Lecture 16

CSE 331

Quiz 1 this FRIDAY

note @261 stop following 128 views Actions

Quiz 1 on Friday, March 11

The first quiz will be from **3:00-3:15pm in class** (Cooke 121) on **Friday, March 11**. We will have a 5 mins break after the quiz and the lecture will start at 3:20pm.

We will hand out the quiz paper at 2:55pm but you will **NOT** be allowed to open the quiz to see the actual questions till 3:00pm. 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: @244.) Also quiz 1 will cover all topics that we cover in class till Wednesday, March 2.

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 @311

Project Group Final Confirmations Sent

Everybody who filled in the project group sign-up form should have received an email from me by now. Please contact me if

- you received two emails:
  - member of a group
  - person with no group
- your group information is incorrect
- you filled in the form but didn't receive an email.

Any other concern, please contact me as soon as possible (preferably by tonight).

I'll send out group information emails tomorrow (hopefully) to those who don't have a group currently.

Interval Scheduling Problem

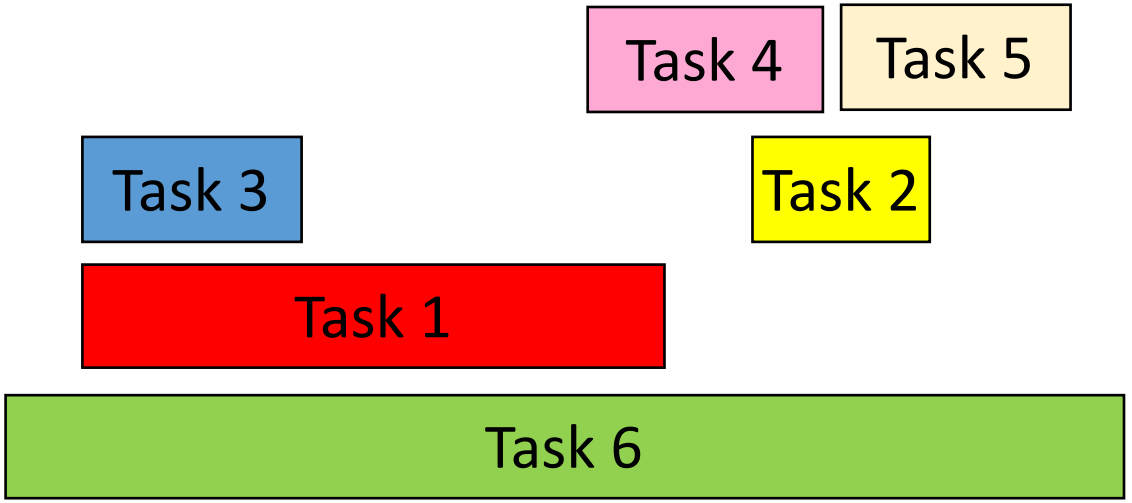
Input: n intervals $[s(i), f(i))$ for $1 \leq i \leq n$

Output: A *schedule* S of the n intervals

No two intervals in S conflict

|S| is maximized

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)

