Lecture 29

CSE 331

Nov 9, 2022

Homework 6 out

Homework 6

Due by **11:30pm, Tuesday, November 15, 2022**.

Make sure you follow all the homework policies.

All submissions should be done via Autolab.

Question 1 (Exponentiation) [50 points]

The Problem

We will consider the problem of exponentiating an integer to another. In particular, for non-negative integers a and n, define `Power` (a, n) be the number a^n. (For this problem assume that you can multiply two integers in $O(1)$ time.) Here are the two parts of the problem:

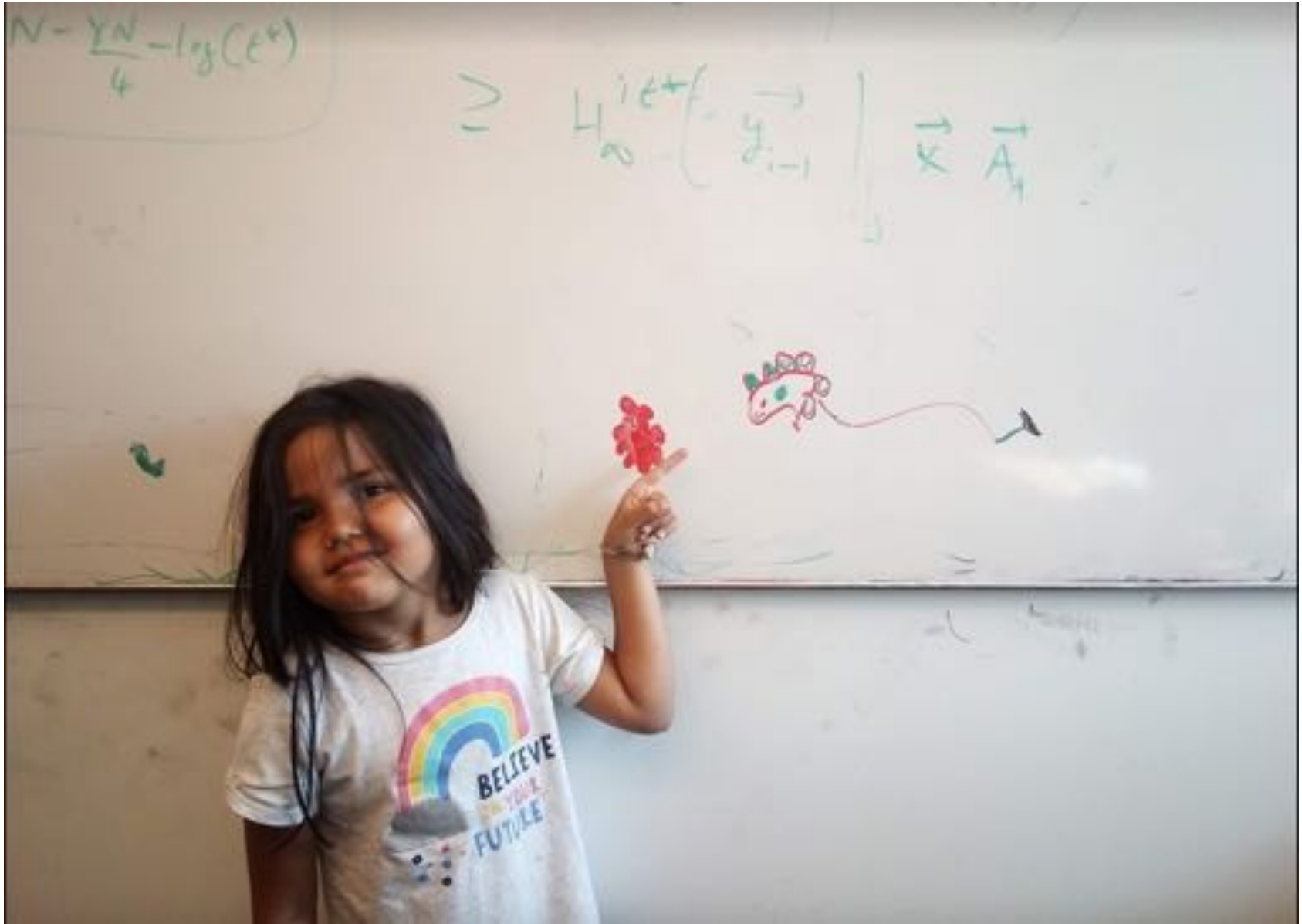
- Part **(a)**: Present a naive algorithm that given non-negative integers a and n computes `Power` (a, n) in time $O(n)$.

