Curtis Kargus

t = number of trees

b = number of business

Node(int n = 0) Space O(1) Time O(1)

void addUnderling(Node \* ptr, int id) Space O(1) Time O(1)

void addUnderling(Node \* ptr, Node \*underPtr) Space O(1) Time O(1)

void print(Node \* ptr, int level) Space O(n) Time O(n)

Node \* find(Node \* root, int target) Space O(n) Time O(n)

int minRaise(Node \* root, double percent) Space O(n) Time O(n)

Tree(Node \* rt = nullptr) Space O(1) Time O(1)

Tree(int id) Space O(1) Time O(1)

Node \* getRoot() Space O(1) Time O(1)

void printTree() Space O(n) Time O(n)

Node \* find(int target) Space O(n) Time O(n)

int minRaise(double percent) Space O(n lg(n)) Time O(n)

int minRaise(double percent) Space O(n lg(n)) Time O(n)

Forest() Space O(1) Time O(1)

void clear() Space O(1) Time O(1)

void print() Space O(n) Time O(tn)

Node \* find(int target) Space O(n) Time(tn)

void insert(int boss, int underling) Space O(n) Time(tn)

int main() Time(btn^2) Space(n)

Forest

-trees: vector<trees>

+minRaise(double percent) : int

+ Forest()

+clear()

+print()

+find(int target) : Node \*

+ insert(int boss, int underling)

Node

+ id : int

+ minRequests4Raise : int

+ underlings: vector<Node \*>

+ Node(int n = 0)

Tree

-root : Node \*

- print(Node \* ptr, int level)

- find(Node \* root, int target) : Node \*

- minRaise(Node \* root, double percent): int

+ Tree(Node \* rt = nullptr)

+ Tree(int id)

+ getRoot() : Node \*

+ printTree()

+ find(int target) : Node \*

+ minRaise(double percent) : int