Homework 7

Daoyu Wang PB21030794

October 28

Contents

1 15.2-1

Implement **MATRIX CHAIN ORDER**. The result is as follows:

array m The lowestcost way to compute A_{16} is stored in array $m[7][7]$:

	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

array s The best order of matrix chain is stored in array $s[7][7]$:

	1	**2**	**3**	**4**	**5**
1	1	2	2	4	2
2		2	2	2	2
3			3	4	4
4				4	4
5					5

order The order of matrix chain is $((A_1A_2)((A_3A_4)(A_5A_6)))$

2 15.2-5

$$
\begin{aligned}
\sum_{i=1}^{n}\sum_{j=i}^{n} R(i,j) &= \sum_{l=2}^{n}\sum_{1}^{n-l+1}(i+l-1-i)*2 \\
&= 2\sum_{l=2}^{n}(l-1)(n-l+1) \\
&= 2\sum_{l=1}^{n-1} l(n-l) \\
&= 2n \cdot \frac{n(n-1)}{2} - 2 \cdot \frac{n(n-1)(2n-1)}{6} \\
&= n(n^2 - n - \frac{2n^2}{3} + n - \frac{1}{3}) \\
&= \frac{n^3-n}{3}
\end{aligned}
$$

3 15.3-2

Recursive call tree of **MERGE-SORT** is as follows:

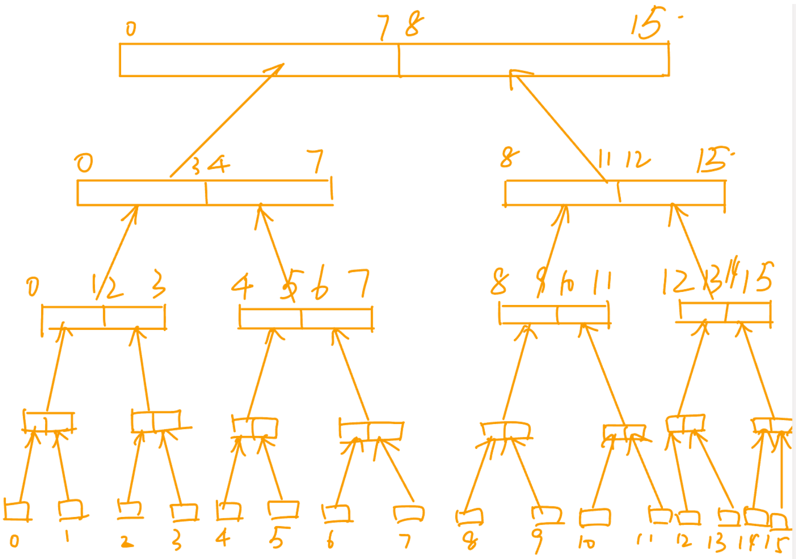

Figure 1: Recursive tree

Memoization here doesn't work because there is no sub-problem depends on any other sub-problems. Anyway, a good divide-and-conquer algorithm such as MERGE-SORT has no overlapping sub-problem.

4 15.3-4

Assume that we have a matrix chain: A_1, A_2, A_3, A_4 whose sequence of dimensions is $< a, b, c, d, e >$. c is the smallest among the sequence.

Capulet's method According to Professor Capulet, who believe the optimal solution to the matrix chain multiplication problem can be dealed greedily. So the optimal order is $((A_1A_2)(A_3A_4))$ whose cost is $abc + cde + ace$

Another way We also have another method to divide the matrix chain: $(((A_1A_2)A_3)A_4)$ whose cost becomes $abd + acd + ade$

Compare To refute the professor's point, we can establish the following inequality:

$$abc + cde + ace > abc + acd + ade \tag{1}$$
$$cde + ace > acd + ade \tag{2}$$

Assume that $a = \alpha c, b = \beta c, d = \delta c, e = \epsilon c$. Obviously $\alpha, \beta, \delta, \epsilon > 1$:

$$cd(e - a) > ae(d - c) \tag{3}$$
$$\frac{1}{\alpha} - \frac{1}{\epsilon} > 1 - \frac{1}{\delta} \tag{4}$$

