CSL 852, Computational Geometry: Problem Set 3

1. **Motion planning** Given a scene with rectangular obstacles, and a disk of radius r, we would like to find out a feasible path from an initial position to a final position (avoiding the obstacles).

   Design and implement an efficient algorithm for this. You can use existing code for Voronoi Diagrams and integrate with your algorithm.

   Possible approaches:
   1. You can grow the obstacles using Minkowski's sum and compute the maximal connected regions.

   2. Alternatively, you can approximate the obstacles by points on the boundary that are less than $2r$ distance - denote this by S'. You can then compute the Voronoi Diagram $Vor(S')$ and only retain those edges in $Vor(S')$ that at least r from the closest obstacle point. Now work with this sub-graph. This has the disadvantage that the input size may blow up significantly but you can make reasonable assumptions.

2. Given a set S of n points in the plane, design an efficient algorithm to find the *largest empty circle*, that doesn't contain any point of S and the center of the disk is within the $CH(S)$.

3. Given a set S of n points in the plane, design an efficient algorithm to find the *smallest enclosing disk*, that contains all point of S.
   Hint: Use Randomized incremental construction and analyse carefully.

4. Given a set S of n points in the plane, design an efficient algorithm to find a disk D that encloses k points (any of the k out of n points) and whose radius is no more than twice that of $D_o(n,k)$, the smallest disk that contains k points. When k is $\Omega(n)$, show that your algorithm runs in $O(n)$ time.
   Hint: Prove the property that the smallest disk centered at $q \in D_o(n,k)$ that contains k points, will be no more than twice the radius of D_o.

5. For the 2d range search tree, we would like to modify the construction in the following way to save space.
   Instead of storing the points in every level, we will store the points in every t-th level, thereby reducing the storage to $O(n/t \log n)$. Consequently, we cannot do the search for the y interval in every node of every level but only in every t-th node of the search path (the canonical subintervals of the x interval). How would you modify the range search reporting and obtain an exact expression for the trade-off between search time and space in terms of t.