```
        j = i+l;
        a[i][j] = k+1;
    }
}
```

```
    }
    inc++;
}
```

mcm

OBST (Optimal binary search tree)

mcm: A B C D E — 5 Matrices

$$mcm[i \ldots j] = \min \left\{ \sum_{k=i}^{k=j-1} mcm(i,k) + mcm(k+1,j) + i_r * j_c * k_c \right\}$$

$$ABCDE = \min \begin{cases} (A)(BCDE) \\ (AB)(CDE) \\ (ABC)(DE) \\ (ABCD)(E) \end{cases}$$

12345 =

(1,1)(2,5)
(1-2)(3-5)
(1-3)(4-5)
(1-4)(5,5)

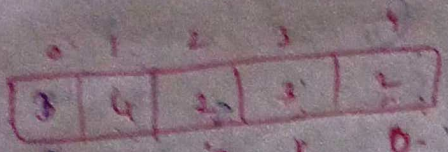
AD.

(i ... j) =

(i,i)(i+1 ... j)
(i,i+1)(i+2 ... j)
(i,i+2)(i+3 ... j)
(i,i+3)(i+4 ... j)

.
.

(i, j-1)(j, j)



i K j

mcm(1,4) = mcm(1,1) + mcm(2,4) + a[0] * a[1] * a[4] = 52
0 + 24 + (3 × 4 × 2) = 24

mcm(1,4) = mcm(1,2) + mcm(3,4) + a[0] * a[2] * a[4] = 48
24 + 12 + (3 × 2 × 2) = 12

mcm(1,4) = mcm(1,3) + mcm(4,4) + a[0] * a[3] * a[4]
0

mcm(1,4)

i j
k

A =

mcm(1,4)

1) A.(BCD)

(B) . CD 2×2 = 12
4×2 16
2×4 28

(A) (BCD)
3×4 4×2
12 16
24 24
24

0) mcm(2,4) = 28 mcm(2,2) + mcm(3,4)
0 + 12 +
2×4×2
16
28

2) mcm(1,2) = mcm(1,1) + mcm(2,
0 + 0
+ a(1×2×4)
a[i] + a[k] + a[j
8 = 24

mcm(3,4) = mcm(3,3) + mcm(4,4)
0 0
+ 3×2×2

```
mcm(i,j)
    if (i==j) {
    }   return 0;
    else {
        min = INT_max)
        for (k=i, k<j, k++)
        Val = mcm(i,k) + mcm(k+1, j)
                + a[i-1] * a[k] * a[j];
        if (val<min)
            min=val;
    }
} }
return min;
```

A B C
3×4 4×2 2×3

$D_{3\times 2}$ → (AB).C
A(BC)

A B C D

① (AA)(BCD) = 52

② (AB)(CD) = 48

③ (ABC)(DD) = 60

① (AA)(BCD) = (AA) + BCD
0 + 3

BCD → BB + (CD) + $B_r \times B_c \times D_c$
0 + (2×3×2) + 4×2×2
0 + (12) + (16)
= 28

BCD → B(CD) = 28 ✓ min
(BC)D = 41

(BC).D = (BC) + (D.D) + $B_r \times C_c \times D_c$
= (4×2×3) + (0) + (4×3×2)
= 24 + 0 + 24
= 41

∴ (AA)+(BCD) = 0 + 29 = 29
(24) + 12 = 52

② (AB)(CD) = (AB) + (CD) + $A_r \times B_c \times D_c$
(AA) (BB) (CC) (DD)
3×2×2
= 12
24 + 12 + 12 = 48

