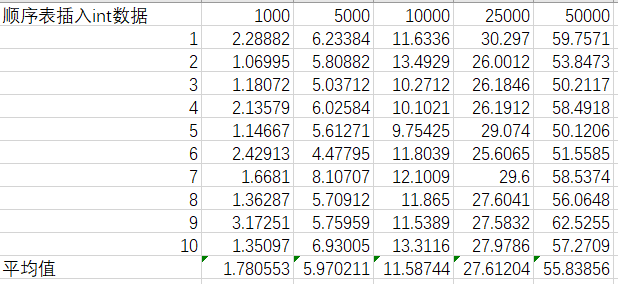
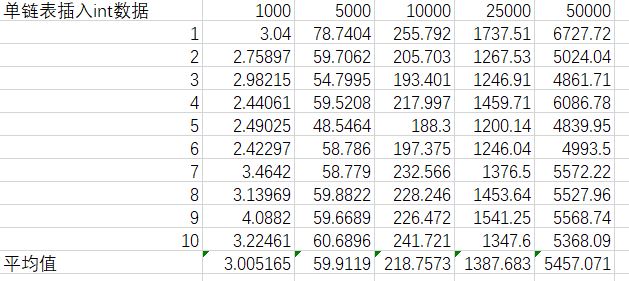
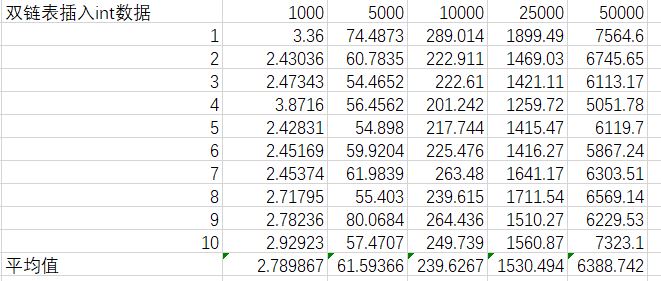
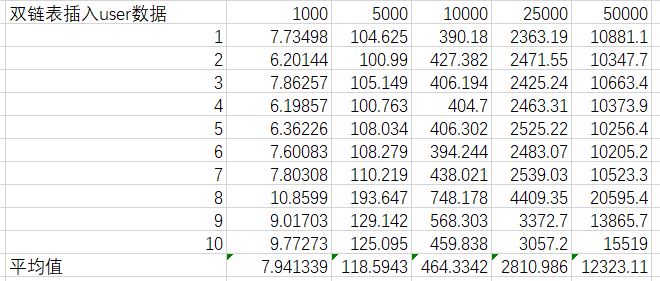
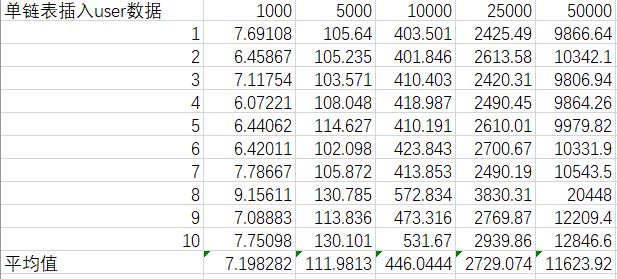
