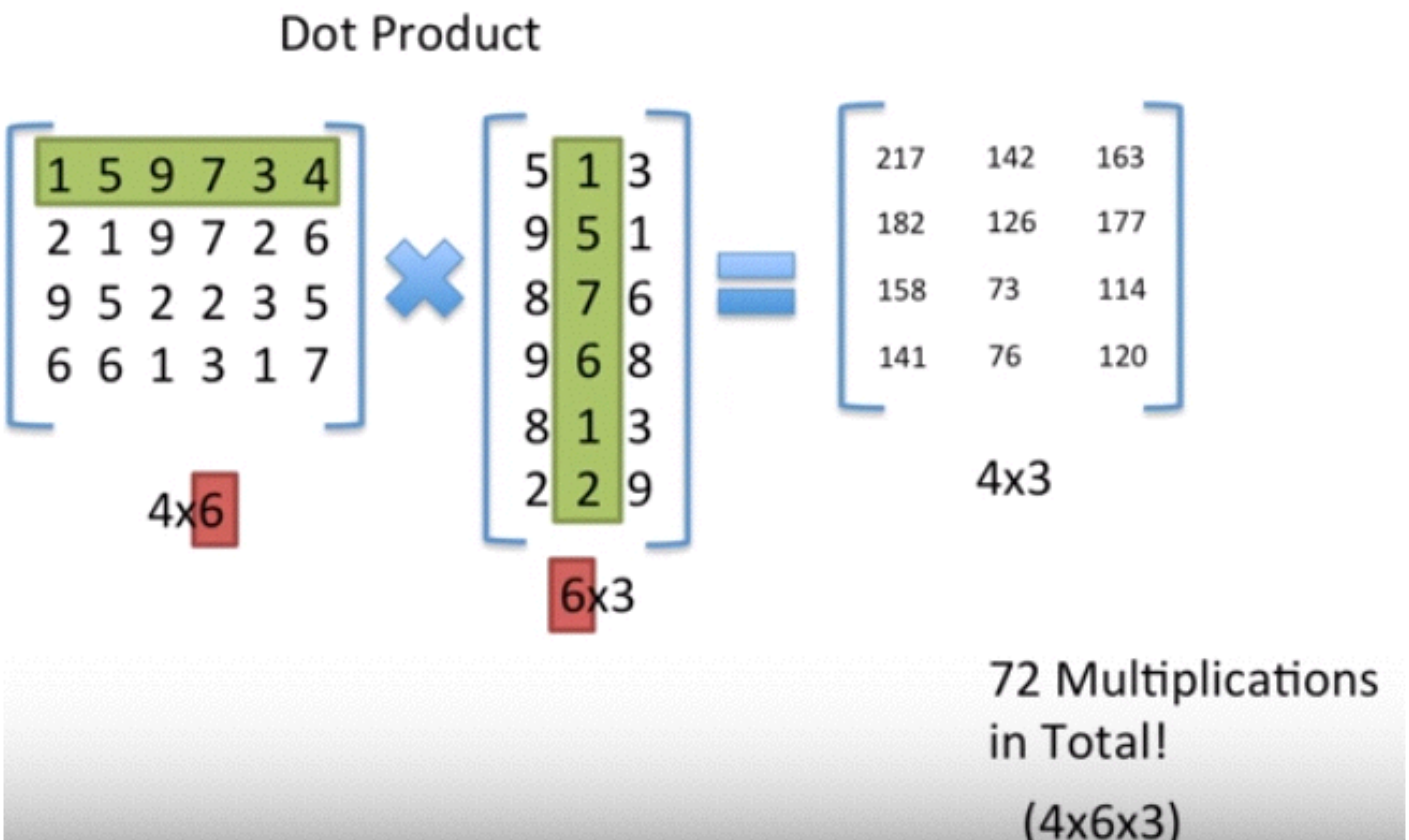
Dynamic Programming - Part 2

Debarka Sengupta

Matrix Chain Multiplication

Revisit matrix multiplication



Matrix-chain Multiplication

- Suppose we have a sequence or chain A_1, A_2, …, A_n of *n* matrices to be multiplied
  - That is, we want to compute the product $A_1A_2\ldots A_n$
- There are many possible ways (parenthesizations) to compute the product