> **Note**
> For this part, there is no need to prove correctness of the naive algorithm but you do need a runtime analysis.

- Part **(b)**: Present a divide and conquer algorithm that given non-negative integers a and n computes `Power` (a, n) in $O(\log n)$ time.

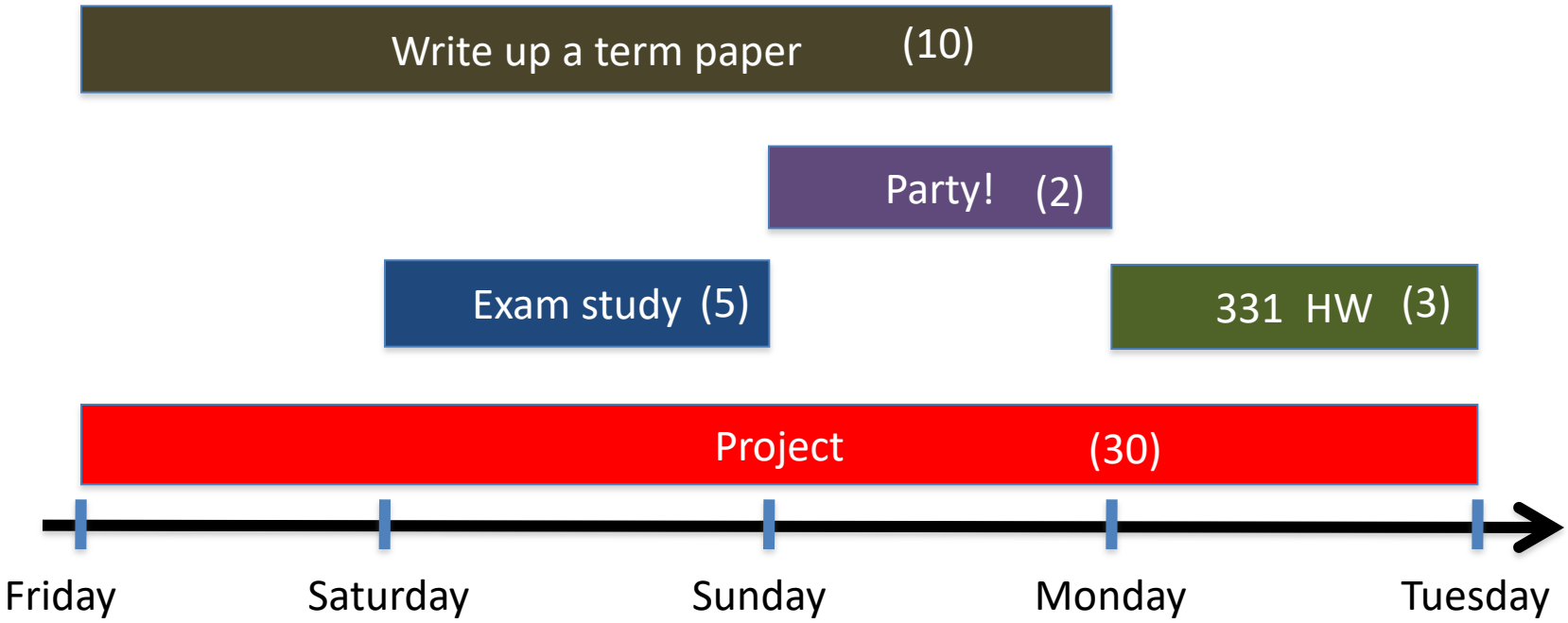
> **Important Note**
> **To get credit you must present a recursive divide and conquer algorithm and then analyze its running time by solving a recurrence relation. If you present an algorithm that is not a divide and conquer algorithm you will get a level 0 on this entire part.**

Questions/Comments?



