# ECE250 – Project 4

# msttest
# Design Document

Liahng-Jyun, Liu, UW UserID: lj2liu
March 31th,2020

## 1. Overview of Classes

node- to store user input for edge weight and vertice- a fundatoin of the net work.
mst- the mst by the Greedy Method sorts each input nodes (one node contains the edge weight and the two vertices that forms it) from minimum weight cost to maximum weight cost. A disjoing set helps tracking cycyles and disconections when mst total weight is called.
illegal_argument- this will try catching invalid input in msttest.cpp directly from user input and will be catched by msttest to cout a *failure message. (*failure type specified in msttest).

**Class: node**

**Description:**
to store user input for edge weight and vertice.
**Member variables:**
This class has double weight, int child and int parent. Parent is defined as the vertex with smaller value.
**Member functions (operations) –** no operations, this class only stores data.

| **node** |
| --- |
| weight: double<br>child: int<br>parent: int |
| node(node const &p2): copy constructor.<br>node(): constructor<br>node(int parent_in, int child_in,double weight_in): constructor |

**Class: mst**

**Description:**

The mst by the Greedy Method sorts each input nodes (one node contains the edge weight and the two vertices that forms it) from minimum weight cost to maximum weight cost. A disjoing set helps tracking cycyles and disconections when mst total weight is called.

**Member variables:**
minimum spanning tree weight count is recorded by mst_weight. The number of vertices in a mst (defined as always being the first input of msttest) is given by int m. The not_con of boolean type helps identify if disconnection has occured when calling mst. The array tracks parent with recursion with an operation of O(m).vector<int> check contains the vertex as array index, and the content check[vertex] equal to a positive or negative value. Positive means the parent vertex, negative means the vertex itseld is a parent of this many verices (including itself). ie. if it's (-2), then the vertex is in a network with two vertices.The network array tracks degree of each vertex O(1) with the member variable, vector<int> degree. The index is used as vertex number and the degree is is the corresponding content (ie degree[vertex]).The dynamically growing size upon user insertions is stored in net of node vector type. A sort() is used to find the minimum sort. O(m*log (m)). To check if disconnection occurs, the integer array vertex_count keep tracking initiated vertex. This uses index as vertex and vertex_count[vertex]= 1 means the vertex is connected by some sort of edges; 0 means it's disconnected.
**Member functions (operations) –**

- del: deletion on edges; if deleted or found, then true. otherwise, false.
- degreev: checking the degree of the node v by looping through the net work, the output is the original network vertex degree
- sortnet: for vector net sorting from low to high edge weight.
- findparent: a recursion for finding parents using the check array.
- mintree: this mintree compute the minimum cost edge weight of a network with at most m-1 edges. It uses the check array to filter out cycles.

| **mst** |
| --- |
| mst_weight: double<br>m: int<br>not_con: bool<br>check: int vector<br>degree: int vector<br>net: node vector<br>vertex_count: int pointer |

| |
|---|
| insert(int parent_in, int child_in, double weight_in): void<br>cleanptr(node* target): void<br>del(int parent_in, int child_in): bool<br>degreev(int v): int<br>sortnet() : void<br>findparent(): int<br>mintree(): void |

**Class: illegal_argument**

**Description:** this will try catching invalid input in msttest.cpp directly from user input and cout a failure message.

**Member variables:** not aplicatble.

**Member functions (operations) –** not aplicatble.

## 2. Constructors/Destructor/Operator overloading

class node: needs 1 edge and 2 vertices to start with. Support copy constructor with all 3 info given.
class mst: As per msttest, there is only one tree but a copy constructor available for duplications. All functions are avalible if and only if a valid m is given as the first command of msttest.

## 3. Test Cases

For class tree:
Test 1 : let n=0 and perform all mst functions
Test 2 : create a valid network and call mst. Clear all and call all mst member functions
Test 3 : Create a network of two subtrees that are not connected and run mst.
Test 4 : Create a network without cycle and are connected and run mst.
Test 5 : Create a network of cycles are connected and run mst.

## 4. Performance

Functions such as insert, cleanptr takes O(1). The degreev(the degree of a vertex) funcion takes O(n) and Ω(n) with n= the number of edges in the network. Hence, average is n. sortnet recursively look for parent (findparent function) with complexity of O(m) and Ω(1) with average A(m/2) with m= number of vertex given in the network.
mst function calls the recursion of finding parents (the findparent) and sortnet with the c++ sort(), which takes O(nlog(n))=average complexity. Since nlog(n) overweights linear complexity, the worst case and average complexity is nlog(n). reference to sort() complexity :https://en.cppreference.com/w/cpp/algorithm/sort.