1. 9 /10

A1=5*10

A2=10*3

A3=3*12

A4=12*5

A5=5*50

A6=50*6

what method are you using ? how does it work ? what is the recursive definition that it gives ? What do the tables below represent ?

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

A_1 A_2 A_3 A_4

5×10 10×3 3×12 12×5

A_5 A_6

5×50 50×6

$m(1,3) = A_1 \cdot (A_2 \cdot A_3)$ or $(A_1 \cdot A_2) \cdot A_3$

$= m[1,1] + m[2,3] + 5\times10\times12$ $m[1,2] + m[3,3] + 5\times3\times12.$

$= 960$ $= 330.$

so $m(1,3) = 330.$ $s(1,3) = 2.$

$m(2,4) = A_2 \cdot (A_3 \cdot A_4)$ or $(A_2 \cdot A_3) \cdot A_4.$

$= m[2,2] + m[3,4] + 10\times3\times5$ or $m[2,3] + m[4,4] + 10\times12\times5$

$= 180 + 150 = 330$ $= 360 + 600 = 960.$

so $m(2,4) = 330$ $s(2,4) = 2$

$m(3,5) = A_3 \cdot (A_4 \cdot A_5)$ or $(A_3 \cdot A_4) \cdot A_5$

$= m[3,3] + m[4,5] + 3\times12\times50$ $m[3,4] + m[5,5] + 3\times5\times50$

$= 3000 + 1800 = 4800$ $= 180 + 750 = 930$

so $m(3,5) = 930$ $s(3,5) = 4.$

$m(4,6) = A_4 \cdot (A_5 \cdot A_6)$ or $(A_4 \cdot A_5) \cdot A_6$

$= m[4,4] + m[5,6] + 12\times5\times6$ $m[4,5] + m[6,6] + 12\times50\times6$

$= 1500 + 360 = 1860$ $= 3000 + 3600 = 6600.$

so $m(4,6) = 1860$ $s(4,6) = 4$

$m(1,4) = (A_1 \cdot A_2 \cdot A_3) \cdot m_4$ or $(m_1 \cdot m_2) \cdot (m_3 \cdot m_4)$ or $m_1 \cdot (m_2 \cdot m_3 \cdot m_4)$

$= m[1,3] + m[4,4] + 5\times12\times5$ $= m[1,2] + m[3,4] + 5\times3\times5$ $= m[1,1] + m[2,4] + 5\times10\times5$

$= 630$ $= 150 + 180 + 75 = 405$ $= 330 + 250 = 580$

so $m(1,4) = 405$ $s(1,4) = 2.$

$m(2,5) = (m_2 \cdot m_3 \cdot m_4) \cdot m_5$ or $(m_2 \cdot m_3) \cdot (m_4 \cdot m_5)$ or $m_2 \cdot (m_3 \cdot m_4 \cdot m_5)$

$= m[2,4] + m[5,5] + 10\times5\times50$ $= m[2,3] + m[4,5] + 10\times12\times50$ $= m[2,2] \times m[3,5] + 10\times3\times5$

$= 330 + 2500 = 2830$ $= 360 + 3000 + 6000 = 9360$ $= 930 + 1500 = 2430$

so $m(2,5) = 2430$ $s(2,5) = 2$

$m(3,6) = \cancel{m_3 \cdot m_4 \cdot m_5 \cdot m_6}$ or $\cancel{m_3 \cdot m_4 \cdot m_5 \cdot m_6}$ or $A_3 \cdot (A_4 \cdot A_5 \cdot A_6)$

$(A_3 \cdot A_4 \cdot A_5) A_6$

$= m[3,5] + m[6,6] + 3\times5\times6$

$= 830 + 800$

$= 1830$

$(A_3 \cdot A_4)(A_5 \cdot A_6)$

$= m[3,4] + m[5,6] + 3\times5\times6$

$= 180 + 1500 + 90$

$= 1770$

$= m[3,3] + m[4,6] + 3\times12\times6$

$= 1860 + 216$

$= 2076$

so $m(3,6) = 1770$ $\quad s(3,6) = 4$

$m(1,5) = (A_1 \cdot A_2 \cdot A_3 \cdot A_4) A_5$ or $A_1 \cdot (A_2 \cdot A_3 \cdot A_4 \cdot A_5)$