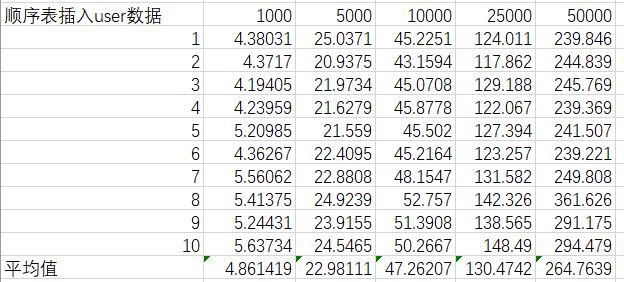
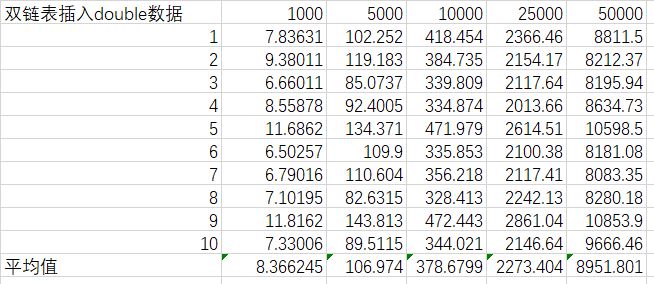
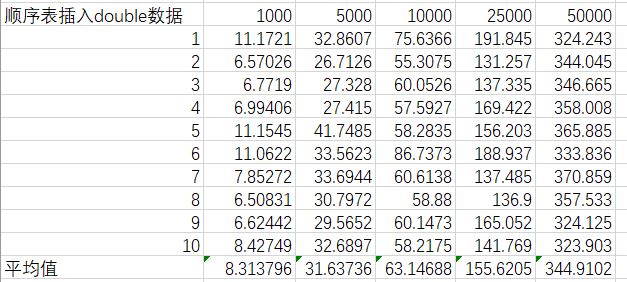
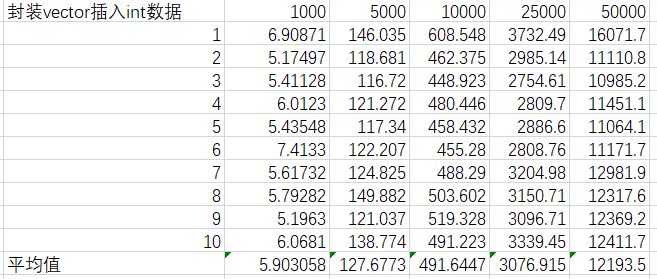
