Lecture 28

CSE 331

Nov 8, 2021

Please have a face mask on

Masking requirement

UB requires all students, employees and visitors – regardless of their vaccination status – to wear face coverings while inside campus buildings.

https://www.buffalo.edu/coronavirus/health-and-safety/health-safety-guidelines.html

Reflection P2 due TODAY!

Fri, Nov 5	Kickass Property Lemma F19 F18 F17 x²	[KT, Sec 5.4] (**Project (Problem 2 `Coding`) in**)
Mon, Nov 8	Weighted Interval Scheduling F19 F17 x²	[KT, Sec 6.1] (**Project (Problem 2 `Reflection`) in**)
Wed, Nov 10	Recursive algorithm for weighted interval scheduling problem F19 F17 x²	[KT, Sec 6.1] (**HW 6 out**)
Fri, Nov 12	Subset sum problem F19 F18 F17 x²	[KT, Sec 6.1, 6.2, 6.4]
Mon, Nov 15	Dynamic program for subset sum F19 F18 F17 x²	[KT, Sec 6.4]
Wed, Nov 17	Shortest path problem F19 F18 F17 x²	[KT, Sec 6.8] (**HW 7 out, HW 6 in**)
Fri, Nov 19	Bellman-Ford algorithm F19 F18 F17 x²	[KT, Sec 6.8]
Mon, Nov 22	The P vs. NP problem F19	[KT, Sec 8.1]
Wed, Nov 24	**No class**	Fall Recess
Fri, Nov 26	**No class**	Fall Recess
Mon, Nov 29	More on reductions F19	[KT, Sec 8.1]
Wed, Dec 1	The SAT problem F19	[KT, Sec 8.2] (**HW 8 out, HW 7 in**)
Fri, Dec 3	NP-Completeness F19	[KT, Sec. 8.3, 8.4] (**Project (Problem 3 `Coding`) in**)
Mon, Dec 6	*k*-coloring problem F19	[KT, Sec 8.7] (**Quiz 2**) (**Project (Problem 3 `Reflection`) in**)

Group formation instructions

Autolab group submission for CSE 331 Project

The lowdown on submitting your project (especially the coding and reflection) problems as a group on Autolab.

Follow instructions **EXACTLY** as they are stated

The instruction below are for Coding Problem 1

You will have to repeat the instructions below for EACH coding AND reflection problem on project on Autolab (with the appropriate changes to the actual problem).

Form your group on Autolab

Groups on Autolab will NOT be automatically created

You will have to form a group on Autolab by yourself (as a group). Read on for instructions on how to go about this.

Click to add notes

Reflection P1 graded

note @456 stop following 3 views

Reflection Problem 1 graded

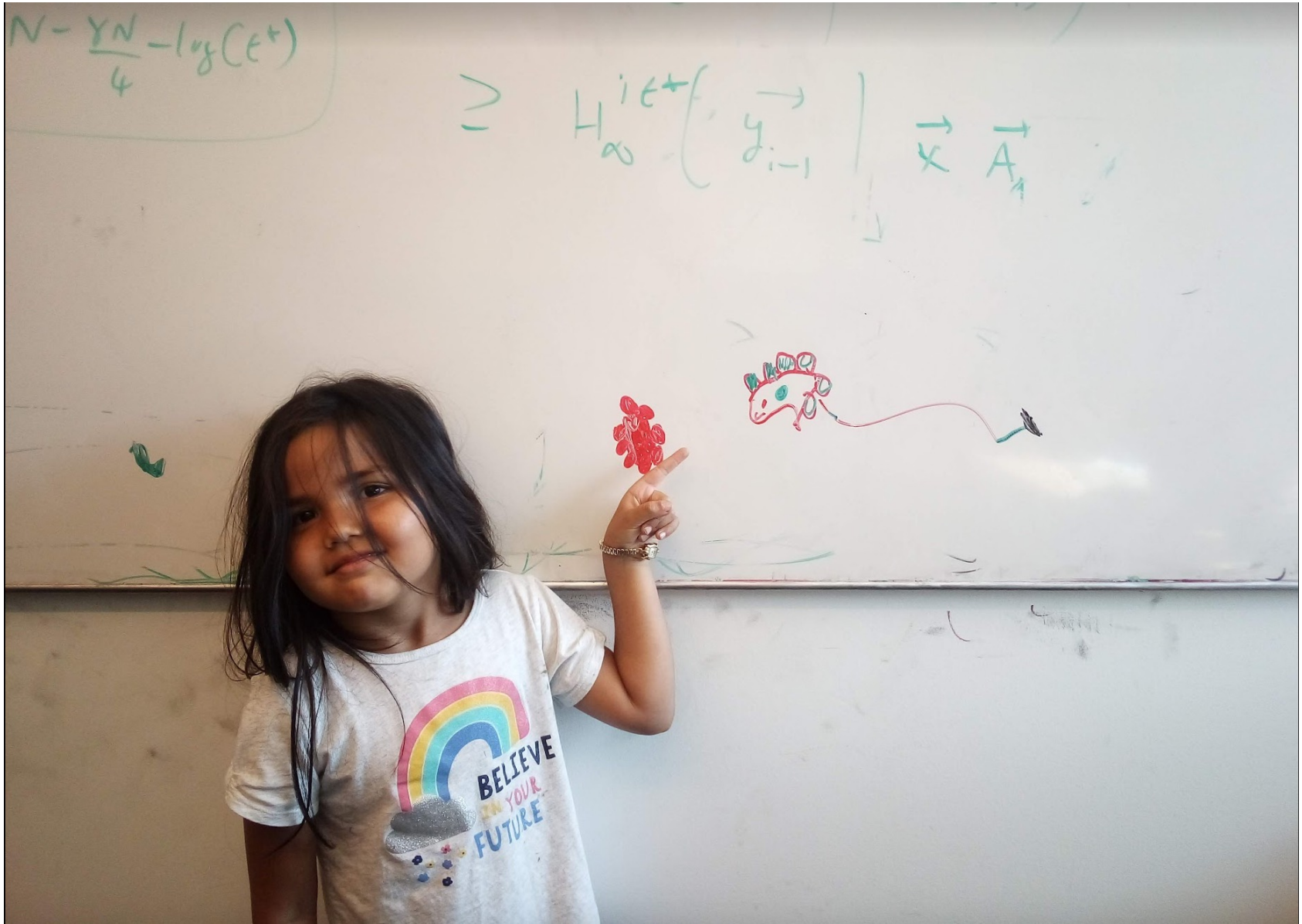
Reflection Problem 1 has now been graded and the scores and feedback released on Autolab! Hopefully the feedback is helpful as y'all are finishing off your Reflection Problem 2

(Please see the re-grade policy as well as the grading rubric below before contacting us with questions on grading.)

