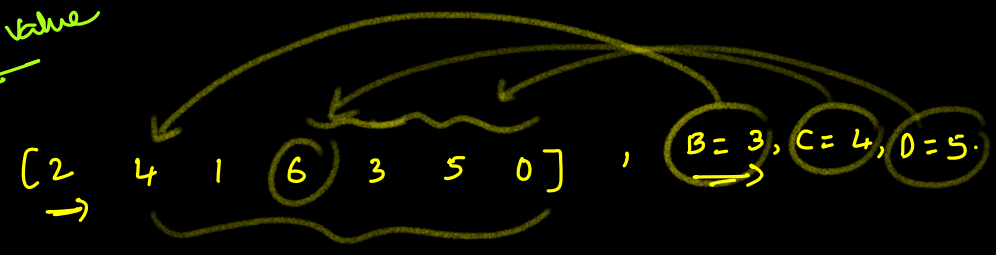
max Sum value




```
for (i=0; i<N; i++)
    for(j=i  ; j<N; j++)
        for (k=j  ; k<N; k++)
            maxValue = max (A[i]*B + A[j]*C + A[k]*D , maxValue)
```

m_1 is max → +ve

m_2 is min

B, C, D.

+ve +ve +ve

m1*B m1*C m1*D.

B C D

-ve -ve -ve

m_2*B m_2*C m_2*D.

If B is -ve

(min) * B or (max) * B.

[-21, 34, 3, 46, 8, -47, 47) , B=-13, C=10, D=9.

[273, 273, 273, 273, 273, 611, 611]

[273-210, 273+340, 273+340, 273+460, 273+460, 273+460, 273+460)

[63, 613, 613, 733, 733, 733, 733]

[63-189, 613+306, 613+306, 613+414, 613+414, 613+414, 613+414)

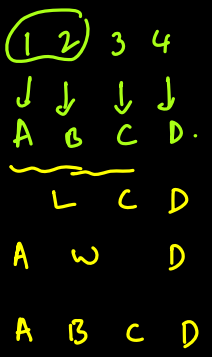
[-126, 919, 919, 147, 107, 127, 127]

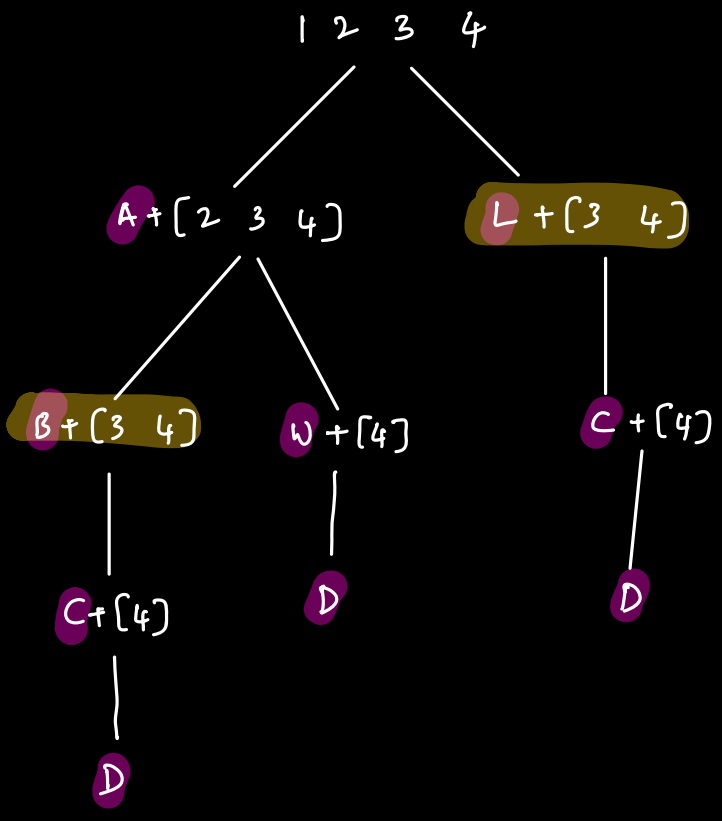
$\frac{46 \times 9}{414}$ 5

```java
public int solve(int[] A, int B, int C, int D) {

    int[] pf = new int[A.length];
    pf[0] = A[0]*B;
    for(int i=1; i<A.length; i++){
        pf[i] = Math.max(pf[i-1], A[i]*B);
    }

    pf[0] = pf[0]+C*A[0];
    for(int i=1; i<A.length; i++){
        pf[i] = Math.max(pf[i-1], pf[i]+C*A[i]);
    }

    pf[0] = pf[0]+D*A[0];
    for(int i=1; i<A.length; i++){
        pf[i] = Math.max(pf[i-1], pf[i]+D*A[i]);
    }

    return pf[A.length-1];
}
```

[2 4 1 6 3 5 0] , B = 3, C = 4, D = 5.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26





dp[i] → No. of ways to decode chars from (0-i)

dp[i] = dp[i-1] + dp[i-2]

if [i i-1] < 26.

```
boolean canDecodeTwoChars(String word, int i)
    if (word.charAt(i-1) > 2)
        return false
    if (word.charAt(i-1) == 2 && word.charAt(i) > 6)
        return false
    return true
```

```
int waysToDecode(String word, int i)
    if (i < 0)
        return 0
    if (i<=0)
        return 1
    if(dp[i]!= -1)
        int count = waysToDecode(word, i-1)
        if (canDecodeTwoChars(i, word))
            count += waysToDecode(word, i-2)
        dp[i] = count
    return dp[i]
```