

Lecture 29

CSE 331

Nov 10, 2023

HW 6 Q1+2 templates buggy

question @461 ⊞ ★ 🔒 stop following 32 views Actions ▾

Question of subsections in LaTeX

It seems that Collaborator and Sources have disappeared from latex template since HW5. Even though I can type them by myself, may I ask that is there any chance we'll see them back in the rest of HWs (I'm a bit lazy)?

Thank you!

(Making this public. --Atri)

homework6

~ An instructor (Atri Rudra) endorsed this question ~

Reflections 1+2 grading

note @466   

stop following **16 views**

Actions ▾

Reflections 1+2 grading timeline

I'll be grading all the reflections for the project.

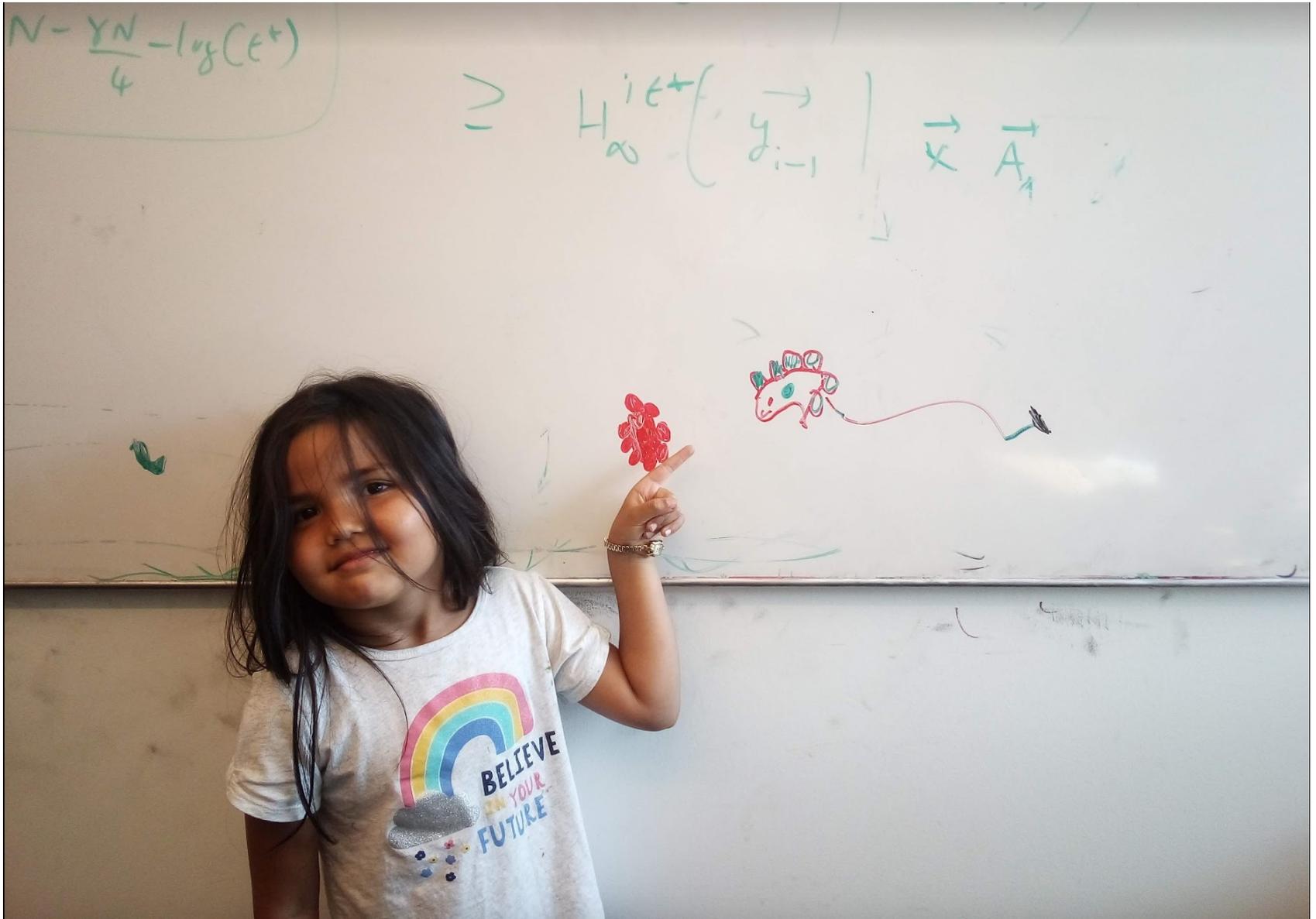
I'm hoping to get both the reflections graded in 1-2 weeks time but in the worst-case I'll get them done around a week before reflections 3 is done (so worst-case a day or two after the Thanksgiving break).