Matrix-chain Multiplication …contd

- Example: consider the chain A_1, A_2, A_3, A_4 of 4 matrices
  - Let us compute the product $A_1A_2A_3A_4$
- There are 5 possible ways:

1. $(A_1(A_2(A_3A_4)))$
2. $(A_1((A_2A_3)A_4))$
3. $((A_1A_2)(A_3A_4))$
4. $((A_1(A_2A_3))A_4)$
5. $(((A_1A_2)A_3)A_4)$

Matrix-chain Multiplication …contd

- To compute the number of scalar multiplications necessary, we must know:
  - Algorithm to multiply two matrices
  - Matrix dimensions

- Can you write the algorithm to multiply two matrices?

Algorithm to Multiply 2 Matrices

Input: Matrices $A_{p\times q}$ and $B_{q\times r}$ (with dimensions $p\times q$ and $q\times r$)

Result: Matrix $C_{p\times r}$ resulting from the product $A\cdot B$

MATRIX-MULTIPLY($A_{p\times q}$, $B_{q\times r}$)

1. **for** $i \leftarrow 1$ **to** p
2.   **for** $j \leftarrow 1$ **to** r
3. $C[i, j] \leftarrow 0$
4.   **for** $k \leftarrow 1$ **to** q
5. $C[i, j] \leftarrow C[i, j] + A[i, k] \cdot B[k, j]$
6. **return** C

Scalar multiplication in line 5 dominates time to compute
CNumber of scalar multiplications = *pqr*

Matrix-chain Multiplication …contd

- Example: Consider three matrices $A_{10\times100}$, $B_{100\times5}$, and $C_{5\times50}$
- There are 2 ways to parenthesize
  - $((AB)C) = D_{10\times5} \cdot C_{5\times50}$
    - $AB \Rightarrow 10\cdot100\cdot5=5{,}000$ scalar multiplications
    - $DC \Rightarrow 10\cdot5\cdot50 =2{,}500$ scalar multiplications

    Total: 7,500
  - $(A(BC)) = A_{10\times100} \cdot E_{100\times50}$
    - $BC \Rightarrow 100\cdot5\cdot50=25{,}000$ scalar multiplications
    - $AE \Rightarrow 10\cdot100\cdot50 =50{,}000$ scalar multiplications

    Total: 75,000

Matrix-chain Multiplication …contd

- Matrix-chain multiplication problem
  - Given a chain $A_1, A_2, \ldots, A_n$ of n matrices, where for $i=1, 2, \ldots, n$, matrix A_i has dimension $p_{i-1} \times p_i$
  - Parenthesize the product $A_1A_2\ldots A_n$ such that the total number of scalar multiplications is minimized
- Brute force method of exhaustive search takes time exponential in n

Brute force

- For $n \geq 2$, a fully parenthesized matrix product is the product of 2 fully parenthesized matrix subproducts.
- The split can occur between k^{th} and $(k+1)^{th}$ matrices, for any k = 1, 2, …, n-1
- So, the recurrence representing the total # of possible parenthesizations is:
  - $P(n) = \begin{cases} 1 & \text{if } n = 1 \\ \sum_{k=1}^{n-1} P(k)P(n-k) & \text{if } n \geq 2 \end{cases}$
- Solution is tricky -- turns out, it grows as $\Omega\left(\frac{4^n}{n^{3/2}}\right)$
- Or, also true that it grows as $\Omega(2^n)$

Recursion (example of the first level)

- Consider the case multiplying these 4 matrices:
  - A: 2x4
  - B: 4x2
  - C: 2x3
  - D: 3x1

- 1. (A)(BCD) - This is a 2x4 multiplied by a 4x1,
  - so 2x4x1 = 8 multiplications, plus whatever work it will take to multiply (BCD).

- 2. (AB)(CD) - This is a 2x2 multiplied by a 2x1,
  - so 2x2x1 = 4 multiplications, plus whatever work it will take to multiply (AB) and (CD).

- 3. (ABC)(D) - This is a 2x3 multiplied by a 3x1,
  - so 2x3x1 = 6 multiplications, plus whatever work it will take to multiply (ABC).

