Lecture 15

CSE 331

Project groups due Friday!

Deadline: Tonight (Friday, March 3), 11:59pm

CSE 331 Syllabus Piazza Schedule Homeworks ▾ Autolab Project ▾ Support Pages ▾

- Project Overview
- Group signup form

Spring 2023

Details and motivations for the project.

Motivation

CSE 331 is primarily concerned with the technical aspects of algorithms: how to design them and then how to analyze their correctness and runtime. However, algorithms are pervasive in our world and are common place in many aspects of society. The main aim of the project is to have you explore in some depth some of the social implications of algorithms.

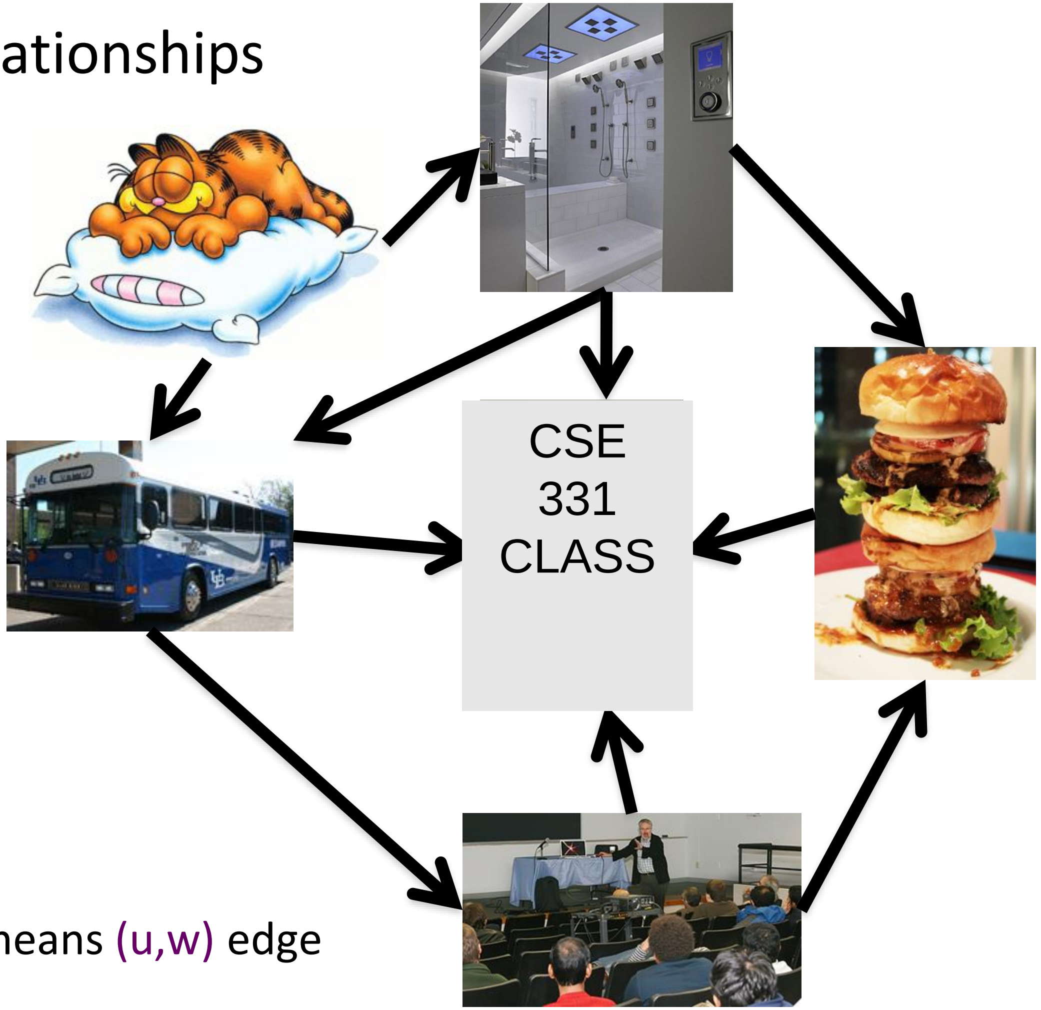
Just to give some examples for such implications:

- Big data is hot these days and there is a (not uncommon) belief that by running (mainly machine learning) algorithms on big data, we can detect patterns and use those to potentially make policy decisions. Here is a cautionary talk:

Directed graphs

Model asymmetric relationships

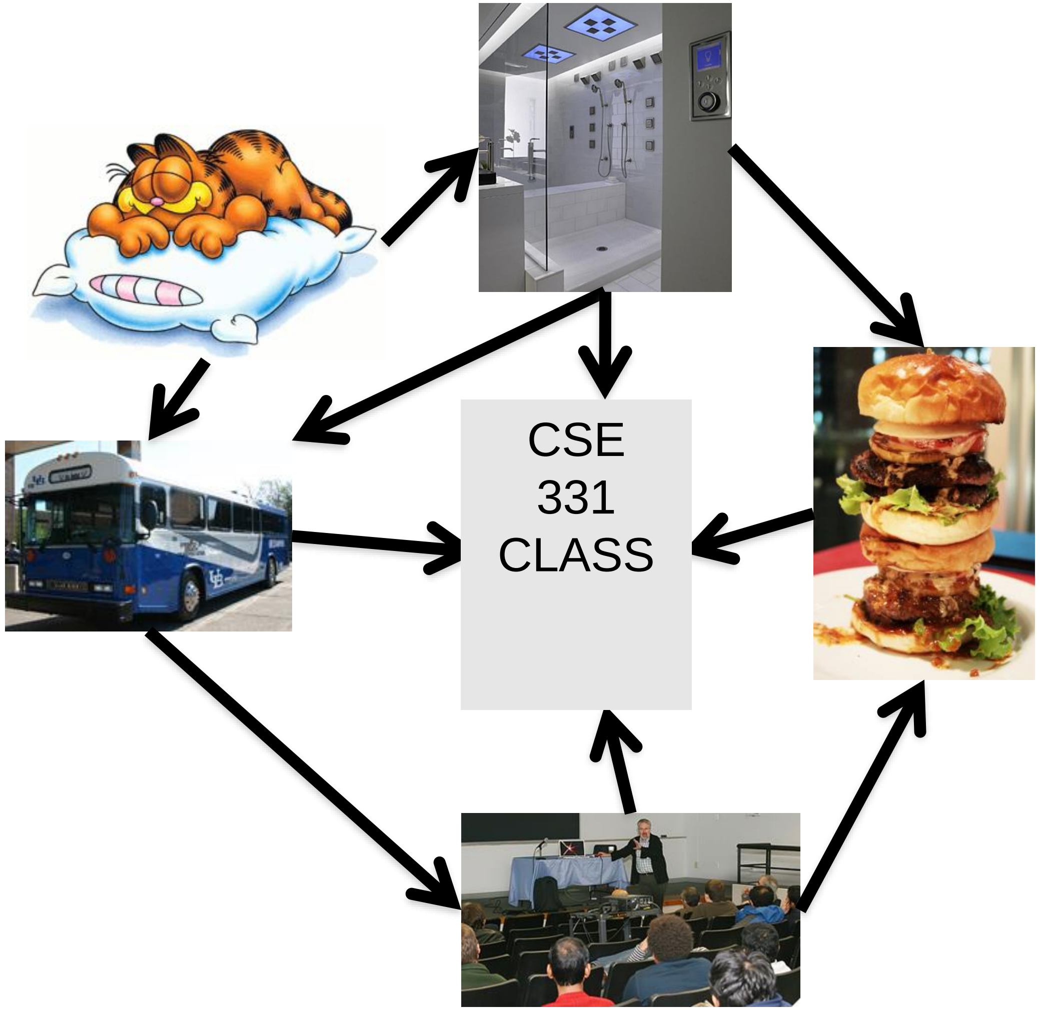
Precedence relationships



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

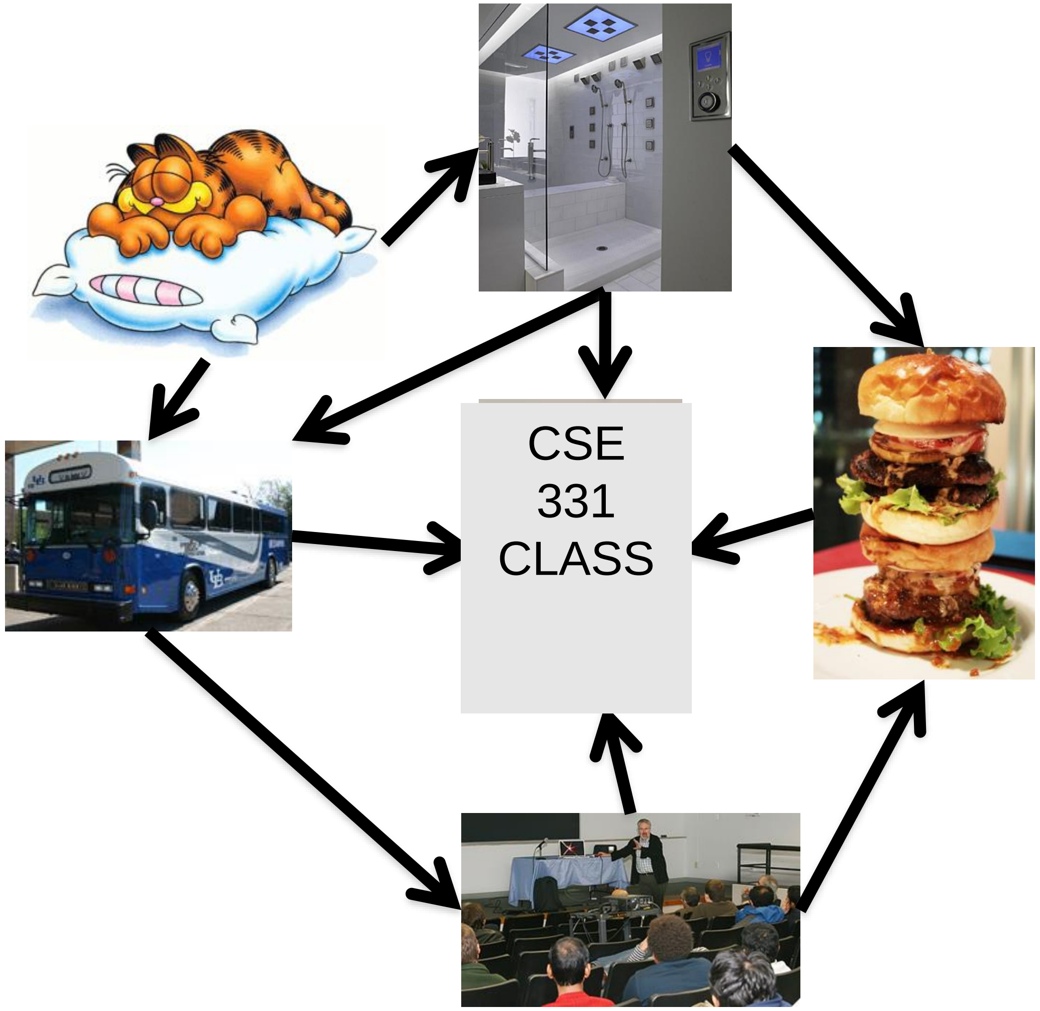
Directed graphs

Adjacency matrix is not symmetric



Directed Acyclic Graph (DAG)

