Lecture 32

CSE 331

Nov 15, 2017

Comments on Feedback

note

stop following 3 views

Comments on feedback

Thanks for everyone who have feedback (@640). Over the course of this week, I will address/respond to some of the feedback (both the quantitative ones and the written comments).

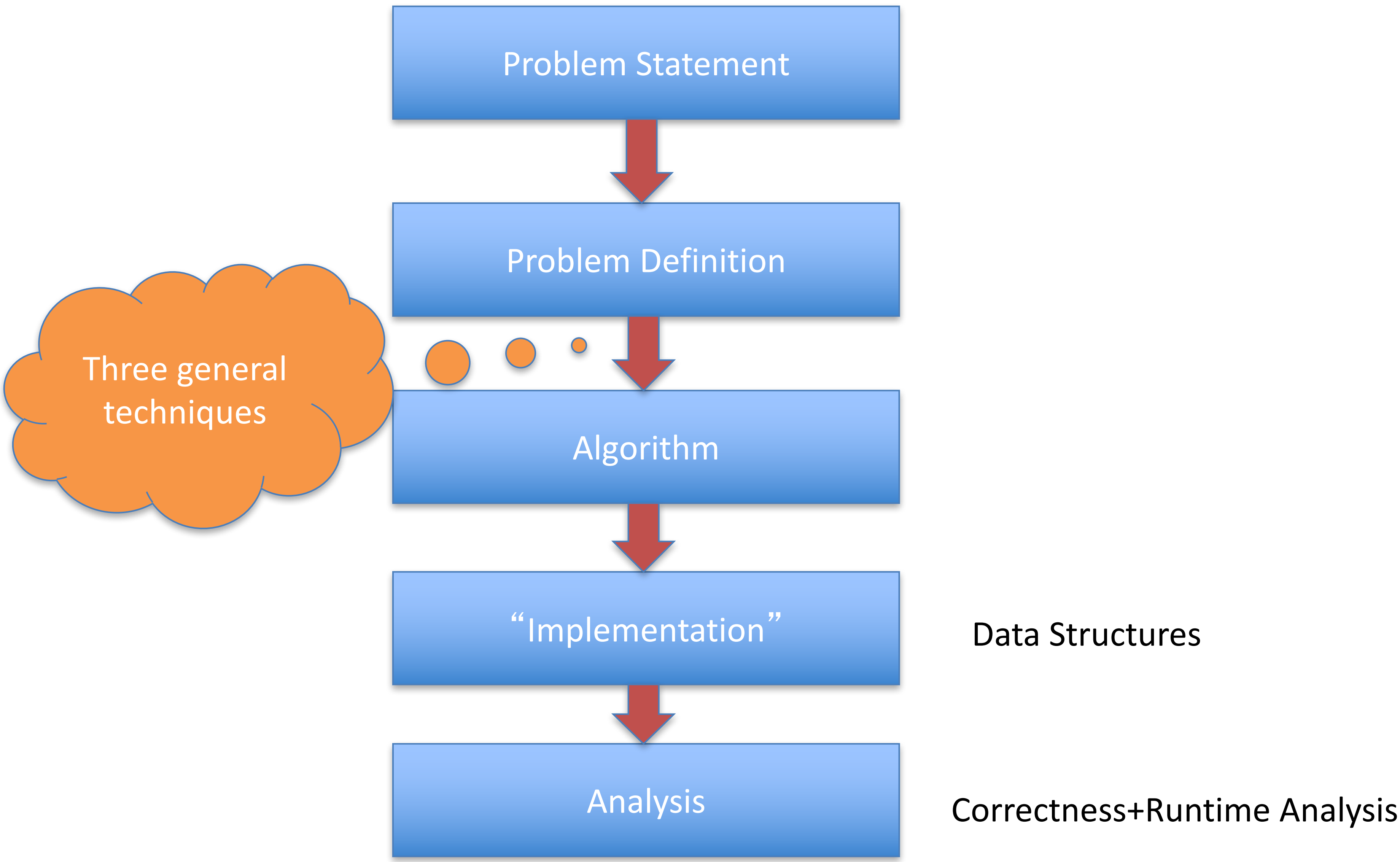
In some cases I will be able to incorporate your comments this year. For others, it might not be but I will at least present you my rationale for for why not.

