Turner Trowbridge
11/2/2022
827959204

<table>
<tr>
<th>findClosestSharedManager Code</th>
<th>Time Complexity for Worst Case</th>
</tr>
<tr>
<td>

```
vector<Employee*> ances

Employee *Orgtree::findClosestSharedManager(Employee
*e1, Employee *e2, vector<Employee*> e1_ancestorList,
vector<Employee*> e2_ancestorList) {

   bool e1AtHead = false
   bool e2AtHead = false

  if(e1 == nullptr) {
     e1AtHead = true
  }

   if(e2 == nullptr) {
     e2AtHead = true
  }

  if(e1AtHead && e2AtHead) {
    if(e1_ancestorList.size() >= e2_ancestorList.size() {
      vector<Employee*> maxList = e1_ancestorList
      vector<Employee*> minList = e2_ancestorList
    } else
      vector<Employee*> maxList = e2_ancestorList
      vector<Employee*> minList = e1_ancestorList

 // compare lists using a data structure to find the lowest
matching employee_id between the two lists

return matchingEmployee
    }

   if(!e1AtHead){
   for (Employee *e: e1->getParentNode()) {
      return findClosestSharedManager(e, e2,
e1_ancestorList,
      e2_ancestorList)
      e1_ancestorList.push_back(e->getEmployeeID)
}

 if(!e2AtHead){
   for (Employee *e: e2->getParentNode()) {
      return findClosestSharedManager(e1, e,
e1_ancestorList,
```

</td>
<td>The worst case scenario for the findClosestSharedManger function would be two leaf nodes that are not in the same subtree after the head node and are both at the height of the tree. Therefore, their closest shared manager would be the head of the tree, making the size of the lists of the ancestors equal to the height of tree. This would make the time complexity for a leaf search O(h), where h is the height of the tree.</td>
</tr>
</table>

```
        e2_ancestorList)
        e2_ancestorList.push_back(e->getEmployeeID)
    }
return NULL
}
```