AB = AA + BB + $A_r \times A_c \times D_c$
= 0 + 0 + 3×4×2
= 0 + 0 + 24

CD = CC + DD + $C_r \times C_c \times D_c$
= 0 + 0 + 2×3×2
= 0 + 0 + 12

① $(AB)_{4\times2} + (DD)_{0} + A_{3\times3} \times C_{c} \times D_{c} = 60$

$ABC = AA + (BC) + A_{r} \times B_{c} \times C_{c}$

$= 0 + 4\times2\times3 + 3 * 4 \times 3$

$= 0 + 24 + 36$

$= 52$

$mcm(1,1) + mcm(2,5) + 1_r \times 1_c \times 5_c$

$mcm(1,2) + mcm(3,5) + 1_r \times 2_c \times 5_c$

$mcm(1,3) + mcm(4,5) + 1_r \times 3_c \times 5_c$

$mcm(1,4) + mcm(5,5) + 1_r \times 4_c \times 5_c$

$mcm(1,2345)$ $(A\,B)_c$

$k = i$

$mcm(i,j) + mcm(i+1,j) + i_r * i_c * j_c$

$mcm(i,j)$

$mcm(i,i+1) + mcm(i+2,j) + i_r * i+1_c * j_c$

$mcm(i,i+2) + mcm(i+3,j) + i_r * i+2_c * j_c$

$k = j-1$

$mcm(i,j-1) + mcm(j,j) + i_r * j-1_c * j_c$

$$mcm(i,j) = \min_{k=i}^{j-1} mcm(i,k) + mcm(k+1,j) + i_r * k_c * j_c$$

	0	1	2	3	4	5
a =	2	3	4	3	2	3

A	B	C	D	E
2×3	3×4	4×3	3×2	2×3

$i_r * k_c * j_c$

$a[i-1] * a[k] * a[j]$

Memoization :

```
mcm(i,j) {
if (a[i][j] != -1) {
    return a[i][j];
}
if (i==j) {
    return a[i][j]=0;
}
min = INT_MAX;
for (k=i; k<j; k++) {
    val = mcm(i,k) + mcm(k+1,j) +
          a[i-1] * a[k] * a[j];
    if (val < min)
        min = val;
    }
}
return a[i][j] = min;
}
```

Tabulation :

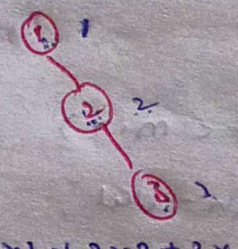
```
for (k=0; k<=n-1; k++) {
    for (i=1; i<=n-k; i++) {
        j = i+k;
        if (i==j) {
            a[i][j]=0
            continue;
        }
```

```
min = INT_MAX;
for (k=i; k<=j-1; k++){
    val = dp[i][k] + dp[k+1][j] + a[i-1]*a[k]*a[j];
    if (val < min) {
        min = val;
    }
}
dp[i][j] = min;
}
}
```

OBST

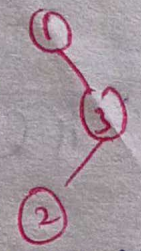
i	2	3

3	2	4



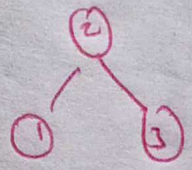
$8\times1 + 2\times2 + 3\times4$

$= 24$



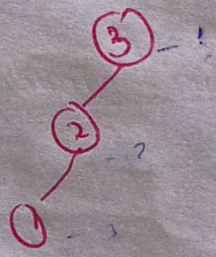
$1\times8 + 2\times4 + 3\times2$

$= 22$



$1\times2 + 2\times8 + 2\times4 =$

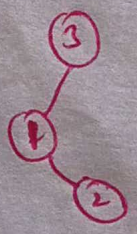
26



$3\times4 + 2\times2 + 3\times8$

$4 + 4 + 24$

$= 32$



$1\times4 + 2\times8 + 3\times2$

$= 4 + 16 + 6$

$= 26$

OBST

memoization

```
Obst(i,j){
    if(i==j){
        freq[i];
    }
    min = INT_MAX;
    for(k=i; k<=j; k++){
        val = obst(i,k-1) + obst(k+1,j) + sum(freq,i,j)
        if (val<min)
            min=val;
    }
    return min;
}
```

if a[i][j] != -1
return a[i][j]

return a[i][j] = min;

```
Sum(freq, i, j){
    int s=0
    for(k=i; k<=j; k++){
        s=s+freq[k];
    }
    return s;
}
```