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)

Friday Saturday Sunday Monday Tuesday

Weighted Interval Scheduling

Input: n jobs (s_i, f_i, v_i)

Output: A schedule S s.t. no two jobs in S have a conflict

Goal: max $\Sigma_{i \text{ in } S} v_j$

Assume: jobs are sorted by their finish time

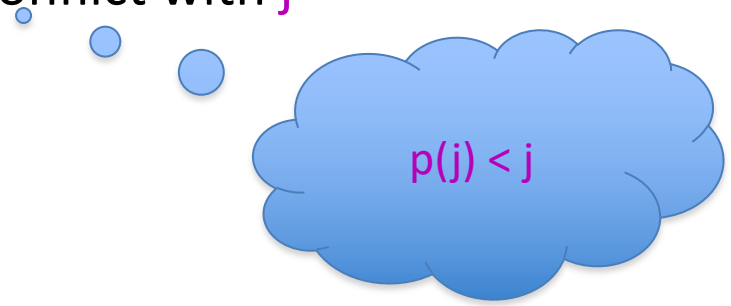
Today's agenda

Finish designing a recursive algorithm for the problem

Couple more definitions

p(j) = largest i < j s.t. i does not conflict with j

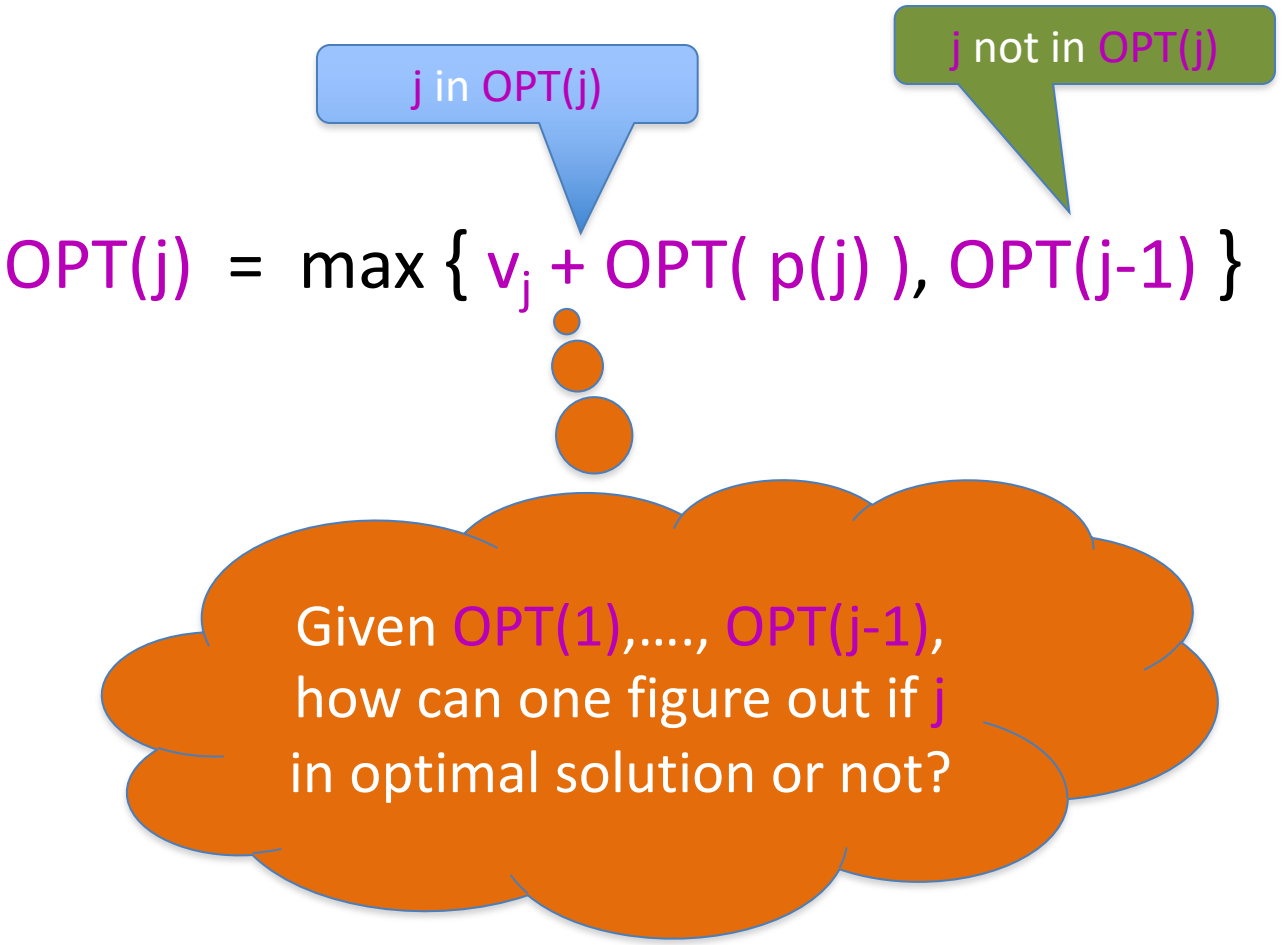
= 0 if no such i exists



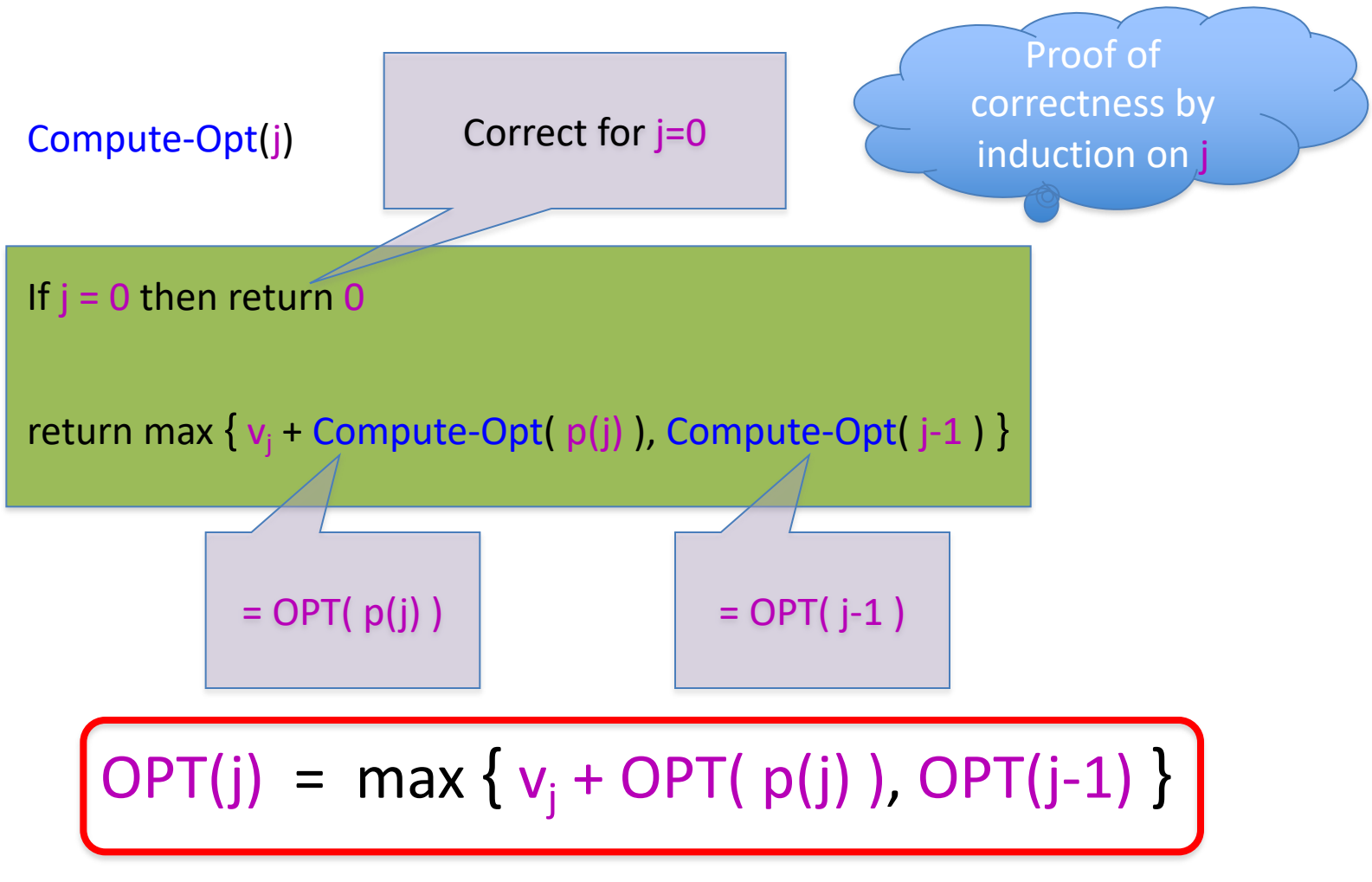
OPT(j) = optimal value on instance 1,..,j

