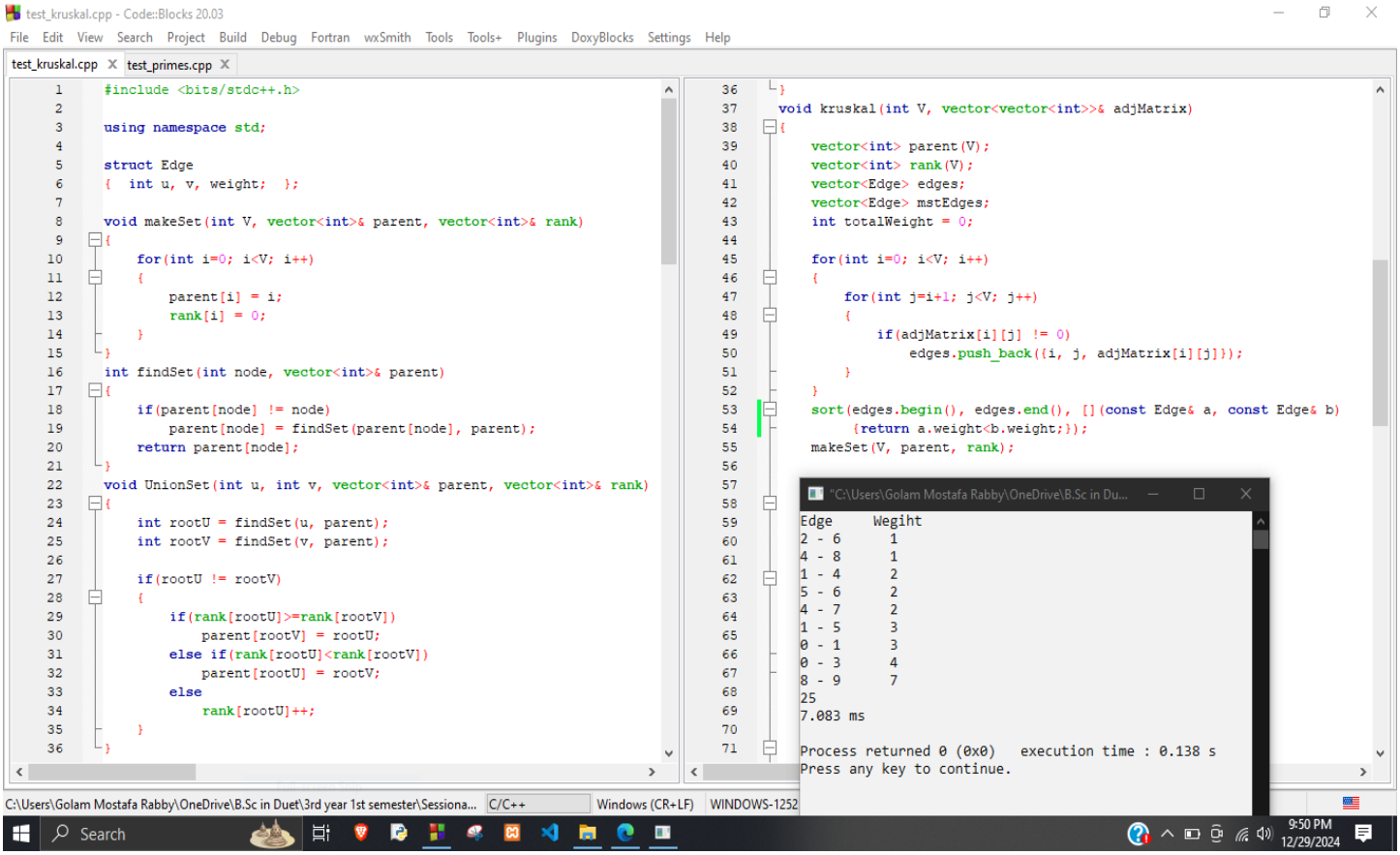
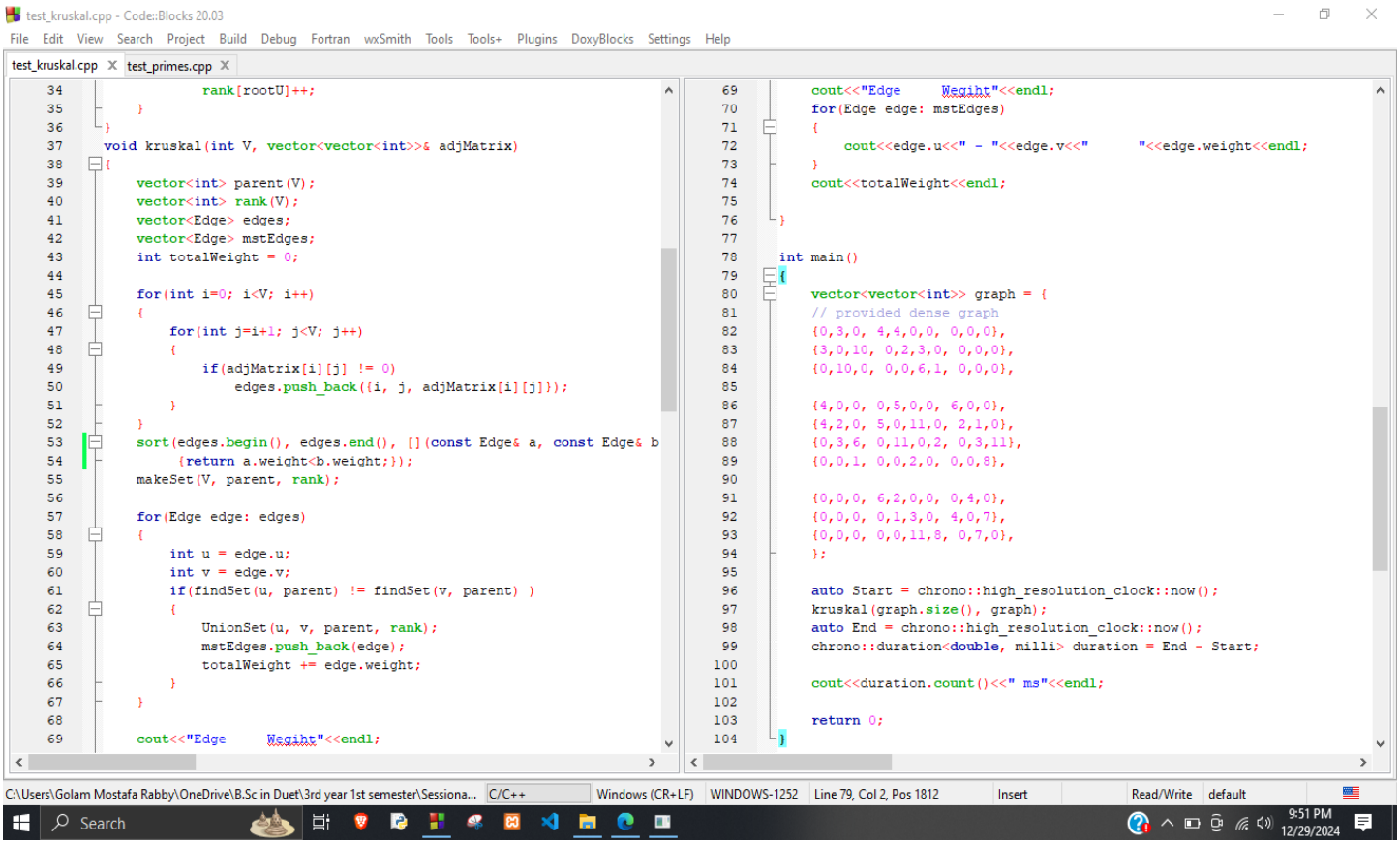
Dense Graph:

Kruskal's Algorithm



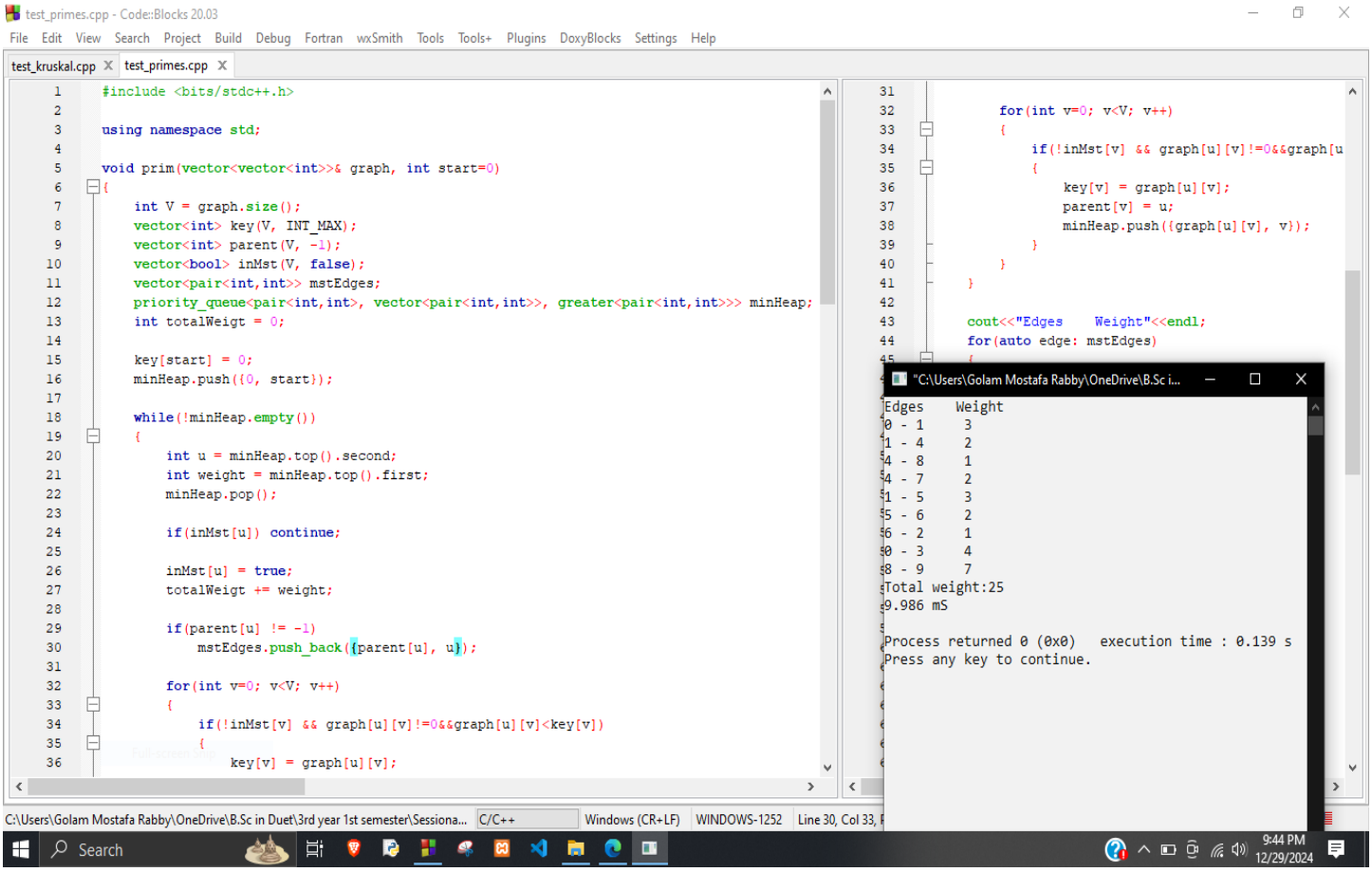
```cpp
#include <bits/stdc++.h>

using namespace std;

struct Edge
{  int u, v, weight;  };

void makeSet(int V, vector<int>& parent, vector<int>& rank)
{
    for(int i=0; i<V; i++)
    {
        parent[i] = i;
        rank[i] = 0;
    }
}
int findSet(int node, vector<int>& parent)
{
    if(parent[node] != node)
        parent[node] = findSet(parent[node], parent);
