数据结构实验报告

软件学院 软件工程 4 班 2113850 李鹏

——实验题目：查找树的性能对比（时间性能）

一、整体思路及算法：

1. 二叉搜索树相关概念：
左子树上全部结点值均小于根结点
右子树上全部结点值均大于根结点
结点的左右子树本身是一棵二叉搜索树
二叉搜索树中序遍历所得结果是递增排序的结点序列。

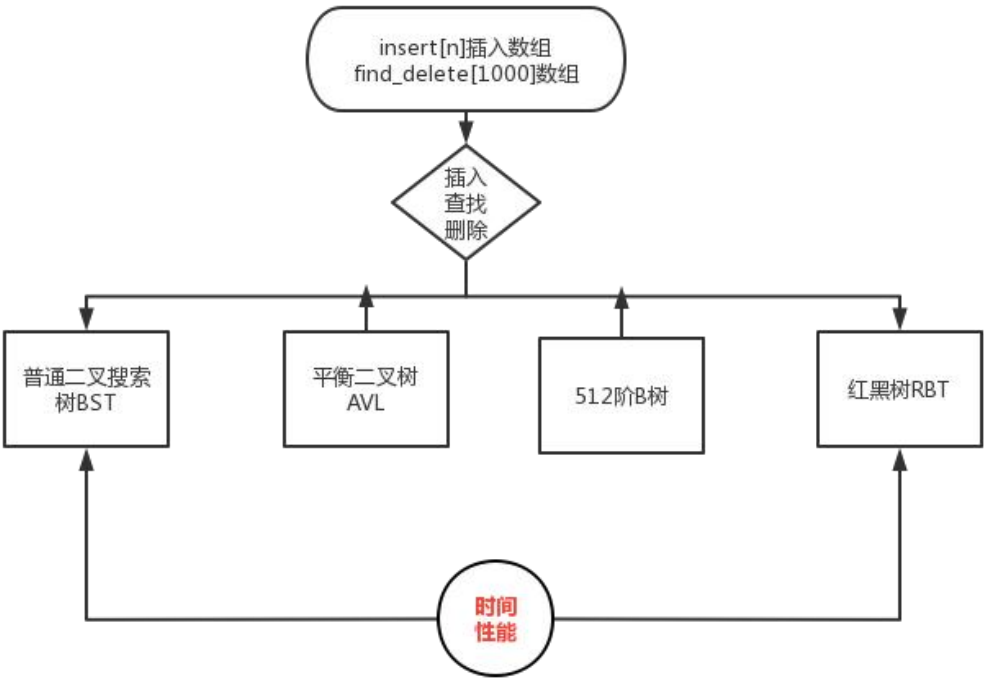
2.分别构建普通二叉搜索树（**BST**），平衡二叉搜索树（**AVL**），512 阶 B 树，红黑树（**RBT**），实现**插入、查找、删除**的操作

