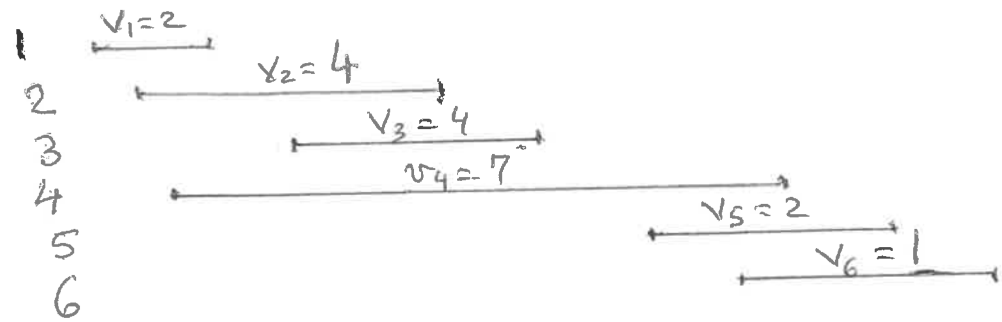
Dynamic Programming (DP)

Weighted Interval Scheduling

$n = 6$ requests



$p(1)=0$
$p(2)=0$
$p(3)=1$
$p(4)=0$
$p(5)=3$
$p(6)=3$

Characteristics of DP

- natural ordering of subproblems from smallest to largest

whole problem:

requests $\{1, 2, 3, \ldots, j, \ldots, n\}$

$O = O_n$ - optimal solution

$OPT = OPT(n)$ - the value of an optimal solution

subproblem

requests $\{1, 2, 3, \ldots, j\}$

O_j - optimal solution

$OPT(j)$ - the value of an optimal solution

- write solution of a problem recursively, based on solutions to subproblems

$p(j)$ - rightmost request smaller than j which does not overlap with j

Key observation: two possibilities - request j belongs to O_j or it doesn't belong to O_j

$$\boxed{OPT(j) = \max\left(v_j + OPT(p(j)),\ OPT(j-1)\right)}$$

→ recurrence with smaller subproblems

· request j belongs to O_j iff $v_j + OPT(p(j)) \geq OPT(j-1)$

Iterative approach

$$M[j] = \max\left(v_j + M[p(j)],\ M[j-1]\right)$$

$n = 6$

M	0	1	2	3	4	5	6
	0	2	4	6	7	8	8

$M[1] = \max(\underline{2 + M[0]},\ M[0]) = 2$

$M[2] = \max(\underline{4 + M[0]},\ M[1]) = 4$

$M[3] = \max(\underline{4 + M[1]},\ M[2]) = 6$

$M[4] = \max(\underline{7 + M[0]},\ M[3]) = 7$

$M[5] = \max(\underline{2 + M[3]},\ M[4]) = 8$

$M[6] = \max(1 + M[3],\ \underline{M[5]}) = 8$

$M[n] = M[6] = 8 \Rightarrow$ value of an optimal solution is 8

$OPT = 8$

Find-Solution(6)

Find-Solution(5)

Find-Solution(3) $\{5\}$

Find-Solution(1) $\{3\}$

Find-Solution(0) $\{1\}$

stop!

optimal solution consists of the requests $O = \{1, 3, 5\}$ with value $OPT = 8$.

Change-making problem

N - amount to be changed

whole problem

coin denominations : $d_1, d_2, \ldots, d_i, \ldots, d_n$

subproblem

coin denominations : $d_1, d_2, \ldots, d_i$

$c[i,j]$ – minimum number of coins required to pay an amount j using only coins with denominations 1 to i

$0 \le j \le N$

$1 \le i \le n$

$c[i,j]$

use coins $d_1, d_2, \ldots, d_i$

j is the amount that we need to change

$c[n,N]$ – min number of coins for the whole (original) problem

Recurrence

To compute $c[i,j]$ we have 2 choices

- do not use any coins with denomination i, then $c[i,j] = c[i-1,j]$
- use at least one coin with denomination i, then $c[i,j] = 1 + c[i, j-d_i]$

$$\boxed{c[i,j] = \min(c[i-1,j],\ 1 + c[i, j-d_i])}$$

$c[i,0] = 0$ for all i

compute $c[i,j]$:
1. if $i = 1$ and $j < d_1$ then ∞ (no solution!)
2. elseif $i = 1$ then $1 + c[1, j-d_1]$
3. elseif $j < d_i$ then $c[i-1,j]$
4. else $\min(c[i-1,j], 1 + c[i, j-d_i])$

example

$N = 8$

$n = 3$

coin denominations

	1	2	3
d	1	4	6

$n = 3$

$c[1..n, 0..N]$

$c[n, N] = c[3, 8]$ [min. number of coins to change $N = 8$; solution to the whole (original) problem

table c

$i \backslash j$	0	1	2	3	4	5	6	7	8
1	0	1	2	3	4	5	6	7	8
2	0	1	2	3	1	2	3	4	2
3	0	1	2	3	1	2	1	2	2