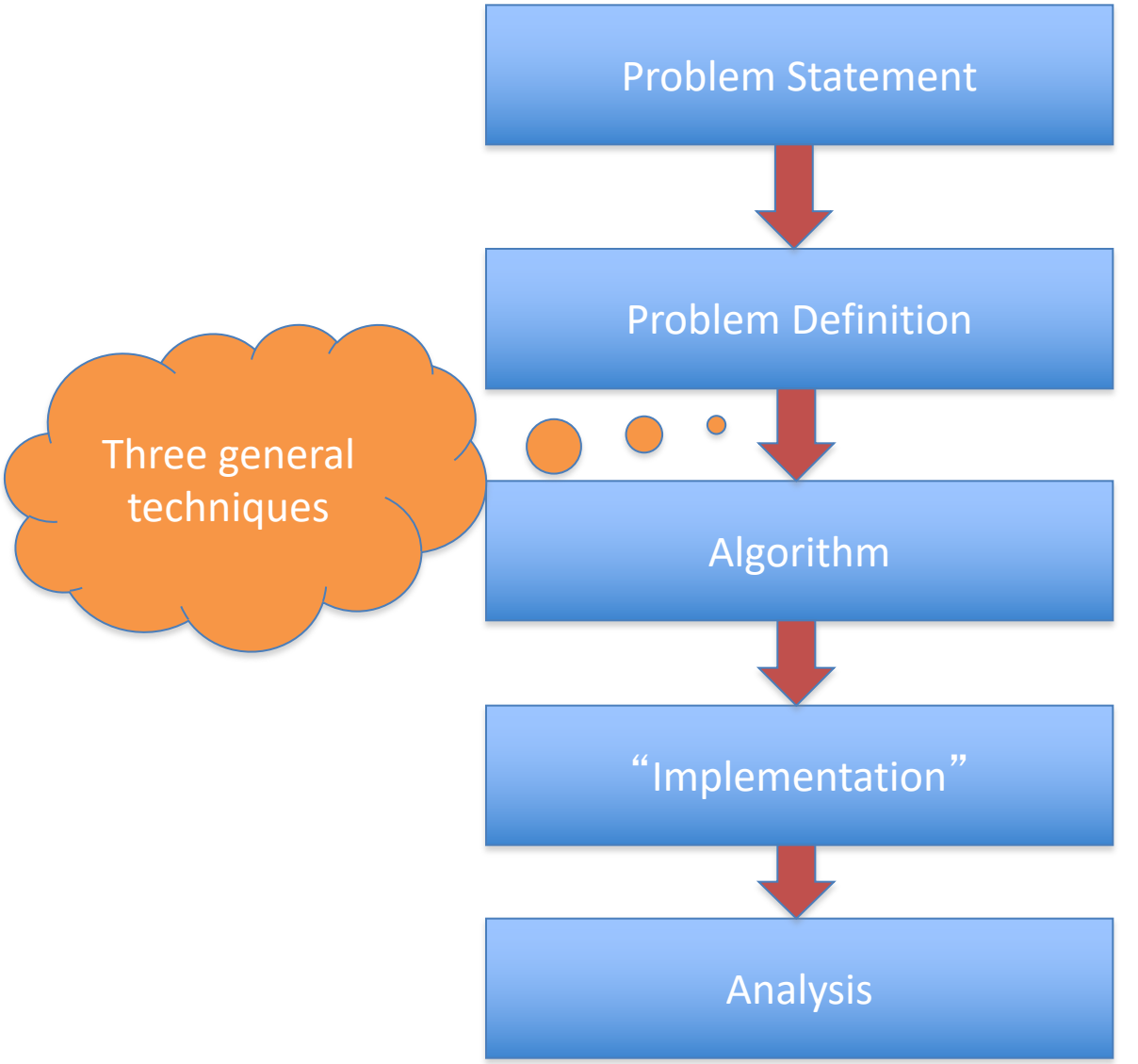
Here are the stats:

Problem	Mean	Median	StdDev	Max	Min
Stakeholder 1	0.7	0.5	0.3	1.0	0.0
Stakeholder 2	0.8	1.0	0.3	1.0	0.0
Stakeholder 3	0.7	0.5	0.4	1.0	0.0

Questions/Comments?



