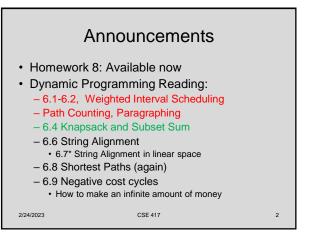
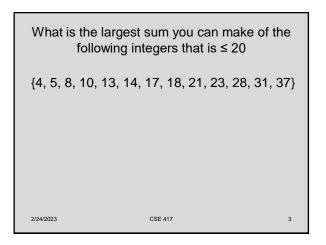
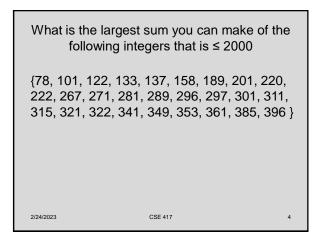
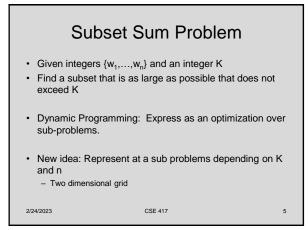
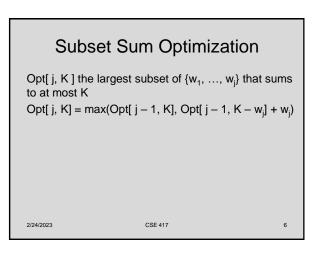
	CSE 417 Algorithms	
	Lecture 20, Winter 2023 Dynamic Programming Subset Sum etc.	
2/24/2023	CSE 417	1

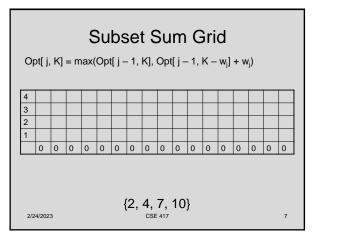




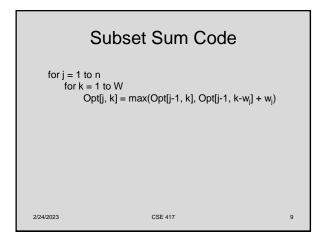


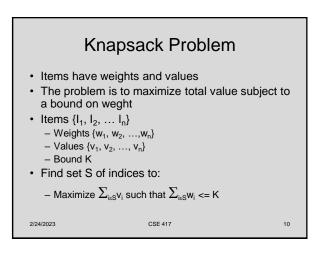


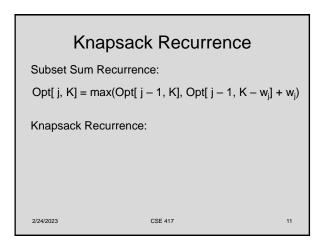


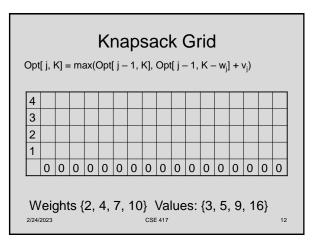


	1	-	-				-	-									
4	0	2	2	4	4	6	7	7	9	10	11	12	13	14	14	16	17
3	0	2	2	4	4	6	7	7	9	9	11	11	13	13	13	13	13
2	0	2	2	4	4	6	6	6	6	6	6	6	6	6	6	6	6
1	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
						4	[2,	4,	7,	10]	}						





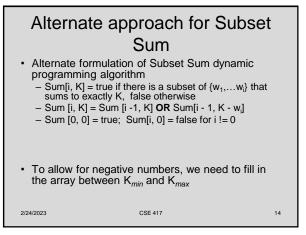




## Knapsack Grid

 $Opt[j, K] = max(Opt[j - 1, K], Opt[j - 1, K - w_j] + v_j)$ 

	17	17	47			16	16	12	9	9	8	5	5	3	3	0	4
			17	17	14	14	12	12	9	9	8	5	5	3	3	0	3
8	8	8	8	8	8	8	8	8	8	8	8	5	5	3	3	0	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	



## Run time for Subset Sum With n items and target sum K, the run time is O(nK) If K is 1,000,000,000,000,000,000,000,000 this is very slow Alternate brute force algorithm: examine all subsets: O(n2<sup>n</sup>) Point of confusion: Subset sum is NP Complete

2/24/2023

CSE 417

15

2/24/2023

 Two dimensional dynamic programming

 Subset sum and knapsack

 Opt[ j, K] = max(Opt[ j - 1, K], Opt[ j - 1, K - w<sub>j</sub>] + w<sub>j</sub>)

 Opt[ j, K] = max(Opt[ j - 1, K], Opt[ j - 1, K - w<sub>j</sub>] + v<sub>j</sub>)

 4

 0
 0
 0
 0
 0
 0
 0

 2
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0</

CSE 417

