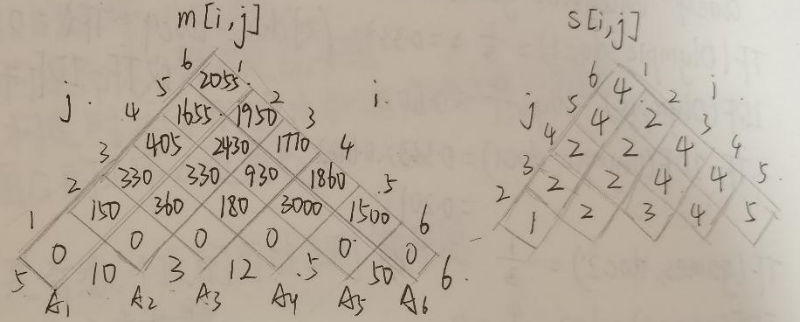
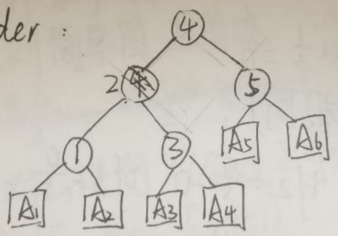
(1).



∴final order:



(2). ① 有 ~~n-2~~ n 个顶点，~~$\frac{(n-2)(n-1)}{2}$条边~~，$\frac{n(n-1)}{2}$条边．

其中有 ~~n-2~~ n-1 条树边，~~$\frac{(n-2)(n-3)}{2}$~~ $\frac{(n-1)(n-2)}{2}$ 条前向边．

~~②~~.

② chain(4) → chain(3) → chain(2) → chain(1).

(3). X: 1 0 0 1 0 1 0 1

Y: 0 1 0 1 1 0 1 1 0

		Y: 0	1	0	1	1	0	1	1	0
	0	0	0	0	0	0	0	0	0	0
X: 1	0	0	1	1	1	1	1	1	1	1
0	0	1	1	2	2	2	2	2	2	2
0	0	1	1	2	2	2	3	3	3	3
1	0	1	2	2	3	3	3	4	4	4
0	0	1	2	3	3	3	4	4	4	5
1	0	1	2	3	4	4	4	5	5	5
0	0	1	2	3	4	4	5	5	5	6
1	0	1	2	3	4	5	5	6	6	6

∴最长公共子序列长度为6，为100110或101010或101011或001011或 ~~001101~~ 001101或101101或010101或001010

（4）思路：设数组为 a[0:n]，用数组 b[0:n]纪录以 a[i] (0<= i< n)，为结尾元素的最长递增子序列的长度。则对于序列中第一个数 a[0]，显然有 b[0]=1，之后可以从左向右按顺序计算 a[i]（i=1，2，3，…… ，n）作为结尾元素时，最长递增子序列的长度 b[i]。当计算到 a[i]的最长递增子序列时，应该选择比 a[i]小的数中以该数结尾的递增子序列长度最长的那个数，作为子序列的倒数第二个数，即 b[i]=max{b[j]+1}（0≤ j < i 且 a[j] < a[i]）;如果序列 a[0:i-1]中所有的数均不小于 a[i]，则令 b[i]=1，即以 a[i]为结尾元素时的最长递增子序列只包含 a[i]本身。

伪代码：

```
LongestSubstr(a[0:n]){
	for( i = 1 to n){
		max = 0;
		for( j = 0 to i){
			if(a[j] < a[i]){
				if(max < b[j]){
					max = b[j];
				}
			}
		}
		b[i] = max + 1;
}
```

最后找到 b 数组中的最大值（下标记为 k）即为最长递增子序列的长度。从 a[k]开始，从右向左依次遍历数组 a[n]。对于某个下标 l（0 ≤ l < k）,如果 l 满足 a[l] < a [k]且 b[l] = b[k] - 1，则 a[l]可以作为最长递增子序列的倒数第二个数。以此类推，可以重复直至得到最长递增子序列。