project

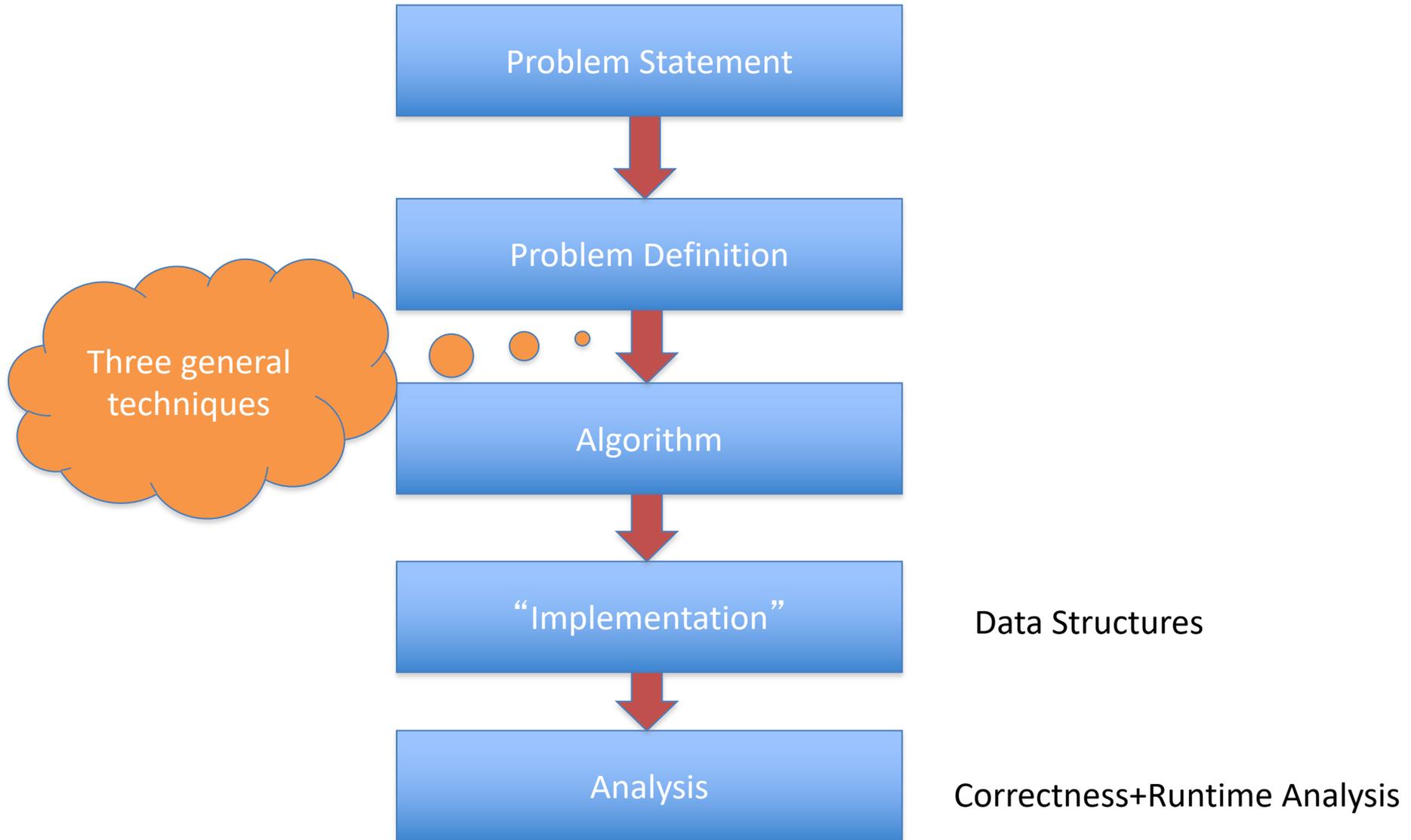
Edit good note | 1

Updated 3 minutes ago by Atri Rudra

Questions/Comments?

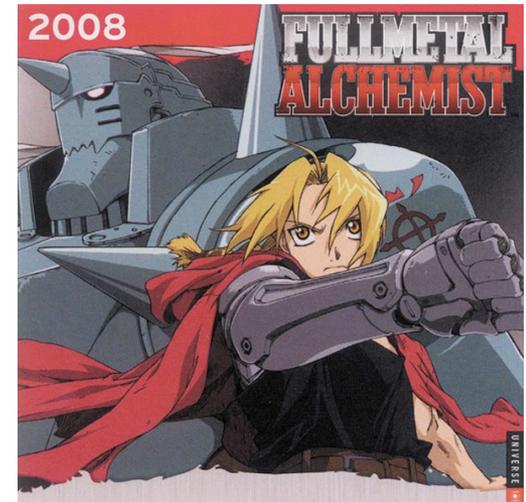


High level view of CSE 331



Greedy Algorithms

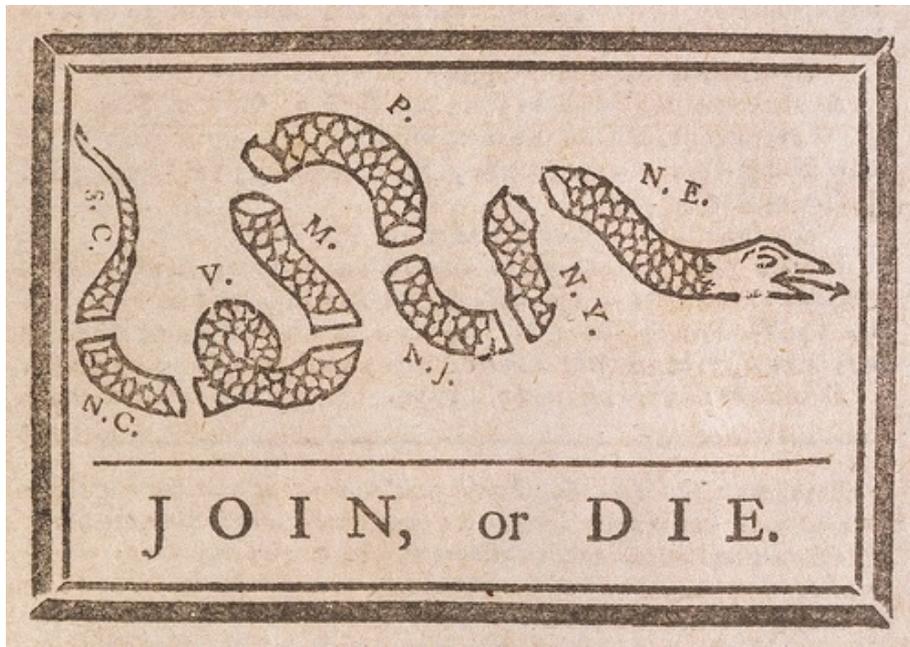
Natural algorithms



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?



Write up a term paper (10)

Party! (2)

Exam study (5)

331 HW (3)

Project (30)

Friday

Saturday

Sunday

Monday

Tuesday

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)

Project (30)

