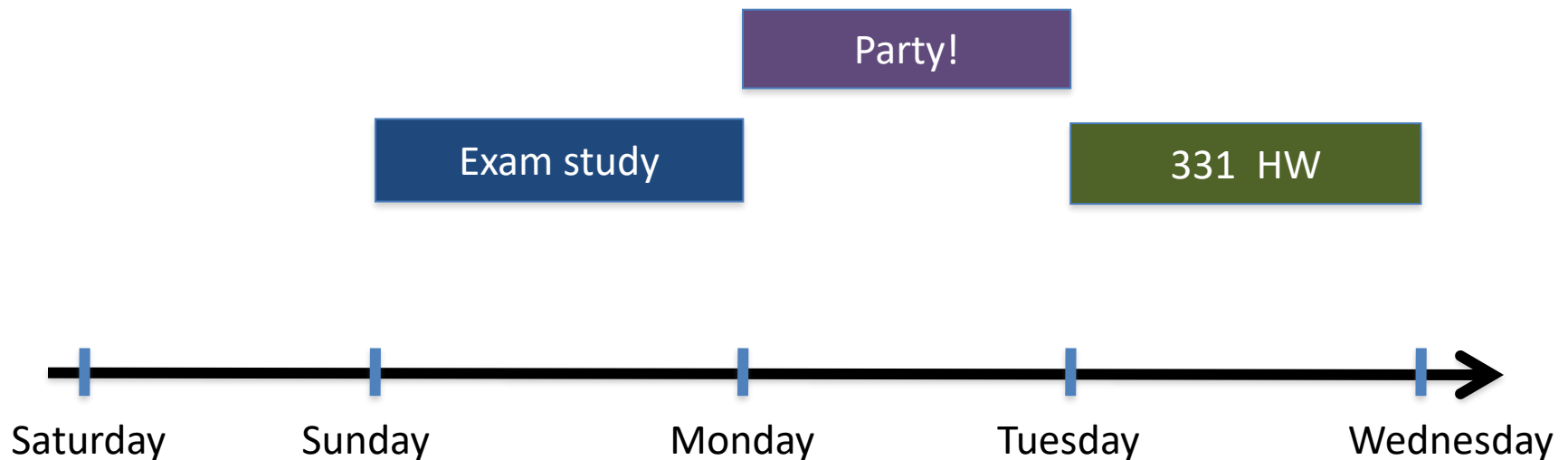
Monday

Tuesday

Wednesday

The optimal solution

Can only do one thing at any day: what is the maximum number of tasks that you can do?



Interval Scheduling Problem

Input: n intervals $[s(i), f(i))$ for $1 \leq i \leq n$



$\{ s(i), \dots, f(i)-1 \}$

Output: A *schedule* S of the n intervals

No two intervals in S conflict

$|S|$ is maximized