Lecture 15

CSE 331

Sep 30, 2019

If you need it, ask for help

Quiz 1 on Monday

note

stop following 96 views

Quiz 1 on Monday, Oct 7

The first quiz will be from **1-1:10pm in class** on **Monday, October 7**. We will have a 5 mins break after the quiz and the lecture will start at 1:15pm.

We will hand out the quiz paper at 12:55pm 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: @641.) Also quiz 1 will cover all topics we cover in class till Friday, Oct 4.

#pin

quiz1

edit · good note | 0

Updated 1 day ago by Atri Rudra

Mid-term post 1

note

4 views

The mid-term post

First, midterm-I is on **Monday, Oct 14** and midterm-II is on **Wednesday, Oct 16** during the usual class timings (i.e. 1:00-1:50pm in Norton 112). Below are some comments that might be helpful to prepare for the mid-term.

(Thoughts on what to do *during* the exam here: @662)

- Work through the sample mid-term exams (@641). 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 (@642) 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 11. (Details TBA later this week.)
- The exam will be closed book and closed notes. However, you can bring in **one** 8.5" X 11" review sheet. (If you prefer you can bring in different review sheets for the two mid-term exams.) You can write anything that you want on the sheet as long as it is one sheet (you can use both sides). It can hand-written or typed up doesn't matter-- however, you are not allowed to bring in a magnifying glass. The review sheet is to make sure you do not spend time memorizing definitions etc. but can concentrate on the main ideas in the material we have covered. The exam (as you can probably make out from the sample mid-term) will focus on how well you understand the material and not how well you can memorize. However, see next point.
- **Do not spend too much time cramming stuff into the review sheet**. In my experience (both as a student and instructor), it never helps to just put in arbitrary stuff. **However, you should use the review sheet to write down references for various algos etc. we have seen in class/HWs/recitation notes etc., so that you can just read off the reference during the exams.**
- We know that you have much less time to write answers on the exams than the homeworks so a correct proof/algorithm idea will be worth at least **80%** of the grade. (If the question does not *specifically* ask for proof/algorithm detail, then we're *only* looking for a proof/algorithm idea.) Further for the question on mid-term I the justifications only have to be at the level of a proof idea.

Mid-term post 2

note

stop following 3 views

Few thoughts on what to do during the exam

In a previous post @663, 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 its 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 be 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.
     - Remember how easy it was to get most points on 1(a) and 2(a) by just referring to the recitation notes. Y'all should try to do that as much as possible!
     - **If you try and build something from scratch (like and algorithm or a proof) that you could have just referenced away, then you will be short on time**. *The exam is timed in such a way that if something can be just referenced, then you are expected to do so.*
   - If the above fails then you should see if you can slightly tweak a previous solution to work in the current situation. *Most* of the problems in the mid terms will either be in the previous category or this one.
   - If both of the above fail, then try to answer from scratch but **this should be your last option.**
3. **Use the cheatsheet well.** Think about what parts of 331 you struggle with and write those down on the cheatsheet, This e.g. could be runtimes of various algorithms or maybe outline of an algorithm etc. You should not waste real estate on things you can recollect immediately rather you should use it as an aid to remember things immediately. As another example, if for some reason you cannot never remember how to solve a particular problem or T/F poll etc: just write that down. Or if you have trouble remembering precise references to existing algorithms/proofs, then write those down on the cheatsheet. On the other hand do not spend a huge amount of time writing up a cheatsheet (at the expense of reviewing the material itself)

My UG algorithms story

When I was an undergrad

- Took algorithms as a sophomore

Understood all the lectures

Did not study outside of lectures

- (We had no homeworks)

Did decent on the mid-term

Nearly flunked the finals

Got a **C**

Questions?

Interval Scheduling Problem

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

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

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.

