Lecture 32

CSE 331

Nov 17, 2021

Please have a face mask on

Masking requirement

UB requires all students, employees and visitors – regardless of their vaccination status – to wear face coverings while inside campus buildings.

https://www.buffalo.edu/coronavirus/health-and-safety/health-safety-guidelines.html

HW 7 out

Homework 7

Due by **8:00am, Wednesday, December 1, 2021**.

Make sure you follow all the homework policies.

All submissions should be done via Autolab.

Question 1 (Ex 2 in Chap 6) [50 points]

The Problem

Exercise 2 in Chapter 6. The part `(a)` and `(b)` for this problem correspond to the part **(a)** and part **(b)** in Exercise 2 in Chapter 6 in the textbook.

Sample Input/Output

See the textbook for a sample input and the corresponding optimal output solution.

! Note on Timeouts

For this problem the total timeout for Autolab is 480s, which is higher the 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.

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?

Algo run on the board…

Recursive formula

$OPT(j, B)$ = max value out of $w_1,\ldots,w_j$ 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

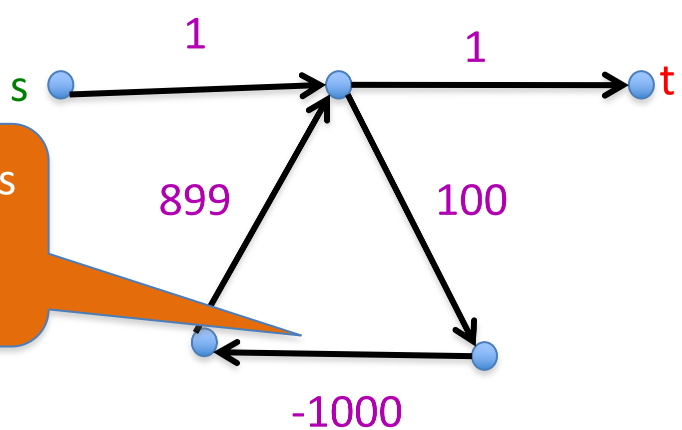
Questions?

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

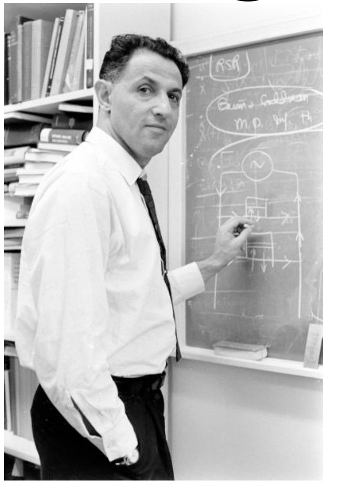
Output: Shortest path from every s to t



Shortest path has cost negative infinity

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