

1.

$$A1=5*10$$

$$A2=10*3$$

$$A3=3*12$$

$$A4=12*5$$

$$A5=5*50$$

$$A6=50*6$$

BUILD THE TABLE

| m | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|-----|-----|-----|------|------|
| 1 | 0 | 150 | | | | |
| 2 | | 0 | 360 | | | |
| 3 | | | 0 | 180 | | |
| 4 | | | | 0 | 3000 | |
| 5 | | | | | 0 | 1500 |
| 6 | | | | | | 0 |

$m(1,1)=0$ because it is just $A1$ and no need for calculation

$m(1,2)=150=5*10*3$ from $A1, A2$ the rest are for the same reason

| s | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | | 1 | | | | |
| 2 | | | 2 | | | |
| 3 | | | | 3 | | |
| 4 | | | | | 4 | |
| 5 | | | | | | 5 |
| 6 | | | | | | |

$s(1,2)$ is 1 because the multiplication of $A1*A2$ is based on $A1$

$$m(4,3) = A_1 \cdot (A_2 \cdot A_3) \quad \text{or} \quad (A_1 \cdot A_2) \cdot A_3$$

$$= m[1,1] + m[2,3] + 5 \times 6 \times 2$$

$$= 960$$

$$= m[1,2] + m[3,3] + 5 \times 5 \times 2$$

$$= 330$$

$$\text{so } m(1,3) = 330 \quad s(1,3) = 2$$

| | | | |
|--------------|--------------|--------------|--------------|
| A_1 | A_2 | A_3 | A_4 |
| 5×6 | 6×3 | 3×2 | 2×5 |
| A_5 | A_6 | | |
| 6×5 | 5×6 | | |

$$m(2,4) = A_2 \cdot (A_3 \cdot A_4) \quad \text{or} \quad (A_2 \cdot A_3) \cdot A_4$$

$$= m[2,2] + m[3,4] + 10 \times 3 \times 5$$

$$= 180 + 150 = 330$$

$$= m[2,3] + m[4,4] + 10 \times 2 \times 5$$

$$= 360 + 600 = 960$$

$$\text{so } m(2,4) = 330 \quad s(2,4) = 2$$

$$m(3,5) = A_3 \cdot (A_4 \cdot A_5) \quad \text{or} \quad (A_3 \cdot A_4) \cdot A_5$$

$$= m[3,3] + m[4,5] + 3 \times 2 \times 5$$

$$= 300 + 1800 = 2100$$

$$= m[3,4] + m[5,5] + 3 \times 5 \times 5$$

$$= 180 + 750 = 930$$

$$\text{so } m(3,5) = 930 \quad s(3,5) = 4$$

$$m(4,6) = A_4 \cdot (A_5 \cdot A_6) \quad \text{or} \quad (A_4 \cdot A_5) \cdot A_6$$

$$= m[4,4] + m[5,6] + 12 \times 5 \times 6$$

$$= 1800 + 3600 = 5400$$

$$= m[4,5] + m[6,6] + 12 \times 5 \times 6$$

$$= 3000 + 3600 = 6600$$

$$\text{so } m(4,6) = 5400 \quad s(4,6) = 4$$

$$m(1,4) = (A_1 \cdot A_2) \cdot A_3 \cdot A_4 \quad \text{or} \quad (m_1 \cdot m_2) \cdot (m_3 \cdot m_4)$$

$$= m[1,3] + m[4,4] + 5 \times 12 \times 5$$

$$= 630$$

$$= m[1,2] + m[3,4] + 5 \times 3 \times 5$$

$$= 150 + 180 + 75 = 405$$

$$= m[1,1] + m[2,4] + 5 \times 1 \times 5$$

$$= 330 + 150 = 480$$

$$\text{so } m(1,4) = 405 \quad s(1,4) = 2$$

$$m(2,5) = (m_2 \cdot m_3) \cdot m_4 \cdot m_5 \quad \text{or} \quad (m_2 \cdot m_3) \cdot (m_4 \cdot m_5) \quad \text{or} \quad m_2 \cdot (m_3 \cdot m_4 \cdot m_5)$$