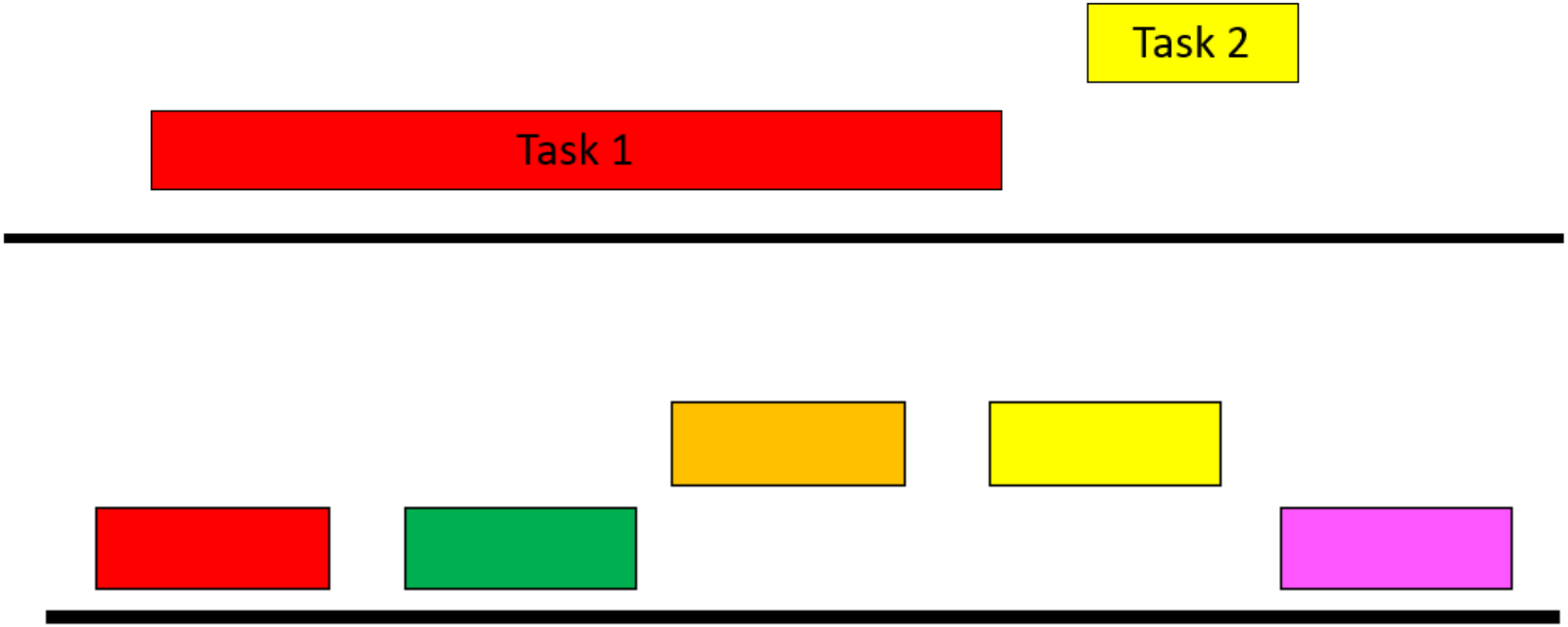
Output: A schedule S of n intervals where no two intervals in S conflict, and the total number of intervals in S is maximized.

Sample Input and Output

Input:

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

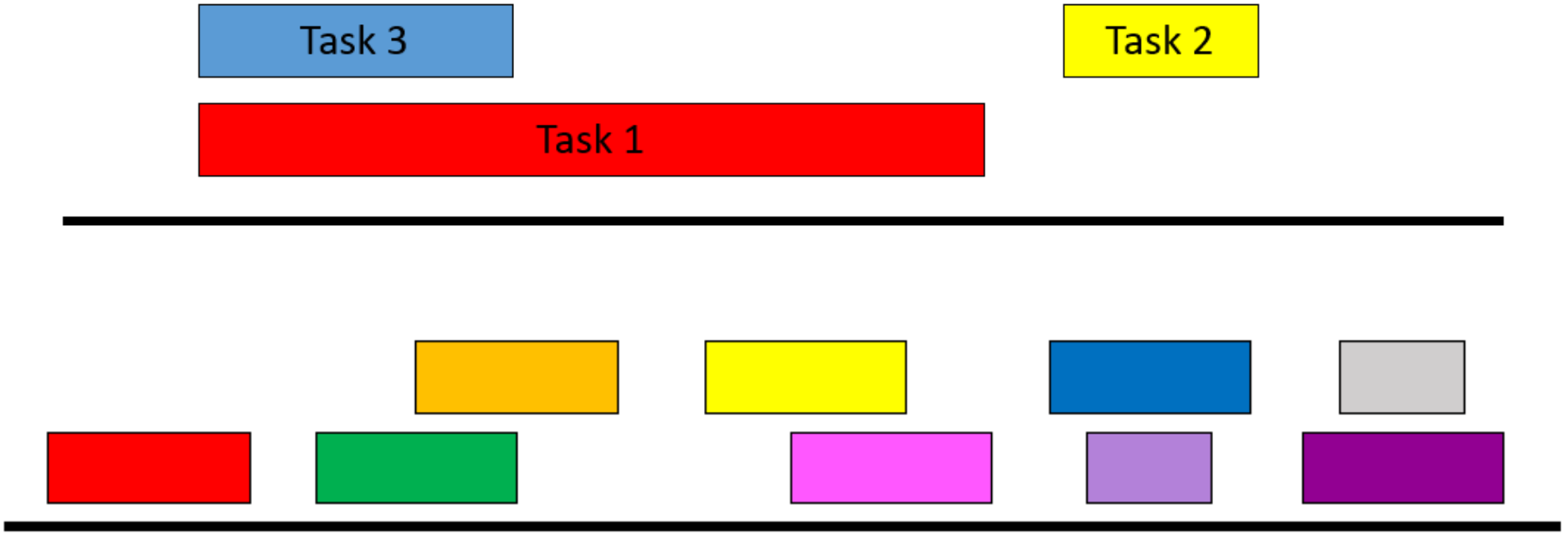
Remove i from R

Return $S^* = S$

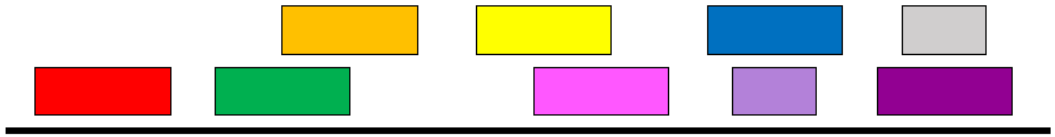
Questions?