$i = 1$ $\quad d_1 = 1$

$c[1,1] \overset{(2)}{=} 1 + c[1,0] = 1$

$c[1,2] \overset{(2)}{=} 1 + c[1,1] = 2$

$c[1,3] \overset{(2)}{=} 1 + c[1,2] = 3$

...

$i = 2$ $\quad d_1, d_2 = 4$

$c[2,1] \overset{(3)}{=} c[1,1] = 1$

$c[2,2] \overset{(3)}{=} c[1,2] = 2$

$c[2,3] \overset{(3)}{=} c[1,3] = 3$

$c[2,4] \overset{(4)}{=} \min(c[1,4], \underline{1 + c[2,0]}) = 1$

$c[2,5] \overset{(4)}{=} \min(c[1,5], \underline{1 + c[2,1]}) = 2$

$c[2,6] \overset{(4)}{=} \min(c[1,6], \underline{1 + c[2,2]}) = 3$

$c[2,7] \overset{(4)}{=} \min(c[1,7], \underline{1 + c[2,3]}) = 4$

$c[2,8] \overset{(4)}{=} \min(c[1,8], \underline{1 + c[2,4]}) = 2$

$i = 3$ $\quad d_1, d_2, d_3 = 6$

$c[3,1] \overset{(3)}{=} c[2,1] = 1$

$c[3,2] \overset{(3)}{=} c[2,2] = 2$

...

$c[3,5] \overset{(3)}{=} c[2,5] = 2$

$c[3,6] \overset{(4)}{=} \min(c[2,6], \underline{1 + c[3,0]}) = 1$

$c[3,7] \overset{(4)}{=} \min(c[2,7], \underline{1 + c[3,1]}) = 2$

$c[3,8] \overset{(4)}{=} \min(\underline{c[2,\overset{2}{8}]}, 1 + c[3,\overset{2}{2}]) = 2$

The minimum number of coins needed to change the amount $N=8$ is $c[n,N] = c[3,8] = 2$

PrintCoins(c, 3, 8)
|
PrintCoins(c, 2, 8) — $\{d_2\}$
|
PrintCoins(c, 2, 4) — $\{d_2\}$
|
PrintCoins(c, 2, 0)
stop!

Optimally, we change the amount $N=8$ using 2 coins with denominations $\{d_2, d_2\}$

0-1 Knapsack problem

whole problem

n objects $\{o_1, o_2, \ldots, o_i, \ldots, o_n\}$

subproblem

i objects $\{o_1, o_2, \ldots, o_i\}$

$V[i,j]$ – maximum value of the objects we can transport if the weight limit is j and if we only include objects from 1 to i

$0 \leq j \leq W$

$1 \leq i \leq n$

$V[i,j]$

select objects from $\{o_1, o_2, \ldots, o_i\}$

knapsack capacity available

$V[n, W]$ – solution to the whole (original) problem

Recurrence

To compute $V[i,j]$ we have 2 choices.

- not adding object i to the knapsack, $V[i,j] = V[i-1, j]$
- adding object i to the knapsack, $V[i,j] = v_i + V[i-1, j-w_i]$

$$\boxed{V[i,j] = \max(V[i-1,j],\ v_i + V[i-1, j-w_i])}$$

Rules for filling up the table

$V[i,0] = 0$ for all i

$V[0,j] = 0$ for all $j \geq 0$

$V[i,j] = -\infty$ for all i when $j < 0$

$V[i,j] = \max(V[i-1,j],\ v_i + V[i-1, j-w_i])$

example

$n = 5$ objects

Knapsack capacity $W = 11$

object	weight	value
1	1	1
2	2	6
3	5	18
4	6	22
5	7	28

solution

$V[1..n, 0..W]$

$V[1..5, 0..11]$

$V[n, W] = V[5, 11]$ – solution to the whole (original) problem

table V

$i \backslash j$	0	1	2	3	4	5	6	7	8	9	10	11
1	0	1	1	1	1	1	1	1	1	1	1	1
2	0	1	6	7	7	7	7	7	7	7	7	7
3	0	1	6	7	7	18	19	24	25	25	25	25
4	0	1	6	7	7	18	22	24	28	29	29	40
5	0	1	6	7	7	18	22	28	29	34	35	40