$$= m[2,4] + m[5,5] + 6 \times 5 \times 5$$

$$= 330 + 2250 = 2580$$

$$= m[2,3] + m[4,5] + 10 \times 2 \times 5$$

$$= 360 + 3000 + 600 = 3960$$

$$= m[2,2] \times m[3,5] + 10 \times 3 \times 5$$

$$= 930 + 1500 = 2430$$

$$\text{so } m(2,5) = 2430 \quad s(2,5) = 2$$

$$m(3,6) = m_3 \cdot m_4 \cdot m_5 \cdot m_6 \\ (A_3 \cdot A_4 \cdot A_5) A_6 \\ = m[3,5] + m[6,6] + 3 \times 5 \times 6 \\ = 180 + 36 \\ = 216$$

$$s(3,6) = 1770$$

$$m_3 = m_4 \cdot m_5 \cdot m_6 \\ (A_3 \cdot A_4) A_5 A_6 \\ = m[3,4] + m[5,6] + 3 \times 5 \times 6 \\ = 180 + 180 + 90 \\ = 450$$

$$s(3,6) = 4$$

$$m_3 = (A_4 \cdot A_5 \cdot A_6) \\ = m[3,3] + m[4,6] + 3 \times 12 \times 6 \\ = 180 + 24 \\ = 204$$

$$m(1,5) = (A_1 \cdot A_2 \cdot A_3 \cdot A_4) A_5 \\ = m[1,4] + m[5,5] + 4 \times 5 \times 5 \\ = 40 + 125 \\ = 165$$

$$m_1 = (A_2 \cdot A_3 \cdot A_4 \cdot A_5) \\ = m[1,1] + m[2,5] + 5 \times 6 \times 5 \\ = 2430 + 2520 \\ = 4950$$

$$m(1,5) = (A_1 \cdot A_2) \cdot (A_3 \cdot A_4 \cdot A_5) \\ = m[1,2] + m[3,5] + 5 \times 3 \times 5 \\ = 15 + 180 + 75 \\ = 270$$

$$m_1 = (A_1 \cdot A_2 \cdot A_3 \cdot A_4) A_5$$

$$m_1 = (A_1 \cdot A_2 \cdot A_3) \cdot (A_4 \cdot A_5) \\ = m[1,3] + m[4,5] + 5 \times 12 \times 5 \\ = 330 + 300 + 300 \\ = 930$$

$$s(1,5) = 1655, \quad s(1,5) = 4$$

$$m(2,6) = (A_2 \cdot A_3 \cdot A_4 \cdot A_5) A_6 \\ = m[2,5] + m[6,6] + 10 \times 5 \times 6 \\ = 2410 + 3600 + 300 \\ = 6370$$

$$m_2 = (A_3 \cdot A_4 \cdot A_5 \cdot A_6) \\ = m[2,2] + m[3,6] + 10 \times 3 \times 6 \\ = 1770 + 180 = 1950$$

$$m(2,6) = (A_2 \cdot A_3) \cdot (A_4 \cdot A_5 \cdot A_6) \\ = m[2,3] + m[4,6] + 10 \times 12 \times 6 \\ = 360 + 1860 + 720 \\ = 2940$$

$$m_2 = (A_2 \cdot A_3 \cdot A_4) \cdot (A_5 \cdot A_6) \\ = m[2,4] + m[5,6] + 10 \times 5 \times 6 \\ = 330 + 1500 + 300 \\ = 2130$$

$$s(2,6) = 1950$$

$$s(2,6) = 2$$

$$m(1,0) = A_0(A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6) \quad \text{or} \quad (A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5) \cdot A_6$$

$$= m(1,1) + m(2,6) + 5 \times 6 \times 6 = 160 + 360 + 180 = 900$$

$$= m(1,0) + m(6,6) + 5 \times 5 \times 6 = 160 + 150 + 150 = 460$$

$$\text{or } (A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6) \quad \text{or} \quad (A_1 \cdot A_2 \cdot A_3 \cdot A_4) \cdot (A_5 \cdot A_6)$$

