

Matrix Chain Order Problem

- Multiplying non-square matrices:
 - A is $n \times m$ B is $m \times p$ must be equal
 - AB is $n \times p$ whose (i,j) entry is $\sum a_{ik} b_{kj}$
- Computing AB takes nmp scalar multiplications and $n(m-1)p$ scalar additions (using basic algorithm).
- Suppose we have a sequence of matrices to multiply.
- What is the best order?

Why Order Matters

- Suppose we have 4 matrices:
 - A , 30 x 1
 - B , 1 x 40
 - C , 40 x 10
 - D , 10 x 25
- $((AB)(CD))$: requires 41,200 mults.
- $(A((BC)D))$: requires 1400 mults.

Matrix Chain Order Problem

- Given matrices A_1, A_2, \dots, A_n , where A_i is $d_{i-1} \times d_i$:
 - [1] What is minimum number of scalar mults required to compute $A_1 \cdot A_2 \cdot \dots \cdot A_n$?
 - [2] What order of matrix multiplications achieves this minimum?

A Possible Solution

- Try all possibilities and choose the best one.
- Let $P(n)$ = # of alternative parenthesizations of a seq of n matrices.
- Then $P(n) = \sum_{k=1}^{n-1} P(k)P(n-k)$ if $n \geq 2$; $P(n)=1$ if $n=1$;
- Sol. $\Omega(4^n/n^{3/2})$ (*Hint: Catalan number*)
- Drawback is there are too many of them (exponential in the number of matrices to be multiplied)
- Need to be more clever - try dynamic programming!

Step 1: Develop a Recursive Solution

- Define $M(i,j)$ to be the minimum number of mults. needed to compute $A_i \cdot A_{i+1} \cdot \dots \cdot A_j$
- Goal: Find $M(1,n)$.
- Basis: $M(i,i) = 0$.
- Recursion: How to define $M(i,j)$ recursively?

Defining $M(i,j)$ Recursively

- Consider all possible ways to split A_i through A_j into two pieces.
- Compare the costs of all these splits:
 - best case cost for computing the product of the two pieces
 - plus the cost of multiplying the two products
- Take the best one
- $M(i,j) = \min_k (M(i,k) + M(k+1,j) + d_{i-1}d_kd_j)$

Defining $M(i,j)$ Recursively

$$\underbrace{(A_i \cdot \dots \cdot A_k)}_{P_1} \cdot \underbrace{(A_{k+1} \cdot \dots \cdot A_j)}_{P_2}$$

- minimum cost to compute P_1 is $M(i,k)$
- minimum cost to compute P_2 is $M(k+1,j)$
- cost to compute $P_1 \cdot P_2$ is $d_{i-1}d_kd_j$
- $M(i,j) = M(i,k) + M(k+1,j) + d_{i-1}d_kd_j$ for a k
- k run from i to $j-1$
- $M(i,j) = \min_k (M(i,k) + M(k+1,j) + d_{i-1}d_kd_j)$

Step 2: Find Dependencies Among Subproblems

M:

	1	2	3	4	5
1	0				GOAL!
2	n/a	0			
3	n/a	n/a	0		
4	n/a	n/a	n/a	0	
5	n/a	n/a	n/a	n/a	0

computing the pink square requires the purple ones: to the left and below.

Defining the Dependencies

- Computing $M(i,j)$ uses
 - everything in same row to the left:
 $M(i,i), M(i,i+1), \dots, M(i,j-1)$
 - and everything in same column below:
 $M(i,j), M(i+1,j), \dots, M(j,j)$


Step 3: Identify Order for Solving Subproblems

- Recall the dependencies between subproblems just found
- Solve the subproblems (i.e., fill in the table entries) this way:
 - go along the diagonal
 - start just above the main diagonal
 - end in the upper right corner (goal)

Order for Solving Subproblems

M:

	1	2	3	4	5
1	0				
2	n/a	0			
3	n/a	n/a	0		
4	n/a	n/a	n/a	0	
5	n/a	n/a	n/a	n/a	0



Pseudocode

```
for i := 1 to n do M[i,i] := 0
for d := 1 to n-1 do // diagonals
  for i := 1 to n-d to // rows w/ an entry on d-th diagonal
    j := i + d // column corresponding to row i on d-th diagonal
    M[i,j] := infinity
    for k := i to j-1 to
      M[i,j] := min(M[i,j], M[i,k]+M[k+1,j]+di-1dkdj)
    endfor
  endfor
endfor
endfor
```

pay attention here
to remember actual
sequence of mults.

running time $O(n^3)$

Example

M:

	1	2	3	4
1	0	1200	700	1400
2	n/a	0	400	650
3	n/a	n/a	0	10,000
4	n/a	n/a	n/a	0

1: A is 30x1
2: B is 1x40
3: C is 40x10
4: D is 10x25

Example

```
for i := 1 to n do M[i,i] := 0
for d := 1 to n-1 do // diagonals
  for i := 1 to n-d to // rows w/ an entry on d-th diagonal
    j := i + d          // column corresponding to row i on d-th diagonal
    M[i,j] := infinity
    for k := i to j-1 to
      M[i,j] := min(M[i,j], M[i,k]+M[k+1,j]+di-1dkdj)
    endfor
  endfor
endfor
```

M:

	1	2	3	4
1				
2	n/a			
3	n/a	n/a		
4	n/a	n/a	n/a	

1: A is 30x1
2: B is 1x40
3: C is 40x10
4: D is 10x25

Example

```

for i := 1 to n do M[i,i] := 0
for d := 1 to n-1 do // diagonals
  for i := 1 to n-d to // rows w/ an entry on d-th diagonal
    j := i + d          // column corresponding to row i on d-th diagonal
    M[i,j] := infinity
    for k := i to j-1 to
      M[i,j] := min(M[i,j], M[i,k]+M[k+1,j]+di-1dkdj)
    endfor
  endfor
endfor
endfor

```

M:

	1	2	3	4
1	0			
2	n/a	0		
3	n/a	n/a	0	
4	n/a	n/a	n/a	0

1: A is 30x1
 2: B is 1x40
 3: C is 40x10
 4: D is 10x25

Example

```

for i := 1 to n do M[i,i] := 0
for d := 1 to n-1 do // diagonals
  for i := 1 to n-d to // rows w/ an entry on d-th diagonal
    j := i + d          // column corresponding to row i on d-th diagonal
    M[i,j] := infinity
    for k := i to j-1 to
      M[i,j] := min(M[i,j], M[i,k]+M[k+1,j]+di-1dkdj)
    endfor
  endfor
endfor
endfor

```

M:

	1	2	3	4
1	0	1200		
2	n/a	0		
3	n/a	n/a	0	
4	n/a	n/a	n/a	0

d=1 i= 1, 2, 3

j= 2, 3, 4

k=1, 2, 3

M[1,2]

M[2,3]

M[3,4]

1: A is 30x1
 2: B is 1x40
 3: C is 40x10
 4: D is 10x25

Example

```

for i := 1 to n do M[i,i] := 0
for d := 1 to n-1 do // diagonals
  for i := 1 to n-d to // rows w/ an entry on d-th diagonal
    j := i + d          // column corresponding to row i on d-th diagonal
    M[i,j] := infinity
    for k := i to j-1 to
      M[i,j] := min(M[i,j], M[i,k]+M[k+1,j]+di-1dkdj)
    endfor
  endfor
endfor
endfor

```

M:

	1	2	3	4
1	0	1200		
2	n/a	0	400	
3	n/a	n/a	0	
4	n/a	n/a	n/a	0

d=1 i= 1, 2, 3

j= 2, 3, 4

k=1, 2, 3

M[1,2]

M[2,3]

M[3,4]

1: A is 30x1
 2: B is 1x40
 3: C is 40x10
 4: D is 10x25

Example

```

for i := 1 to n do M[i,i] := 0
for d := 1 to n-1 do // diagonals
  for i := 1 to n-d to // rows w/ an entry on d-th diagonal
    j := i + d // column corresponding to row i on d-th diagonal
    M[i,j] := infinity
    for k := i to j-1 to
      M[i,j] := min(M[i,j], M[i,k]+M[k+1,j]+di-1dkdj)
    endfor
  endfor
endfor
endfor

```

M:

	1	2	3	4
1	0	1200		
2	n/a	0	400	
3	n/a	n/a	0	10,000
4	n/a	n/a	n/a	0

d=1 i= 1, 2, 3

j= 2, 3, 4

k=1, 2, 3

M[1,2]

M[2,3]

M[3,4]

1: A is 30x1

2: B is 1x40

3: C is 40x10

4: D is 10x25

Example

```

for i := 1 to n do M[i,i] := 0
for d := 1 to n-1 do // diagonals
  for i := 1 to n-d to // rows w/ an entry on d-th diagonal
    j := i + d // column corresponding to row i on d-th diagonal
    M[i,j] := infinity
    for k := i to j-1 to
      M[i,j] := min(M[i,j], M[i,k]+M[k+1,j]+di-1dkdj)
    endfor
  endfor
endfor
endfor

```

M:

	1	2	3	4
1	0	1200		
2	n/a	0	400	
3	n/a	n/a	0	10,000
4	n/a	n/a	n/a	0

d=1 i= 1, 2, 3

j= 2, 3, 4

k=1, 2, 3

M[1,2]

M[2,3]

M[3,4]