$= m[1,4] + m[5,5] + 5\times5\times50$

$= 405 + 1250$

$= 1655$

$= m[1,1] + m[2,5] + 5\times10\times50$

$= 2430 + 2500$

$= 4930$

$m(1,5) = (A_1 \cdot A_2) \cdot (A_3 \cdot A_4 \cdot A_5)$ or $\cancel{A_1 \cdot (A_2 \cdot A_3 \cdot A_4) \cdot A_5}$ or $(A_1 \cdot A_2 \cdot A_3) \cdot (A_4 \cdot A_5)$

$= m[1,2] + m[3,5] + 5\times3\times50$

$= 150 + 830 + 750$

$= 1830$

$= m[1,3] + m[4,5] + 5\times12\times50$

$= 330 + 3000 + 3000$

$= 6330$

$\cancel{m(1,5)=}$ so $m(1,5) = 1655$, $\quad s(1,5) = 4$

$m(2,6) = (A_2 \cdot A_3 \cdot A_4 \cdot A_5) A_6$ or $A_2 \cdot (A_3 \cdot A_4 \cdot A_5 \cdot A_6)$

$= m[2,5] + m[6,6] + 10\times50\times6$

$= 2430 + 3000 = 5430$

$= m[2,2] + m[3,6] + 10\times3\times6$

$= 1770 + 180 = 1950$

$m(2,6) = (A_2 \cdot A_3) \cdot (A_4 \cdot A_5 \cdot A_6)$ or $(A_2 \cdot A_3 \cdot A_4) \cdot (A_5 \cdot A_6)$

$= m[2,3] + m[4,6] + 10\times12\times6$

$= 360 + 1860 + 720$

$= 2940$

$= m[2,4] + m[5,6] + 10\times5\times6$

$= 330 + 1500 + 300$

$= 2130$

so $m(2,6) = 1950$ $\quad s(2,6) = 2$

$m[1,6] = A_1\cdot(A_2\cdot A_3\cdot A_4\cdot A_5\cdot A_6)$ or $(A_1\cdot A_2\cdot A_3\cdot A_4\cdot A_5)\cdot A_6$

$= m[1,1] + m[2,6] + 5\times10\times6$ $\quad = m[1,5] + m[6,6] + 5\times50\times6$

$= 1950 + 300 = 2250$ $\quad = 1655 + 1500 = 3155$.

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

$= m[1,2] + m[3,6] + 5\times3\times6$ $\quad = m[1,4] + m[5,6] + 5\times5\times6$.

$= 150 + 1770 + \overset{90\quad 2010}{\cancel{80 = 3000}}$ $\quad = 405 + 1500 + 150 = \cancel{2055}\ 2055$

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

$= m[1,3] + m[4,6] + 5\times12\times6$

$= 330 + 1860 + 360$

$= 2550$

So $m[1,6] = \cancel{2055}$ 2010 $\quad s[1,6] = 2$.

so

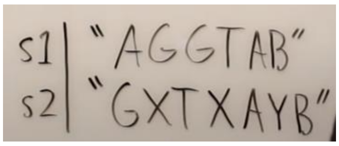
m	1	2	3	4	5	6
1	0	150	330	405	1655	2010
2		0	360	330	2430	1950
3			0	180	930	1770
4				0	3000	1860
5					0	1500
6						0

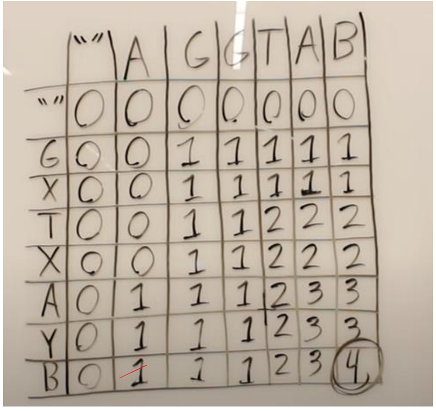
s	1	2	3	4	5	6
1		1	2	2	4	2
2			2	2	2	2
3				3	4	4
4					4	4
5						5
6						

s(1,6)=6
so (a1*a2)(a3*a4*a5*a6)
s(1,2)=1   only 2 elements, ignore
s(3,6)=4
so(a1*a2)(a3*a4)(a5*a6)

so the final answer is (a1*a2)((a3*a4)(a5*a6))

