Ahmet Faruk Ulutaş - 21803717 – CS478 – HW4

1-

Precondition: Polygons must be represented as a circular neighbouring edge list to use the linear polygon intersection technique. [(v1, v2), (v2, v3), (v3, v1)] is an example.

Polygons must be oriented counterclockwise such that the insides of polygons are on the left of the edges in order to use the linear polygon intersection procedure. Change the orientation of any polygon if it is clockwise. It will take O(N + M) time to complete.

Pseudocode:

```
IntersectionOfConvexPolygons(P, Q)
begin
pi, qi, iterator, R <- 0
while iterator < (P.edgeCount + Q.edgeCount) * 3
begin
  W = (edge(pi-1, pi) ∩ edge(qj-1, qj))
  if w is not empty
  begin
        append W to R
  endif
  S <- next edge based on advance rules
  if S is on inner chain
  begin
        append S.destination to R
        iterator <-  iterator + 1
  endif
endwhile
```

```
if R is empty
begin
   point <- P.firstEdge.from
   foreach edge in Q, if edge.leftTest(point) is left
   begin
           return P
   end for
   point <- Q.firstEdge.from
   foreach edge in P, if edge.leftTest(point) is left
   begin
           return Q
   end for
   return []
   end if
end
```

2-

The equation of each line segment is extracted and recorded. Thus, by rotating both line segment sets, it can be understood whether the lines intersect using their equations and slopes in O(n^2) time in brute force logic. If it intersects, it turns right. If it does not intersect, it returns false. Thus, the preprocessing takes O(n+m) while the testing part takes O(n^2).

3-

Add all disks. Keep the total number of discs. Calculate the start and endpoints of all disks. Return all values starting from 0 up to the given time t. If multiple disks have the same x and y values, delete the disks. Update the total number of disks and disk trees. When the value of T is reached, return the number of disks. This implementation can be done efficiently using trees. Thus, O(N log N) is obtained. For N all disks, to find log(N) intersection points in the tree.

4-

Get the minimum x, y and maximum x, y points in the path. From the maximum y point to the minimum y point, suddenly select y points of the same magnitude. As you go from the minimum x value of the points to the maximum x value for each pair of points (sum the y values of the selected point minus the y of the current x point). If this value is higher than the previous water volume value, change the water volume value to this. If not, continue the loop. If the part before the minimum x or after the maximum x is smaller than the y values of the selected points and is in the path (i.e. its y does not go to infinity), apply the same operation in that part and add it above the volume value. Return the maximum volume after the whole scanning process is finished.

In summary, find each bucket, find the lower buckets that the buckets can go to. Do not go out of the path while doing these operations. Find the sum of the y-values of each bucket set and return the largest one. If there is a path to the lower bucket, do not add y values along that path (Assuming there is no bucket along the way until the lower bucket so it flows not stores).