Before you start:

Homework Files

You can download the starter files for coding as well as this *tex* file (you only need to modify *homework3.tex*) on canvas and do your homework with latex. Or you can scan your handwriting, convert to pdf file, and upload it to canvas before the due date. If you choose to write down your answers by hand, you can directly download the pdf file on canvas which provides more blank space for solution box.

Submission Form

A pdf file as your solution named as ECE2810J_HW3_[Your Student ID]_[Your name].pdf uploaded to canvas
Estimated time used for this homework: **4-5 hours.**

0 Student Info

Your name and student id:

Solution: Gong Zimu 520370910037

1 Choices (20 points)

1.1 Shortest distance

Choose the shortest distance between Node 4 and Node 6.

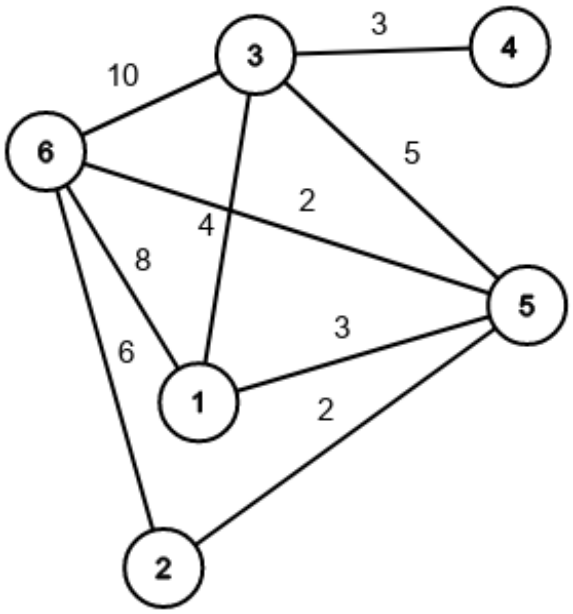

Figure 1: Distance

A) 3

B) 13

C) 10

D) 0

Solution: C

1.2 Binary indexed tree

Binary indexed tree is like a special kind of array that can help to calculate sum from a[0] to a[n] while you can easily access or edit elements in it. Here is one possible implement. Please choose the time complexity of *prefix_sum* and *add*.

```
//N==1+2^a
int A[N];

int LSB (int i)
{
    return  ((i) & -(i));
}
int prefix_sum(int i)
{
  int sum = A[0];
  for (; i != 0; i -= LSB(i))
    sum += A[i];
  return sum;
}
void add(int i, int delta)
{
  if (i == 0)
    {
    A[0] += delta;
    return;
  }
  for (; i < SIZE; i+= LSB(i))
    A[i] += delta;
}
```

A) O(N) and O(N)

B) O(N) and O(log N)

C) O(log N) and O(N)

D) O(log N) and O(log N)

Solution: D

1.3 Heap

What will the heap will be after push 20, 11. The initial heap contains : 18, 25, 32, 77, 86, 35, 93, 80

A) 18, 25, 32, 77, 86, 35, 93, 80, 20, 11

B) 11, 18, 20, 25, 32, 77, 86, 35, 93, 80

C) 11, 18, 32, 25, 20, 35, 93, 80, 77, 86

D) 11, 18, 25, 32, 77, 86, 35, 93, 80, 20

Solution: C

1.4 Spanning trees

The number of distinct minimum spanning trees for the following weighted graph is

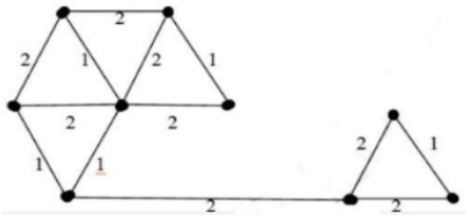

Figure 2: The weighted graph

A) 4

B) 5

C) 6

D) 7

Solution: C

2 Binary Search Tree (30 points)

2.1 Simple simulation (16 points)

Perform the following operations to construct a binary search tree. Show the result of the BST after each operation in either tree form or array form (level order traversal with null pointer marked). For deletion, if it is a two-degree node, replace with the largest key in the left subtree.

