Lecture 16

CSE 331

Oct 2, 2024

Upcoming exams

Mid-term 1 on Mon Oct 7

Mid-term 2 next Wed Oct 9

Sample mid-terms

note @133 stop following 40 views

Actions

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/fall24/index.html

I would highly recommend that you do **not** peek into the solutions until you have tried to solve the sample mid-terms on your own first.

Here are the direct links:

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

Over the next day or two, I will 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). But for now, here is one important clarification: **The sample mid-terms are ONLY meant for y'all to get familiar with the format of the exams**. *You should not deduce ANYTHING about the coverage of topics or hardness of the exams from the sample mid-terms above*. (The mid-term post will talk more about the topics and the actual mid-terms will be harder than the sample exams above).

The questions in quiz 1 on Monday @126 will have the exact same format as the questions in the sample mid-term 1.

mid-term

Edit good note 0 Updated 1 day ago by Atri Rudra

Mid-term post 1

note @134 stop following 38 views

Actions

The mid-term post

First, midterm-I is on **Monday, Oct 7** and midterm-II is on **Wednesday, Oct 9** during the usual class timings (i.e. 11:00-11:50am in KNOX 104). Below are some comments that might be helpful to prepare for the mid-term.

(Thoughts on what to do *during* the exam here: @135)

- Work through the sample mid-term exams (@133). 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 Sep 30 (@126) to get some practice in solving T/F questions under some time pressure. Also review the T/F polls (@41) for more examples of such T/F questions.
- Review the HW problems/solutions. HW solutions are here: @46.
- 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 (recitations next week will be review sessions) and office hours.
- 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.** Here are some of suggestions on what to put in your cheatsheet:
    - If there are things that you cannot for whatever reason remember, note those down.

Mid-term post 2

note @135 stop following 37 views

Actions

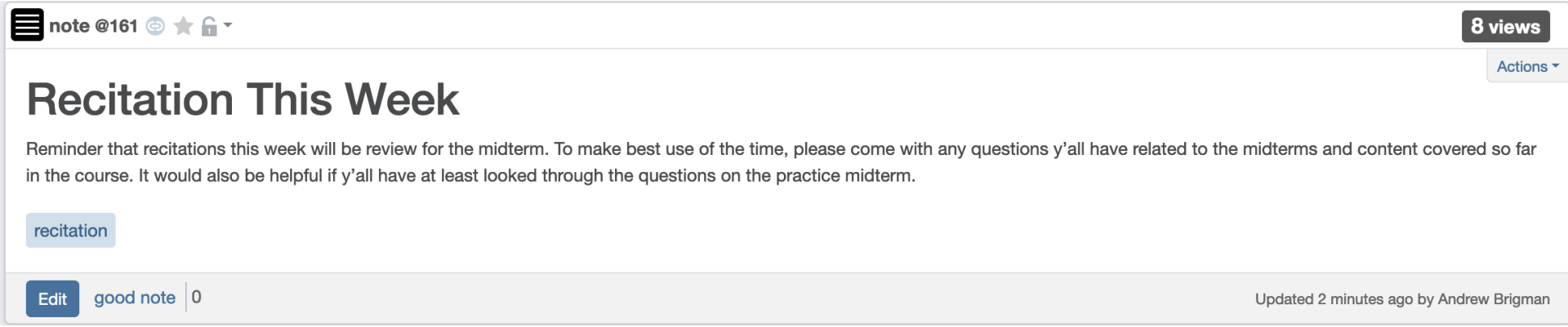
Few thoughts on what to do during the exam

In a previous post @134, I listed some pointers on how *I* think you should prepare for the mid-term exams.

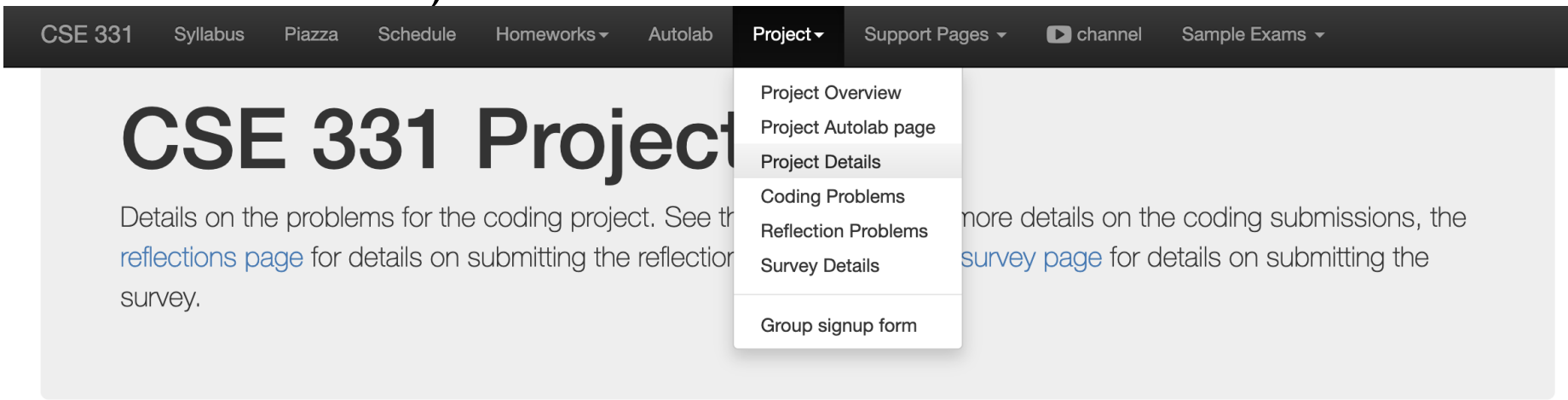
Below are (in no particular order) some thoughts on how you should work on the actual exam:

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 *black box*. 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 (very) 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.**

Recitations this week



Project has been released!



Acknowledgment

The development of the project was supported by a Mozilla Responsible Computer Science award. The support is gratefully acknowledged.

Some Suggestions and Warnings

While this coding parts of the project is somewhat similar to Question 3s on the homework, there are some crucial differences and we wanted to highlight few things for y'all upfront:

Form groups of size EXACTLY 3

This is a group project (unlike Q3s on the HWs that had to be done individually) and you can work in groups of size **exactly 3**. The submissions will be on Autolab and *everyone in the group will get the same grade.*

Groups should agree on one programming language

While the submission for the coding problems in this project can be done in any of `C++,Java` and `Python` like in Question 3s on the homework, we **highly recommend that the group agrees on one programming language for the group**. This will make it much easier for your group to make progress and collaborate.