High level view of CSE 331

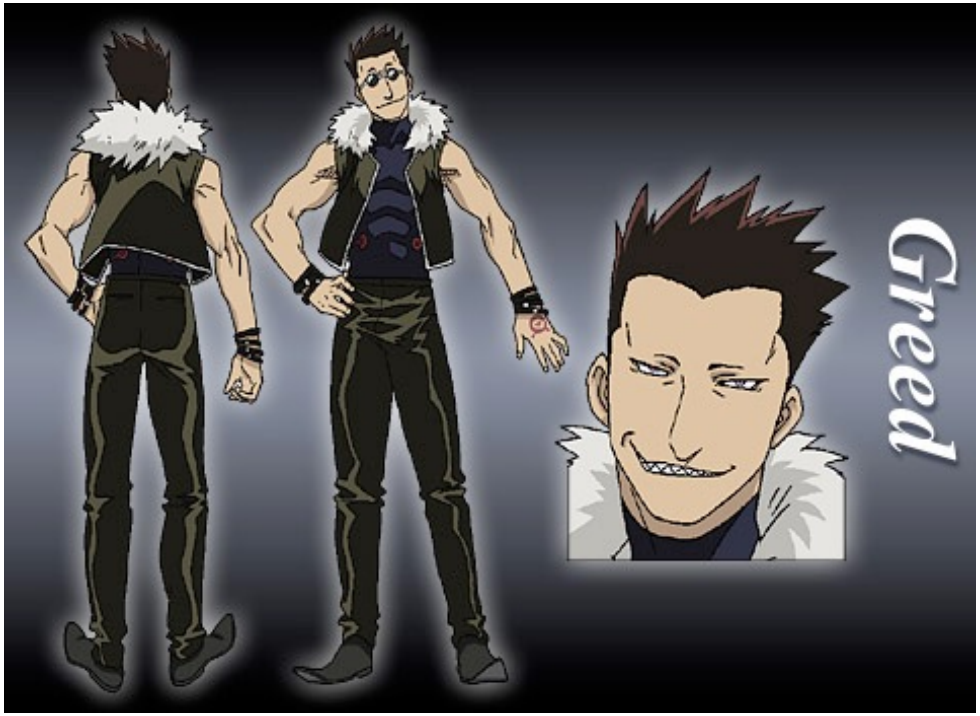


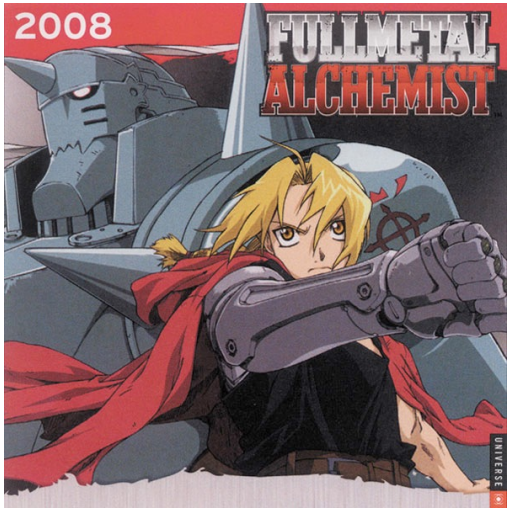
Data Structures

Correctness+Runtime Analysis

Greedy Algorithms

Natural algorithms





Reduced exponential running time to polynomial

Divide and Conquer

Recursive algorithmic paradigm

Reduced large polynomial time to smaller polynomial time

A new algorithmic technique

Dynamic Programming

Dynamic programming vs. Divide & Conquer

Same same because

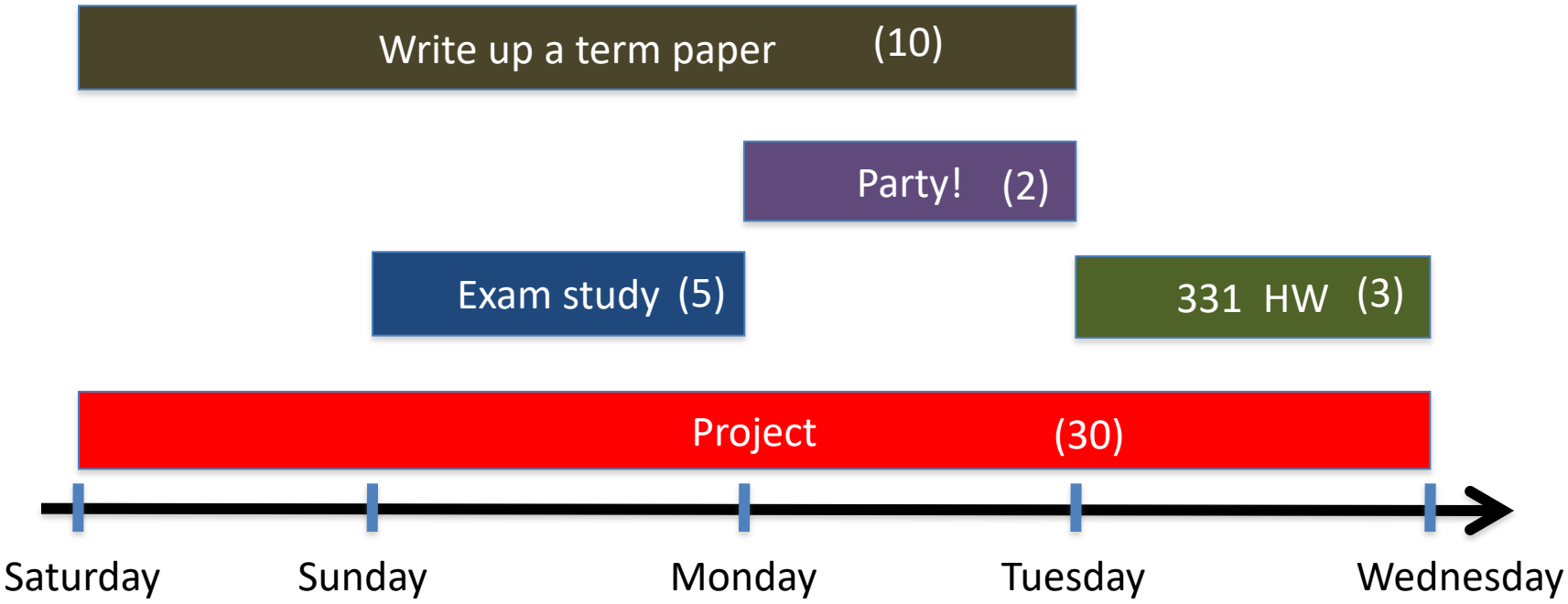
Both design recursive algorithms

Different because

Dynamic programming is smarter about solving recursive sub-problems

End of Semester blues

Can only do one thing at any day: what is the optimal schedule to obtain maximum value?



Previous Greedy algorithm

Order by end time and pick jobs greedily

Greedy value = 5+2+3= 10

Write up a term paper (10)

Party! (2)

Exam study (5)

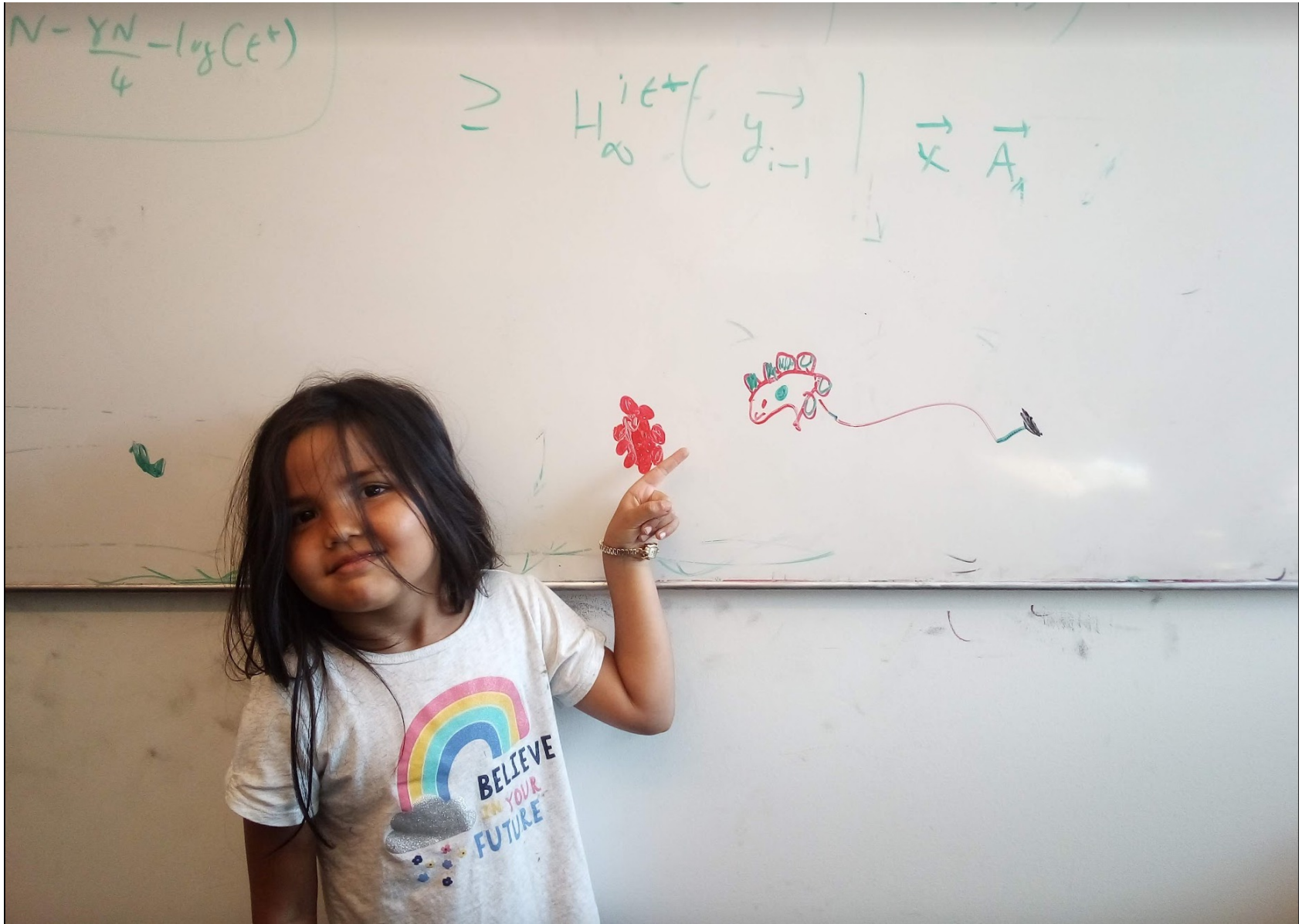
331 HW (3)

OPT = 30

Project (30)

Saturday Sunday Monday Tuesday Wednesday

Questions/Comments?



Today's agenda

Formal definition of the problem

Start designing a recursive algorithm for the problem

Weighted Interval Scheduling

Input: n jobs/intervals. Interval i is triple (s_i, f_i, v_i)