$$= m(1,2) + m(3,6) + 5 \times 3 \times 6 = 405 + 100 + 90 = 595$$

$$= 160 + 770 + 200 = 2030$$

$$= 405 + 100 + 150 = 655$$

$$\text{or } (A_1 \cdot A_2 \cdot A_3) \cdot (A_4 \cdot A_5 \cdot A_6)$$

$$= m(1,3) + m(4,6) + 5 \times 2 \times 6 = 330 + 160 + 360 = 850$$

$$\text{So } m(1,6) = 2030 \quad s(1,6) = 2$$

so

| m | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|-----|-----|-----|------|------|
| 1 | 0 | 150 | 330 | 405 | 1655 | 2010 |
| 2 | | 0 | 360 | 330 | 2430 | 1950 |
| 3 | | | 0 | 180 | 930 | 1770 |
| 4 | | | | 0 | 3000 | 1860 |
| 5 | | | | | 0 | 1500 |
| 6 | | | | | | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| s | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | | 1 | 2 | 2 | 4 | 2 |
| 2 | | | 2 | 2 | 2 | 2 |
| 3 | | | | 3 | 4 | 4 |
| 4 | | | | | 4 | 4 |
| 5 | | | | | | 5 |
| 6 | | | | | | |

$s(1,6)=6$

so $(a_1*a_2)(a_3*a_4*a_5*a_6)$

$s(1,2)=1$ only 2 elements, ignore

$s(3,6)=4$

so $(a_1*a_2)(a_3*a_4)(a_5*a_6)$

so the final answer is $(a_1*a_2)((a_3*a_4)(a_5*a_6))$

2.

s1 | "AGGTAB"
s2 | "GXTXAYB"

| | " | A | G | G | T | A | B |
|---|---|---|---|---|---|---|---|
| " | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| G | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| X | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| T | 0 | 0 | 1 | 1 | 2 | 2 | 2 |
| X | 0 | 0 | 1 | 1 | 2 | 2 | 2 |
| A | 0 | 1 | 1 | 1 | 2 | 3 | 3 |
| Y | 0 | 1 | 1 | 1 | 2 | 3 | 3 |
| B | 0 | 1 | 1 | 1 | 2 | 3 | 4 |

The example and table is from <https://www.youtube.com/watch?v=ASoaQg66foQ>

if current column=current row, which means that the current two substring have the same last char and can eliminate them, and **move to the left-up cell**

| | " | A | G | G | T | A | B |
|---|---|---|---|---|---|---|---|
| " | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| G | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| X | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| T | 0 | 0 | 1 | 1 | 2 | 2 | 2 |
| X | 0 | 0 | 1 | 1 | 2 | 2 | 2 |
| A | 0 | 1 | 1 | 1 | 2 | 3 | 3 |
| Y | 0 | 1 | 1 | 1 | 2 | 3 | 3 |
| B | 0 | 1 | 1 | 1 | 2 | 3 | 4 |

if current column != current row, which means that current two substring have different last chars, so the length of LCS won't change, **we move to the cell with same number.**

| | " | A | G | G | T | A | B |
|---|---|---|---|---|---|---|---|
| " | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| G | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| X | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| T | 0 | 0 | 1 | 1 | 2 | 2 | 2 |
| X | 0 | 0 | 1 | 1 | 2 | 2 | 2 |
| A | 0 | 1 | 1 | 1 | 2 | 3 | 3 |
| Y | 0 | 1 | 1 | 1 | 2 | 3 | 3 |
| B | 0 | 1 | 1 | 1 | 2 | 3 | 4 |

| | " | A | G | G | T | A | B |
|---|---|---|---|---|---|---|---|
| " | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| G | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| X | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| T | 0 | 0 | 1 | 1 | 2 | 2 | 2 |
| X | 0 | 0 | 1 | 1 | 2 | 2 | 2 |
| A | 0 | 1 | 1 | 1 | 2 | 3 | 3 |
| Y | 0 | 1 | 1 | 1 | 2 | 3 | 3 |
| B | 0 | 1 | 1 | 1 | 2 | 3 | 4 |

and use stringBuilder to remember the cells that goes left-up, which is G T A B

pseudo code

```
String reverse_LCS(table t, int row, int col){
    StringBuilder temp = new StringBuilder();
    if (row == 0 || col == 0)
        return temp.toString();
    if (t[row, 0] == t[0, col])
        reverse_LCS(t, row - 1, col - 1);
        temp.append(t[row, 0]);
    else if (t[row - 1, col] == t[row, col])
        reverse_LCS(t, row - 1, col);
    else
        reverse_LCS(t, row, col - 1);
}
```

cause there is no loop, there is at most $m+n$ recursions. so the time complexity is $m+n$

3.