a) Insert 21, 29, 25, 18, 19, 32, 15, 37, 22, 6, 30, 17 (3 points)

Solution:
21, null, null, null, null, null, null, null, null, null, null, null, null, null, null
21, null, 29, null, null, null, null, null, null, null, null, null, null, null, null
21, null, 29, null, null, 25, null, null, null, null, null, null, null, null, null
21, 18, 29, null, null, 25, null, null, null, null, null, null, null, null, null
21, 18, 29, null, 19, 25, null, null, null, null, null, null, null, null, null
21, 18, 29, null, 19, 25, 32, null, null, null, null, null, null, null, null

21, 18, 29, 15, 19, 25, 32, null, null, null, null, null, null, null, null
21, 18, 29, 15, 19, 25, 32, null, null, null, null, null, null, null, 37
21, 18, 29, 15, 19, 25, 32, null, null, null, null, 22, null, null, 37
21, 18, 29, 15, 19, 25, 32, 6, null, null, null, 22, null, null, 37
21, 18, 29, 15, 19, 25, 32, 6, null, null, null, 22, null, 30, 37
21, 18, 29, 15, 19, 25, 32, 6, 17, null, null, 22, null, 30, 37

b) Delete 22 (3 points)

Solution:
21, 18, 29, 15, 19, 25, 32, 6, 17, null, null, 22, null, 30, 37
21, 18, 29, 15, 19, 25, 32, 6, 17, null, null, null, null, 30, 37

c) Delete 18 (3 points)

Solution:
21, 18, 29, 15, 19, 25, 32, 6, 17, null, null, null, null, 30, 37
21, 17, 29, 15, 19, 25, 32, 6, 17, null, null, null, null, 30, 37
21, 17, 29, 15, 19, 25, 32, 6, null, null, null, null, null, 30, 37

d) Insert 20 (3 points)

Solution:
21, 17, 29, 15, 19, 25, 32, 6, null, null, null, null, null, 30, 37
21, 17, 29, 15, 19, 25, 32, 6, null, null, 20, null, null, 30, 37

e) What is the in-order predecessor of node 25? What is the post-order successor of node 17? (4 points)

Solution:
In-order predecessor of node 25: node 21
Post-order successor of node 17: node 25

2.2 Better than linear selection? (13 points)

After learning the application of the BST, Jeremy thinks that with BST, rank search can be done either in a faster way or with less space compared with the linear selection algorithms introduced in the lecture.

a) Compare BST rank search with deterministic selection algorithm in terms of time complexity and space usage. Assume that for each node, a variable $left\_size$ is maintained as introduced in the lecture. (7 points)

Solution:
BST rank search:
Time Complexity: $O(\log n)$
Space usage: Traverse through $\log k$ nodes to find the k-th node.
Deterministic selection algorithm:
Time Complexity: $O(n)$
Space usage: Traverse through $\log k$ nodes on average.

b) However, master of algorithm, has a different idea again. He states that for the linear selection algorithm, the input can be arbitrary array while for BST rank search, you have to first build up a binary search tree from the arbitrary array, which will also take some time. Therefore, the average time complexity of BST rank search is not better than $O(n)$. Is he right? (7 points) Assume that we are working on a fixed array and doing lots of rank search.

Solution:
He would be correct if we are working on multiple different arrays and doing a few rank searches because it takes time to initialize every array into BST.

However, he would not be correct since we are working on a fixed array and doing lots of rank searches. We can average the time to initialize the BST in every single rank seearch, and initialize time would not be long since we only need to initialize once.