Recursive formula

$$m(i,j) = \begin{cases} 0 & \text{If } i = j \\ \min_{i \le k < j}\{m(i,k) + m(k+1,j) + p_{i-1} \cdot p_k \cdot p_j\} & \text{If } i < j \end{cases}$$

Pseudocode

```
MATRIX-CHAIN(i, j)
    IF i = j THEN return 0
    m = ∞
    FOR k = i TO j − 1 DO
        q = MATRIX-CHAIN(i, k) + MATRIX-CHAIN(k + 1, j) + p_{i−1} · p_k · p_j
        IF q < m THEN m = q
    OD
    Return m
END MATRIX-CHAIN

Return MATRIX-CHAIN(1, n)
```

Recurrence relation

$$
\begin{aligned}
T(n) &= \sum_{k=1}^{n-1}(T(k) + T(n-k) + O(1)) \\
&= 2 \cdot \sum_{k=1}^{n-1} T(k) + O(n) \\
&\geq 2 \cdot T(n-1) \\
&\geq 2 \cdot 2 \cdot T(n-2) \\
&\geq 2 \cdot 2 \cdot 2 \ldots \\
&= 2^n
\end{aligned}
$$

Explaining he deduction applied to reach 2^{nd} step

If $n = 7$, for $k = 1, 2, .., 6$ we get the following

$1+ (7-1) = (1+6)$
$2+ (7-2) = (2+5)$
$3+ (7-3) = (3+4)$
$4+ (7-4) = (4+3)$
$5+ (7-5) = (5+2)$
$6+ (7-6) = (6+1)$

On the right hand side it is simply $(n-1) + (n-1)$ i.e., $2(n-1)$

Recursion to DP through memorisation - trivial

```
MATRIX-CHAIN(i, j)
    IF T[i][j] < ∞ THEN return T[i][j]
    IF i = j THEN T[i][j] = 0, return 0
    m = ∞
    FOR k = i to j − 1 DO
        q = MATRIX-CHAIN(i, k) + MATRIX-CHAIN(k + 1, j) + p_{i−1} · p_k · p_j
        IF q < m THEN m = q
    OD
    T[i][j] = m
    return m
END MATRIX-CHAIN

return MATRIX-CHAIN(1, n)
```

Runtime complexity

- Quadratic due to the size of T i.e., the memo

Putting brackets optimally

$A_1 \times A_2 \times A_3 \times A_4 \times A_5$

A_1	A_2	A_3	A_4	A_5
4x10	10x3	3x12	12x20	20x7

Goal: Find the optimal way to multiply these matrices to perform the fewest multiplications.

Naïve Approach: Try them all, and pick the most optimal one.

Running time: $\Omega(4^n/n^{3/2})$ - 4^n dominates! Exponential

Substructure Opimality

There is a better way! Dynamic Programming!

