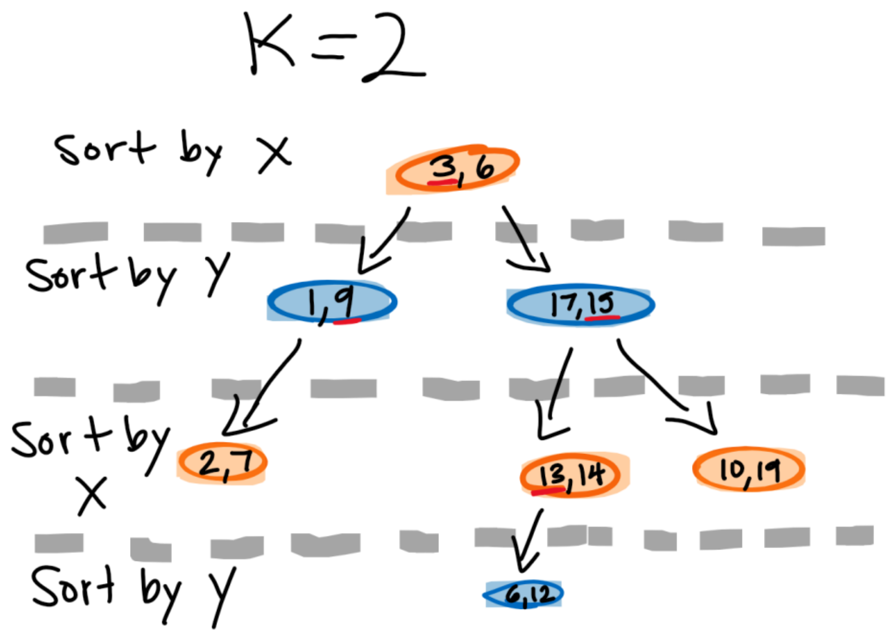
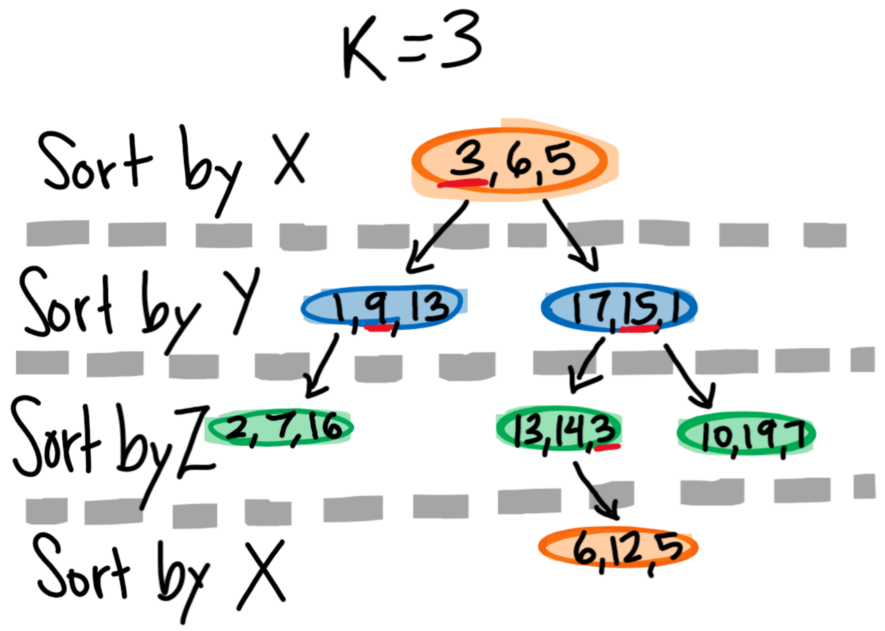
Data Structures: KD TREES TRIES