    return parent[node];
}
void UnionSet(int u, int v, vector<int>& parent, vector<int>& rank)
{
    int rootU = findSet(u, parent);
    int rootV = findSet(v, parent);

    if(rootU != rootV)
    {
        if(rank[rootU]>=rank[rootV])
            parent[rootV] = rootU;
        else if(rank[rootU]<rank[rootV])
            parent[rootU] = rootV;
        else
            rank[rootU]++;
    }
}
void kruskal(int V, vector<vector<int>>& adjMatrix)
{
    vector<int> parent(V);
    vector<int> rank(V);
    vector<Edge> edges;
    vector<Edge> mstEdges;
    int totalWeight = 0;

    for(int i=0; i<V; i++)
    {
        for(int j=i+1; j<V; j++)
        {
            if(adjMatrix[i][j] != 0)
                edges.push_back({i, j, adjMatrix[i][j]});
        }
    }
    sort(edges.begin(), edges.end(), [](const Edge& a, const Edge& b)
         {return a.weight<b.weight;});
    makeSet(V, parent, rank);
```

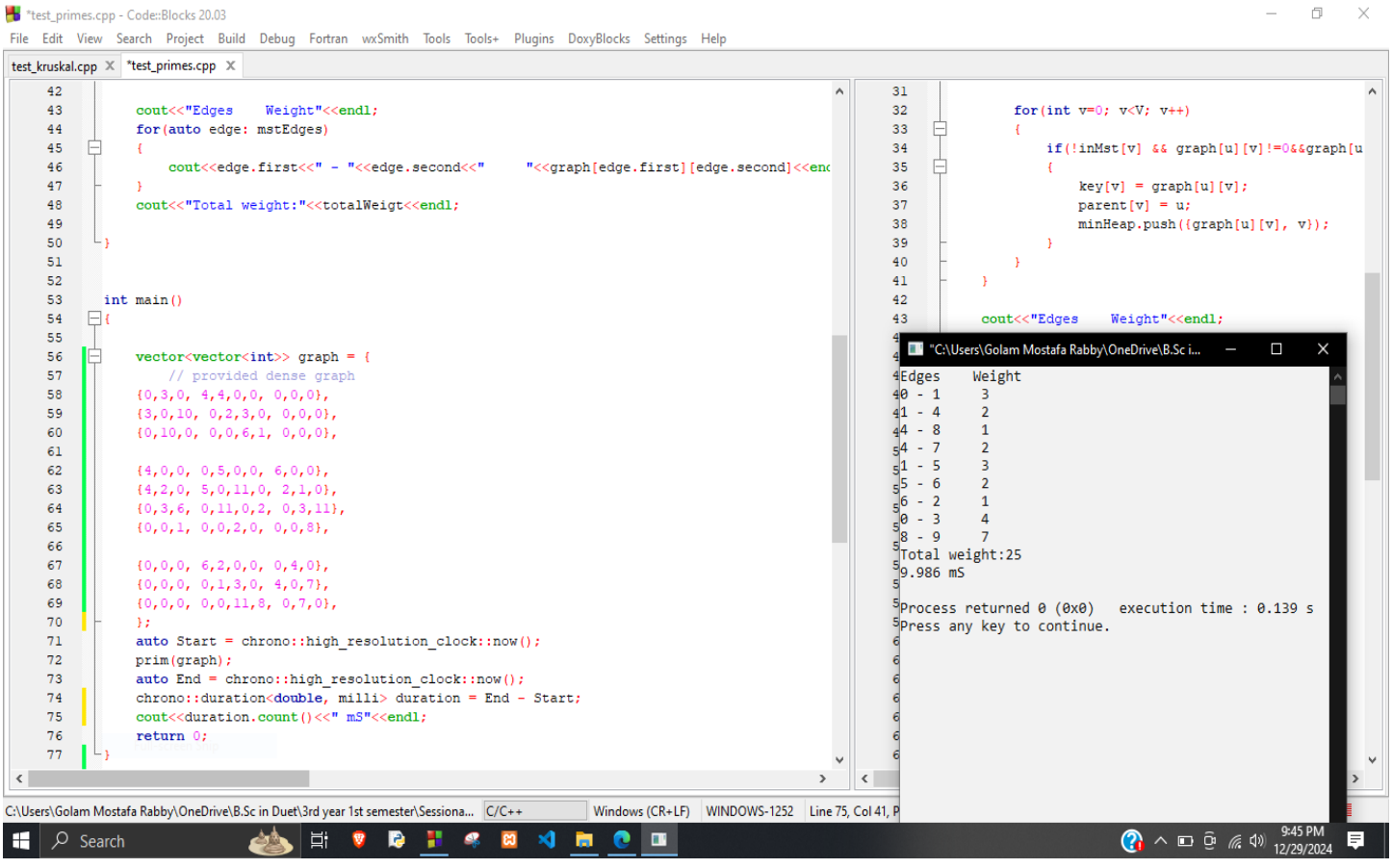
```
"C:\Users\Golam Mostafa Rabby\OneDrive\B.Sc in Du...