$\underline{i=1}$ o_1

$V[1,1] = \max(\overset{0}{V[0,1]}, \underline{1 + \overset{0}{V[0,0]}}) = 1$

$V[1,2] = \max(V[0,2], \underline{1 + V[0,1]}) = 1$

...

$\underline{i=2}$ o_1, o_2

$V[2,1] = \max(\underline{V[1,1]}, 6 + \overset{-\infty}{V[1,-1]}) = 1$

$V[2,2] = \max(V[1,2], \underline{6 + V[1,0]}) = 6$

$V[2,3] = \max(V[1,3], \underline{6 + V[1,1]}) = 7$

...

$\underline{i=3}$ $o_1, o_2, \underline{o_3}$

$V[3,1] = \max(\underline{V[2,1]}, 18 + \overset{-\infty}{V[2,-4]}) = 1$

...

$V[3,4] = \max(\underline{V[2,4]}, 18 + V[2,-1]) = 7$

$V[3,5] = \max(V[2,5], \underline{18 + V[2,0]}) = 18$

...

$i=4$ $\quad o_1, o_2, o_3, \underline{o_4}$

$$V[4,1] = \max(\underline{V[3,1]},\ 22 + V[3,-5]) = 1$$

$\vdots$

$$V[4,5] = \max(\underline{V[3,5]},\ 22 + V[3,-1]) = 18$$

$$V[4,6] = \max(V[3,6],\ \underline{22 + V[3,0]}) = 22$$

$\vdots$

$$V[4,11] = \max(V[3,11],\ \underline{22 + \overset{18}{V[3,5]}}) = 40$$

$i=5$ $\quad o_1, o_2, o_3, o_4, \underline{o_5}$

$$V[5,1] = \max(\underline{V[4,1]},\ 28 + V[4,-6]) = 1$$

$\vdots$

$$V[5,11] = \max(\underline{\overset{40}{V[4,11]}},\ 28 + \overset{7}{V[4,4]}) = 40$$

The maximum value that we can transport in the knapsack is $V[n,W] = V[5,11] = \underline{40}$

- Which objects have been selected for maximum value?

PrintObjects(V, 5, 11)

PrintObjects(V, 4, 11)

PrintObjects(V, 3, 5) $\qquad \{o_4\}$

PrintObjects(V, 2, 0) $\qquad \{o_3\}$

stop!

The optimal solution is $\{o_3, o_4\}$ with total value 40.

Sequence Alignment problem

X: a b d c

Y: a c a a d

(1,1) (1,2) (3,5) is <u>NOT a matching</u>

X: a b a c d e f

Y: b a c c e

(1,3) (2,1) (4,4) (6,5) is a <u>matching</u>, but it is <u>NOT an alignment</u>.

<u>Alignment</u>: no crossing pairs are allowed!

X: a b a c d e f

Y: b a c c e

–	a	b	a	c	d	–	e	f
b	a	–	–	c	–	c	e	–

alignment: (1,2) (4,3) (6,5)

* for any pairs (i,j) and (i',j')
  if $i<i'$ then $j<j'$

Sequence Alignment Problem

Given two sequences X and Y, find an <u>optimal alignment</u> for X and Y.

Sequence Alignment Problem

- optimal alignment of X and Y

$X = x_1 x_2 \cdots x_m$ input size: m, n

$Y = y_1 y_2 \cdots y_n$

- subproblem: optimal alignment of a prefix X_i and Y_j

$X_i = x_1 x_2 \cdots \underline{x_i}$

$Y_j = y_1 y_2 \cdots \underline{y_j}$

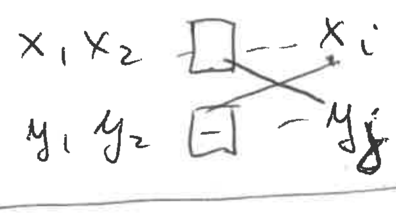
$OPT(i,j)$ – minimum cost of an alignment of X_i and Y_j

