

Design and Analysis of Algorithms

Assignment #3

1. Describe an $O(n)$ algorithm that, given a set of n distinct numbers (s) and a positive integer $k \leq n$, determines the k numbers in s that are closest to the median of s .

Sol.

```

Select(A, p, r, i)
    if p == r
        return A[p]
    x = median(A, p, r)
    q = partition(A, p, r, x)
    k = q - p + 1
    if i == k
        return A[q]
    else if i < k
        return select(A, p, q - 1, i)
    else
        return select(A, q + 1, r, i - k)
  
```

Median(A, p, r)

if $r - p < 5$

return partitions(A, p, r)

for $i \leftarrow p$ to r

SR = $i + 4$

if $SR > r$

SR = r

medS = partitions(A, i, SR)

swap $A[\text{medS}] \leftrightarrow A[p + \lfloor \frac{i-p}{5} \rfloor]$

return select(A, p, $p + \lfloor \frac{r-p}{5} \rfloor - 1$, $p + \frac{r-p}{10}$)

partition(A, p, r, x)

pV = $A[x]$

swap $A[r] \leftrightarrow A[x]$

SI = p

for $i \leftarrow p$ to $r-1$

if $A[i] < pV$

& swap $A[\text{SI}] \leftrightarrow A[i]$

SI++

swap $A[r] \leftrightarrow A[\text{SI}]$

return SI

Complexity: $O(n)$.

2. Find an optimal parenthesization of a matrix chain multiplication whose sequence of dimensions is $(9, 10, 9, 5, 12, 6)$.

Q.2

A	B	C	D	E
7×10	10×9	9×5	5×12	12×6

	A	B	C	D	E
A	A 0 7×10	AB 630 7×9	ABC 800 7×5	ABCD 1220 7×12	ABCDE 1370 7×6
B	—	B 0 10×9	BC 450 10×5	BCD 1050 10×12	BCDE 1110 10×6
C	—	—	C 0 9×5	CD 540 9×12	CDE 630 9×6
D	—	—	—	D 0 5×12	DE 360 5×6
E	—	—	—	—	E 0 12×6

$m[i, j]$

k-table

—	1	1	3	3
—	—	2	3	3
—	—	—	3	3
—	—	—	—	4
—	—	—	—	—

$s[i, j]$

→ To obtain the matrix A from A, we do not need to multiply anything.

→ Hence, the value would be zero.

→ Similar is the case with B, C, D and E.

A	B	C	D	E
7×10	10×9	9×5	5×12	12×6

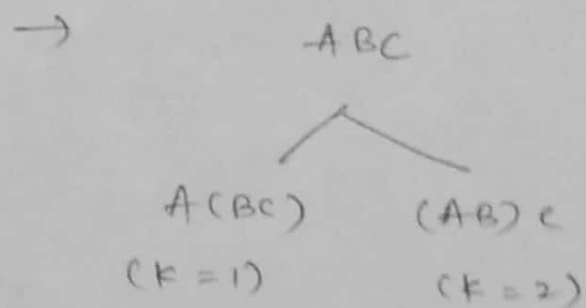
No. of multiplications for :

$$AB = 7 \times 10 \times 9 = 630 \quad (k=1)$$

$$BC = 10 \times 9 \times 5 = 450 \quad (k=2)$$

$$CD = 9 \times 5 \times 12 = 540 \quad (k=3)$$

$$DE = 5 \times 12 \times 6 = 360 \quad (k=4)$$

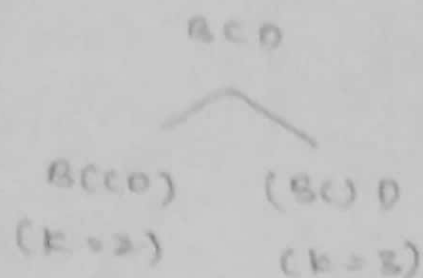


$$A(BC) = 0 + 450 + (7 \times 10 \times 5) = 800$$

$$(AB)C = 630 + 0 + (7 \times 9 \times 5) = 945$$

→ Since, 800 is the minimum value among 800 and 945, we consider $A(BC) \Rightarrow k=1$

→

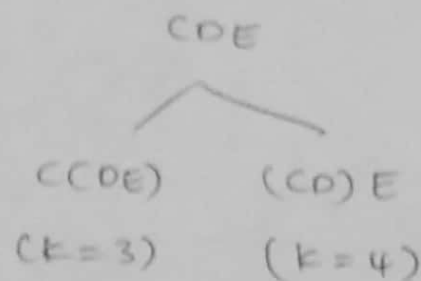


$$B(CD) = 0 + 540 + (10 \times 9 \times 12) = 1620$$

$$(BC)D = 450 + 0 + (10 \times 5 \times 12) = 1050$$

→ Since, 1050 is the minimum value among 1620 and 1050, we consider $(BC)D \Rightarrow k=3$

→

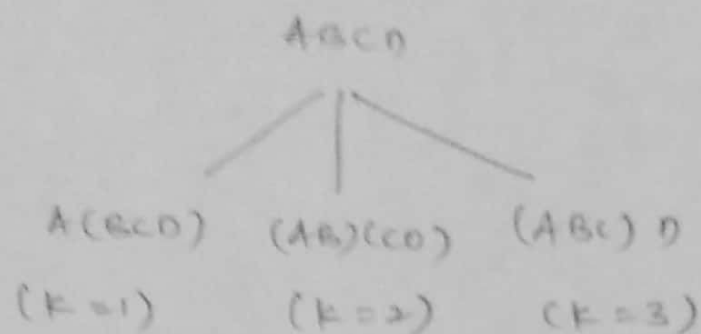


$$C(DE) = 0 + 360 + (9 \times 5 \times 6) = 630$$

$$(CD)E = 540 + 0 + (9 \times 12 \times 6) = 1188$$

→ Since, 630 is the minimum value among 630 and 1188, we consider $C(DE) \Rightarrow k=3$