OPT = 30



Friday

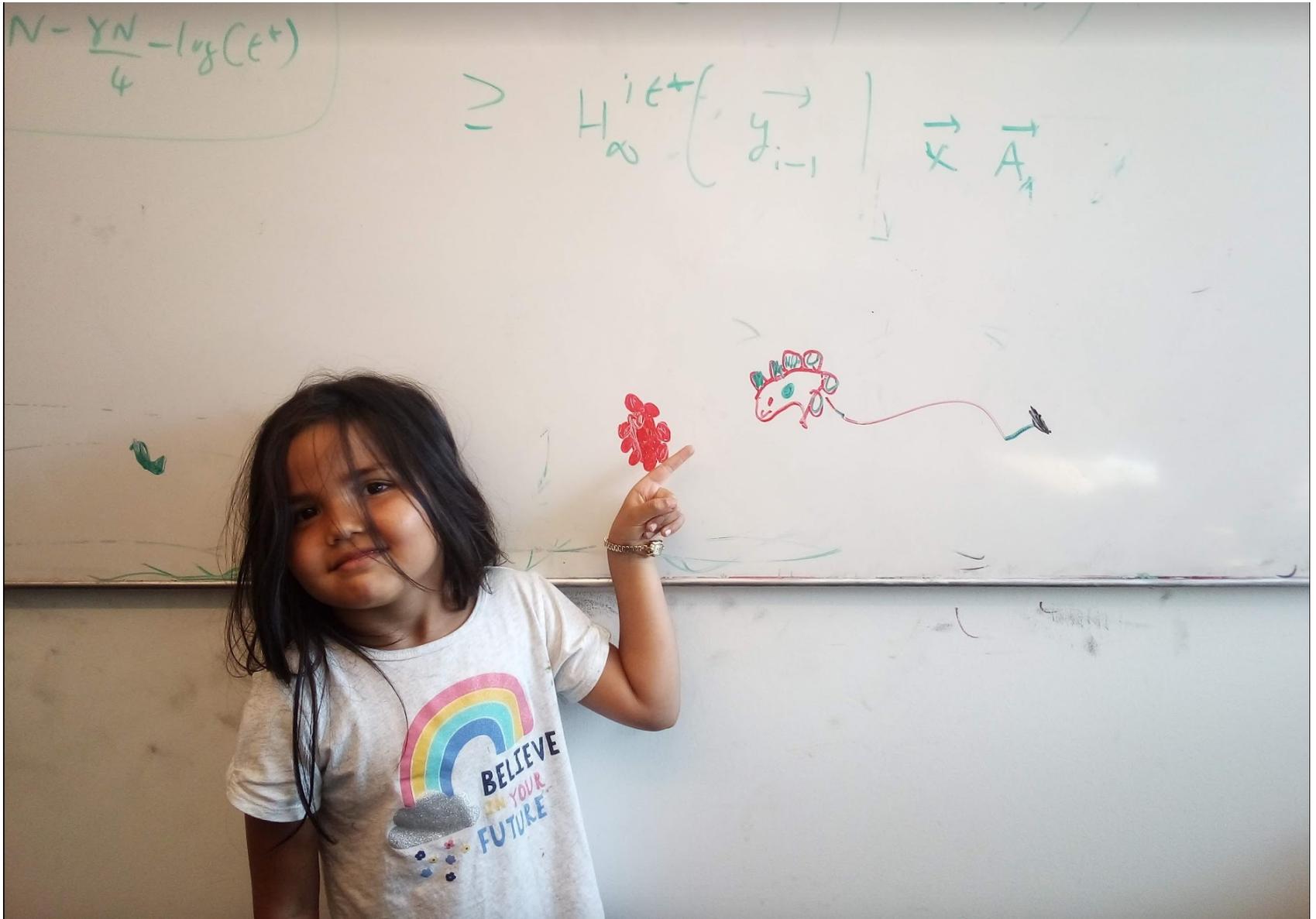
Saturday

Sunday

Monday

Tuesday

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)

