DESIGN & ANALYSIS OF ALGORITHM

PCC-CS501

DESIGN & ANALYSIS OF ALGORITHM SCHEDULE ----TOPIC WISE

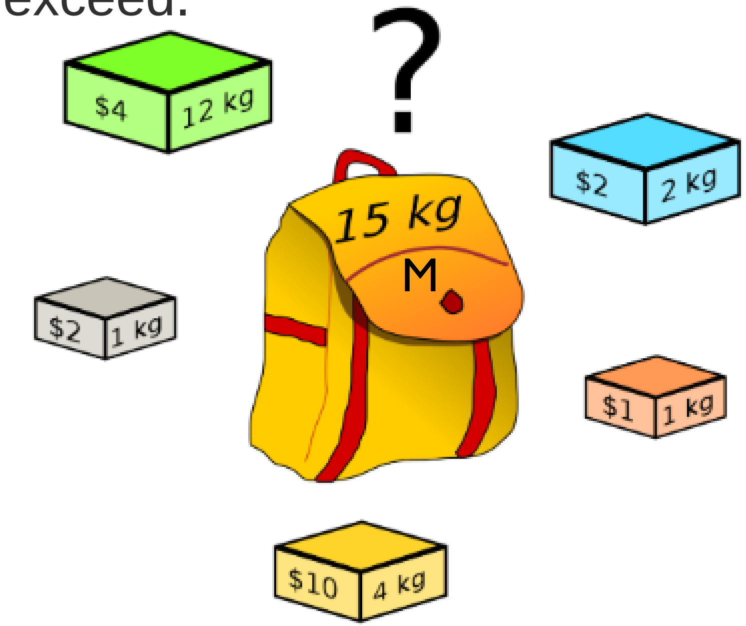
	Topic	Sub Topic
1	INTRODUCTION	DESIGN OF ALGORITHM ,ANALYSIS OF ALGORITHM, ALGORITHM PROPERTIES
2	FRAMEWORK FOR ALGORITHM ANALYSIS	HOW TO COUNT EXECUTION TIME OF ALGORITHM,INPUT INSTANCES
3	ASYMPTOTIC NOTATION	BEST CASE,AVERAGE CASE, WORST CASE
4	SOLVING RECURRENCE RELATION	SUBSTITUTION METHOD, MASTER THEOREM
5	ALGORITHM DESIGN TECHNIQUES	DIVIDE & CONQUER, GREEDY,DYNAMIC PROGRAMMING, BACKTRACKING,
6	DISJOINT SET MANIPULATION	UNION FIND
7	NETWORK FLOW PROBLEM	FORD FULKERSON ALGORITHM
8	NP COMPLETENESS	NP,NP HARD………ALGORITHM
9	APPROXIMATION ALGORITHM	COMPLEXITY ANALYSIS OF NP COMPETE PROBLEM

0/1 KNAPSACK PROBLEM

- The value or profit obtained by putting the items into the knapsack is maximum.
- And the weight limit of the knapsack does not exceed.

Maximize $\sum_{i=1}^{n} p_i w_i$

$\sum_{i=1}^{n} x_i w_i$ <= M



7/30/2020

Knapsack Problem

DYNAMIC PROGRAMMING

- Considers All possible solution, then consider the optimal solution.
- Time consuming Method.
- Follows Principle of Optimality: Problem must be solved in sequence of decision.
- Overlapping Sub-problem.
- Memorization
- Tabulation

0/1 KNAPSACK PROBLEM

P = {1,2,5,6}

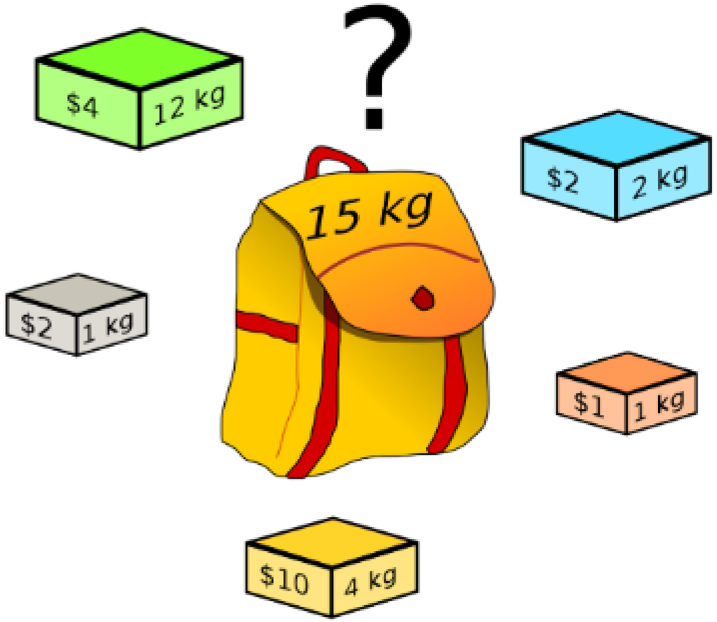
W = {2,3,4,5}

M = 8

N = 4

X = ?

