Lecture 32

CSE 331

Nov 13, 2019

Suggestions on coding project

note

stop following 84 views

Comments on coding mini-project

I had few of you ask some questions about the coding mini-project in my office hour today so I figured I should mention my answers in case it is useful to others:

- In case you were wondering about the ethical angle in Problem 2-4 in the coding project: the short answer is that there is none. In fact, problems 2-4 ask you to behave unethically :-) Problem 5 is where you are supposed to do the right thing.
- There is no "right" solutions for Problem 2-5. When we were trying to design our solutions, we discussed few heuristics and then saw what worked well. So as you work with you group, here are some recommendations to keep in mind--
  - For problem 2-4 you are trying to maximize the overall revenue so try and think of heuristics that would allow you to get revenue from "most" clients.
  - You should implement more than one heuristic and then try out and see what works well and what does not.
  - While Problem 3 and 4 build on problem 2 for its definition, y'all need not work on them sequentially-- go ahead and try all your heuristics on all problems and see what works and what does not. Then go back and revise your heuristics and iterate this process.
  - The grading scheme is at the very bottom on the coding mini-project page.
  - Even though Problem 2 is worth twice the points in Problem 1, the actual solution will take much more work than Problem 1 solution (which basically was to call the BFS function provided with the template).

#pin

office_hours coding_mini_project

edit · good note | 1

Updated 22 hours ago by Atri Rudra

Give feedback!

note

stop following | 19 views

Feedback on CSE 331

I generally try to get feedback earlier in the semester but for various reasons this got pushed back this fall. Anyhow, please do give feedback via this anonymous Gform:

https://docs.google.com/forms/d/e/1FAIpQLSce1q7pg7TYdl1jA7g-SKbizWX1OOQZ4C2HUKr0bU8cN_zuxA/viewform?usp=sf_link

Filling in this form is **completely optional and anonymous**.

In particular, I would love feedback (even if it is critical). Many of the aspects of CSE 331 that you like were suggested by someone in a previous incarnation of CSE 331. While I'll try and incorporate as much as I can this fall, some of your suggestions might have to wait for the next offering.

I might also dis-agree with your feedback but after a week or so, I'll post my response to the feedback from y'all. So at the very least y'all would get to hear my reasoning for why certain things are the way they are in CSE 331. And then we can agree to disagree :-)
#pin

feedback

~ An instructor (Sanchit Batra) thinks this is a good note ~

edit · good note | 1

Updated 15 minutes ago by Atri Rudra

Today’s OH– 2:50 to 3:40pm

note ☆ **35** views

Atri OH will start 10 mins early tomorrow (Wed) and next week Wed

Sorry to do this again but just for **this week AND next week**, my **Wed** office hours will move up by 10 mins. I.e. it'll be from **2:50-3:40pm.** The 3:40pm is a hard stop and I apologize in advance if this causes any inconvenience.
#pin

office_hours

edit · good note | 0 Updated Just now by Atri Rudra

Same change next week as well

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

Can compute final S with recursion/ backtracking

If $w_j > B$

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

else

j not in OPT

j in OPT

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

Knapsack problem

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

n pairs $(w_1, v_1), \ldots, (w_n, v_n)$,

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

(2) $v(S)$ is maximized

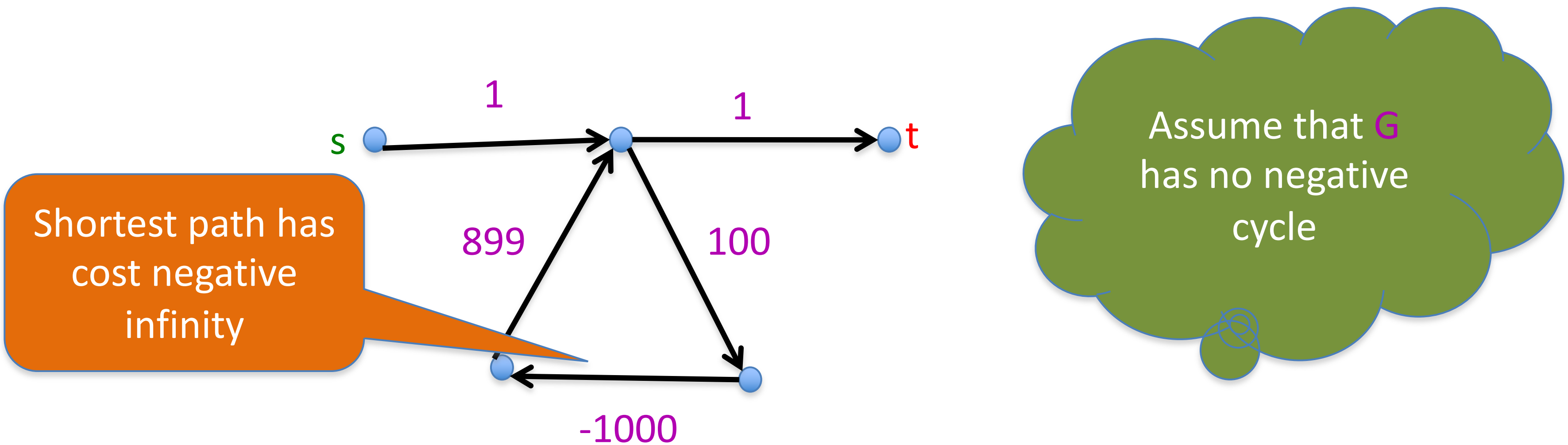
Questions?

Shortest Path Problem

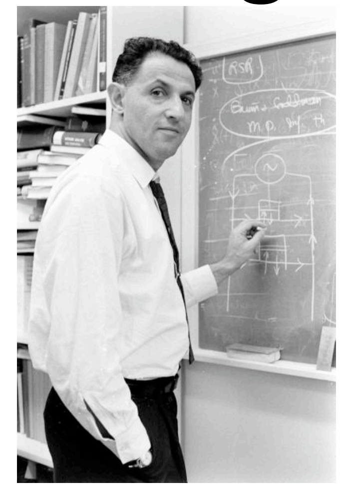
Input: (Directed) Graph G=(V,E) and for every edge e has a cost c_e (can be <0)

t in V

Output: Shortest path from every s to t



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

Today’s agenda

Bellman-Ford algorithm