

Let $C[i, j]$ = smallest cost to multiply $A_i A_{i+1} \dots A_j$

$$C[1, n] = \min_{1 \leq k \leq n-1} \{ C[1, k] + C[k+1, n] + p_0 p_k p_n \}$$

In general:

$$\text{for } i < j \quad C[i, j] = \min_{i \leq k \leq j-1} \{ C[i, k] + C[k+1, j] + p_{i-1} p_k p_j \}$$

$$C[i, i] = 0$$

For example:

	1	2	3	4
1	0	1000	1500	7000
2		0	200	3000
3			0	1000
4				0

$$p_0 = 50 \quad p_1 = 20 \quad p_2 = 1 \quad p_3 = 10 \quad p_4 = 100$$

$$C[1, 2] = \min \{ C[1, 1] + C[2, 2] + p_0 p_1 p_2 \} = 1000$$

$$C[2, 3] = 200$$

$$C[3, 4] = 1000$$

$$C[1, 3] = \min \left\{ \begin{array}{l} C[1, 1] + C[2, 3] + p_0 p_1 p_3 \\ C[1, 2] + C[3, 3] + p_0 p_2 p_3 \end{array} \right\} = 1500$$

$0 + 200 + 10,000 \quad ; \quad 1000 + 0 + 500$

$$C[2, 4] = \min \left\{ \begin{array}{l} C[2, 2] + C[3, 4] + p_1 p_2 p_4 \\ C[2, 3] + C[4, 4] + p_1 p_3 p_4 \end{array} \right\} = 3000$$

$0 + 1000 + 2000 \quad ; \quad 200 + 0 + 20,000$

$$C[1, 4] = \min \left\{ \begin{array}{l} C[1, 1] + C[2, 4] + p_0 p_1 p_4 \\ C[1, 2] + C[3, 4] + p_0 p_2 p_4 \\ C[1, 3] + C[4, 4] + p_0 p_3 p_4 \end{array} \right\} = 7000$$

$0 + 3000 + 100,000 \quad ; \quad 1000 + 1000 + 5000 \quad ; \quad 1500 + 0 + 50,000$

Matrix Chain Cost ($p_0 p_1 \dots p_n$)

$$C[i, i] = 0 \quad \text{for all } 1 \leq i \leq n$$

for $d = 1$ to $n-1$

for $i = 1$ to $n-d$

$$\left[\begin{array}{l} j = i+d \\ C[i, j] = \min_{i \leq k \leq j-1} \{ C[i, k] + C[k+1, j] + p_{i-1} p_k p_j \} \end{array} \right.$$

return $C[1, n]$

running time = $O(n^3)$
(also $\Omega(n^3)$)

How would you output the parenthesized expression?

Longest Common Subsequence

(52)

Given sequences $X[1..m]$ and $Y[1..n]$

Find the longest subsequence of X that is also a subsequence of Y .

← e.g. strings of characters

Example: $X = \text{FANTASTIC}$

$Y = \text{ATINYS ET}$

$\text{LCS}(X, Y) = \text{ANST}$

Equivalent problems:

- ① Delete the fewest characters from X and Y so that the remaining strings are identical
- ② Find the most non-crossing matching lines between X and Y

Observation

→ If $X[m] = Y[n]$ (last characters match)

$X = \circ \circ \circ \circ \circ \circ A$
 $Y = \circ \circ \circ \circ A$

then there is some optimal solution that contains this match. We are left with matching $X[1..m-1]$ with $Y[1..n-1]$ optimally.

→ If $X[m] \neq Y[n]$

$X = \circ \circ \circ \circ \circ \circ A$

$Y = \circ \circ \circ \circ B$

then $X[m]$ might match something in $Y[1..n-1]$
or $Y[n]$ might " " " $X[1..m-1]$

but not both (It also could be that neither match)

So either optimally match $X[1..m]$ with $Y[1..n-1]$
or optimally match $X[1..m-1]$ with $Y[1..n]$

Let $F[i, j]$ = length of LCS of $X[1...i]$ and $Y[1...j]$

$$F[i, j] = \begin{cases} F[i-1, j-1] + 1 & \text{if } X[i] = Y[j] \\ \max \{ F[i, j-1], F[i-1, j] \} & \text{otherwise} \end{cases}$$

$$F[0, 0] = 0$$

$$F[i, 0] = 0 \quad 1 \leq i \leq m$$

$$F[0, j] = 0 \quad 1 \leq j \leq n$$

LCS Length (X, Y)

For $i = 1$ to m $F[i, 0] = 0$

For $j = 1$ to n $F[0, j] = 0$

$F[0, 0] = 0$

For $i = 1$ to m

For $j = 1$ to n

if $X[i] = Y[j]$ then

$$F[i, j] = F[i-1, j-1] + 1$$

$$\text{arrow}[i, j] = (i-1, j-1)$$

else if $F[i, j-1] > F[i-1, j]$ then

$$\begin{cases} F[i, j] = F[i, j-1] \\ \text{arrow}[i, j] = (i, j-1) \end{cases}$$

$$\text{else } \begin{cases} F[i, j] = F[i-1, j] \\ \text{arrow}[i, j] = (i-1, j) \end{cases}$$