CS5008 – Final Project

Meilin, Rebecca, Sommer, Katie



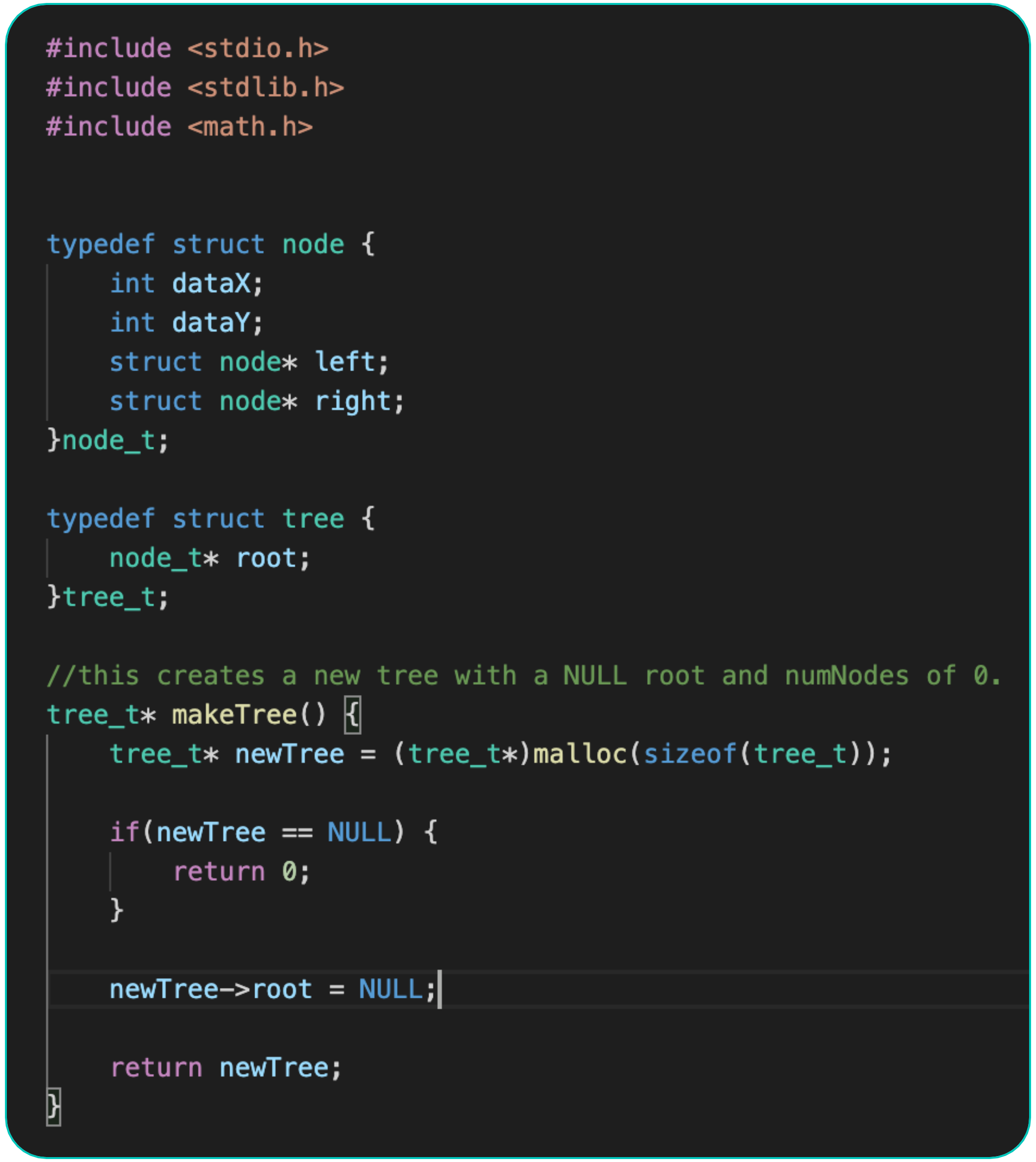


Introduction to KD TREES

K = 2: (3, 6), (17, 15), (13, 14), (6, 12), (1, 9), (2, 7), (10, 19)

K = 3: (3, 6, 5), (17, 15, 1), (13, 14, 3), (6, 12, 5), (1, 9, 13), (2, 7, 16), (10, 19, 7)

BASIC STRUCTURING:

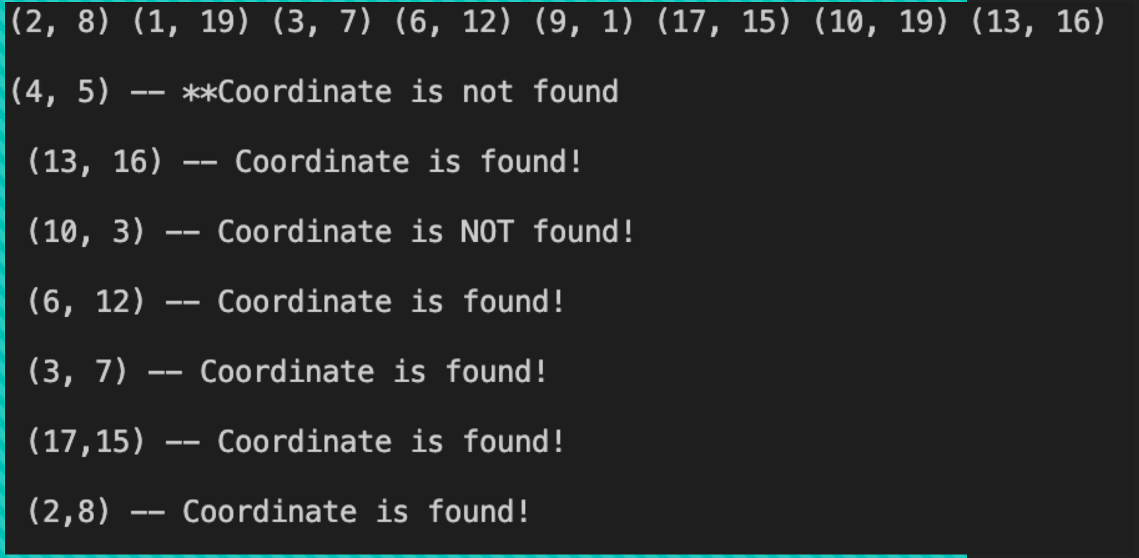


```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>


typedef struct node {
    int dataX;
    int dataY;
    struct node* left;
    struct node* right;
}node_t;

typedef struct tree {
    node_t* root;
}tree_t;

//this creates a new tree with a NULL root and numNodes of 0.
tree_t* makeTree() {
    tree_t* newTree = (tree_t*)malloc(sizeof(tree_t));

    if(newTree == NULL) {
        return 0;
    }

    newTree->root = NULL;

    return newTree;
}
```

- In the node struct is where we see the differentiation between “k” dimensional trees.

- In a 2D tree, we have dataX and dataY. In a 3D tree, we would see dataX, dataY, and dataZ.

Insertion:
Two Helper functions:
1. insertHelperX
2. insertHelperY



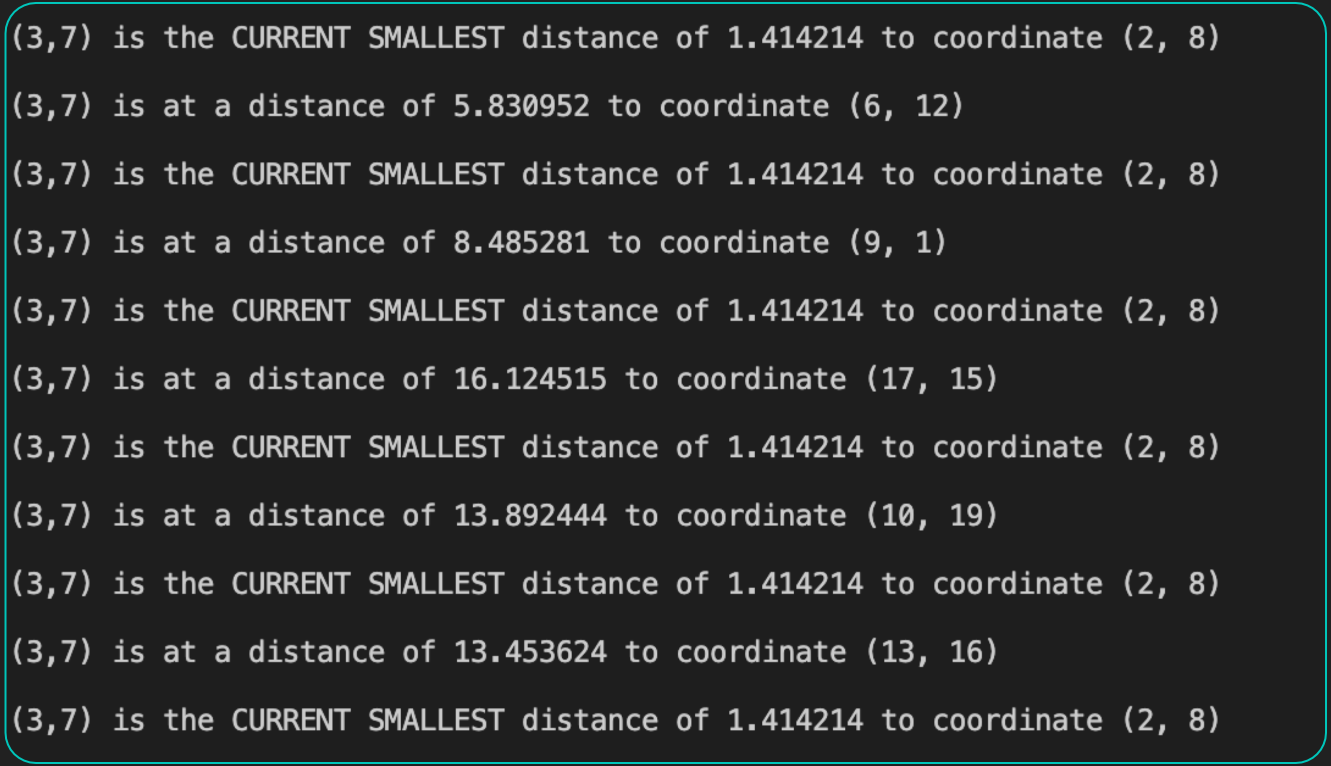
```
(2, 8) (1, 19) (3, 7) (6, 12) (9, 1) (17, 15) (10, 19) (13, 16)

(4, 5) -- **Coordinate is not found

 (13, 16) -- Coordinate is found!

 (10, 3) -- Coordinate is NOT found!

 (6, 12) -- Coordinate is found!

 (3, 7) -- Coordinate is found!

 (17,15) -- Coordinate is found!

 (2,8) -- Coordinate is found!
```