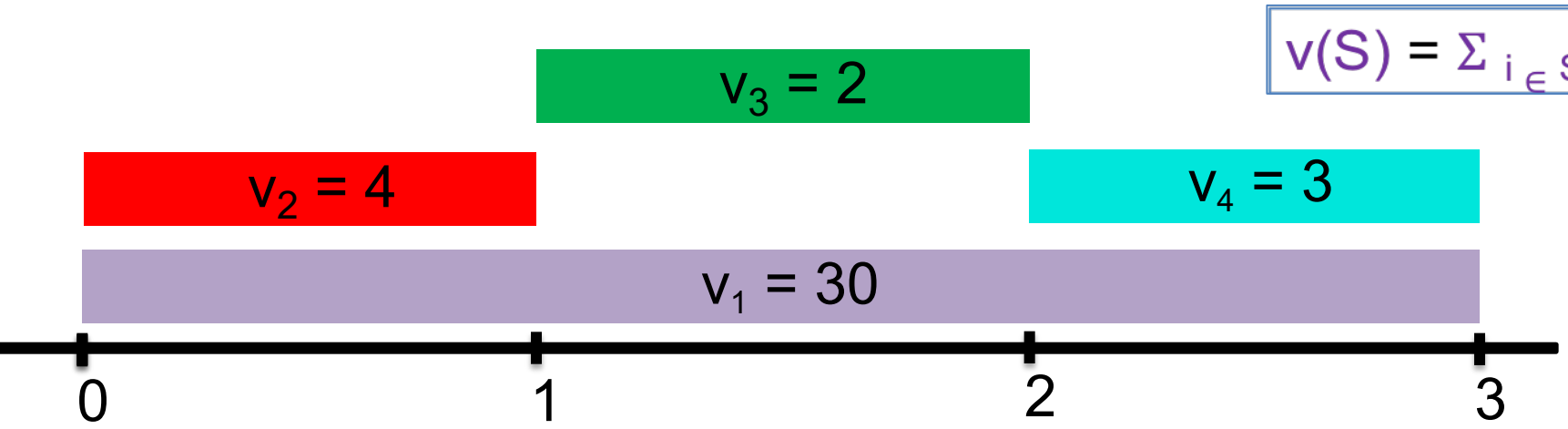
start time

finish time

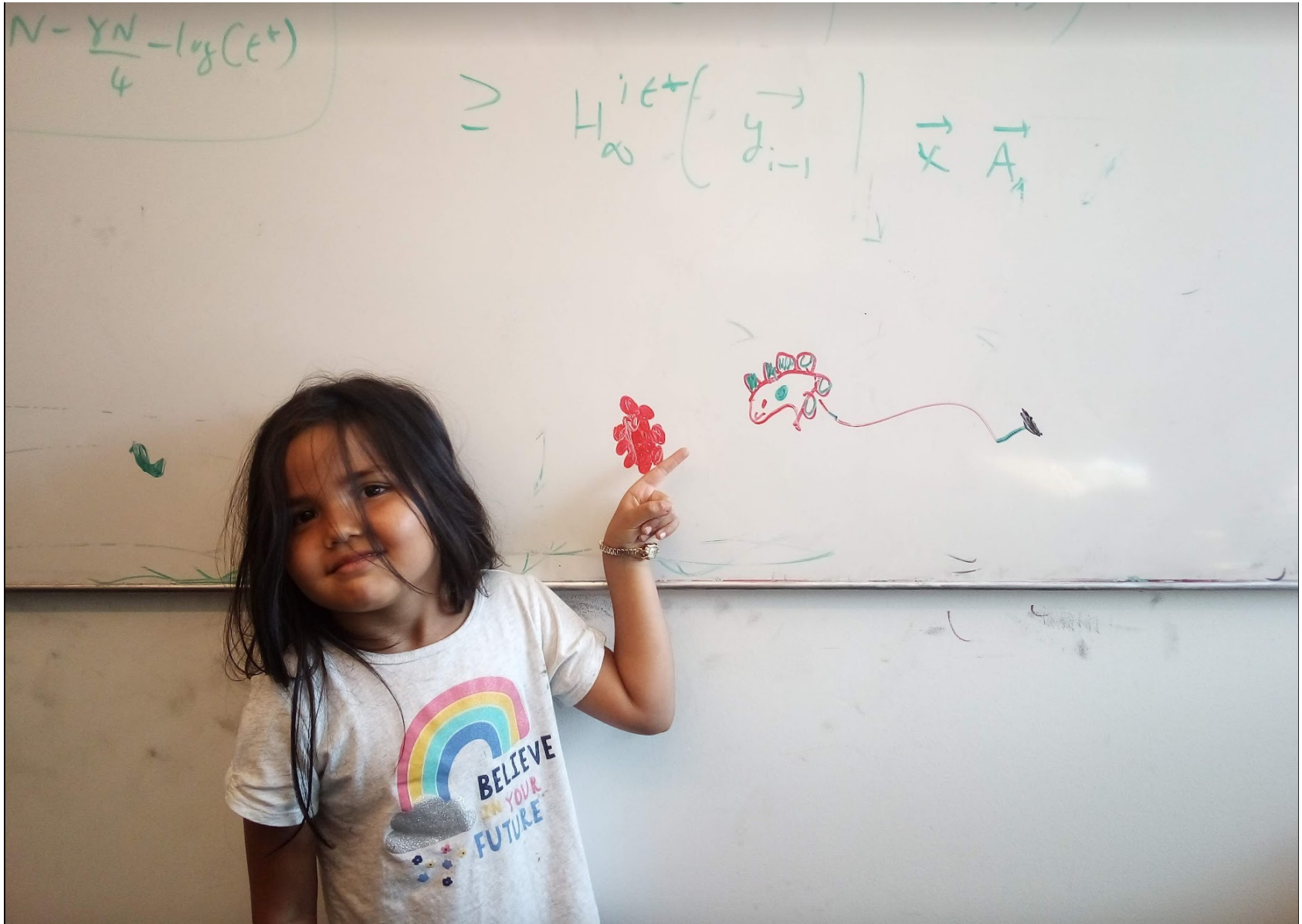
value

Output: A valid schedule $S \subseteq [n]$ that maximizes $v(S)$

$v(S) = \Sigma_{i \in S} v_i$



Questions/Comments?



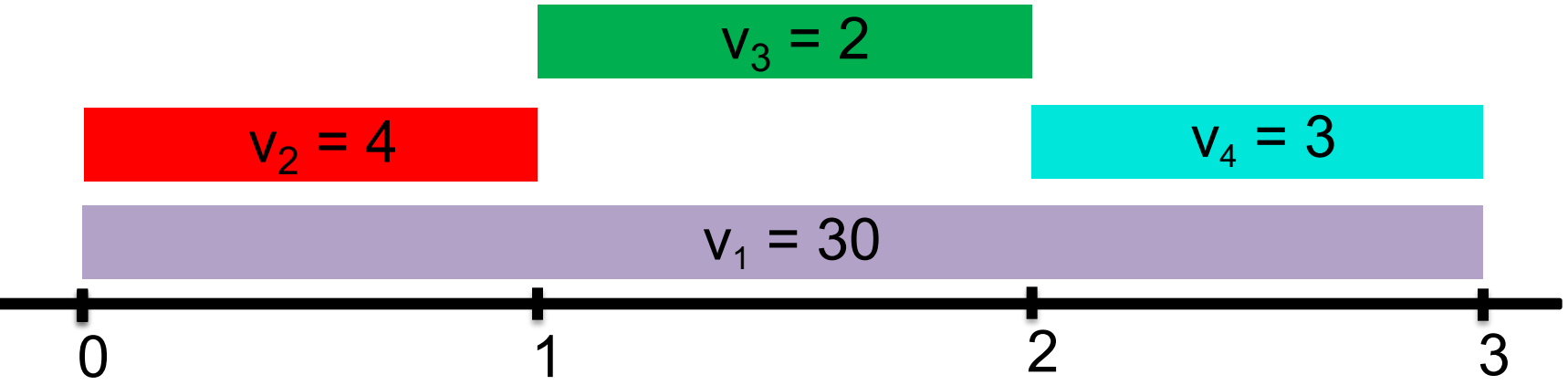
Previous Greedy Algorithm

R = original set of jobs

S = ϕ

While R is not empty
    Choose i in R where f_i is the smallest
    Add i to S
    Remove all requests that conflict with i from R