Tabulation:

```
obst(i,j){
    for(l=0; l<n; l++){
        for(i=1; i<=n-l; i++){
            j = i+l;
            for(k=i; k<=j; k++){
                if(i==j){
                    d[i][j] = freq[i];
                    Continue;
                }
```

```
val = dp[i][k-1] + dp[k+1][j] + sum(freq, i, j)

if (val < min) {
    min = val;
}

return dp[i][j] = min;
}

sum(freq, i, j) {
    s = 0;
    for (k = i; i <= j; k++) {
    } s = s + freq[k];
    return s;
}
```

Greedy Approach:

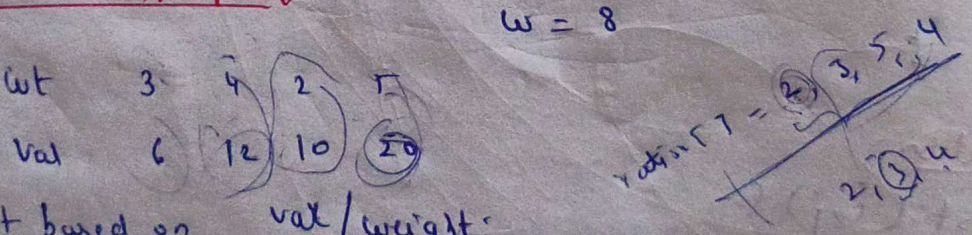
① Fractional Knapsack:

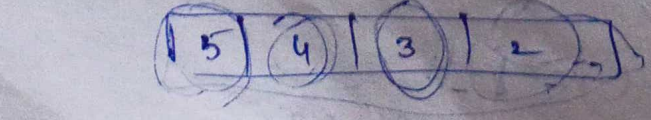
wt	3	4	2	5
val	6	12	10	20

sort first based on val/weight;

wt	2	5	4	3
val	10	20	12	6

W = 8

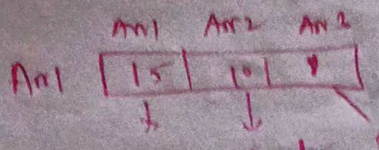


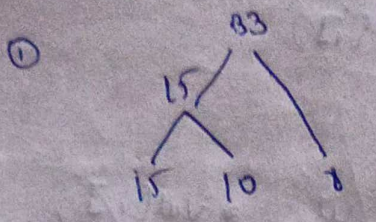


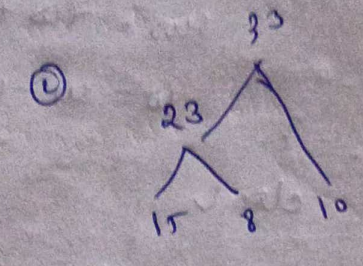
```
fractionalKnapSack (wt, val, n, w) {
    res = 0;
    for (i=0; i<n; i++) {
        if (wt[i] < w) {
            res = res + val[i];
            w = w - wt[i];
        }
        else {
            res = res + val[i] * w / wt[i];
            break;
        }
    }
    return res;
}
```

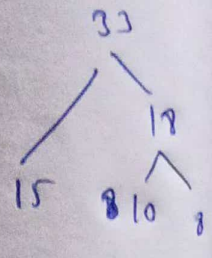
Greedy - (Optimal Merge Patterns)