Step 1: Check if the problem has Optimal Substructure

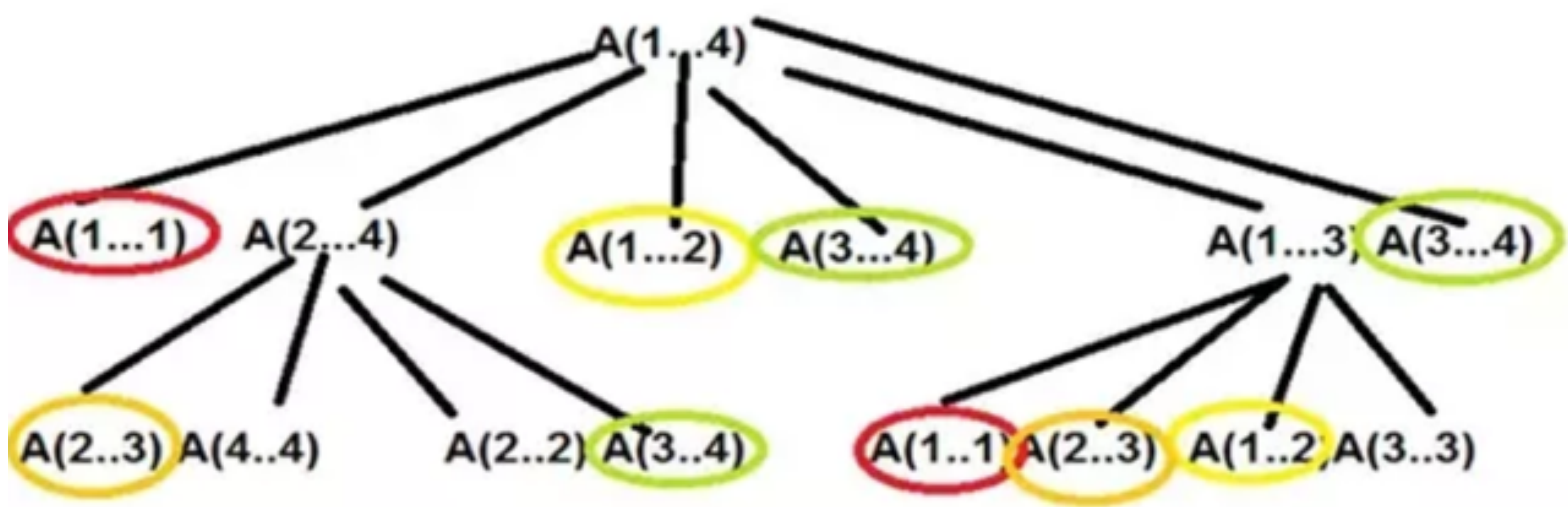
If we have an optimal solution for $A_{i\ldots j}$

Assume the solution has the following parentheses:

$$(\underline{A_{i\ldots k}})(\underline{A_{k+1\ldots j}})$$

If there is a better way to multiply $(A_{i\ldots k})$, then we would have a more optimal solution.

Overlapping subproblems



Recursive formula

Now we want to try out a bunch of values for 'k' in order to see what the best one is:

$$M[i,j] = M[i,k] + M[k+1,j] + p_{i-1}p_kp_j$$
$$100 \qquad 200 \qquad 2x3x4$$

Since we don't know what k is, we try this range of k:

$i \leq k < j$

$$\overset{100}{(A_{i...k})}\overset{200}{(A_{k+1...j})}$$
$$2x3 \qquad 3x4$$

The minimum returned value is our solution!

Our Final Recursive Formula:

$$M[i,j] = \begin{cases} 0 \quad \text{if} \quad i=j \\ \min_{i \leq k < j} \{M[i,j] = M[i,k] + M[k+1,j] + p_{i-1}p_kp_j\} \end{cases}$$

An instance

$$M[i,j] = \begin{cases} 0 & \text{if } i=j \\ \min_{i \le k < j} \{M[i,j] = M[i,k] + M[k+1,j] + p_{i-1}p_kp_j\} \end{cases}$$

Matrix Chain Multiplication

We want to start with i = j, then i<j starting with a spread of 1, working our way up

$$\underset{\substack{4x10 \\ p_0\ p_1}}{A_1} \times \underset{\substack{10x3 \\ p_1\ p_2}}{A_2} \times \underset{\substack{3x12 \\ p_2\ p_3}}{A_3} \times \underset{\substack{12x20 \\ p_3\ p_4}}{A_4} \times \underset{\substack{20x7 \\ p_4\ p_5}}{A_5}$$

i \ j	1	2	3	4	5
1	0	120			
2	x	0	360		
3	x	x	0		
4	x	x	x	0	
5	x	x	x	x	0

$M[1,2] = \min_{1 \le k < 2} \{M[1,1] + M[1+1,2] + p_0p_1p_2\}$

$M[1,2] = \min_{1 \le k < 2} \{0 + 0 + 4x10x3\}$

$M[1,2] = 120$

$M[2,3] = \min_{2 \le k < 3} \{M[2,2] + M[2+1,3] + p_1p_2p_3\}$

$M[2,3] = \min_{2 \le k < 3} \{0 + 0 + 10x3x12\}$

$M[2,3] = 360$

$M[3,4] = \min_{3 \le k < 4} \{M[3,3] + M[3+1,4] + p_2p_3p_4\}$

$M[3,4] = \min_{3 \le k < 4} \{0 + 0 + 3x12x20\}$

An instance

$$M[i,j] = \begin{cases} 0 & \text{if } i=j \\ \min_{i \le k < j} \{M[i,j] = M[i,k] + M[k+1,j] + p_{i-1}p_k p_j\} \end{cases}$$