Withdraw submission due to AI

Withdrawing a submission for academic integrity violation

Sometime mistakes can happen so you have the **option of withdrawing any of your Homework submission with 24 HOURS of the assignment deadline**. You can do this by sending Atri an email, e.g. by using the following template (thanks to Oliver Kennedy for providing us the template):

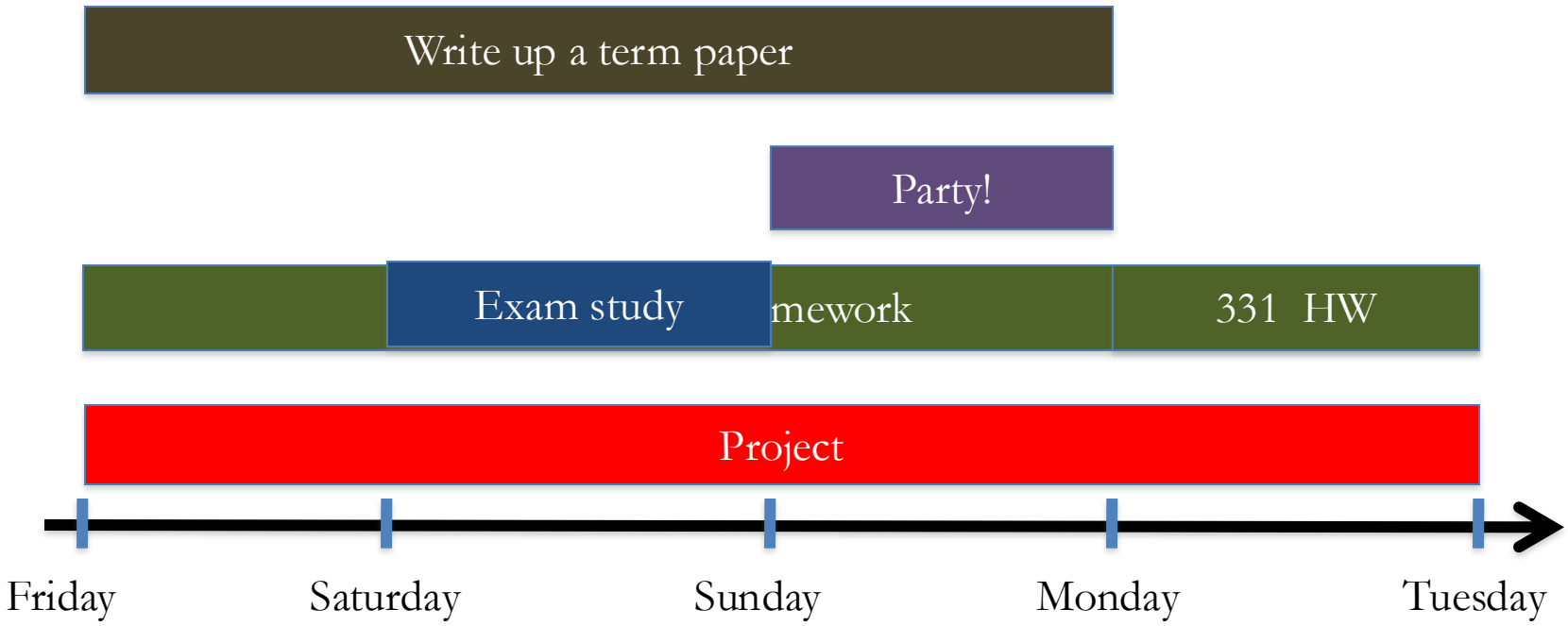
> ### Email template for withdrawing submission
> Dear Dr. Rudra/Atri,
>
> I wish to inform you that I have violated CSE 331 policies on my submission for Question X on Homeworks/Assignment N. I wish to withdraw my submission to preserve academic integrity.
>
> J.Q. Student
> Person #12345678
> UBIT: jqstuden
>
> Sincerely, J

On receiving the above email, I will assign J a 0 on Question X on Homeworks/Assignment N but disregard any Academic Integrity issues with the problematic submission. Note that J is not required to present any details on how they violated academic integrity.

Questions?

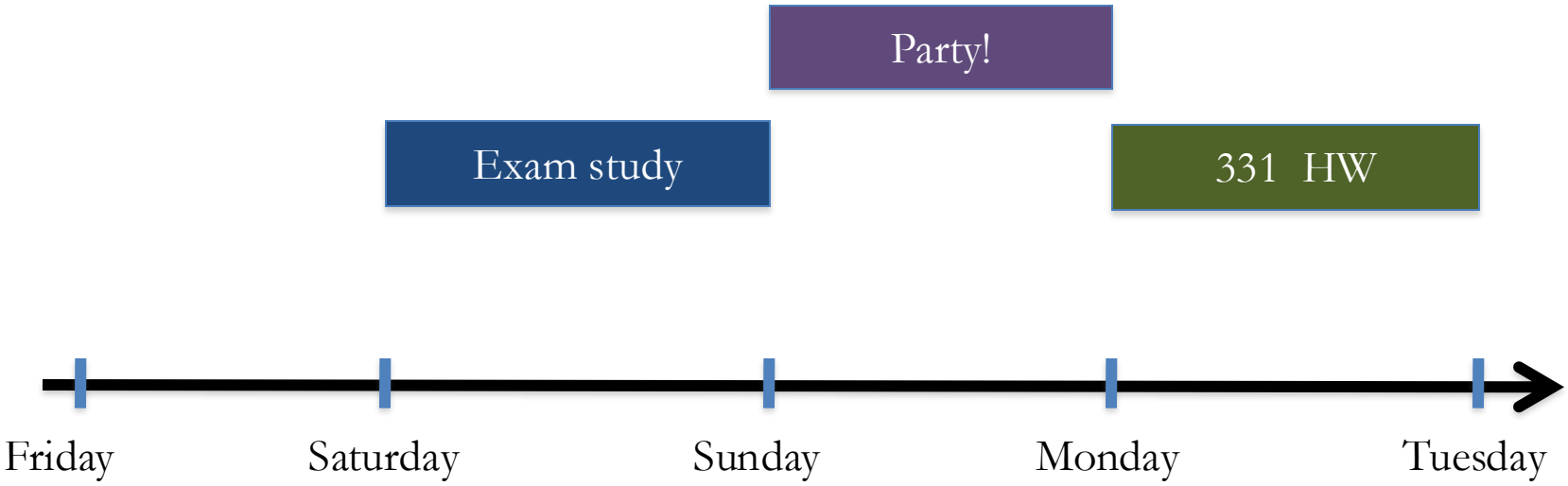
End of Semester blues

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

Questions?

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.