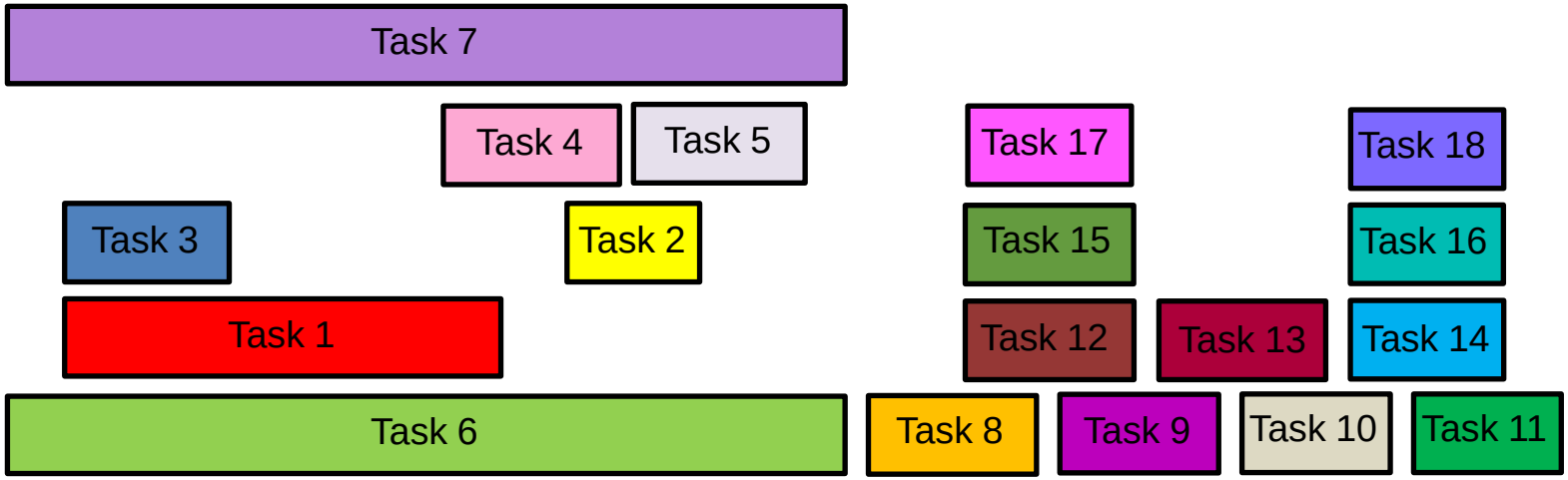
Set S to be the empty set

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

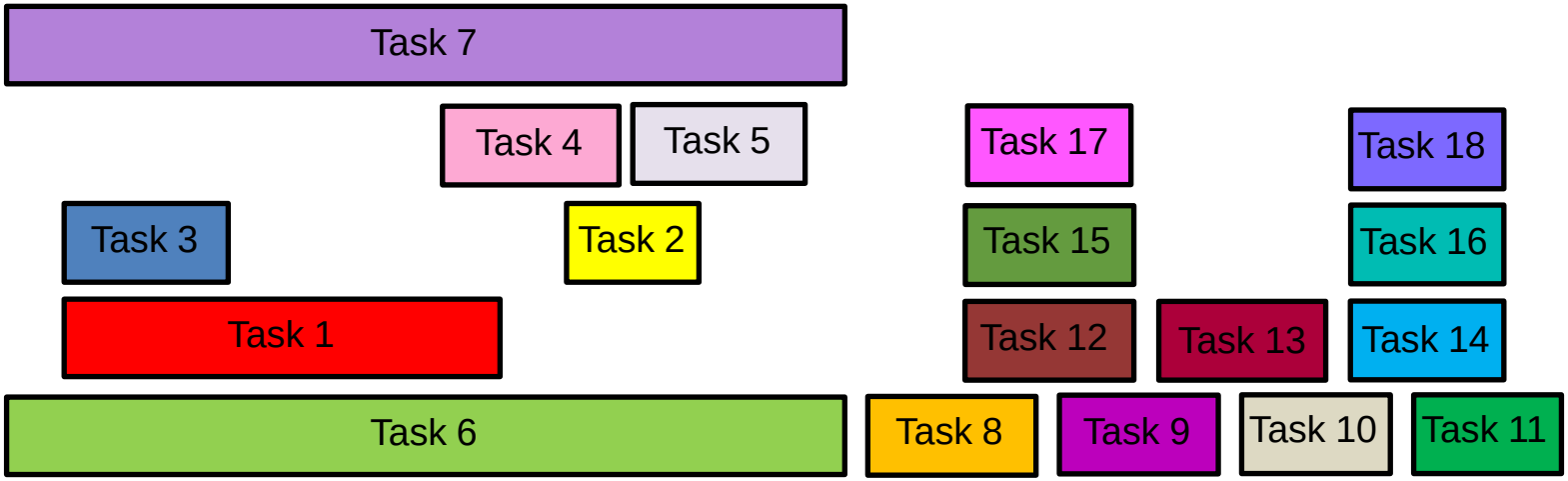
Choose i in R that has smallest number of conflicts