We use two separate functions because at each level of the tree, the data values being compared to determine a new placement alternates.

The helper functions recursively call one another since the comparisons always begin with X, move to Y, and back to X.

Nearest Neighbor:

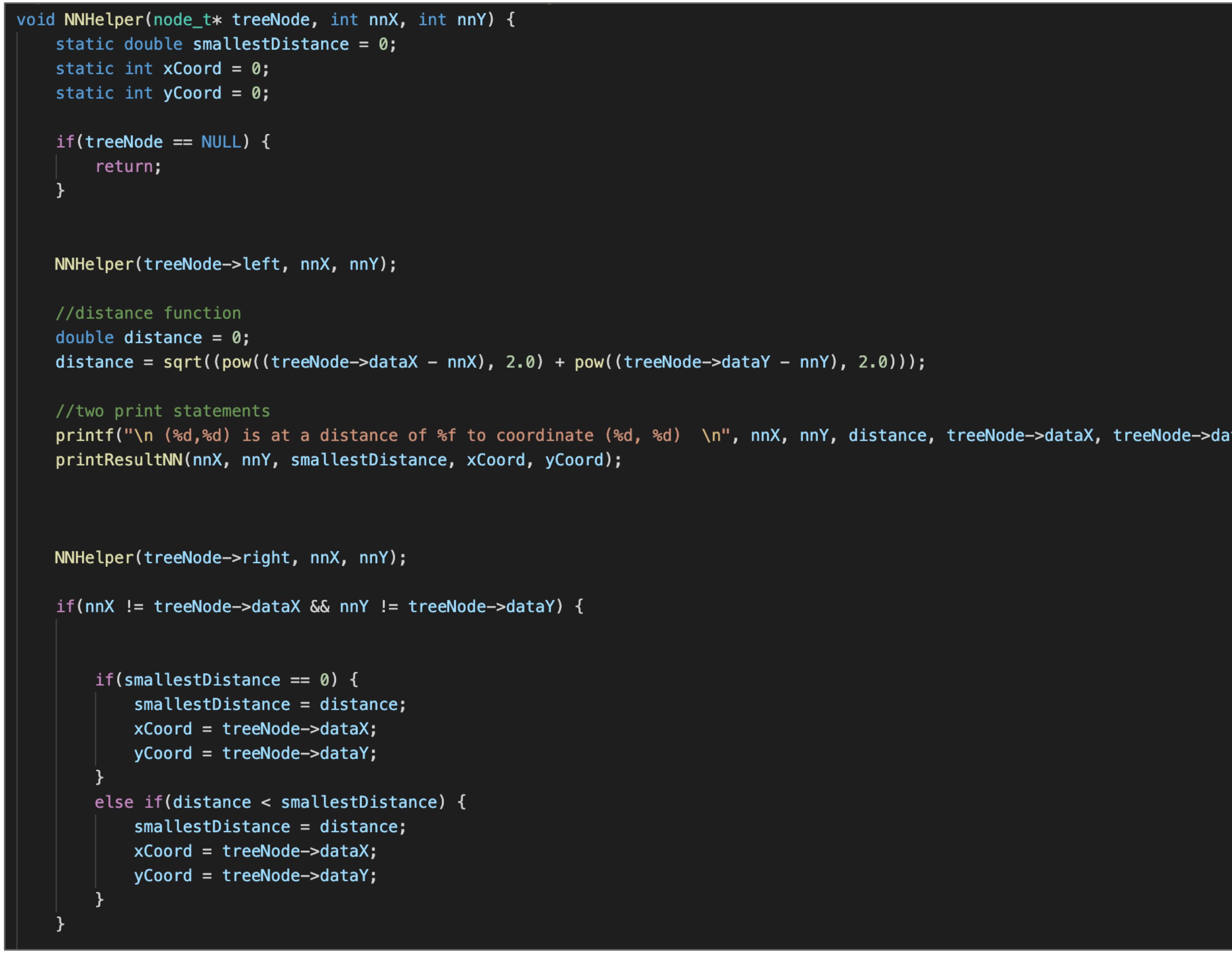
- Created static variables to store x coordinate, y coordinate, and distance.
- Recursively traverse using BFS to each node and calculate the distance between the current node and the node of interest.
- Use conditionals to determine if the current node is smaller than the previous node’s calculations.
- If it is, update the static variables so we can continue to compare previous values to the new “current” node.



```
(3,7) is the CURRENT SMALLEST distance of 1.414214 to coordinate (2, 8)

(3,7) is at a distance of 5.830952 to coordinate (6, 12)

(3,7) is the CURRENT SMALLEST distance of 1.414214 to coordinate (2, 8)

(3,7) is at a distance of 8.485281 to coordinate (9, 1)

(3,7) is the CURRENT SMALLEST distance of 1.414214 to coordinate (2, 8)

(3,7) is at a distance of 16.124515 to coordinate (17, 15)

(3,7) is the CURRENT SMALLEST distance of 1.414214 to coordinate (2, 8)

(3,7) is at a distance of 13.892444 to coordinate (10, 19)

(3,7) is the CURRENT SMALLEST distance of 1.414214 to coordinate (2, 8)

(3,7) is at a distance of 13.453624 to coordinate (13, 16)

(3,7) is the CURRENT SMALLEST distance of 1.414214 to coordinate (2, 8)
```

