Homework 2: Skiplist report

Part 1: 对跳表的插入、搜索等关键代码的说明

1.1 搜索

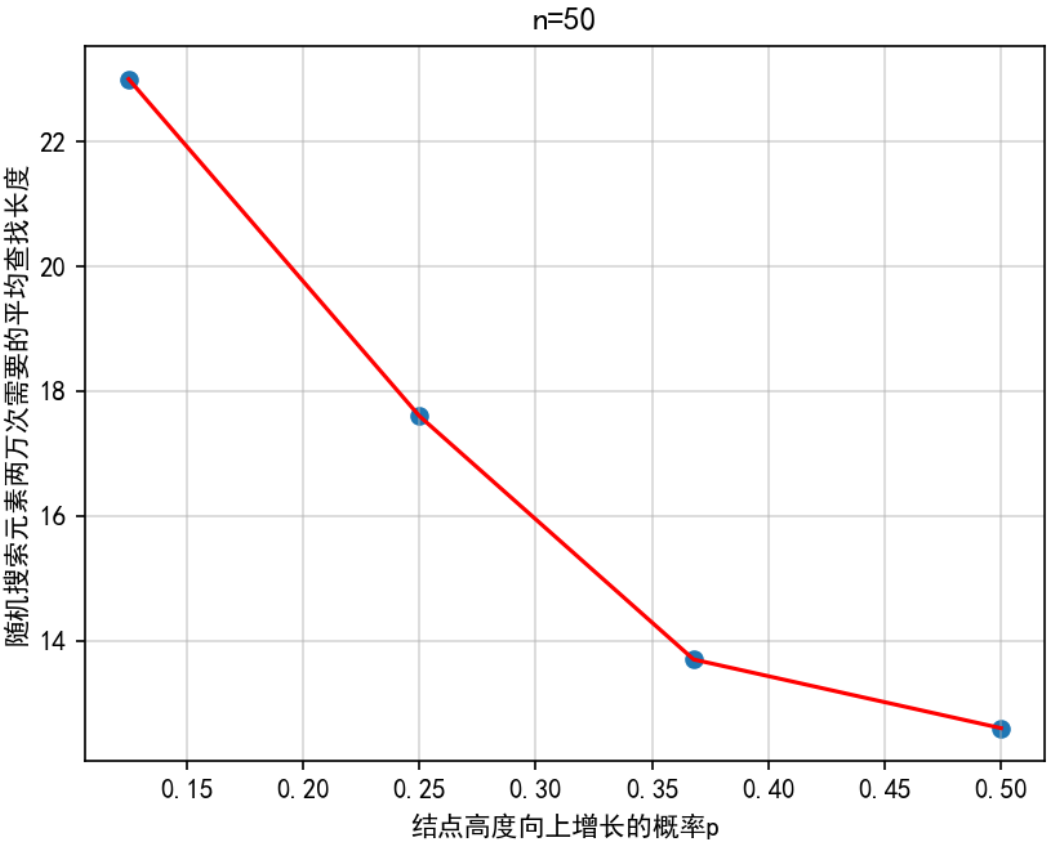
搜索操作建立在已有一个跳表的基础上，从head结点开始，首先向右进行搜索，当所在节点的右节点小于搜索值时，向右走，直至右侧结点值大于搜索值或到达跳表最右端，此时向下移动一层，再进行上述操作与判断，当此过程中有搜索到该节点，将该节点返回，否则返回空指针。

