Graph with given data

1:The graph that we already used in our code is an eight-way connected graph.If we try to make such a graph , with following data ,

(7,0,0)	(0,1,0)	(0,0,1)
(2,0,0)	(0,8,0)	(0,0,5)
(1,0,0)	(0,1,0)	(0,0,8)

(0,0,0)	(0,0,0)	(0,0,0)
(0,0,0)	(0,0,0)	(0,0,0)
(0,0,0)	(0,0,0)	(0,0,0)

then we get a graph as shown below,three numbers in a cell represent the color of that pixel and we use this information to compute edge weights. For the distance computation between two points we use a formula:

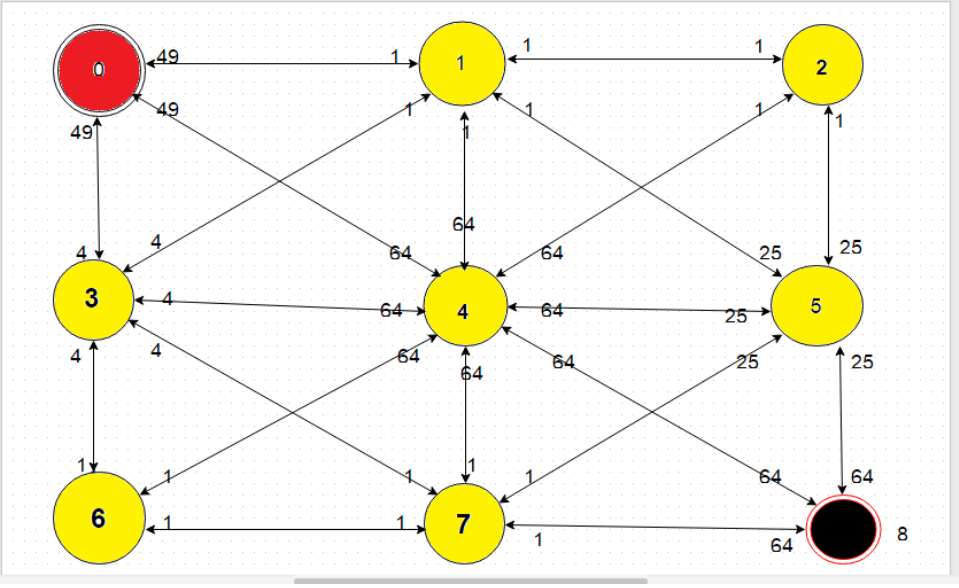
X = x1 - x2 where x1 represents the red color of first image and x2 is the red color of second image.

Y = y1 - y2 where y1 represents the green color of first image and y2 is the green color of second image.

Z = z1- z2 where z1 represents the blue color of first image and z2 is the blue color of second image .

The final result is returned as , result = (X*X+Y*Y+Z*Z)

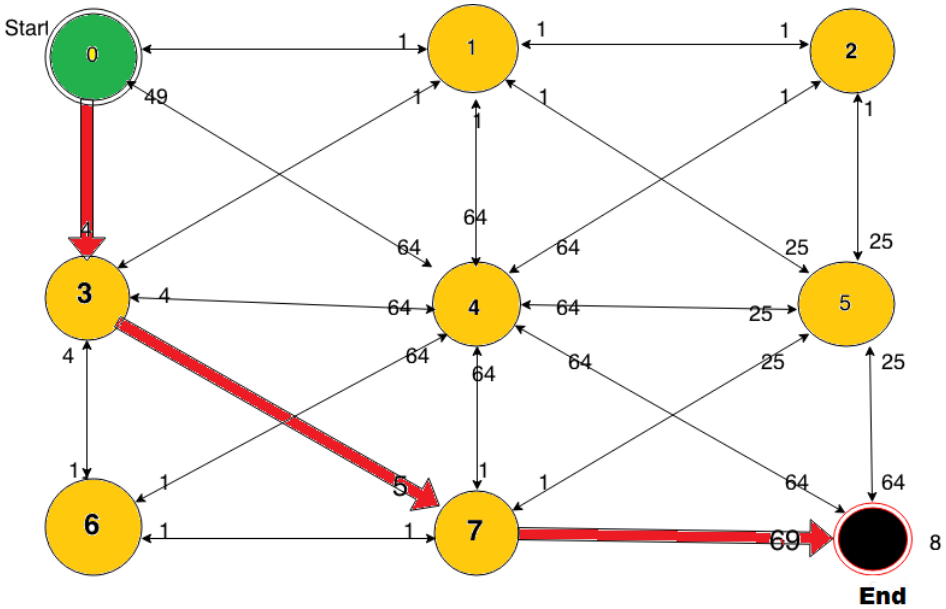
Here we have two different tables so if we try to compare the source with the destination(2nd pixel) then we would not get a better result because we are just ignoring second table. In order to get better result only the destinations(2nd pixel) of the both tables will be compared. For example to go from (7,0,0) to (0,1,0), we have to compute the difference between (0,1,0) of first table with (0,0,0) of second table, difference between (2,0,0) and (0,0,0) of second table and finally the diagonal cell (0,8,0) and (0,0,0). The we take the pixel which has smallest difference as compared to the second image.