I will start off with responses to how you felt about the class overall. In particular, I pay close attention to the fraction of students who say they are "challenged and unhappy." Last year this was around 17%: higher than what I would like but still something that I can potentially live with. However, this year's number are not good:

Overall your feeling about CSE 331

71 responses

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

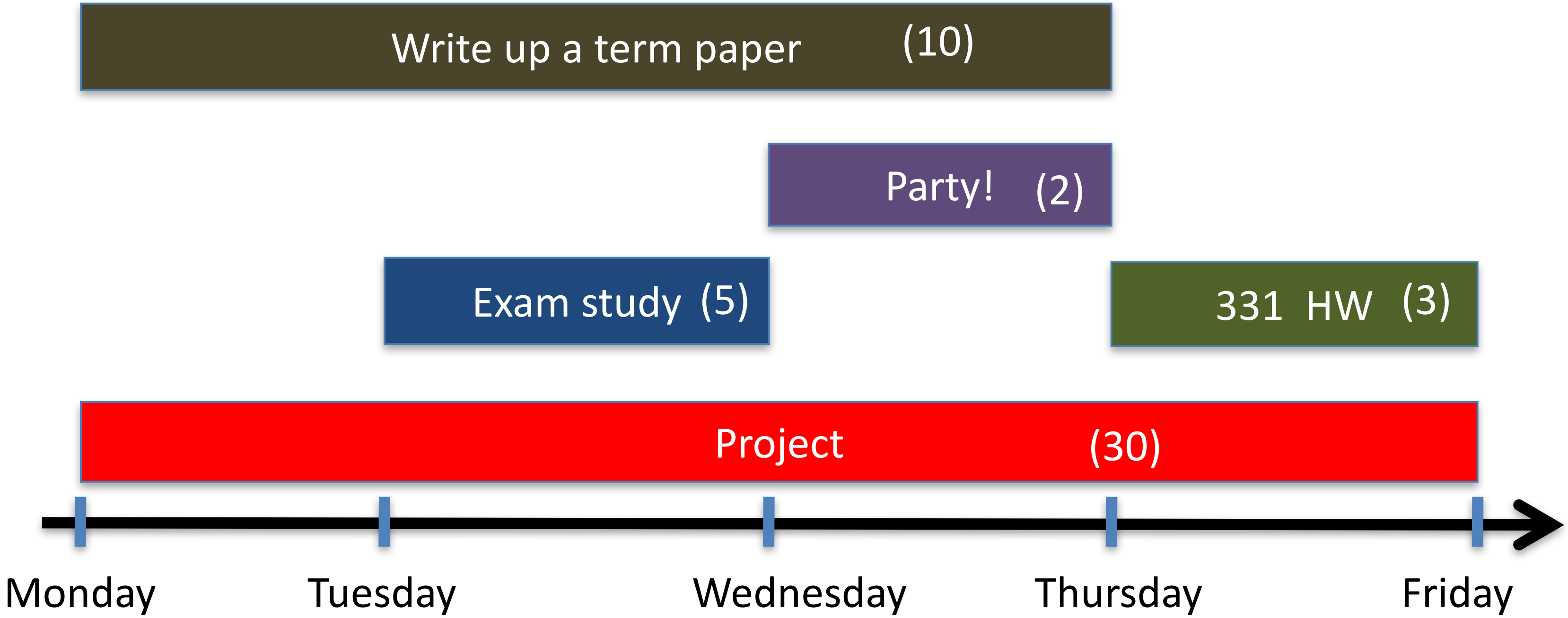
Both design recursive algorithms

Different because

Dynamic programming is smarter about solving recursive sub-problems

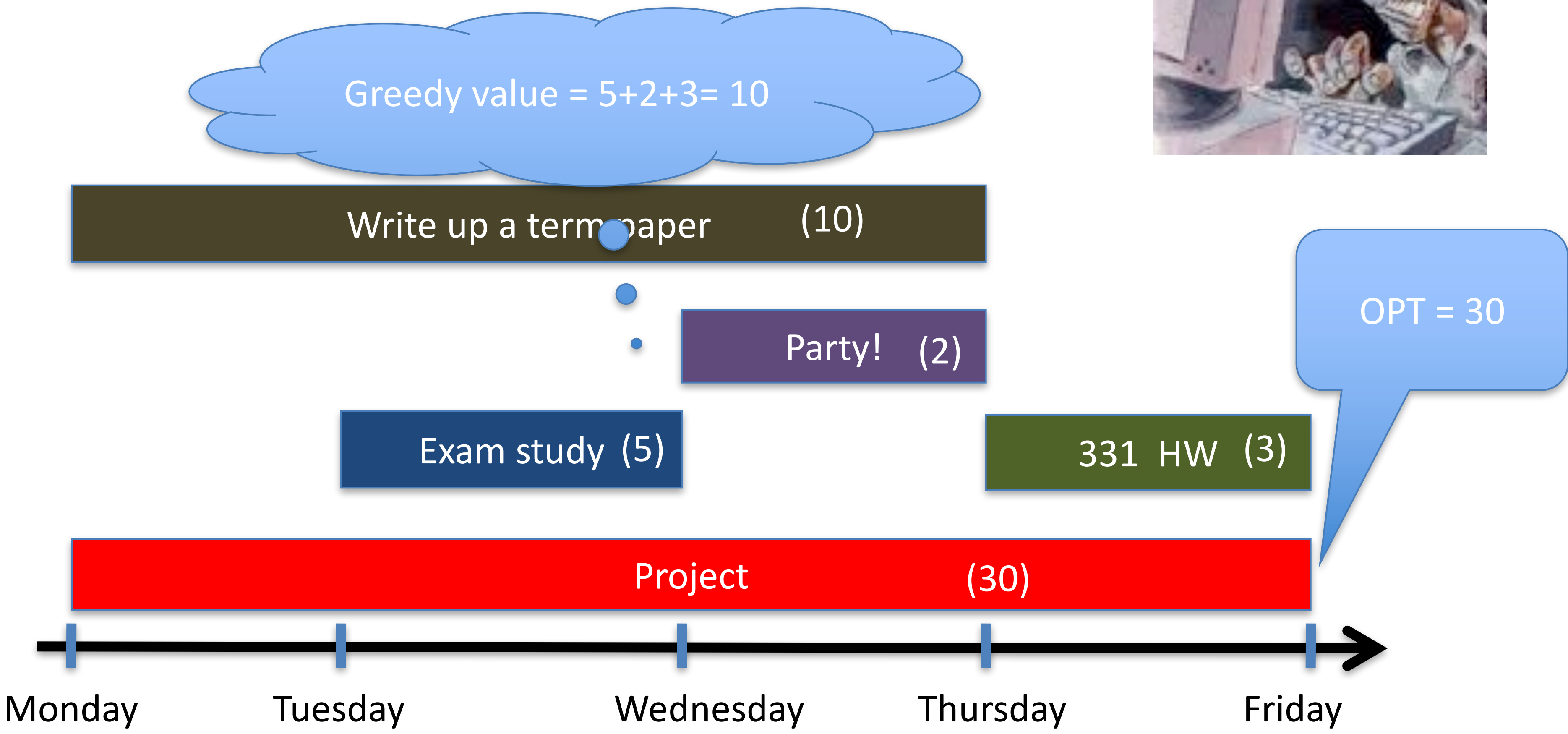