Return S* = S



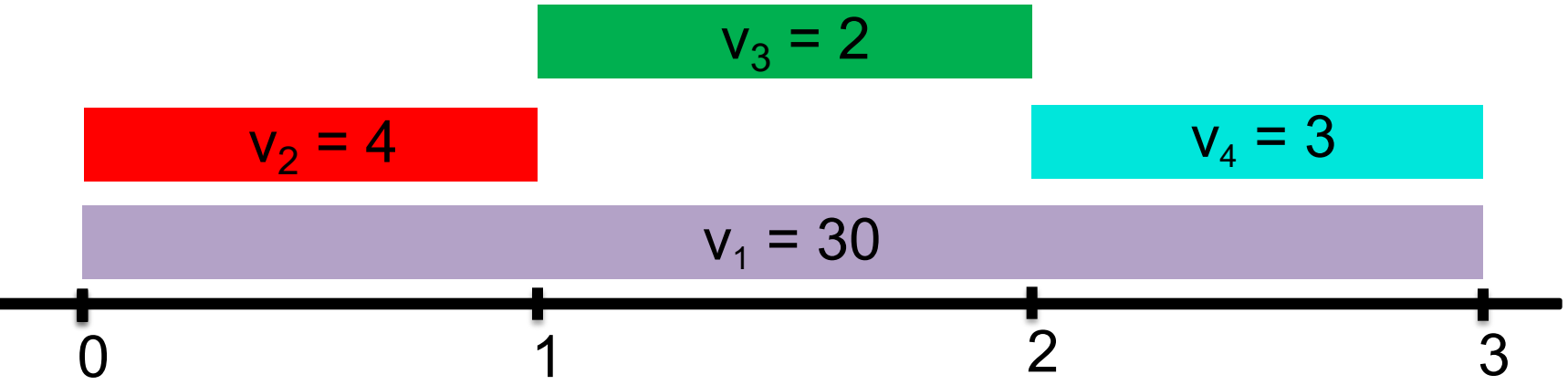
Perhaps be greedy differently?

R = original set of jobs

S = ϕ

While R is not empty
  Choose i in R where $v_i/(f_i - s_i)$ is the largest
  Add i to S
  Remove all requests that conflict with i from R

Return S* = S



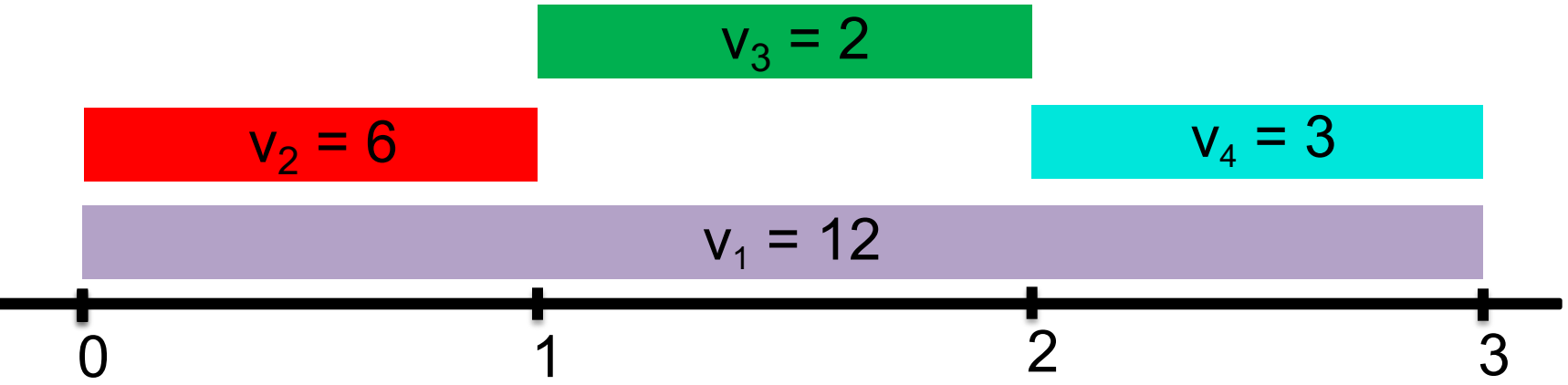
Can this work?

R = original set of jobs

S = ϕ

While R is not empty
    Choose i in R where $v_i/(f_i - s_i)$ is the largest
    Add i to S
    Remove all requests that conflict with i from R

Return S* = S



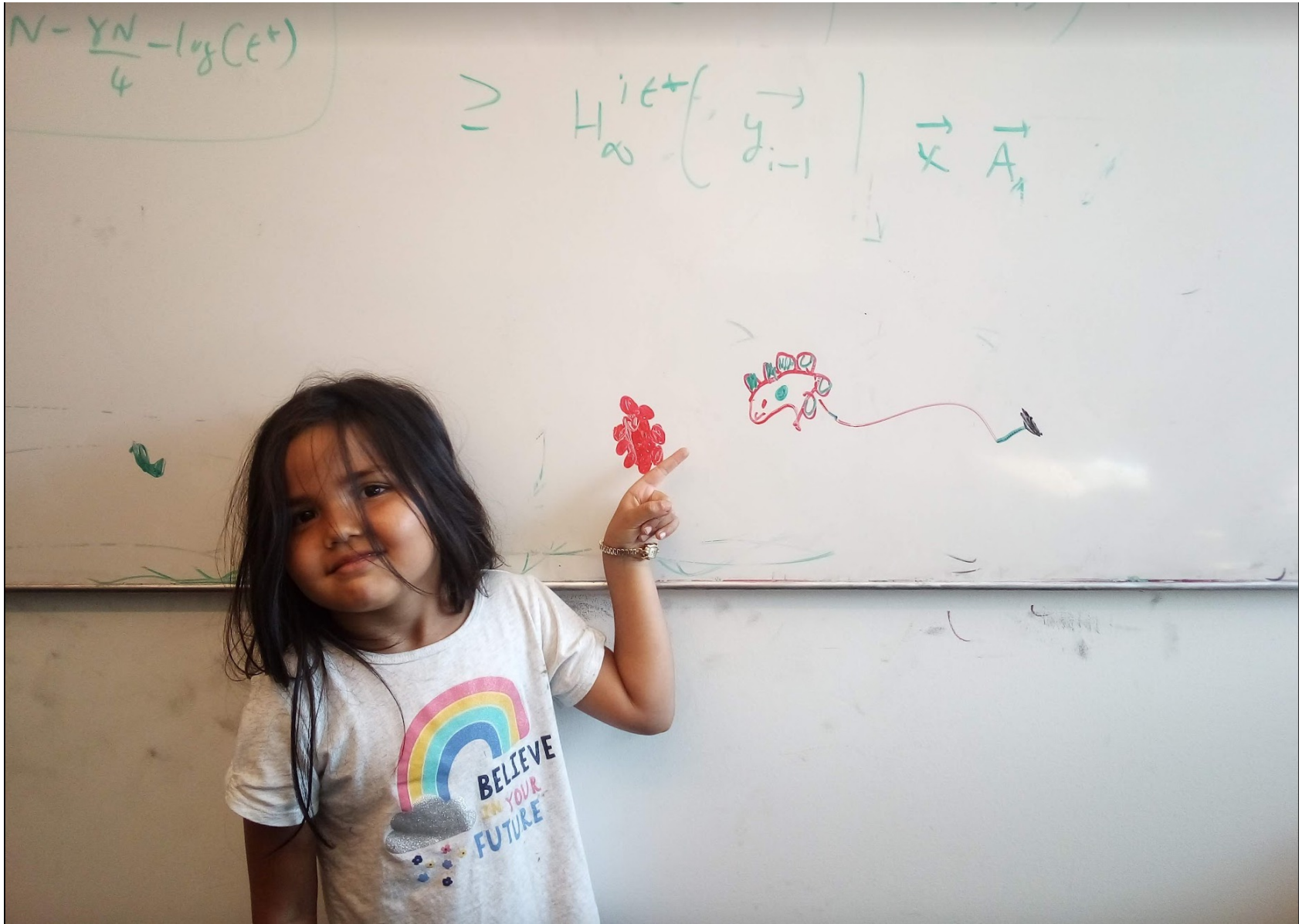
Avoiding the greedy rabbit hole

https://www.writerightwords.com/down-the-rabbit-hole/

Provably IMPOSSIBLE for a large class of greedy algos

There are no known greedy algorithm to solve this problem

Questions/Comments?