Edge     Wegiht
2 - 6      1
4 - 8      1
1 - 4      2
5 - 6      2
4 - 7      2
1 - 5      3
0 - 1      3
0 - 3      4
8 - 9      7
25
7.083 ms

Process returned 0 (0x0)   execution time : 0.138 s
Press any key to continue.
```



```cpp
            rank[rootU]++;
    }
}
void kruskal(int V, vector<vector<int>>& adjMatrix)
{
    vector<int> parent(V);
    vector<int> rank(V);
    vector<Edge> edges;
    vector<Edge> mstEdges;
    int totalWeight = 0;

    for(int i=0; i<V; i++)
    {
        for(int j=i+1; j<V; j++)
        {
            if(adjMatrix[i][j] != 0)
                edges.push_back({i, j, adjMatrix[i][j]});
        }
    }
    sort(edges.begin(), edges.end(), [](const Edge& a, const Edge& b
         {return a.weight<b.weight;});
    makeSet(V, parent, rank);

    for(Edge edge: edges)
    {
        int u = edge.u;
        int v = edge.v;
        if(findSet(u, parent) != findSet(v, parent) )
        {
            UnionSet(u, v, parent, rank);
            mstEdges.push_back(edge);
            totalWeight += edge.weight;
        }
    }

    cout<<"Edge     Wegiht"<<endl;
    for(Edge edge: mstEdges)
    {
        cout<<edge.u<<" - "<<edge.v<<"      "<<edge.weight<<endl;
    }
    cout<<totalWeight<<endl;

