Matrix Chain Multiplication

Suppose we want to multiply four matrices:

$$A \times B \times C \times D$$

$$50\begin{bmatrix} & \\ & \\ & \end{bmatrix}_{20} \quad 20\begin{bmatrix} & \\ & \end{bmatrix}_{1} \quad 1\begin{bmatrix} & \end{bmatrix}_{10} \quad 10\begin{bmatrix} & \\ & \end{bmatrix}_{100}$$

with sizes 50×20, 20×1, 1×10, and 10×100.

We do this by multiplying two matrices at a time, maybe:

$$((A \times B) \times C) \times D$$

Note: Matrix multiplication is not commutative ($A \times B \neq B \times A$ in general) but it is associative ($(A \times B) \times C = A \times (B \times C)$)

Multiplying an $m \times n$ matrix by an $n \times p$ matrix takes mnp regular multiplications, and results in an $m \times p$ matrix.

So:

$$(\overbrace{(A \times B) \times C}^{50 \times 10}) \times D \quad \text{takes} \quad 50 \cdot 20 \cdot 1 + 50 \cdot 1 \cdot 10 + 50 \cdot 10 \cdot 100 = 51{,}500 \text{ multiplications}$$

(50×1)

but

$$(A \times B) \times (C \times D) \quad \text{takes} \quad 50 \cdot 20 \cdot 1 + 1 \cdot 10 \cdot 100 + 50 \cdot 1 \cdot 100 = 7{,}000 \text{ multiplications}$$

Given a chain of matrices $A_1, A_2 \ldots A_n$ with dimensions $P_0 \times P_1, P_1 \times P_2, \ldots, P_{n-1} \times P_n$ (so matrix A_i has dimension $P_{i-1} \times P_i$)
Find a way to parenthesize the product $A_1 \times A_2 \times \ldots \times A_n$ that minimizes the (number of regular multiplications) ← cost

Observation: To multiply $A_1 A_2 \ldots A_n$ optimally we must
- multiply $A_1 A_2 \ldots A_k$ optimally (to get a matrix of size $P_0 \times P_k$)
- and multiply $A_{k+1} \ldots A_n$ optimally (to get a matrix of size $P_k \times P_n$)
- and multiply the resulting two matrices together

But we don't know k so try all possible values of k and pick the smallest cost.

optimal substructure. Note: All subproblems are of the form "multiply $A_i A_{i+1} \ldots A_j$" optimally.

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

$$C[1,n] = \min_{1 \le k \le n-1} \{ C[1,k] + C[k+1,n] + P_0 P_k P_n \}$$

In general:

$$\text{for } i<j \quad C[i,j] = \min_{i \le k \le j-1} \{ C[i,k] + C[k+1,j] + P_{i-1} P_k P_j \}$$

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

For example:

$C =$

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

$P_0 = 50 \quad P_1 = 20 \quad P_2 = 1 \quad P_3 = 10 \quad P_4 = 100$

$$C[1,2] = \min \{ C[1,1] + C[2,2] + P_0 P_1 P_2 \}$$
$$= 1000$$

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

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

$$C[1,3] = \min \{ \underbrace{C[1,1] + C[2,3] + P_0 P_1 P_3}_{0 + 200 + 10{,}000}, \underbrace{C[1,2] + C[3,3] + P_0 P_2 P_3}_{1000 + 0 + 500} \}$$
$$= 1500$$

$$C[2,4] = \min \{ \underbrace{C[2,2] + C[3,4] + P_1 P_2 P_4}_{0 + 1000 + 2000}, \underbrace{C[2,3] + C[4,4] + P_1 P_3 P_4}_{200 + 0 + 20\,000} \}$$
$$= 3000$$

$$C[1,4] = \min \{ \underbrace{C[1,1] + C[2,4] + P_0 P_1 P_4}_{0 + 3000 + 100{,}000}, \underbrace{C[1,2] + C[3,4] + P_0 P_2 P_4}_{1000 + 1000 + 5000}, \underbrace{C[1,3] + C[4,4] + P_0 P_3 P_4}_{1500 + 0 + 50\,000} \}$$
$$= 7000$$

Matrix Chain Cost $(P_0 P_1 \cdots P_n)$

```
C[i,i] = 0   for all 1 ≤ i ≤ n
for d = 1 to n-1
    [for i = 1 to n-d
        [j = i + d
        C[i,j] = min_{i ≤ k ≤ j-1} { C[i,k] + C[k+1,j] + P_{i-1} P_k P_j }
return C[1,n]
```

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

How would you output the parenthesized expression?

Longest Common Subsequence

Given sequences $X[1...m]$ and $Y[1...n]$ ← e.g. strings of characters

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

Example: X = F A N T A S T I C

Y = A T I N Y S E T

$LCS(X, Y)$ = A N S T

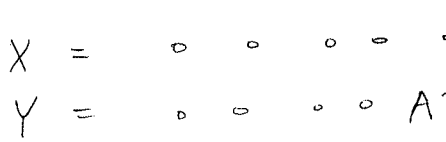
Equivalent problems:

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

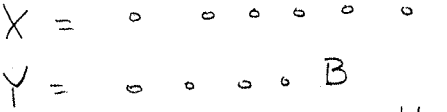
Observation

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



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

→ If $X[m] \neq Y[n]$



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

or $Y[n]$ might " " " $X[1...m-1]$

but <u>not</u> 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 $X[1 \ldots i]$ and $Y[1 \ldots j]$

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

$F[0,0] = 0$

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

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

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
                arrow[i,j] = (i-1, j-1)
        else if F[i, j-1] > F[i-1, j] then
                [ F[i,j] = F[i,j-1]
                  arrow[i,j] = (i, j-1)
        else    [ F[i,j] = F[i-1, j]
                  arrow[i,j] = (i-1, j)
```

running Time $= O(nm)$

```
i = m
j = n
while (i > 0 && j > 0)
    [ if arrow[i,j] = (i-1, j-1) then LCS = X[i] • LCS
      (i,j) = arrow[i,j]
return LCS
```

		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

Given directed graph $G=(V,E)$ with edge weights $w: E \to \mathbb{R}$ 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.

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

$D[0,s] = 0$

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

$D[n-1,v]$ = 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 ∈ V
        [D[i,v] = D[i-1,v]
        [for (u,v) ∈ 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] = ∞ , D[s] = 0
② for i = 1 to n-1
    2a  for (u,v) ∈ E
            [If D[v] > D[u] + w(u,v) then
                D[v] = D[u] + w(u,v)
```

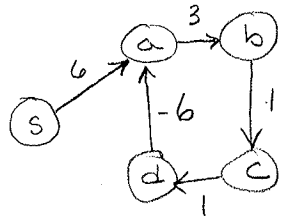
just use a one-dimensional array D

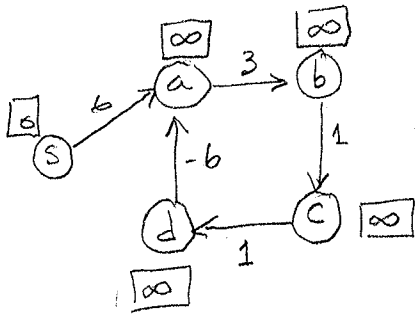
Running Time $O(nm)$

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!"

$G =$





i=	s	a	b	c	d
0	0	∞	∞	∞	∞
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 $D[1,b] = \infty$