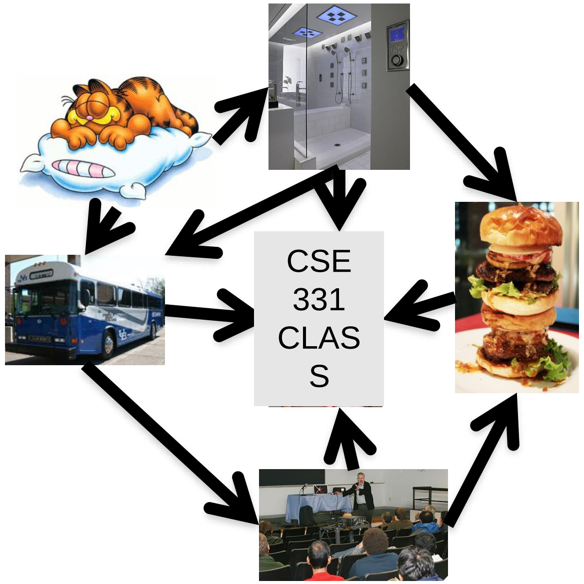
No directed cycles

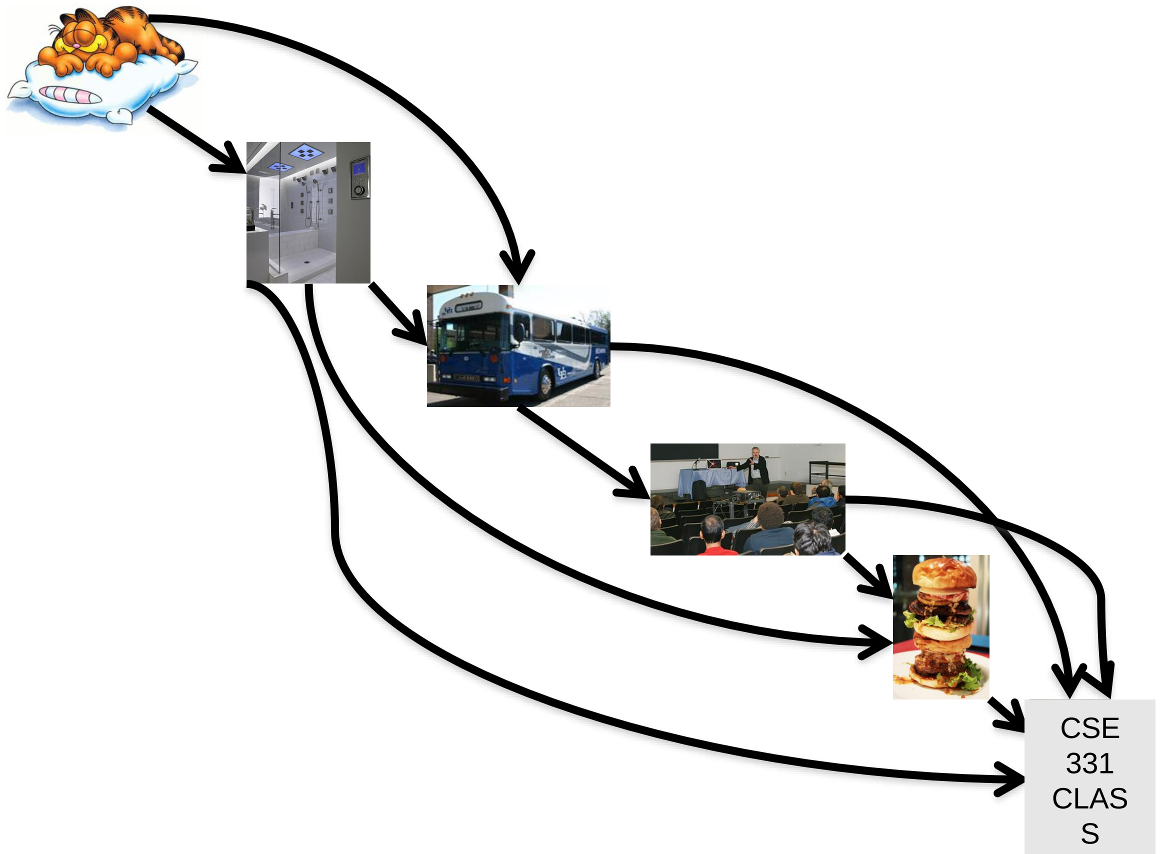


Precedence relationships are consistent

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.