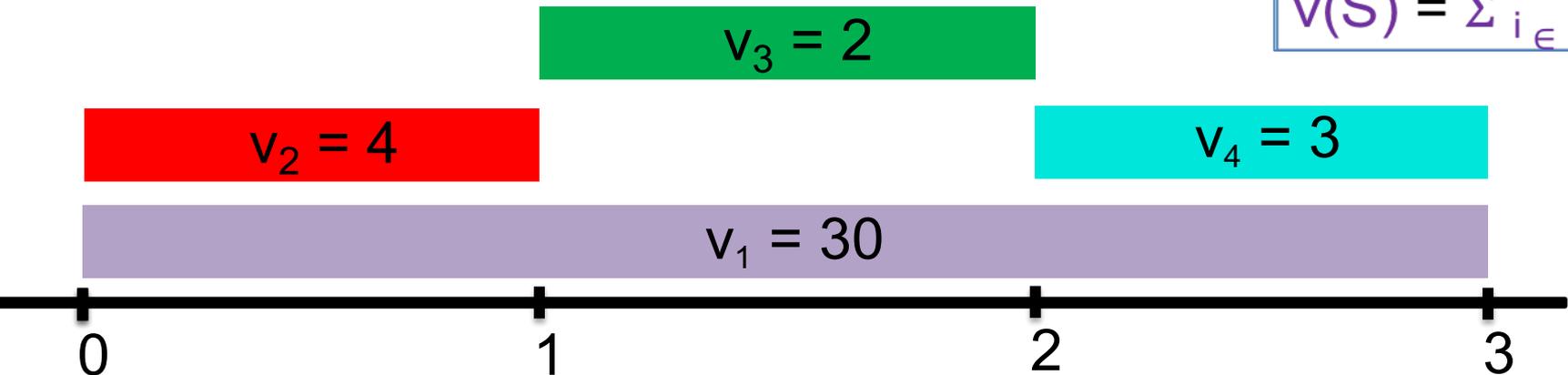
start time

finish time

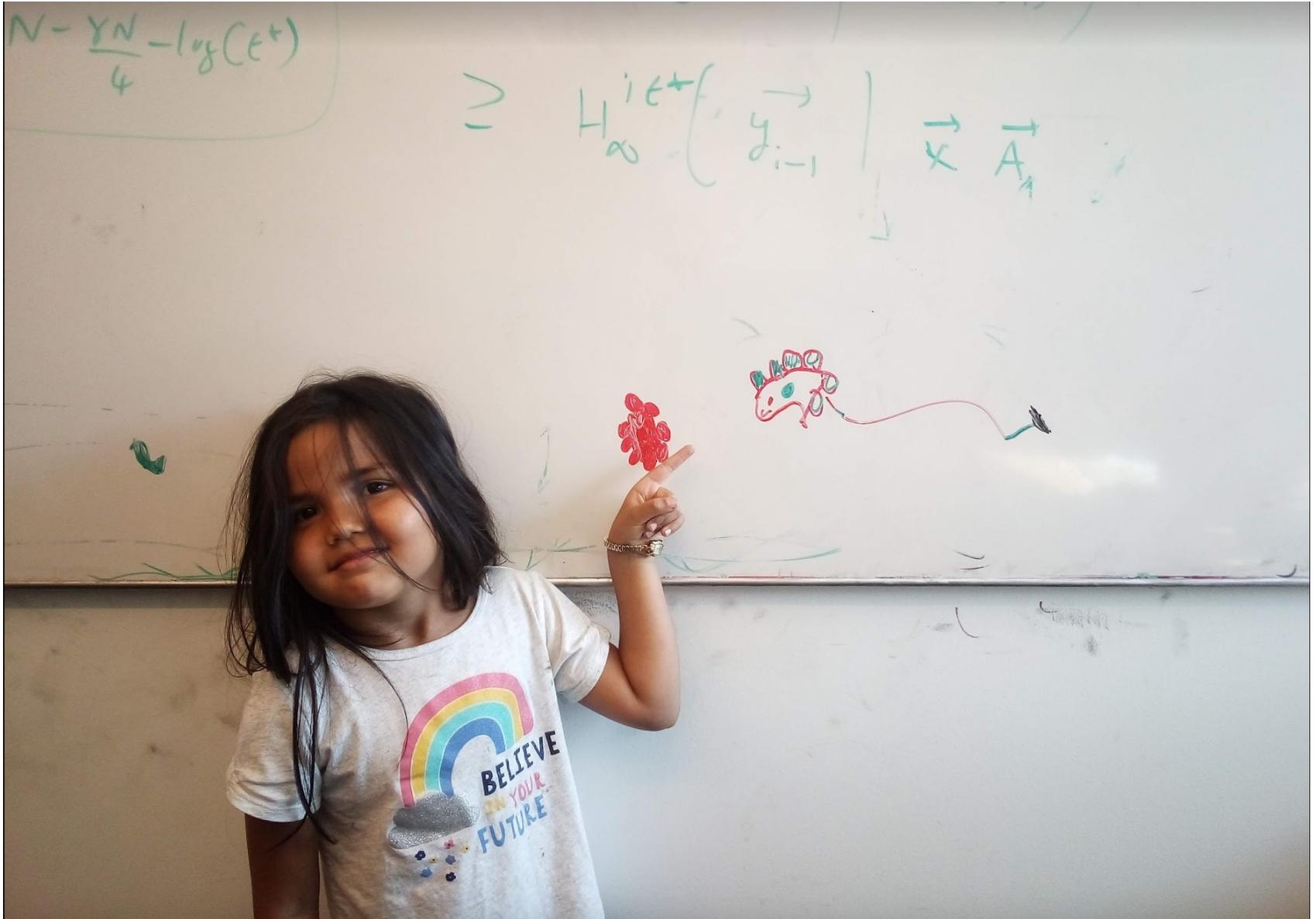
value

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

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



Questions/Comments?



