**Problem: Finding shortest path to a destination**

**Given:**

* 2d Array of slope color coded: Minimum – black, Maximum – white.
* Maximum slope allowed: 15 degrees.
* Start index (a, b). End index (c, d).
* Can **not** travel diagonally.

**Solution: (Hopefully)**

* Start by converting color coded array to array filled with distances from end index.
  + Position with slope >= 15 degrees has infinite distance
  + Distance = sqrt( (c-x)^2+(d-y)^2 ) for (x, y) of selected position
  + Takes O(n^2) Time.
* Algorithm:
  + Set S contains all nodes in T.
  + Store external nodes in priority queue with distance (primary) and node object as reference
  + Tree T contains all the nodes travelled to. Let internal nodes be paths travelled and external nodes be paths yet to travel.
  + ‘Add’ node v to T which is the external node with the least distance (from priority queue) and if it is not already in T (from set).
    - Each added node keeps track of ‘steps’ which are the distance traveled to get there.
    - If external node appears in two places – travel with the node that has less steps.
    - Add all edges from the ‘Added’ node to T except the node just travelled from.
    - Add v to the set S.