$i = m$

$j = n$

while $(i > 0 \ \&\& \ j > 0)$

if $\text{arrow}[i, j] = (i-1, j-1)$ then $\text{LCS} = X[i] \cdot \text{LCS}$

$(i, j) = \text{arrow}[i, j]$

return LCS

running Time
= $O(nm)$

		A	T	I	N	Y	S	E	T
	0	0	0	0	0	0	0	0	0
F	0	↑0	↑0	↑0	↑0	↑0	↑0	↑0	↑0
A	0	↘1	←1	←1	←1	←1	←1	←1	←1
N	0	↑1	↑1	↑1	↘2	←2	←2	←2	←2
T	0	↑1	↘2	←2	↑2	↑2	↑2	↑2	↘3
A	0	↘1	↑2	↑2	↑2	↑2	↑2	↑2	↑3
S	0	↑1	↑2	↑2	↑2	↑2	↘3	←3	↑3
T	0	↑1	↘2	↑2	↑2	↑2	↑3	↑3	↘4
I	0	↑1	↑2	↘3	←3	←3	↑3	↑3	↑4
C	0	↑1	↑2	↑3	↑3	↑3	↑3	↑3	↑4

Bellman Ford Algorithm

55

Given directed graph $G=(V,E)$ with edge weights $w: E \rightarrow \mathbb{R}$ and a source vertex s

Find (the length of) a shortest path from s to every other vertex in G . (even if G has negative weight edges but no negative wt. cycles)

Observation

A shortest path from s to v with at most i edges is a shortest path from s to u with at most $i-1$ edges plus the edge (u,v) (for some vertex u), or is a shortest path from s to v with at most $i-1$ edges.

Let $D[i,v]$ = the length of a shortest path that has at most i edges from s to v .

$$D[i,v] = \min \left\{ \min_{(u,v) \in E} \{ D[i-1,u] + w(u,v) \}, D[i-1,v] \right\}$$

$$D[0,s] = 0$$

$$D[0,v] = \infty \quad \text{for } v \neq s, v \in V \quad (\text{assume vertices are integers } 1, 2, \dots, n)$$

$D[n-1,v]$ = length of shortest path from s to v

↑ why? This is true iff no negative weight cycles.

for $i = 1$ to $n-1$

[for $v \in V$
 $D[i,v] = D[i-1,v]$
 for $(u,v) \in E$
 [if $D[i,v] > D[i-1,u] + w(u,v)$ then
 $D[i,v] = D[i-1,u] + w(u,v)$

Running Time
 $O(n^2 + nm)$

And then Bellman and Ford noticed something...

We don't need to store $D[j, v]$ for $j < i$

- ① $D[v] = \infty, D[s] = 0$
- ② for $i = 1$ to $n-1$

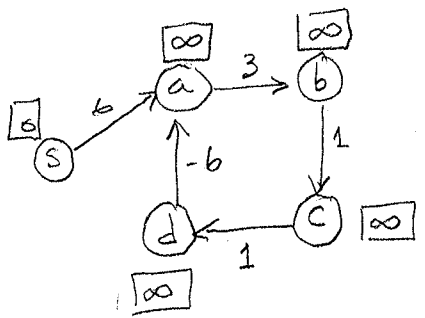
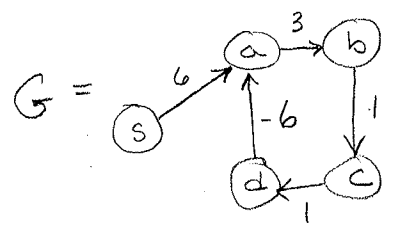
Running Time
 $O(nm)$

just use
a one-dimensional
array D

- ②a) for $(u, v) \in E$
 [If $D[v] > D[u] + w(u, v)$ then
 $D[v] = D[u] + w(u, v)$]

Bellman - Ford also noticed they could detect Neg. Wt. cycles (reachable from s)

- ③ for $(u, v) \in E$
 If $D[v] > D[u] + w(u, v)$ then stop "Negative Weight Cycle!"



	s	a	b	c	d
0	0	∞	∞	∞	∞
$i = 1$	0	6	∞	∞	∞
2	0	6	9	∞	∞
3	0	6	9	10	∞
4	0	6	9	10	11
	0	5	9	10	11

↑
Neg. Weight Cycle!

Notice: $D[v]$ after i^{th} iteration of loop ②
 may not be $D[i, v]$

For example:

If edge (s, a) is considered before edge (a, b)
 in ②a) then after 1 iteration of loop ②
 $D[b] = 9$ where $D[1, b] = \infty$