Previous Greedy Algorithm

R = original set of jobs

$S = \phi$

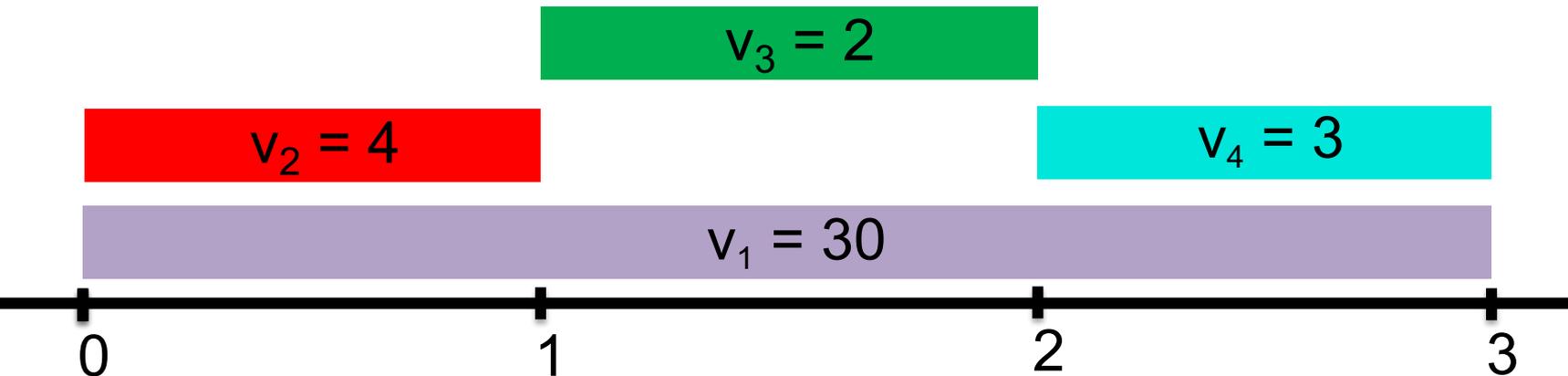
While R is not empty

 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

$S = \phi$

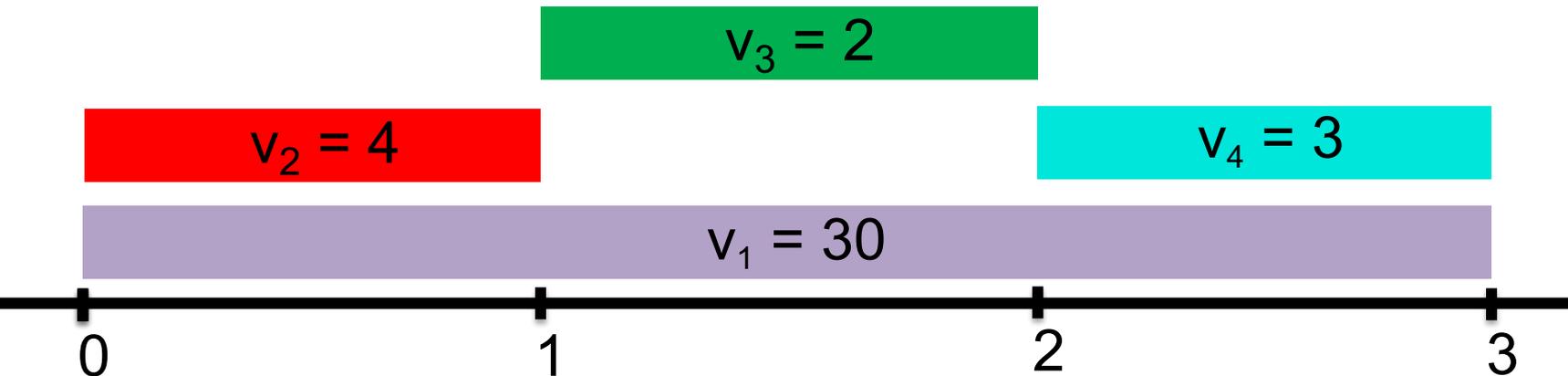
While R is not empty

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

$S = \phi$

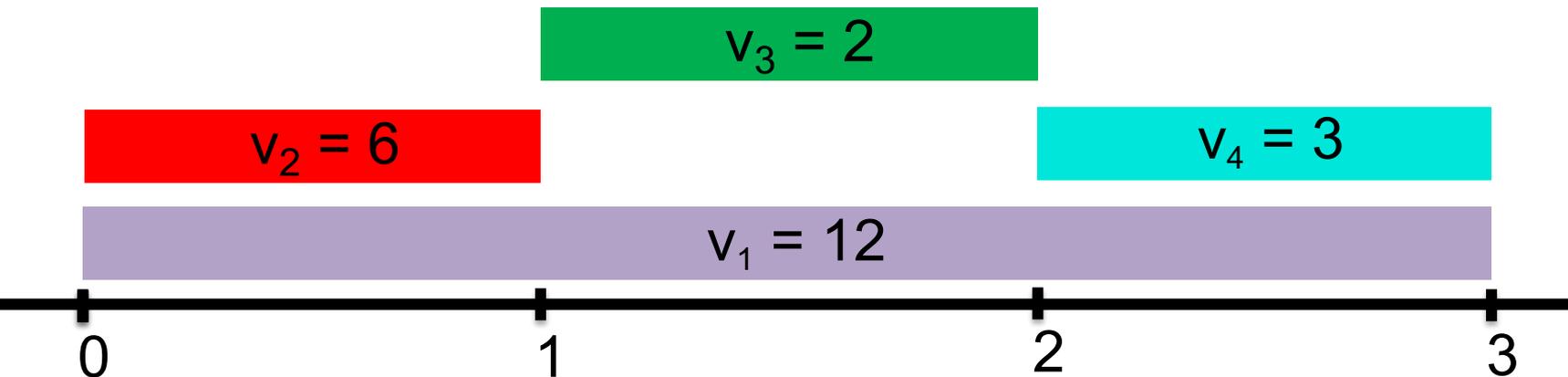
While R is not empty

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

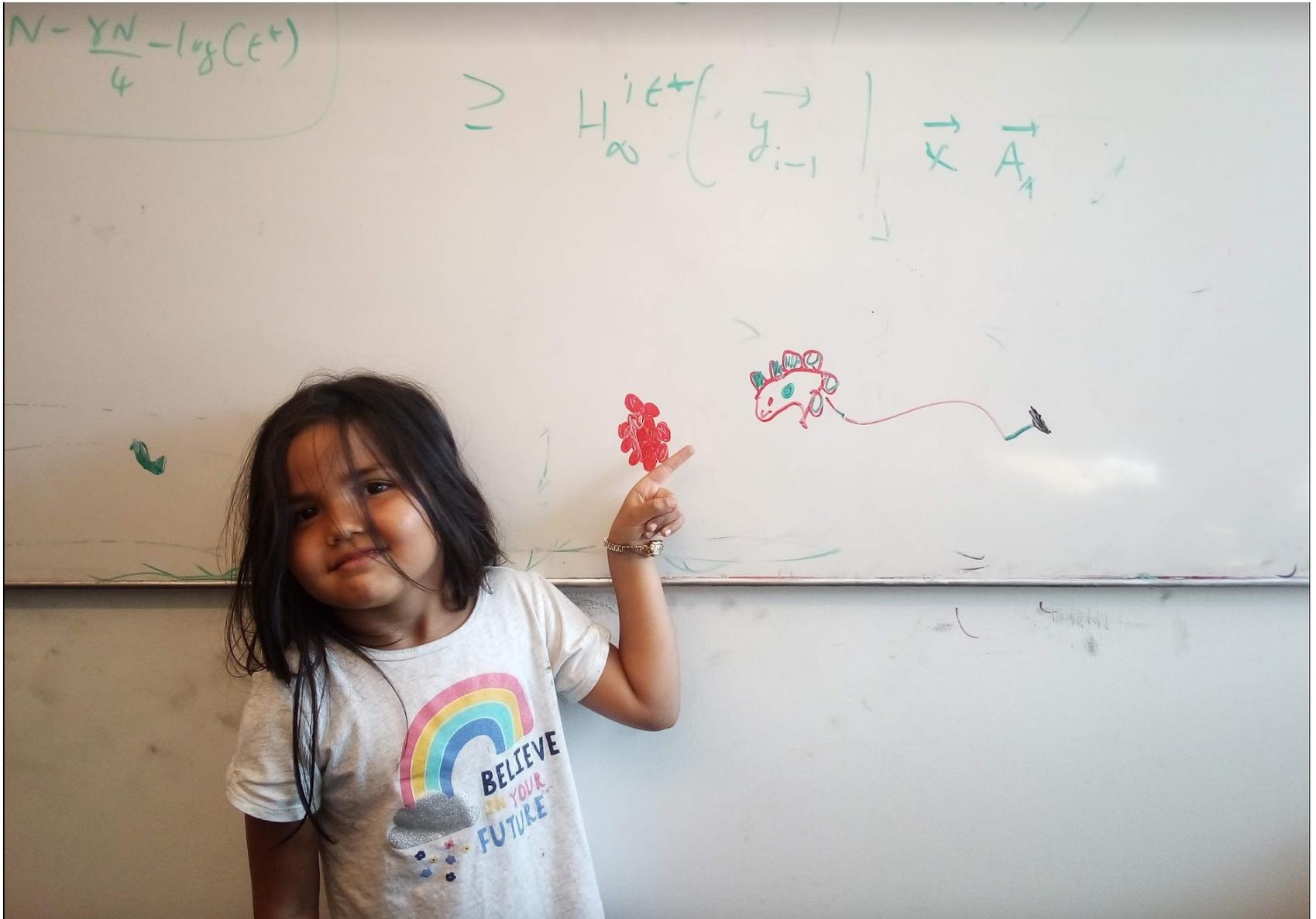


<https://www.wrightwords.com/down-the-rabbit-hole/>

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

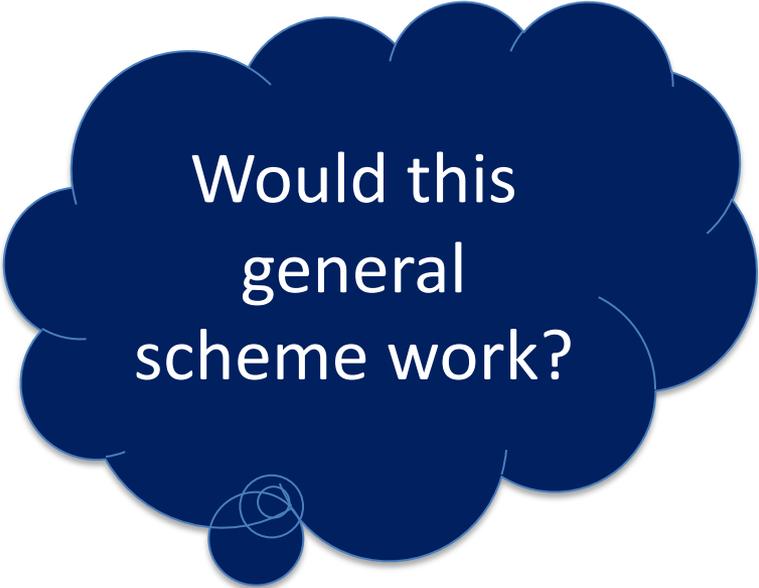
L = first $n/2$ intervals

R = last $n/2$ intervals

S_L = RecurWeightedInt(L)

S_R = RecurWeightedInt(R)

