Lecture 16

CSE 331

Quiz 1 this FRIDAY

note @342 stop following **105 views**

Actions

Quiz 1 on Friday, March 10

The first **in-class** quiz will be on **Friday, March 10**. We will have a 5 mins break after the quiz and the lecture will start after the break.

We will hand out the quiz papers 5 mins before the start of the class, but you will **NOT** be allowed to open the quiz to see the actual questions. 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). Also quiz 1 will cover all topics that we discussed in class till Wednesday, March 1.

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.

Project groups finalized

note @359 stop following 67 views

Actio

Project Group Compositons Done and Emails Sent

Project group composition is now done for those who submitted the group sign-up form. I've sent emails to all groups. Please contact me if

- you have received two emails
  - you were mistakenly included in two groups.
- your group information is incorrect.
- you filled in the form but didn't receive an email.

Everyone who submitted the group sign-up form should have received an email by now. Please contact me by 5pm tomorrow if you notice any error in your group confirmation.

Interval Scheduling Problem

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

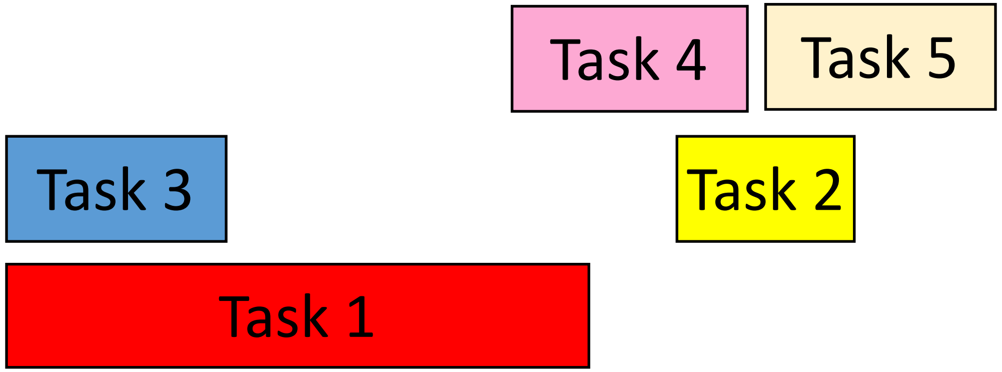
Output: A *schedule* S of the n intervals