Example 2

At most one overlap



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

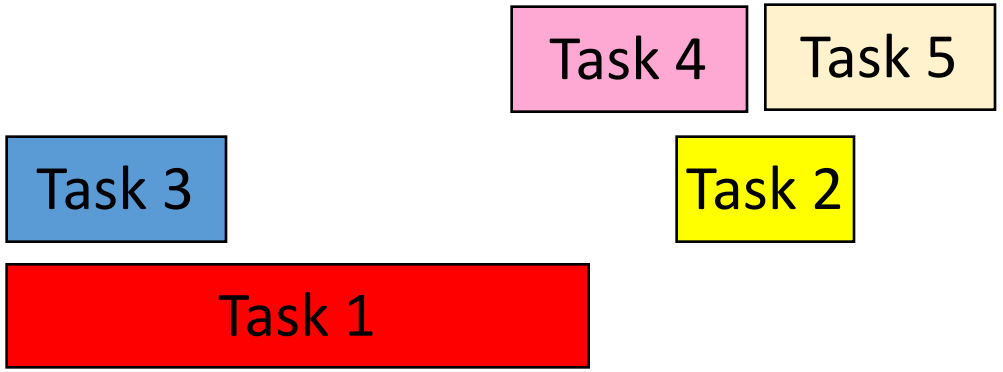
Add i to S

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

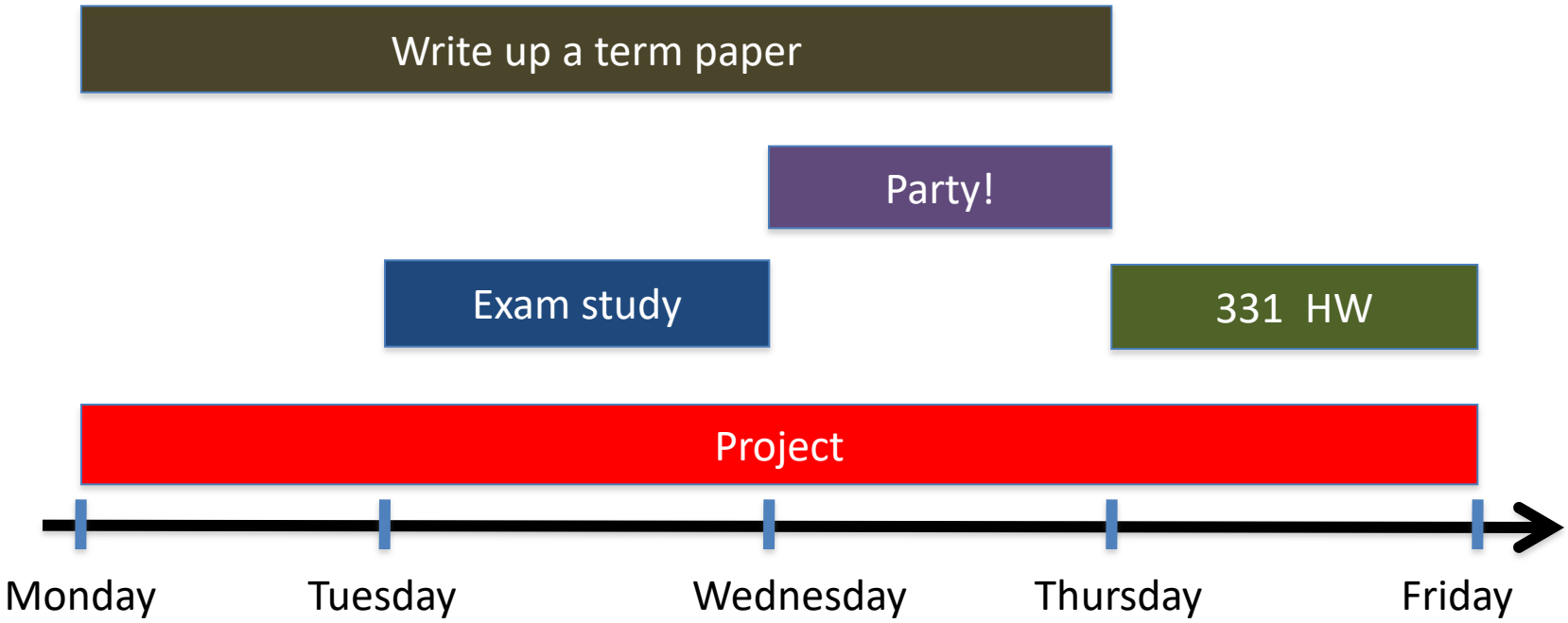
Add i to S

Remove all tasks that conflict with i from R

Return $S^* = S$

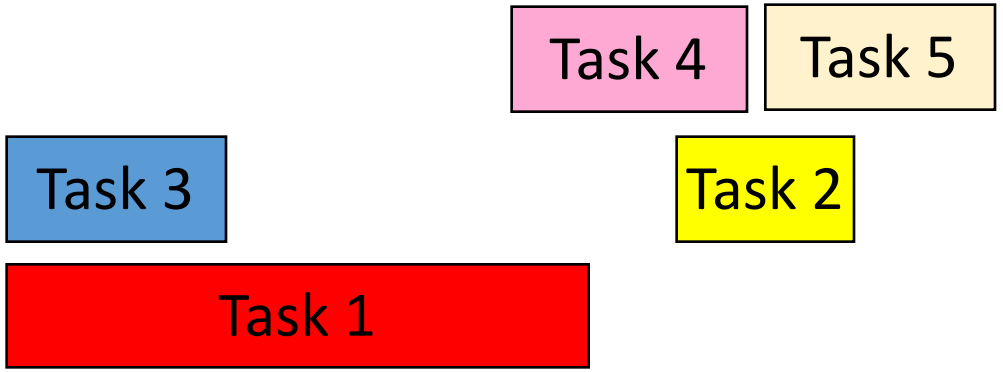
Greedily solve your blues!

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

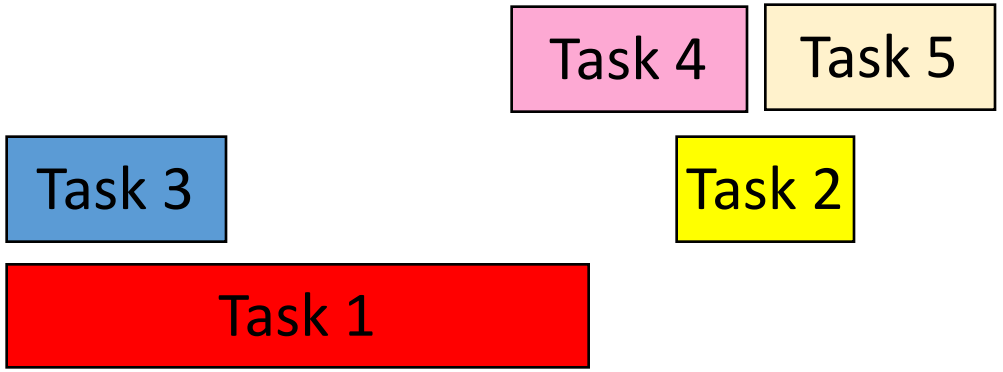
Remove all tasks that conflict with i from R