Tradeoffs in the code

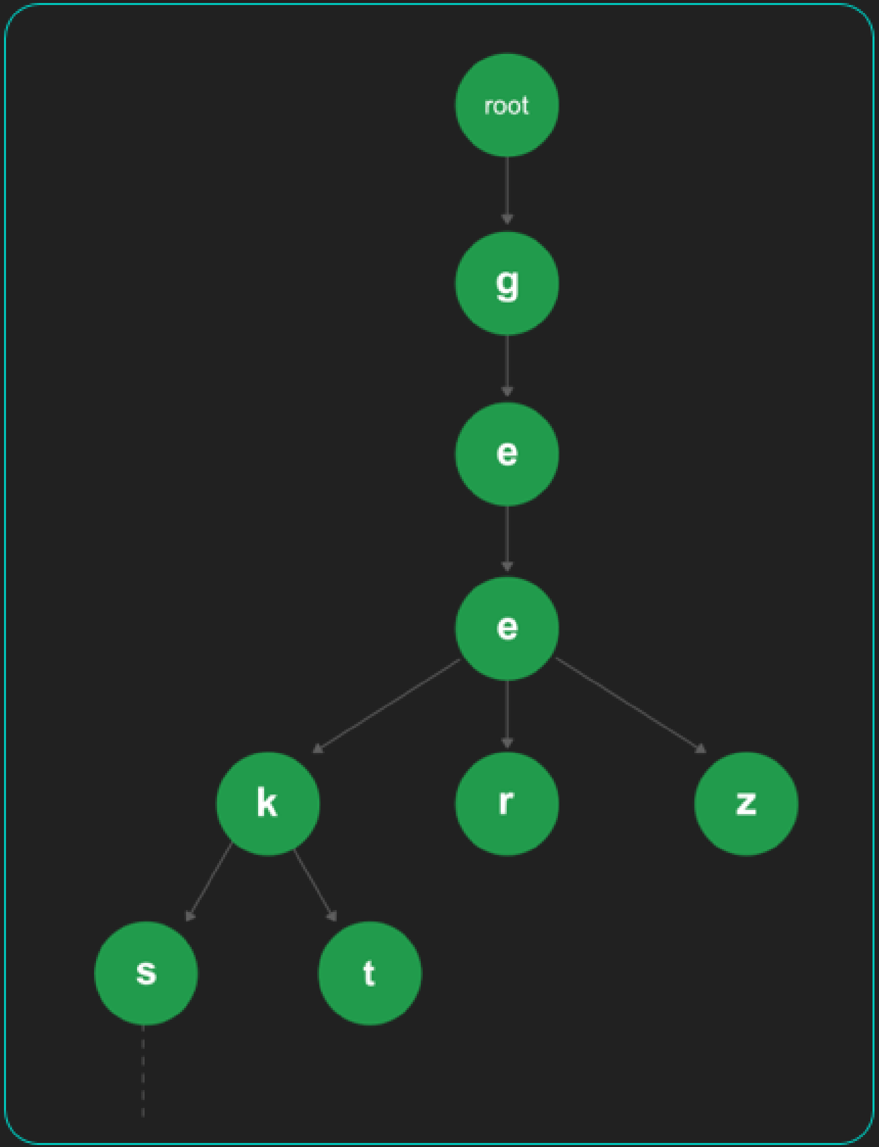
- Instead of static variables, there could be an opportunity to utilize dynamic programming for a function like this.
- Static variables mean that the nearest neighbor function (currently) cannot compute more than one proper nearest neighbor as the variables do not reset until the program is terminated.
- Cannot currently add two data points that contain the same x or y values as another node. (This will result in memory leaks).



```c
void NNHelper(node_t* treeNode, int nnX, int nnY) {
    static double smallestDistance = 0;
    static int xCoord = 0;
    static int yCoord = 0;

    if(treeNode == NULL) {
        return;
    }


    NNHelper(treeNode->left, nnX, nnY);

    //distance function
    double distance = 0;
    distance = sqrt((pow((treeNode->dataX - nnX), 2.0) + pow((treeNode->dataY - nnY), 2.0)));

    //two print statements
    printf("\n (%d,%d) is at a distance of %f to coordinate (%d, %d)  \n", nnX, nnY, distance, treeNode->dataX, treeNode->da
    printResultNN(nnX, nnY, smallestDistance, xCoord, yCoord);



    NNHelper(treeNode->right, nnX, nnY);

    if(nnX != treeNode->dataX && nnY != treeNode->dataY) {


        if(smallestDistance == 0) {
            smallestDistance = distance;
            xCoord = treeNode->dataX;
            yCoord = treeNode->dataY;
        }
        else if(distance < smallestDistance) {
            smallestDistance = distance;
            xCoord = treeNode->dataX;
            yCoord = treeNode->dataY;
        }
    }
```

Introduction to TRIES (Retrieve)

Trie is efficient because it always **tries** to reuse existing nodes.



BASIC STRUCTURING:

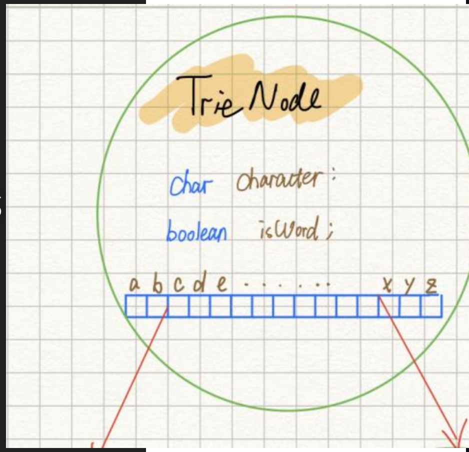
- Each node contains:

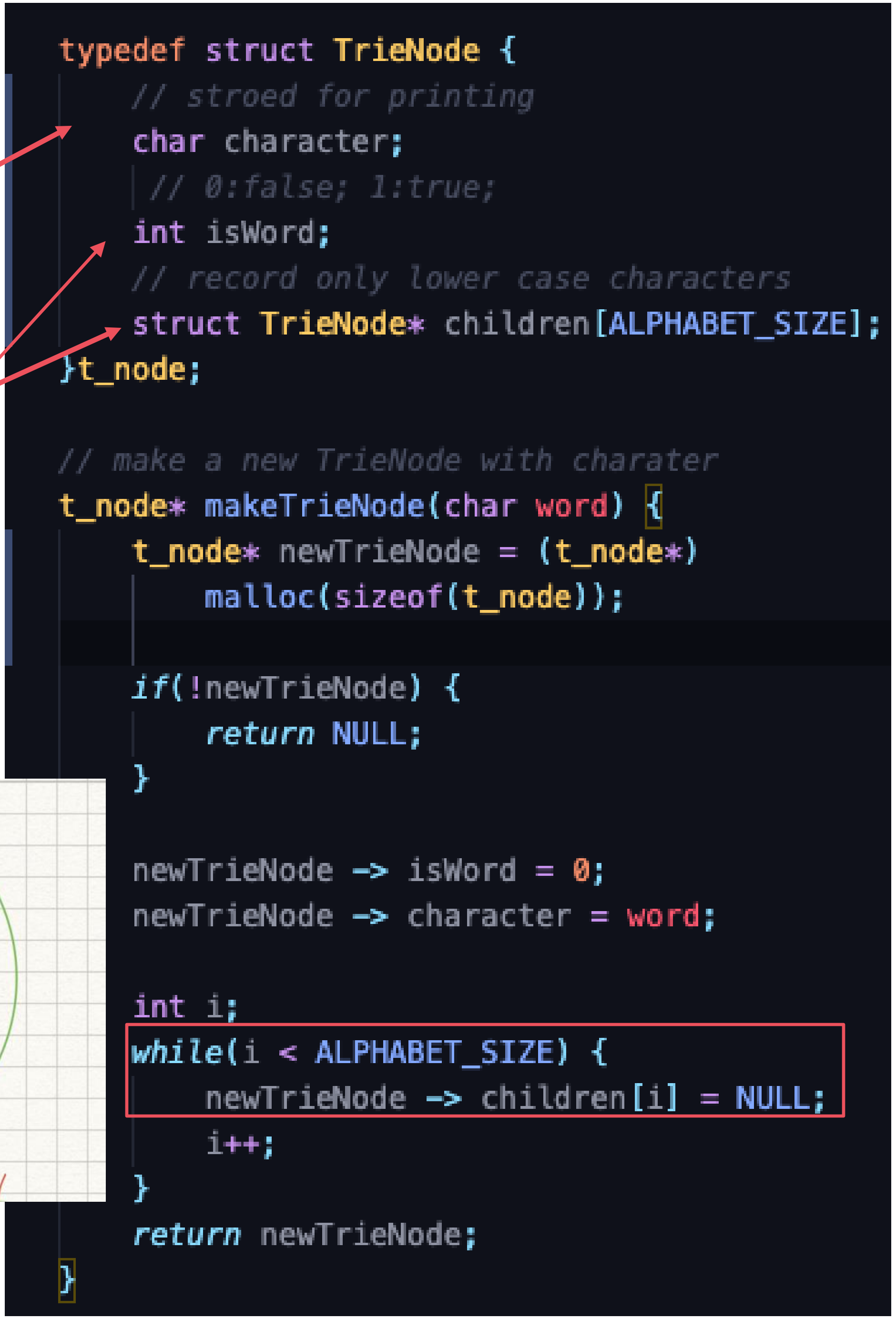
  - a letter,

  - an array of pointers for each letter of the alphabet,

  - and an indicator for the end of a word

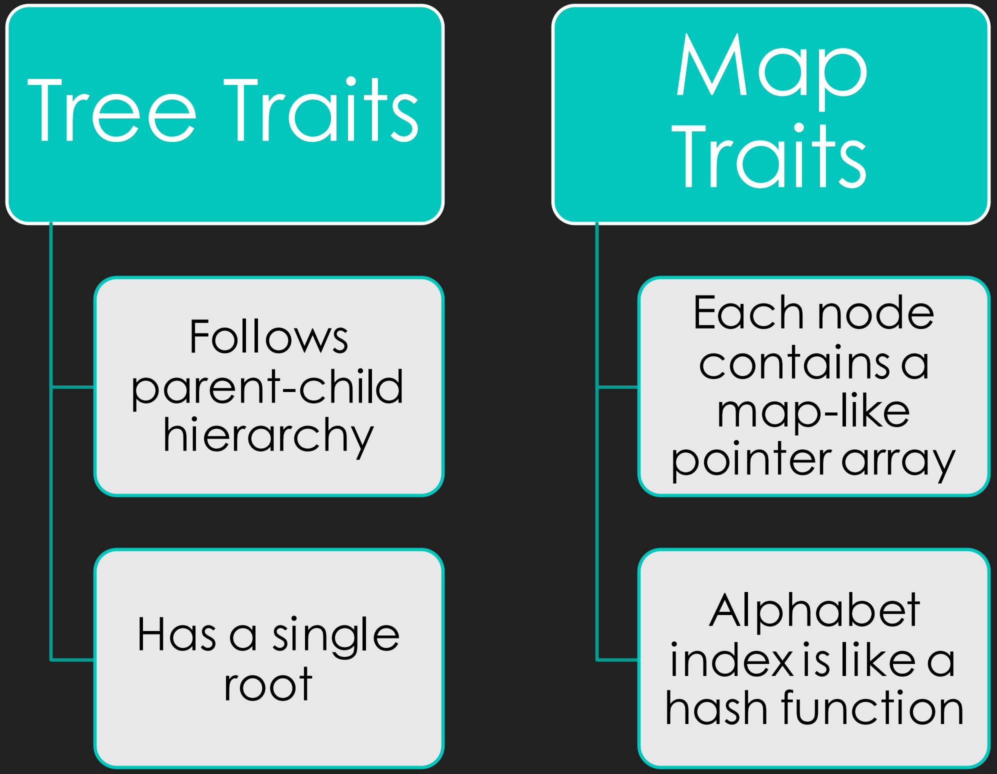
- The pointers assigned as letters are added to the Trie





```
typedef struct TrieNode {
    // stroed for printing
    char character;
    // 0:false; 1:true;
    int isWord;
    // record only lower case characters
    struct TrieNode* children[ALPHABET_SIZE];
}t_node;

// make a new TrieNode with charater
t_node* makeTrieNode(char word) {
    t_node* newTrieNode = (t_node*)
        malloc(sizeof(t_node));

    if(!newTrieNode) {
        return NULL;
    }

    newTrieNode -> isWord = 0;
    newTrieNode -> character = word;

    int i;
    while(i < ALPHABET_SIZE) {
        newTrieNode -> children[i] = NULL;
        i++;
    }
    return newTrieNode;
}
```

