Lecture 15

CSE 331

Few points…

- Project group signups
  - Your UBIT ID is XXX if XXX@buffalo.edu is the email ID
    - Please don’t enter your person number!
- HWs
  - Cite your sources
  - Answers should be self-contained
  - Separate out proof idea and proof details
    - Summary in idea and detailed proof in details.
  - Upload a legible PDF file. If Autograder can’t open it, we can’t grade it.
  - Please don’t cheat!
- Recitations in week 6 and 7
  - Week 6: TAs will briefly go over the sample midterm, suggest studying tips, etc.
  - Week 7: (this is the midterm week!) Q/A with the TAs.

Project groups due TODAY!

Deadline: Friday, March 4, 11:59pm

CSE 331 Syllabus Piazza Schedule Homeworks Autolab Project Support Pages channel

Project Overview

Group signup form

CSE 331 Project

Spring 2022

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
in our world and is common place in many aspects of society. The main aim of the project is to have you explore in some depth some of the

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
potentially make policy decisions. Here is a cautionary talk:

Greedy algorithms

Build the final solution piece by piece

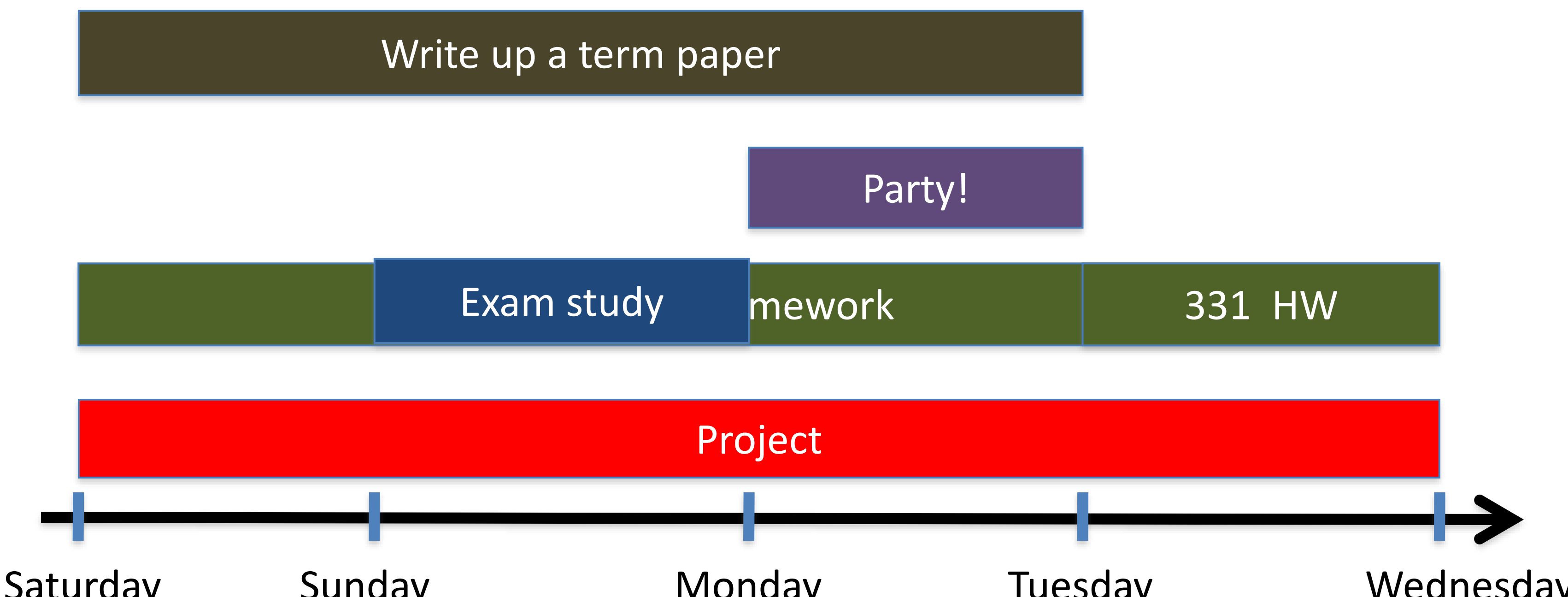
Being short sighted on each piece

Never undo a decision

Know when you see it

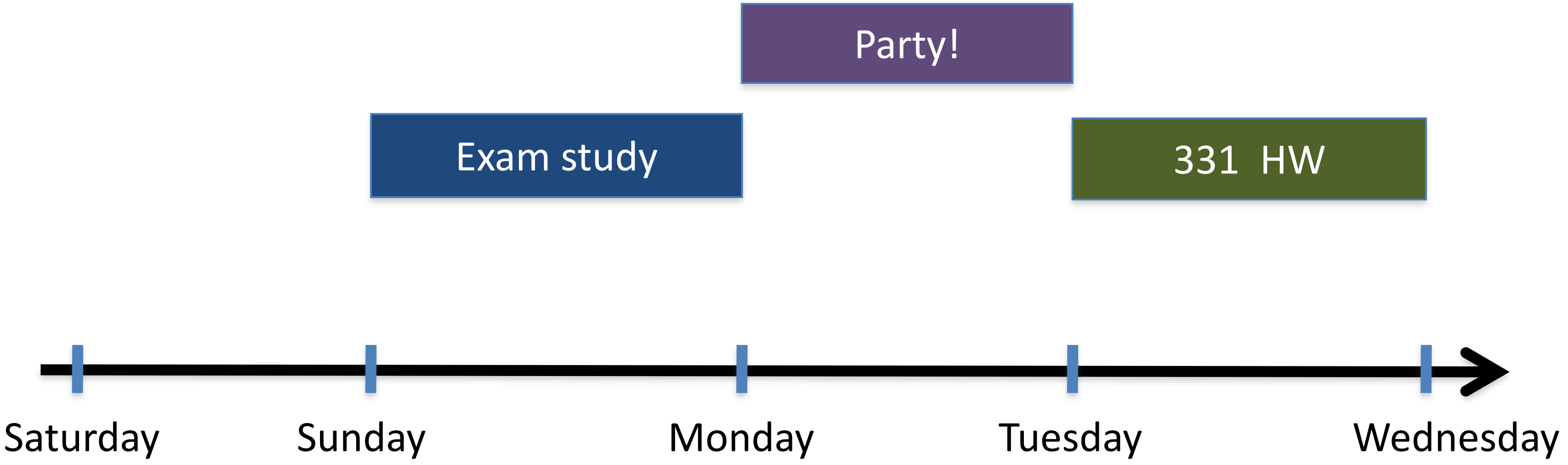
End of Semester blues

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≤ i ≤ n

{ s(i), … ,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

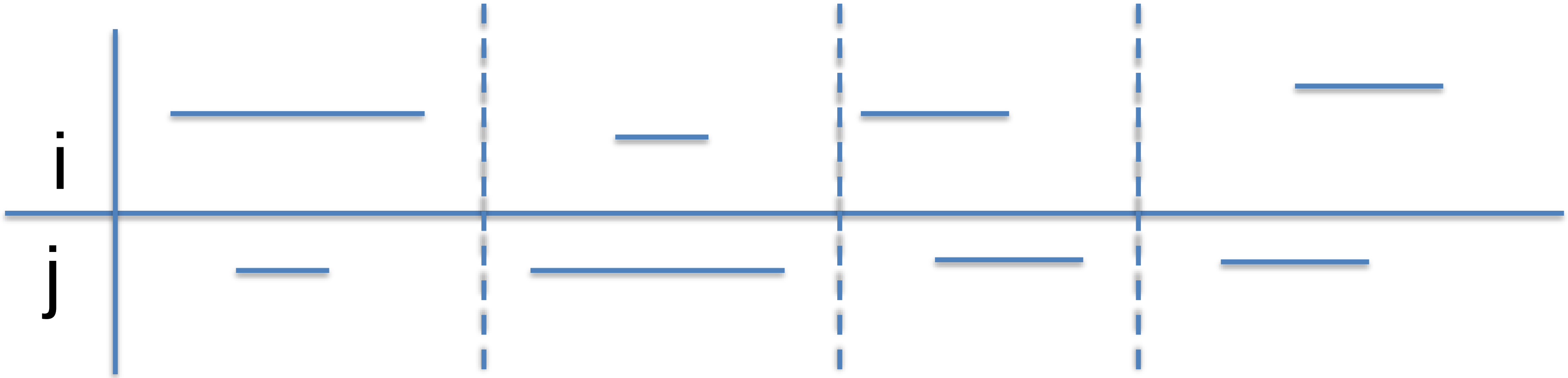
Input:

Interval Scheduling Problem

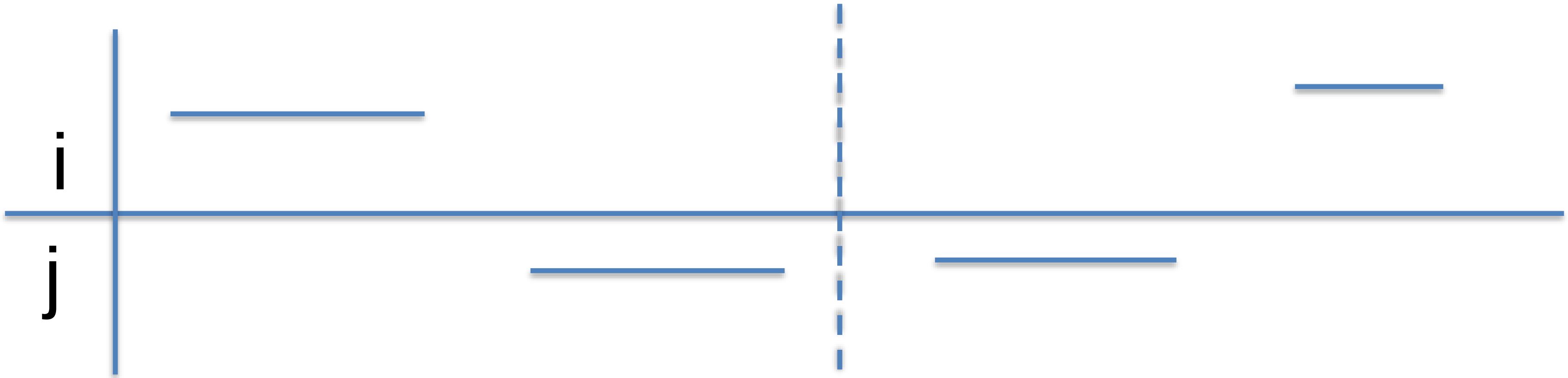
- **Input**: n intervals; ith interval: $[s(i), f(i))$.
- **Output**: A valid schedule with maximum number of intervals in it (over all valid schedules).
- **Def**: A schedule $S \subseteq [n]$ ($[n] = \{1, 2, \ldots, n\}$)
- **Def**: A valid schedule S has no **conflicts**.
- **Def**: intervals i and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

Interval Scheduling Problem

Conflicts:

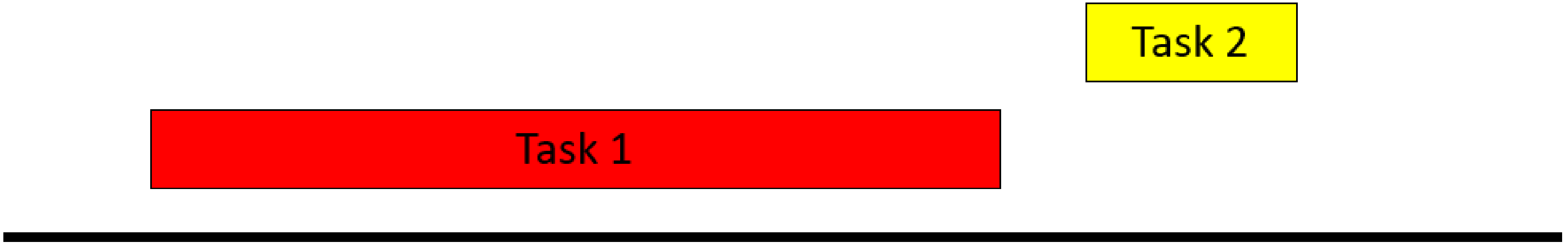


No conflicts:



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

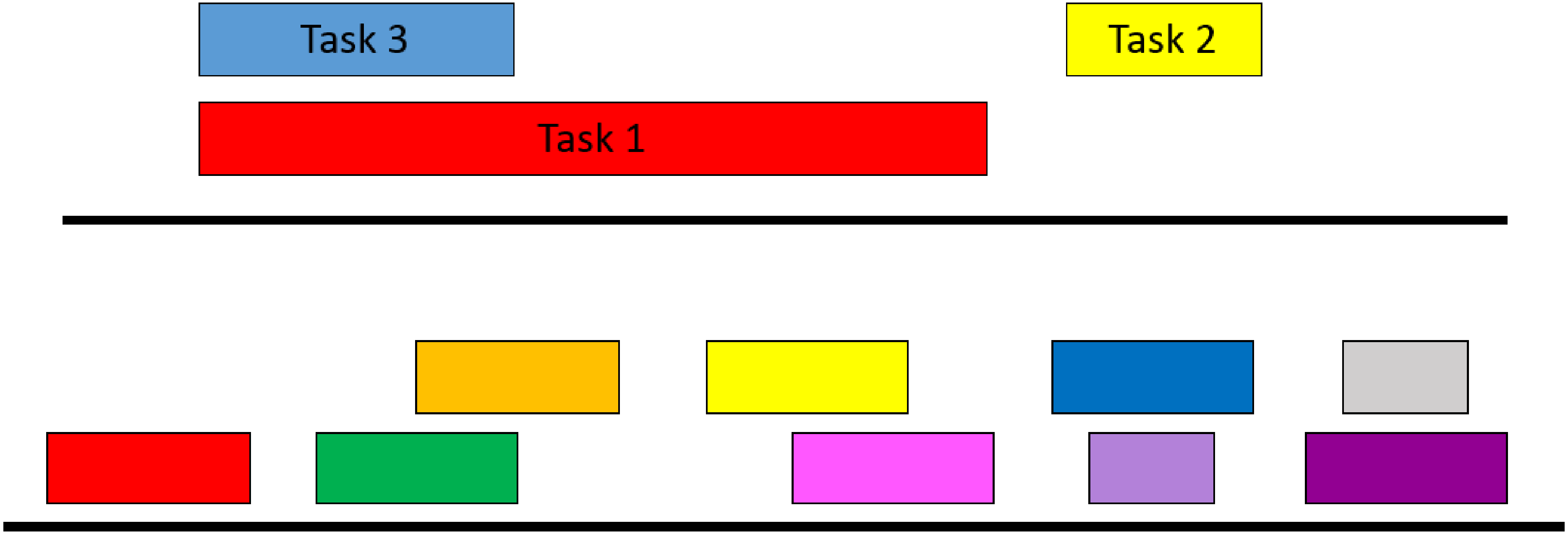
Add i to S

Remove i from R

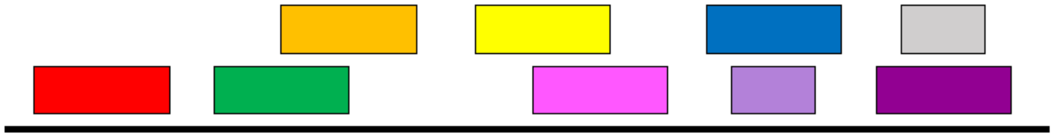
Return $S^* = S$

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

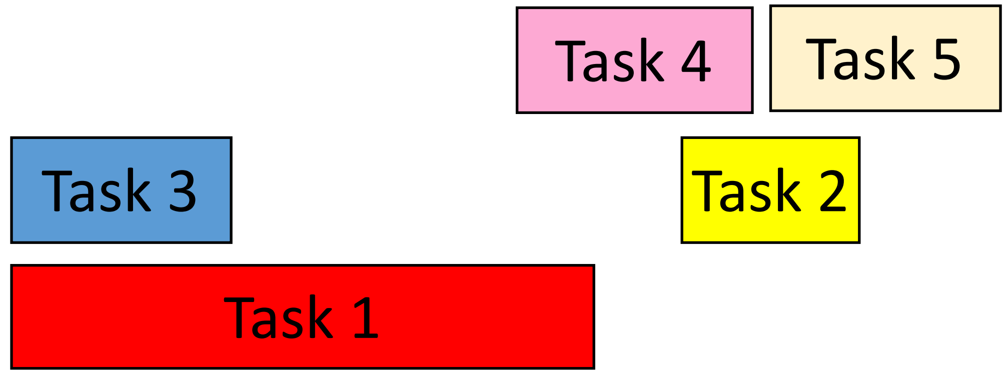
Add i to S

Remove all tasks that conflict with i from R

Return $S^* = S$

Example 3

More than one conflict



Set S to be the empty set

