DP : MCM (Matrix Chain Multiplication) → Simple → Toughest if you can't visualize recursion.

Train of thought - 1.

So far recursion, what is the structure of calls ?

```
fn (arr, n, ...)
  // Base case
  if (most trivial cond?) return —

  // You make recursive calls under conditions.
  if ___
     fn (arr, — ...)
  else
     fn(— ... ), fn(— ...)
```

Have we done ?

```
fn ( arr, .... ) :
  if . . . .

  for (int i = 0; i < n ; i++)
      fn(arr, i)
      fn(arr, n-i)
```

Is this doing pick/skip ?

~ op.

Train-of-thought-2.

① Fibonacci

```
f(n)
  if ....
  f(n-1)
  f(n-2)
```

② Knapsack

```
f(arr, ...)
  if — ..
  if
     f(n-1)
  else
     f(n-1), f(n, ...)
```

③ LCS. f(n,m).

```
  if ....
  if —.
     lcs(n-1, m-1)
  else
     f(n,m-1)
     f(n-1,m)
```

Matrix multiplication :

$$\underbrace{\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}}_{2\times 2}\underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_{2\times 1} = \underbrace{\begin{bmatrix} 1a+2b \\ 3a+4b \end{bmatrix}}_{2\times 1}$$

multiplications = 4

2 facts.

① AB = C

$a\times b \cdot b\times c = a\times c$ | size

② # multiplications = a.b.c

Sizes

$(a\times b \cdot b\times c) \to a\times c$

2.2.1

= 4

size = 2x2

#mul = 2.2.2 = 8

$$\underset{2\times 2}{\begin{bmatrix} a & b \\ c & d \end{bmatrix}} \underset{2\times 2}{\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}} = \underset{2\times 2}{\begin{bmatrix} 1a+3b & 2a+4b \\ 1c+3d & 2c+4d \end{bmatrix}}$$

#mul = 8

Observation : 3 matrices

AB ≠ BA

2 x 2	2 x 1	1 x 4		2 x 4
A .	B .	C	→	ans

total no. of multiplications across all steps.

multiplications

Case 1

$\underset{2\times 1}{(AB)} . \underset{1\times 4}{C}$

cost = 2.2.1 + 2.1.4

= 4 + 8

= 12

faster!!

Case 2

$\underset{2\times 2}{A} . \underset{2\times 4}{(BC)}$

cost = 2.1.4 + 2.2.4

= 8 + 16

= 24

4 matrices : A B C D

((A B).C). D

(A.(B.C)).D

A ((B.C) .D)

((AB)(CD))

A. (B.(CD))

which order gives me minimum no. of multiplications ?

#5 possibilities.

Q. Given the size of matrices, I want to multiply them in certain order such that no. of multiplications has to be minimum ?

o/p : # min no. of multiplication

MCM.

Parent → 2 variations | 3rd variation → 4 variations.

i/p : 2×1 $\quad 1 \times 4$ $\quad 4 \times 2$ $\quad 2 \times 4$ $\longrightarrow$ 2×4 size

arr :

2	1	4	2	4

n $\longrightarrow$ n−1 matrices.

$\sqcup$ $\longrightarrow$ sizes of matrices.

possibilities.

A | B C D E F G

A B | C D E F G

$\boxed{\text{A B C | D E F G}}$

G

I		II	III	
2	1	4	2	4

A (BCD) $\quad$ × $\quad$ (ABC) D

AB (CD)

A B C | D E F G

Restart : 2 matrices $\quad$ AB $\longrightarrow$ no choice.

3 matrices $\quad$ A B C

A | BC $\qquad$ AB | C

A (BC) $\qquad$ (AB) C

4 matrices $\quad$ A B C D

One of them, will be min. cost.

A | BCD $\qquad$ AB | CD $\qquad$ ABC | D

done.

2. A ((BC).D) $\qquad$ 3. $\qquad$ 4. (A(BC))D $\qquad$ 5. ((AB).C) D

1. A (B (CD))

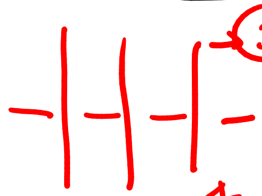
logic MCM

0 1 2 3 4 5 6 7 8

→ 8 Matrices.

mcm(i)

mcm(n-i) 1 2 3 4 5 6 7



3

4

≠

i-1 i

1 2 3

1x2 2x3

1 2x3

1x2 3

rec

) 1 instance / particular iteration

rec

0 1 2 3 4 5 6 7 8

a b c e f

3 possibilities 2 possibilities

3x2

→ 6 poss.

