HW4 is about AVL Trees. There are two problems:

1. Print out nodes of the AVL tree in breadth-first order;
2. Compute the minimum-cost path from the root to a leaf in the AVL tree, for which the total sum of absolute differences **|child.value - parent.value|** is minimum.

1. General Instructions

For this assignment you will implement functions in C for an AVL Tree. You are given a skeleton C code in the following files:

- **avl.h**
- **avl.c**
- **main.c**

and the file with values that should be added to the AVL Tree for testing your code:

- **input.txt**

and a make file for compiling the code on the School's server:

- **Makefile**

Complete all functions in the skeleton code. You should use the same names of files, functions, and variables as we specified in the skeleton code. This is because our grading will be based on a script that assumes that you use the same names. Make sure your code compiles without errors using the provided makefiles for the two abstract data structures on the ENGR server.

To compile on the server, enter the command: make
To run and test your code on the server, enter the program argument: ./prog input.txt

For testing, use your own, different input.txt files. Generate a large number of random numbers and save them in the file input.txt, so that they can be added to the AVL tree in the main function. Test the case when the size of the AVL tree is 10000 nodes. For this homework, we will not use larger trees than 10000 nodes.

What to submit: Please submit the completed avl.c and main.c via TEACH by the deadline. Please do not submit the header, compiled object or executable files.

Questions regarding HW4 should be posted to HW4 Discussions on Canvas for fast response.

2. AVL Tree

The main function in main.c performs the following:

- Initializes an empty AVL tree.
- Reads values from an input file and adds them to the AVL tree.
- Prints on the terminal all values of the AVL tree in the breadth-first fashion.

2. AVL Tree

The main function in main.c performs the following:

- Initializes an empty AVL tree.
- Reads values from an input file and adds them to the AVL tree.
- Prints on the terminal all values of the AVL tree in the breadth-first fashion.
- Finds the minimum-cost path in the AVL tree (see details below).
- Prints the execution time for finding the min-cost path, on the terminal.
- Prints values of nodes that lie on the minimum-cost path in the AVL tree, on the terminal. The order of nodes in the print-out must strictly follow their order in the min-cost path from the root to the leaf.
- Prints the cost of the min-cost path of the AVL tree.

Your task is to complete functions in **avl.c** and **main.c** marked by "FIX ME". The comments before each function are aimed at clarifying the function's input arguments, outputs, and what the function is supposed to do. Note that we will test your code with another input.txt file.

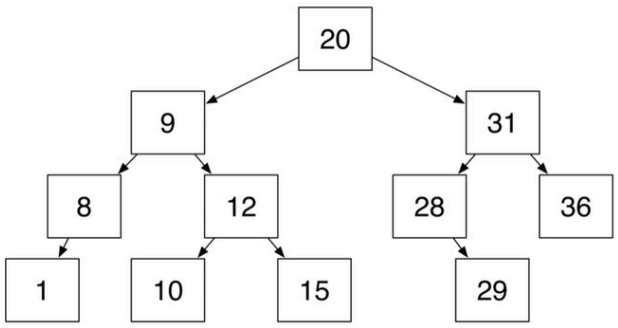
3. Specification of the Min-Cost Path

The minimum-cost path (or min-cost path) of an AVL tree is defined as a path from the root to a leaf for which the total sum of absolute differences **|child.value - parent.value|** is minimum. More formally, let π denote a path from the root to a leaf of the AVL tree, and i and j denote child and parent nodes, respectively. Then, we define the minimum-cost path π^* as

$$\pi^* = \arg\min_\pi \sum_{\substack{\text{forall } i \in \pi \\ j \text{ is parent of } i}} |i.\,\text{value} - j.\,\text{value}|$$

When i indicates the root in the formula above, we assume that $|i.\,\text{value} - j.\,\text{value}| = i.\,\text{value}$.

For example, the AVL tree shown in the figure below has the following paths:



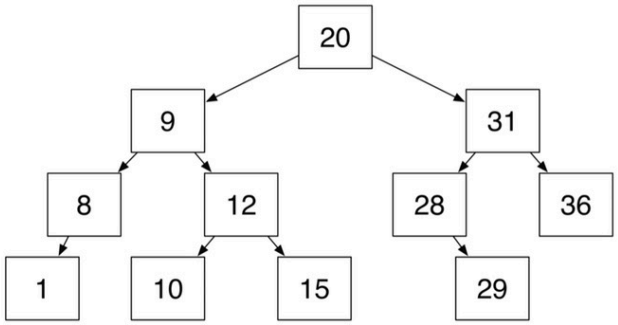
3. Specification of the Min-Cost Path

The minimum-cost path (or min-cost path) of an AVL tree is defined as a path from the root to a leaf for which the total sum of absolute differences **|child.value - parent.value|** is minimum. More formally, let π denote a path from the root to a leaf of the AVL tree, and i and j denote child and parent nodes, respectively. Then, we define the minimum-cost path π^* as

$$\pi^* = \arg\min_\pi \sum_{\substack{\text{forall } i \in \pi \\ j \text{ is parent of } i}} |i.\,\text{value} - j.\,\text{value}|$$

When i indicates the root in the formula above, we assume that $|i.\,\text{value} - j.\,\text{value}| = i.\,\text{value}$.

For example, the AVL tree shown in the figure below has the following paths:



$\pi_1 = 20 - 9 - 8 - 1\ ,\ \text{cost} = 19$
$\pi_2 = 20 - 9 - 12 - 10\ ,\ \text{cost} = 16$
$\pi_3 = 20 - 9 - 12 - 15\ ,\ \text{cost} = 17$
$\pi_4 = 20 - 31 - 28 - 29\ ,\ \text{cost} = 15$
$\pi_5 = 20 - 31 - 36\ ,\ \text{cost} = 16$

Using the above formula, in this example, the minimum-cost path is:

$\pi^* = \pi_4 = 20 - 31 - 28 - 29$

4. Grading policy: Max 100 points

Your code must compile without errors and run without any memory issues so that we can grade it.

- 30 points: The code prints out all nodes of the AVL tree **but not** in the breadth-first order, and **does**

$\pi_1 = 20 - 9 - 8 - 1\ ,\ \text{cost} = 19$
$\pi_2 = 20 - 9 - 12 - 10\ ,\ \text{cost} = 16$
$\pi_3 = 20 - 9 - 12 - 15\ ,\ \text{cost} = 17$
$\pi_4 = 20 - 31 - 28 - 29\ ,\ \text{cost} = 15$
$\pi_5 = 20 - 31 - 36\ ,\ \text{cost} = 16$

Using the above formula, in this example, the minimum-cost path is:

$\pi^* = \pi_4 = 20 - 31 - 28 - 29$

4. Grading policy: Max 100 points

Your code must compile without errors and run without any memory issues so that we can grade it.

- 30 points: The code prints out all nodes of the AVL tree **but not** in the breadth-first order, and **does not** compute the correct minimum-cost path.
- 50 points: The code prints out all nodes of the AVL tree in the breadth-first order, and **does not** compute the correct minimum-cost path.
- 75 points: The code prints out all nodes of the AVL tree in the breadth-first order, and computes the correct minimum cost, but does not print out nodes of the min-cost path in the correct order from the root to the leaf (e.g., some nodes are missing, or wrongly swapped).
- 90 points: The code prints out all nodes of the AVL tree in the breadth-first order, and nodes of the min-cost path in the correct order from the root to the leaf, but for a tree with 10000 nodes computing takes longer than 250 microseconds on the School's server.
- 100 points: all conditions for 90 points and computing takes less than 250 microseconds for a tree with 10000 nodes on the School's server.