}

int main()
{
    vector<vector<int>> graph = {
    // provided dense graph
    {0,3,0, 4,4,0,0, 0,0,0},
    {3,0,10, 0,2,3,0, 0,0,0},
    {0,10,0, 0,0,6,1, 0,0,0},

    {4,0,0, 0,5,0,0, 6,0,0},
    {4,2,0, 5,0,11,0, 2,1,0},
    {0,3,6, 0,11,0,2, 0,3,11},
    {0,0,1, 0,0,2,0, 0,0,8},

    {0,0,0, 6,2,0,0, 0,4,0},
    {0,0,0, 0,1,3,0, 4,0,7},
    {0,0,0, 0,0,11,8, 0,7,0},
    };

    auto Start = chrono::high_resolution_clock::now();
    kruskal(graph.size(), graph);
    auto End = chrono::high_resolution_clock::now();
    chrono::duration<double, milli> duration = End - Start;

    cout<<duration.count()<<" ms"<<endl;

    return 0;
}
```

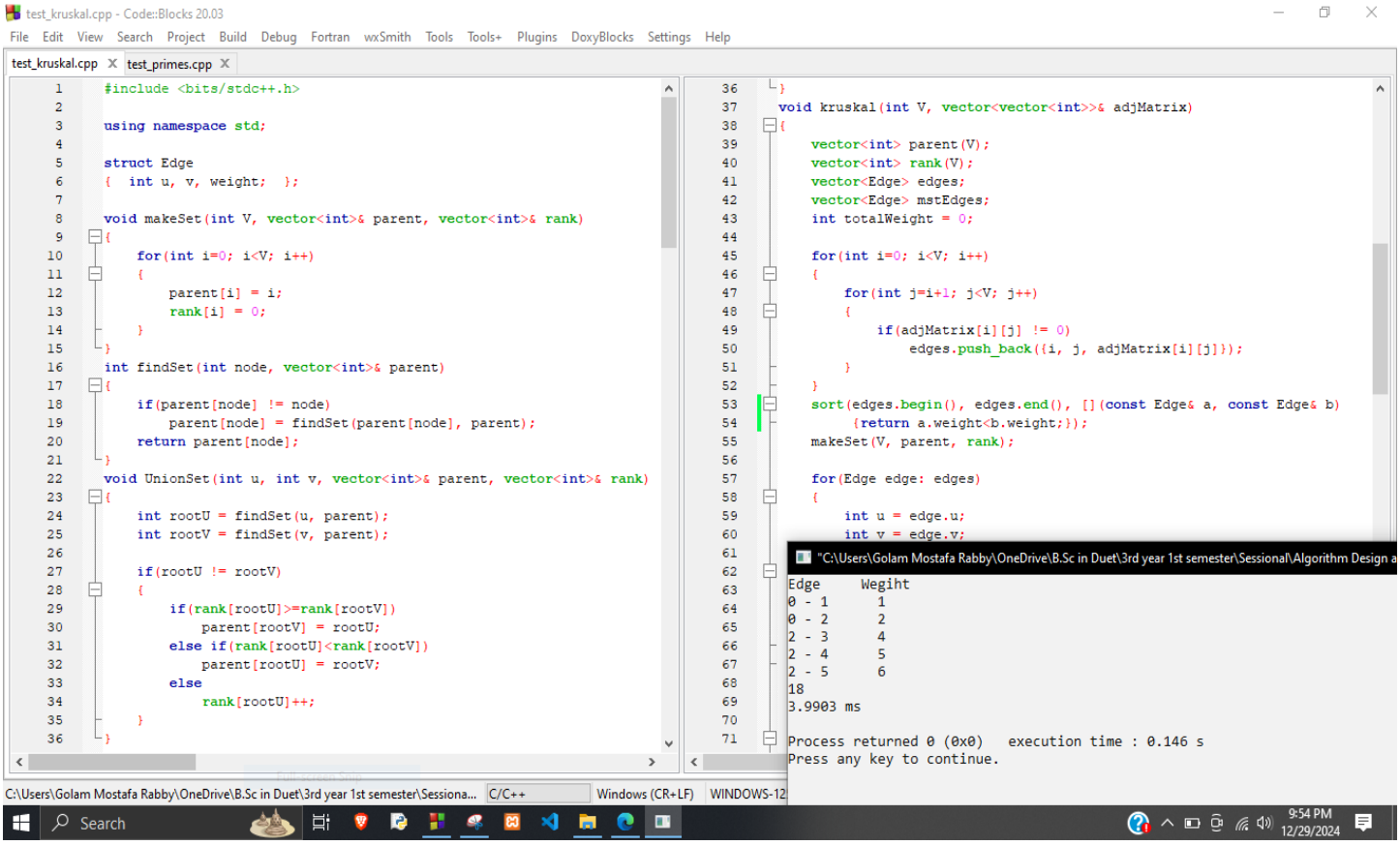
Prim's Algorithm





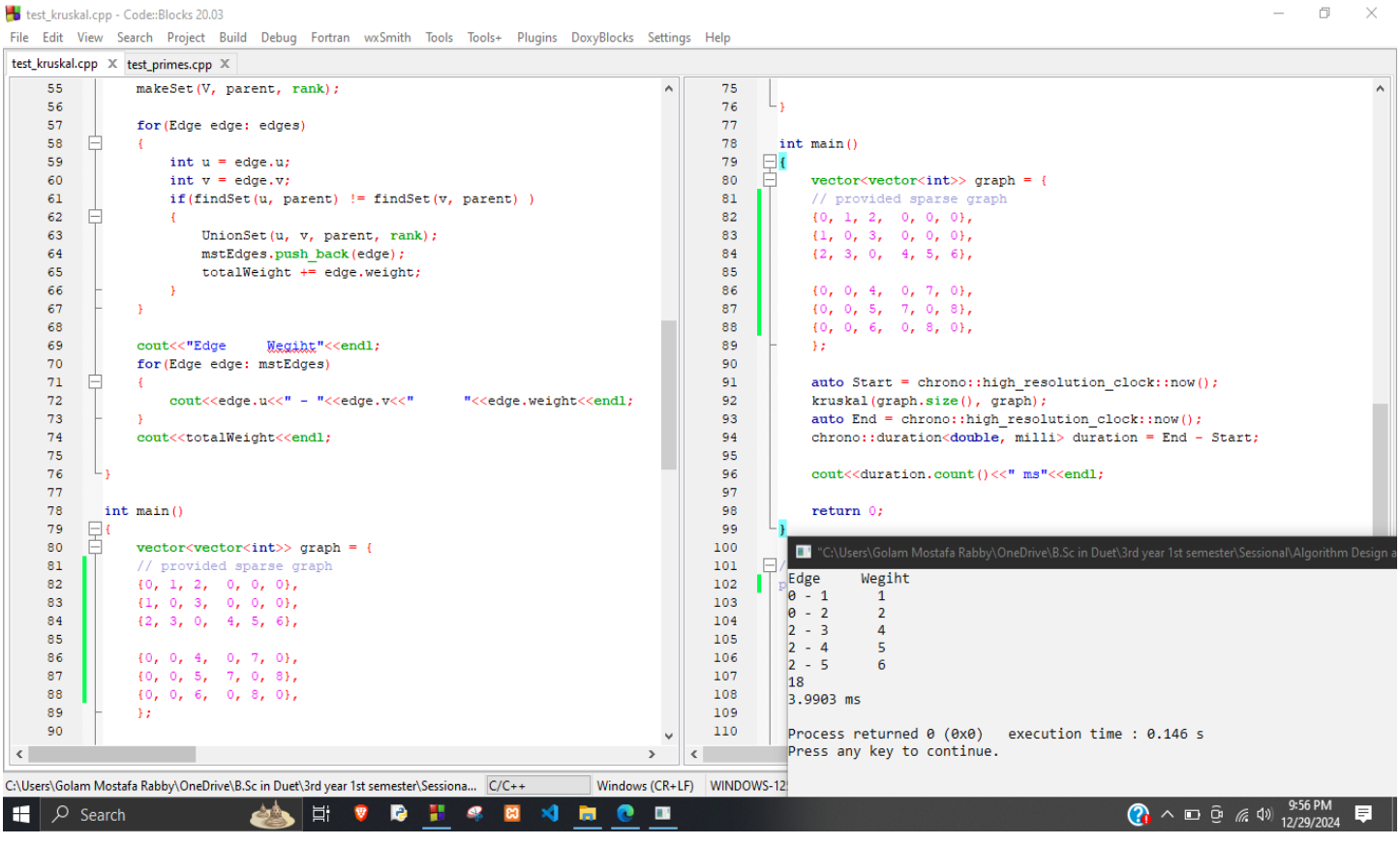
Sparse Graph:

Kruskal's Algorithm



```
#include <bits/stdc++.h>

using namespace std;

struct Edge
{  int u, v, weight;  };

void makeSet(int V, vector<int>& parent, vector<int>& rank)
{
    for(int i=0; i<V; i++)
    {
        parent[i] = i;
        rank[i] = 0;
    }
}
int findSet(int node, vector<int>& parent)
{
    if(parent[node] != node)
        parent[node] = findSet(parent[node], parent);
    return parent[node];
}
void UnionSet(int u, int v, vector<int>& parent, vector<int>& rank)