While R is not empty

Choose i in R

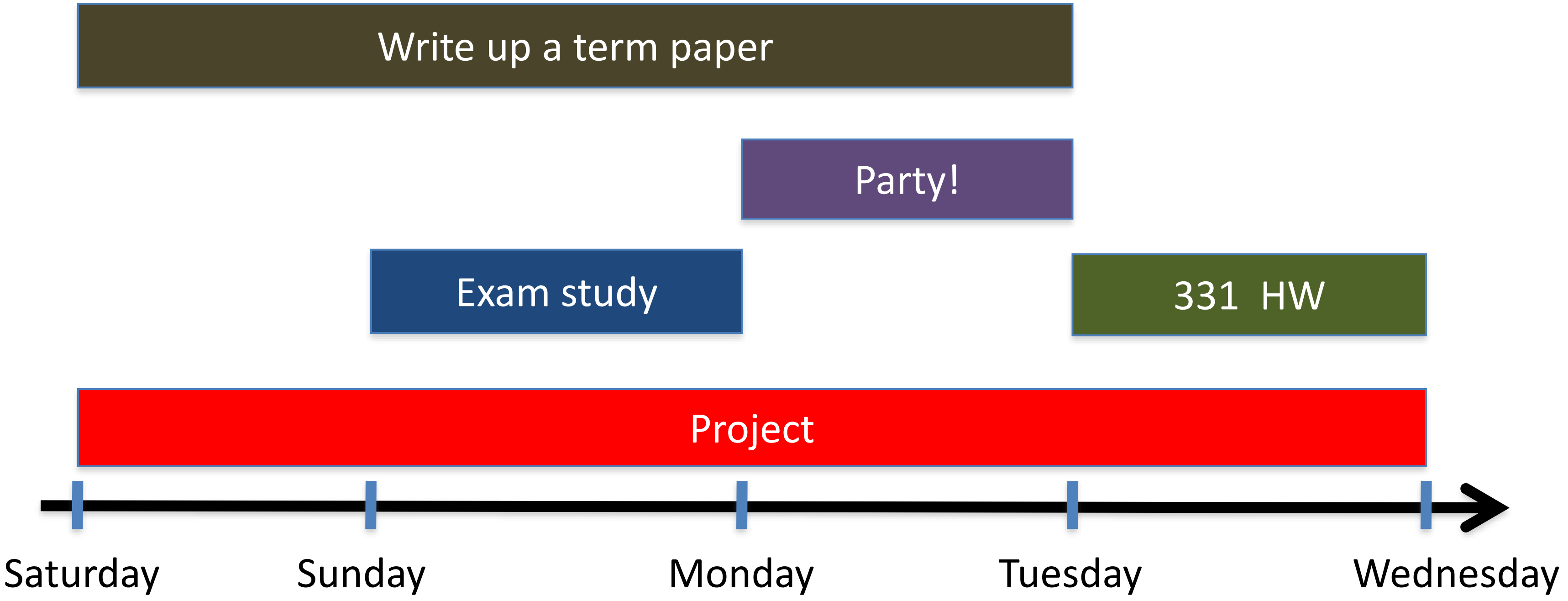
Add i to S

Remove all tasks that conflict with i from R

Return $S^* = S$

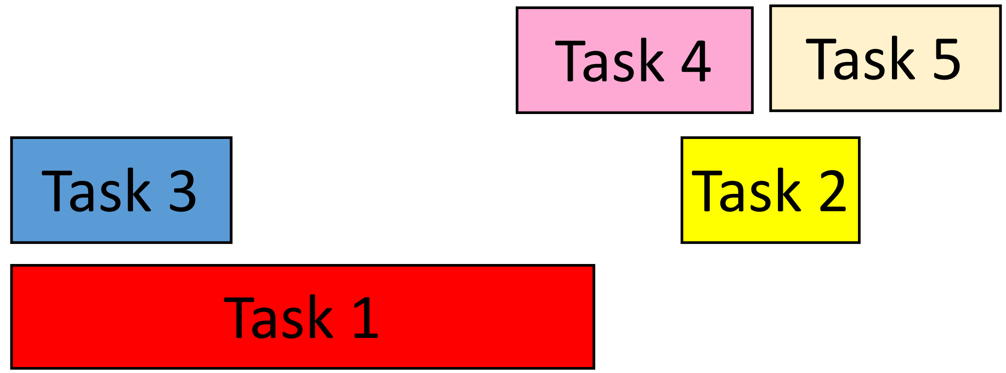
Greedily solve your blues!

Arrange tasks in some order and iteratively pick non-overlapping tasks



Making it more formal

More than one conflict



Set S to be the empty set

While R is not empty

    Choose i in R that minimizes v(i)