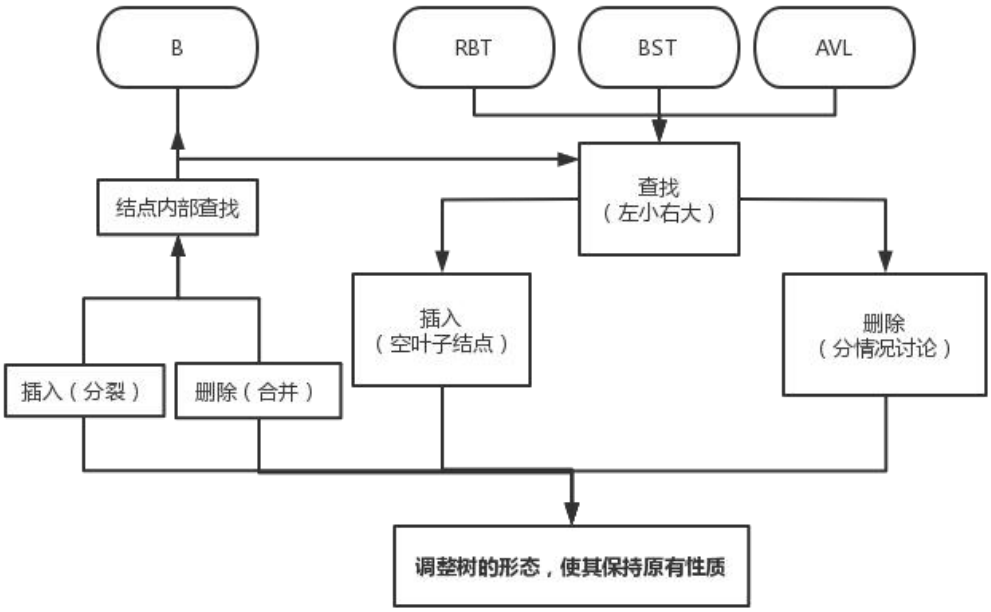
插入、查找、删除操作样例分为以下情况：

① 递增插入 N 个整数，随机查找 1000 个数，按相同顺序删除
② 递增插入 N 个整数，随机查找 1000 个数，按相反顺序删除
③ 随机插入 N 个整数，随机查找 1000 个数，按随机顺序删除

其中，N 从 10000 到 400000 取值，间隔 20000，以数据规模 N 为横轴，以运行时间为纵轴，分别画出普通二叉搜索树（**BST**），平衡二叉搜索树（**AVL**），512 阶 B 树，红黑树（**RBT**）的时间性能图。