No two intervals in S conflict

$|S|$ is maximized

Example 3

More than one conflict



Set S to be the empty set

While R is not empty

Choose i in R

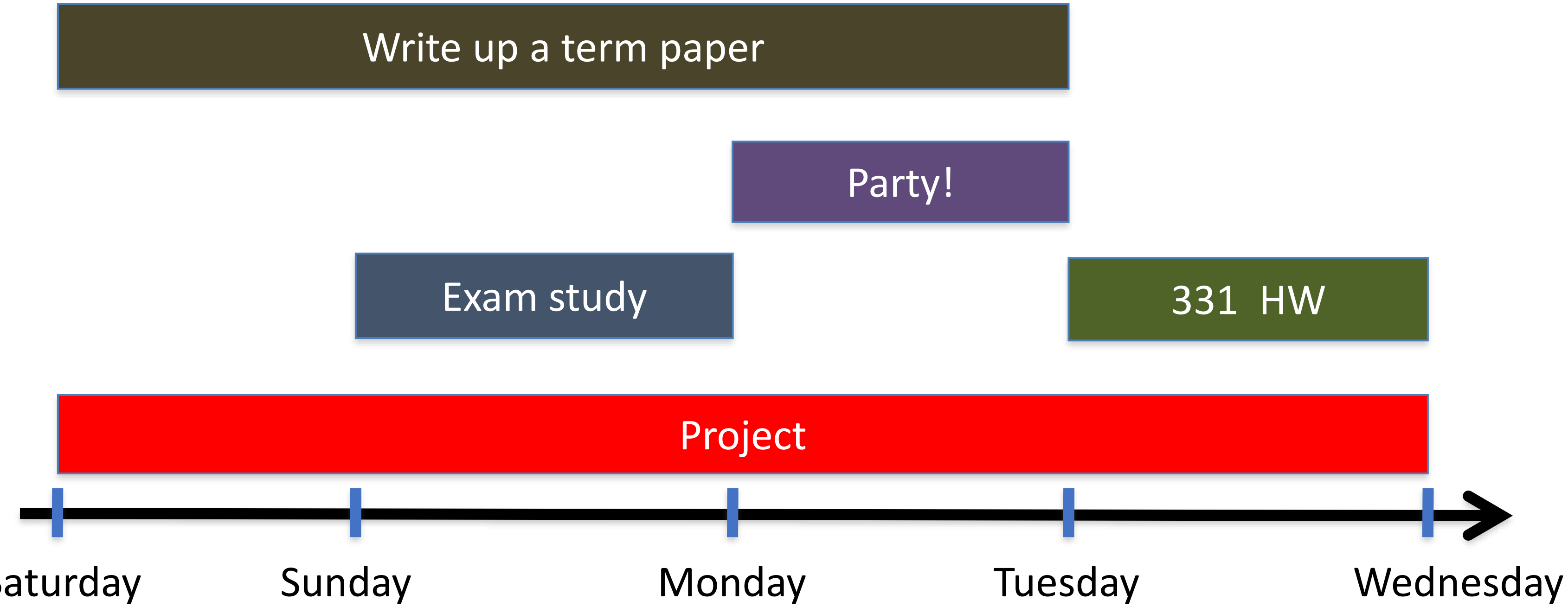
Add i to S

Remove all tasks that conflict with i from R

Return $S^* = S$

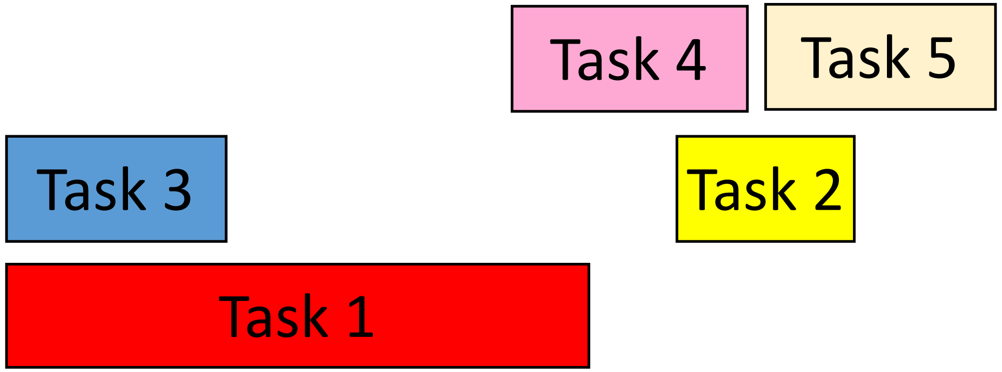
Greedily solve your blues!

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

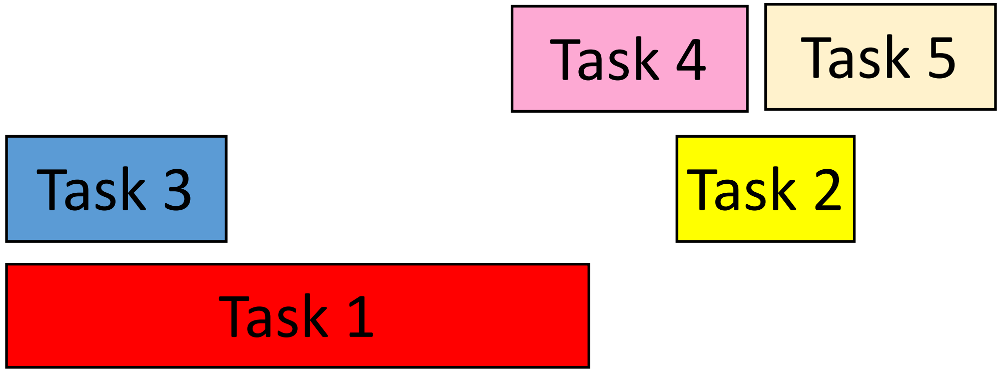
    Remove all tasks that conflict with i from R

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

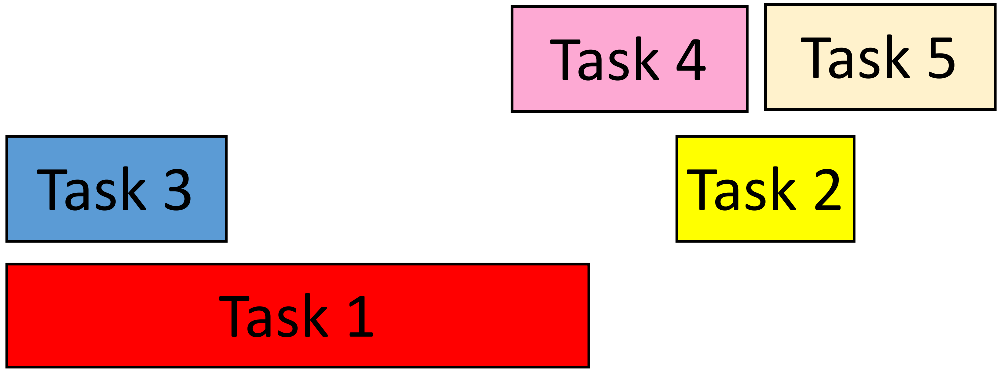
  Remove all tasks that conflict with i from R