→



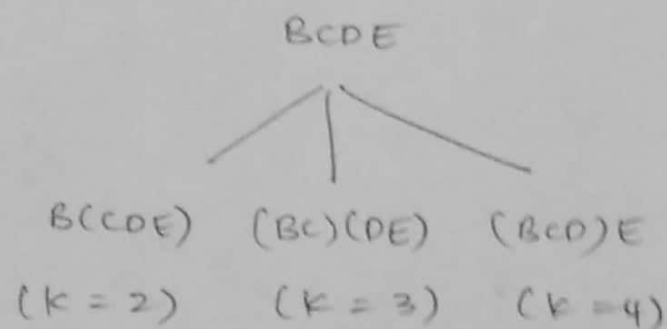
$$A(BCD) = 0 + 1050 + (7 \times 10 \times 12) = 1870$$

$$(AB)(CD) = 630 + 540 + (7 \times 9 \times 12) = 1926$$

$$(ABC)D = 800 + 0 + (7 \times 5 \times 12) = 1220$$

→ Since, 1220 is the minimum value among the three, we consider $(ABC)D \Rightarrow k=3$.

→



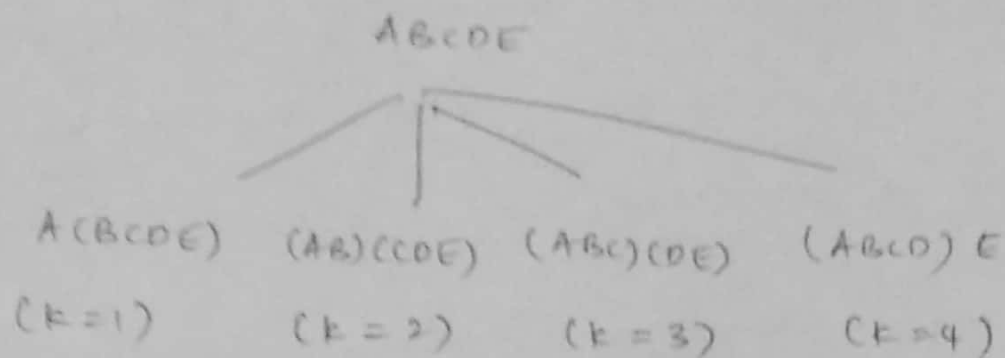
$$B(CDE) = 0 + 630 + (10 \times 9 \times 6) = 1170$$

$$(BC)(DE) = 450 + 360 + (10 \times 5 \times 6) = 1110$$

$$(BCD)E = 1050 + 0 + (10 \times 12 \times 6) = 1770$$

→ Since, 1110 is the minimum value among the three, we consider $(BC)(DE) \Rightarrow k=3$.

→



$$A(BCDE) = 0 + 1110 + (7 \times 10 \times 6) = 1530$$

$$(AB)(CDE) = 630 + 630 + (7 \times 9 \times 6) = 1638$$

$$(ABC)(DE) = 800 + 360 + (7 \times 5 \times 6) = 1370$$

$$(ABCD)E = 1220 + 0 + (7 \times 12 \times 6) = 1724$$

→ Since, 1370 is the minimum value among all the 4, we consider $(ABC)(DE) \Rightarrow k=3$

→ Hence, the paranthesization occurs as follows considering the k-table

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

$$\Rightarrow \underline{\underline{(A(BC))(DE)}}$$

3. Design an $O(n^2)$ dynamic programming algorithm to find a set of compatible activities such that the total amount of time the resource is used is maximized.

i	1	2	3	4	5	6	7	8	9	10	11
S(i)	2	3	5	6	7	9	10	12	13	14	16
F(i)	6	5	7	10	8	13	16	14	14	18	20
L(i)	1	1	2	2	3	4	4	4	5	6	6
P(i)	\emptyset	\emptyset	2	1	3	5	5	5	6	9	9

Soln

Compatibility ($A[n] = [], i, S(i), F(i), L(i), P(i)$)

if $i = 1$

$L = 1,$

$P = \emptyset$

return $A(i)$

for $i \leftarrow 2$ to n

for $j \leftarrow i-1$ ~~to~~ to 1

if $S[i] < F[j]$

$L(i) = L(j)$

if $L = 1$

else $P = \emptyset$

$P(i) = P(i-1)$

return $L(i), P(i)$

else

$$L(i) = L(i) + 1$$

$$P(i) = j$$

return $L(i), P(i)$

end

end

$$i = n$$

while $(i \neq \phi)$

add i to $A[n]$

$$i = P(i)$$

return A ;

Initial conditions and Sub-problem

$$i = 1$$

$$\Rightarrow L = 1, P = \phi$$

$$i = 2, j = 1$$

$$\Rightarrow S(2) < F(1) \text{ True}$$

$$L = 1$$

$$P = \phi$$

$$i = 3, j = 2$$

$$\Rightarrow S(3) < F(2) \text{ False}$$

$$L = 2$$

$$P = 2$$

Complexity

→ Since there are 2 nested 'for' loops, the time complexity would be $O(n^2)$.