Homework 5

1 Trie (prefix tree)

Implement a trie with

- *insert*: insert a string into the trie.
- *search*: return true if the string is in the trie.
- *startsWith*: return true if there is any string in the trie that starts with the given prefix.

NOTE:

1. You may assume that all inputs are consist of lowercase letters a-z.
2. All inputs are guaranteed to be non-empty strings.

2 K-D Tree

Implement a k-d tree, includes

- *insert*: insert a new node with key into k-d tree;
- *search*: search node with the key;
- *remove*: remove node with the key;
- *findMin*: find the node with minimum value given the dimension;
- *rangeSearch*: find a list of nodes whose values are within the given range.

Note:

1. Keys contain two dimensional integers, such as (1, 2), (-5, 11).

3 Due

1. Due is Oct. 25th, 23:59.
2. It would be helpful to refer to our slide *Lecture 12-Data Structures Recap (3)*.
3. We provide a code template in the header file *tree.h* and cpp file *tree.cpp*. Fill in the codes and submit two files on the canvas.