Perhaps a divide & conquer algo?

Divide the problem in 2 or more many EQUAL SIZED INDEPENDENT problems

Recursively solve the sub-problems

Patchup the SOLUTIONS to the sub-problems

Perhaps a divide & conquer algo?

RecurWeightedInt([n])

if n = 1 return the only interval

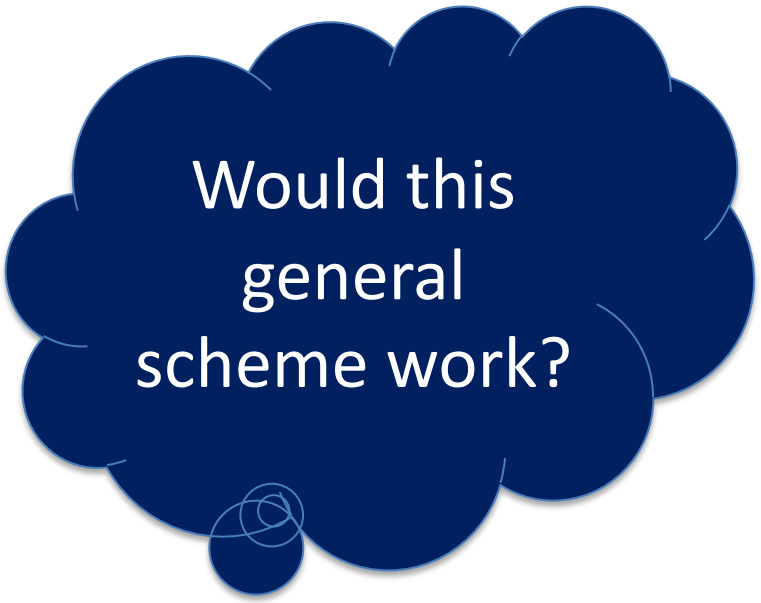
L = first n/2 intervals

R = last n/2 intervals

S_L = RecurWeightedInt(L)

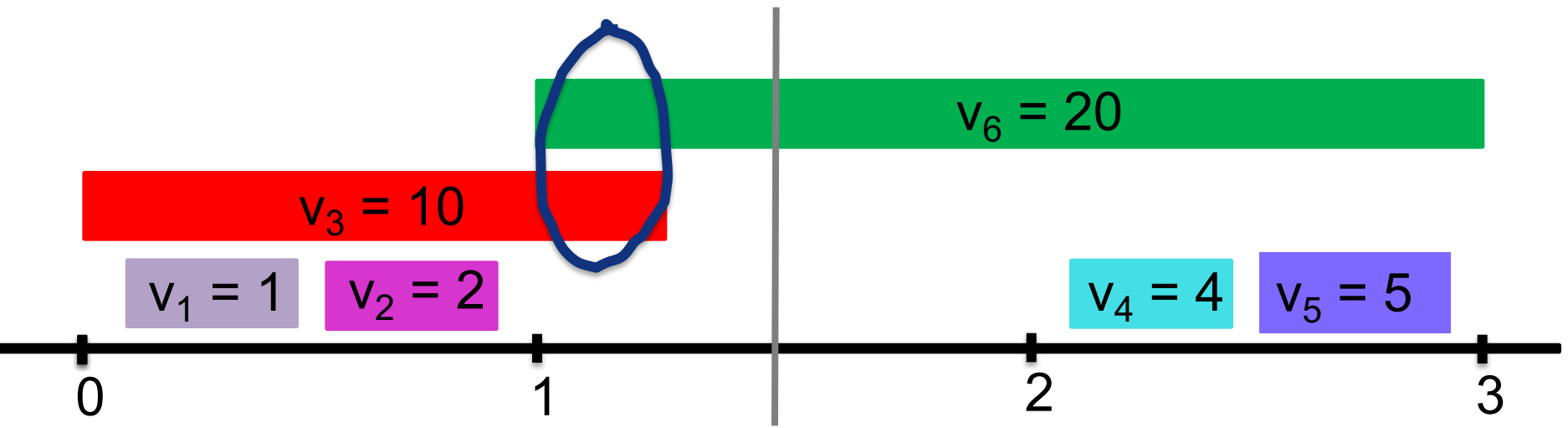
S_R = RecurWeightedInt(R)

PatchUp(S_L, S_R)



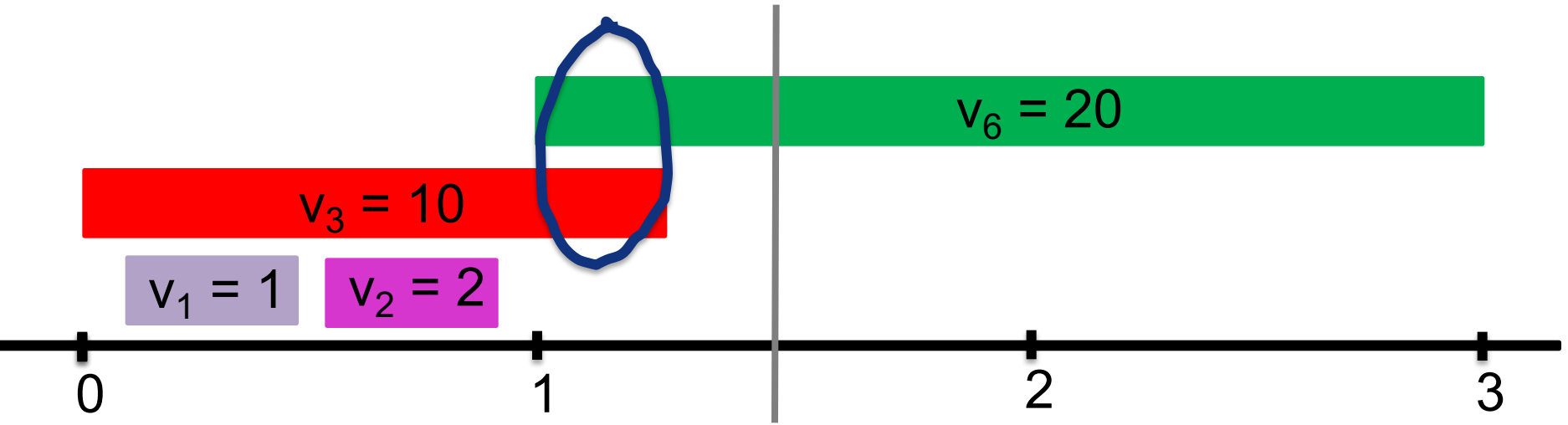
Divide the problem in 2 or more many EQUAL SIZED INDEPENDENT problems

Sub-problems NOT independent!