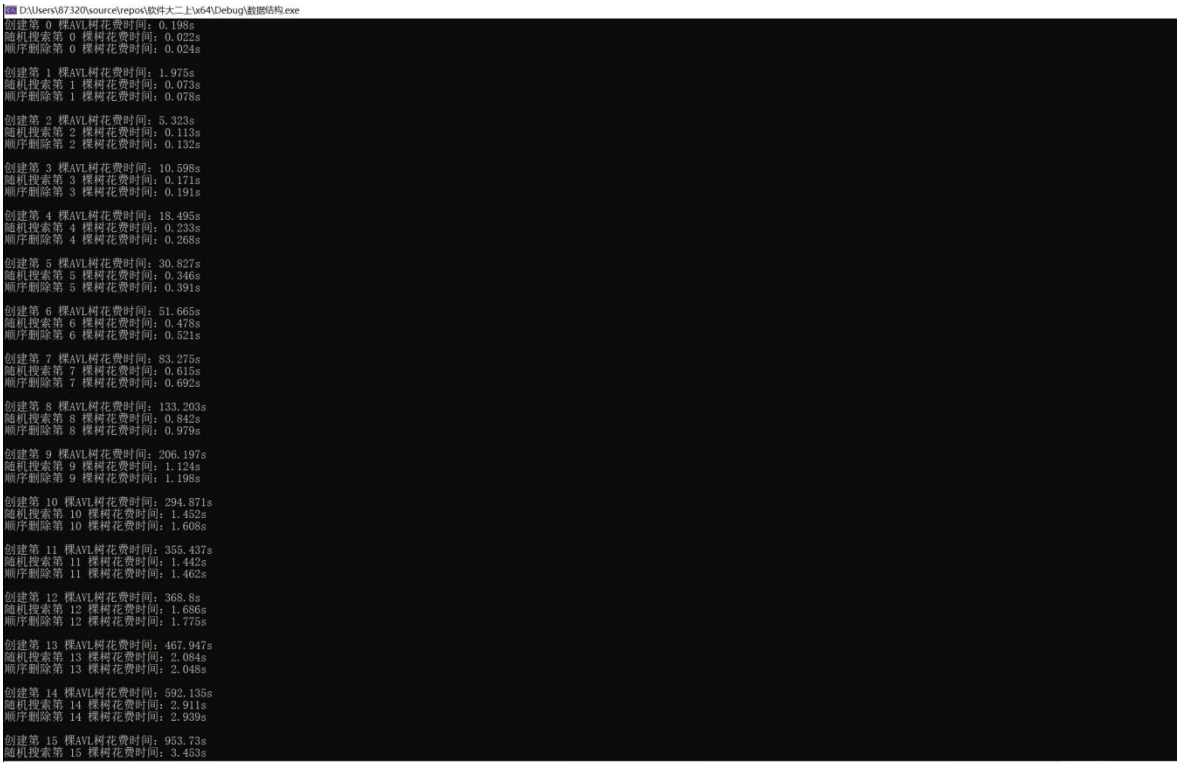
二、流程图：


insert[n]插入数组
find_delete[1000]数组
插入
查找
删除
普通二叉搜索树BST
平衡二叉树AVL
512阶B树
红黑树RBT
时间性能


B
RBT
BST
AVL
结点内部查找
查找
（左小右大）
插入
（空叶子结点）
删除
（分情况讨论）
插入（分裂）
删除（合并）
调整树的形态，使其保持原有性质

三、测试结果：

测试过程


D:\Users\87320\source\repos\软件大二上\x64\Debug\数据结构.exe
创建第 0 棵AVL树花费时间：0.198s
随机搜索第 0 棵树花费时间：0.022s
顺序删除第 0 棵树花费时间：0.024s
创建第 1 棵AVL树花费时间：1.975s
随机搜索第 1 棵树花费时间：0.073s
顺序删除第 1 棵树花费时间：0.078s
创建第 2 棵AVL树花费时间：5.323s
随机搜索第 2 棵树花费时间：0.113s
顺序删除第 2 棵树花费时间：0.132s
创建第 3 棵AVL树花费时间：10.598s
随机搜索第 3 棵树花费时间：0.171s
顺序删除第 3 棵树花费时间：0.191s
创建第 4 棵AVL树花费时间：18.495s
随机搜索第 4 棵树花费时间：0.233s
顺序删除第 4 棵树花费时间：0.268s
创建第 5 棵AVL树花费时间：30.827s
随机搜索第 5 棵树花费时间：0.346s
顺序删除第 5 棵树花费时间：0.391s
创建第 6 棵AVL树花费时间：51.665s
随机搜索第 6 棵树花费时间：0.478s
顺序删除第 6 棵树花费时间：0.521s
创建第 7 棵AVL树花费时间：83.275s
随机搜索第 7 棵树花费时间：0.615s
顺序删除第 7 棵树花费时间：0.692s
创建第 8 棵AVL树花费时间：133.203s
随机搜索第 8 棵树花费时间：0.842s
顺序删除第 8 棵树花费时间：0.979s
创建第 9 棵AVL树花费时间：206.197s
随机搜索第 9 棵树花费时间：1.124s
顺序删除第 9 棵树花费时间：1.198s
创建第 10 棵AVL树花费时间：294.871s
随机搜索第 10 棵树花费时间：1.452s
顺序删除第 10 棵树花费时间：1.608s
创建第 11 棵AVL树花费时间：355.437s
随机搜索第 11 棵树花费时间：1.442s
顺序删除第 11 棵树花费时间：1.462s
创建第 12 棵AVL树花费时间：368.8s
随机搜索第 12 棵树花费时间：1.686s
顺序删除第 12 棵树花费时间：1.775s
创建第 13 棵AVL树花费时间：467.947s
随机搜索第 13 棵树花费时间：2.084s
顺序删除第 13 棵树花费时间：2.048s
创建第 14 棵AVL树花费时间：592.135s
随机搜索第 14 棵树花费时间：2.911s
顺序删除第 14 棵树花费时间：2.939s
创建第 15 棵AVL树花费时间：953.73s
随机搜索第 15 棵树花费时间：3.453s