2. 8/10





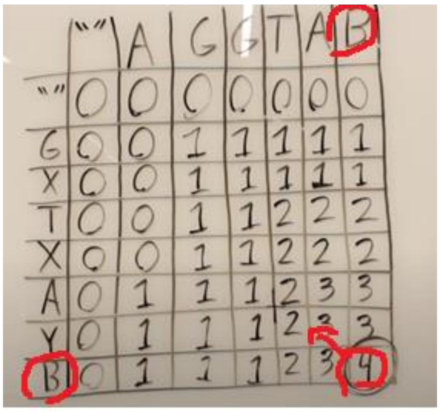
	""	A	G	G	T	A	B
""	0	0	0	0	0	0	0
G	0	0	1	1	1	1	1
X	0	0	1	1	1	1	1
T	0	0	1	1	2	2	2
X	0	0	1	1	2	2	2
A	0	1	1	1	2	3	3
Y	0	1	1	1	2	3	3
B	0	1	1	1	2	3	4

The example and table is from https://www.youtube.com/watch?v=ASoaQq66foQ

bad name of variable (maybe "current row header")

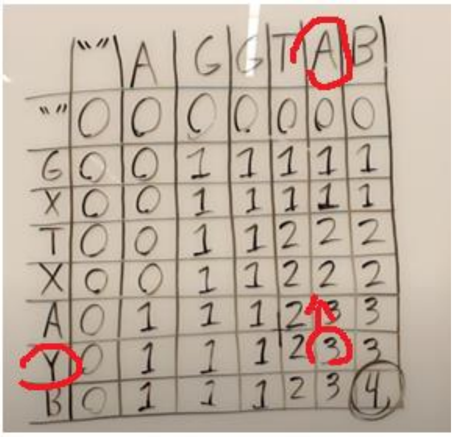
if current column=current row, which means that the current two substring have the same last char and can eliminate them, and **move to the left-up cell** ✓

(no points removed for this of course :))



	""	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'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, rol -1, col -1);    √
                temp.append(t[row,0]);
        else if (t[row-1,col] == c[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

bad explanation. Should be : because at most ~~only~~ one recursive call is made each function call, and it lowers either row by one, either col by one, or both, and base case is at $i=0$, $j=0$, and the rest is constant time

⇒ worst case = $O(m+n)$

3. 10 /10

Firstly copy these numbers into a new array and sort the array , O (nlogn)

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

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

4. 4 /10

YES,

ek choose the activity that have the latest start time

ek-1 to e1 choose the compatiable acativity with latest start time at that stage

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

What? seems like

take e1',e2',e3'...ek' as an optimal schedule

if ek'!= ek, then we can replace ek' with ek, because ek have later start time than ek' and get an optimal schedule → true but write this more formally.

The rest are for same reason.

So it is optimal. optimal substructure property?

5. 10 /10

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

pseudo code Very difficult to read, next time just take a picture of your code in your programming editor

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

3 / 10

just an algo to build an optimal prefix tree, we are talking about optimal prefix trees in general, which may not be constructed by that algo

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.

technically true. But explain this in a more formal, general manner, not just for one example:
Assume our tree is not full. So, there exists some node x which is not the root, with a prefix P = <p_1, p_2, ... p_{x - 1}, p_x> which has no sibling (ie there doesnt exist anyone with a prefix <p_1, p_2, ... , p_{x-1}, not(p_x)>. Then delete x's parent and move x and its subtree in its place. Then all nodes that are x's descendants can still be differentiated with the differences they have from x's subtree, and there are no other children of x's parent that need any prefix code. All other nodes are unaffected by this move, so they also keep their valid prefix. So our prefix tree is valid for the original alphabet. Then, it has a more optimal cost because... (continue this proof yourself)

7. 1/10

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.

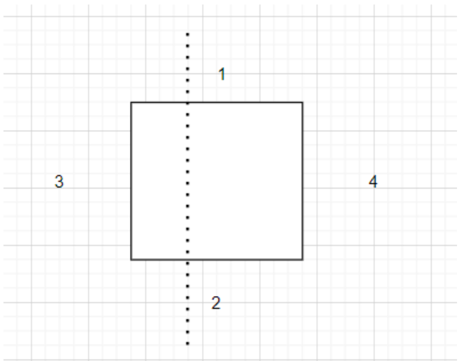
it can be (what if all edges on cycle cost the same?

then you need to prove if this is optimal or not: kruskal is optimal but does not guarantee you get to do this.

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

10/10

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