Matrix Chain Multiplication

We want to start with i = j, then i<j starting with a spread of 1, working our way up

$A_1 \times A_2 \times A_3 \times A_4 \times A_5$

	A_1	A_2	A_3	A_4	A_5
	4x10	10x3	3x12	12x20	20x7
	$p_0\ p_1$	$p_1\ p_2$	$p_2\ p_3$	$p_3\ p_4$	$p_4\ p_5$

i \ j	1	2	3	4	5
1	0	120			
2	x	0	360		
3	x	x	0	720	
4	x	x	x	0	1680
5	x	x	x	x	0

$M[1,3] = \min_{1 \le k < 3}$

k=1

$= M[1,1] + M[1+1,3] + p_0p_1p_3$

$= 0 + 360 + 4 \times 10 \times 12$

$= 840$

k=2

$= M[1,2] + M[2+1,3] + p_0p_2p_3$

$= 120 + 0 + 4 \times 3 \times 12$

$= 264$

An instance

i \ j	1	2	3	4	5
1	0	120	264	1080	1344
2	x	0	360	1320	1350
3	x	x	0	720	1140
4	x	x	x	0	1680
5	x	x	x	x	0

Trace the solution

$$(A_1 \times A_2)((A_3 \times A_4)\ A_5)$$

A_1	A_2	A_3	A_4	A_5
4x10	10x3	3x12	12x20	20x7
p_0 p_1	p_1 p_2	p_2 p_3	p_3 p_4	p_4 p_5

k=2

$M[1,5] = \underline{M[1,2]} + \underline{M[3,5]} + p_0p_2p_5$

k=4

$M[3,5] = M[3,4] + M[5,5] + p_2p_4p_5$

Edit Distance

Edit distance

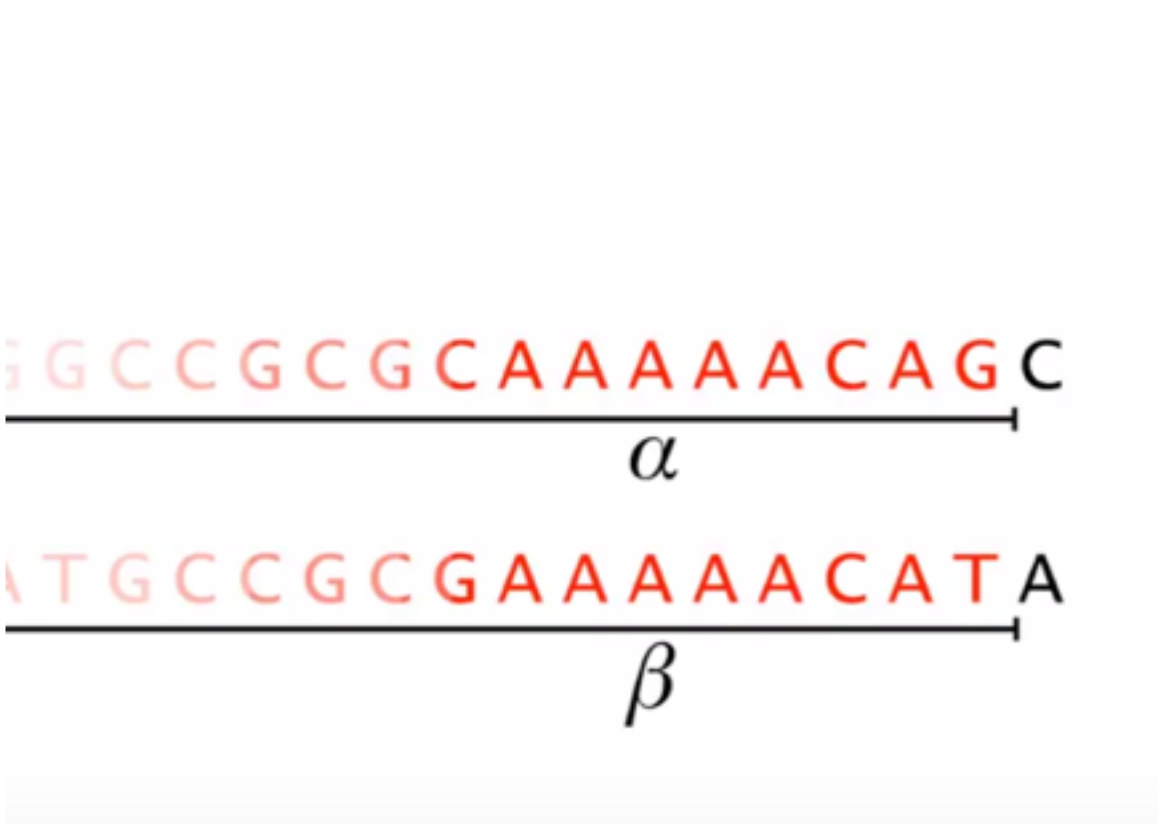
For X, Y where $|X| = |Y|$, *hamming distance* = minimum # substitutions needed to turn one into the other