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

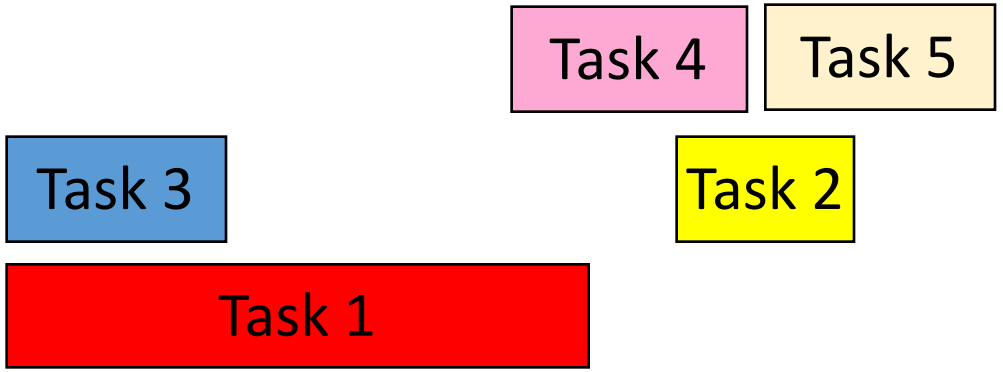
Remove all tasks that conflict with i from R

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)

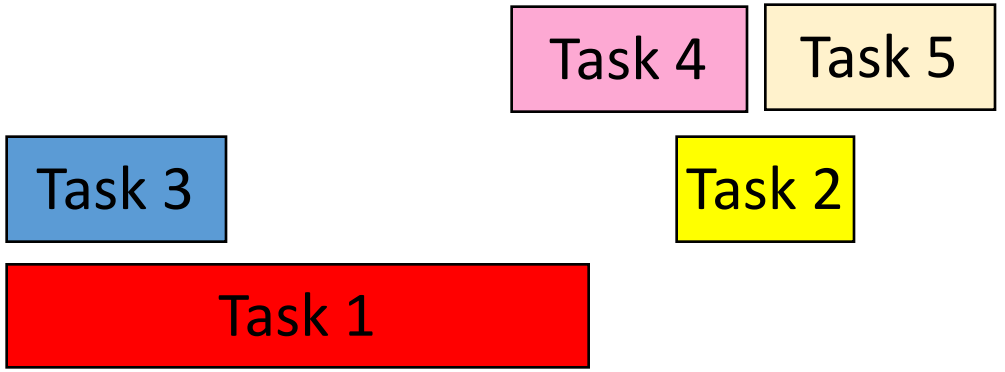
Add i to S

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

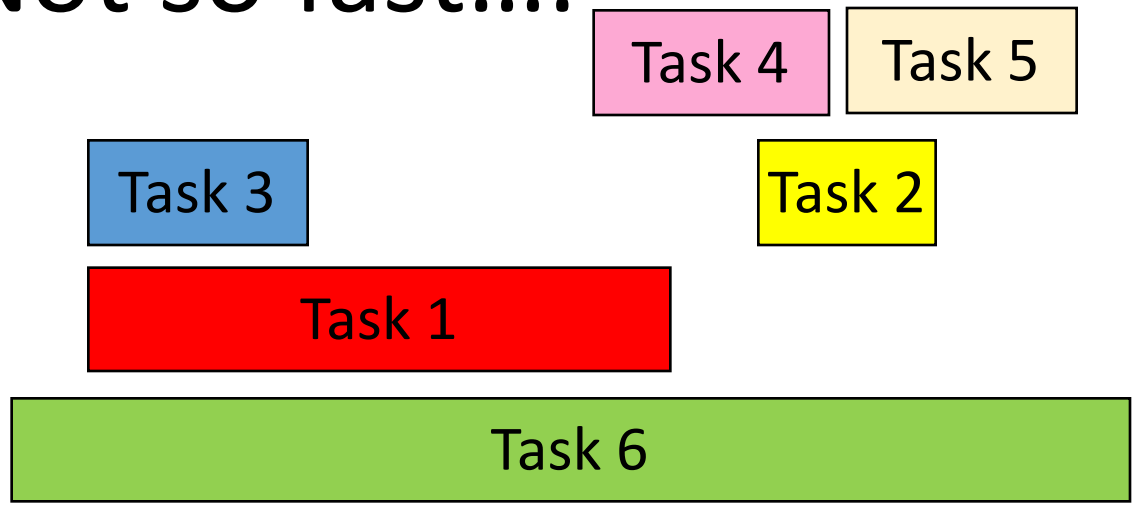
Remove all tasks that conflict with i from R

