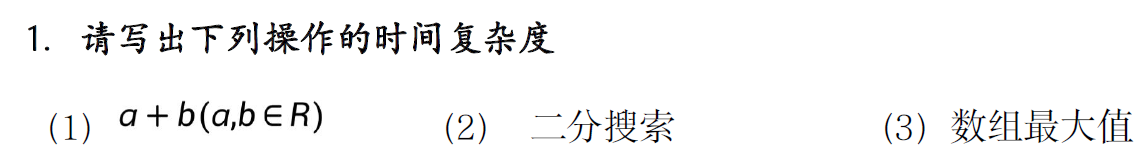
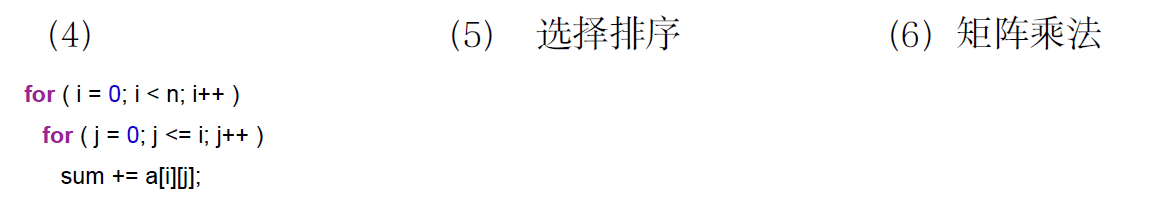
**数据结构 第一次作业**

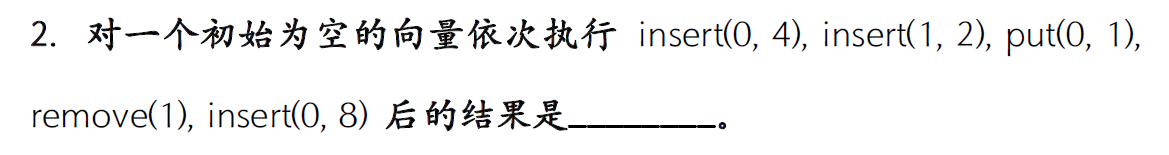
徐浩博 2020010108



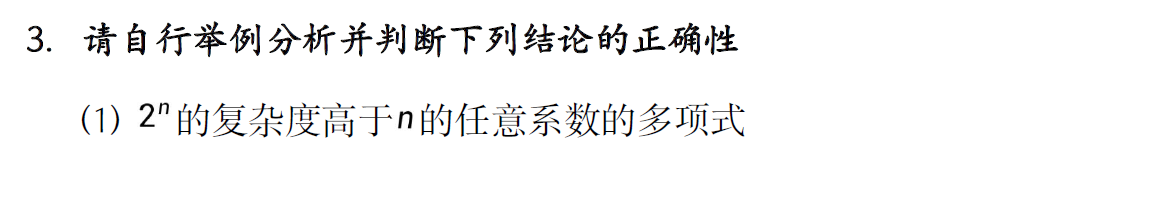
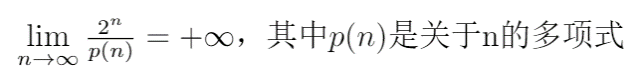
**O(1)**  **O(logn)**  **O(n)**