We are given arrays with different no. of elements



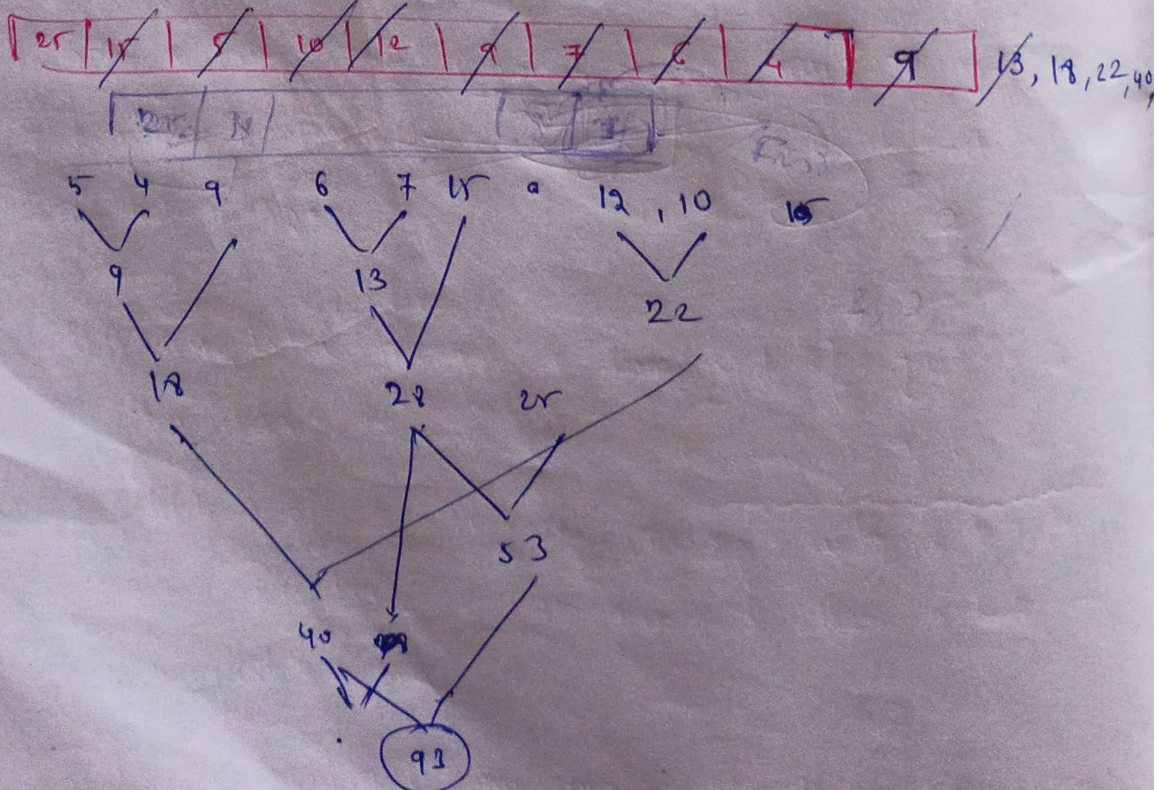






∴ we have to return minimum no. of operations to be done to merge the elements.

given n = 9 (9 arrays with different sizes)



∴ 93 + 40 + 53 + 29 + 13 + 22 + 18 + 1 = 276

① Find min in $0 \to n-1$ elements
② swap the min to with n^{th} place
③ find min in $n-1$ elements
④ swap the min to $n-1$ position
⑤ add $n + n-1$ position values
⑥ Now the result sum result in $n-1^{th}$ position
add it to global sum (which is result)
⑧ $n = n-1$;
⑨ and repeat the steps (1- 6) until n becomes <= 2

Huffman Encoding

Length:- we have length of character is 5,00,000
and each character has size 8 bit so it becomes
total 5,00,000 × 8 bit = 40,00,000 bits

Now- we charater are normally ~~from~~ 0~~-26~~ 26 (alphabets)
so we need to transfer the character from computer
A to B with encoded character.

Let us discuss:
Instead of sending large bits (eg: 40,00,000)
we ~~have~~ can have own code (encoding) to the
characters to be transfered..
Jam Assuming each character size be 5 bits
So Now, the total length is will become 5 × 50,10,000
= 25,00,000 bits

A	B	C	D	E	F	G	H	I	J
00000	00001	...	...	...	...	...	...	...	...

Encrypted code)

25,00,000 bit (00000 - 11111) = ~~130~~

00001 - ... decrypted

encrypted 00011 ...