Firstly copy these numbers into a new array and sort the array , $O(n \log n)$

Then run LCS algorithm on the sorted array and the original sequence of n numbers, $O(n^2)$

Then the result will must be sorted longest increasing subsequence of the original sequence

4.

YES,

e_k choose the activity that have the latest start time

e_{k-1} to e_1 choose the compatible activity with latest start time at that stage

So it is greedy, because we choose our best choice at every stage

take $e_1', e_2', e_3' \dots e_k'$ as an optimal schedule

if $e_k' \neq e_k$, then we can replace e_k' with e_k , because e_k have later start time than e_k' and get an optimal schedule

The rest are for same reason.

So it is optimal.

5.

| | | | | | |
|--------|---|---|---|---|---|
| item# | 1 | 2 | 3 | 4 | 5 |
| profit | 3 | 6 | 8 | 2 | 3 |
| weight | 5 | 3 | 2 | 1 | 3 |

pseudo code

```
int[][] arr = new int[n][W]; //initialize 2-d array
int max_Profit(int[][] arr, int[] weightArr, int[] profitArr, int n, int W){
    if (arr[n][W] != null) return arr[n][W]; //reuse the buffer arr to save time
    int result = 0;
    if (n == 0 || W == 0){
        result = 0;
    } //base case
    else if (weightArr[n] > W) //the weight of current item > current capacity, skip
        result = maxProfit(arr, weightArr, profitArr, n-1, W);
    else{
        result = Math.max(
            maxProfit(arr, weightArr, profitArr, n-1, W), //skip this item
            profitArr[n] + maxProfit(arr, weightArr, profitArr, n-1, W-weightArr[n]) //pick this item
        );
        arr[n][W] = result;
    }
}
```



```
return result;  
}
```

we only need to build the $n \times W$ 2D arr (which means we only need to call this function $n \times w$ times including recursion) , and there is no loop in the function, so it is $O(nW)$ time

6.

in Huffman code, the leaf represents the char, path to the leaf represents the code. Assume there is a char 0001, then in full tree, it will have a sibling node 0000, 000 need two nodes to differ these two chars. If we remove 0000, then there is no need to differ. then 000 is enough to represent the original char. Then 0001 isn't the optimal prefix code.

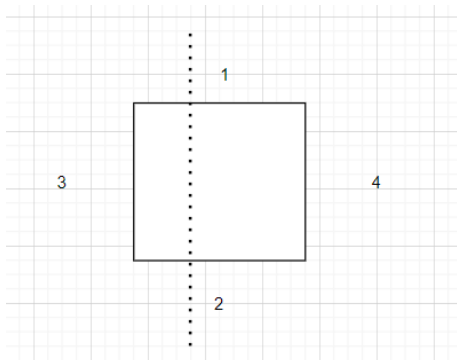
7.

assume we've already known the minimum spanning tree. The G and G' are just the graphs after adding some new edges. //so that two graphs share the same minimum spanning tree. And we need to prove e is not part of minimum spanning tree. So we can add it and remove it without influencing the minimum spanning tree.

Cause the e is an edge in **cycle**. We use the Kruskal's method to find the minimum spanning tree. However, we keep all edges of this cycle. And then the final graph will be a minimum spanning tree+ the cycles including e . Cause the property of cycle, even remove one edge, the vertices are still connected, so if we want to get the minimum tree. We must remove the edge e . So e is not part of minimum spanning tree

8. No, at this step "**Finally, select the minimum-weight edge in E that crosses the cut (V_1, V_2)** ", we cannot ensure the edges that are removed are the maximum edge in cycle that won't influence the minimum spanning tree.

For example



we will remove 2, then the minimum will be 1,3,4
but in fact the minimum is 1 2 3