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 $f(i) - s(i)$

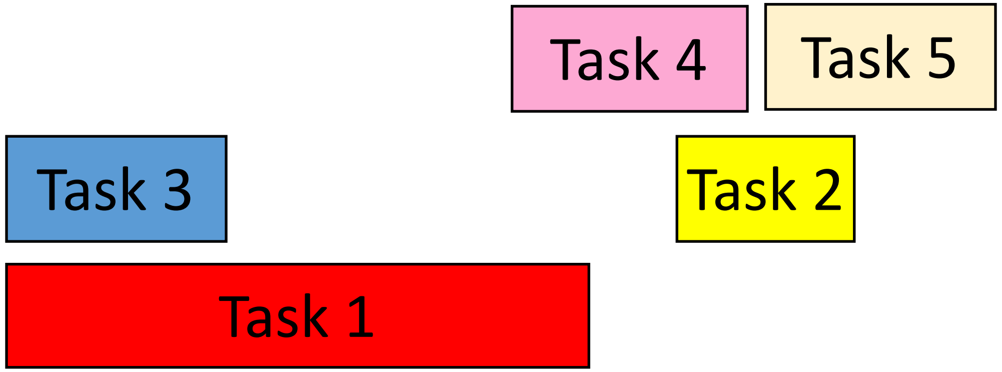
Add i to S

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

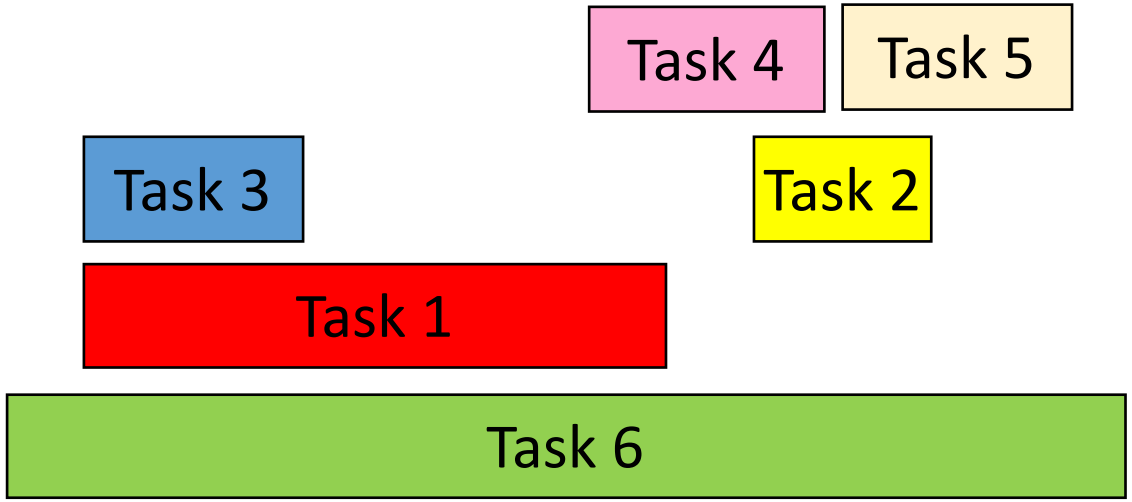
Remove all tasks that conflict with i from R

