

Lecture 30

CSE 331

Nov 8, 2019

HW 8 out

Homework 8

Due by **11:00am, Friday, November 15, 2019.**

! Note on Timeouts

For this problem the total timeout for Autolab is 480s, which is higher than the usual timeout of 180s in the earlier homeworks. So if your code takes a long time to run it'll take longer for you to get feedback on Autolab. **Please start early to avoid getting deadlocked out before the submission deadline.**

Also for this problem, `C++` and `Java` are way faster. The 480s timeout was chosen to accommodate the fact that Python is much slower than these two languages.

Question 1 (Finding a sink) [50 points]

The Problem

Given a directed graph $G = (V, E)$, a vertex $s \in V$ is called a **sink** if there are incoming edges from every other vertex to s but no outgoing edge from s , i.e. $|\{(u, s) \in E\}| = |V| - 1$ and $|\{(s, u) \in E\}| = 0$.

The goal of this problem is to design an algorithm to find out if G has a sink and if so, to output it. (Recall that $n = |V|$). Your algorithm is given G in its adjacency matrix A (i.e. if an ordered pair $(u, v) \in E$, then $A[u][v] = 1$ and if $(u, v) \notin E$, then $A[u][v] = 0$).

Sample Input/Output pairs

Here are two sample input/output pairs (input is the matrix, with vertex set $\{u, v, x, y, z\}$ and the rows (top to bottom) and column (from left to right) are in the order u, v, x, y, z) and the output is a vertex (if it is a sink) or `null` otherwise):

HW 7 solutions

At the end of the lecture

HW 6 grading

By Sunday morning

Video mini project grading

After Thanksgiving break

Coding Project is all live now!



note ★

stop following

8 views

All of Coding Project is now live!

As promised in @1143, Autolab is now accepting submissions for Problem 5 for the coding project as well. The coding project webpage has been updated with the required coding details:

<http://www-student.cse.buffalo.edu/~atri/cse331/fall19/coding-project/index.html>

Few things to keep in mind:

- As with problem 1 (@801), 2 (@980), 3 (@1114) and 4(@1143) this is group submission-- please see the webpage for instructions on how to do so. *Please follow the instructions EXACTLY. Not following the instructions might make the group submission on Autolab not behave as intended.*
- You have to form your group **AGAIN** for Problem 5 on Autolab-- it unfortunately does not carry over from Problem 1 or Problem 2 or Problem 3 or Problem 4.
- **Please download the zip for Problem 5 and use that for Problem 5 submission.** In particular, do NOT use the zip for Problem 1 or 2 or 3 or 4 for Problem 5.
- Now for some points that are not repeated from the earlier post:
 - In case you have not noticed already, HW6 Q3 *might* be useful for this problem (on all of Problems 2 to 5)
 - The requirement for Problem 5 is **much** more stringent from the previous problems and I would recommend that you do not leave it to the very last moment.

If you have questions, please post on piazza! Or go to office hours. Have fun :-)

#pin

[coding_mini_project](#)

Problem 2 and 3 due at **11am, Friday, November 22, 2019.**

Problem 3 now due at **11am, Tuesday, November 26, 2019.**

Problems 4 and 5 due at **11am, Friday, December 6, 2019.**

edit

good note

Atri Rudra



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

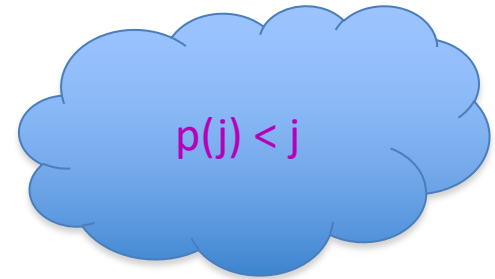
Goal: $\max \sum_{i \in S} v_j$

Assume: jobs are sorted by their finish time

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 $1, \dots, j$

Property of OPT

j in $\text{OPT}(j)$

j not in $\text{OPT}(j)$

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

Given $\text{OPT}(1), \dots, \text{OPT}(j-1)$,
how can one figure out if j
in optimal solution or not?