{
    int rootU = findSet(u, parent);
    int rootV = findSet(v, parent);

    if(rootU != rootV)
    {
        if(rank[rootU]>=rank[rootV])
            parent[rootV] = rootU;
        else if(rank[rootU]<rank[rootV])
            parent[rootU] = rootV;
        else
            rank[rootU]++;
    }
}
void kruskal(int V, vector<vector<int>>& adjMatrix)
{
    vector<int> parent(V);
    vector<int> rank(V);
    vector<Edge> edges;
    vector<Edge> mstEdges;
    int totalWeight = 0;

    for(int i=0; i<V; i++)
    {
        for(int j=i+1; j<V; j++)
        {
            if(adjMatrix[i][j] != 0)
                edges.push_back({i, j, adjMatrix[i][j]});
        }
    }
    sort(edges.begin(), edges.end(), [](const Edge& a, const Edge& b)
         {return a.weight<b.weight;});
    makeSet(V, parent, rank);

    for(Edge edge: edges)
    {
        int u = edge.u;
        int v = edge.v;
```

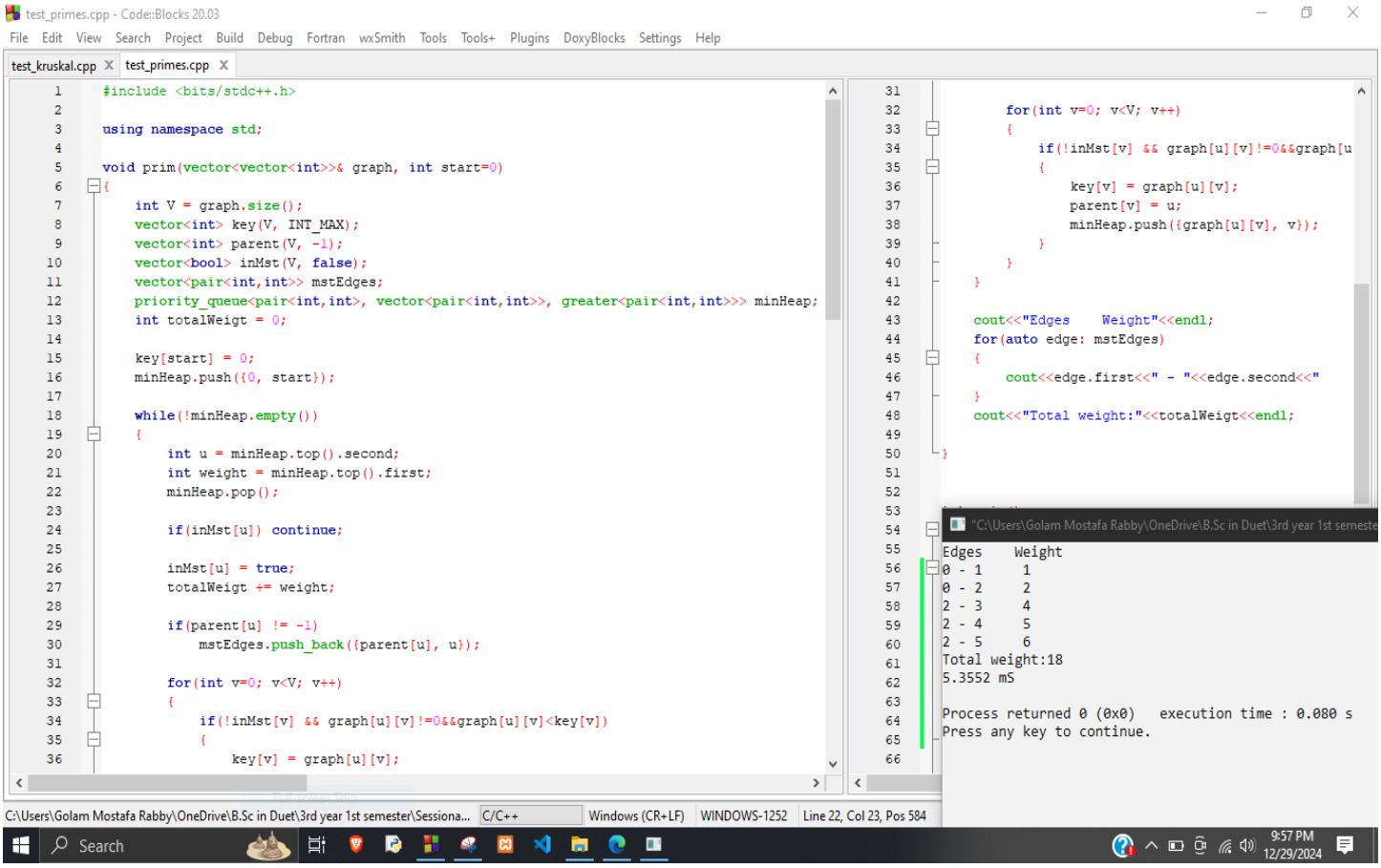
```
"C:\Users\Golam Mostafa Rabby\OneDrive\B.Sc in Duet\3rd year 1st semester\Sessional\Algorithm Design a
Edge     Wegiht
0 - 1      1
0 - 2      2
2 - 3      4
2 - 4      5
2 - 5      6
18
3.9903 ms

Process returned 0 (0x0)   execution time : 0.146 s
Press any key to continue.