The Code:
Similarities to Trees and Maps

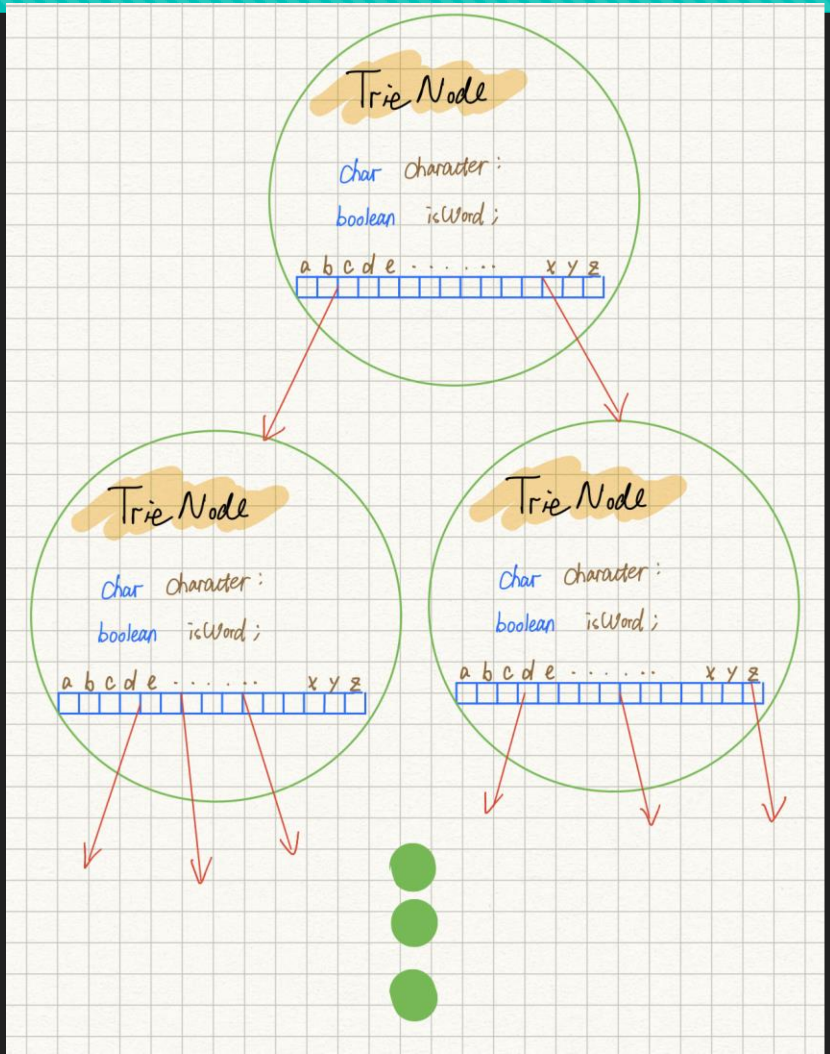


Tree Traits

- Follows parent-child hierarchy
- Has a single root

Map Traits

- Each node contains a map-like pointer array
- Alphabet index is like a hash function



The Code: Tradeoffs

Pros

No data collisions

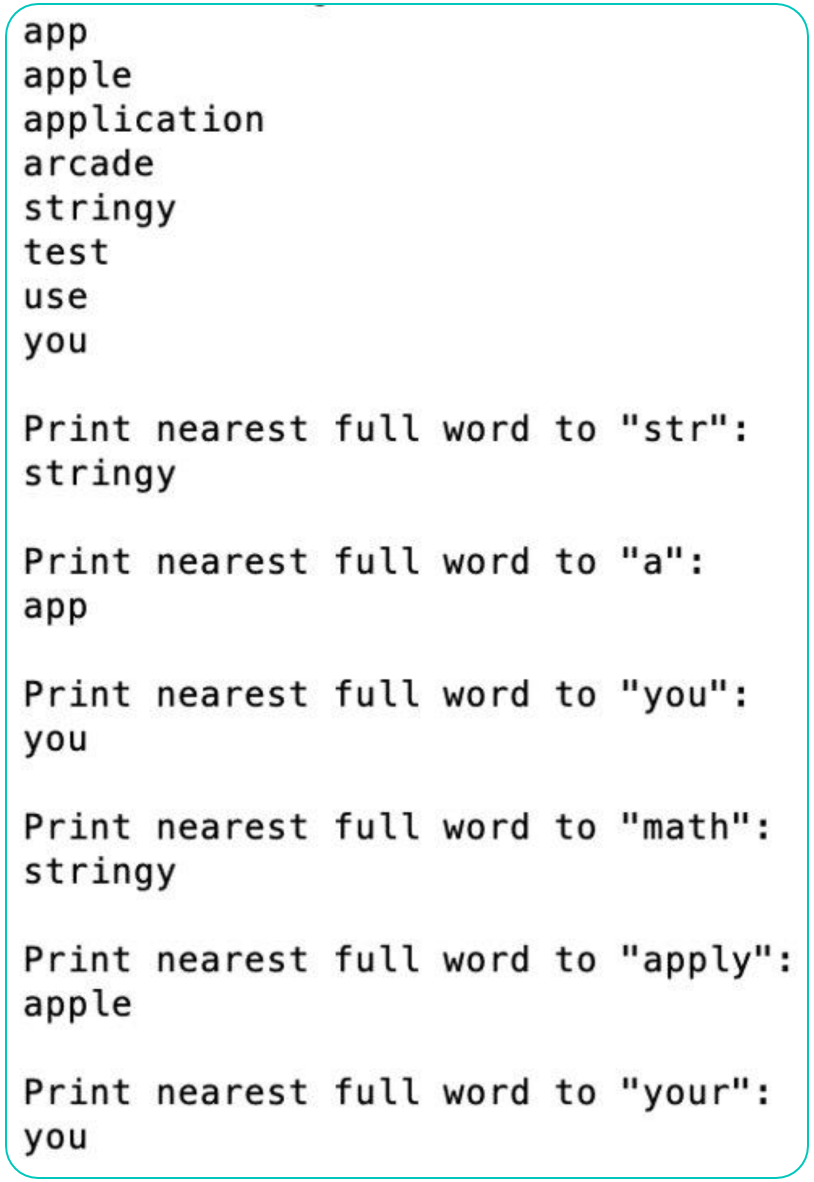
Worst-case look-up time is O(m), where m is length of string

