*// O(Height \* Width)*

**private** **void** ConstructGraph()

{

AdjacencyList = **new** List<KeyValuePair<int, double>>[Height \* Width];

*// O(Height \* Width)*

**for** (int i = 0; i < Height \* Width; i++)

{

Fixed[i] = **false**;

AdjacencyList[i] = **new** List<KeyValuePair<int, double>>();

}

TestingHandling.SetStartTime(DateTime.Now);

*// O(Height \* Width)*

**for** (int i = 0; i < Height; i++)

{

**for** (int j = 0; j < Width; j++)

{

Vector2D Energy = ImageOperations.CalculatePixelEnergies(i, j, ImageMatrix);

int Idx = GetIndex(i, j); *// O(1)*

int RightIdx = GetIndex(i, j + 1), BottomIdx = GetIndex(i + 1, j);

*// Check if right node exists*

**if** (j + 1 < Width)

{

*// Calculating weight = 1 / G*

double w = Energy.X;

**if** (w == 0)

w = 1e16;

**else**

w = 1 / w;

*// Connect undirected edge between node and right node*

AdjacencyList[Idx].Add(**new** KeyValuePair<int, double>(RightIdx, w));

AdjacencyList[RightIdx].Add(**new** KeyValuePair<int, double>(Idx, w));

}

*// Check if bottom node exists*

**if** (i + 1 < Height)

{

*// Calculating weight = 1 / G*

double w = Energy.Y;

**if** (w == 0)

w = 1e16;

**else**

w = 1 / w;

*// Connect undirected edge between node and bottom node*

AdjacencyList[Idx].Add(**new** KeyValuePair<int, double>(BottomIdx, w));

AdjacencyList[BottomIdx].Add(**new** KeyValuePair<int, double>(Idx, w));

}

}

}

*// Write the constructed graph into text file*

**if** (TestingHandling.GetImageFilePath().IndexOf("Complete") != -1)

TestingHandling.PrintConstructedGraphCompleteTest(**this**);

**else**

TestingHandling.PrintConstructedGraphSampleTest(**this**);

Reset();

}

*// O(WindowSize^2 \* log(WindowSize^2))*

**private** **void** RunDijkstra()

{

id++; *// Counter to reset the values of ShortestPath and ParentNode*

PriorityQueue pq = **new** PriorityQueue(**false**);

pq.Enqueue(**new** KeyValuePair<double,int>(0, CurAnchor)); *// O(log(WindowSize^2))*

ShortestPath[CurAnchor] = 0;

ParentNode[CurAnchor] = -1;

vis[CurAnchor] = id;

**while** (pq.Count > 0)

{

var Node = pq.Dequeue(); *// O(log(WindowSize^2))*

double CurDist = Node.Key;

int NodeIdx = Node.Value;

*// Check if the node is out of the WindowSize*

**if** (NotValid(NodeIdx))

**continue**;

**foreach** (var Child **in** AdjacencyList[NodeIdx]) *// O(log(WindowSize^2))*

{

*// Check if it is the first time to visit the node or*

*// the current path is shorter than the previous path*

**if** (vis[Child.Key] != id || ShortestPath[Child.Key] > CurDist + Child.Value)

{

vis[Child.Key] = id;

ParentNode[Child.Key] = NodeIdx;

ShortestPath[Child.Key] = CurDist + Child.Value;

*// O(log(WindowSize^2))*

pq.Enqueue(**new** KeyValuePair<double, int>(CurDist + Child.Value, Child.Key));

}

}

}

}