$OPT(m,n)$ – minimum cost to align X and Y (whole problem)

- to align X_i and Y_j, there are 3 cases:

$$\begin{array}{l|c|} x_1 x_2 \cdots x_{i-1} & x_i \\ y_1 y_2 \cdots y_{j-1} & y_j \end{array} \qquad \begin{array}{l|c|} x_1 x_2 \cdots x_{i-1} & x_i \\ y_1 y_2 \cdots y_j & - \end{array} \qquad \begin{array}{l|c|} x_1 x_2 \cdots x_i & - \\ y_1 y_2 \cdots y_{j-1} & y_j \end{array}$$

Note: cannot have both x_i and y_j matched to other characters. This will result in a crossing, which is not allowed in an alignment.



$$\boxed{OPT(i,j) = \min\{\alpha_{x_i y_j} + OPT(i-1,j-1),\ \delta + OPT(i-1,j),\ \delta + OPT(i,j-1)\}}$$

Sequence Alignment – Example

Find an optimal alignment of the sequences X and Y, where

X = mean

Y = name

Consider the following costs:

gap cost $\delta = 2$

matching cost:
- same symbols, cost = 0
- vowel & different vowel, cost = 1
- consonant & different consonant, cost = 1
- vowel & consonant, cost = 3

solution

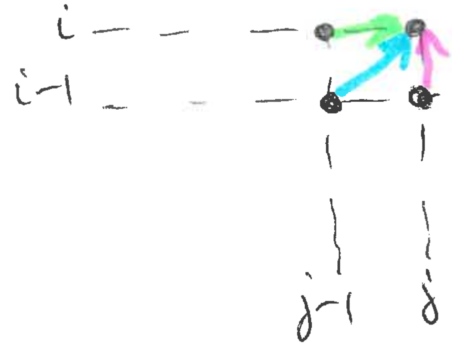
A

$m = 4$

$n = 4$

X	i					
n	4	8	6	5	$\boxed{4}$	$\boxed{6}$
a	3	6	5	$\boxed{3}$	5	5
e	2	4	$\boxed{3}$	2	4	4
m	1	2	$\boxed{1}$	3	4	6
_	0	$\boxed{0}$	2	4	6	8
		0	1	2	3	4
		_	n	a	m	e
			Y	j		

$$\begin{cases} A[i,0] = i\delta \text{ for each } i \\ A[0,j] = j\delta \text{ for each } j \\ A[i,j] = \min\{\alpha_{x_i y_j} + A[i-1,j-1],\ \delta + A[i-1,j],\ \delta + A[i,j-1]\} \end{cases}$$



$\underline{j=1}$

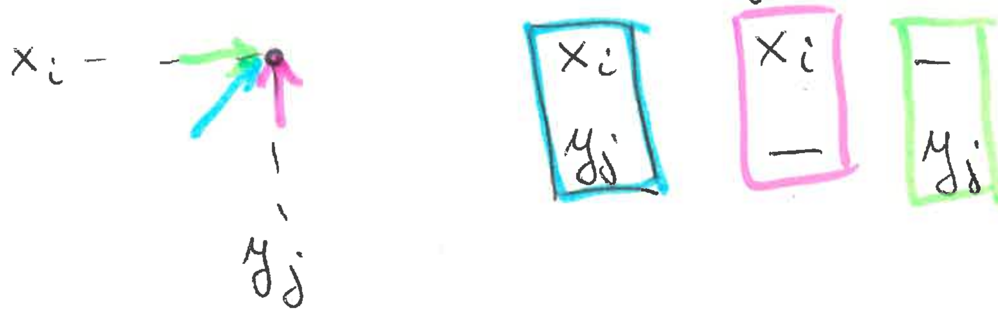
$A[1,1] = \min\{\underline{1+0}, 2+2, 2+2\} = 1$

$A[2,1] = \min\{3+2, \underline{2+1}, 2+4\} = 3$

...

optimal alignment cost is $A[m,n] = A[4,4] = \underline{6}$

- How can we find an optimal alignment with cost 6?



Optimal alignment:

$$\begin{array}{lccccc} & 1 & 2 & 3 & 4 & \\ X: & m & e & a & n & - \\ Y: & n & - & a & m & e \\ & 1 & & 2 & 3 & 4 \end{array}$$

$(1,1)\ (3,2)\ (4,3)$

$\text{cost} = 1+2+0+1+2 = \underline{6}$