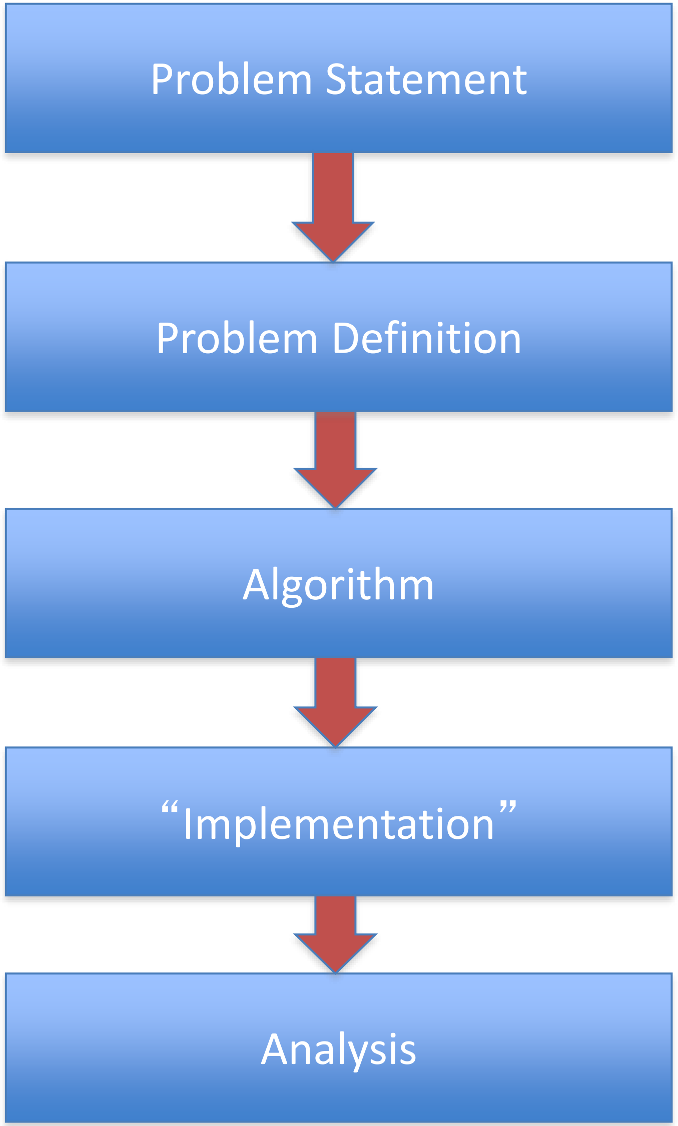
Fall 2018 material

First lecture

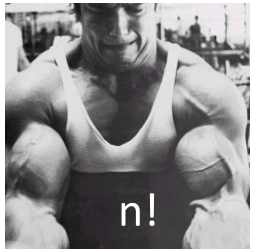
Here is the lecture video:

Mid-term material until here

Main Steps in Algorithm Design



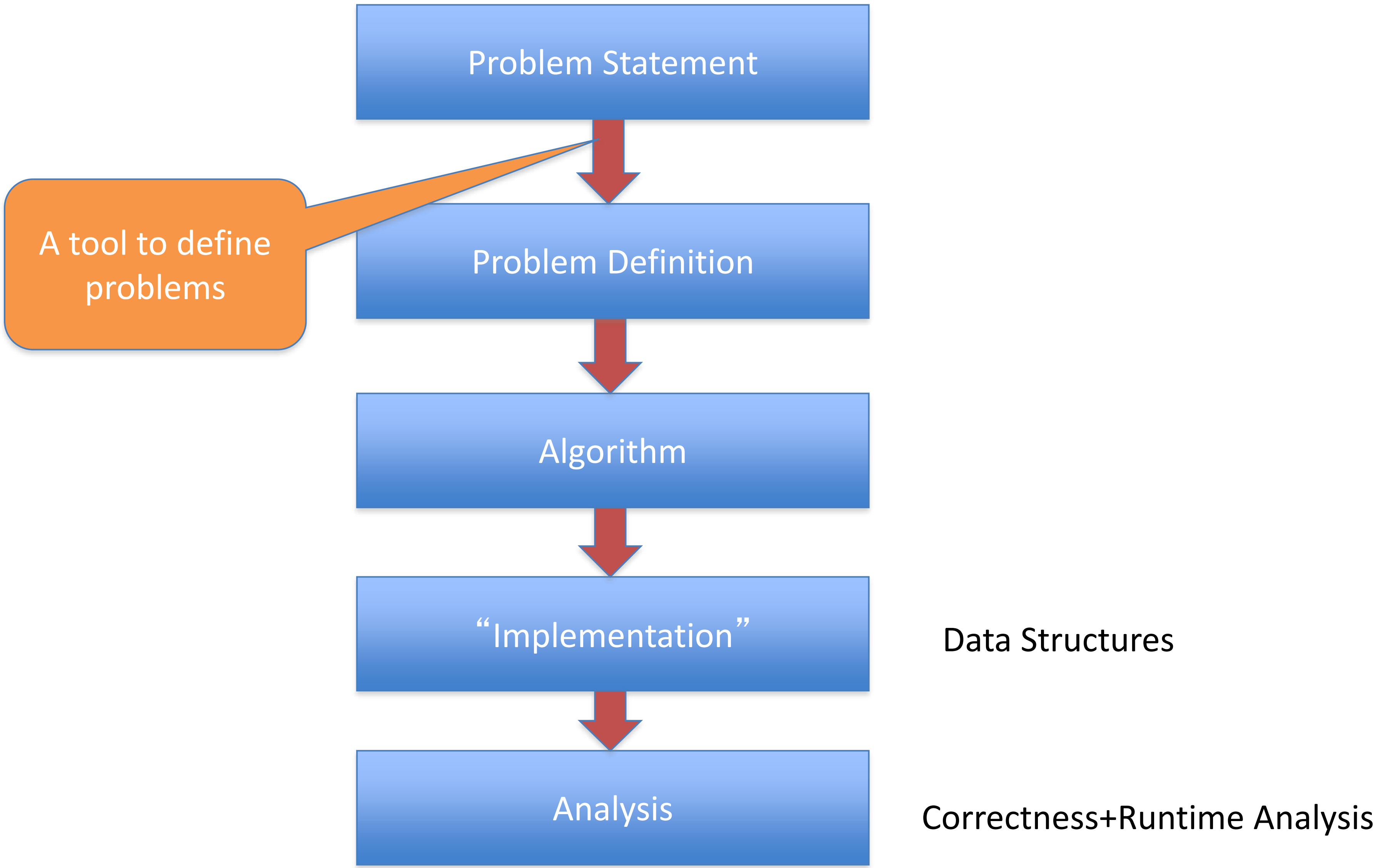
NRMP
National Resident Matching Program



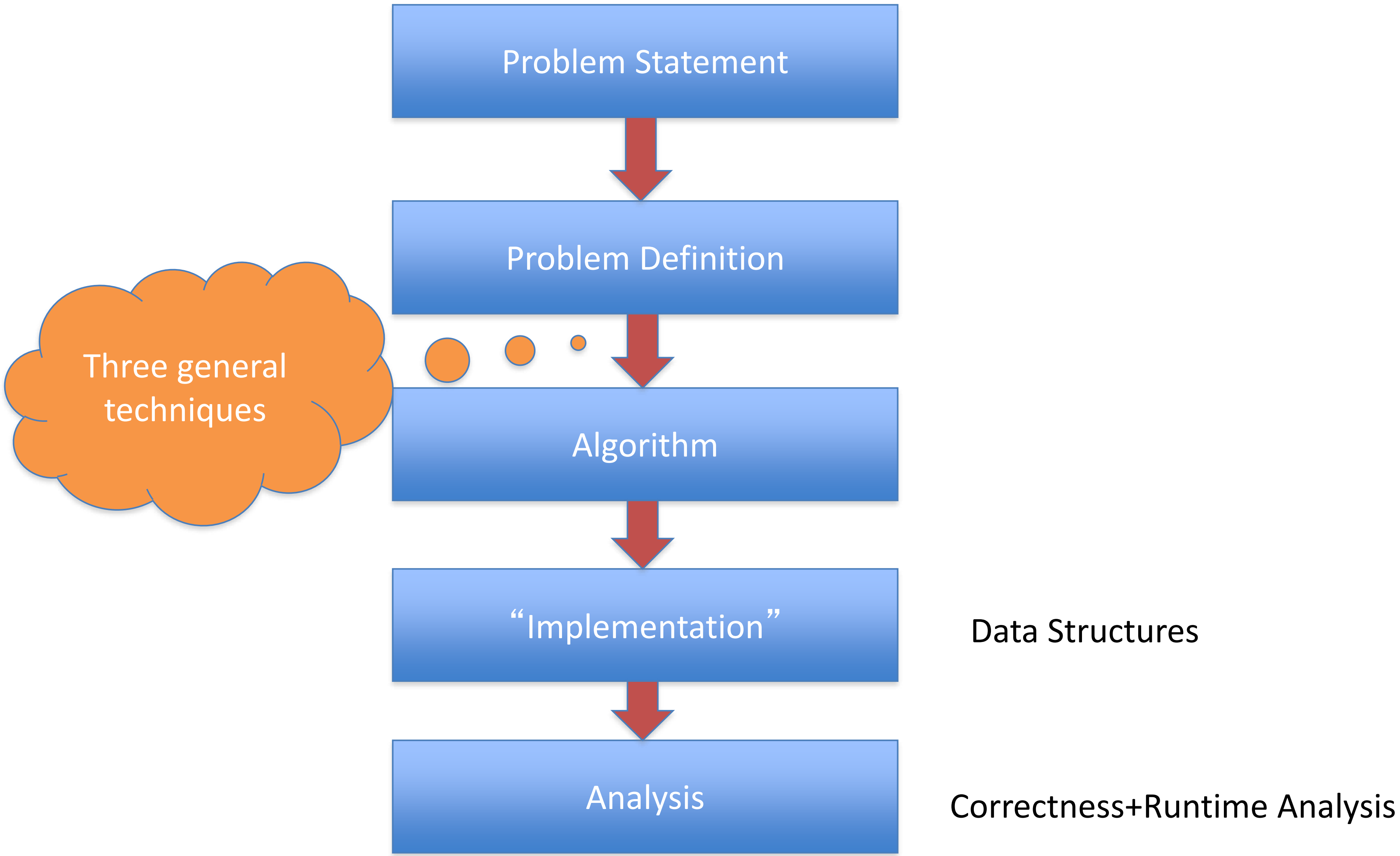
Data Structures

Correctness+Runtime Analysis

Where do graphs fit in?



