Homework 5 ***Goldy***

PART1:

Method:

1. Divide arbitrary simple polygon B into convex pieces Ci.
2. Find the minkowski sum of all the pieces Ci with the image of the convex polygon A.
   Mi = MinkowskiSum(Ci, -A)   i = 1,2,…nt (number of convex pieces)
3. Get the line segment OS, starting from (0,0) and ending at the vector value given as input.
4. Get the intersection (t values) of OS with polygon Mi treating OS as a line.
5. Merge the interval of intersection
   a. Remove any common intersection interval
   b. Take parameter value (t) only between 0 and 1
6. Get the length of intersection from the parametric interval(s).

A) Getting Convex pieces:
   a. Get a reflex vertex 'vo' (invalid vertex with internal angle greater than 180)
   b. Join 'vo' to a different vertex 'v1' which gives a valid decomposition of the simple polygon into two.
   c. 'v1' is found by moving to adjacent vertices on one side till you get a division where optimally 'vo' is no longer reflex. To optimize the number of divisions 'v1' is chosen such that it is closest to the angle bisector of 'vo'. If this doesn't exist it is just joined to a vertex which gives a valid subdivision.
   d. Recursively divide these two polygons till there are no reflex vertices left.

B) Minkowski Sum:
   a. Minkowski sum of two convex pieces, Mi, is a linear time operation.
   b. Arrange the edges of reflection of A (-A) by the increasing order of angle of its edges
   c. Do the same with Ci.
   d. Merge the two sorted lists.
   e. One by one walk on the items in the list, adding the vertices. Vertices being added are flipped one by one when its crossed over in the list.

C) Line Intersection:
   a. Represent the line OS parametrically with 't' as the parameter which is 0 at O an 1 at S.
   b. Intersect the line with with the convex polygon Mi and save parameters of intersection 'tmin' and 'tmax'.

c. Special cases are handled automatically as I am only saving min and max.
d. Only keep the interval which lies between 0 and 1.

D) <u>Merging of Interval :</u>
   a. Get all the intervals of intersection
   b. Sort the intervals by ‘tmin’.
   c. Add the total amount of interval.
   d. Merge the intervals so that overlapping regions are not added twice

<u>Complexity :</u>

1. Dividing into convex pieces: O(m^2)
   a. Number of reflex vertices in a polygon B – O(m)
   b. For each reflex vertex I am going over other vertices to find vertices which gives be valid division. O(m)
   c. So complexity of getting convex pieces is O(m^2)
2. Minkowski Sum: O(mn)
   a. Complexity of Minkowski sum by rotating caliper is linear time in total number of edges of the two polygons
   b. In worst case scenario (maximum edges) all the convex pieces are triangles. So for polygon B we have (m-2) triangles.
   c. Complexity of Minkowski sum of A and a triangle– O(n+3)
   d. For all Minkowski pieces- O ((m-2)*(n+3)) = O(mn)
3. Line Intersection: O(mn)
   a. Intersecting line with all the edges:
   b. In worst case (Minkowski sum of A and a triangle) number of edges are : (n+3)
   c. Total number of edges for (m-2) triangles : (m-2)(n+3)
   d. Intersecting two lines is constant time.
   e. So complexity of finding all the intersection: O(mn)
4. Merging intervals: mlog(m)
   a. One convex piece has one interval.
   b. (m-2) convex piece has (m-2) interval.
   c. Sorting: mlog(m)
   d. Walking over all to get rid of common region – O(m)

So complexity of the whole algorithm - O(m^2 + mn + mlog(m)) = O(m^2 + mn)= O(m^2)

Part2

1. In worst case we will have to find the intersection of all the line segments with the all the convex pieces (Minkowski Sum).
2. But since we have to find out if they intersect even once, so we can optimize our algorithm for average situations
   a. Get the bounding box of all convex pieces
   b. Intersect line segments with the bounding box. If there is no intersection means the line segment do not intersect
   c. But in case of intersection we will have to go and intersect with all the edges of the convex pieces.
3. Further improvement can be done by not intersecting the lines sequentially but by intersecting the middle line segment of all the line segments recursively.

Part3

1. Get the visibility graph of all the convex Minkowski polygons.
   a. A naïve algorithm for visibility graph is for every vertex ‘a’ and ‘b’ check if line segment ‘ab’ intersect with any edge. If it doesn’t it form an edge in the visibility graph. Edges of the polygons are also tested.
   b. Starting and the ending points are treated as polygon with one vertex each.
   c. A graph is formed from those edges with vertices as the nodes and edges as the edge of the graph. Length of the graph forms the weight of the edges.
2. After getting the graph use Dijkstra’s Algorithm to find the shortest path between the stating node and the ending node.
   a. Begin from the start node and do a depth first search
   b. At each step keep track of the lowest weight path so far.
   c. Use predecessor path information to create best path for a node