    Add i to S

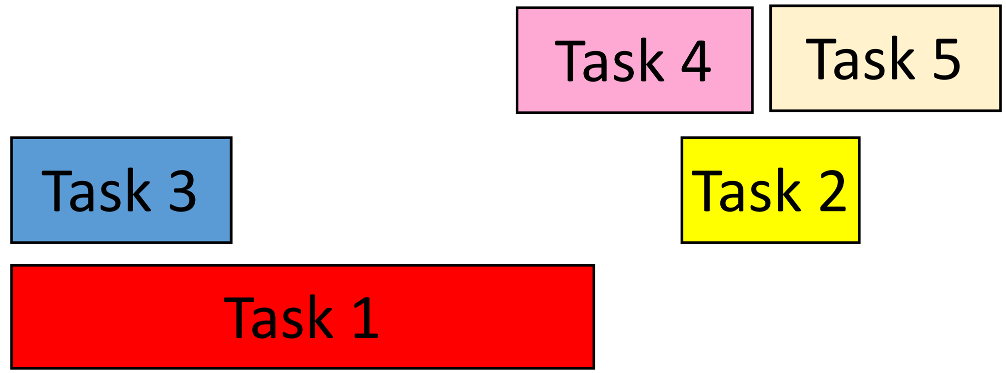
    Remove all tasks that conflict with i from R

Return $S^* = S$

Associate a value v(i) with task i

What is a good choice for v(i)?

More than one conflict



Set S to be the empty set

While R is not empty

Choose i in R that minimizes v(i)

Add i to S

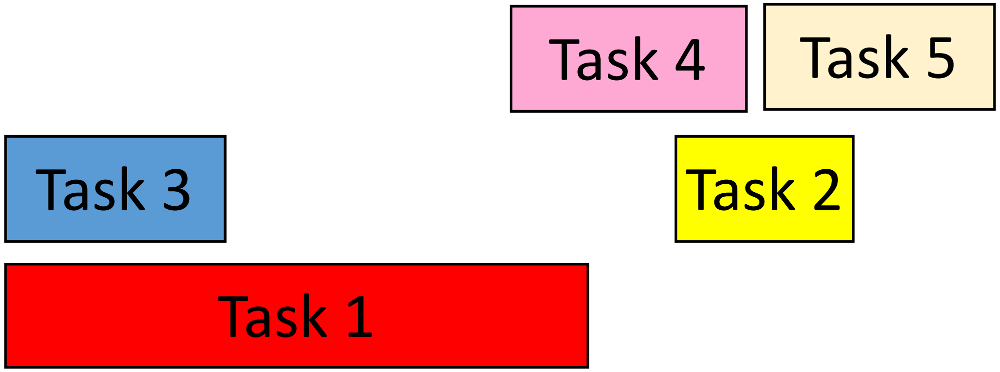
Remove all tasks that conflict with i from R