1: A is 30x1
 2: B is 1x40
 3: C is 40x10
 4: D is 10x25

Example

```

for i := 1 to n do M[i,i] := 0
for d := 1 to n-1 do // diagonals
  for i := 1 to n-d to // rows w/ an entry on d-th diagonal
    j := i + d          // column corresponding to row i on d-th diagonal
    M[i,j] := infinity
    for k := i to j-1 to
      M[i,j] := min(M[i,j], M[i,k]+M[k+1,j]+di-1dkdj)
    endfor
  endfor
endfor
endfor

```

M:

	1	2	3	4
1	0	1200	700	
2	n/a	0	400	
3	n/a	n/a	0	10,000
4	n/a	n/a	n/a	0

d=2 i= 1, 2
j= 3, 4
k=1,2, 2,3

M[1,3]

M[2,4]

A(BC)

1: A is 30x1
2: B is 1x40
3: C is 40x10
4: D is 10x25

Example

```

for i := 1 to n do M[i,i] := 0
for d := 1 to n-1 do // diagonals
  for i := 1 to n-d to // rows w/ an entry on d-th diagonal
    j := i + d          // column corresponding to row i on d-th diagonal
    M[i,j] := infinity
    for k := i to j-1 to
      M[i,j] := min(M[i,j], M[i,k]+M[k+1,j]+di-1dkdj)
    endfor
  endfor
endfor

```

M:

	1	2	3	4
1	0	1200	700	
2	n/a	0	400	
3	n/a	n/a	0	10,000
4	n/a	n/a	n/a	0

13200

d=2 i= 1, 2
j= 3, 4
k=1,2, 2,3

M[1,3]

M[2,4]

(AB)C

1: A is 30x1
2: B is 1x40
3: C is 40x10
4: D is 10x25

Example

```

for i := 1 to n do M[i,i] := 0
for d := 1 to n-1 do // diagonals
  for i := 1 to n-d to // rows w/ an entry on d-th diagonal
    j := i + d // column corresponding to row i on d-th diagonal
    M[i,j] := infinity
    for k := i to j-1 to
      M[i,j] := min(M[i,j], M[i,k]+M[k+1,j]+di-1dkdj)
    endfor
  endfor
endfor
endfor

```

M:

	1	2	3	4
1	0	1200	700	
2	n/a	0	400	
3	n/a	n/a	0	10,000
4	n/a	n/a	n/a	0

d=2 i= 1, 2
 j= 3, 4
 k=1,2, 2,3
 M[1,3]
 M[2,4]

A(BC)

1: A is 30x1
 2: B is 1x40
 3: C is 40x10
 4: D is 10x25

Example

```

for i := 1 to n do M[i,i] := 0
for d := 1 to n-1 do // diagonals
  for i := 1 to n-d to // rows w/ an entry on d-th diagonal
    j := i + d // column corresponding to row i on d-th diagonal
    M[i,j] := infinity
    for k := i to j-1 to
      M[i,j] := min(M[i,j], M[i,k]+M[k+1,j]+di-1dkdj)
    endfor
  endfor
endfor
endfor

```

M:

	1	2	3	4
1	0	1200	700	
2	n/a	0	400	650
3	n/a	n/a	0	10,000
4	n/a	n/a	n/a	0

d=2 i= 1, 2
 j= 3, 4
 k=1,2, 2,3
 M[1,3]
 M[2,4]

(BC)D

1: A is 30x1
 2: B is 1x40
 3: C is 40x10
 4: D is 10x25

Example

```

for i := 1 to n do M[i,i] := 0
for d := 1 to n-1 do // diagonals
  for i := 1 to n-d to // rows w/ an entry on d-th diagonal
    j := i + d          // column corresponding to row i on d-th diagonal
    M[i,j] := infinity
    for k := i to j-1 to
      M[i,j] := min(M[i,j], M[i,k]+M[k+1,j]+di-1dkdj)
    endfor
  endfor
endfor
endfor

```

d=3 i= 1
 j= 4
 k=1, 2, 3
 M[1,4]

M:

	1	2	3	4
1	0	1200	700	1400
2	n/a	0	400	650
3	n/a	n/a	0	10,000
4	n/a	n/a	n/a	0

1: A is 30x1
 2: B is 1x40
 3: C is 40x10
 4: D is 10x25

A((BC)D)

Example

```

for i := 1 to n do M[i,i] := 0
for d := 1 to n-1 do // diagonals
  for i := 1 to n-d to // rows w/ an entry on d-th diagonal
    j := i + d // column corresponding to row i on d-th diagonal
    M[i,j] := infinity
    for k := i to j-1 to
      M[i,j] := min(M[i,j], M[i,k]+M[k+1,j]+di-1dkdj)
    endfor
  endfor
endfor
endfor