```



```
    makeSet(V, parent, rank);

    for(Edge edge: edges)
    {
        int u = edge.u;
        int v = edge.v;
        if(findSet(u, parent) != findSet(v, parent) )
        {
            UnionSet(u, v, parent, rank);
            mstEdges.push_back(edge);
            totalWeight += edge.weight;
        }
    }

    cout<<"Edge     Wegiht"<<endl;
    for(Edge edge: mstEdges)
    {
        cout<<edge.u<<" - "<<edge.v<<"      "<<edge.weight<<endl;
    }
    cout<<totalWeight<<endl;

}

int main()
{
    vector<vector<int>> graph = {
    // provided sparse graph
    {0, 1, 2,  0, 0, 0},
    {1, 0, 3,  0, 0, 0},
    {2, 3, 0,  4, 5, 6},

    {0, 0, 4,  0, 7, 0},
    {0, 0, 5,  7, 0, 8},
    {0, 0, 6,  0, 8, 0},
    };

    auto Start = chrono::high_resolution_clock::now();
    kruskal(graph.size(), graph);
    auto End = chrono::high_resolution_clock::now();
    chrono::duration<double, milli> duration = End - Start;

    cout<<duration.count()<<" ms"<<endl;

    return 0;
}
```

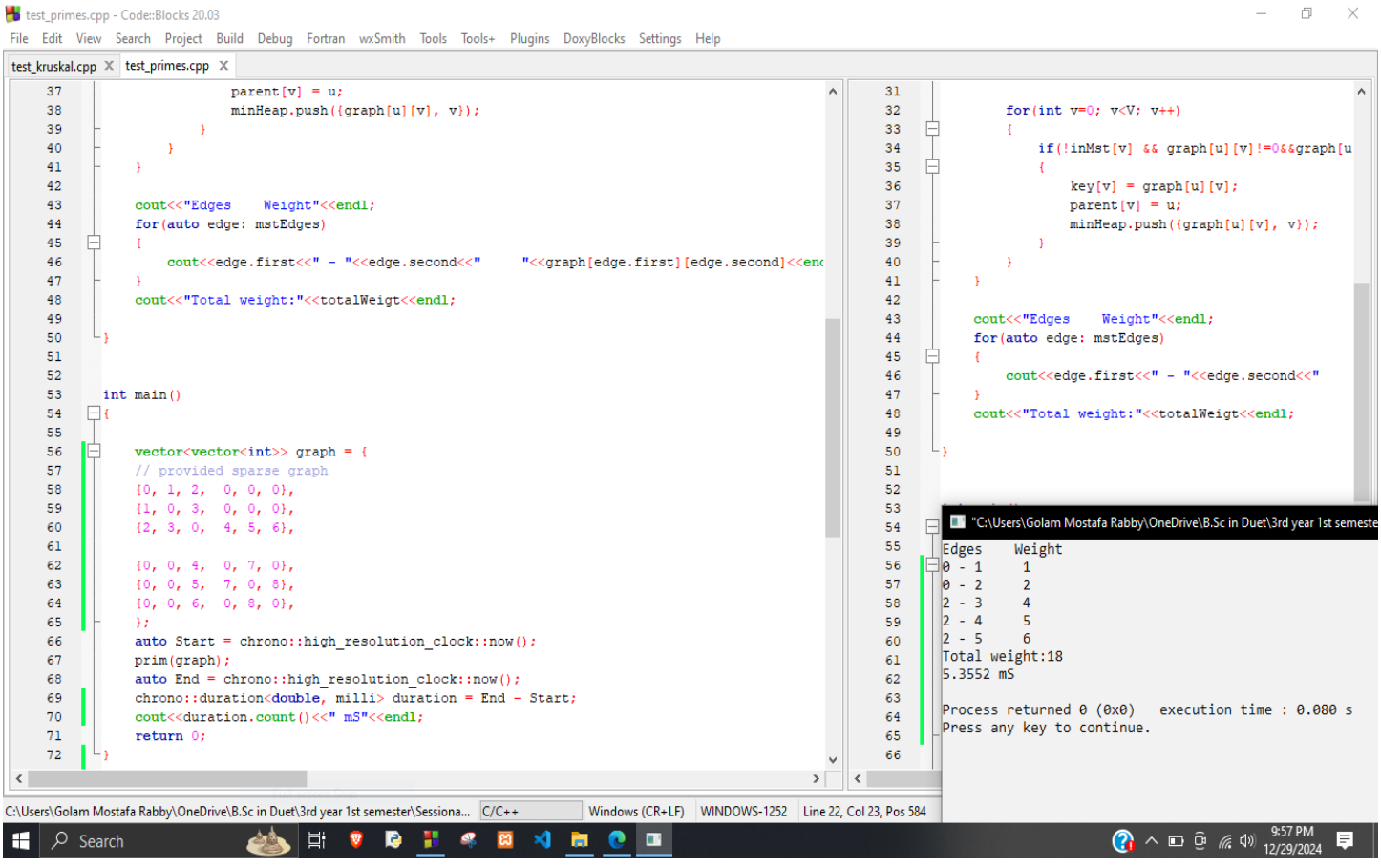
```
"C:\Users\Golam Mostafa Rabby\OneDrive\B.Sc in Duet\3rd year 1st semester\Sessional\Algorithm Design a
Edge     Wegiht
0 - 1      1
0 - 2      2
2 - 3      4
2 - 4      5
2 - 5      6
18
3.9903 ms

Process returned 0 (0x0)   execution time : 0.146 s
Press any key to continue.