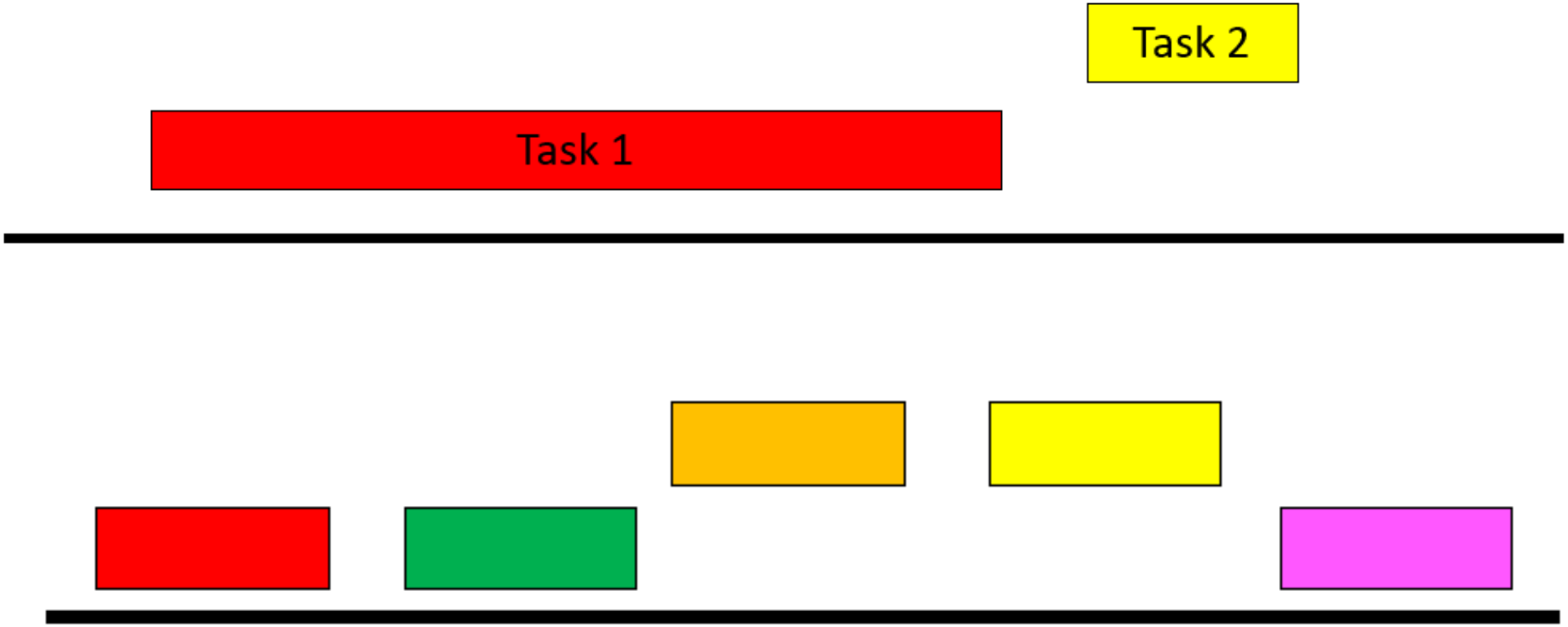
Sample Input and Output

Input:

Re-define problem on the board…

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

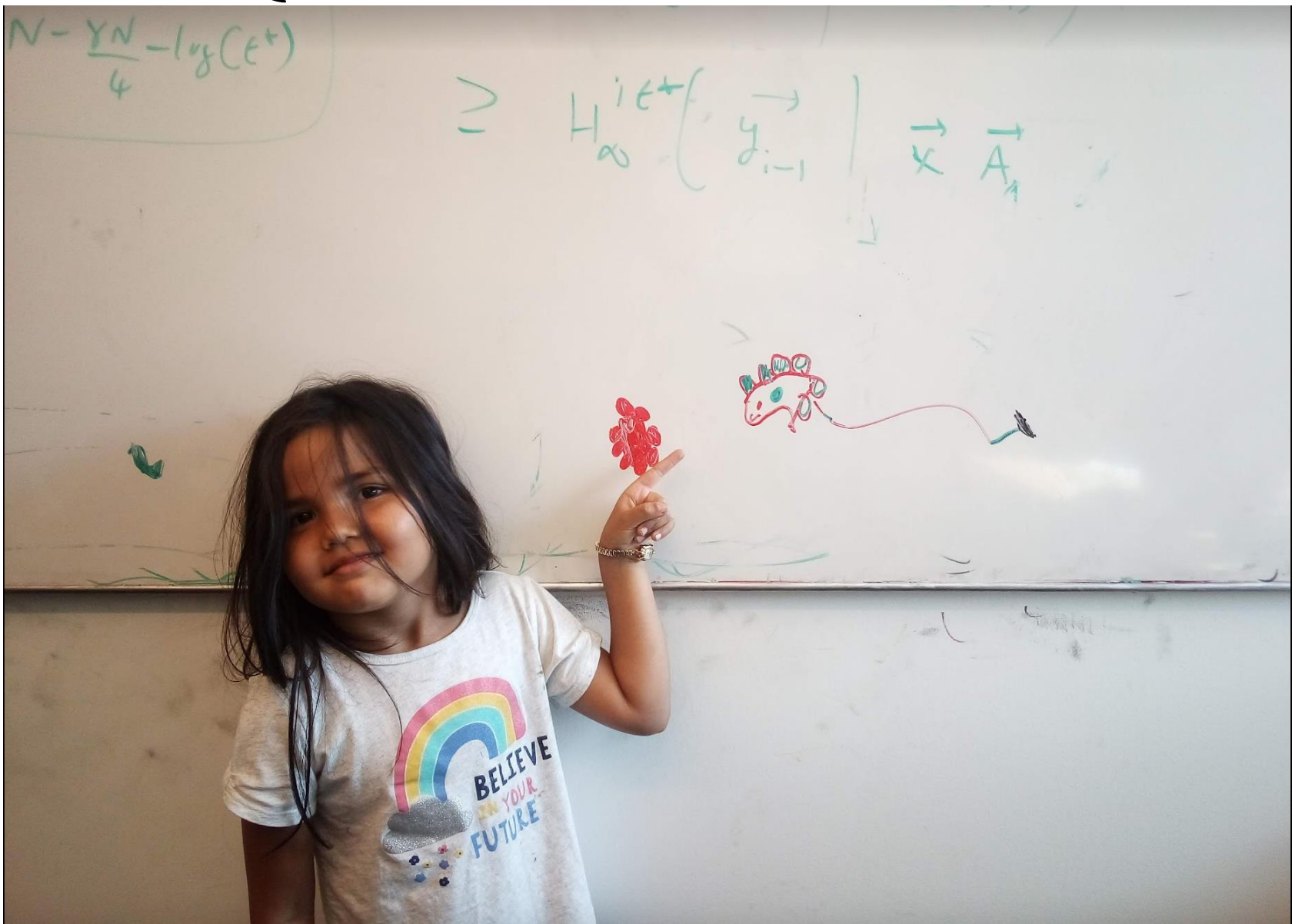
Choose i in R

Add i to S

Remove i from R

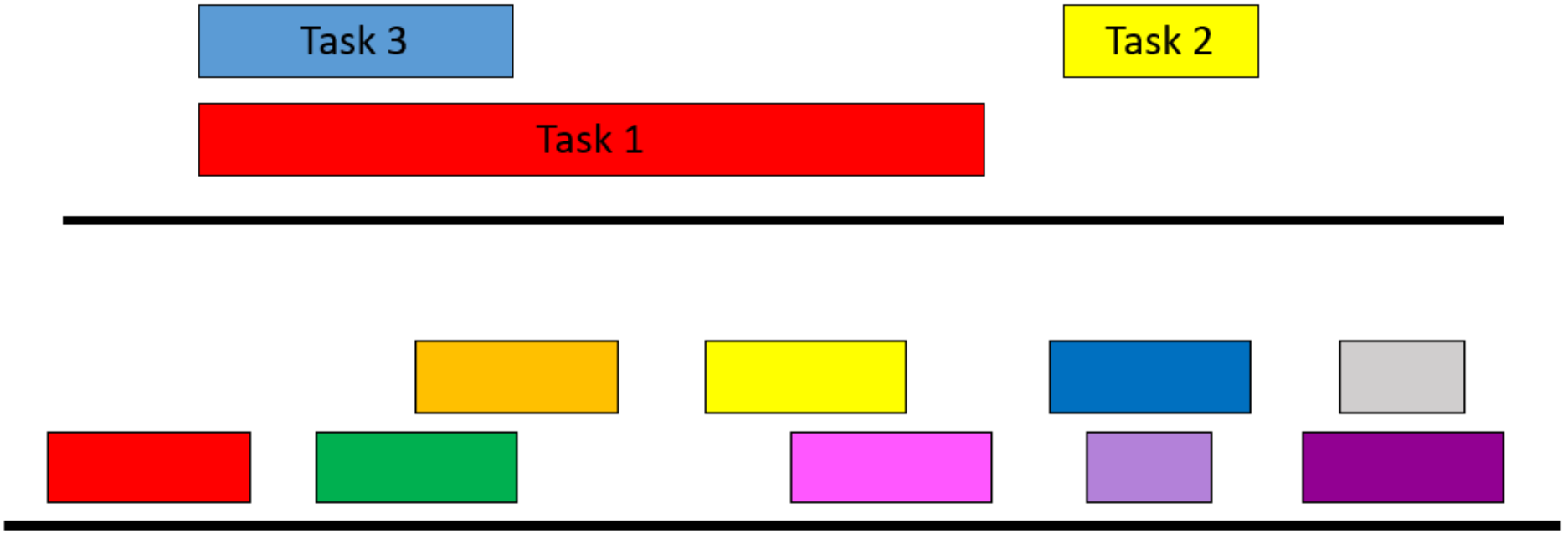
Return $S^* = S$

Questions/Comments?