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

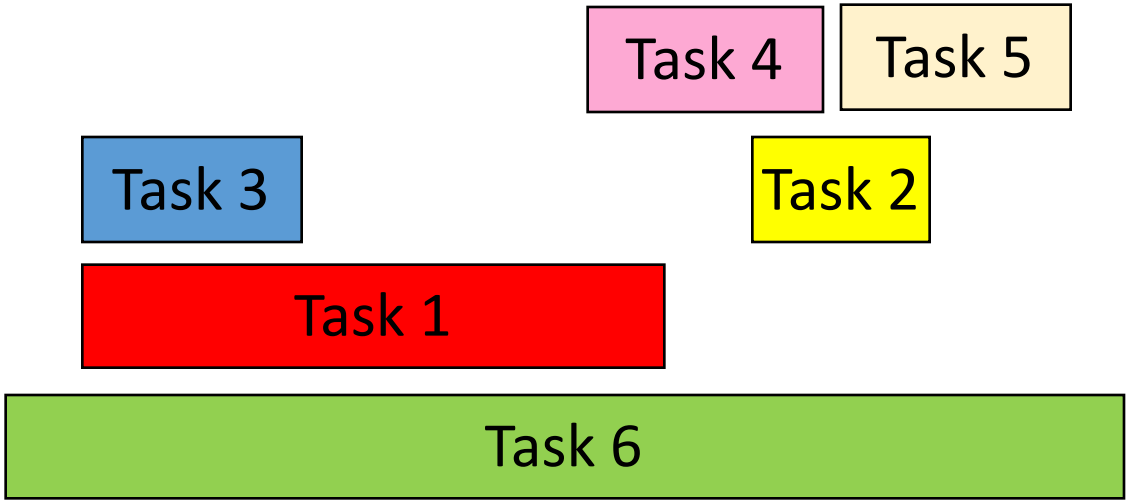
Choose i in R that minimizes s(i)