Rest of the course*



Greedy algorithms

Build the final solution piece by piece

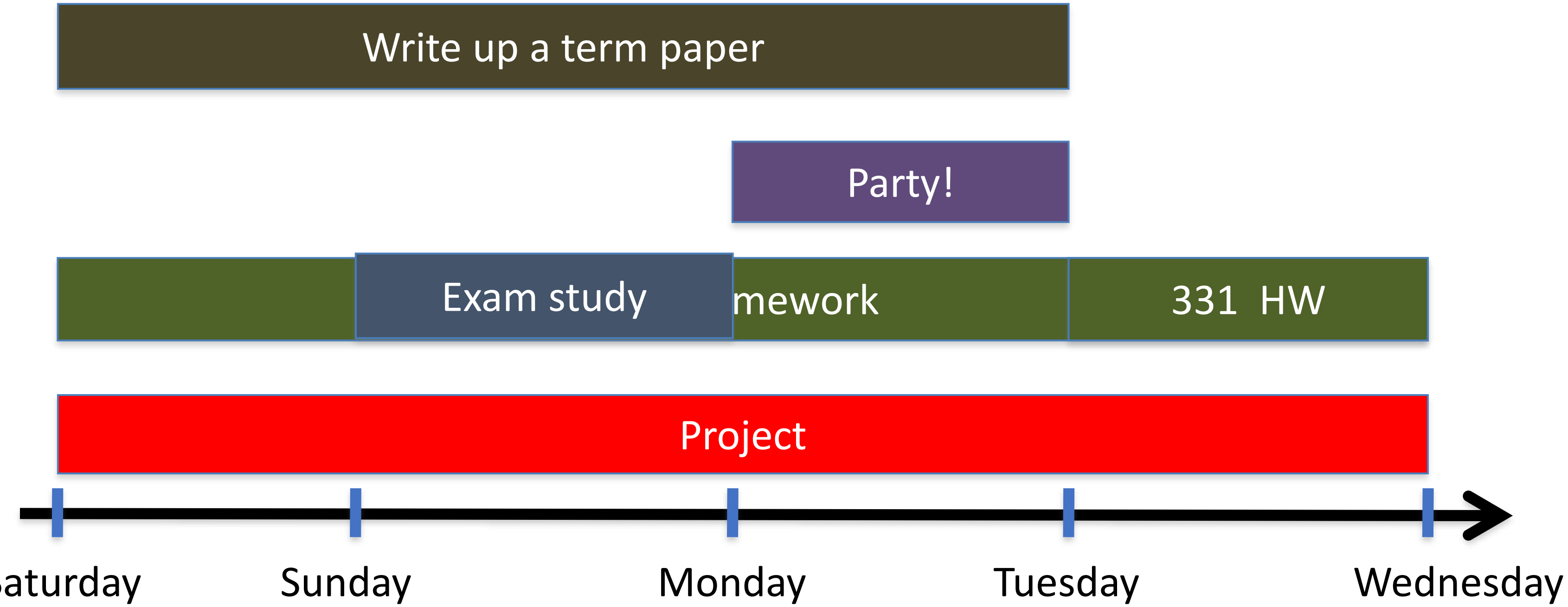
Being short sighted on each piece

Never undo a decision

Know when you see it

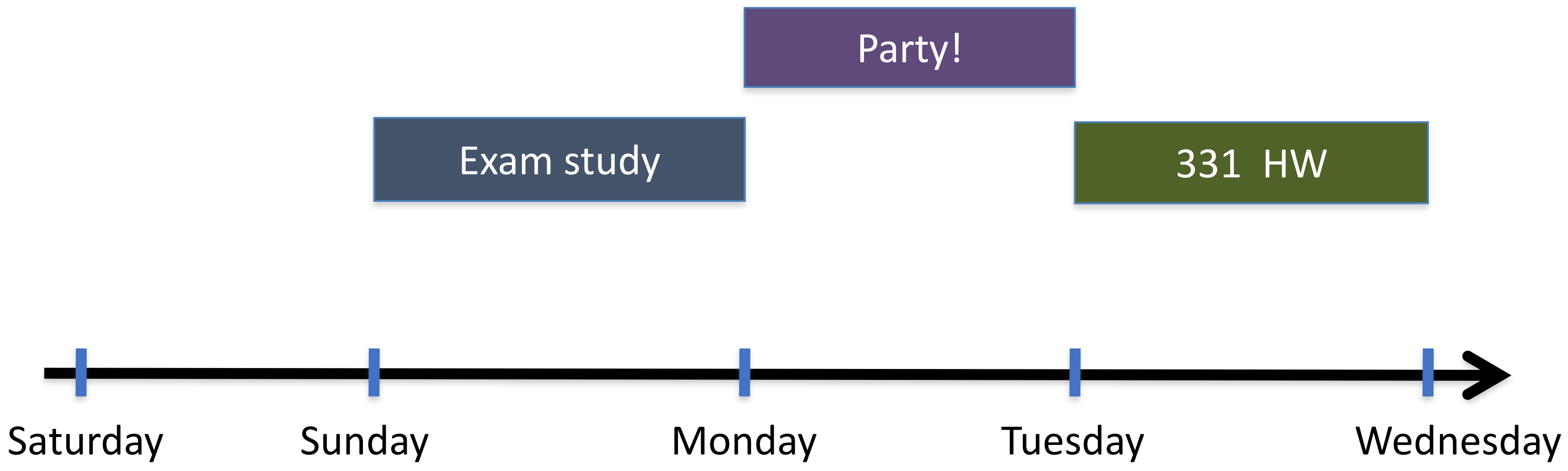
End of Semester blues

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



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 $1 \leq i \leq n$

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

Output: A *schedule* S of the n intervals

No two intervals in S conflict

$|S|$ is maximized

Algorithm with examples

Interval Scheduling via examples

In which we derive an algorithm that solves the Interval Scheduling problem via a sequence of examples.

The problem

In these notes we will solve the following problem:

Interval Scheduling Problem

`Input:` An input of n intervals $[s(i), f(i))$, or in other words, $\{s(i), \ldots, f(i) - 1\}$ for $1 \leq i \leq n$ where i represents the intervals, $s(i)$ represents the start time, and $f(i)$ represents the finish time.

`Output:` A schedule S of n intervals where no two intervals in S conflict, and the total number of intervals in S is maximized.