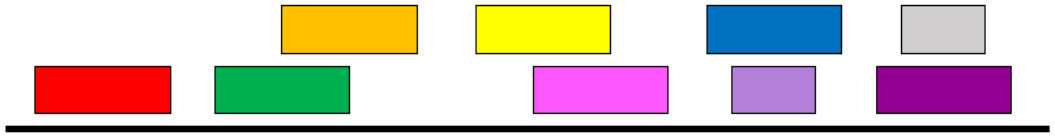


Example 2

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

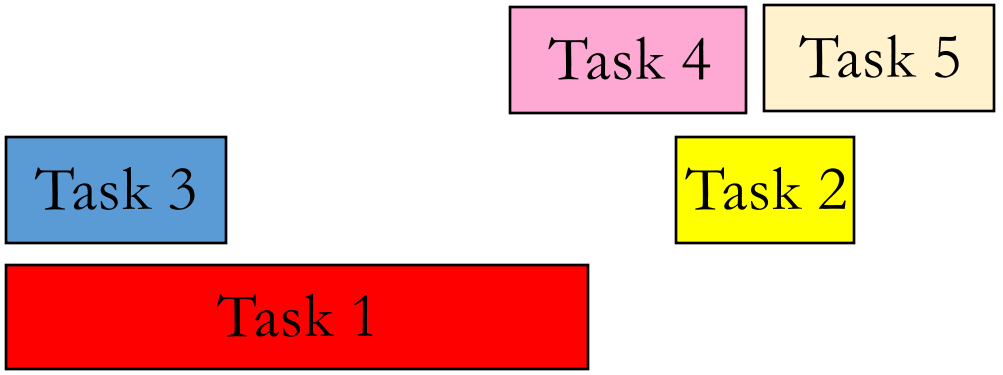
Add i to S

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

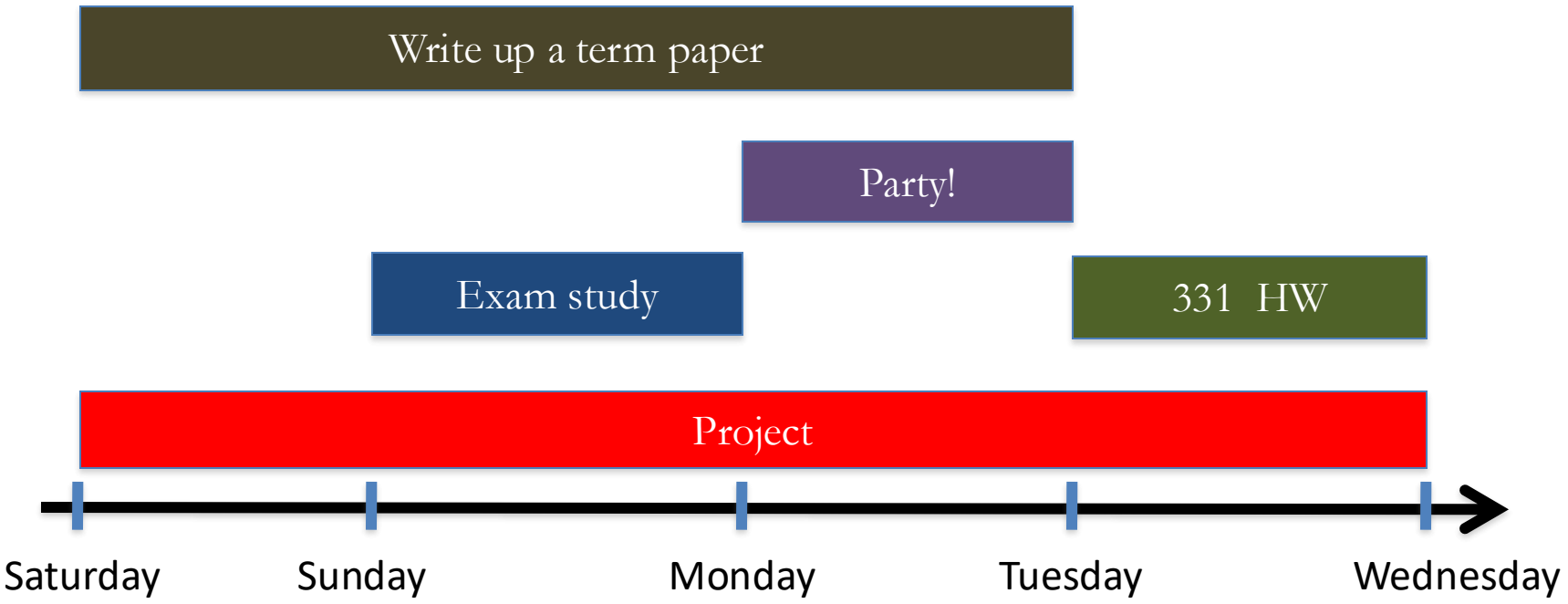
Add i to S

Remove all tasks that conflict with i from R

Return $S^* = S$

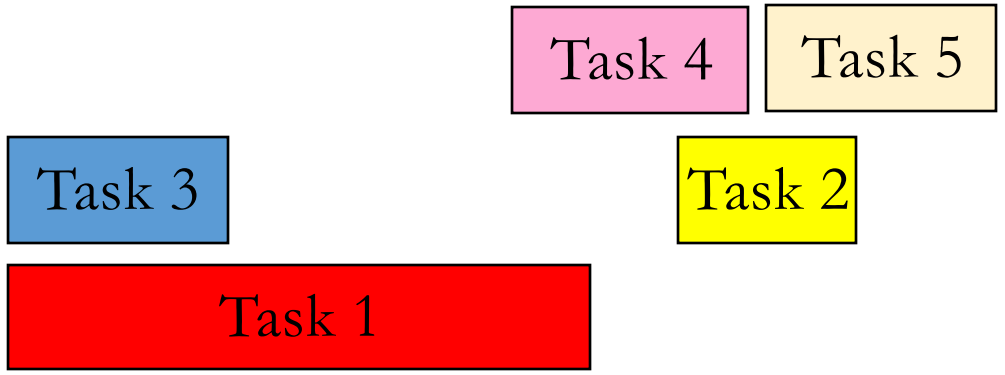
Greedily solve your blues!

