## Lab8 范围搜索实验

### 1. 实验要求

本次实验你将基于BST扩展order table 的ADT接口，完成一些基本函数，你可以从一般的库函数出发扩展此库。此外，你将完成一个范围搜索实验，即给定一个二维点集，以及一个矩形（用左上和右下坐标表示）范围，找出在此范围内点的个数，你需要自定义数据结构以满足复杂度需求。

2. 回答问题

2.1 完成函数first，last简述思路。

Task 4.1 (6%). Implement the functions

fun first (T : 'a table) : (key \* 'a) option fun last (T : 'a table) : (key \* 'a) option Given an ordered table T, first T should evaluate to SOME (k,v) iff (k,v)∈T and k is the minimum key in T. Analagously, last T should evaluate to SOME (k,v) iff (k,v)∈T and k is the maximum key in T. Otherwise, they evaluate to NONE.

|  |
| --- |
| * 算法思路： |
| * 代码实现： |
| * 关于测试： |

2.2 完成函数previous和last并简述思路。

Task 4.2 (8%). Implement the functions

fun previous (T : 'a table) (k : key) : (key \* 'a) option fun next (T : 'a table) (k : key) : (key \* 'a) option Given an ordered table T and a key k, previous T k should evaluate to SOME (k',v) if (k0,v)∈ T and k0 is the greatest key in T strictly less than k. Otherwise, it evaluates to NONE. Similarly, next T k should evaluate to SOME (k',v) iff k0 is the least key in T strictly greater than k.

|  |
| --- |
| * 算法思路： |
| * 代码实现： |
| * 关于测试： |

2.3 完成下列函数，做必要说明。

Task 4.3 (2%). Implement the function

fun join (L : 'a table, R : 'a table) : 'a table

Given ordered tables L and R, where all the keys in L are strictly less than those in R, join (L, R) should evaluate to an ordered table containing all the keys from both L and R.

|  |
| --- |
|  |
| * 关于测试： |

Task 4.4 (2%). Implement the function

fun split (L : 'a table, k : key) : 'a table \* 'a option \* 'a table

Given an ordered table T and a key k, split should evaluate to a triple consisting of

1. an ordered table containing every (k',v)∈T such that k' < k,

2. SOME v if (k,v)∈T and NONE otherwise, and

3. an ordered table containing every (k',v)∈T such that k'> k.

|  |
| --- |
|  |
| * 关于测试： |

2.4 完成函数getRange并详述思路

Task 4.5 (7%). Implement the function

fun getRange (T : 'a table) (low : key, high : key) : 'a table

Given an ordered table T and keys l and h, getRange T (l, h) should evaluate to an ordered table containing every (k,v)∈T such that l≤k≤h.

|  |
| --- |
| * 算法思路： |
| * 代码实现： |
| * 关于测试： |

2.5 完成函数makeCountTable并回答相关问题

Task 5.1 (25%). In the MkRangeQuery functor, deﬁne the countTable type and implement the function

fun makeCountTable: point seq -> countTable

The type point is deﬁned to be OrdTable.Key.t \* OrdTable.Key.t where OrdTable is an ordered table structure provided toyou. You should choose the type of countTable such that you can implement count (range queries) in O(logn) work and span. For full credit, your makeCountTable must run within O(nlogn) expected work.

|  |
| --- |
| * 算法思路： |
| * 代码实现： |
| * 复杂度分析： |

Task 5.2 (10%). Brieﬂy describe how you would parallelize your code so that it runs in O(log2 n) span. Does the work remain the same? You don’t need to formally prove the bounds, just brieﬂy justify them.

|  |
| --- |
|  |

Task 5.3 (5%). What is the expected space complexity of your countTable in terms of n the number of input points? That is, how many nodes in the underlying binary search tree(s) does your countTable use in expectation? Explain in a few short sentences.

|  |
| --- |
|  |

2.6 完成函数count并做相关分析

Task 5.4 (25%). Implement the function

count: countTable -> point \* point -> int

Asdescribedearlier, count T ((x\_1,y\_1), (x\_2, y\_2))willreportthenumberofpointswithinthe rectangle with the top-left corner (x1, y1) and bottom-right corner (x2, y2). Your function should return the number of points within and on the boundary of the rectangle. You may ﬁnd the OrdTable.size function useful here. Your implementation should have O(logn) work and span.

|  |
| --- |
| * 算法思路： |
| * 代码实现： |
| * 复杂度分析： |
| * 关于测试： |