For X, Y, *edit distance* = minimum # edits (substitutions, insertions, deletions) needed to turn one into the other

If $|X| = |Y|$ what can we say about the relationship between `editDistance`(X, Y) and `hammingDistance`(X, Y)?

`editDistance`(X, Y) ≤ `hammingDistance`(X, Y)

Substructure optimality



Recurrence

$$\alpha\ \mathrm{C}$$

$$\beta\ \mathrm{A}$$

$$\texttt{edist}(\alpha\mathrm{C}, \beta\mathrm{A}) = \min \begin{cases} \texttt{edist}(\alpha, \beta) + 1 \\ \texttt{edist}(\alpha\mathrm{C}, \beta) + 1 \\ \texttt{edist}(\alpha, \beta\mathrm{A}) + 1 \end{cases}$$

Recurrence general

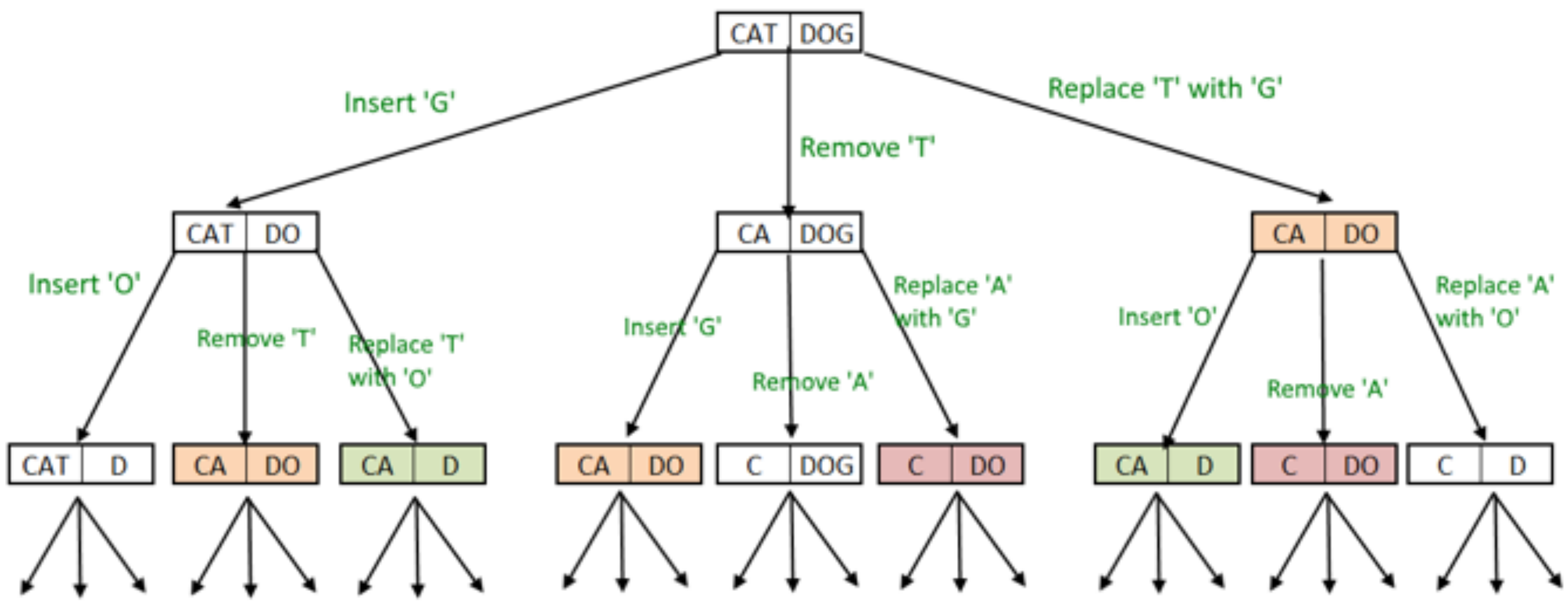