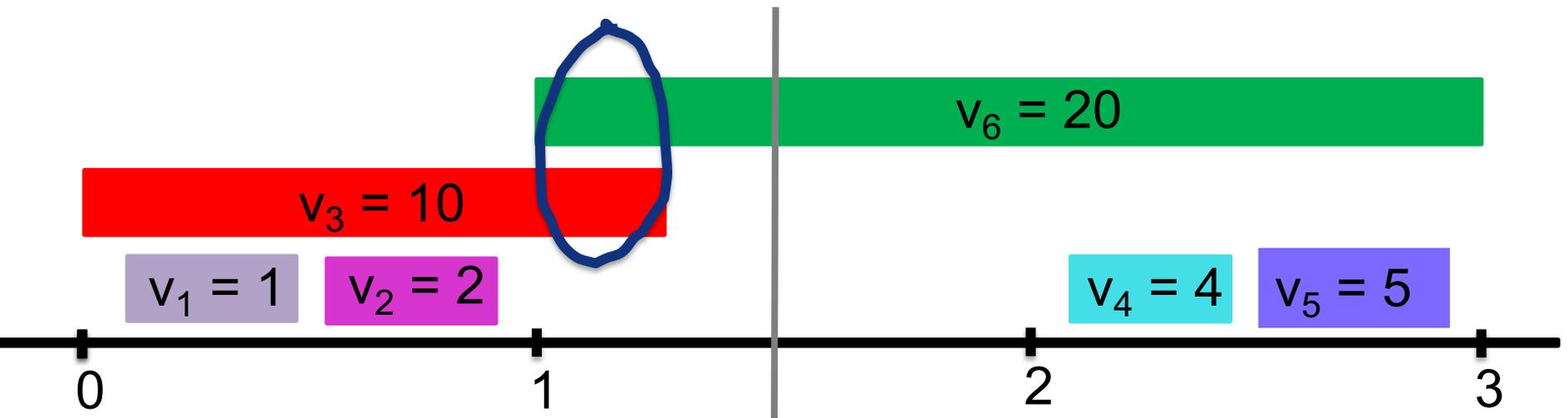
PatchUp(S_L , S_R)



Would this
general
scheme work?

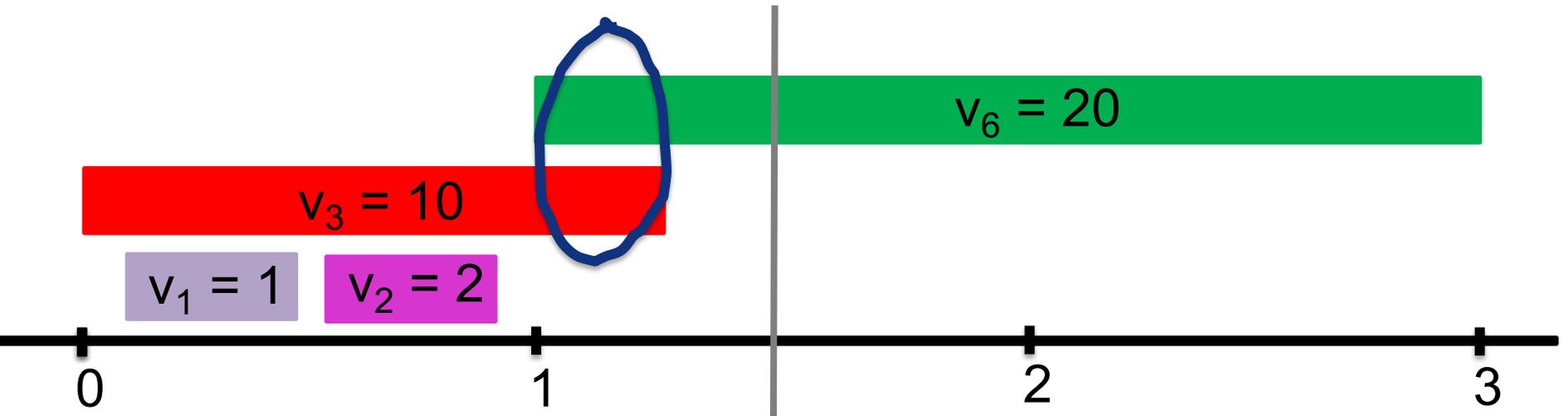
Divide the problem in 2 or more many EQUAL SIZED
INDEPENDENT problems

Sub-problems NOT independent!

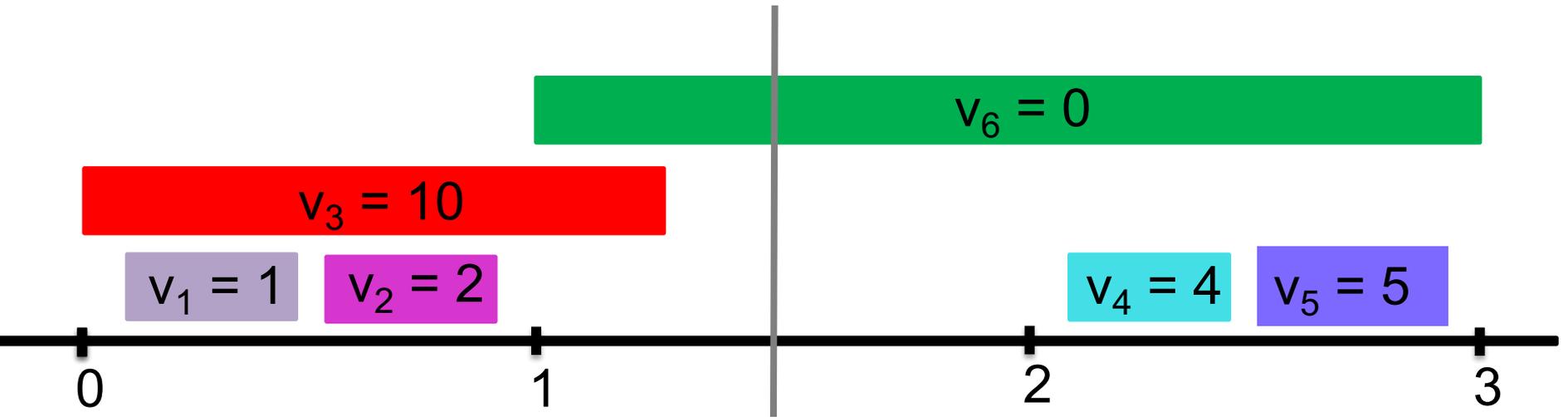


Perhaps patchup can help?

Patchup the SOLUTIONS to the sub-problems



Sometimes patchup NOT needed!



Check for two cases?