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)

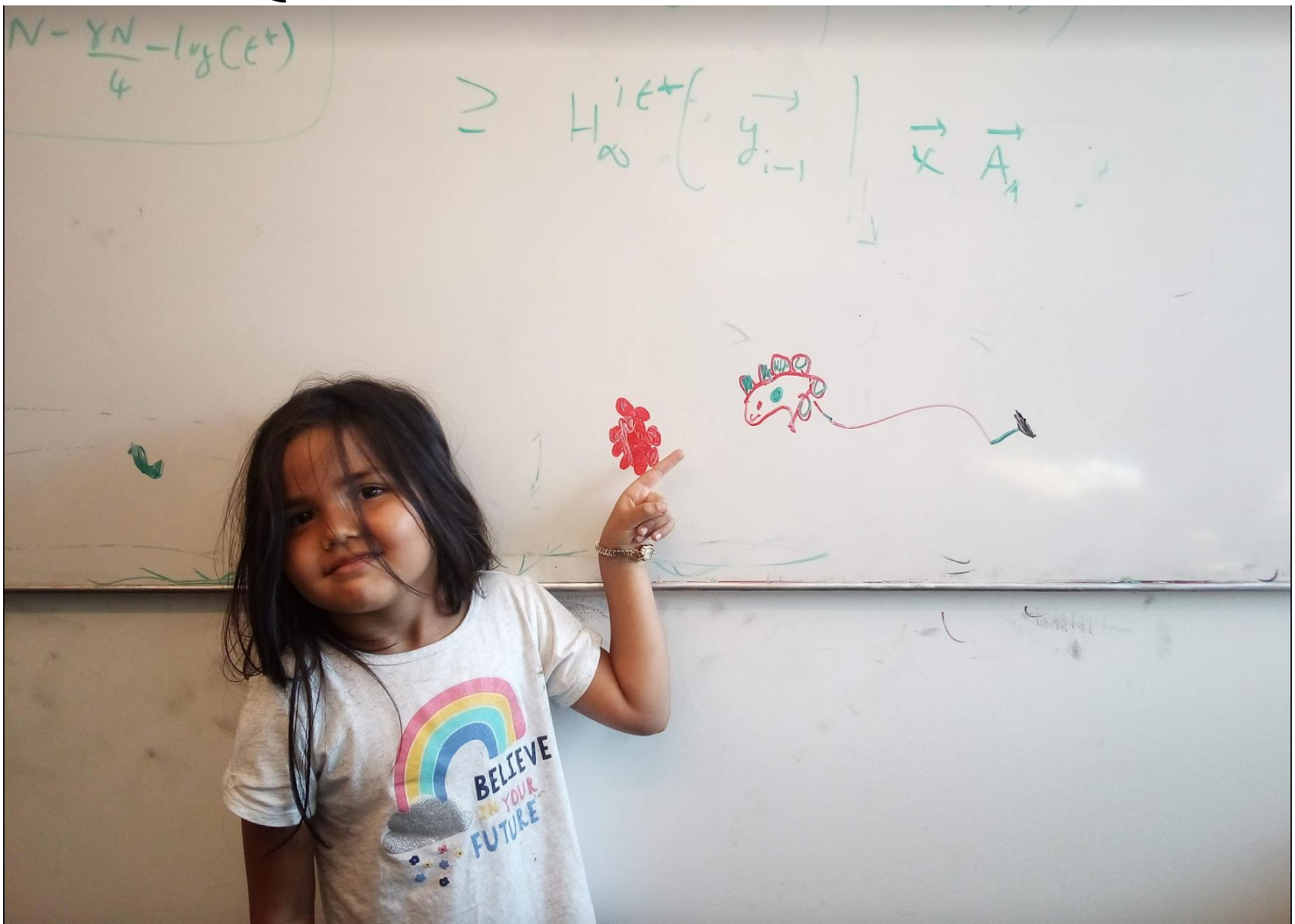
Add i to S

Remove all tasks that conflict with i from R

Return $S^* = S$

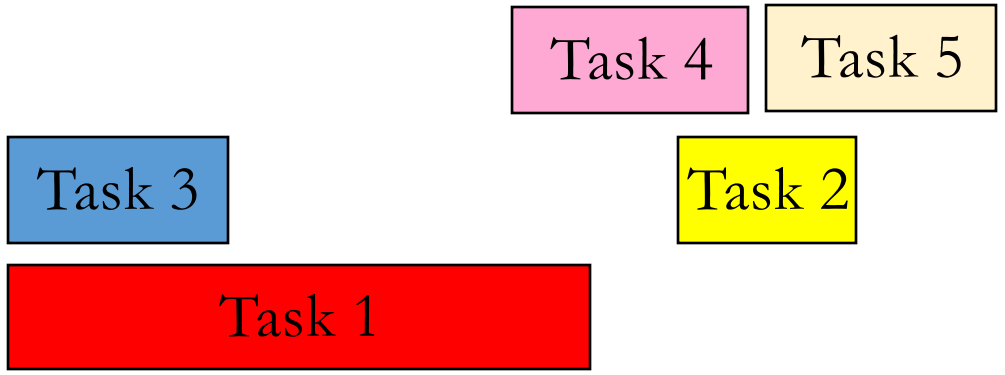
Associate a value v(i) with task i

Questions/Comments?



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

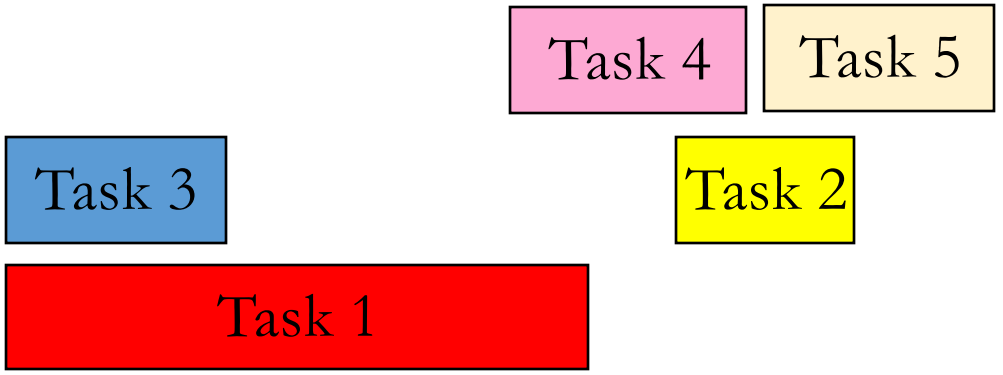
Remove all tasks that conflict with i from R

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)