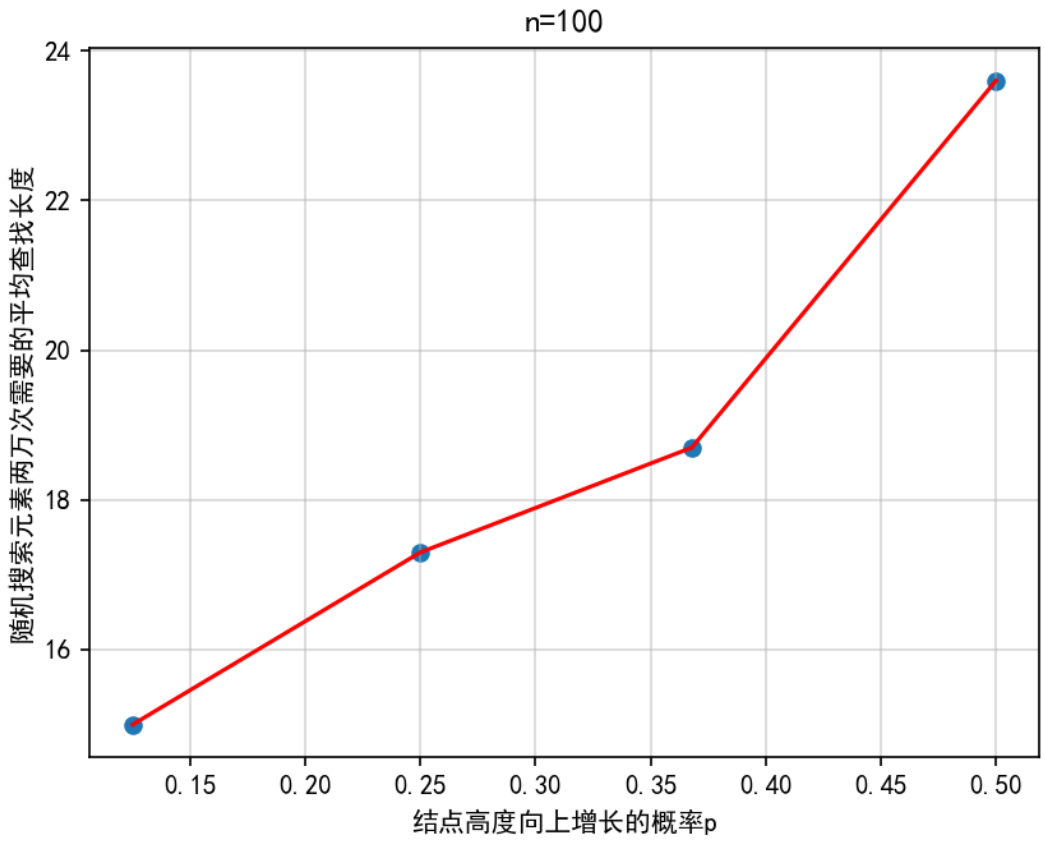
1.2 插入

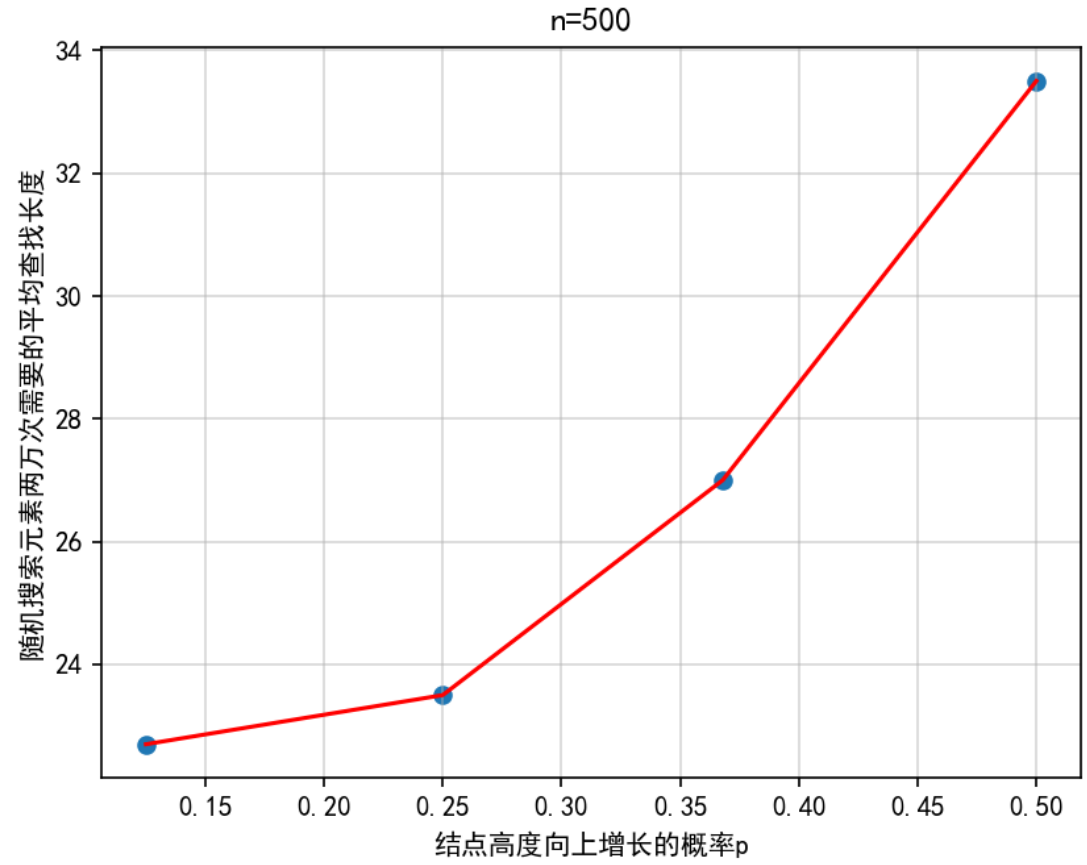
在跳表中插入元素的操作建立在搜索操作能够进行的基础上。首先进行搜索判断，即原本就在跳表中的元素不再进行插入，若该元素原本不存在，那么考虑插入。插入过程和搜索过程共用一个搜索体系，因此，当搜索出来的值返回后，可以在此返回值的基础上进行插入，省去一半的重复操作。每个元素初始时默认有一层的高度，然后使用播种随机种子生成随机数的方法，按一定的概率让新加入的节点的高度增加，当节点的高度不大于跳表的高度时，需要在新加节点的左侧及右侧找到同层次最近的元素，将链接关系改变，而由于这两个最近的左右元素原本应该相邻，因此找到一侧便可以引到另外一侧，当节点的高度已经大于跳表的高度时，则需要创建新的头尾节点head、tail，并将其与新结点连接，直至此元素插入结束。