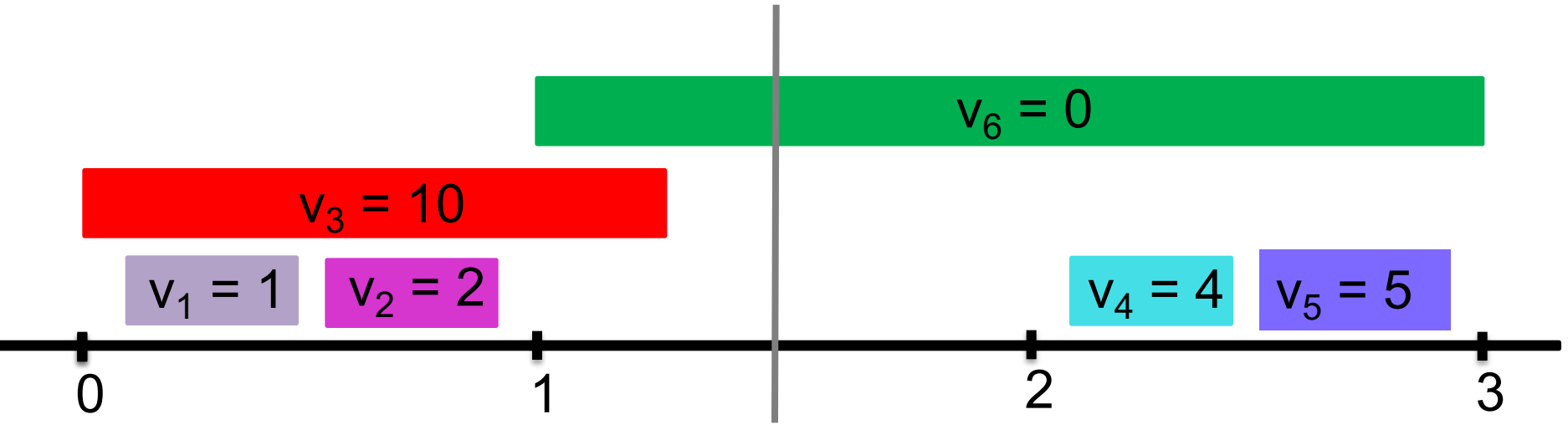


Perhaps patchup can help?

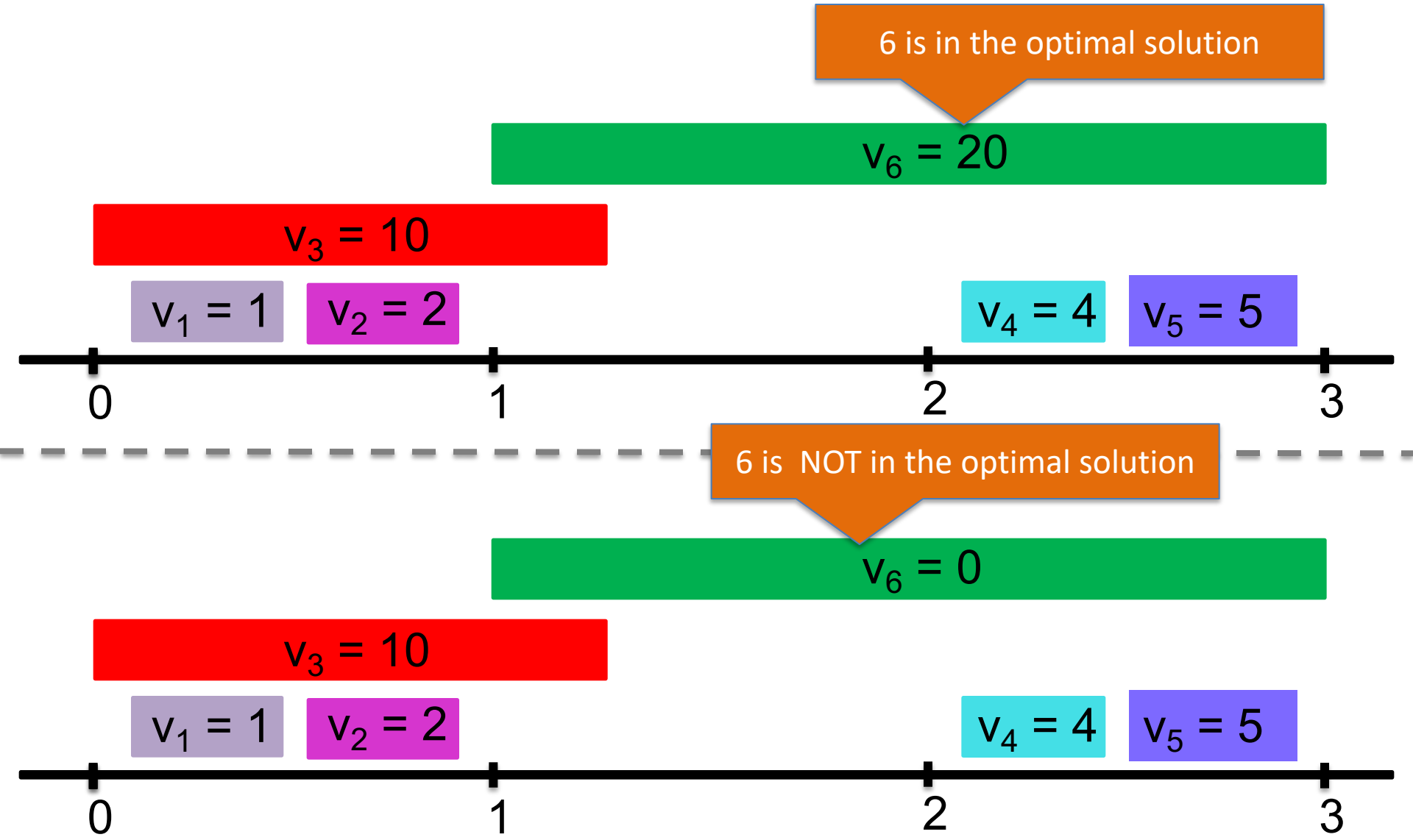
Patchup the SOLUTIONS to the sub-problems



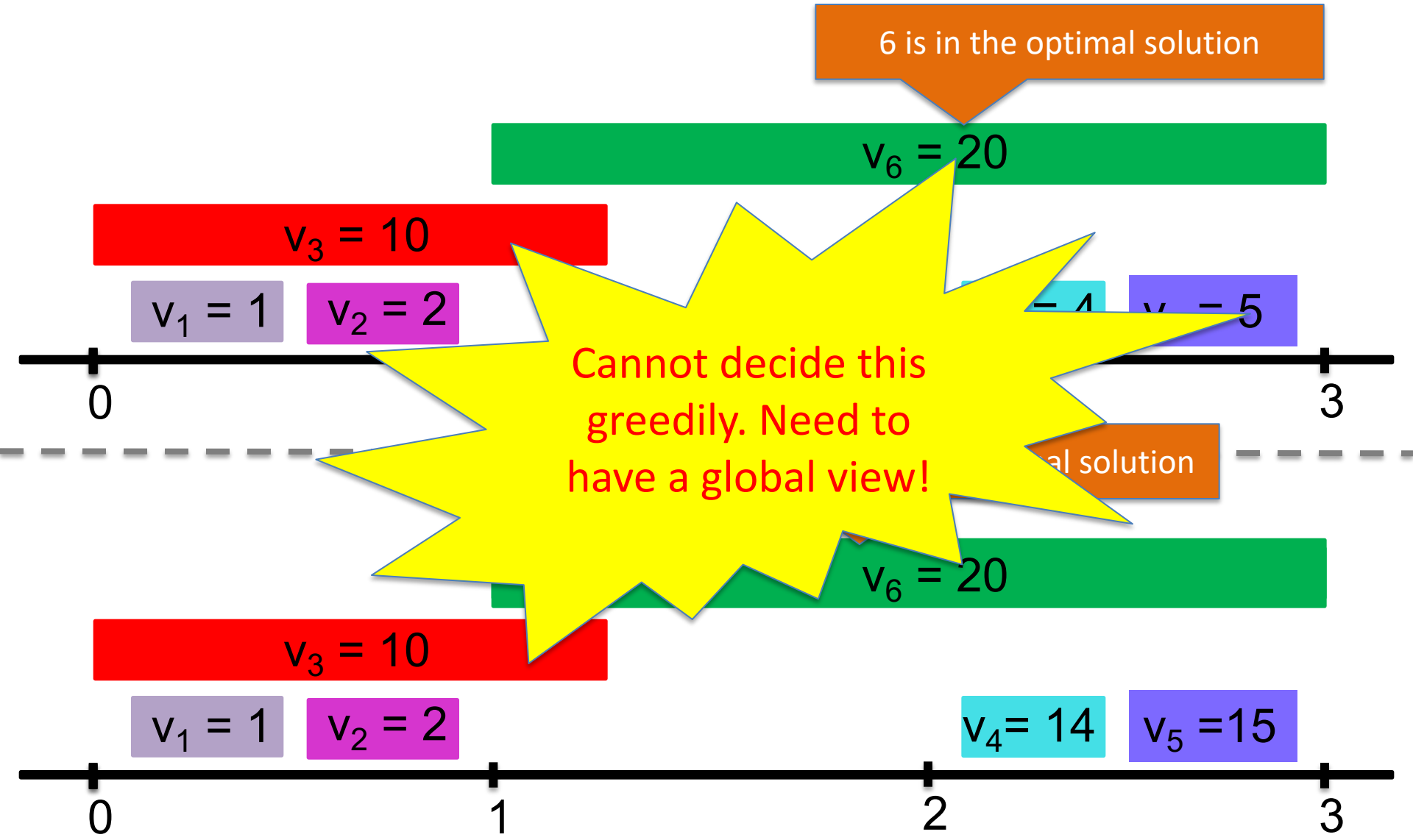
Sometimes patchup NOT needed!