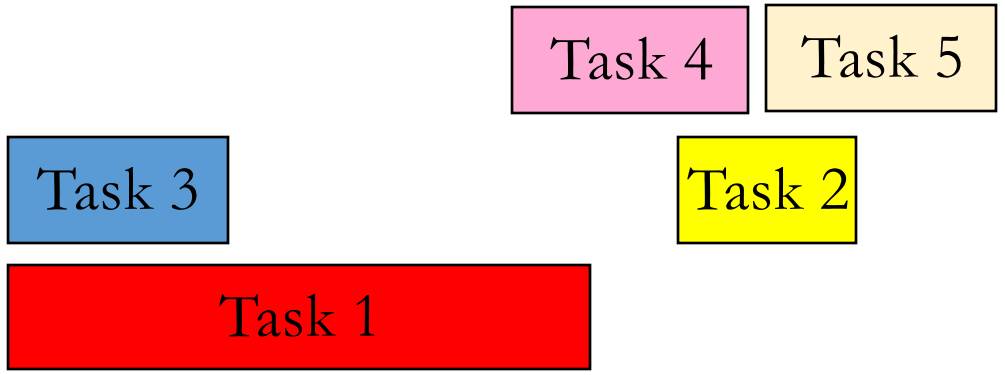
Add i to S

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

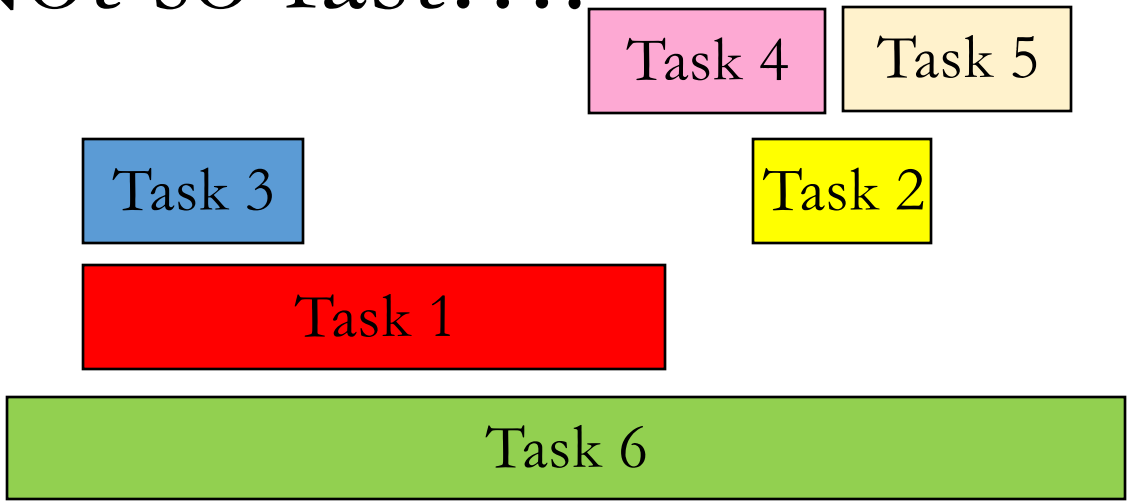
Remove all tasks that conflict with i from R

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

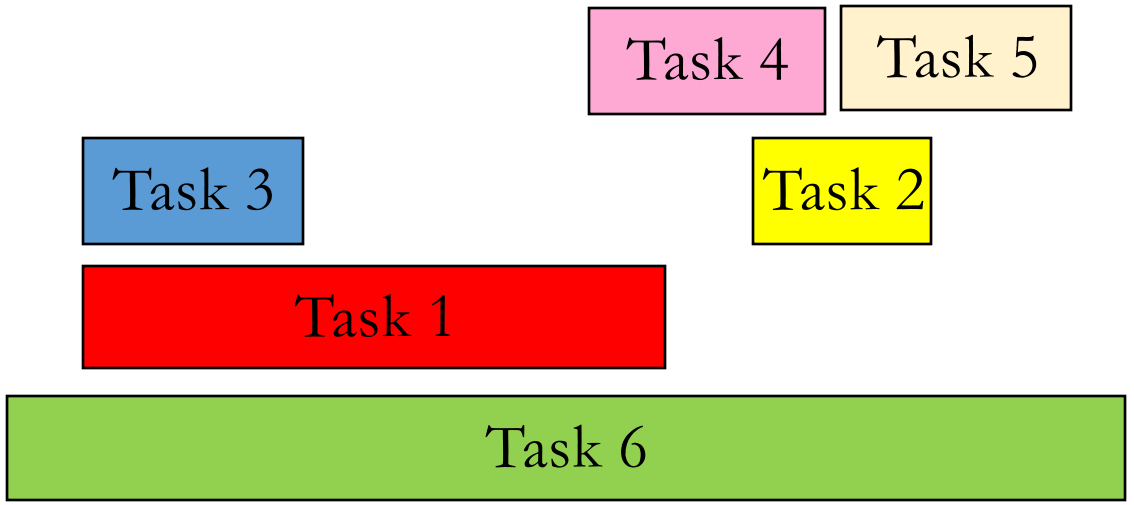
Choose i in R that minimizes $s(i)$

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

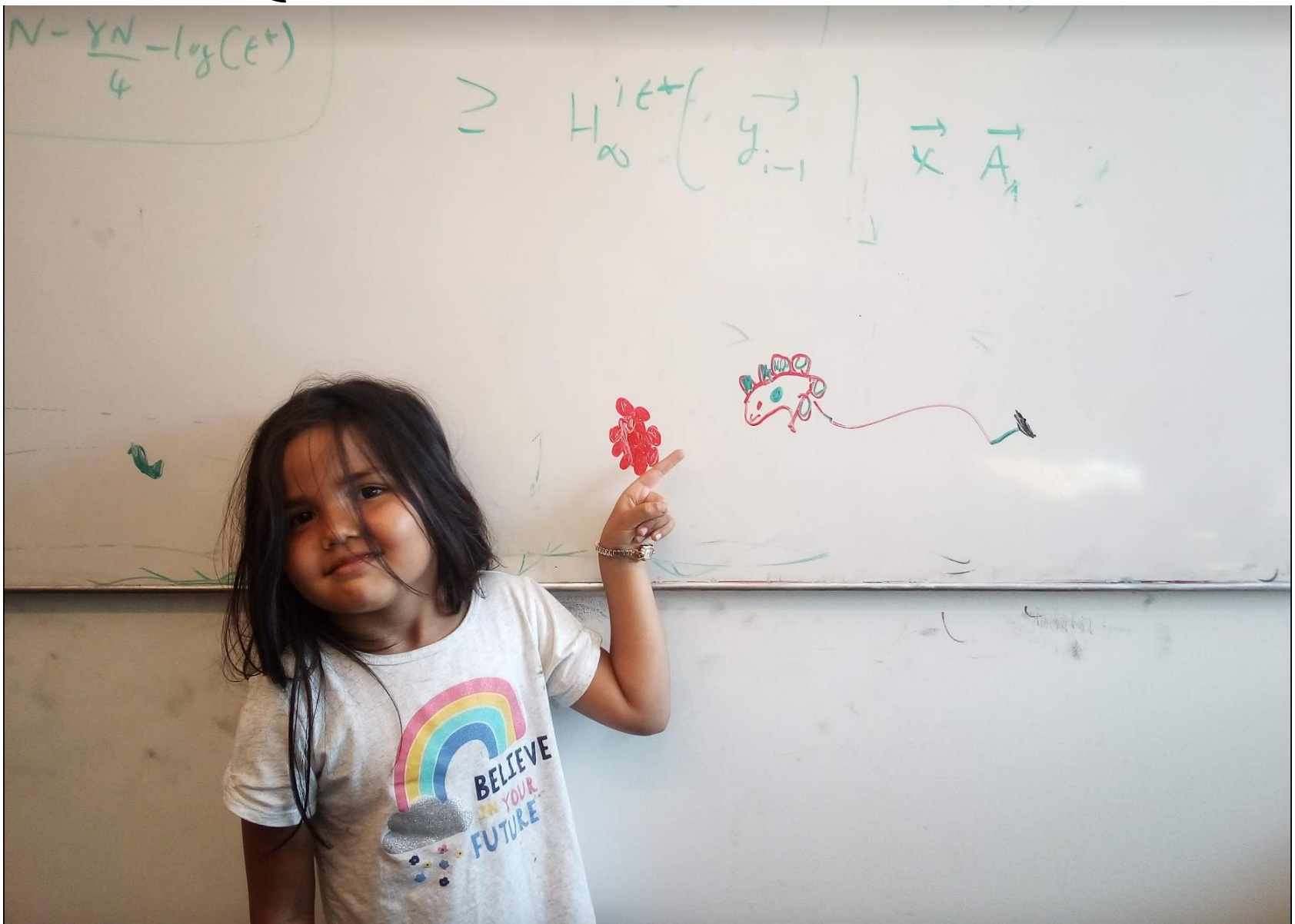
Add i to S

Remove all tasks that conflict with i from R

Return $S^* = S$

So are we done?