Sample Input and Output

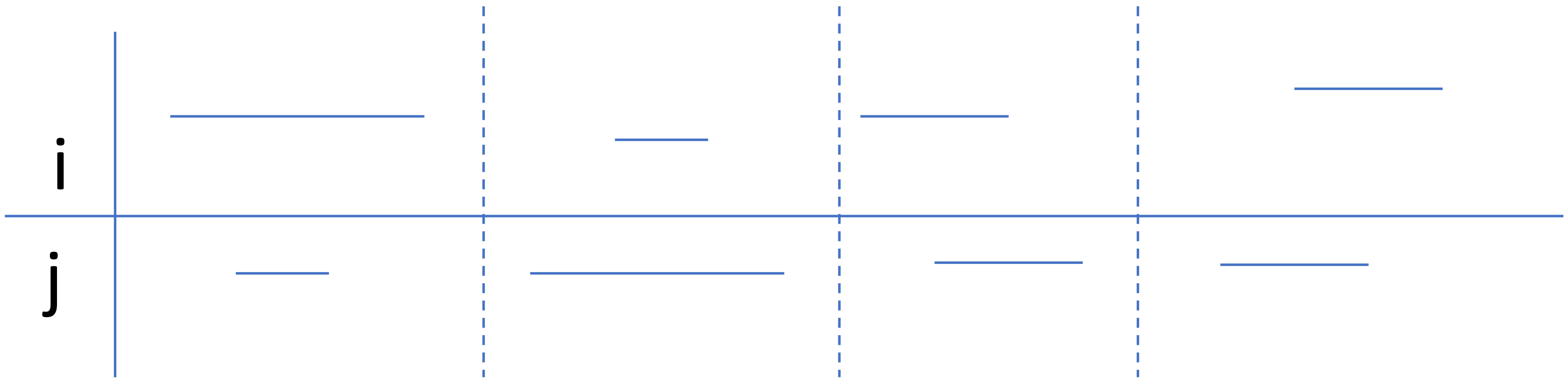
`Input:`

Interval Scheduling Problem

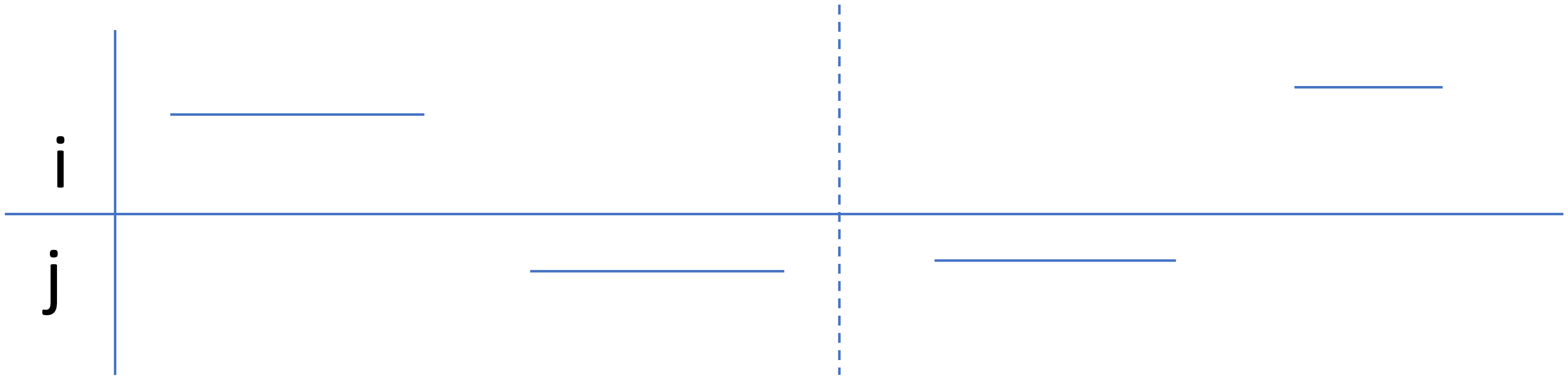
- **Input**: n intervals; ith interval: [s(i), f(i)).
- **Output**: A valid schedule with maximum number of intervals in it (over all valid schedules).
- **Def**: A schedule $S \subseteq [n]$ $([n] = \{1, 2, \ldots, n\})$
- **Def**: A valid schedule S has no **conflicts**.
- **Def**: intervals i and j conflict if they overlap.

Interval Scheduling Problem

Conflicts:

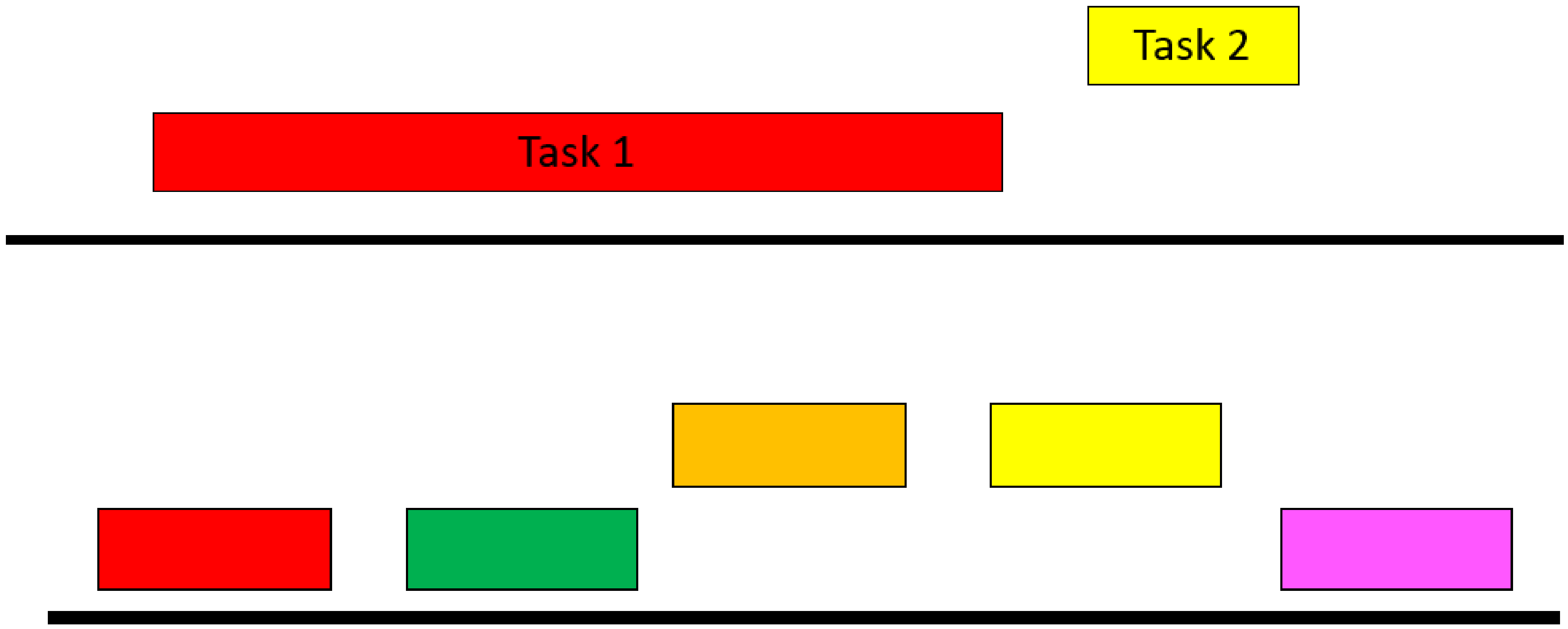


No conflicts:



Example 1

No intervals overlap



Algorithm?

No intervals overlap

R: set of requests

Set S to be the empty set

While R is not empty

  Choose i in R

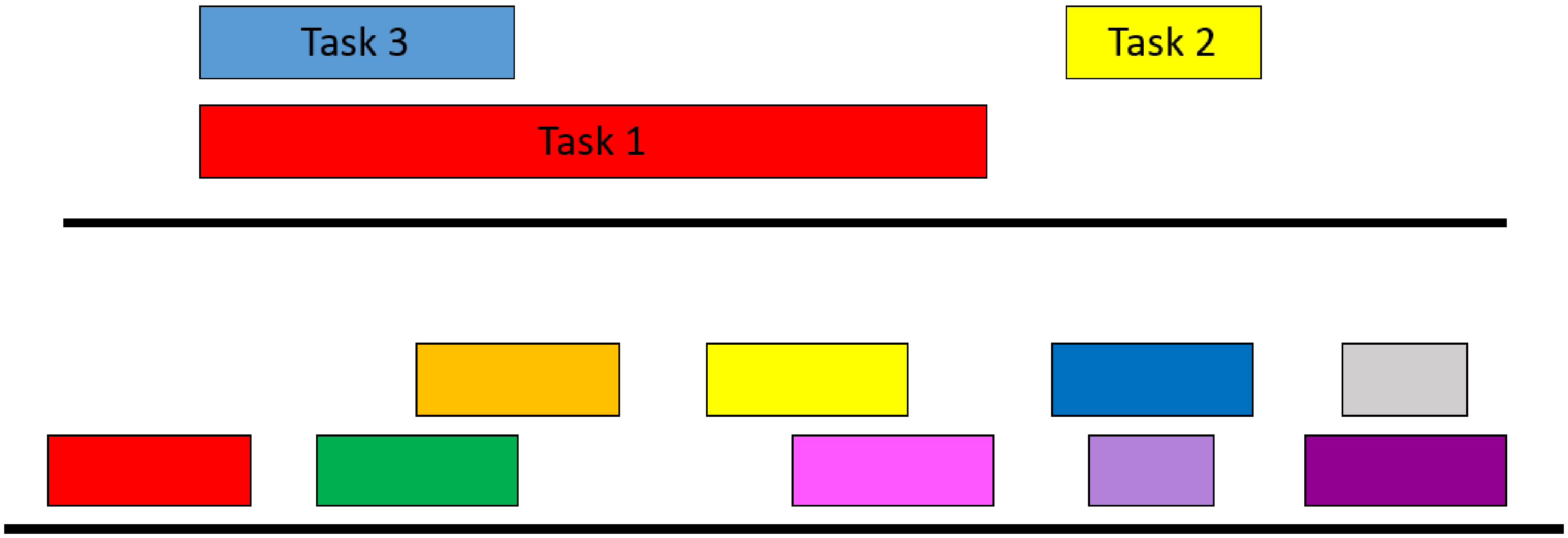
  Add i to S

  Remove i from R

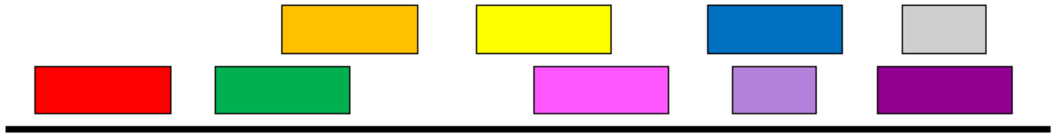
Return $S^* = S$

Example 2

At most one overlap/task



Algorithm?



At most one overlap

R: set of requests

Set S to be the empty set

While R is not empty

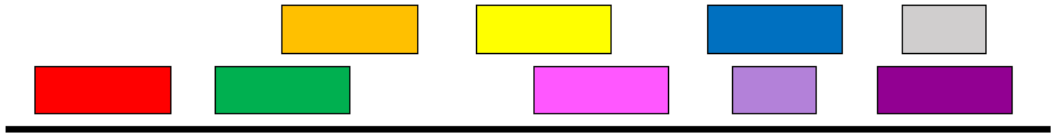
  Choose i in R

  Add i to S

  Remove i from R

Return $S^* = S$

Algorithm?



At most one overlap

R: set of requests

Set S to be the empty set

While R is not empty

    Choose i in R

    Add i to S

    Remove all tasks that conflict with i from R

Return $S^* = S$