End of Semester blues

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



Previous Greedy algorithm

Order by end time and pick jobs greedily



Today’s agenda

Formal definition of the problem

Start designing a recursive algorithm for the problem

Property of OPT

j in OPT(j)

j not in OPT(j)

$$OPT(j) = \max \{ v_j + OPT(\ p(j)\ ),\ OPT(j-1) \}$$

Given OPT(1),…., OPT(j-1),
how can one figure out if j
in optimal solution or not?

A recursive algorithm

Proof of correctness by induction on j

Compute-Opt(j)

Correct for $j=0$

If $j = 0$ then return 0

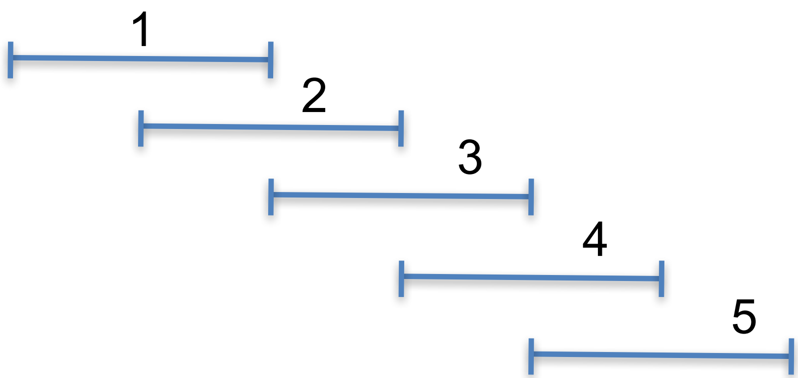
return max { v_j + Compute-Opt($p(j)$), Compute-Opt($j-1$) }