3 Proof Questions(30 points)

3.1 Order of deletion(5 points)

Does the deletion of two nodes in BST exchangeable? For example, we want to delete node x and y, if the result of first deleting x then y is the same with first y then x, then it's called exchangeable. If you agree, give a brief proof. If you disagree, give a example.

Solution:
Deletion of two nodes in BST is not commutative.
Counterexample:
Consider a BST A, B, C, null, D, null, null, and we want to delete node A and C.
Case 1: Delete A, then C
After deleting A, the BST becomes: D, B, C
After deleting C, the BST becomes: D, B, null
Case 2: Delete C, then A
After deleting C, the BST becomes: A, B, null, null, D, null, null
After deleting A, the BST becomes: B, null, D
The two results are not the same!
Thus, deletion of two nodes in BST is not commutative.

3.2 Transverse of a tree(10 points)

In the three orders of transverse taught in class, we will use the recursion to do the transverse. And can you give a way of transverse without using recursion? And also show what kind of transverse it is. (One simple way is to use the idea of topological sort taught in class, and you can also try to think how to do all the three tranverses, but it's not required)

Solution:
Based on a queue.
Algorithm:

1. Enqueue the root node into a queue.
2. While the queue is not empty:
   (a) Dequeue a node v from the queue and visit it.
   (b) Enqueue the children of the node into the queue.

4 Appliaction(20 points)

4.1 New Edge in BST(5 points)

Suppose we now have a tree. Try to prove that, if a new edge is added, there should be a cycle.

Solution:
By definition, a tree is connected and has no circuits.
Let u, v be any two nodes of tree T.
Then there must exist a path $(u, u_1, u_2, ..., u_{n-1}, v)$ from u to v.
If a new edge $\{u, v\}$ is added.
Then $(u, u_1, u_2, ..., u_{n-1}, v, u)$ is now a cycle.
Proved.

4.2 New Edge in MST(5 points)

Suppose edge (u, v) is one edge in a connected graph with the minimum weight. Prove that this edge is in at least one minimum span tree of the graph.

Solution:
Suppose none of the MSTs contain edge (u, v).
By definition, all the MSTs are connected.
Then, for one of the MSTs T, there must exist a path from u to v.
However, since edge (u, v) has the mimimum weight, the path $(u, u_1, u_2, ..., u_k, v)$ must be longer than the edge, then T is not a MST, contradiction.
Thus, one of the MSTs must contain edge (u, v).

4.3 K's smallest number(10 points)

Suppose we have a binary search tree. Output the k's smallest number of the tree. Write the pseudocode or C++ code under. You can just call the root of the tree *root*
The difinition of the node and function in C++ is shown below:

```
Struct TreeNode{
   int data;
   TreeNode* left, right;
}
int kthSmallest(TreeNode* root, int k)

```

Write your answer here.

```
TreeNode* getMin(TreeNode* node) {
    while (node->left) {
        node = node->left;
    }
    return node;
}
TreeNode* increment(TreeNode* node) {
    if (node->right) {
        node = node->right;
        return node;
    }
    else {
        while (node->parent!=nullptr && node->parent->left!=node)
            {
                node = node->parent;
            }
            node = node->parent;
        }
    }
}
TreeNode* ksSearch(TreeNode* root, int k) {
    if (root == nullptr) return nullptr;
    TreeNode* node = getMin(root);
    for (auto i = 0, i<k, i++) {
        increment(node);
    }
    return node;
}

```

5 Coding question for you to practice

This part will not be counted for you homework score, just for you to practice.

1. Leetcode 207 course-schedule
2. Leetcode 114 flatten-binary-tree-to-linked-list
3. Leetcode 100 same-tree
4. Leetcode 130 surrounded-regions
5. Leetcode 102 binary-tree-level-order-traversal
6. Leetcode 127 word-ladder
7. Leetcode 208 implement-trie-prefix-tree
8. Leetcode 218 the-skyline-problem
9. Leetcode 18 4sum
10. Leetcode 264 ugly-number-ii