A recursive algorithm

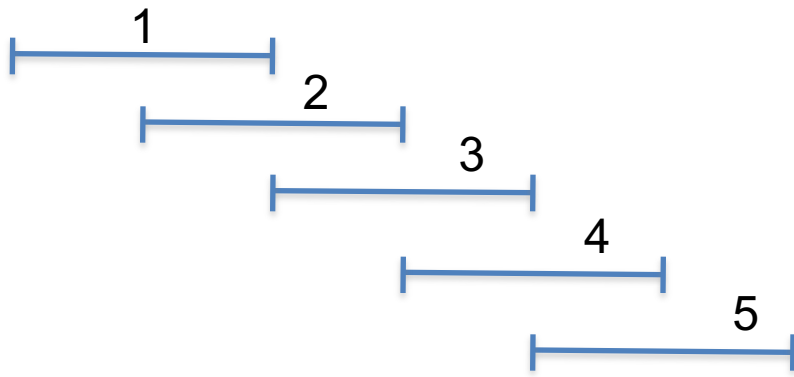
Compute-Opt(j)

If $j = 0$ then return 0

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

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

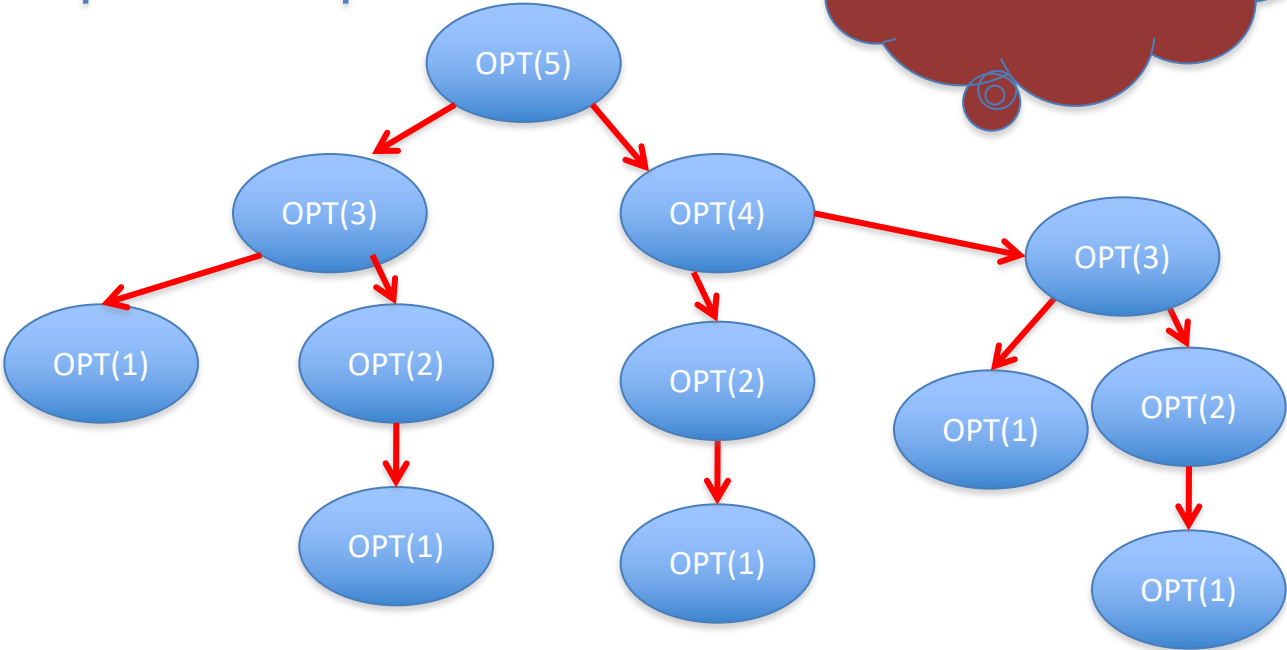
Exponential Running Time



$$p(j) = j-2$$

Only 5 OPT values!

Formal proof: Ex.



A recursive algorithm

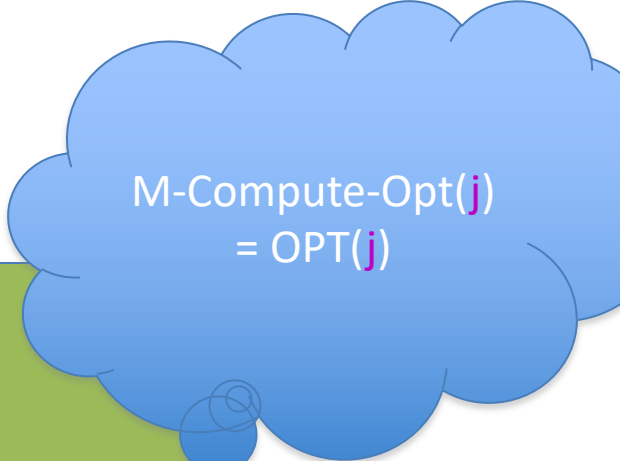
M-Compute-Opt(j)