Questions/Comments?



Nope (but harder to show)

Set S to be the empty set

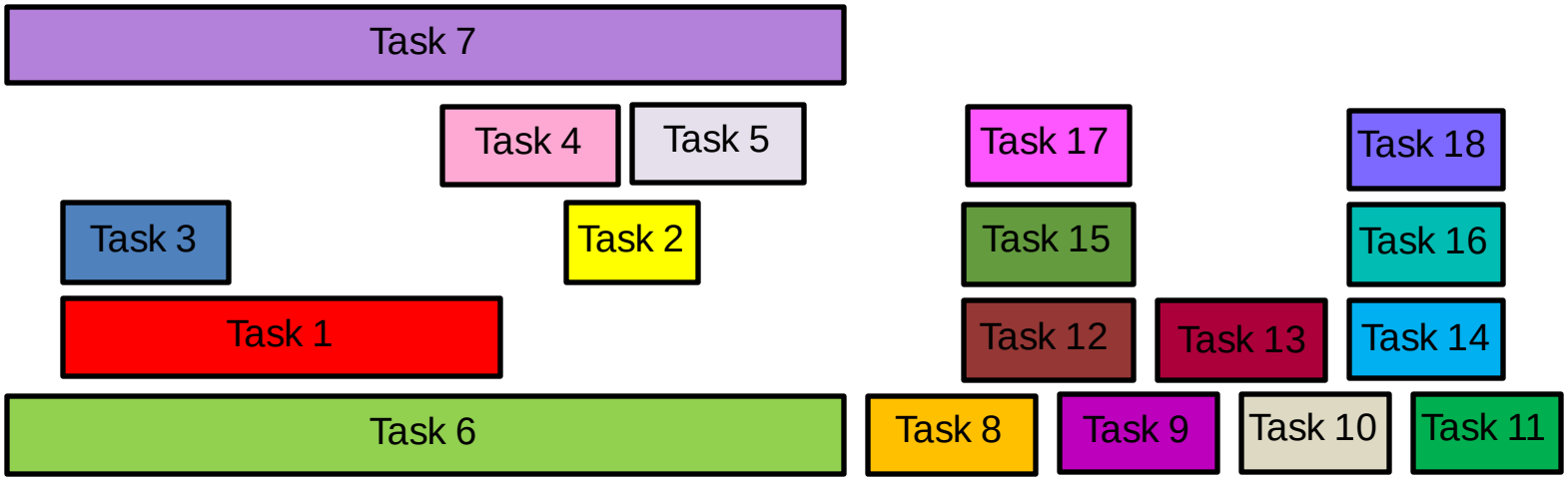
While R is not empty

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

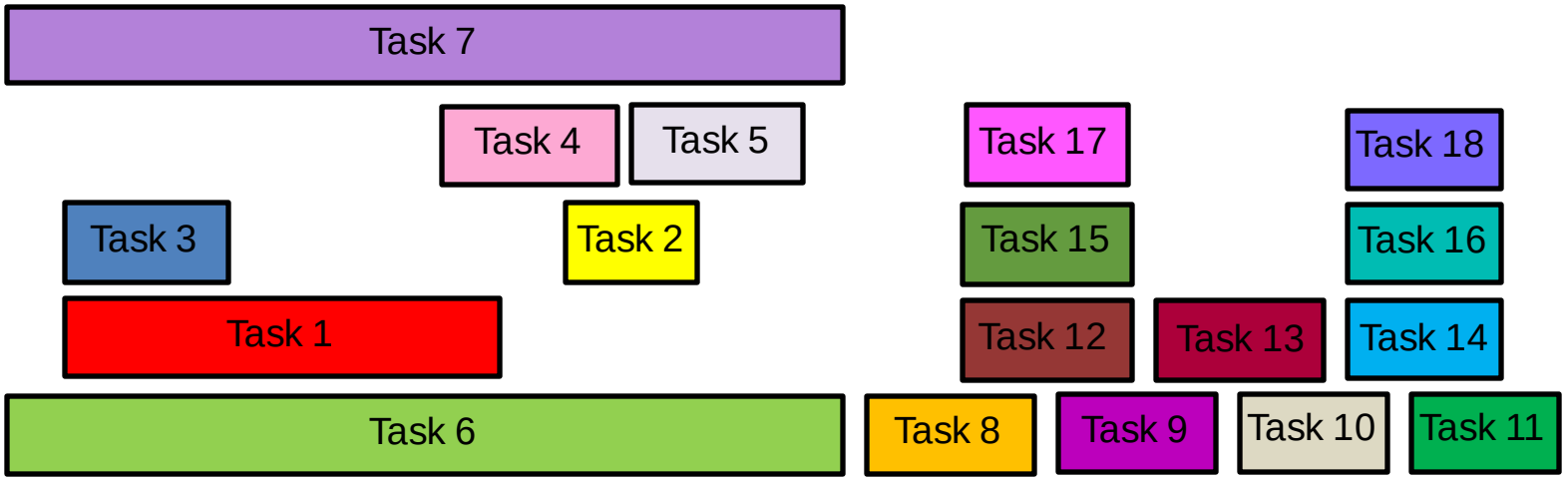
Choose i in R that has smallest number of conflicts

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

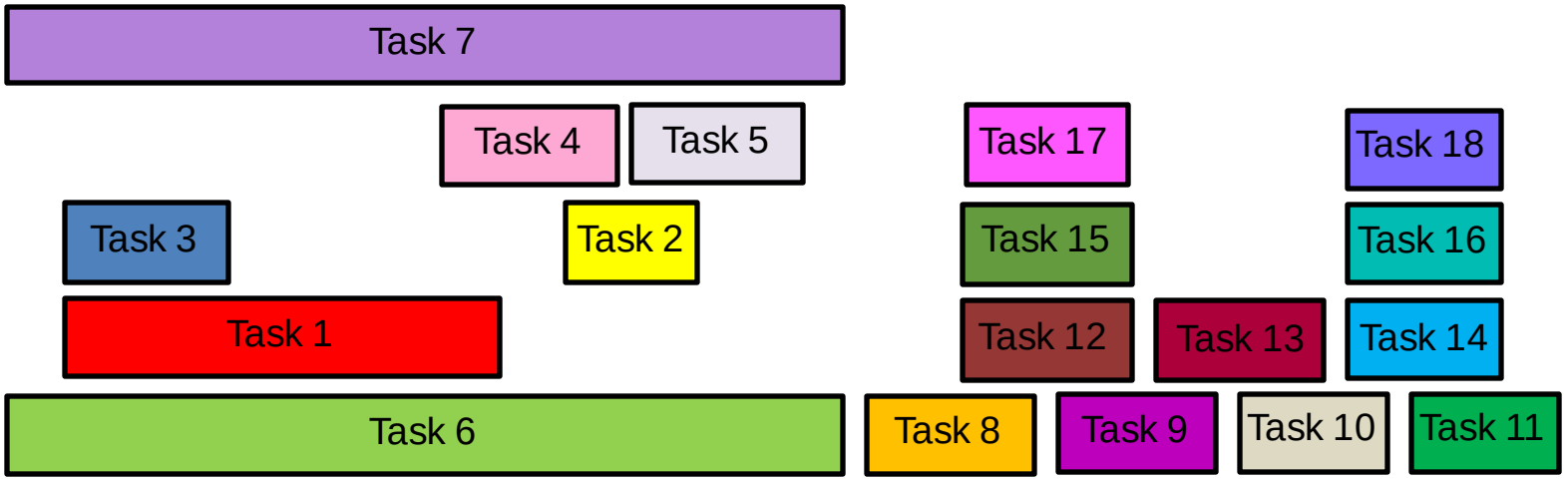
Choose i in R that minimizes v(i)