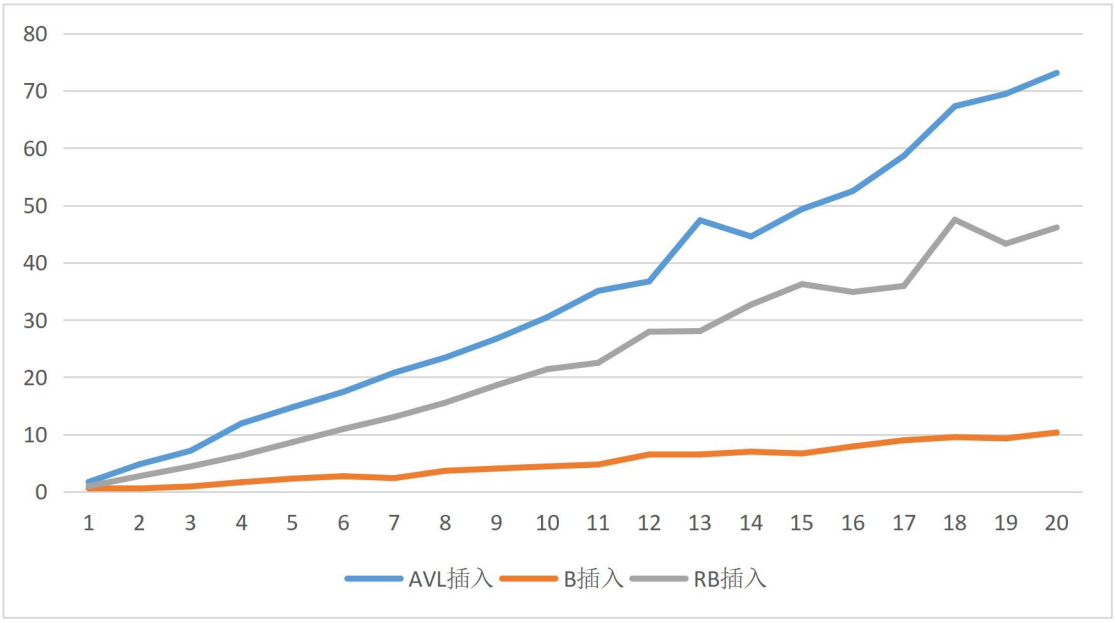

Microsoft Visual Studio 调试控制台
创建第 7 棵AVL树花费时间：83.275s
随机搜索第 7 棵树花费时间：0.615s
顺序删除第 7 棵树花费时间：0.692s
创建第 8 棵AVL树花费时间：133.203s
随机搜索第 8 棵树花费时间：0.842s
顺序删除第 8 棵树花费时间：0.979s
创建第 9 棵AVL树花费时间：206.197s
随机搜索第 9 棵树花费时间：1.124s
顺序删除第 9 棵树花费时间：1.198s
创建第 10 棵AVL树花费时间：294.871s
随机搜索第 10 棵树花费时间：1.452s
顺序删除第 10 棵树花费时间：1.608s
创建第 11 棵AVL树花费时间：355.437s
随机搜索第 11 棵树花费时间：1.442s
顺序删除第 11 棵树花费时间：1.462s
创建第 12 棵AVL树花费时间：368.8s
随机搜索第 12 棵树花费时间：1.686s
顺序删除第 12 棵树花费时间：1.775s
创建第 13 棵AVL树花费时间：467.947s
随机搜索第 13 棵树花费时间：2.084s
顺序删除第 13 棵树花费时间：2.048s
创建第 14 棵AVL树花费时间：592.135s
随机搜索第 14 棵树花费时间：2.911s
顺序删除第 14 棵树花费时间：2.939s
创建第 15 棵AVL树花费时间：953.73s
随机搜索第 15 棵树花费时间：3.453s
顺序删除第 15 棵树花费时间：3.489s
创建第 16 棵AVL树花费时间：1123s
随机搜索第 16 棵树花费时间：3.748s
顺序删除第 16 棵树花费时间：3.932s
创建第 17 棵AVL树花费时间：1318.71s
随机搜索第 17 棵树花费时间：3.86s
顺序删除第 17 棵树花费时间：3.943s
创建第 18 棵AVL树花费时间：1495.44s
随机搜索第 18 棵树花费时间：4.302s
顺序删除第 18 棵树花费时间：4.379s
创建第 19 棵AVL树花费时间：1737.56s
随机搜索第 19 棵树花费时间：4.711s
顺序删除第 19 棵树花费时间：4.903s
创建第 20 棵AVL树花费时间：1883.33s
随机搜索第 20 棵树花费时间：4.672s
顺序删除第 20 棵树花费时间：4.781s
D:\Users\87320\source\repos\软件大二上\x64\Debug\数据结构.exe (进程 72428)已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口. . .