~~00~~ 040 message

Code original

26 x 5 = 130

26 x 7 = 182

312

Total = 25,00,000 + 312

= 25,00,312 bits

Encoding

A	B	C	D	E	F	G	H	I
12	10	9	5	4	6	7	11	25

= 93

My message is having 93 characters in that 93

A = 12, B = 10, C = 9, D = 5 ..... I = 25

⇒ In order to send this 93 characters message on a network, the original size of my message becomes 93 x 8 = 744 bits as each character takes 8 bits of size.

⇒ But my total message is composed of just 9 characters, ie A, B, C, D ..... I (9 characters)

So, Instead of sending my message with 8 bits for each character, I can represent these 9 characters with just 4 bits.

⇒ I will encrypt my message with 4 bits encryption instead 8 bits.

⇒ The message size will become 93 x 4 = 372 bits along with the encrypted message, I have to sent decrypting codes in my message., So the length of decryption message will become ---

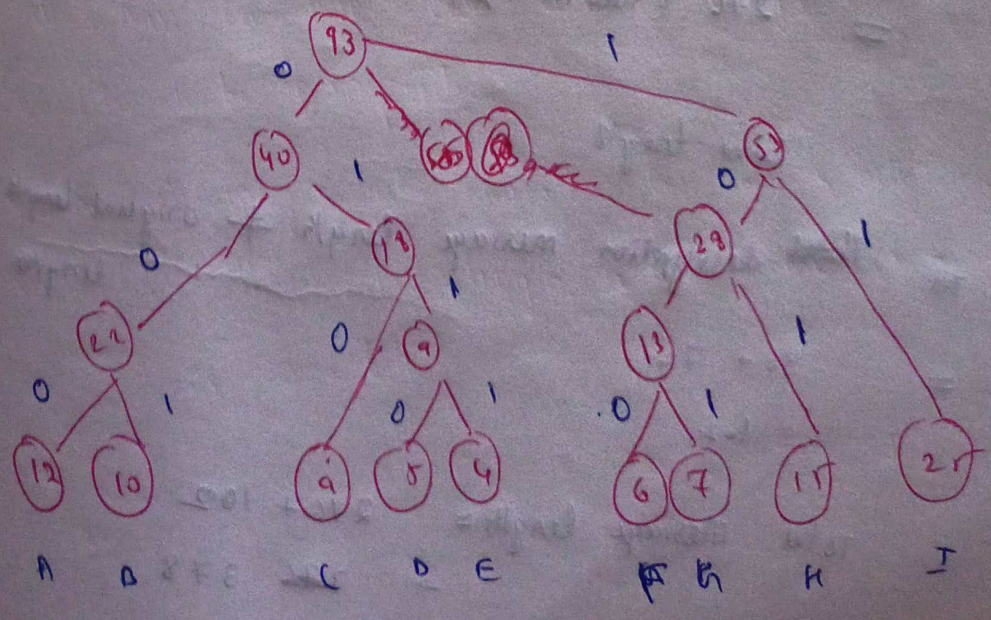
= 372 + 9x4 + 9x8 =
(encrypted codes) (original code)

= 480 bits

⇒ This 480 encrypted bits ~~we~~ message we got using Uniform encrypted message.

⇒ Instead of using ~~this~~ Uniform Encryption, we can use Variable encryption, the size of the message is further reduced this can be done using Huffman encoding...

⇒ For Variable encryption, for each character in above message, the encoding can be done using optimal merge pattern. i.e



Variable length code

A = 000
B - 001
C = 010
D - 0110
E - 0111
F - 1000
G - 1001
H - 101
I - 11

} encrypted new codes

With my new encrypted codes, to ~~Co~~ the size of the above message will become.

A	B	C	D	E	F	G	H	I
12	10	9	5	4	6	7	15	25
12x3	10x3	9x3	5x4	4x4	6x4	7x4	15x3	25x2
36	30	27	20	16	24	28	45	50

= 276 (raw message)

~~Total length~~

⇒ ~~Total~~ decryption message length + original ~~length~~ bits length.

= 80 + 72
= 102

Total Message length = 276 + 102
= ~~326~~ 378