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

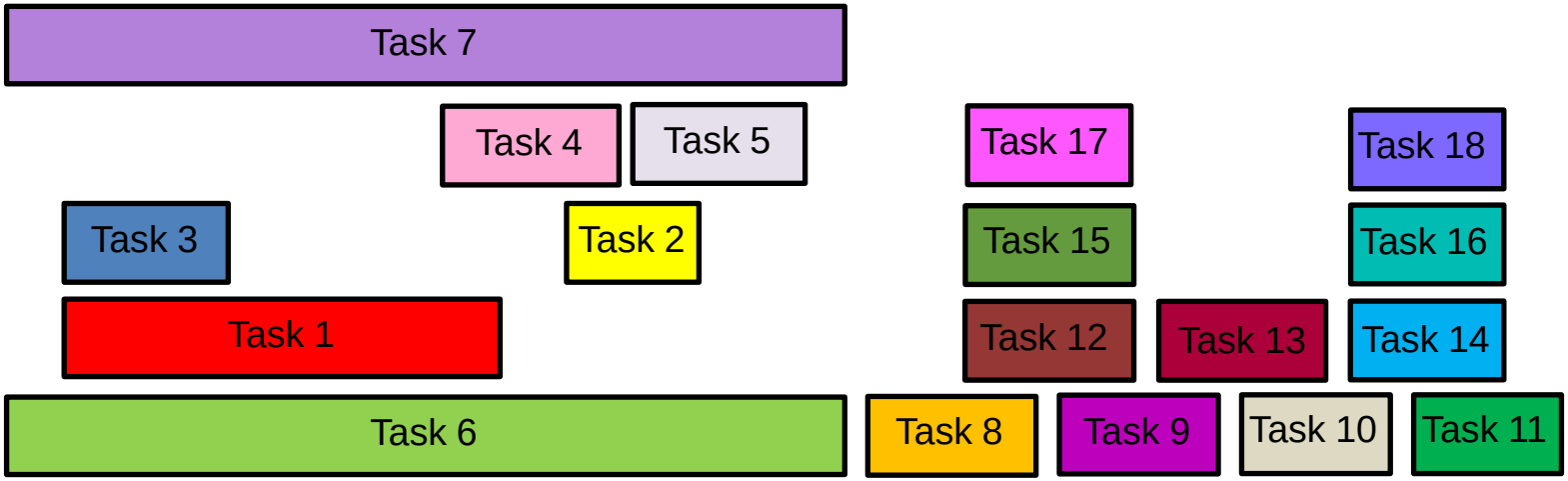
Choose i in R that minimizes v(i)

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

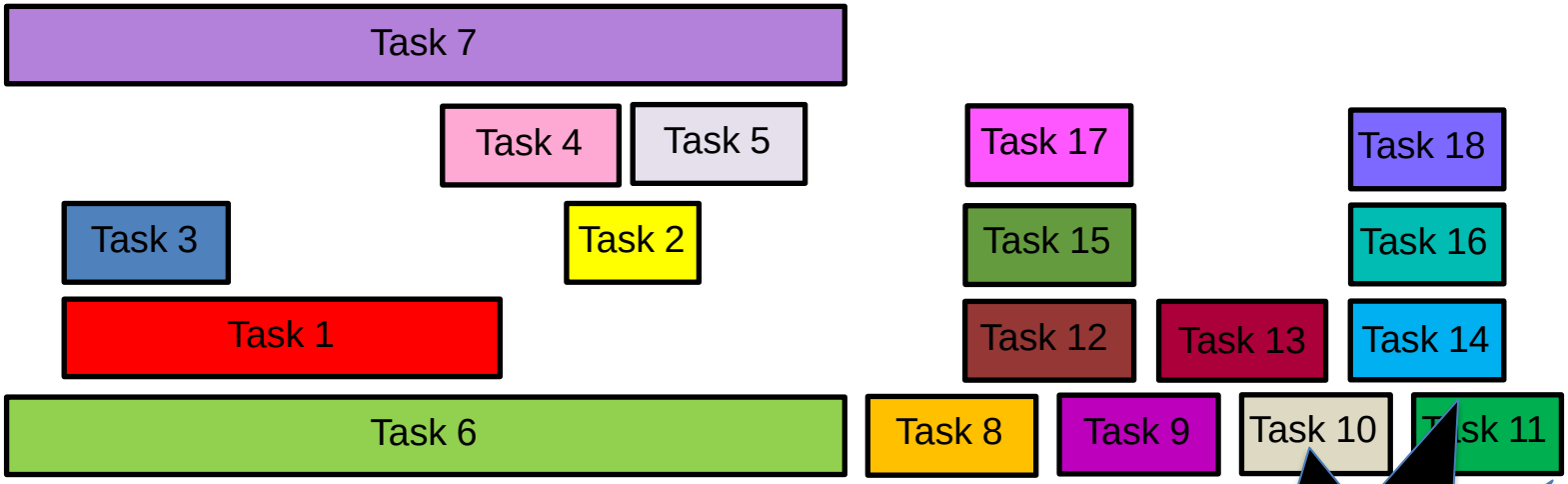
Choose i in R that minimizes f(i)

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!

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

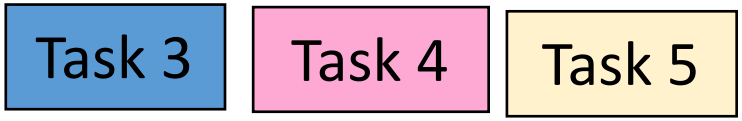
    Add i to S

    Remove all requests that conflict with i from R

Return $S^* = S$

Argue correctness

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"

Today's agenda

Prove the correctness

Analyze run-time of the greedy algorithm

Argue correctness on the board…