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

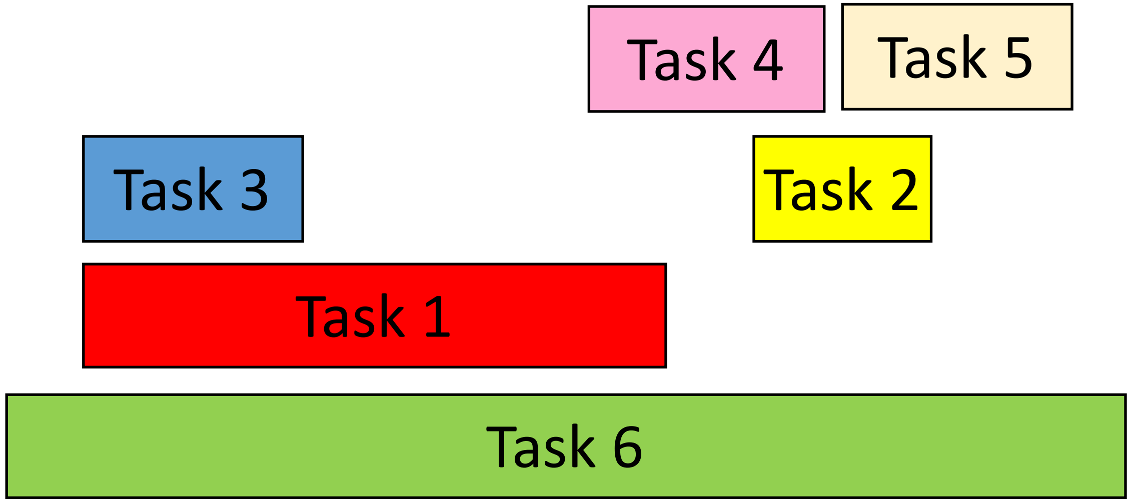
Choose i in R that minimizes s(i)