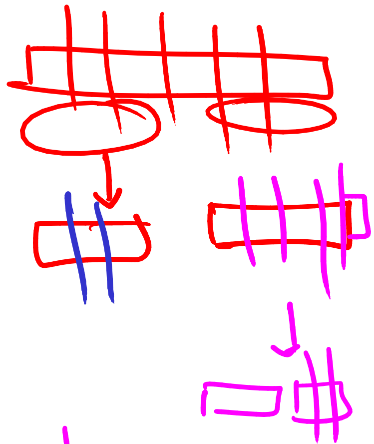
6 { a, e b, e c, e
a, f b, f c, f

i, j → range

k → partition place.

logic : int mcm(arr, i, j)

K



2 variations

↓

Samsung

Internship.

for k = i , k < j , k++ :

mcm(arr, i, k)

mcm(arr, k+1, j) }

0 1 2 3 4 5

left → mcm(1,1) mcm(1,2) mcm(1,3) mcm(1,4)

→ mcm(2,5) mcm(2,5) mcm(3,5) mcm(4,5)

right

i j

mcm(arr, 1, 5)

k : 1...4 1, 2, 3, 4

Dry run   string   A | B | C | D | E   original array

→ k for 1st call

~~A BCDE~~   ~~AB CDE~~   ~~ABC DE~~   ABCD E 5

~~B CDE~~ ~~BC DE~~ ~~BCD E~~

~~CDE~~ ~~B CD~~ ~~BC D~~

~~CDE~~

(A(BC)) (DE)

((AB).C) (DE)

5

A(((BC).D)E)

A((B.(CD))E)

A((BC)(DE))

A(B(C(DE)))

A(B((CD)E))

(AB)((CD)E)

(AB)(C(DE))

Total ways → 14.   5 + 2 + 2 + 5

idx   1×2 2×3 3×4 → x

arr: (1 | 3 | 4 | 2 | 1 | 5 | 1 | 2 | 3 | 4)

i   k   i   j

mcm(i, k)   mcm(k+1, j)

size : arr[i-1] × arr[k]   arr[k] × arr[j]

a   b   b   a×b   b×c

1×3 3×4 4×2 2×1 1×5 → 1×5 → abc

i-1

★ → 5×1 1×2 2×3 3×4

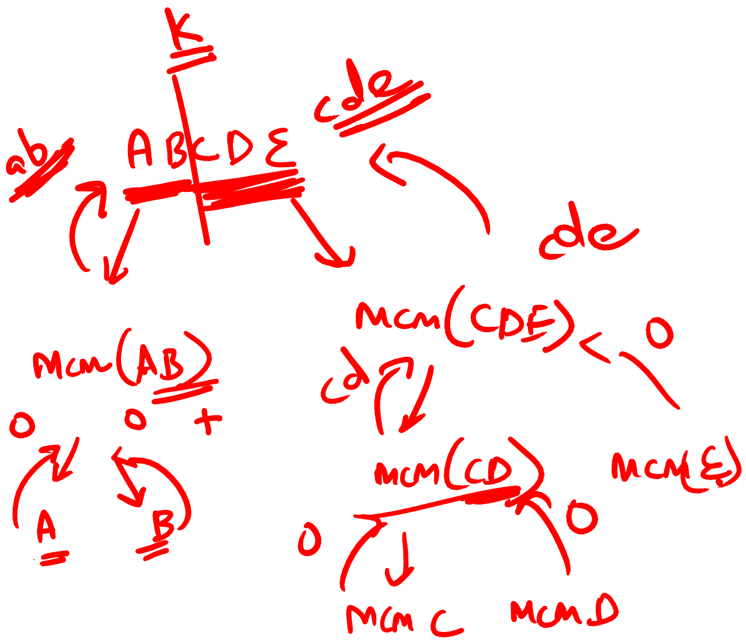
↳ 5×4

cost = arr[i-1] × arr[k] × arr[j] ←

Final logic:

```
int mcm (arr, i, j)                    mcm (arr, 1, n-1)
    if i >= j : return 0

    int ans = 10e8

    for (k = i , k < j , k++):
        int cost = mcm (arr, i, k) +
                   mcm (arr, k+1, j) +
                   arr[i-1] x arr[k] x arr[j]

        ans = min (ans, cost)

    return ans
```

MCM code done !



arr[k] x arr[j]

↓

cost = ab + cde + ____

↑

arr[i-1] x arr[k]

arr[i-1] x arr[k]

x arr[j]