3.24.2017

Change-making problem

Consider a given amount of money n and the coin system consisting of quarters, dimes, nickels, and pennies. How can the given amount of money n be made with the least number of coins?

Formulate this problem using Linear Programming.

x_1 = no. of quarters
x_2 = no. of dimes
x_3 = no. of nickels
x_4 = no. of pennies

Integer Linear Programming (ILP)

minimize $x_1 + x_2 + x_3 + x_4$

subject to $25x_1 + 10x_2 + 5x_3 + x_4 = n$

$x_1, x_2, x_3, x_4 \geq 0$

$x_1, x_2, x_3, x_4 \in \mathbb{Z}$

$x_2 \in \{0, 1, 2\}$

$x_3 \in \{0, 1\}$

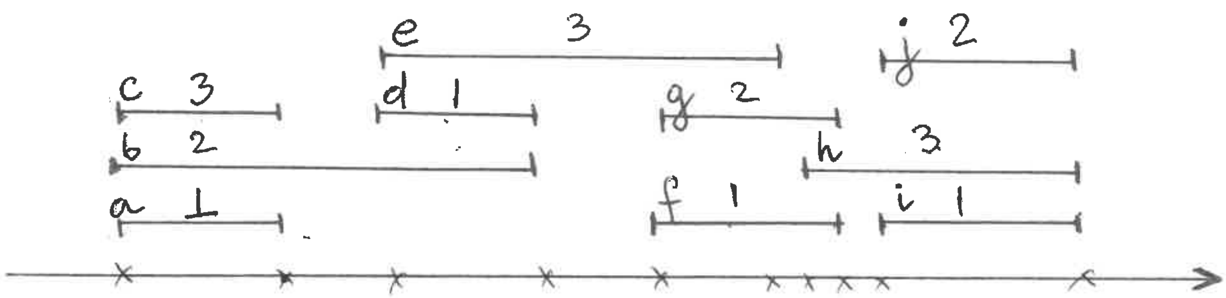
$x_4 \in \{0, 1, 2, 3, 4\}$

Greedy Algorithms

Scheduling All Intervals

$n = 10$ intervals

$d = 3$ depth



$O(n \lg n)$ sort the intervals by their start times, breaking ties arbitrarily
let $I_1, I_2, \ldots, I_n$ denote the intervals in this order

$O(n^2)$

for $j = 1, 2, 3, \ldots, n$
  $O(n)$
  for each interval I_i that precedes I_j in sorted order and overlaps it
    exclude the label of I_i from consideration for I_j
  endfor
  $O(n)$
  if there is any label from $\{1, 2, \ldots, d\}$ that has not been excluded then
    assign a nonexcluded label to I_j
  else
    leave I_j unlabeled
  endif
endfor

- Merge Sort or HeapSort has $RT = O(n \lg n)$

- each interval $[s_i, f_i)$
- we can compute d with time $O(n \lg n)$

$1 \le d \le n$

labels

1	2	3
~~F~~	~~F~~	F
~~T~~	T	

$(d=3)$

total $RT = O(n^2)$

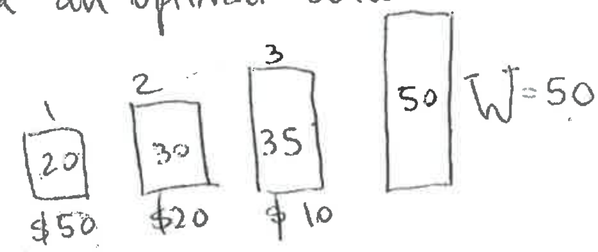
Greedy Algorithms

Fractional Knapsack Problem

* greedy choice : choose the object with the largest weight first

- may not yield an optimal solution

example



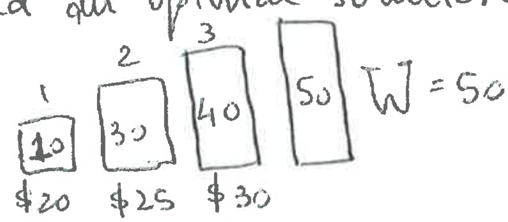
$\begin{cases} \text{item } 3 + \frac{1}{2} \text{ item } 2 \\ \text{value} = \$10 + \frac{1}{2} \cdot \$20 = \$20 \end{cases}$ → not optimal

optimal solution : $\begin{cases} \text{item } 1 + \text{item } 2 \\ \text{value} = \$50 + \$20 = \$70 \end{cases}$

* greedy choice : choose the object with the largest value first

- may not yield an optimal solution

example



$\begin{cases} \text{item } 3 + \frac{1}{3} \text{ item } 2 \\ \text{value} = \$30 + \frac{1}{3} \cdot \$25 = \$38.33 \end{cases}$ → not optimal

optimal solution : $\begin{cases} \text{item } 1 + \text{item } 2 + \frac{1}{4} \cdot \text{item } 3 \\ \text{value} = \$20 + \$25 + \frac{1}{4} \cdot \$30 = \$52.5 \end{cases}$

* greedy choice : choose the object with the largest v_i / w_i value

- always yields an optimal solution

example

$n = 5$, $W = 100$

	2	3	1	5	4
w	10	20	30	40	50
v	20	30	66	40	60
$\frac{v_i}{w_i}$	2	1.5	2.2	1	1.2

→ order of sorted objects

- sort objects in decreasing order of $\frac{v_i}{w_i}$

$$\begin{cases} load = 0 \\ value = 0 \end{cases}$$

$x_1 = 1$	$x_2 = 1$	$x_3 = 1$	$x_4 = \frac{4}{5}$
load $= 30$	load $= 30 + 10 = 40$	load $= 40 + 20 = 60$	load $= 60 + \frac{4}{5} \cdot 50 = 100$
value $= 66$	value $= 66 + 20 = 86$	value $= 86 + 30 = 116$	value $= 116 + \frac{4}{5} \cdot 60 = 164$

optimal solution: object 1 + object 2 + object 3 + $\frac{4}{5}$ object 4

value $= 164$