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

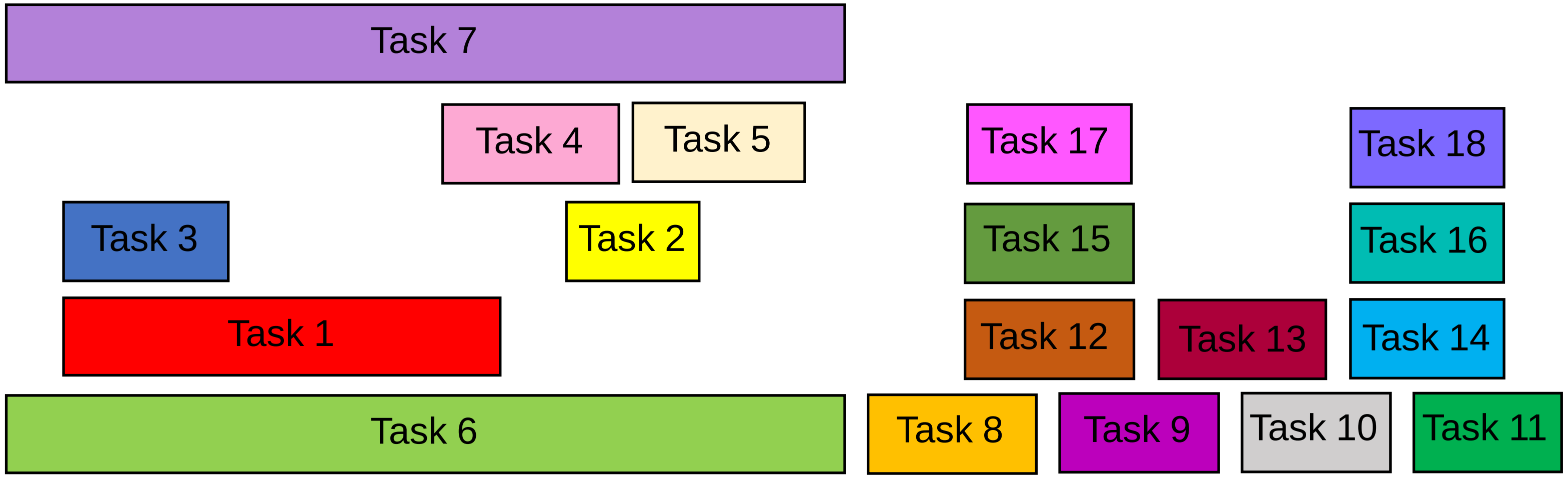
While R is not empty

  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

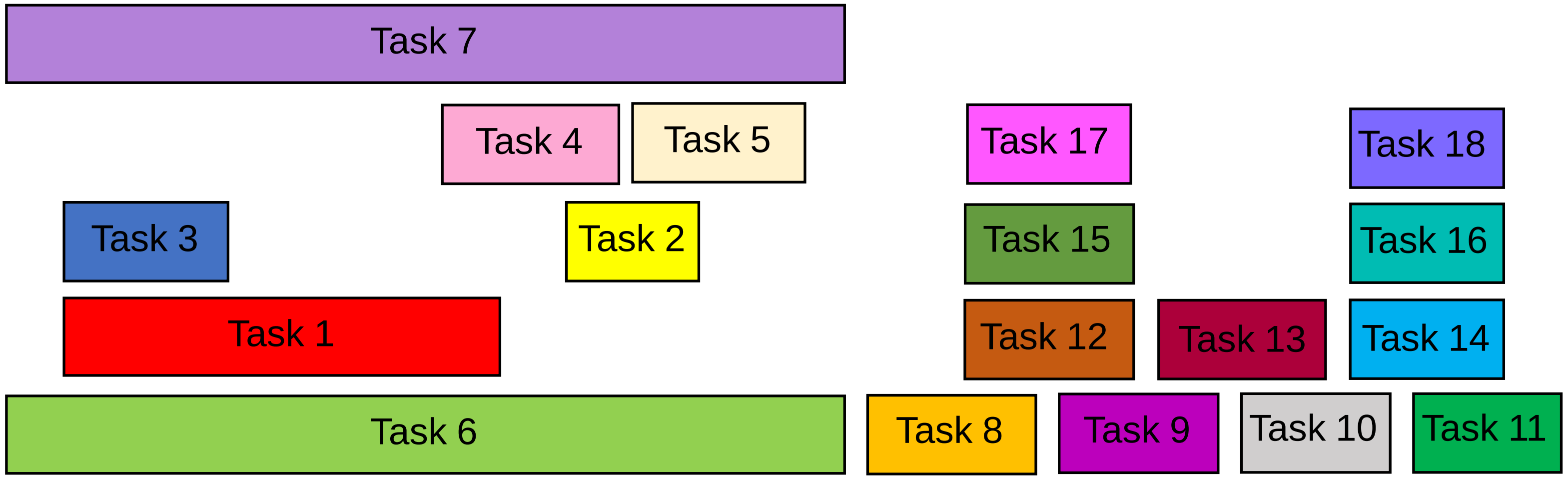
Choose i in R that has smallest number of conflicts

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

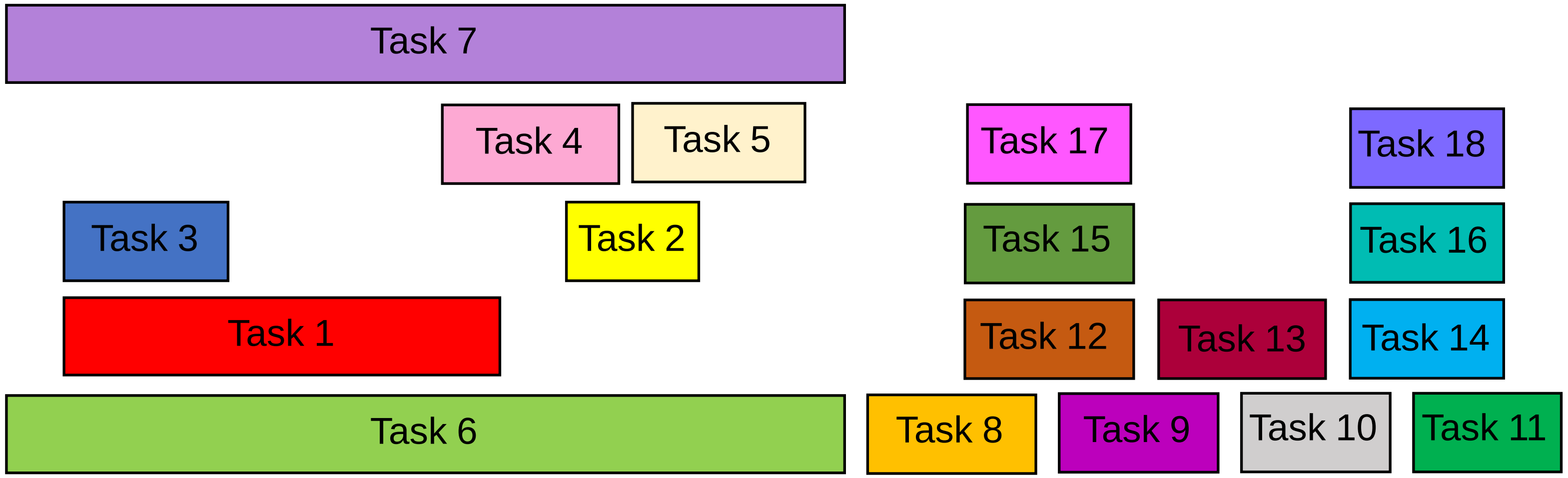
Choose i in R that minimizes v(i)