$$\alpha \; \mathrm{x}$$

$$\beta \; \mathrm{y}$$

$$\texttt{edist}(\alpha \mathrm{x}, \beta \mathrm{y}) = \min \begin{cases} \texttt{edist}(\alpha, \beta) + \delta(\mathrm{x}, \mathrm{y}) \\ \texttt{edist}(\alpha \mathrm{x}, \beta) + 1 \\ \texttt{edist}(\alpha, \beta \mathrm{y}) + 1 \end{cases}$$

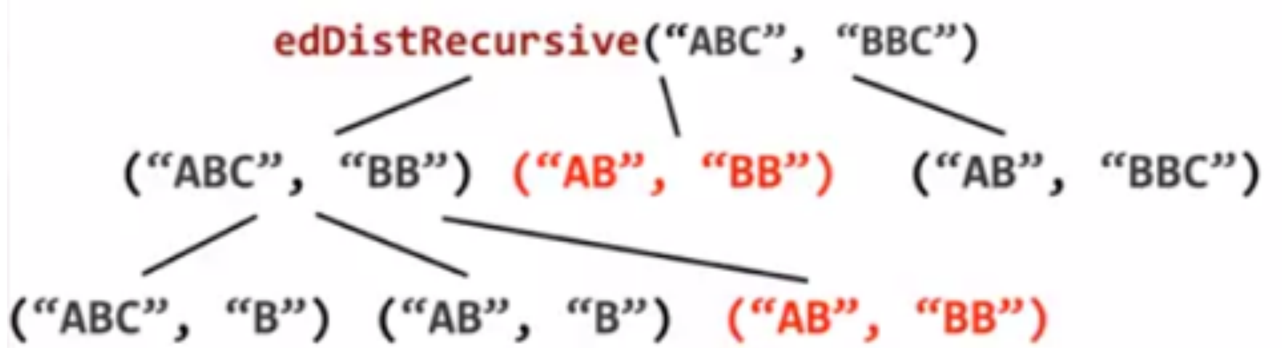
$\delta(\mathrm{x}, \mathrm{y}) = 0$ if $\mathrm{x} = \mathrm{y}$, or 1 otherwise

Recursion tree for an example



Recursive function

```
>>> import datetime as d
>>> st = d.datetime.now(); \
... edDistRecursive("Shakespeare", "shake spear"); \
... print (d.datetime.now()-st).total_seconds()
3
31.498284
```

