# SplayTree 与 AvlTree 的对比实验

## 1. 实验思路

根据实验要求，选则数据规模为 n=1000，查询次数为 m=1000000，m/n=$10^3$，k/n 取 0.5%，1%，10%，20%，50%，80%.

首先由 random 库随机生成长度为 1000, 无重复元素的输入集, 并插入到 avl 树及 splay 树中，随后对输入集进行排序，对二者在有序插入集中指定的的连续数据范围的搜索时间进行测试。由于 splay 树每次测试后树的储存状态发生更新，因此对不同查找范围进行测试时需重置 splay 树。

本实验采用的 avl 树以及 splay 树代码来自 https://github.com/BigWheel92/

## 2. 实验数据

在 1. 的条件下，所测得的实验数据为

**表 1 SplayTree 与 AvlTree 对比测试实验数据**

| k/n | 时间（μs） | |
|---|---|---|
| | SplayTree | AvlTree |
| 0.5% | 34070.3 | 38484.7 |
| 1% | 41959.4 | 42731.8 |
| 2% | 47137.2 | 42786.8 |
| 20% | 74406.4 | 56610.3 |
| 50% | 84352.8 | 63405.7 |
| 80% | 90272.0 | 68056.4 |

## 3. 对实验结果的分析

两种数据结构的搜索时间增长趋势符合预期，即：Splay 树时间复杂度随 k/n 的增长快于 Avl 树，在 k/n 较小时，Splay 树优于 Avl 树，k/n 较大时则反之。观察发现，Splay 树仅在 k/n 小于 1%时优于 Avl 树，k/n=1%时二者的时间成本已经较为接近。可能的原因是，由于程序级实现的不足，导致 top-down splay 树在 splay 的过程中资源消耗较严重，相比于简单高效的 avl 树查找操作，时间成本随 k/n 的增长速度较快。

## 4. 对 Splay 树的理解

在实际的信息处理场景中，刚刚被访问过的数据，极有可能再次被访问，若需要访问的数据在查找树中位于距离根节点较远的位置，则重复多次查询需要较高的时间复杂度。而 Splay 针对以上现象对查找算法做出调整，将每次被访问后的数据节点更新至根节点，使得键值与上一次访问的键值较为接近的数据节点（包括该节点本身）在下一次访问中可以较为容易地被查询到。

Splay 树由于其算法特性，在查找次数远大于数据规模、数据规模远大于查询范围时具有明显的优势，而在其他情况下，由于每次查询需要对查找树的结构进行调整，相比于 avl 树等静态查找树将不再有优势，实验数据在一定程度上说明了这一点。