Lecture 33

CSE 331

Nov 18, 2024

Apply to be a CSE 331 TA in 2025!

note @295 stop following 1 view

Actions

Want to be a UTA for 331 in 2025?

Profs. Bosse and Hayes be teaching 331 in the upcoming Spring semester and is looking for UTAs. I expect to be teaching 331 again in Fall 2025 (though this is **not** finalized and is subject to change) and will be looking for TAs then as well. So Profs. Bosse, Hayes and I are looking to jointly interviewing candidates for CSE 331 TAs for 2025 (on **zoom** tentatively the final week (Dec 18 and after) and/or the week after that (week of Dec 23, 2024).

(As an aside: I also have openings for doing research but I'll post on those once I'm done with all 331 related stuff: i.e. after the grades have been submitted.)

These will be *paid* positions. Time-commitment wise here is what we're looking for

- *Ideally,* you should be able to commit close to 10 hours/week on average. More is of course better!
- Depending on your background (e.g. if you have TAed before), we're willing to be OK with ~5 hours/week on average but no lower than that (and no more than 1-2 TAs with << 10 hrs/week).

A few important points:

- There is *no* formal minimum grade requirement to be a 331 UTA (Of course you don't know your grade by now). For now, we're basically looking for interested students who enjoyed 331 so far and would be excited to help others.
- A large fraction of your current TAs will be TAing CSE 331 this spring (but pretty much all of them will be gone by the summer) but the SP 25 class will be about twice as large this semester so SP 25 will have many more openings (15-20) as compared to Fall 25 (10+).
- Being a 331 UTA is definitely a great experience (feel free to ask one of your TAs!) and also **a great preparation for your interviews – there is no better way to learn algorithms than to teach it!**
- The application process is basically you presenting an algorithm that is covered in class to a "mock recitation"-- once you apply, we will provide more details on the process.

Reflection 2 has been graded

note @304 stop following 28 views Actions

Reflection 2 has been graded

Reflection problem 2 has now been graded and the scores and feedback released on Autolab! Hopefully the feedback is helpful as y'all work on your Reflection Problem 3.

PLEASE READ THE RUBRIC CAREFULLY TO SEE THE COMMON MISTAKES, which y'all should avoid making for future reflection problem submissions. *Also take a look at the feedback since in many cases I did not deduct points since this was the first submission where we have asked you to think about implications of your algorithm-- the grading would be stricter reflection 3 onwards.*

Few common mistakes:

- *In argument for in favor vs not: not clearly stating why your algo idea leads to the claimed group being favored vs. not*
- *Not explicitly taking the routing algorithm in your argument.*
  - *Just because your algorithm picked certain paths in certain priority, it does not mean that has to play "nice" with the routing algo-- so your argument should explicitly state why you expect (at least a high level) the routing algo to not completely undo the priority in your algo.*
- *The algo idea not matching the code that was submitted.*
- *More generally, your document should be self-contained. So if you state your algo generates path that prioritizes certain types of clients, you should state* explicitly how *your algo implements that.*

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

Questions?

Subset sum problem

Input: n integers $w_1, w_2, \ldots, w_n$

bound W

Output: subset S of $[n]$ such that

(1) sum of w_i for all i in S is at most W

(2) $w(S)$ is maximized

Recursive formula

$OPT(j, B)$ = max value out of $w_1,..,w_j$ with bound B

If $w_j > B$

$OPT(j, B) = OPT(j-1, B)$

else

$OPT(j, B) = \max \{ OPT(j-1, B), w_j + OPT(j-1, B-w_j) \}$

Questions?

Runtime analysis on the board…

Recursive formula

$OPT(j, B)$ = max value out of $w_1,..,w_j$ with bound B

If $w_j > B$

$OPT(j, B) = OPT(j-1, B)$

else

$OPT(j, B) = \max \{ OPT(j-1, B), w_j + OPT(j-1, B-w_j) \}$

Can compute final S with recursion/ backtracking

j not in OPT

j in OPT

Knapsack problem

Input: n pairs $(w_1, w_1), (w_2, w_2), \ldots, (w_n, w_n)$,

bound W

Output: subset S of $[n]$ such that

(1) sum of w_i for all i in S is at most W

(2) $w(S)$ is maximized

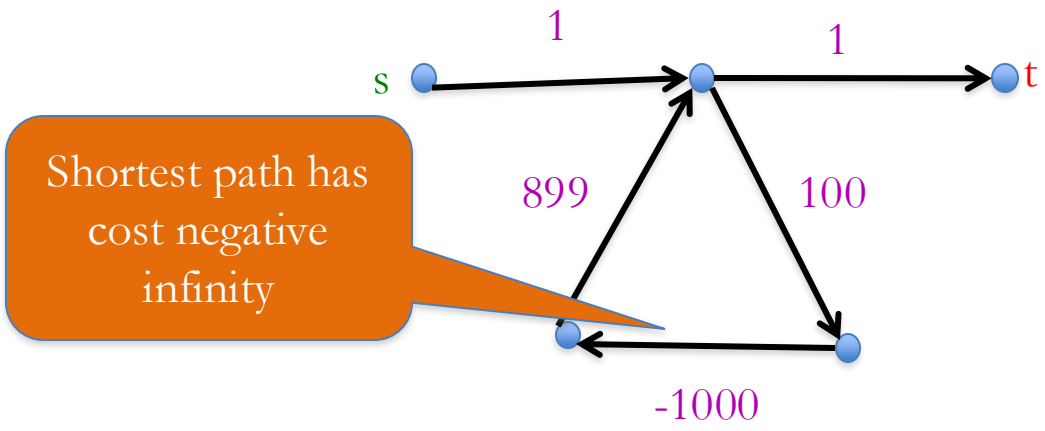
Questions?

Shortest Path Problem

Input: (Directed) Graph G=(V,E) and for every edge e has a cost c_e (can be <0)

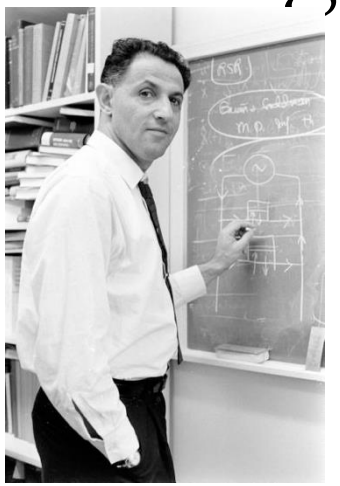
t in V

Output: Shortest path from every s to t



Assume that G has no negative cycle

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

Rest of today's agenda

Bellman-Ford algorithm