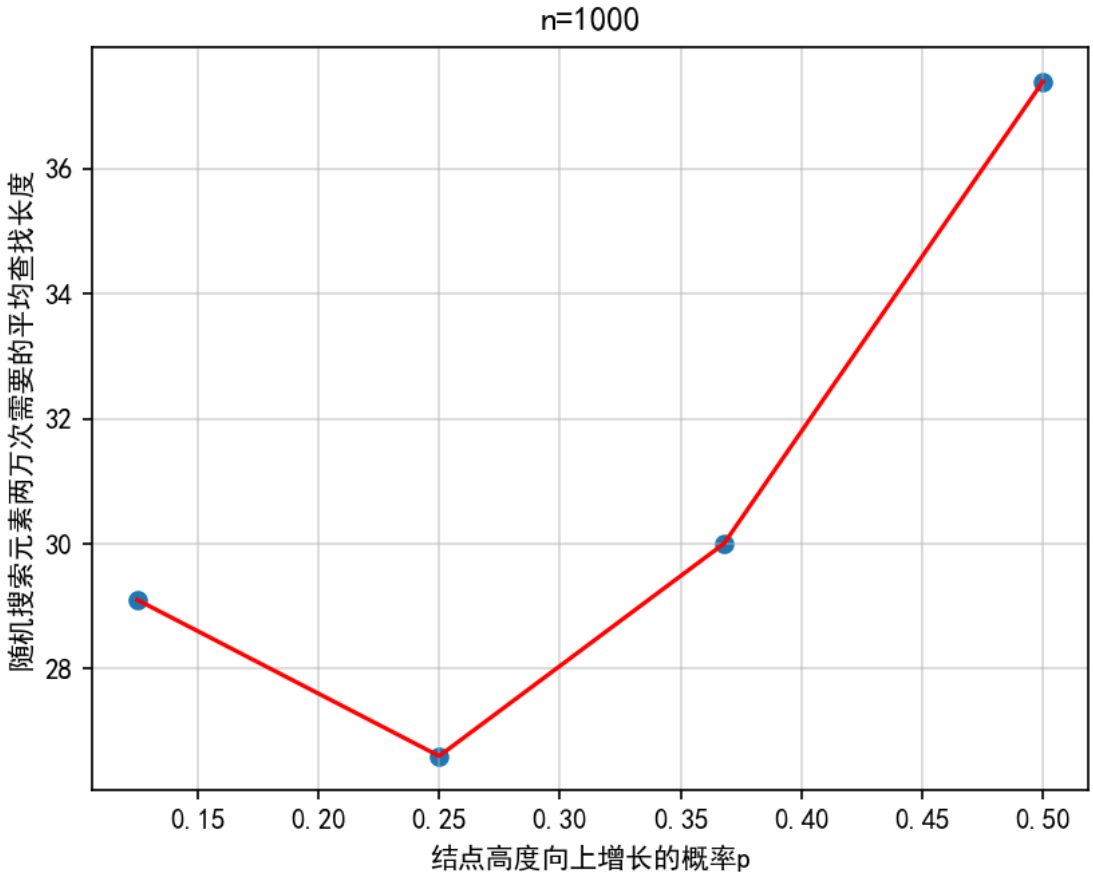
Part 2: 设计测试用例

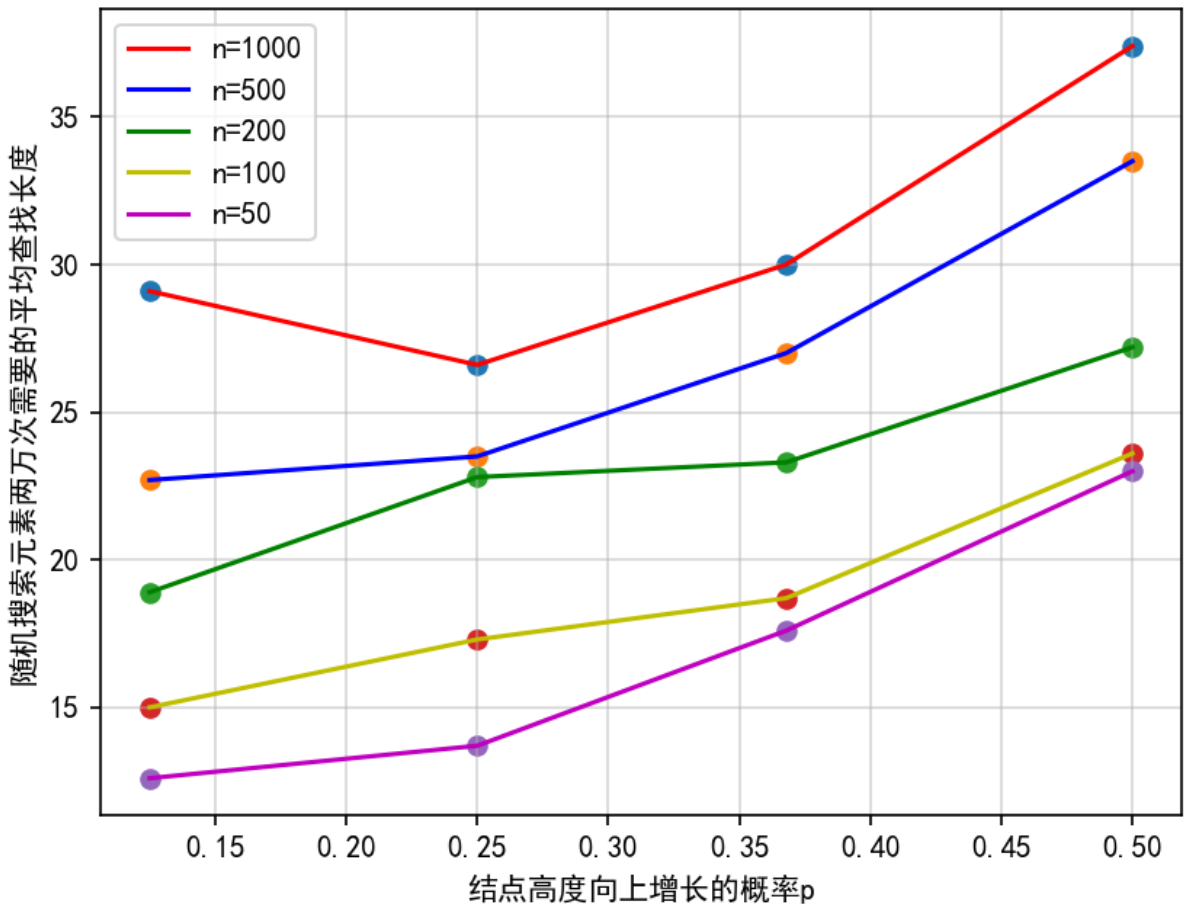
1. 建立长度（**跳表元素个数**）分别为 50, 100，200，500，1000，概率 p 分别为 1/2，1/e，1/4，1/8 的跳表。
2. 随机搜索至少 10000 次，并记录每组对应的平均搜索长度。（请在跳表长度范围内随机搜索，比如跳表长度为 100，生成的随机数就为 1-100 之间的一个值；长度为 1000，生成的随机数就为 1-1000 之间的一个值。）
3. 探究相同**跳表长度**下，增长率 p 和平均**搜索长度**的关系（**画出折线图或者柱状图**），并分析是否符合理论情况，如果不符合分析原因。