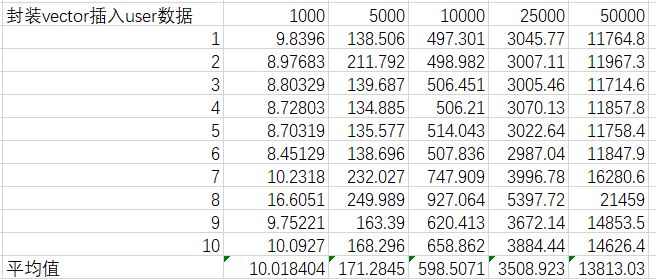
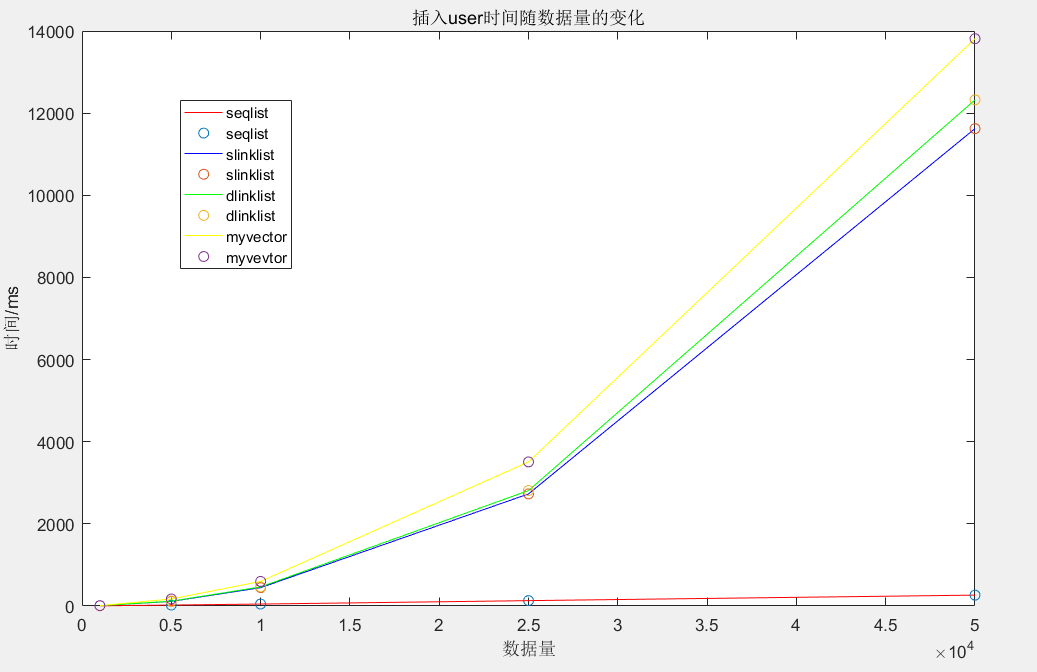
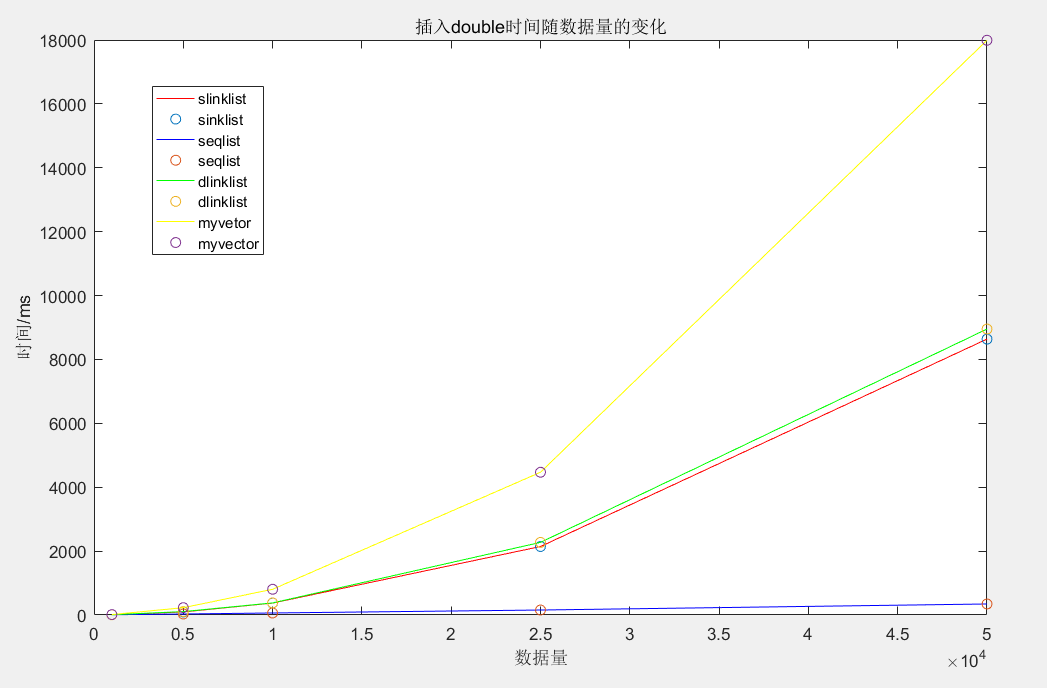
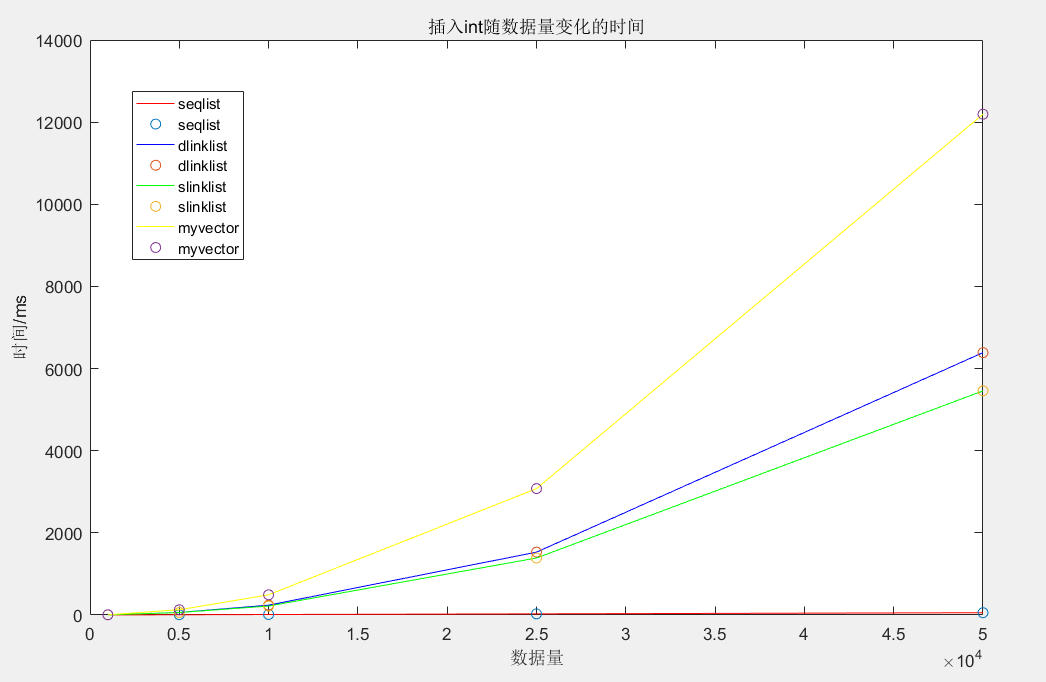
**1．插入前1000, 5000,10000,25000,50000个数据的时间开销**