6 is in the optimal solution

$$v_6 = 20$$

$$v_3 = 10$$

$$v_1 = 1$$

$$v_2 = 2$$

$$v_4 = 4$$

$$v_5 = 5$$

6 is NOT in the optimal solution

$$v_6 = 0$$

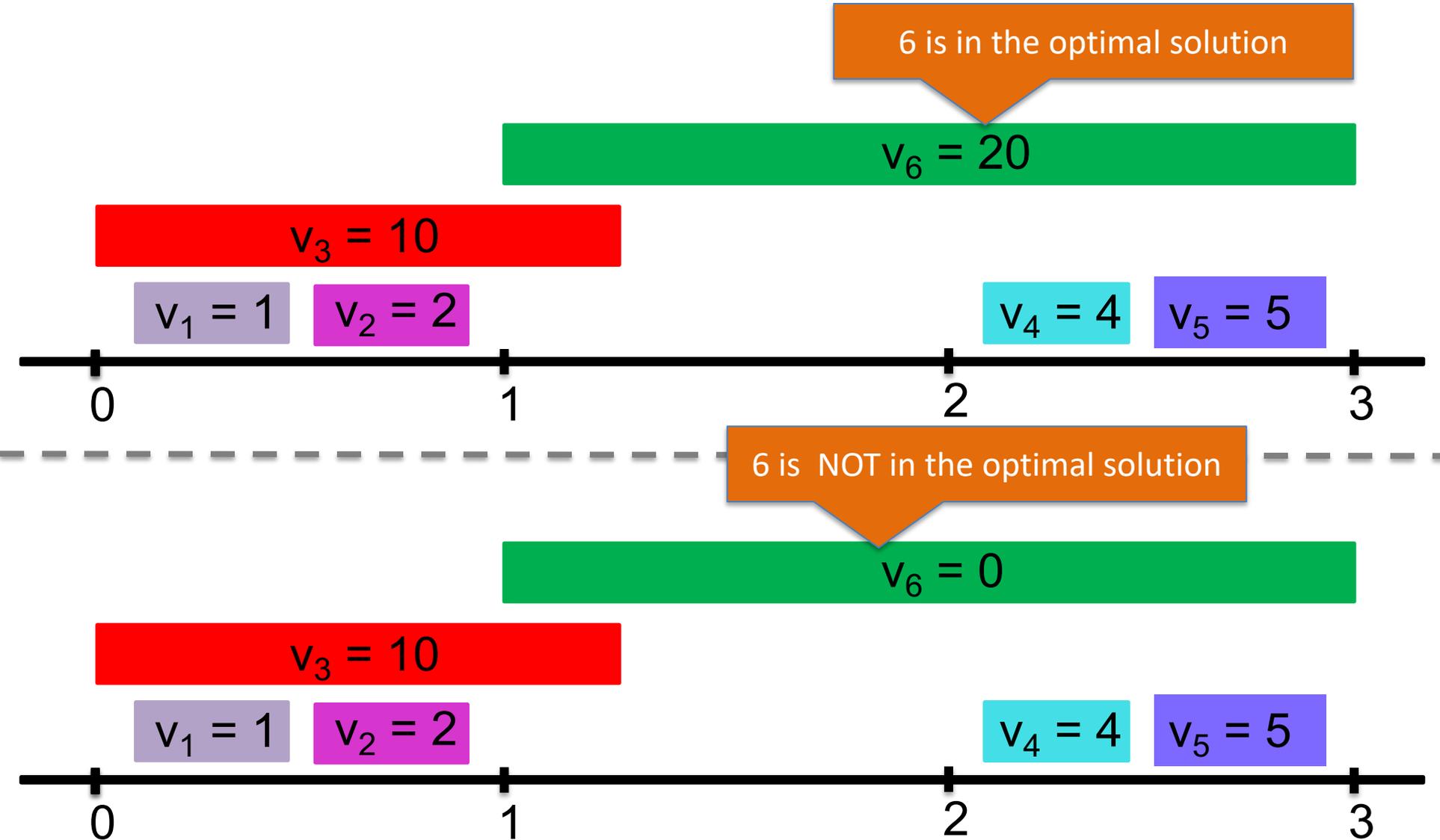
$$v_3 = 10$$

$$v_1 = 1$$

$$v_2 = 2$$

$$v_4 = 4$$

$$v_5 = 5$$



Check if v_6 is the largest value?

6 is in the optimal solution

$v_6 = 20$

$v_3 = 10$

$v_1 = 1$

$v_2 = 2$

$v_4 = 4$

$v_5 = 5$

Cannot decide this greedily. Need to have a global view!