**O(n2)**   **O(n2)**  **n×m与m×k矩阵相乘O(nmk)**



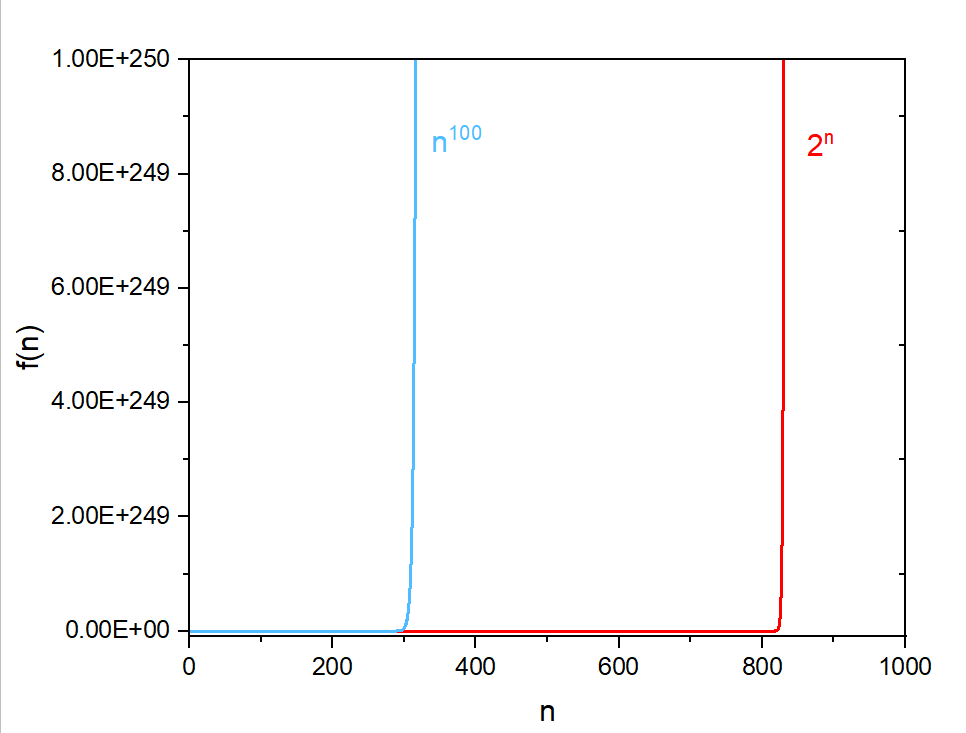
**8, 1**

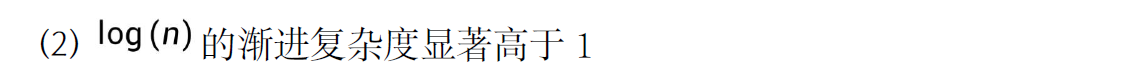


解：可以证明 ，因此存在n0，使得对任意n0，有2n≥p(n)，因此由渐进复杂度的定义，2n的渐进复杂度高于n的任意系数的多项式。

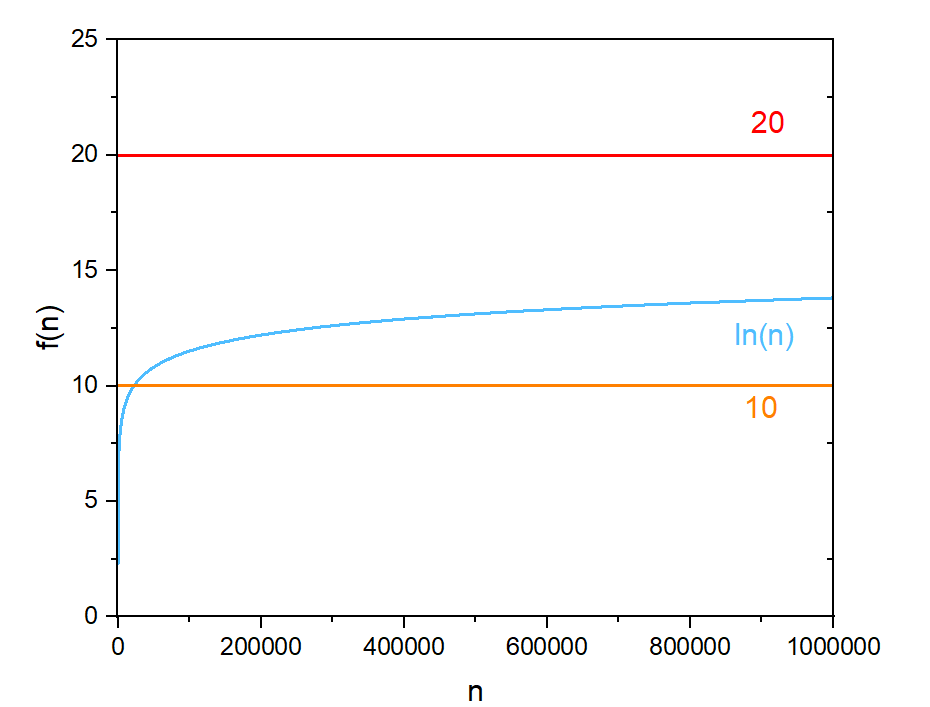
然而实际上，当n的系数、指数较大且n较小时，在一定范围内，仍有n多项式时间开销大于2n。下图对比了2n与n100的函数图像，虽然通过数学的方法可以得出在n≥1e6时恒有2n≥n100的关系[[1]](#footnote-1)，但由图像能够明显看出，在n≤800的范围内时间开销上有2n≤n100。