$= OPT(p(j))$

$= OPT(j-1)$

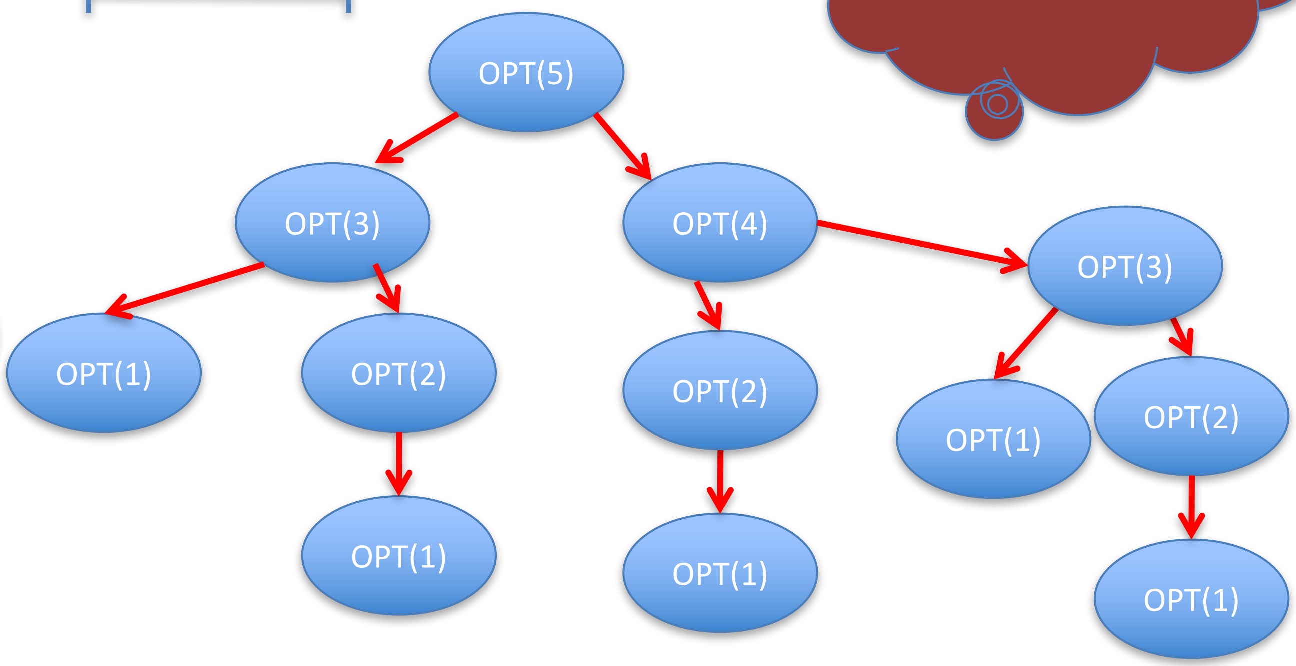
$$OPT(j) = \max \{ v_j + OPT(p(j)), OPT(j-1) \}$$

Exponential Running Time



p(j) = j-2

Only 5 OPT values!



Formal proof: Ex.

How many distinct OPT values?

A recursive algorithm

M-Compute-Opt(j)

If j = 0 then return 0

If M[j] is not null then return M[j]

M[j] = max { v_j + M-Compute-Opt(p(j)), M-Compute-Opt(j-1) }

return M[j]

M-Compute-Opt(j)
= OPT(j)

Run time = O(# recursive calls)

Bounding # recursions

M-Compute-Opt(j)

If j = 0 then return 0

If M[j] is not null then return M[j]

M[j] = max { v_j + M-Compute-Opt(p(j)), M-Compute-Opt(j-1) }

return M[j]

O(n) overall

Whenever a recursive call is made an M value of assigned

At most n values of M can be assigned

Reading Assignment

Sec 6.1, 6.2 of [KT]

When to use Dynamic Programming

Richard Bellman

There are polynomially many sub-problems

Optimal solution can be computed from solutions to sub-problems

There is an ordering among sub-problem that allows for iterative solution