1A

Voronoi diagram is a partition of a plane into regions based on the closest proximity to a set of points. The set of segment lines should be equidistant from the closest points. The regions are a set of polygons that every point within a polygon is closer to the associated point than to any other point. The edges of these regions can be used as the navigation paths.

It is good for planning safe paths to stay away from obstacles when navigating through complex environments. And it is also convenient because we get the navigation paths directly.

1B

When we have already a map of the environment, it is possible to compute the Voronoi diagram directly. But I don’t think the Fortune's algorithm will work because it is designed to generate the Voronoi diagram for a set of input points. But in the case of robot navigation, our obstacles are polygons. Fortune's algorithm can’t take regions as input. There are some other methods to convert the polygonal regions into a set of points and then compute the Voronoi diagram by Fortune's algorithm or some other algorithms. Once we have the Voronoi diagram, we can use it to plan a path that avoids obstacles.

When we do not have a map of the environment, the Fortune's algorithm can not help us to obtain the map. We usually need simultaneous localization and mapping (SLAM) algorithms to create the map of the environment.

In summary, Fortune's algorithm is still a practical tool for a robotics researcher.

1C

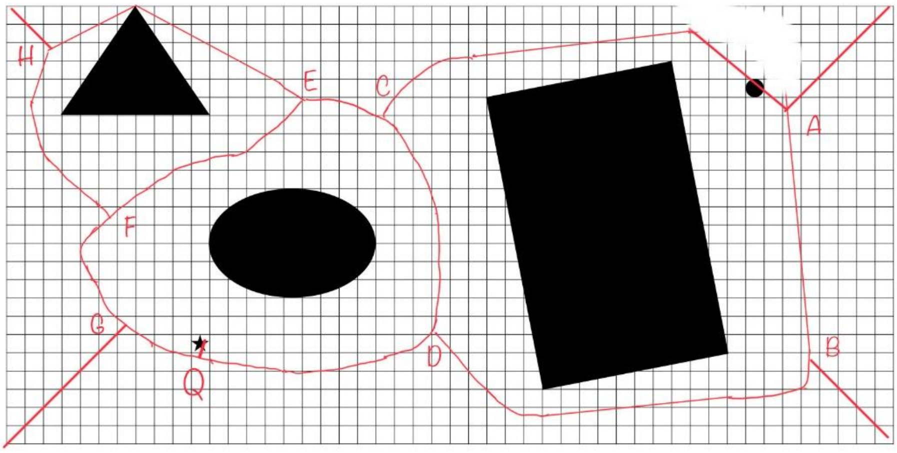
In this case, we need simultaneous localization and mapping (SLAM) algorithms.

The robot has sensors which can measure the distance such as camera, lidar or radar. The robot starts by exploring its surroundings. These sensors collect data about the environment, including landmarks, obstacles, and other features. The robot uses this data to gradually construct a map of the environment. It is called “mapping”.

While mapping, the robot also needs to simultaneously determine its own position within the map by comparing sensor data to the map it is building. It is called “localization”. The robot also need odometry sensors to record the movement information. Some state estimator like Extended Kalman Filter (EKF) and Particle Filter (PF) can be used to help estimating the robot's pose (position and orientation) relative to the map constructed.

After collecting enough information and building a map of the environment, we can construct a Voronoi graph based on the map and some other methods to convert the polygonal regions into a set of points and then compute the Voronoi diagram by Fortune's algorithm.

1D



2

We apply the A* algorithm to the Voronoi graph we obtained. Starting from the start point, we calculate the total estimated cost F to reach the goal point by passing through a particular connected node. The total estimated cost F is the sum of two parts, G and H. G is the exact cost of the path from the starting point to the particular node, H is the heuristic estimated cost from this node to the goal which is an optimistic estimate ignoring the obstacles. And each time we choose the node which gives the lowest total cost F.

In this problem the robot can make diagonal moves, and each touched cell counts as a distance 1 step.

Starting from the start position, it is on the line of Voronoi graph. There are two connected nodes, A and C. We examine the F value of them.

For A, G = 2, H = 31, F = 33

For C, G = 22, H = 12, F = 34

So we choose node A for now. Then we have two options, B and C.

For B, G = 14, H = 34, F = 48

For E, G = 22+4=26, H = 14, F = 40

For D, G = 22+12=34, H = 12, F = 46

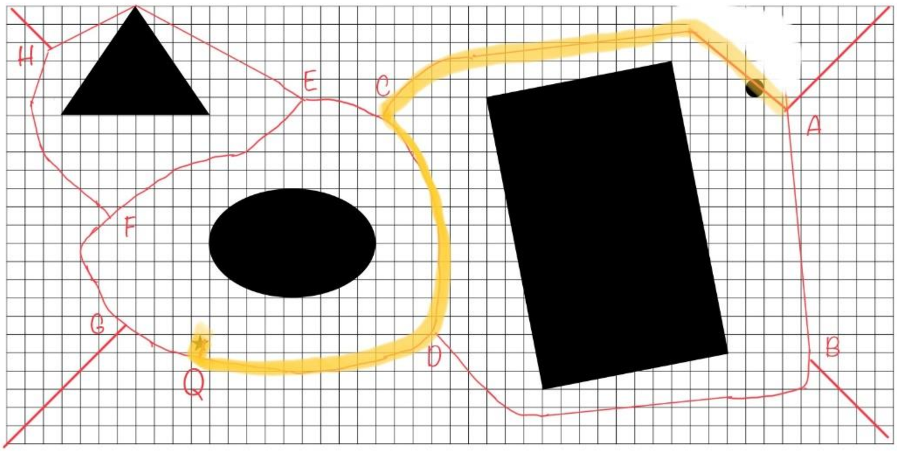
We choose E for now, and examine node F, G = 11+26=37, H = 7, F= 44

Then go to node G, G = 7 +37=44, H=4, F = 48

F(G)>F(D), so we choose node D.

Because the star point is not on the edge, we need to find a closest point Q on the edge G-D to the star point. The we just go to point Q.

The final path is from the start point to node C and then node D then to Q and go to the star point, shown in the yellow line below.



H
E
C
A
F
G
Q
D
B