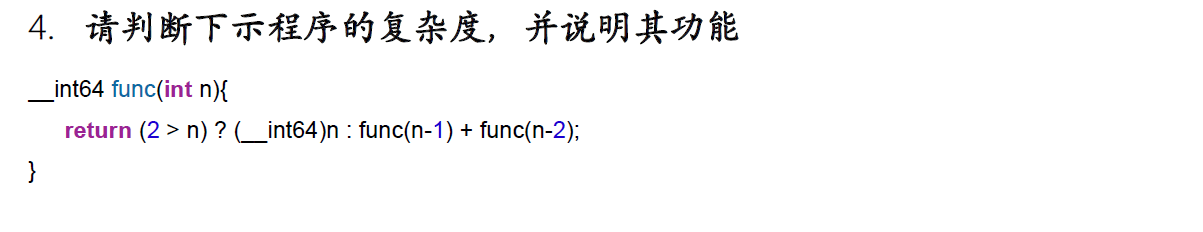
因此，虽然2n的渐进复杂度高于n的任意系数的多项式，但在实际问题中，时间开销是否一定更高，仍需要再具体分析。

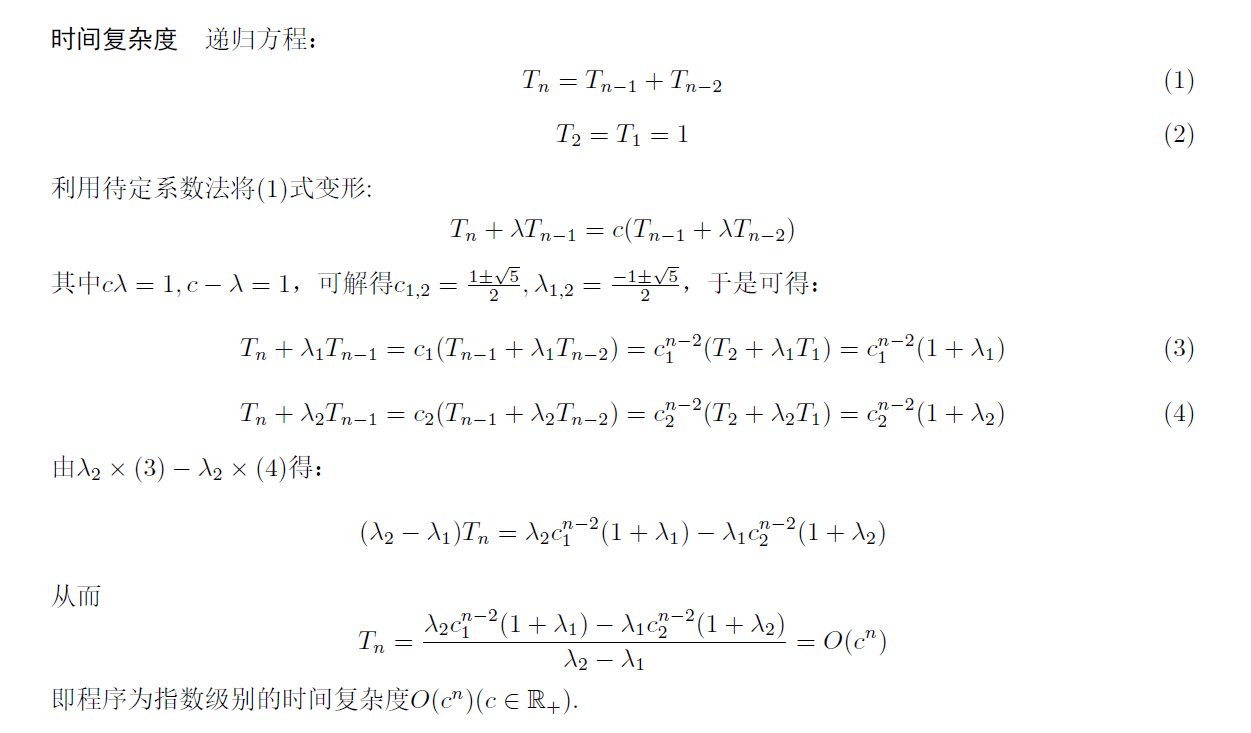


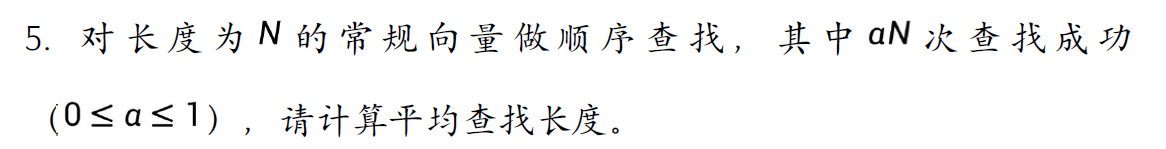


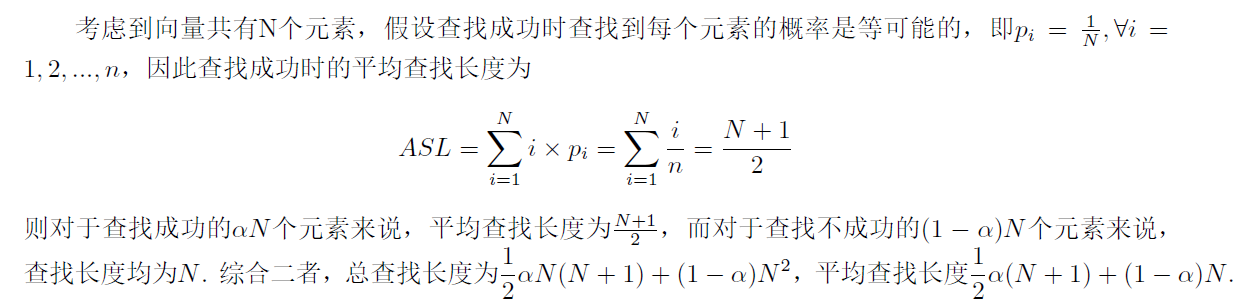
解：由渐进复杂度定义，对于任何实数c∈**R**，均存在n0使得任意n≥n0均有log(n)≥c，因此log(n)渐进复杂度大于O(1)。