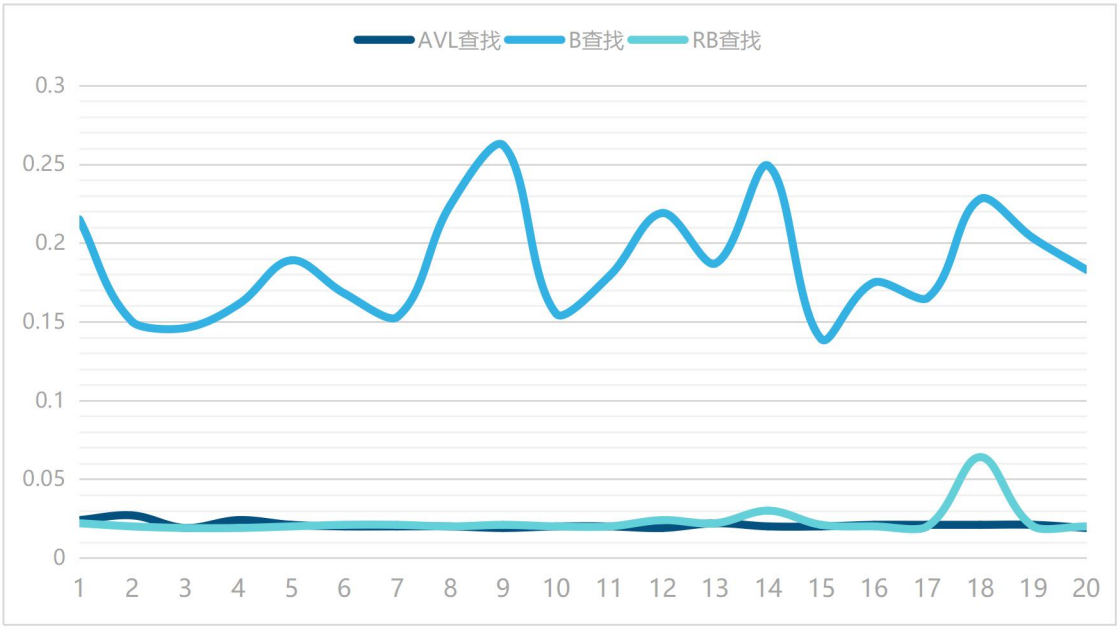
测试数据

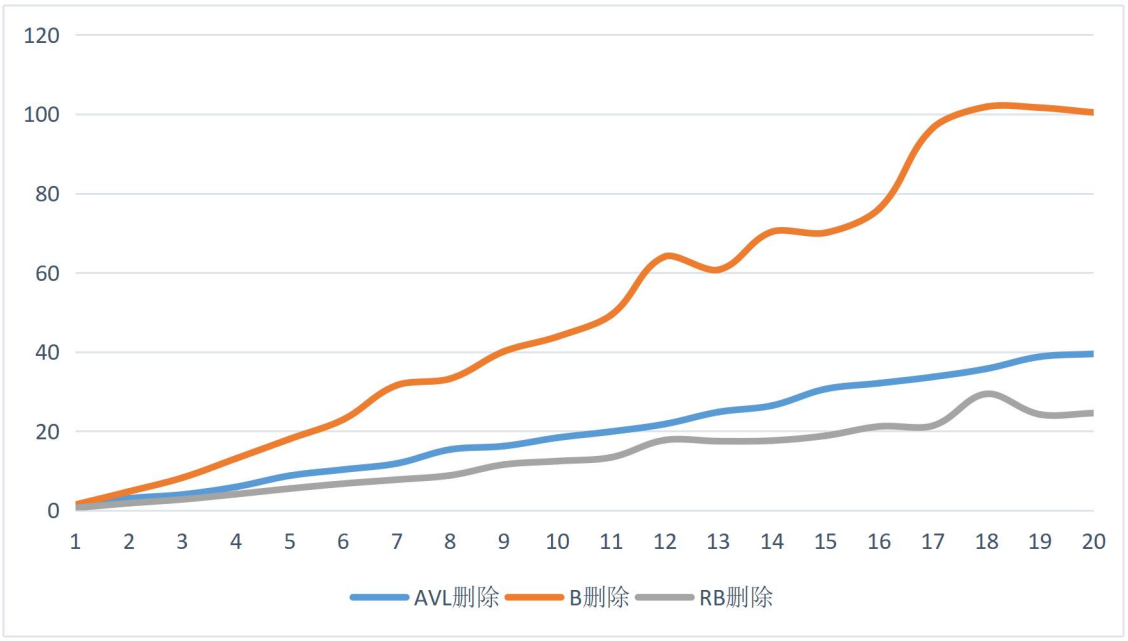
AVL插入	B插入	RB插入	AVL查找	B查找	RB查找	AVL删除	B删除	RB删除
1.715	0.591	0.969	0.024	0.215	0.022	1.123	1.472	0.657
4.809	0.578	2.737	0.027	0.15	0.02	3.053	4.788	1.829
7.152	0.927	4.421	0.019	0.146	0.019	4.03	8.275	2.796
11.934	1.672	6.346	0.024	0.161	0.019	5.957	13.121	4.115
14.758	2.296	8.654	0.021	0.189	0.02	8.776	18.042	5.527
17.443	2.719	10.965	0.02	0.168	0.021	10.322	22.944	6.776
20.799	2.385	13.07	0.02	0.153	0.021	11.873	31.6	7.779
23.438	3.666	15.556	0.02	0.224	0.02	15.393	33.271	8.869
26.706	4.054	18.589	0.019	0.262	0.021	16.251	40.135	11.581
30.474	4.425	21.401	0.02	0.155	0.02	18.368	43.868	12.459
35.077	4.76	22.519	0.02	0.179	0.02	19.909	49.332	13.406
36.713	6.512	27.94	0.019	0.219	0.024	21.807	64.064	17.761
47.398	6.52	28.058	0.022	0.187	0.022	24.841	60.72	17.485
44.572	7.002	32.668	0.02	0.249	0.03	26.464	70.323	17.629