```

d=3 i= 1
 j= 4
 k=1, 2, 3
 M[1,4]

A((BC)D)

M:

	1	2	3	4
1	0	1200	700	∞
2	n/a	0	400	650
3	n/a	n/a	0	10,000
4	n/a	n/a	n/a	0

1400

1: A is 30x1
 2: B is 1x40
 3: C is 40x10
 4: D is 10x25

Example

```

for i := 1 to n do M[i,i] := 0
for d := 1 to n-1 do // diagonals
  for i := 1 to n-d to // rows w/ an entry on d-th diagonal
    j := i + d // column corresponding to row i on d-th diagonal
    M[i,j] := infinity
    for k := i to j-1 to
      M[i,j] := min(M[i,j], M[i,k]+M[k+1,j]+di-1dkdj)
    endfor
  endfor
endfor
endfor

```

$d=3$ $i=1$
 $j=4$
 $k=1, 2, 3$
 $M[1,4]$

(AB)(CD)

M:

	1	2	3	4
1	0	1200	700	1400
2	n/a	0	400	650
3	n/a	n/a	0	10,000
4	n/a	n/a	n/a	0

41200

1: A is 30x1
 2: B is 1x40
 3: C is 40x10
 4: D is 10x25

Example

```

for i := 1 to n do M[i,i] := 0
for d := 1 to n-1 do // diagonals
  for i := 1 to n-d to // rows w/ an entry on d-th diagonal
    j := i + d // column corresponding to row i on d-th diagonal
    M[i,j] := infinity
    for k := i to j-1 to
      M[i,j] := min(M[i,j], M[i,k]+M[k+1,j]+di-1dkdj)
    endfor
  endfor
endfor
endfor

```

d=3 i= 1
 j= 4
 k=1, 2, 3
 M[1,4]

(A(BC))D

M:

	1	2	3	4
1	0	1200	700	1400
2	n/a	0	400	650
3	n/a	n/a	0	10,000
4	n/a	n/a	n/a	0

8200

1: A is 30x1
 2: B is 1x40
 3: C is 40x10
 4: D is 10x25

Example

```

for i := 1 to n do M[i,i] := 0
for d := 1 to n-1 do // diagonals
  for i := 1 to n-d to // rows w/ an entry on d-th diagonal
    j := i + d          // column corresponding to row i on d-th diagonal
    M[i,j] := infinity
    for k := i to j-1 to
      M[i,j] := min(M[i,j], M[i,k]+M[k+1,j]+di-1dkdj)
    endfor
  endfor
endfor
endfor

```

M:

	1	2	3	4
1	0	1200	700	1400
2	n/a	0	400	650
3	n/a	n/a	0	10,000
4	n/a	n/a	n/a	0

1: A is 30x1
 2: B is 1x40
 3: C is 40x10
 4: D is 10x25

A((BC)D)

Keeping Track of the Order

- It's fine to know the cost of the cheapest order, but what *is* that cheapest order?
- Keep another array S and update it when computing the minimum cost in the inner loop
- After M and S have been filled in, then call a recursive algorithm on S to print out the actual order

Modified Pseudocode

```
for i := 1 to n do M[i,i] := 0
for d := 1 to n-1 do // diagonals
  for i := 1 to n-d to // rows w/ an entry on d-th diagonal
    j := i + d // column corresponding to row i on d-th diagonal
    M[i,j] := infinity
    for k := 1 to j -1 to
      M[i,j] := min(M[i,j], M[i,k]+M[k+1,j]+di-1dkdj)
      if previous line changed value of M[i,j] then S[i,j] := k
    endfor
  endfor
endfor
```

keep track of cheapest split point
found so far: between A_k and A_{k+1})
□ □

Example

M:

	1	2	3	4
1	0	1200	700	1400
2	n/a	0	400	650
3	n/a	n/a	0	10,000
4	n/a	n/a	n/a	0

1: A is 30x1
2: B is 1x40
3: C is 40x10
4: D is 10x25

S:

	1	2	3	4
1	n/a	1	1	1
2	n/a	n/a	2	3
3	n/a	n/a	n/a	3
4	n/a	n/a	n/a	n/a

Using S to Print Best Ordering

Call $\text{Print}(S, 1, n)$ to get the entire ordering.

$\text{Print}(S, i, j)$:

if $i = j$ then output "A" i

else $k := S[i, j]$

output "("

$\text{Print}(S, i, k)$

$\text{Print}(S, k+1, j)$

")"

M:

S:

	1	2	3	4
1	0	1200 1	700 1	1400 1
2	n/a	0	400 2	650 3
3	n/a	n/a	0	10,000 3
4	n/a	n/a	n/a	0

$A((BC)D)$

$(A_1((A_2A_3)A_4))$