Moving to the board…

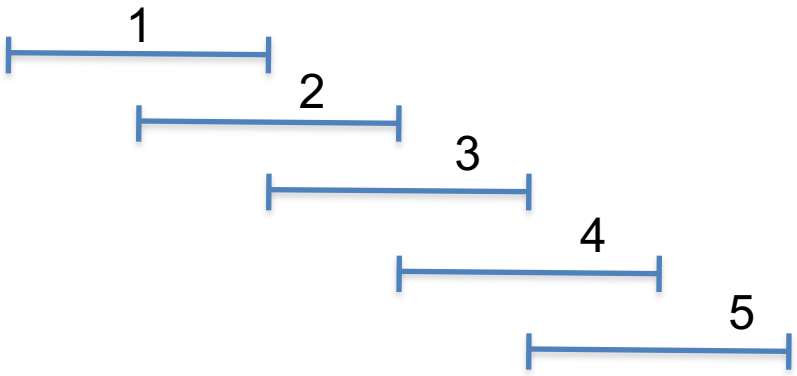
Property of OPT



A recursive algorithm



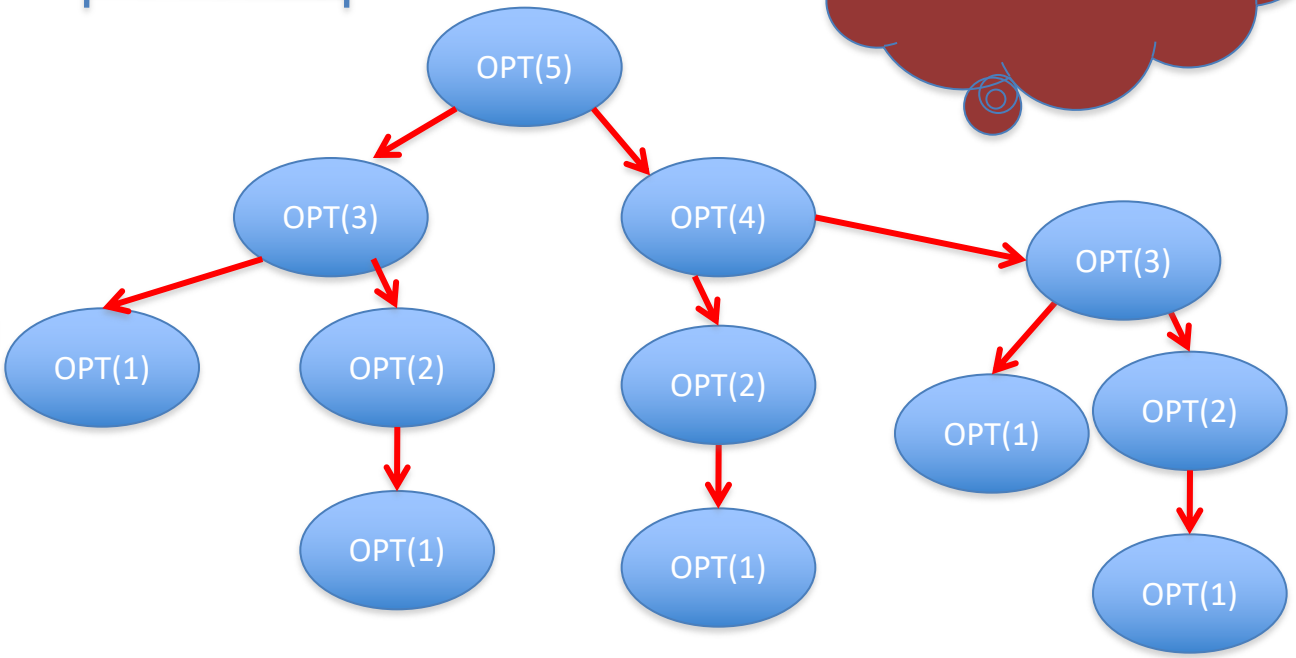
Exponential Running Time



p(j) = j-2

Only 5 OPT values!

Formal proof: Ex.



Using Memory to be smarter

Using more space can reduce runtime!

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)