al solution

$v_6 = 20$

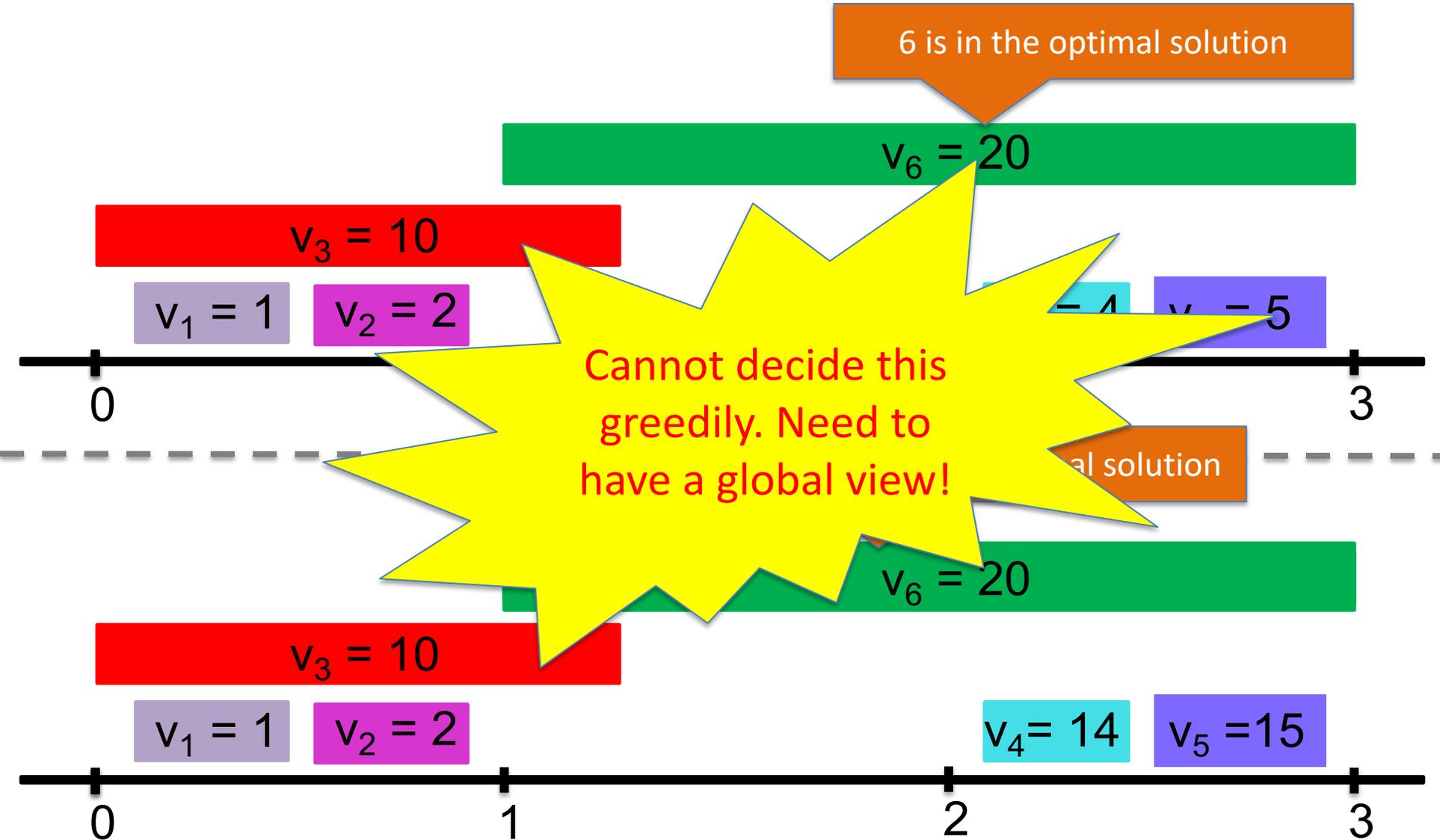
$v_3 = 10$

$v_1 = 1$

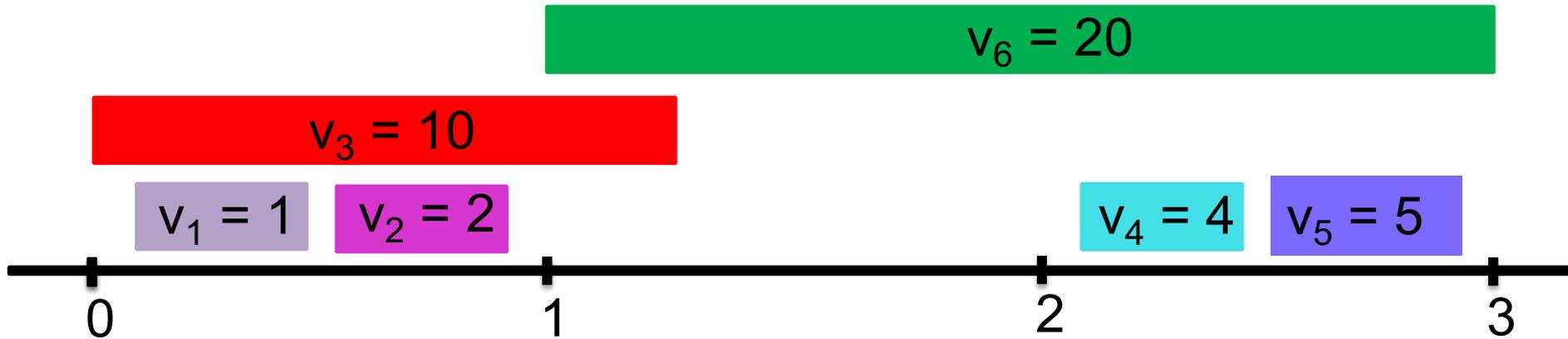
$v_2 = 2$

$v_4 = 14$

$v_5 = 15$



Check out both options!



Case 1: 6 is in the optimal solution

6 is not in optimal solution

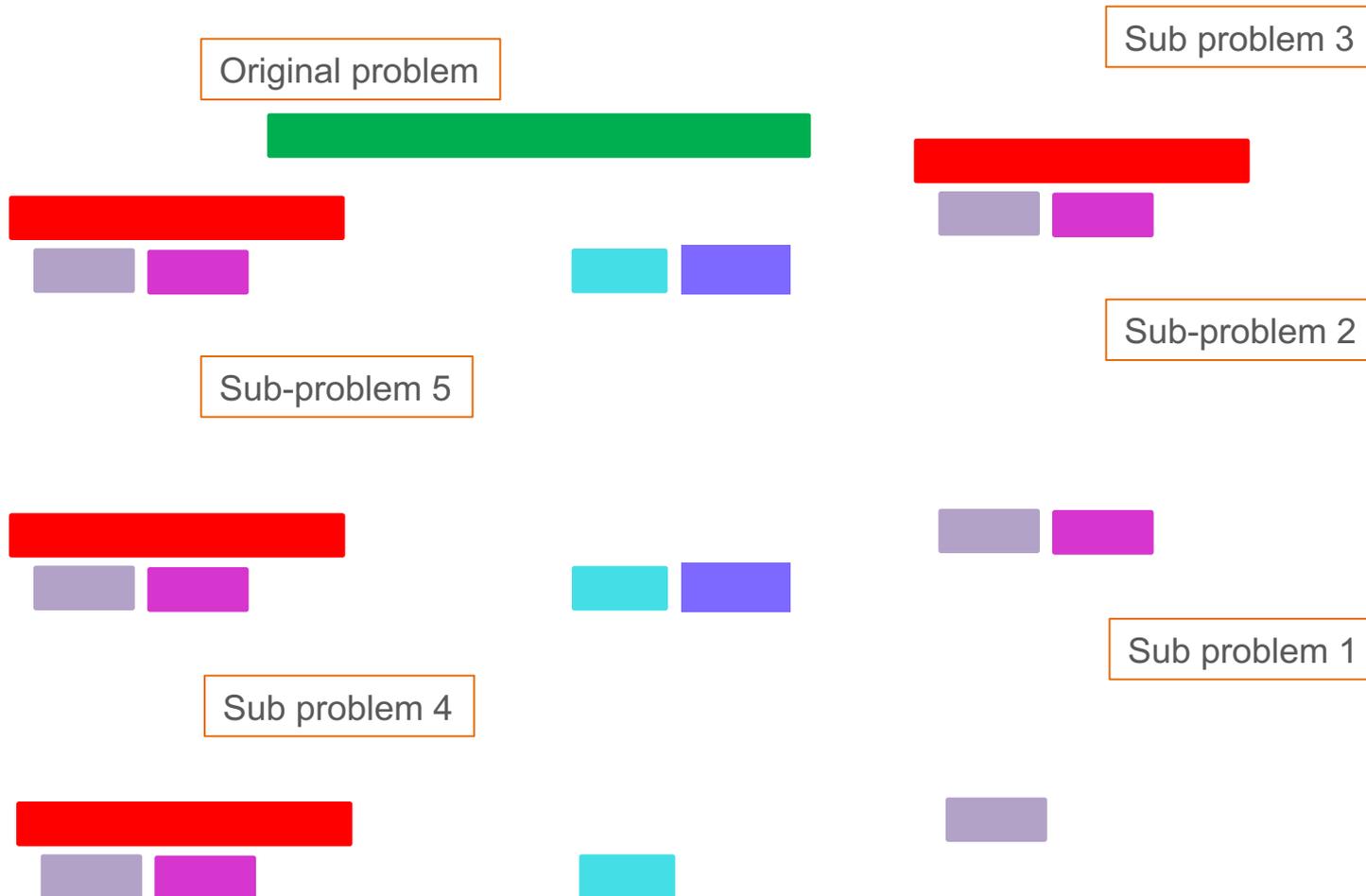




ICANHASCHEEZBURGER.COM 🍷 🍷

So what sub-problems?

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





ICANHASCHEEZBURGER.COM 🍷 🍷