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

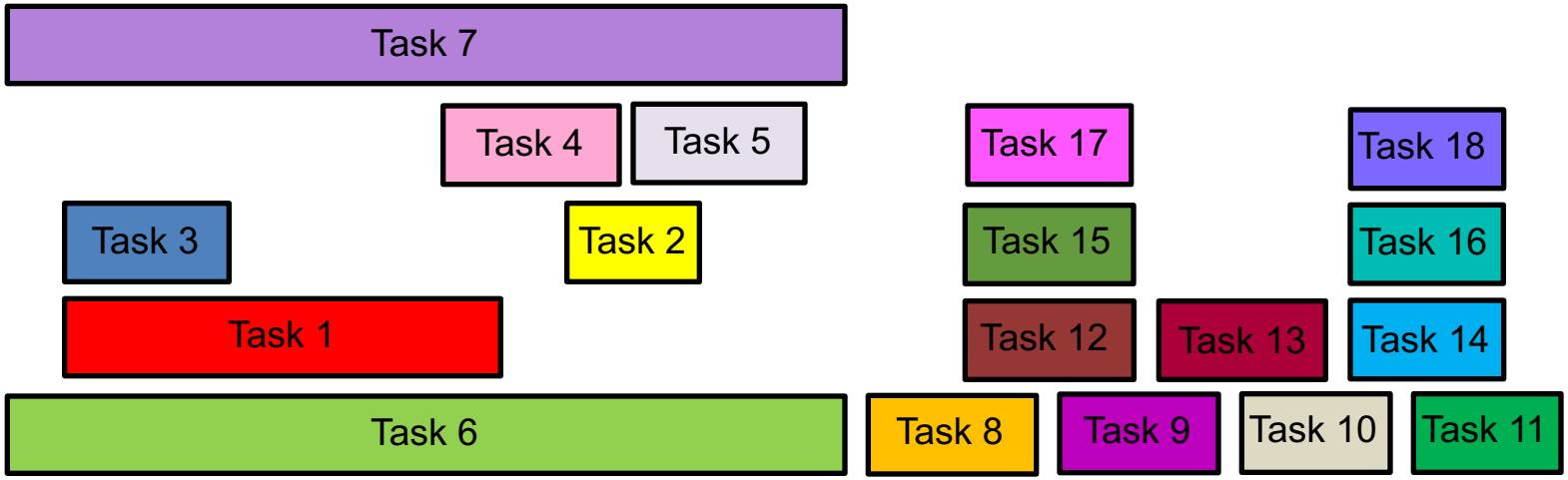
While R is not empty

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

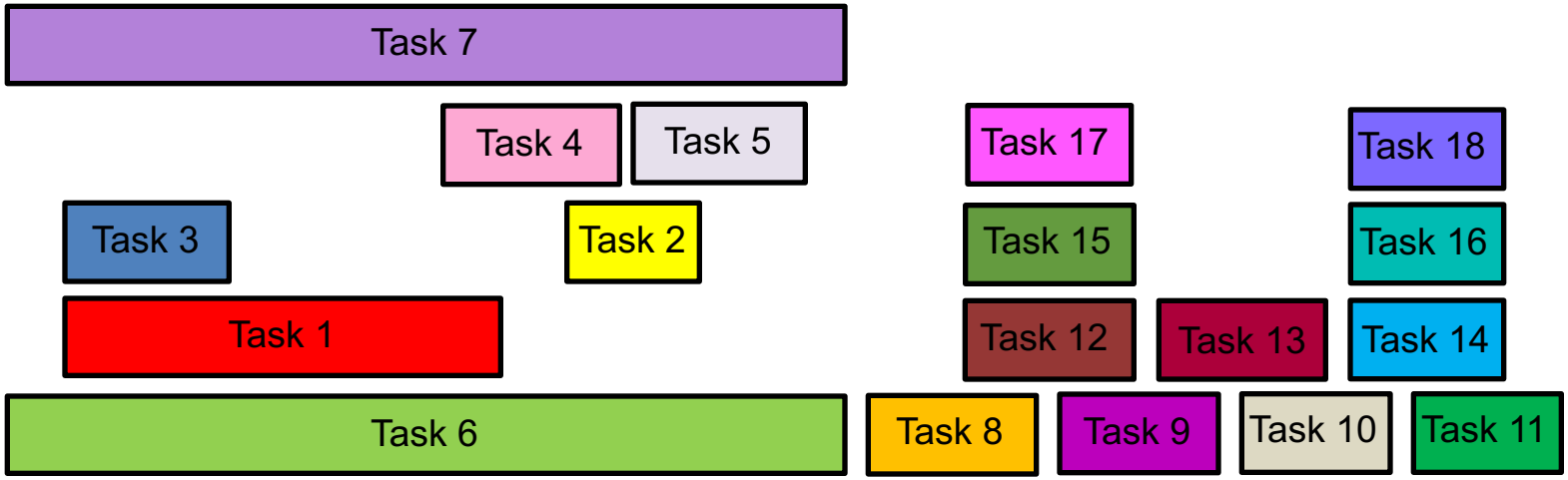
Choose i in R that has smallest number of conflicts

Add i to S

Remove all tasks that conflict with i from R

Return $S^* = S$

Algorithm?



Set S to be the empty set

While R is not empty

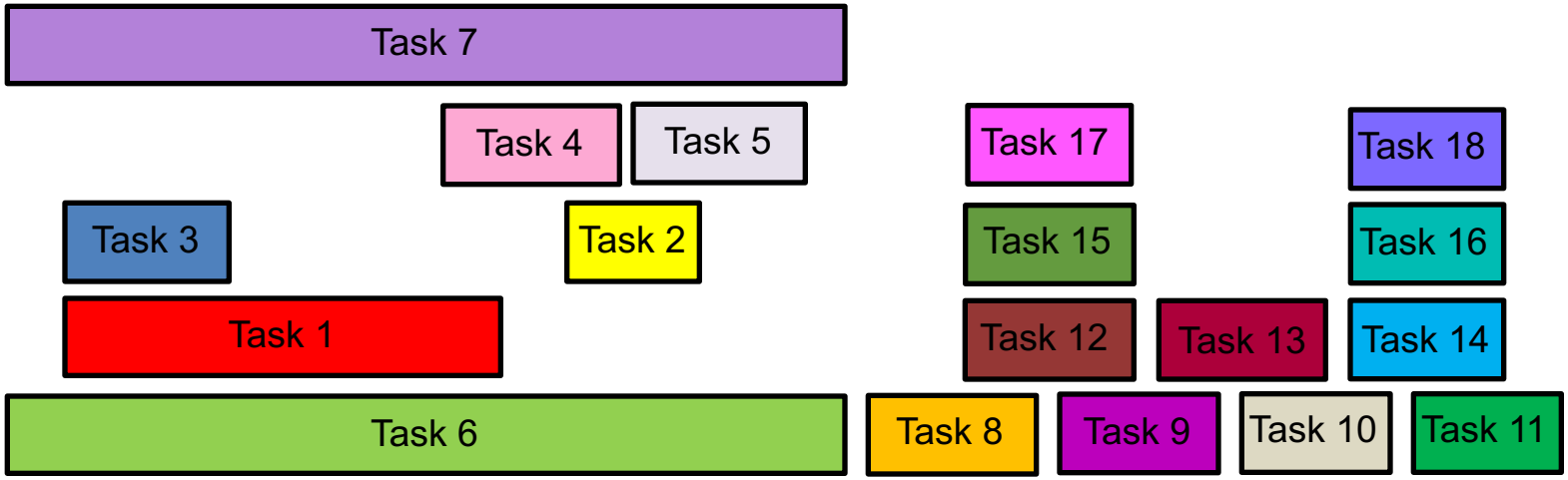
Choose i in R that minimizes v(i)