n=50
随机搜索元素两万次需要的平均查找长度
22
20
18
16
14
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
结点高度向上增长的概率p


n=100
随机搜索元素两万次需要的平均查找长度
24
22
20
18
16
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
结点高度向上增长的概率p

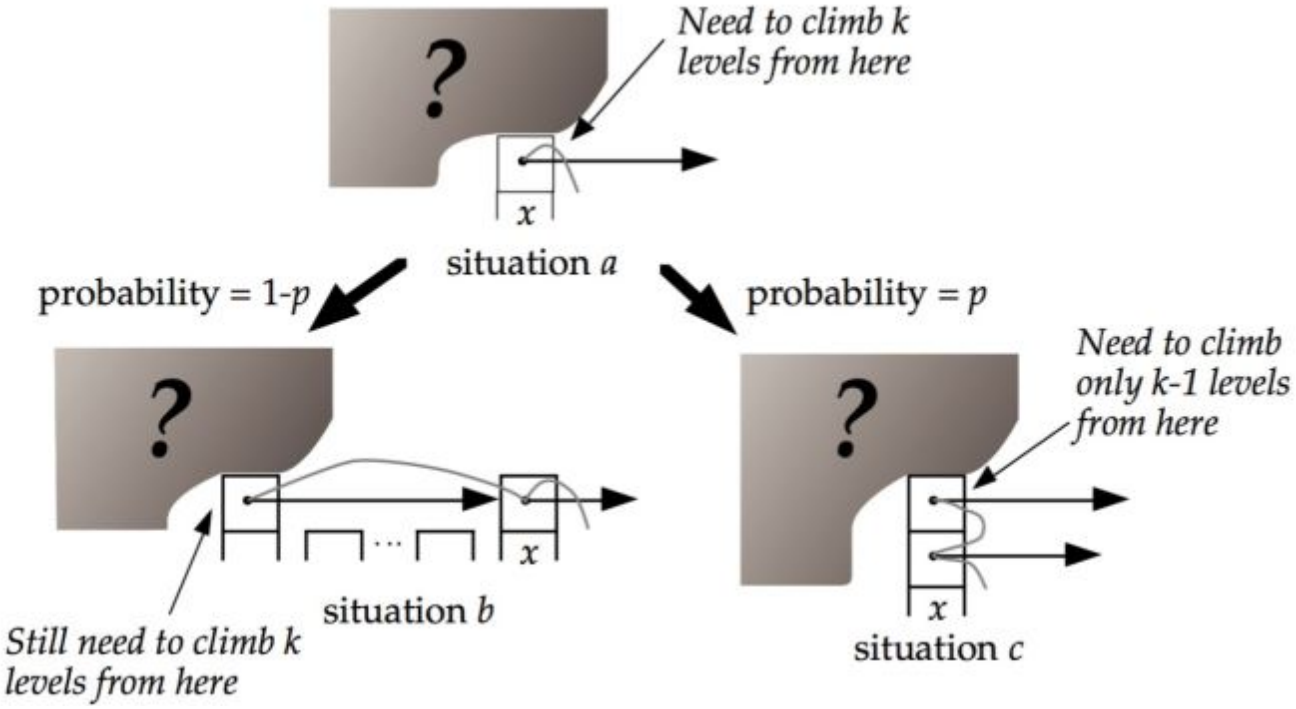

n=500
34
32
30
28
26
24
随机搜索元素两万次需要的平均查找长度
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
结点高度向上增长的概率p


n=1000
36
34
32
30
28
随机搜索元素两万次需要的平均查找长度
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
结点高度向上增长的概率p



Part 3: 分析

可以看出，skiplist每次插入都是独立的,因为random()的随机生成.执行插入操作时计算随机数的过程，是一个很关键的过程，它对skiplist的统计特性有着很重要的影响。这并不是一个普通的服从均匀分布的随机数，它的计算过程如下：首先，每个节点肯定都有第1层指针（每个节点都在第1层链表里）。如果一个节点有第i层(i>=1)指针（即节点已经在第1层到第i层链表中），那么它有第(i+1)层指针的概率为p。节点最大的层数不允许超过一个最大值，记为Max_level。



$C(0) = 0$

$C(k) = (1-p)\times$ (上图中情况b的查找长度) + p×(上图中情况c的查找长度)

代入，得到一个差分方程并化简：

$C(k) = (1-p)(C(k)+1) + p(C(k-1)+1)$

$C(k) = 1/p + C(k-1)$

$C(k) = k/p$

因此，每爬升1个层级，需要在查找路径上走1/p步。而我们总共需要攀爬的层级数等于整个skiplist的总层数-1。

接下来我们需要分析一下当skiplist中有n个节点的时候，它的总层数的概率均值是多少。第1层链表固定有n个节点；第2层链表平均有n*p个节点；第3层链表平均有n*p^2个节点；...

所以，从第1层到最高层，各层链表的平均节点数是一个指数递减的等比数列。容易推算出，总层数的均值为log1/pn，而最高层的平均节点数为1/p。

综上，平均查找长度约等于：

$C(log1/pn - 1) = (log1/pn - 1)/p$

即，平均时间复杂度为O(log n)。

从我们的性能分析图可以看出，同一插入规模下，p越大，节点向上生长的概率越大，层数越高，模拟出来的查找长度随p变大而增大，同一个生长概率p下，规模越大，所需查找次数自然也变高。与理论有一定出入的原因有很多，分析如下：

- 最高跳表高度限制使得曲线趋势并不完全符合我们计算的预期
- 建表时每个数据点的高度是随机的，建表对查找算法的影响是巨大的，即使我们对每个建好的跳表已经随机搜索10000次，但仍然克服不了跳表本身的缺陷和偶然性，随机性很大。

- 我们只插入了至多一千个数据点，如果能进一步增大跳表的规模，理论上就可以减小建表的随机性产生的系统误差。
- 我的查找算法里，向右观测与向下观测都算入查找范围内，而事实上的搜索长度若是想要与理论公式符合，有些程序中的试探性搜索不算入长度中，且理论计算时由于一些不等式的使用有些舍入，得到的公式只是粗略地刻画真实的情景。