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

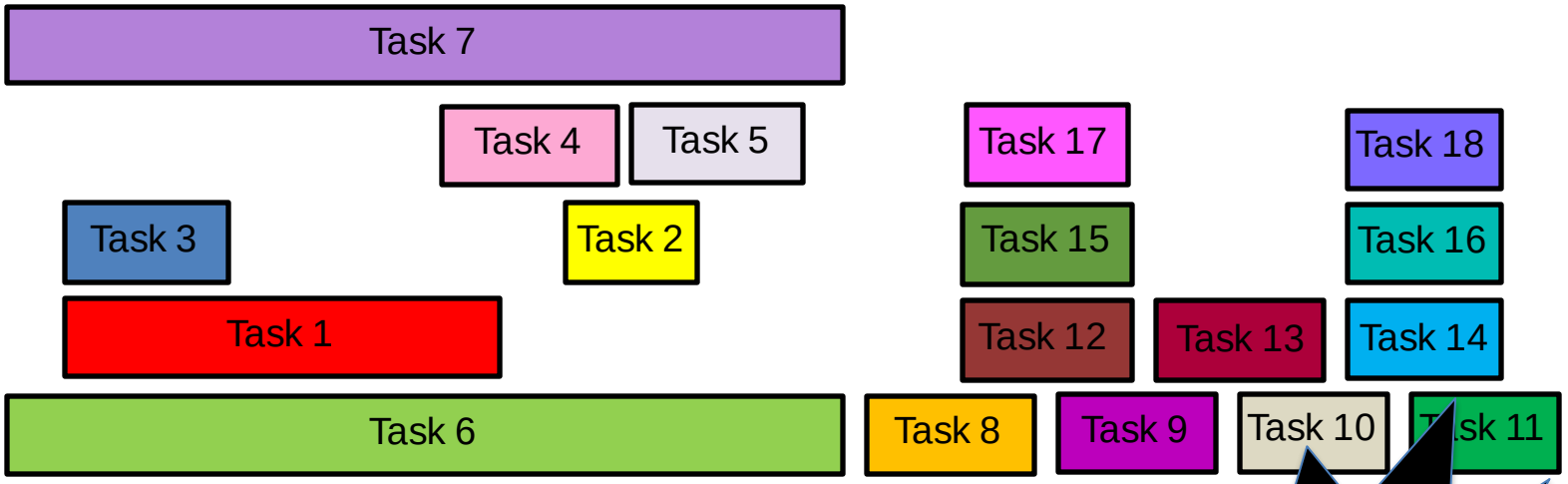
Choose i in R that minimizes f(i)

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)

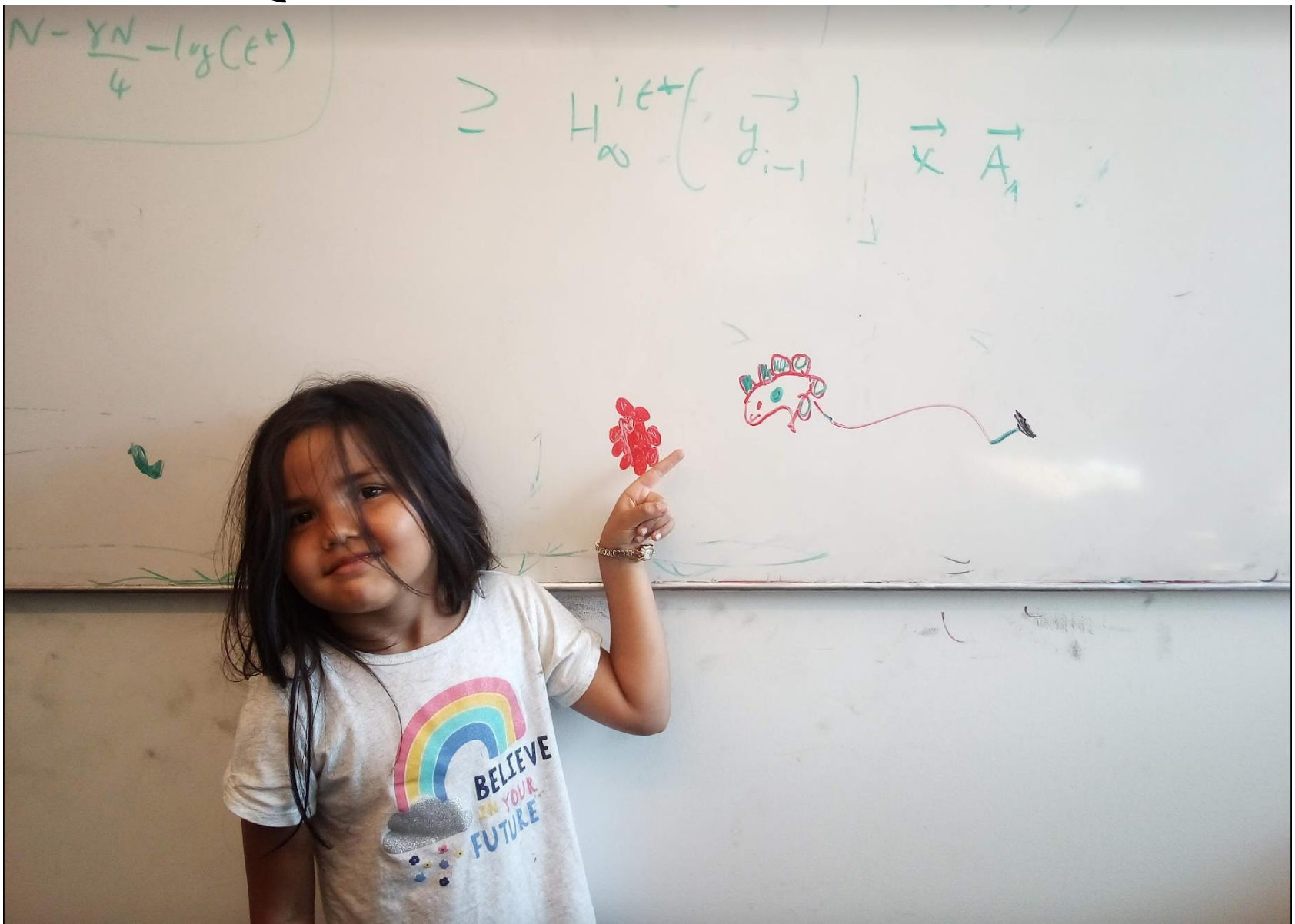
Add i to S

Remove all tasks that conflict with i from R

Return $S^* = S$

It works!

Questions/Comments?



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 $S^* = S$

Argue correctness on the board…