Add i to S

Remove all tasks that conflict with i from R

Return $S^* = S$

Earliest finish time first



Set S to be the empty set

While R is not empty

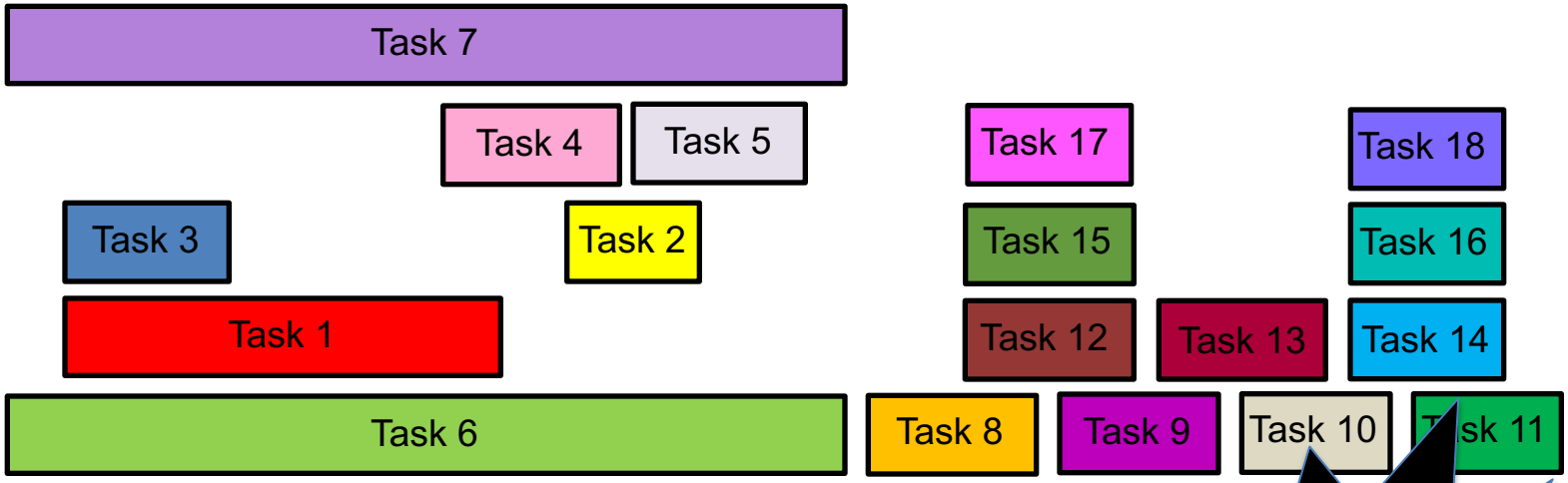
Choose i in R that minimizes f(i)

Add i to S

Remove all tasks that conflict with i from R

Return $S^* = S$

Find a counter-example?



Set S to be the empty set

While R is not empty

Choose i in R that minimizes f(i)

Add i to S

Remove all tasks that conflict with i from R

Return $S^* = S$

It works!

Questions?

Today’s agenda

Prove the correctness of the algorithm

Final Algorithm

R: set of requests

Set S to be the empty set

While R is not empty

    Choose i in R with the earliest finish time

    Add i to S

    Remove all requests that conflict with i from R

Return $S^* = S$