49.349	6.683	36.248	0.02	0.139	0.021	30.644	70.061	18.868
52.501	7.914	34.884	0.021	0.175	0.02	32.115	76.152	21.202
58.652	8.995	35.938	0.021	0.165	0.02	33.694	96.498	21.376
67.302	9.532	47.491	0.021	0.228	0.064	35.752	101.842	29.415
69.47	9.328	43.307	0.021	0.203	0.02	38.793	101.621	24.23
73.124	10.352	46.137	0.019	0.183	0.02	39.508	100.409	24.563

	操作	时间/ms	时间/ms	时间/ms	时间/ms	时间/ms	时间/ms	时间/ms	时间/ms	时间/ms
二叉搜索树	增序插入	580.4	3832.3	9582.4	17128.2	26768.6	39788.1	53418.9	75115.2	91095.9
	随机插入	540.0	2501.7	413.6	14650.8	19807.2	34083.5	46138.1	65155.1	80101.4
	增序插入查找1000个数	26.3	72.9	81.4	73.0	71.7	74.1	72.7	76.3	71.9
	随机插入查找1000个数	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	正向删除	5.0	14.4	19.1	60.3	55.4	156.0	154.0	242.1	251.3
	反向删除	581.0	3882.3	9930.4	17425.3	26669.9	39616.0	52692.9	73440.7	90295.4
	随机删除	350.2	3596.7	4521.5	12413.1	17560.5	35658.8	40177.6	65141.8	67156.6
	N	20000.0	40000.0	60000.0	80000.0	100000.0	120000.0	140000.0	160000.0	180000.0