2	3	4	5
?	?	?	?



7/30/2020

Knapsack Problem

DYNAMIC PROGRAMMING 0/1 KNAPSACK PROBLEM

- Determine how to fill a small knapsack optimally.
- Using the above information try to fill the larger knapsack optimally

DYNAMIC PROGRAMMING 0/1 KNAPSACK PROBLEM

P = {1,2,5,6}
W = {2,3,4,5}
M = 8
N = 4
X = ?

Knapsack size →			0	1	2	3	4	5	6	7	8
Pi	Wi	0									
1	2	1									
2	3	2									
5	4	3									
6	5	4									

DYNAMIC PROGRAMMING 0/1 KNAPSACK PROBLEM

- Recursive relationship is:
  $g(i,w)=\max\{g(i-1,w), g(i-1, w-w[i])+p[i]\}$

- $g(i,w)\rightarrow$ optimum profit of knapsack of a combination of 1 to i, with cumulative weight of w or less.
- $g(i-1,w)\rightarrow$ optimum profit of knapsack up to previous stage.
- $g(i-1, w-w[i])+p[i]\rightarrow$ current profit of w[i] + optimum profit of knapsack up to previous stage w.r.t w-w[i].

DYNAMIC PROGRAMMING 0/1 KNAPSACK PROBLEM

$g(i,w)=\max\{$
$g(i-1,w),$
$g(i-1, w-w[i])+p[i]$
$\}$

Knapsack size →			0	1	2	3	4	5	6	7	8
Pi	Wi	0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0	0	1	2	5	5	6	7	7
6	5	4	0								

$g(4,1)=\max\{ g(3,1) , g(3,1-5)+6\}$

DYNAMIC PROGRAMMING 0/1 KNAPSACK PROBLEM

g(4,1)=max{ g(3,1) , g(3,1-5)+6}
=max{ g(3,1) , g(3,-4)+6}
= g(3,1) = 0
g(4,2) / g(4,3) /g(4,4)
g(4,5) = max{ g(3,5) , g(3,5-5)+6}= 6
g(4,6) = ?
g(4,7) = ?
g(4,8)= ?

Knapsack size →			0	1	2	3	4	5	6	7	8
Pi	Wi	0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0	0	1	2	5	5	6	7	7
6	5	4	0								

g(i,w)=max{g(i-1,w), g(i-1, w-w[i])+p[i]}

DYNAMIC PROGRAMMING 0/1 KNAPSACK PROBLEM

4^{th} object
P=6
Remaining=2
3^{rd} object
2^{nd} object
Remaining=0
1^{st} object
0^{th} object

Knapsack size →			0	1	2	3	4	5	6	7	8
Pi	Wi	0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0	0	1	2	5	5	6	7	7
6	5	4	0	0	1	2	5	6	6	7	8

X={ 0 , 0 , 0 , 0 }

0/1 KNAPSACK PROBLEM

- WHICH STRATEGY IS BETTER?

Dynamic programming OR Greedy Approach

0/1 KNAPSACK PROBLEM

- WHICH STRATEGY IS BETTER? Ans – **Dynamic Programming**

Let $W=c$, and have $n=2$ items.
Also let $w1=c$ and $p1=c-1$, and $w2=1$ and $p2=1$.

The greedy algorithm will select only item 2, but the optimal solution contains only item 1. Meaning that the solution you obtain is $(c-1)$ times as poor as the true optimum.

FRACTIONAL KNAPSACK PROBLEM

- WHICH STRATEGY IS BETTER?

Dynamic programming OR Greedy Approach

FRACTIONAL KNAPSACK PROBLEM

- WHICH STRATEGY IS BETTER? Ans- Greedy Method
- They 'usually' don't use extra memory to keep a memory table (as dynamic programming) and have smaller complexity.
- Greedy always ensures optimal profit because of large fraction of weight/profit ratio.

NEXT CLASS

- Bellman-Ford algorithm.