```

Prim's Algorithm



```
#include <bits/stdc++.h>

using namespace std;

void prim(vector<vector<int>>& graph, int start=0)
{
    int V = graph.size();
    vector<int> key(V, INT_MAX);
    vector<int> parent(V, -1);
    vector<bool> inMst(V, false);
    vector<pair<int,int>> mstEdges;
    priority_queue<pair<int,int>, vector<pair<int,int>>, greater<pair<int,int>>> minHeap;
    int totalWeigt = 0;

    key[start] = 0;
    minHeap.push({0, start});

    while(!minHeap.empty())
    {
        int u = minHeap.top().second;
        int weight = minHeap.top().first;
        minHeap.pop();

        if(inMst[u]) continue;

        inMst[u] = true;
        totalWeigt += weight;

        if(parent[u] != -1)
            mstEdges.push_back({parent[u], u});

        for(int v=0; v<V; v++)
        {
            if(!inMst[v] && graph[u][v]!=0&&graph[u][v]<key[v])
            {
                key[v] = graph[u][v];
                parent[v] = u;
                minHeap.push({graph[u][v], v});
            }
        }
    }

    cout<<"Edges    Weight"<<endl;
    for(auto edge: mstEdges)
    {
        cout<<edge.first<<" - "<<edge.second<<"
    }
    cout<<"Total weight:"<<totalWeigt<<endl;

}
```

```
Edges    Weight
0 - 1     1
0 - 2     2
2 - 3     4
2 - 4     5
2 - 5     6
Total weight:18
5.3552 mS

Process returned 0 (0x0)   execution time : 0.080 s
Press any key to continue.
```



```
#include <bits/stdc++.h>

using namespace std;

void prim(vector<vector<int>>& graph, int start=0)
{
    int V = graph.size();
    vector<int> key(V, INT_MAX);
    vector<int> parent(V, -1);
    vector<bool> inMst(V, false);
    vector<pair<int,int>> mstEdges;
    priority_queue<pair<int,int>, vector<pair<int,int>>, greater<pair<int,int>>> minHeap;
    int totalWeigt = 0;

    key[start] = 0;
    minHeap.push({0, start});

    while(!minHeap.empty())
    {
        int u = minHeap.top().second;
        int weight = minHeap.top().first;
        minHeap.pop();

        if(inMst[u]) continue;

        inMst[u] = true;
        totalWeigt += weight;

        if(parent[u] != -1)
            mstEdges.push_back({parent[u], u});

        for(int v=0; v<V; v++)
        {
            if(!inMst[v] && graph[u][v]!=0&&graph[u][v]<key[v])
            {
                key[v] = graph[u][v];
                parent[v] = u;
                minHeap.push({graph[u][v], v});
            }
        }
    }

    cout<<"Edges    Weight"<<endl;
    for(auto edge: mstEdges)
    {
        cout<<edge.first<<" - "<<edge.second<<"      "<<graph[edge.first][edge.second]<<endl;
    }
    cout<<"Total weight:"<<totalWeigt<<endl;

}


int main()
{

    vector<vector<int>> graph = {
    // provided sparse graph
    {0, 1, 2,  0, 0, 0},
    {1, 0, 3,  0, 0, 0},
    {2, 3, 0,  4, 5, 6},

    {0, 0, 4,  0, 7, 0},
    {0, 0, 5,  7, 0, 8},
    {0, 0, 6,  0, 8, 0},
    };
    auto Start = chrono::high_resolution_clock::now();
    prim(graph);
    auto End = chrono::high_resolution_clock::now();
    chrono::duration<double, milli> duration = End - Start;
    cout<<duration.count()<<" mS"<<endl;
    return 0;
}
```

```
Edges    Weight
0 - 1     1
0 - 2     2
2 - 3     4
2 - 4     5
2 - 5     6
Total weight:18
5.3552 mS

Process returned 0 (0x0)   execution time : 0.080 s
Press any key to continue.
```