Then take $\alpha = 1.2, \delta = 3, \epsilon = 12$ and $c = 5, b = 10$, we can get the following parameter list:

$$< a, b, c, d, e >=< 6, 10, 5, 15, 60 >$$

This not only satisfies the formula (4), but effectively refuted the professor's point of view because the cost of Professor Capulet is $abc+cde+ace = 6600$ while the cost of another way (maybe not the optimal solution) is $abc+acd+ade = 5850$.

5 15.4-1

LCS recursive memoization table is as follows:

	0	1 / 1	2 / 0	3 / 0	4 / 1	5 / 0	6 / 1	7 / 0	8 / 1
0	0	0	0	0	0	0	0	0	0
1 / 0	0	0 $\uparrow$	1 $\nwarrow$	1 $\nwarrow$	1 $\leftarrow$	1 $\nwarrow$	1 $\leftarrow$	1 $\nwarrow$	1 $\leftarrow$
2 / 1	0	1 $\nwarrow$	1 $\uparrow$	1 $\uparrow$	2 $\nwarrow$	2 $\leftarrow$	2 $\nwarrow$	2 $\leftarrow$	2 $\leftarrow$
3 / 0	0	1 $\uparrow$	2 $\nwarrow$	2 $\nwarrow$	2 $\uparrow$	3 $\nwarrow$	3 $\leftarrow$	3 $\nwarrow$	3 $\leftarrow$
4 / 1	0	1 $\nwarrow$	2 $\uparrow$	2 $\uparrow$	3 $\nwarrow$	3 $\uparrow$	4 $\nwarrow$	4 $\leftarrow$	4 $\nwarrow$
5 / 1	0	1 $\nwarrow$	2 $\uparrow$	2 $\uparrow$	3 $\nwarrow$	3 $\uparrow$	4 $\nwarrow$	4 $\uparrow$	5 $\nwarrow$
6 / 0	0	1 $\uparrow$	2 $\nwarrow$	3 $\nwarrow$	3 $\uparrow$	4 $\nwarrow$	4 $\uparrow$	5 $\nwarrow$	5 $\uparrow$
7 / 1	0	1 $\nwarrow$	2 $\uparrow$	3 $\uparrow$	4 $\nwarrow$	4 $\uparrow$	5 $\nwarrow$	5 $\uparrow$	6 $\nwarrow$
8 / 1	0	1 $\nwarrow$	2 $\uparrow$	3 $\uparrow$	4 $\nwarrow$	4 $\uparrow$	5 $\nwarrow$	5 $\uparrow$	6 $\nwarrow$
9 / 0	0	1 $\uparrow$	2 $\nwarrow$	3 $\nwarrow$	4 $\uparrow$	5 $\nwarrow$	5 $\uparrow$	6 $\nwarrow$	6 $\uparrow$

$$LCS(<1,0,0,1,0,1,0,1>, <0,1,0,1,1,0,1,1,0>) = <0,1,0,1,0,1>$$

6 15.4-4

Since every new **row** or **column** in LCS recursive memoization table is generated **only** based on its **upper row** or **left column**. Assume that the size of column is larger than the size of row so $min(m,n) = m$.

$2 \times min(m,n)$ Consider two array of length m, one is current row and another is its upper row. when calculating the current row, we only need data in its upper row. after calculating, we only need to change the current row to upper row, and this process can be finished by pointers. According to this method, we only need space of $2min(m,n) + O(1)$.

$min(m,n)$ The above method places the updated data in a new array. However, we can place the updated data directly in the original address, because for a specific table item, it only depends on the three table items on its left, top, and upper left. When we place the update data in the original address, all we lose is the upper left item message of the next item that needs to be updated. Fortunately, we only need to use an additional variable "pre" to record the overwritten item to solve this problem. According to this method, we only need space of $min(m,n) + O(1)$.