If $j = 0$ then return 0

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

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

return $M[j]$



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

Run time = $O(\# \text{ recursive calls})$

Bounding # recursions

M-Compute-Opt(j)

If $j = 0$ then return 0

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

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

return $M[j]$

$O(n)$ overall

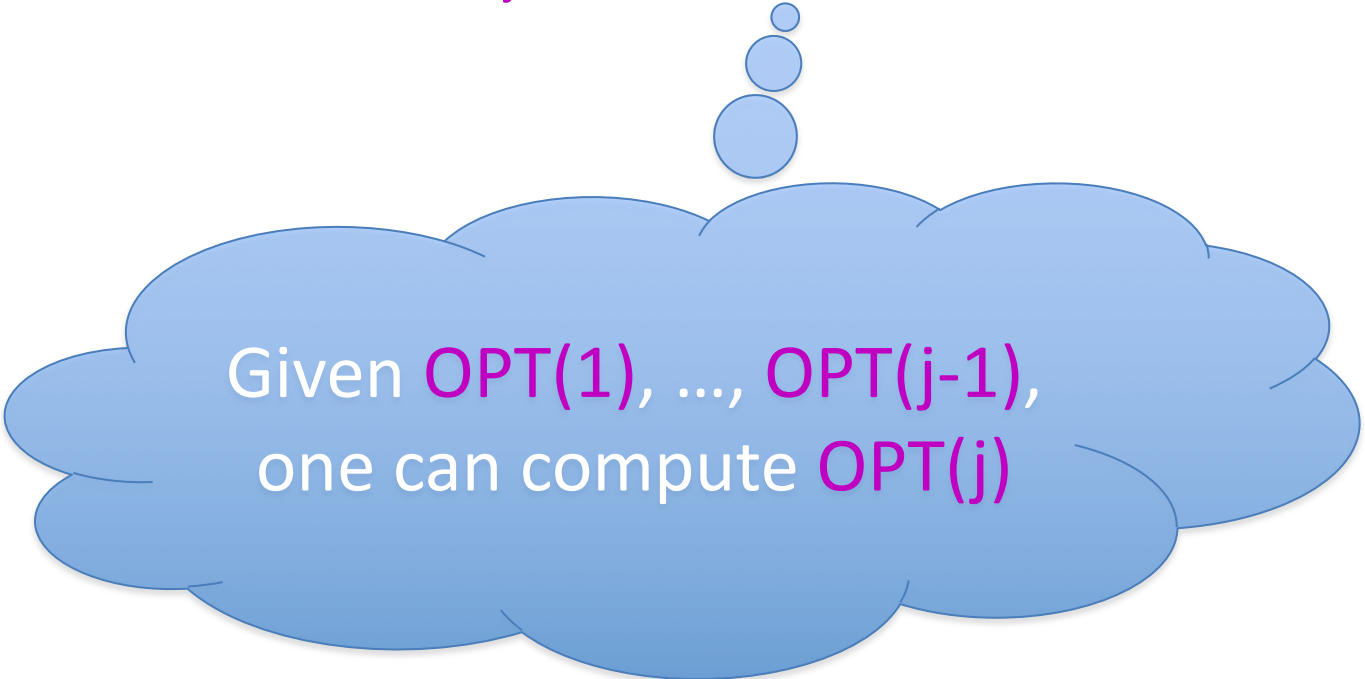
Whenever a recursive call is made an M value is assigned

At most n values of M can be assigned



Property of OPT

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



Given $\text{OPT}(1), \dots, \text{OPT}(j-1)$,
one can compute $\text{OPT}(j)$

Recursion+ memory = Iteration

Iteratively compute the OPT(j) values

Iterative-Compute-Opt

$M[0] = 0$

For $j=1, \dots, n$

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

$M[j] = \text{OPT}(j)$

$O(n)$ run time



ICANHASCHEEZBURGER.COM 🍷 🍷

Reading Assignment

Sec 6.1, 6.2 of [KT]



When to use Dynamic Programming

There are polynomially many sub-problems

$$\text{OPT}(1), \dots, \text{OPT}(n)$$

Optimal solution can be computed from solutions to sub-problems

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

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

$$\text{OPT}(j) \text{ only depends on } \text{OPT}(j-1), \dots, \text{OPT}(1)$$



Richard Bellman

Scheduling to min idle cycles

n jobs, i^{th} job takes w_i cycles

You have W cycles on the cloud



What is the maximum number of cycles you can schedule?

Rest of today's agenda

Dynamic Program for Subset Sum problem

May the Bellman force be with you