Cons

Pointers may go unused

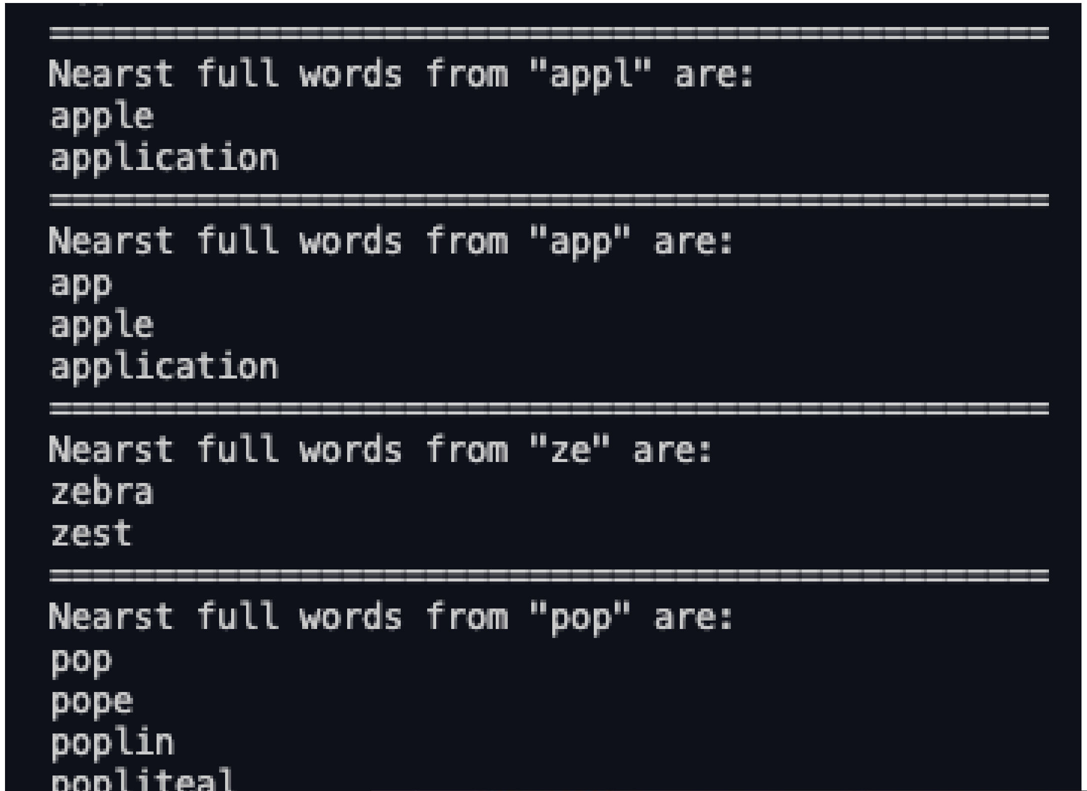
Large memory storage

Nearest Neighbor:

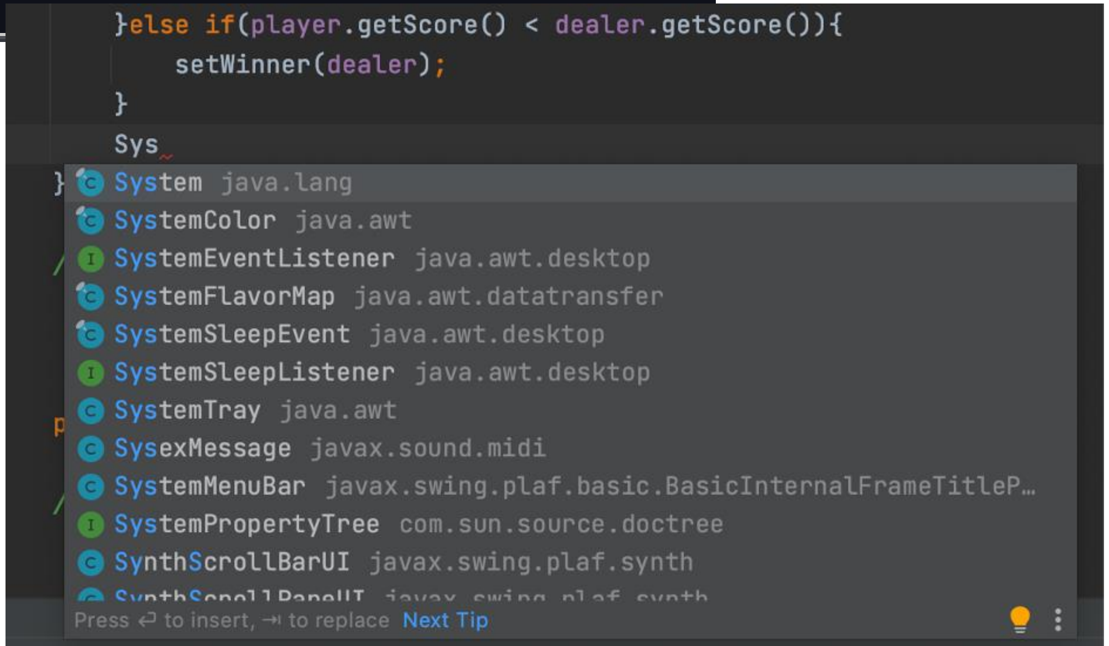


```
app
apple
application
arcade
stringy
test
use
you

Print nearest full word to "str":
stringy

Print nearest full word to "a":
app

Print nearest full word to "you":
you

Print nearest full word to "math":
stringy

Print nearest full word to "apply":
apple

Print nearest full word to "your":
you
```

- Tries can be used for autocorrect and autocomplete algorithms



```
================================================
Nearst full words from "appl" are:
apple
application
================================================
Nearst full words from "app" are:
app
apple
application
================================================
Nearst full words from "ze" are:
zebra
zest
================================================
Nearst full words from "pop" are:
pop
pope
poplin
popliteal
```