The distance from one pixel to other is given with each edge. As the graph is 8-way connected graph so at both sides of edges the weight is given.For example the weight from 0 to 1 is 1 and weight from 1 to 0 is 49.

Shortest-Path

If we want to find the shortest path then as we can see that we have already all weights of all the edges so it would not be difficult to compute shortest-path.Shortest-path that we compute makes uses of Dijkstra shortest-path algorithm . So for this graph we will get shortest path as shown below:



The exact data is here with given weight and vertices.

From (Vertex)	To (Vertex)	Weight(Edge)
0	3	4
3	7	4+1 = 5
7	8	5+64 = 69

Use of other distance formulas:

2:If we use a different formula as shown here:

Formula : Sqrt((rx − ry)^2 + (gx − gy)^2), then we are just ignoring the blue color. We can still find the shortest-path but it,s possible it would not be an optimal solution. If a given pixel has only blue color then the difference between two pixels would not be zero if the previous pixel does not have same color but if we use a formula like this, where we ignore blue color, then we will get difference zero which is not right. So in this way we will mark some wrong pixels. As we see we just created a graph first with edge weights and then we used Dijkstra algorithm ,so in this way the new distance formula would not effect the algorithm,s running time.We can expect a very tiny effect while computing the difference between pixels but it,s so small that it,s ignorable.

Memory Issues

3: If we try to build a graph like this with a very a great table then we have to add as many edges.

Data-structure that will take more memory is Edge data-structure. In 8 way-connected graph the minimum number of edges are three and maximum 8 ,so called (degree). The average number of edges are 5. So a graph which is build of (N) vertices , will take 5*(N) edges to be a 8 way-connected graph.The space complexity will be ~(5N) of edge data-structure.

Seam Behaviour

4: Seam follows a path and paths are created of difference edges.If we want to avoid that seam should not run diagonal or back then we have to use a directed graph.In such a graph we can go from a vertex to other but not in opposite direction. In such way we can say that seam will not run back.(In other words we don,t need to add edges where we want to avoid the seam).

Time Complexity

5:While computing the time complexity of the program when an image is only one pixel high ,so every funtion will take contant time.First in the seam funtion two array initialization take time ~(n+n). A graph is created of V vertices and E edges so we have 1 vertex and 0 edge. In this case the time complexity will be again V. Since there is no edge then we can just ignore Dijkstra shortest-path algorithm. If we sum up all the cost where n is one then we get : ((1+1)+1) where (1+1) represents the array initialization and the last 1 is graph initialization.

Stitch: Because the seam contains only one position so the loop in stitch method will also be executed once.(n times, where n =1)

FloodFill:Floodfill with one pixel will also be executed (n+(n+n) where n == 1) where the first n represents the array initialization and second n is from floodFillIterativeRight and floodFillIterativeLeft.

So the total cost will be (1+(1+1)) ==> 3.

Longest-Acyclic Path

6: If our program finds the longest path instead of shortest then the result of the stitched picture will be very strange because when finding the shortest path we actually mark the positions which are very similar but if we find longest path then we are doing nothing else but trying to mark the positions, at which the distance between two pixels of two images is very great.Then if we stitch two images at that position then the difference will be very great.For example if we try to stitch those two given sea pictures then it,s possible that sea-view from first image will be stitched at mountains from the second image because there is great difference between the color of those images.