Return $S^* = S$

Associate a value v(i) with task i

$v(i) = f(i) - s(i)$

Smallest duration first



Set S to be the empty set

While R is not empty

Choose i in R that minimizes $f(i) - s(i)$

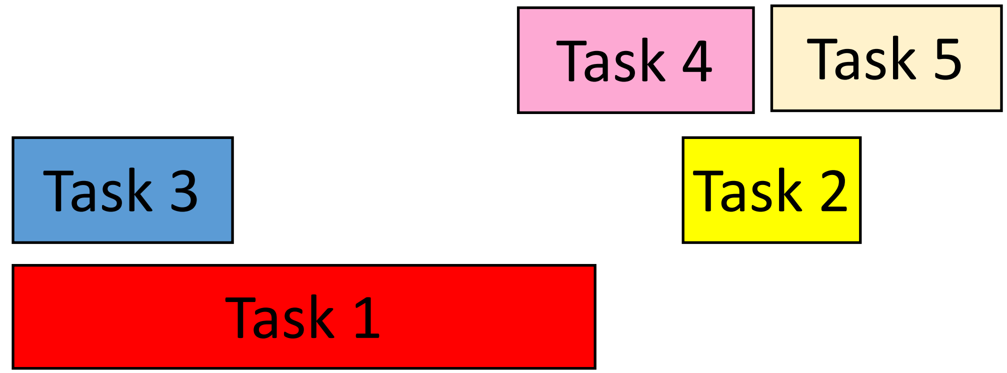
Add i to S

Remove all tasks that conflict with i from R

Return $S^* = S$

$v(i) = s(i)$

Earliest time first?



Set S to be the empty set

While R is not empty

Choose i in R that minimizes s(i)

Add i to S

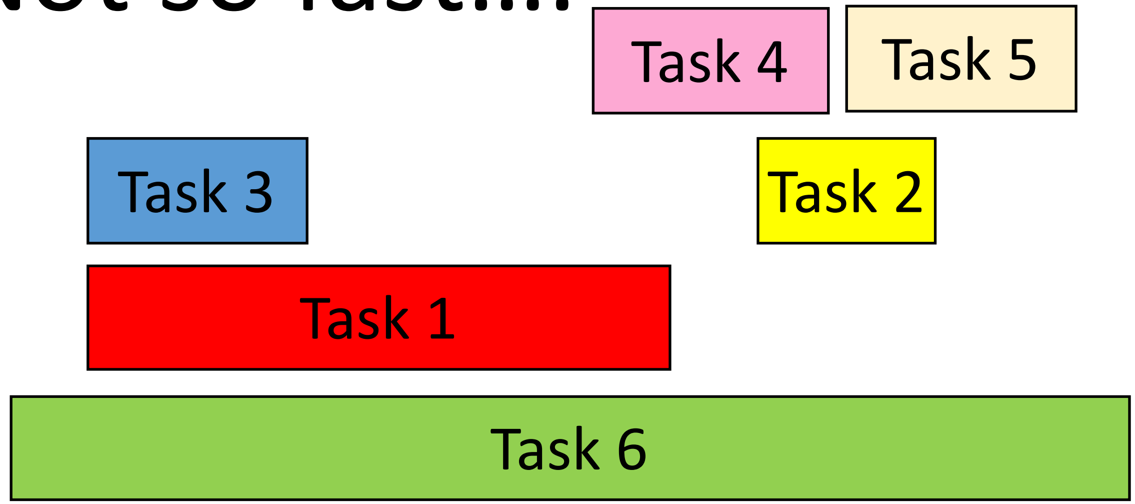
Remove all tasks that conflict with i from R

Return $S^* = S$

So are we done?

Not so fast….

Earliest time first?