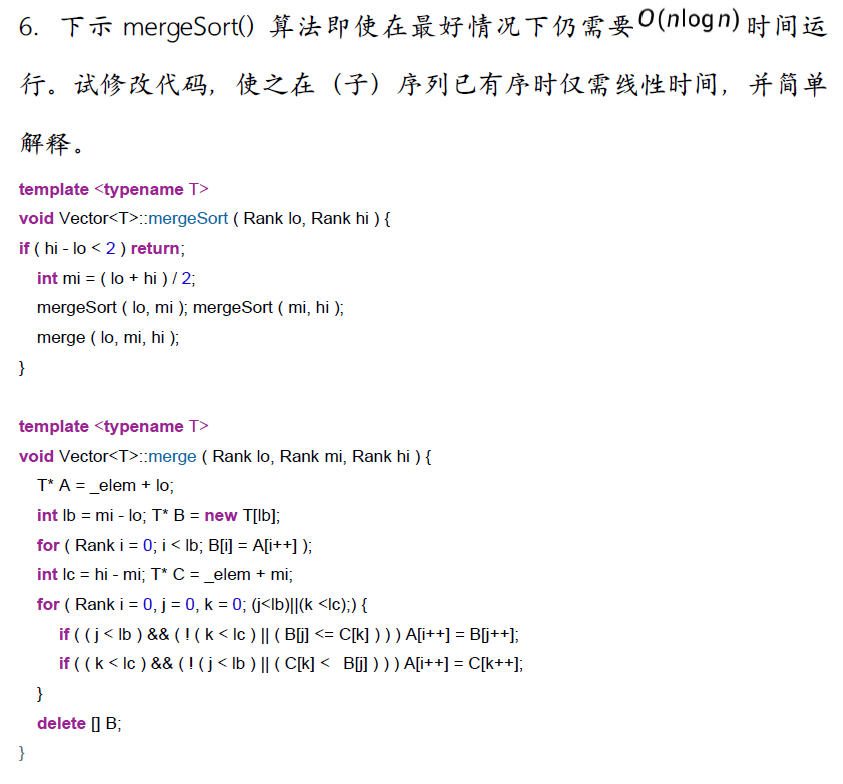
 但本题询问“是否显著大”，而这一点需要具体问题具体分析。下图绘制的是ln(n)的图像，虽然能够看出当n≥5e4时即有ln(n)≥10，但即使到了n≈1e6的情况下，仍有ln(n)≤20，可以认为时间开销仍没有显著大于1。然而，若将ln(n)换作10000ln(n)，那么在一定程度上可以认为时间开销显著大于1。因此，针对“logn渐进复杂度是否显著高于1”这个问题，仍需要结合具体情况具体分析。



