Homework 2

Handed out Mon, April 30. Due: **Thu, May 10 at 11pm**.

Problem 1. Short-answer questions.

(a) Consider a 2-dimensional geometric domain P (e.g., a simply polygon). A circular disk D that is contained within P is said to be *maximal* if there is no circular disk D' of larger radius such that $D \subset D' \subset P$. (For example, in Fig. 1, the disk D_1 is not maximal, since it is contained within D_1', but D_2 is maximal.)

What is the set of centers of all maximal disks called? Why is this structure useful in path planning? (A short answer is sufficient.)

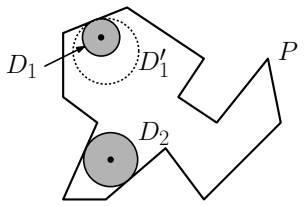

Figure 1: Problem 1(a): Maximal and non-maximal disks.

(b) Consider A*-search in the context of finding a shortest path from a source vertex s to a target vertex t in a graph. For any two vertices u and v, let $\delta(u, v)$ denote the length of the shortest path between them in the graph. For any node u, let $h(u)$ denote the heuristic function.

(i) What property must the heuristic function h satisfy to be *admissible*?

(ii) What property must h satisfy to be *consistent*?

(iii) Suppose that the vertices are points in the Euclidean plane and the weight of each edge of the graph is the Euclidean distance between these vertices. Let $\text{dist}(u, v)$ denote the (straight-line) Euclidean distance between nodes u and v. Is the heuristic $h(u) = 3 \cdot \text{dist}(u, t)$ admissible? Is it consistent? Explain your answers briefly.

(c) Draw a picture showing how to encode the following functionality in your game through the use of a *behavior tree* for a hungry student:

> We [*look in our refrigerator for food*]. If we find some, we [*eat it*] and we are done. Otherwise, we next [*go to our friend's place*], and we [*check our friend's refrigerator*]. If his refrigerator has food, we [*eat it*] and we are again done. Otherwise, we [*go to the 7-Eleven*], and [*pick out some food*]. We then [*attempt to buy it*]. If we have enough money, we [*eat it*] and again we are done. Otherwise, we have failed.

The leaf-level tasks of your behavior tree should correspond to the items in square brackets. Some of these tasks have an implicit outcome that can be success (e.g., found food in refrigerator) or a failure (didn't find food). Others always succeed. Indicate clearly which nodes are *sequence nodes* ("$\longrightarrow$") and which are *selector nodes* ("?")

(d) For each of the following, indicate whether it is (more) true of TCP (transmission control protocol) or UDP (user datagram protocol).

(i) Sends acknowledgments when packets are received.

(ii) Packets can be ordered through the use of sequence numbers.

(iii) Uses checksums to detect errors in packet transmission.

(iv) Has lower overhead in terms of bandwidth and latency.

(e) A cheater in an online FPS game has altered the graphics software to eliminate fog from the game, thus making it easier to see and target the opponents. Under which of the following cheating methods would this fall? (Select one.)

(i) Information Exposure

(ii) Reflex Augmentation

(iii) Authoritative Clients

(iv) Compromised servers

Problem 2. In the Ramer-Douglas-Peucker algorithm, we demonstrated how to reduce the number of vertices from a simple polygon, but what impact does this have on the size of the resulting triangulation? In this problem, we will see how a reduction in the number of vertices affects the number of triangles and chords.

You are given a simple polygon with h holes and n total vertices (including the vertices of the holes). Recall that a *triangulation* is a structure that decomposes the polygon into triangles by adding line segments, called *chords*, between pairs of visible vertices. It is a remarkable fact that the number of triangles and the number of chords in any triangulation of such a polygon is determined exclusively by n and h and does not depend on the shape of the polygon or arrangement of the holes. (A few examples are given in Fig. 2.)