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

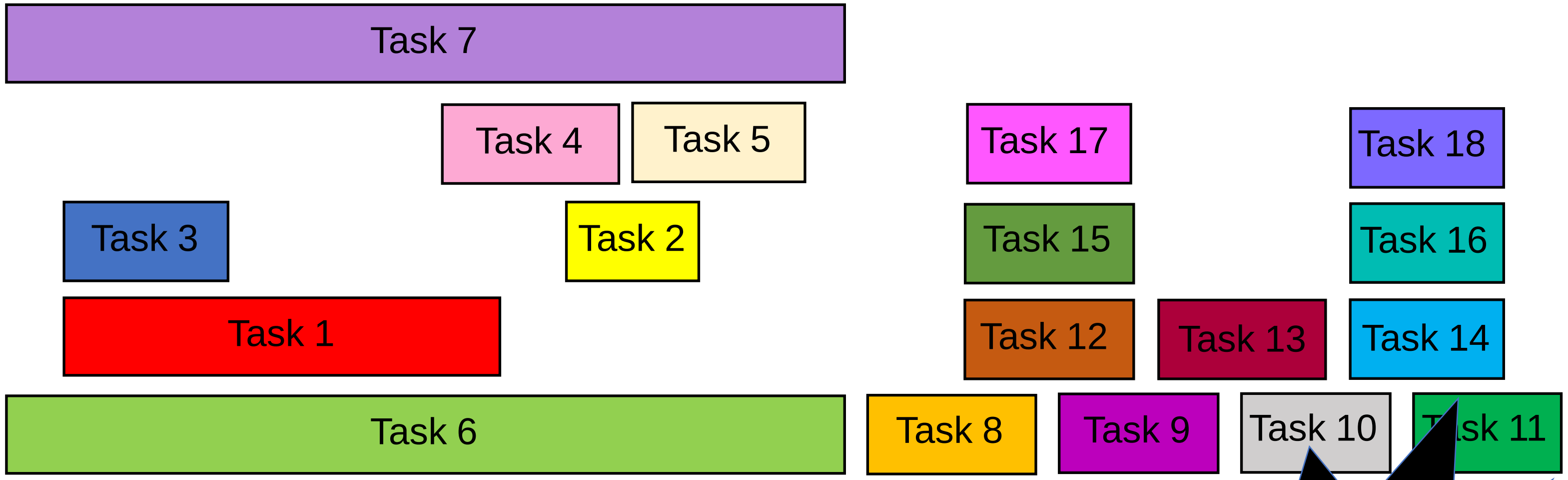
    Choose i in R that minimizes f(i)

    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!

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

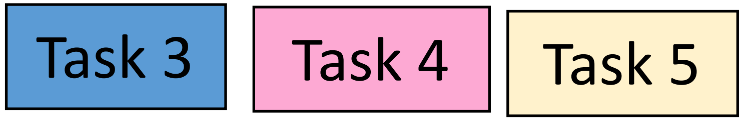
    Add i to S

    Remove all requests that conflict with i from R

Return $S^* = S$

Argue correctness

Observation: A valid schedule sorted by start/finish time gives the same order.



Assume that input intervals are sorted (in increasing order) by finish time
==> $f(1) \leq f(2) \leq f(3) \leq \ldots \leq f(n)$.

(If the input is not sorted, sort it in O(nlogn) time.)

Greedy Algorithm

0. $R \leftarrow [n]$
1. $S \leftarrow \Phi$
2. while $R \neq \Phi$
   (2.1) let i be the smallest index in R
   (2.2) add i to S
   (2.3) remove i from R
   (2.4) delete all $j \in R$ that conflict with i
3. Return $S^* \leftarrow S$

Greedy Algorithm

```
0.  R ← [n]
1.      S ← Φ
2.      while R ≠ Φ
        (2.1) let i be the smallest index in R
        (2.2) add i to S
        (2.3) remove i from R
        (2.4) delete all j ∈ R that conflict with i
3.      Return S* ← S
```

Theorem: S* is an optimal solution.

(that is, ∀ inputs, among all possible valid schedules for the input, S* has the maximum number of intervals.)

Ex 1: Algorithm terminates.
Ex. 2: S* is a valid schedule.

Proof of correctness of Greedy algorithm:

1. Greedy stays ahead
2. Exchange argument (minimize max. lateness, sec. 4.2 of KT)

Greedy "stays ahead"

Argue correctness on the board…