测试图表


80
70
60
50
40
30
20
10
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
AVL插入
B插入
RB插入





四、分析总结：

1. 理论分析：

(1) BST：

a. 插入代价：

普通二叉搜索树插入新结点，需要查找到插入该结点的位置，所需代价与查找一个不存在的结点值相同。故 BST 插入操作的时间复杂度为 O(logN)～O(N)。

b. 查找代价：

普通二叉搜索树的查找过程都需要从根结点出发，沿某一个路径朝叶子结点前进，所以查找中数据比较次数与树的形态密切相关。 当树中每一个结点左右子树高度大体相同时，树高为 logN。平均查找长度与 logN 成正比，查找的平均时间复

杂度为 O(logN)。当前后插入的关键字有序时，BST 退化成单支树结构，此时树高 n。平均查找长度为(n+1)/2，查找的平均时间复杂度为 O(N)。故 BST 查找操作的时间复杂度为 O(logN)～O(N)。

c. 删除代价：

普通二叉搜索树删除一个结点，首先须要定位到该结点，这个过程需要一次查找 O(N)。找到该结点的位置后，若是被删除结点的左、右子树只有一个存在，则改变树形态的代价仅为 O(1)。若是被删除结点的左、右子树均存在，只须要将当 P 的左孩子的最右右结点与该结点互换，直接删除叶结点即可。故 BST 删除操作的时间复杂度为 O(logN)～O(N)。

(2) AVL：

a. 插入代价：

AVL 必须要保证严格平衡(|bf|<=1)，那么每一次插入数据使得 AVL 中某些结点的平衡因子超过 1 就必须进行旋转操作。事实上，AVL 的每一次插入结点操作最多只须要旋转 1 次(单旋转或双旋转)。故 AVL 插入操作的时间复杂度为 O(logN)～O(N)。

b. 查找代价：

AVL 是严格平衡的 BST（平衡因子不超过 1）。那么查找过程与 BST 同样，只是 AVL 不会出现最差状况的 BST(单支树)。故 AVL 查找操作的时间复杂度为 O(logN)。

c. 删除代价：

AVL 删除结点的算法能够类似 BST 的删除结点，可是删除以后必须检查从删除结点开始到根结点路径上的全部结点的平衡因子。所以删除的代价稍微要大一些。每一次删除操作最多须要 O(logN)次旋转。故 AVL 删除操作的时间复杂度为 O(logN)。

(3) 512 阶 B 树：

a. 插入代价：

B 树的插入操作会发生结点的分裂操作。

b. 查找代价：

B 树是一种平衡多路查找树(m-叉)，高度很小，故 B 树的查找时间复杂度很低。

c. 删除代价：

B 树的删除操作会发生结点的合并操作，可能会回溯到根结点。

(4) 红黑树：

a. 插入代价：

红黑树插入结点时，需要旋转操作和变色操作。插入结点最多只需要 2 次旋转，再根据情况染色。故红黑树插入操作的时间复杂度为 O(logN)。

b. 查找代价：

因为红黑树的最长路径长度不超过最短路径长度的 2 倍，所以红黑树虽然不像 AVL 同样是严格平衡的，但平衡性能仍是要比 BST 要好。故红黑树查找操作的时间复杂度为 O(logN)。

c. 删除代价：

红黑树的删除操作代价要比 AVL 要好的多，删除一个结点最多只须要 3 次旋转操作，但仍需进行查找。故红黑树删除操作的时间复杂度为 O(logN)。

2. 实验结论：

① 递增插入 N 个整数，红黑树、AVL 树快，普通二叉树、B 树慢
② 随机插入 N 个整数，红黑树、AVL 树快，普通二叉树、B 树慢
③ 随机查找 1000 个数，B 树、AVL 树快，红黑树、普通二叉树慢
④ 按相同顺序删除，普通二叉树最快，AVL 树、B 树、红黑树慢
⑤ 按相反顺序删除，红黑树最快，普通二叉树、AVL 树、B 树慢
⑥ 按随机顺序删除，红黑树、B 树快，普通二叉树、AVL 树慢