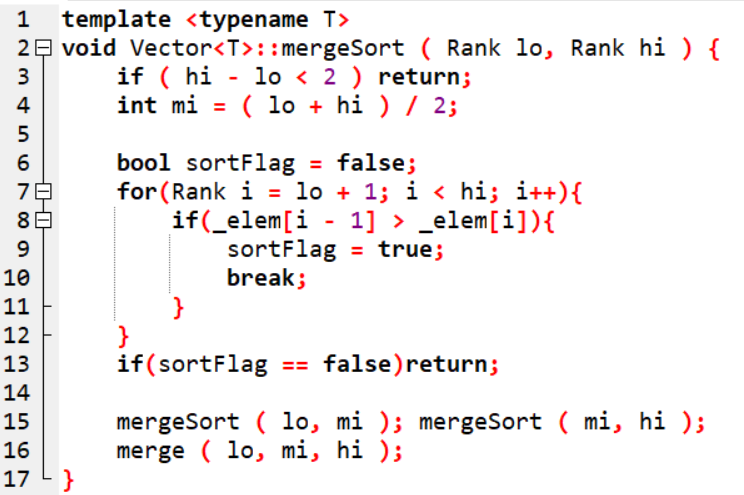
解： 



解：



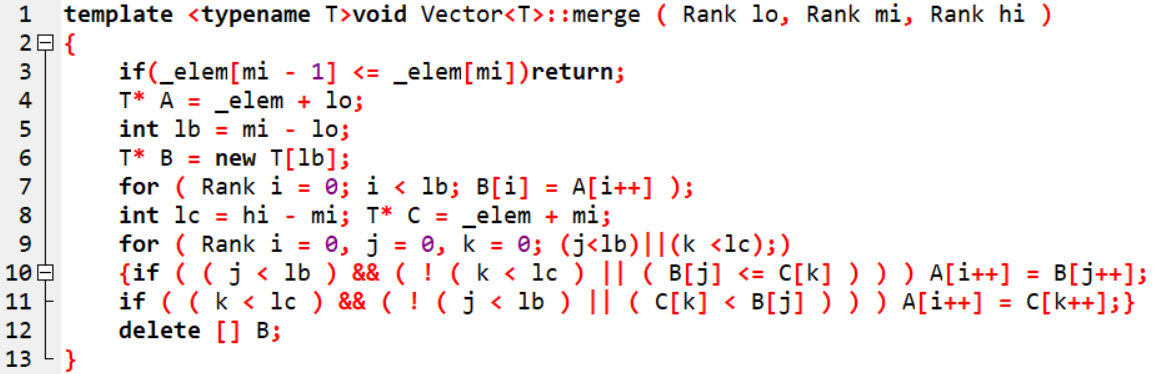
解：修改mergeSort函数为：



具体修改内容为：**在mergeSort内新增上图6-13行的内容**。当程序递归运行到lo-hi部分的子序列时，加入一个判断该序列是否有序的机制，如果序列有序，则不再进行该序列子列的排序，直接退出函数。

首先，加入这个机制使得有序（子）序列只消耗线性时间：这是因为判断有序只需要遍历子序列内容；发现有序则不进行后续排序，因此如果有序，时间开销只有线性的遍历。

其次，该机制没有提高算法原先的复杂度。算法原先的merge函数需要合并两个子序列的元素，时间开销是线性的；而判断是否有序的机制最差的时间复杂度也是线性的，因此并没有提高算法原先的复杂度，反而可能因为减少了有序序列的排序这一时间开销而降低时间复杂度。

另外还有一种做法，修改merge函数为：

具体修改内容为：**在merge内新增上图第3行的内容**，这样可以直接合并子列。在该序列有序时（假设该序列长度为n），若把递归过程化作一棵二叉树，则第k层有2k个子列，则此层计算次数为2k；计算总递归过程，一共的计算量为1+2+22+……+2^(log2 n)=O(n)，为线性复杂度。

1. 欲使2n≥n100，只需nln2≥100ln(n)，而0.5ln(n)<n0.5，故只需n0.5≥200/ln2，因此可取n≥1e6，此时必有2n≥n100. [↑](#footnote-ref-1)