Set S to be the empty set

While R is not empty

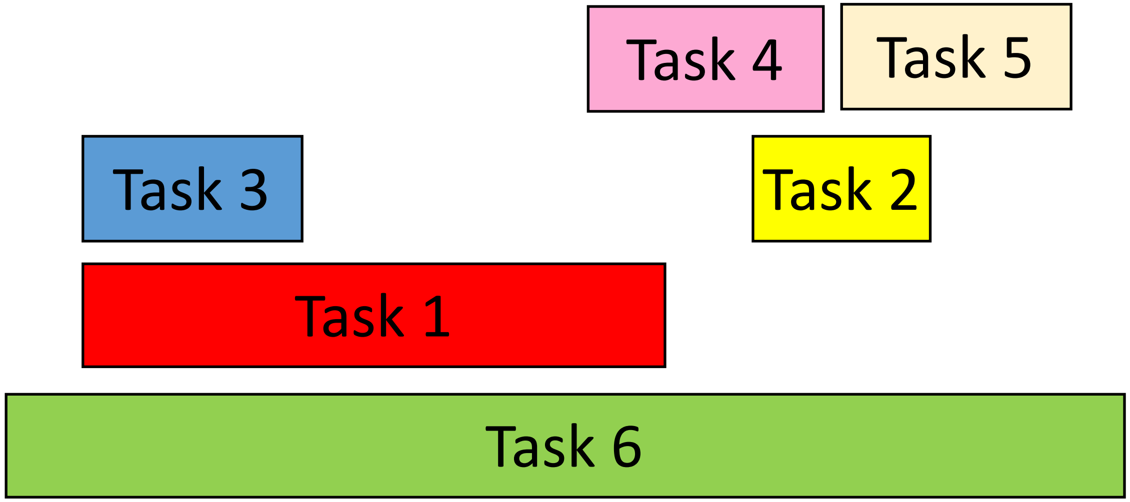
Choose i in R that minimizes s(i)

Add i to S

Remove all tasks that conflict with i from R

Return S^*= S

Pick job with minimum conflicts



Set S to be the empty set

While R is not empty

    Choose i in R that has smallest number of conflicts
    Add i to S
    Remove all tasks that conflict with i from R

Return $S^* = S$

So are we done?

Nope (but harder to show)

Set S to be the empty set

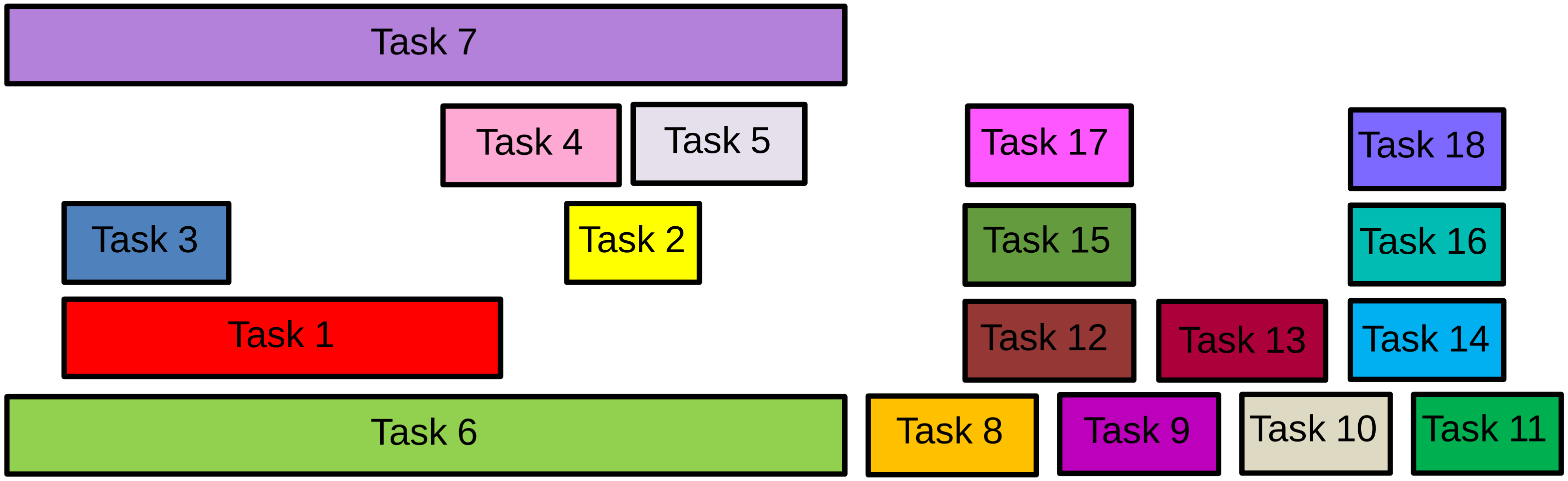
While R is not empty

    Choose i in R that has smallest number of conflicts

    Add i to S

    Remove all tasks that conflict with i from R

Return $S^* = S$



Set S to be the empty set

While R is not empty

Choose i in R that has smallest number of conflicts

Add i to S

Remove all tasks that conflict with i from R

Return S^* = S