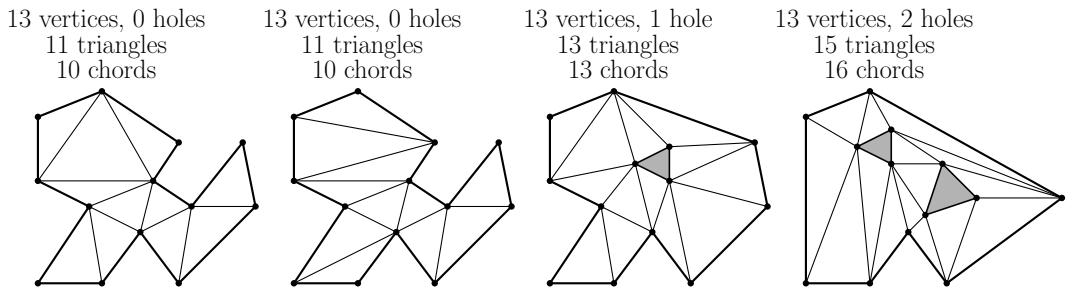

Figure 2: Problem 1(a): Triangles and chords in a polygon.

Give a formula that expresses the number triangles and the number of chords as a function of n and h. (We just need two formulas involving the variables n and h, nothing more.)

Extra Credit: Provide an informal proof of your formula. This can be done by induction. One way is to start from a trivial polygon (a triangle) and successively add vertices until you build up to the final polygon. Another way is to start with the final polygon, and successively add chords until no more can be added.

Problem 3. Given vertex v in a cell complex of a 2-manifold, the *link* of v is defined to be the edges that bound the faces that are incident to v, excluding the edges that are incident to v itself. (For example, in Fig. 3, the link of v consists of the edges $\langle e_0, e_1, \ldots, e_7 \rangle$. Another way to think of the link is to imagine that v and its incident edges had been removed from the cell complex. The link consists of the edges bounding the resulting face.)

Present a procedure (in pseudocode) that, given a vertex v of the DCEL, returns a list L consisting of the half edges of v's link. The half edges should be directed in *counterclockwise* order about v and the list should also be ordered in the same way. The list can start with any half edge of the link. Your procedure should run in time proportional to the length of the output.

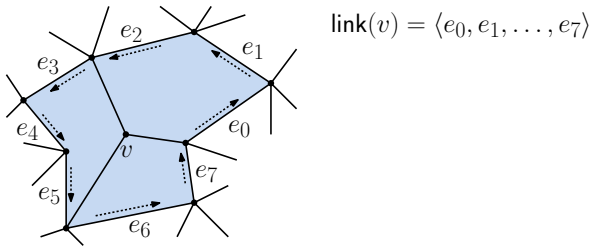

Figure 3: Problem 1: Computing a link in a DCEL.

Problem 4. You are given a triangular robot R and two obstacles O_1 and O_2 as shown in Fig. 4. The reference point of R is its leftmost vertex (indicated by s in the figure).

We want to compute a short path that along which we can translate R from $s = (0, 0)$ to $t = (6, 5)$ while avoiding these two obstacles.

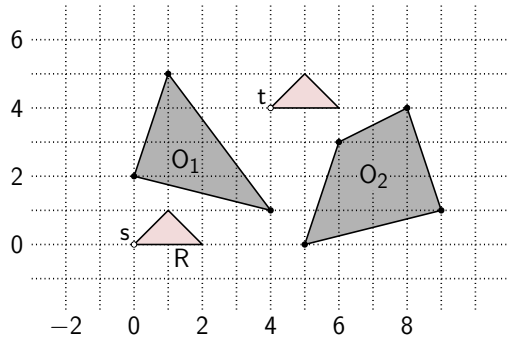

Figure 4: Problem 2: Building configuration obstacles.

(a) Draw a picture showing the C-obstacles for O_1 and O_2 with respect to R. Either overlay your drawing on a piece of graph paper so we can check that the vertices are correct, or label each vertex of the C-obstacles with its coordinates. (Hint: The vertex coordinates are all integer-valued.)

(b) Using your result from (a), draw the shortest path obstacle-avoiding path to translate the robot from s to t.