We will discuss

→ Polynomial Time Approximation Schemes (PTAS)

Consider knapsack

$O_1 (s, v)$ $O_2 (s, v)$ .... $O_n (s, v)$

Capacity = W

Maximize value

→ NP complete

	O_1	O_2	O_3	O_4	O_5	O_6	O_7
Size	3	2	4	5	6	2	1
Value	2	3	5	4	3	5	2

capacity : 10

Ans: $\{O_2, O_3, O_6, O_7\} = \underline{\underline{15}}$

Knapsack has a pseudopolynomial time algorithm

n objects $(s_1 s_2 \cdots s_n)$ $(v_1 v_2 \cdots v_n)$

j →

1 2 3 4 .....

$V = (v_1 + v_2 + \cdots v_n)$

i ↓

1
2
3
.
.
.
.
n

$[i][j]$ ⇒ minimum size reqd to achieve value of exactly j using objects 1...i

mark '–' if not possible

$s = [3, 2, 4, 5, 6, 2, 1]$

$v = [2, 3, 5, 4, 3, 5, 2]$

	1	2	3	4	5	......	24
1	-	3	-	-	-		-
2	-	3	2	-	5	......	-
3	-	3	2	-	4	......	-
.							
.							
.							
7							

$M[i][j] =$

↓	↓	↓
Check $M[i-1][j]$	Check i's Value	i's weight + $M[i-1][j - i\text{'s value}]$
[not i but others]	[only i obj]	[i object and some of the rest]

take minimum of the 3 ...

Size of the table = $n.V$

time to build the table = $O(n.V)$

V could be exponential function of n.

idea: Dividing Columns by V

→ $1, 2, 3 \ldots V \Rightarrow 1/V, 2/V, 3/V, \ldots 1$

→ RAM can only handle numbers with fixed precision

→ Also, table size wouldn't change

let's improve the idea. after dividing each column by

a factor, let's round them off.

→ RAM can handle these rounded values

→ table size would shrink

0.5 1 1.5 2 2.5 ...

$$1 \xrightarrow{} \quad 2 \xrightarrow{}$$

by taking Minimum of the two

→ causes the soln to be sub optimal

	A	B
S	2	3
V	2	4
↳	/3	/3
↳	0.67	1.34
↳	1	1

(loss of info due to round off)

$s_1 \quad s_2 \quad s_3 \quad \dots \quad s_n$

$v_1 \quad v_2 \quad v_3 \quad \dots \quad v_n \} V$

↓

$\left\lfloor \frac{v_1}{x} \right\rfloor \left\lfloor \frac{v_2}{x} \right\rfloor \left\lfloor \frac{v_3}{x} \right\rfloor \dots \left\lfloor \frac{v_n}{x} \right\rfloor$ what should x be?

$$x = \frac{V}{n}(1-c)$$

$$0 \leq c < 1$$

running time ⇒ $O(n \cdot (\text{no of cols}))$

$\#$ cols = V'

$$V' = \left\lfloor \frac{v_1}{x} \right\rfloor + \left\lfloor \frac{v_2}{x} \right\rfloor + \dots \left\lfloor \frac{v_n}{x} \right\rfloor$$

$$= \left\lfloor \frac{n v_1}{V(1-c)} \right\rfloor + \left\lfloor \frac{n v_2}{V(1-c)} \right\rfloor + \dots \left\lfloor \frac{n v_n}{V(1-c)} \right\rfloor$$

$$= O\left(\frac{nV}{V(1-c)}\right) = O\left(\frac{n}{1-c}\right)$$

running time ⇒ $O\left(\frac{n^2}{1-c}\right)$ $0 \leq c < 1$

$c \Rightarrow 0$

running time $\Rightarrow$ polynomial

$c \Rightarrow 1$

running time $\Rightarrow$ exponential

<u>Quantifying the error</u>

optimum soln $\leq n$ objects

Mistake : selecting A instead of B

can only be made if $\left\lfloor \frac{V_A}{x} \right\rfloor = \left\lfloor \frac{V_B}{x} \right\rfloor$

Value lost due to mistake $V_B - V_A < x$

We can make n such mistakes at maximum

Max. absolute value lost / error $= n \cdot x$

Relative error $< \frac{V_{opt}}{V_{opt} - n \cdot x} = 1 + \frac{n \cdot x}{V_{opt} - n \cdot x}$

$$\leq 1 + \frac{n \cdot x}{V - n \cdot x}$$

$$= 1 + \frac{V(1-c)}{V - V(1-c)}$$

$$= 1 + \frac{1-c}{1-1+c} = \frac{1}{c}$$

$c \to 1$ good approx $O\left(\frac{n^2}{\sim 0}\right)$

$c \to 0$ bad approx. $O\left(\frac{n^2}{\sim 1}\right)$