Lecture 32

CSE 331

Subset sum problem

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

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

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

Algo run on the board…

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

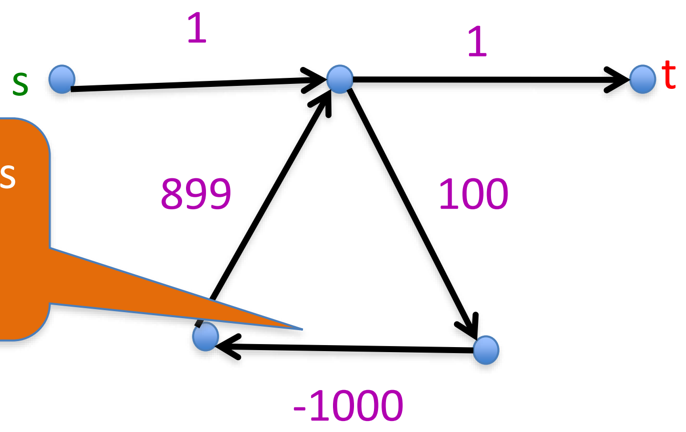
(2) $v(S)$ is maximized

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

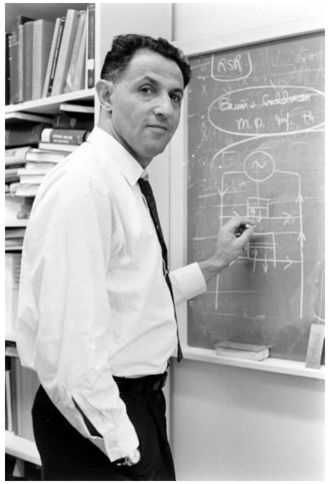
Assume that G has no negative cycle

When to use Dynamic Programming

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



Richard Bellman