将每个数据量下的插入操作运行十次，得数据如下表：  




对表中数据分别取平均值后，利用matlab进行绘图，得图像如下所示





分析：从图像中可以看出，不管哪种类型的对象，所需时间都是顺序表<单链表<双链表<封装vector，其中单链表和双链表所需的时间十分接近。顺序表插入所需时间与数据量近似于线性关系，因此顺序表插入时间随数据量的增长并不明显，而其他三种实现方式随数据量的增长，插入所需时间大幅增长。

书本上说顺序表的插入操作性能较差，而链表的插入操作性能较好，但这是相对于随机插入而言的，此处的情况是依次插入，因此顺序表不存在为了插入一个元素而将其后所有元素后移的情况，而链表需要多次移动指针，所以顺序表时间性能在此种情况下反而优于链表（我是用迭代器对vector进行插入操作，因此所需时间相当长，猜测用下标访问会节省大量时间）。至于单链表、双链表和封装vector时间性能上出现差异的可能原因个人猜测为单链表、双链表、封装vector开辟的空间依次变大，导致所需时间依次变大。

但是在实际应用中会出现很多在中间插入的情况，因此就实际应用来说链表在插入操作上的性能还是优于顺序表。

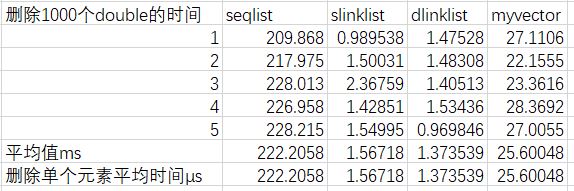
**2. 在数据集数量为50000的情况下，随机生成1000个访问下标，测量访问1000个元素的时间开销**



（\*注：封装vector的visit操作由下标访问实现）

不同数据类型下中随机访问1000个元素所需的时间如上表所示，每个数据集进行了5次试验，对结果求取平均值，并得到访问单个元素所需平均时间。

由表中数据可以看出不论哪种数据类型，顺序表和封装vector的实现方式所需时间都比链表实现方式小了一个数量级，所以可以得出对于访问操作顺序表的性能优于链表，原因在于通过下标访问无需多次移动指针，所以采取下标访问的封装vector时间性能与顺序表在同一数量级。

**3. 在数据集数量为50000的情况下，从提供的数据集中随机选择1000个元素，测量删除这1000个元素的时间开销** 

不同数据类型下中随机删除1000个元素所需的时间如上表所示，每个数据集进行了5次试验，对结果求取平均值，并得到访问单个元素所需平均时间。

由表中数据可以看出不论哪种数据类型，单链表和双链表执行删除操作所需的时间都远小于另外两种实现方式，因为对于链表删除后无需依次向前移动后面的元素，所以对于删除操作链表优于顺序表。

4.书本程序题见源代码