Subset-Sum$(w_1, \dots, w_n; W)$

0. Allocate a $(W+1)\times(n+1)$ matrix M ; $O(nW)$
1. $M[B,0] \leftarrow 0 \quad \forall\ 0 \le B \le W$; $O(W)$
2. for $j = 1 \cdots n$ — $O(nW)$

    for $B = 0 \cdots W$ — $O(1)$

        if $w_j > B$

            $M[B,j] \leftarrow M[B, j-1]$

        else $M[B,j] \leftarrow \max\{w_j + M[B-w_j, j-1], M[B, j-1]\}$

3. return $M[W,n]$

overall: $O(nW) \leftarrow$ pseudopolynomial

$O(nW)$ $\uparrow$ poly-time if $W = \text{poly}(n)$

Apr 26

Example: $n=3$; $w_1 = 1,\ w_2 = 2,\ w_3 = 2$; $W = 3$

$B \backslash j$	0	1	2	3
3	0	1	3	3
2	0	1	2	2
1	0	1	1	1
0	0	0	0	0

$\underset{B,j}{M[1,1]} \leftarrow \max\{w_1 + M[1-1, 1-1], M[1, 1-1]\}$
$= \max\{1+0, 0\} = 1$

$M[2,1] \leftarrow \max\{w_1 + M[2-1, 1-1], M[2, 1-1]\}$
$= \max\{1+0, 0\} = 1$

$M[3,1] \leftarrow \max\{w_1 + M[3-1, 1-1], M[3, 1-1]\} = \max\{1+0, 0\} = 1$

$M[1,2] \leftarrow$ ~~____~~, $2 = w_2 > B = 1 \Rightarrow M[1,2] \leftarrow M[1, 2-1]$

$M[2,2] \leftarrow \max\{w_2 + M[2-2, 2-1], M[2, 2-1]\}$
$= \max\{2+0, 0\} = 2$

$M[3,2] \leftarrow \max\{w_2 + M[3-2, 2-1], M[3, 2-1]\}$
$= \max\{2+1, 1\} = 3$

$M[1,3] \leftarrow 2 = w_3 > B = 1 \Rightarrow M[1, 3-1]$

$M[2,3] \leftarrow \max\{w_3 + M[2-2, 3-1], M[2, 3-1]\}$

$= \max\{2+0, 2\} = 2$

$M[3,3] \leftarrow \max\{w_3 + M[3-2, 3-1], M[3, 3-1]\}$

$= \max\{2+1, 3\} = 3$

Subset Sum Input size: $n + \log W$

If $W = 2^n$ " : $\Theta(n)$; runtime: $(n \cdot 2^n)$

Shortest Path Problem

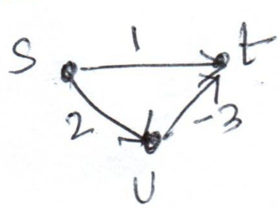
Input: (•) Directed graph $G = (V, E)$

$\forall e \in E$ c_e (Note: $c_e < 0$ allowed)

(•) $t \in V$ | (*) No negative cycles allowed

Output: $\forall s \in V$ a shortest s-t path.

Attempt 1: Run Dijkstra's on each $s \in V$.

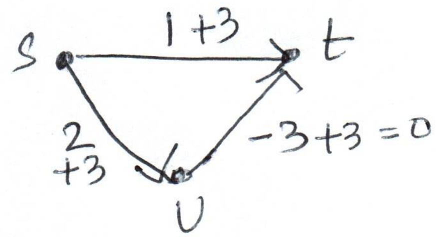


$\leftarrow$ shortest s-t path: s, u, t

Dijkstra's algo will output: s, t

$\hookrightarrow$ Doesn't work.

Attempt 2: Add some $\Delta > 0$ to all edges so that the new costs ≥ 0.



$\Delta = 3$

← shortest path with new costs: s, t

$\Rightarrow$ Reduction doesn't work

No known greedy/Divide & conquer algo for this

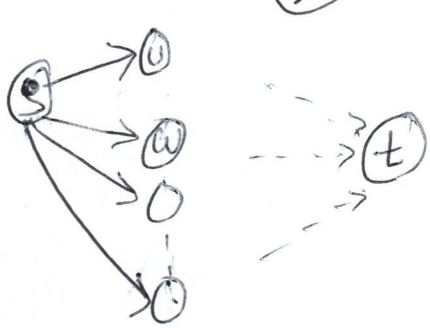
ASSUME: only interested (for ~~me~~ now) in the cost of a shortest s-t path.

Goal: Design a dynamic program

Attempt 3: ① OPT(s) = cost of a shortest s-t path.

If a shortest s-t path uses the ② edge (s, u), then

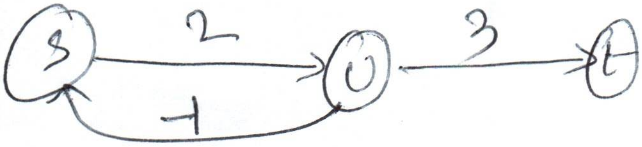
$$OPT(s) = c_{s,u} + OPT(u)$$



In general,

$$OPT(s) = \min_{w:\ (s,w) \in E} \{ c_{s,w} + OPT(w) \}$$

③ An ordering among subproblems:



$OPT(s) = 2 + OPT(u)$

$OPT(u) = \min\{3 + OPT(t), -1 + OPT(s)\}$

ISSUE: OPT(s) depends on OPT(u)
OPT(u) " " OPT(s)