Check for two cases?



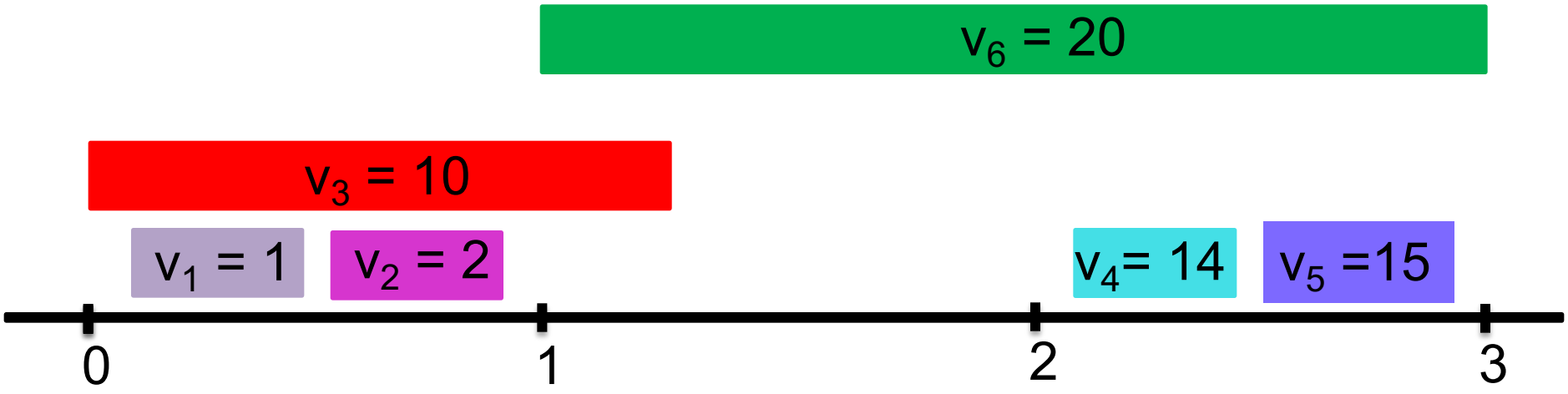
Check if v_6 is the largest value?



Check out both options!

Case 1: 6 is in the optimal solution

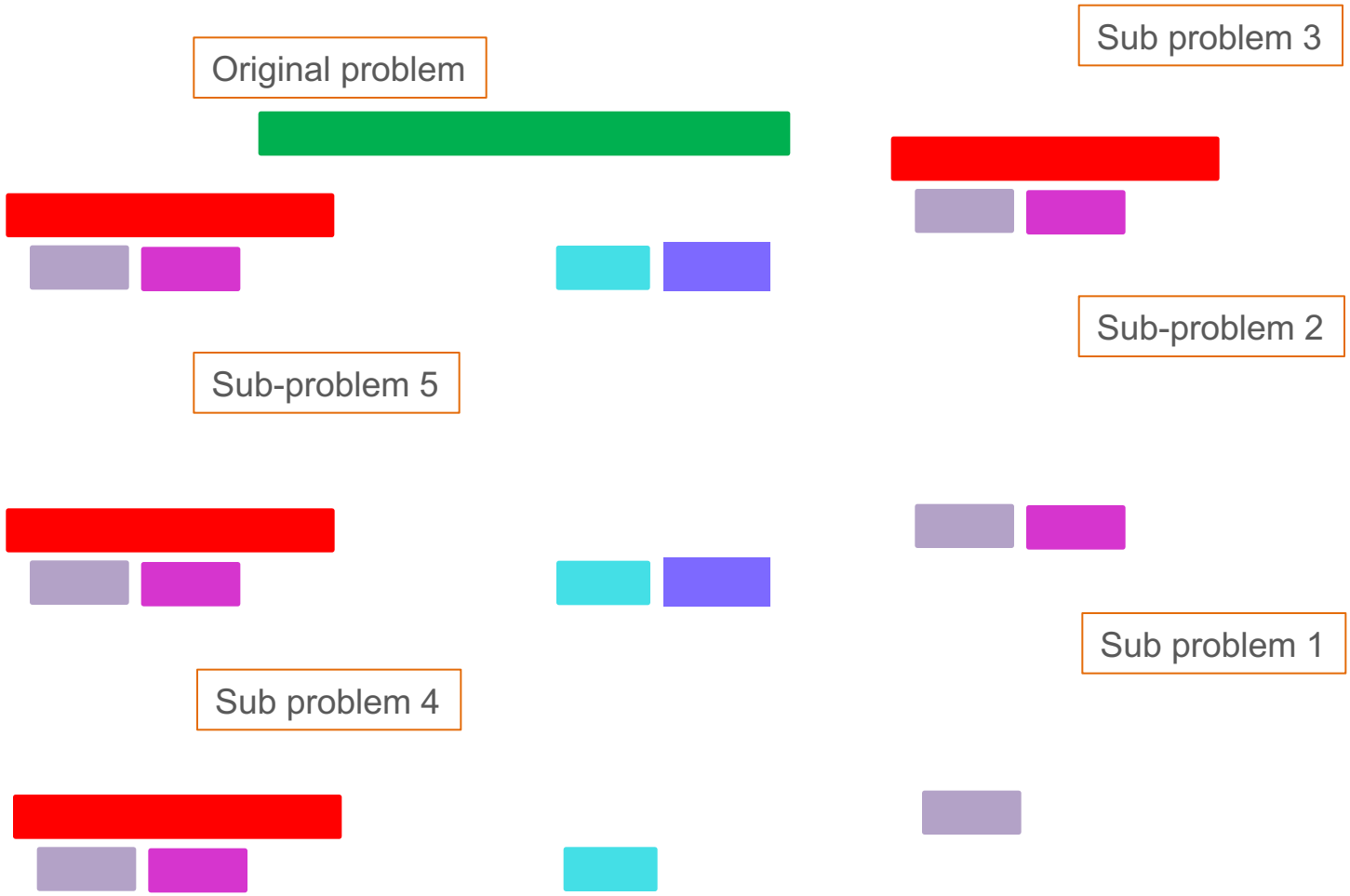
6 is not in optimal solution




?!
ICANHASCHEEZBURGER.COM

So what sub-problems?

Divide the problem in 2 or more many ~~EQUAL SIZED INDEPENDENT~~ problems




?!
ICANHASCHEEZBURGER.COM