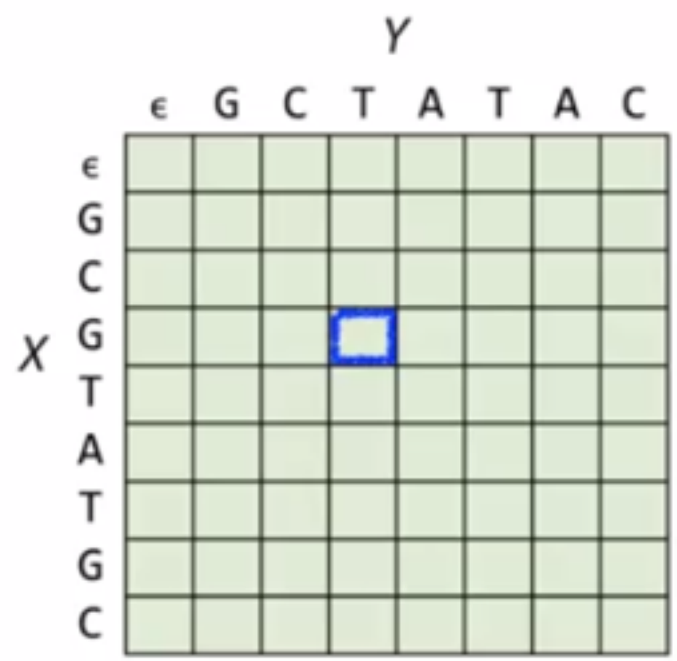


Repetitions

```
n = 0
def edDistRecursive(a, b):
    global n
    if len(a) == 0:
        return len(a)
    if len(b) == 0:
        return len(b)
    if a == 'Shake' and b == 'shake':
        n += 1
    delt = 1 if a[-1] != b[-1] else 0
    return min(edDistRecursive(a[:-1], b[:-1]) + delt,
               edDistRecursive(a[:-1], b) + 1,
               edDistRecursive(a, b[:-1]) + 1)
```

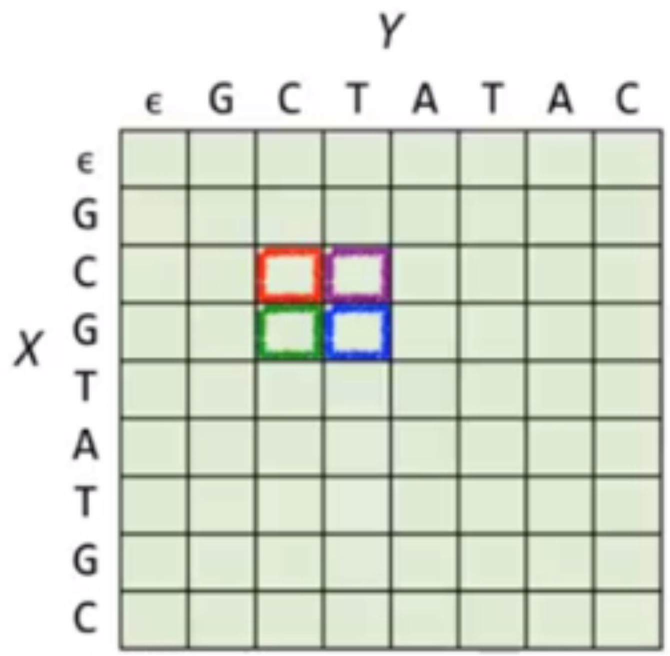
```
>>> edDistRecursive("Shakespeare", "shake spear")
3
>>> n
8989
```

Instance



$$\boxed{\mathbf{edist}(\alpha x, \beta y)} = \min \begin{cases} \mathbf{edist}(\alpha, \beta) + \delta(x, y) \\ \mathbf{edist}(\alpha x, \beta) + 1 \\ \mathbf{edist}(\alpha, \beta y) + 1 \end{cases}$$

Instance



$$\boxed{\mathbf{edist}(\alpha x, \beta y)} = \min \begin{cases} \boxed{\mathbf{edist}(\alpha, \beta) + \delta(x, y)} \\ \boxed{\mathbf{edist}(\alpha x, \beta) + 1} \\ \boxed{\mathbf{edist}(\alpha, \beta y) + 1} \end{cases}$$

Instance

Y

X	ϵ	G	C	T	A	T	A	C
ϵ	0	1	2	3				
G	1	0	1	2				
C	2	1	0	1				
G	3	1	1	1				
T								
A								
T								
G								
C								

$$\boxed{\mathbf{edist}(\alpha x, \beta y)} = \min \begin{cases} \boxed{\mathbf{edist}(\alpha, \beta) + \delta(x, y)} = 0 + 1 = 1 \\ \boxed{\mathbf{edist}(\alpha x, \beta) + 1} = 1 + 1 = 2 \\ \boxed{\mathbf{edist}(\alpha, \beta y) + 1} = 1 + 1 = 2 \end{cases}$$

End