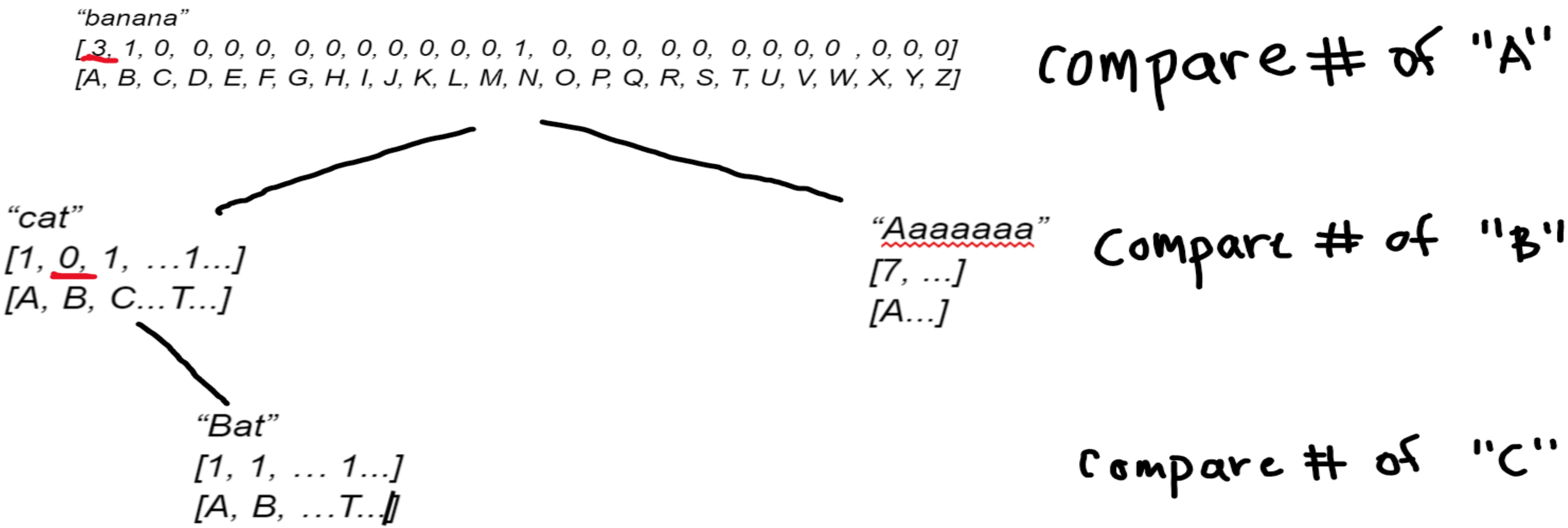


```
    }else if(player.getScore() < dealer.getScore()){
        setWinner(dealer);
    }
    Sys
} System  java.lang
  SystemColor  java.awt
  SystemEventListener  java.awt.desktop
  SystemFlavorMap  java.awt.datatransfer
  SystemSleepEvent  java.awt.desktop
  SystemSleepListener  java.awt.desktop
  SystemTray  java.awt
  SysexMessage  javax.sound.midi
  SystemMenuBar  javax.swing.plaf.basic.BasicInternalFrameTitleP…
  SystemPropertyTree  com.sun.source.doctree
  SynthScrollBarUI  javax.swing.plaf.synth
  SynthScrollPaneUI  javax.swing.plaf.synth
Press ⏎ to insert, ⇥ to replace  Next Tip
```

Nearest Neighbors:

- Tries can be used to provide suggestion lists in a Graphical User Interface
- or source code editing environment
- or recent history list in command line interpreters
- Or guess word game!

https://hryanjones.com/guess-my-word/



"banana"

[3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 , 0, 0, 0]

[A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z]

"cat"

[1, 0, 1, …1…]

[A, B, C…T…]

"Aaaaaaa"

[7, …]

[A…]

"Bat"

[1, 1, … 1…]

[A, B, …T…]

Potential Overlap of Two Structures

Customizable keyboard: how do we represent letters as numbers?

Index is point in alphabet and value is frequency of letter in word, k=26

Potential Overlap of Two Structures

- AI- based translation

Grandpa, I heard you are good at math. Could you help me with my homework?

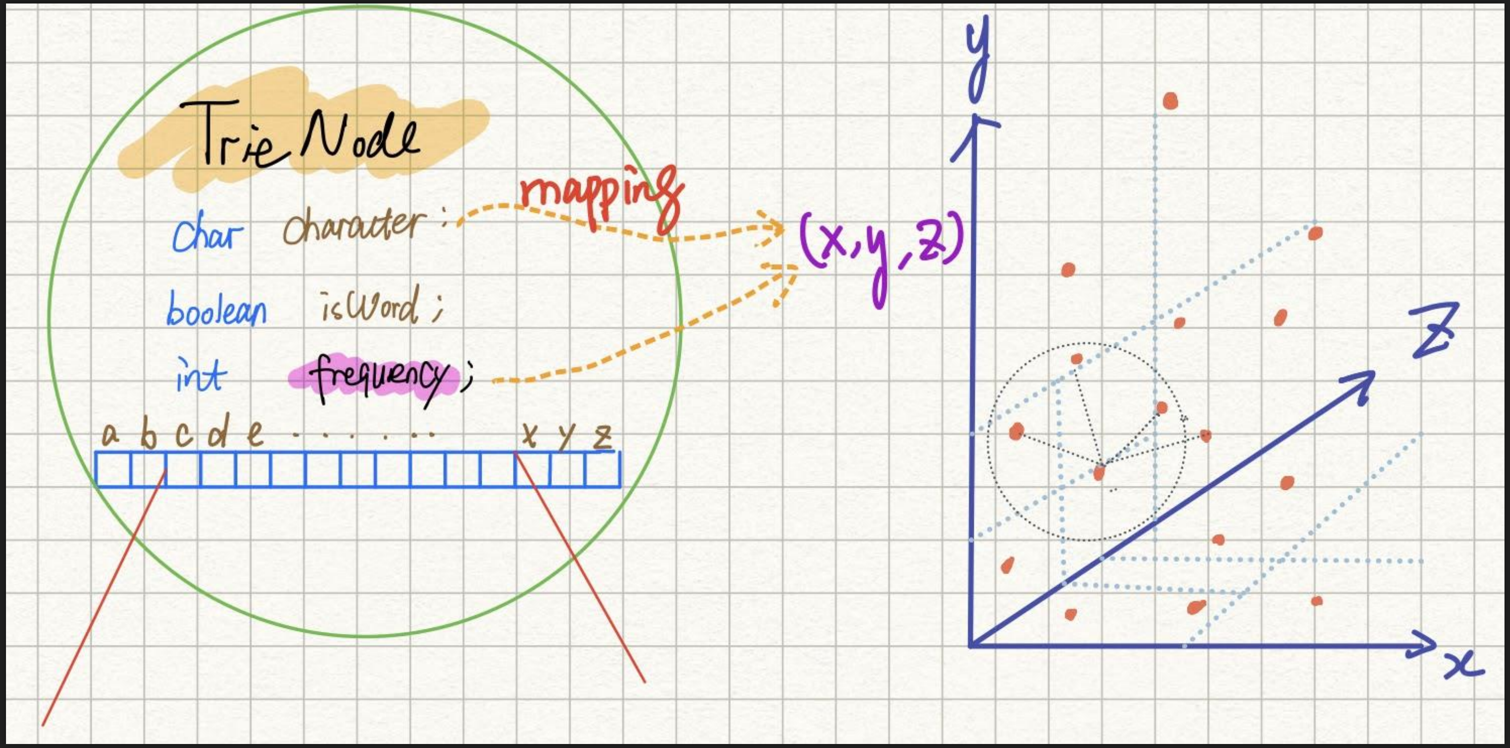
爷爷，我听说您数学很好，你能帮我做作业吗?

没问题。当然可以。

No problem. Of course.

i'm happy to tell you that the ligaments that support the knee are doing much better.

मैं आपको यह बताने में प्रसन्न हूं कि घुटने का समर्थन करने वाले स्नायुबंधन बहुत बेहतर कर रहे हैं।



Trie Node
char character ;
boolean isWord ;
int frequency ;
mapping
(x,y,z)
a b c d e . . . . . . x y z
y
z
x