CS 478 - Computational Geometry Homework 1

Barkın Saday

February 2025

Question 1

For a segment tree $T(l, r)$, the total number of elementary intervals is $n = r - l$.

The segment tree is a binary tree where each node covers a specific interval, and the leaf nodes cover elementary intervals of size 1. Each internal node represents the union of its two children. The tree has a height of $\lceil \log_2 n \rceil$.

When querying an arbitrary interval $[b, e]$, we need to cover this interval with the fewest standard intervals (intervals represented by nodes of the segment tree). The strategy to achieve this minimum is to select the largest possible standard intervals that fit within $[b, e]$.

Key Observations

1. The maximum number of standard intervals needed to cover an arbitrary interval $[b, e]$ is achieved when both the left and right edges of the interval are not aligned with the boundaries of any standard intervals.

2. The coverage strategy involves selecting: - The largest left-leaning interval that ends at or before b. - The largest right-leaning interval that starts at or after e. - Intermediate full intervals if there is any gap remaining.

Maximum Number of Intervals

- The left boundary contributes at most $\lceil \log_2 n \rceil - 1$ intervals. - The right boundary also contributes at most $\lceil \log_2 n \rceil - 1$ intervals. - The middle section, if any, contributes 1 additional interval.

Combining these:

$$(\lceil \log_2 n \rceil - 1) + (\lceil \log_2 n \rceil - 1) + 1 = 2\lceil \log_2 n \rceil - 2.$$

Since

$$\lceil \log_2 n \rceil + \lfloor \log_2 n \rfloor = 2\lceil \log_2 n \rceil - 1$$

when n is not a power of 2, and $-1 = -2 + 1$, we can rewrite this as:

$$\lceil \log_2 n \rceil + \lfloor \log_2 n \rfloor - 2.$$

Example to Achieve Maximum on t(1, 16)

Consider the segment tree $T(1, 16)$ where $n = 16$.

$$\lceil \log_2 16 \rceil = \lfloor \log_2 16 \rfloor = 4.$$

The formula gives:

$$4 + 4 - 2 = 6.$$

Example Interval: $[3, 14]$

- **Left Intervals:** Cover from 1 to 2 using interval $[1, 2]$. - **Right Intervals:** Cover from 15 to 16 using interval $[15, 16]$. - **Middle Intervals:** Cover the remaining part using four intervals: $[3, 4]$, $[5, 8]$, $[9, 12]$, $[13, 14]$.

This gives us:

$$1 \text{ (left)} + 4 \text{ (middle)} + 1 \text{ (right)} = 6.$$

This example achieves the theoretical maximum of 6 intervals.

Question 2

Step-by-Step Algorithm

1. **Initialize Set of Faces to Merge:**
   - Create a set $F_{\text{Union}} = \{F\}$.
   - Iterate over all edges e in the DCEL:
     - If $e.leftF = F$, add $e.rightF$ to F_{Union}.
     - If $e.rightF = F$, add $e.leftF$ to F_{Union}.
2. **Identify Edges to Delete:**
   - Create a set $E_{\text{delete}} = \emptyset$.
   - For each edge e in the DCEL:
     - If both $e.leftF \in F_{\text{Union}}$ and $e.rightF \in F_{\text{Union}}$:
       - Add e to E_{delete}.
3. **Delete Internal Edges:**
   - For each edge $e \in E_{\text{delete}}$:
     - Call `DeleteEdge(e.id)`.
     - Update the `next` and `prev` pointers of surrounding edges.
4. **Update Face References:**
   - For all remaining edges with $e.leftF \in F_{\text{Union}}$ or $e.rightF \in F_{\text{Union}}$:
     - Update the face reference to the merged face (e.g., keep F as the representative face).
5. **Cleanup Unused Faces and Vertices:**
   - Remove any face records that are no longer used.
   - If a vertex has no incident edges, remove it from the DCEL.
6. **Ensure PSLG Validity:**
   - Verify that the `next` and `prev` pointers form a valid cycle around the new face.
   - Confirm that all vertices are connected to at least one edge.

Why This Algorithm Works

This algorithm ensures that the face expansion is done correctly under various scenarios:

- **Simple Adjacent Faces:** When the neighboring faces are simply connected to the initial face F, the algorithm will merge them by deleting the internal edges and updating the boundary correctly.
- **Shared Edges Among Neighbors:** If two neighboring faces share an edge, this edge will also be deleted, maintaining a clean boundary around the merged face. The use of the F_{Union} set ensures that no unnecessary edge deletions occur.
- **Complex Neighboring Structures:** For complex configurations where multiple neighboring faces are interconnected, the algorithm's approach of checking both 'leftF' and 'rightF' in the set guarantees that only internal edges are removed, preserving the PSLG structure.
- **Maintaining the DCEL Validity:** The careful update of 'next' and 'prev' pointers, as well as face references, ensures that the DCEL remains a valid representation of a PSLG. No isolated vertices or invalid edges are left behind.

Conclusion

The proposed algorithm efficiently expands a face in the DCEL by identifying and deleting internal edges while maintaining the PSLG structure. The approach ensures that only edges entirely enclosed by the merged faces are removed, and all necessary pointers and references are updated to preserve the validity of the DCEL.

Question 3

Key Idea

When two adjacent faces F_a and F_b share a common edge E, removing E and combining their boundaries yields a new polygon. To decide if that polygon is convex or concave, the algorithm will:

1. **Retrieve the boundary vertices** of F_a and F_b.
2. **Remove the shared boundary edge** E (and its twin) from the boundary representation.
3. **Merge** the vertex cycles of F_a and F_b into a single cycle representing the union polygon.
4. **Check convexity** of the resulting polygon by examining the cross product of consecutive triples of vertices.

Algorithm

1. **Identify the Shared Edge**:
   - Iterate through the edges of F_a to find the edge e whose $leftF$ or $rightF$ is F_b.
2. **Collect Vertices of F_a and F_b without the Shared Edge**:
   - Traverse the boundary of F_a, collecting vertices while skipping e.
   - Traverse the boundary of F_b, collecting vertices while skipping the twin of e.
3. **Merge the Two Vertex Cycles**:
   - Concatenate the vertex lists of F_a and F_b to form a closed loop.
   - Ensure the orientation of the merged polygon is consistent (clockwise or counterclockwise).
4. **Check Convexity**:
   - For each triplet of consecutive vertices (v_1, v_2, v_3) in the merged cycle:
     - Compute the cross product of vectors $\overrightarrow{v_1v_2}$ and $\overrightarrow{v_2v_3}$.
     - Keep track of the sign of the cross product.
     - If any cross product sign is inconsistent with the initial sign, the polygon is concave.
5. **Return the Result**:
   - If all cross products have a consistent sign, return `true` (polygon is convex).
   - Otherwise, return `false` (polygon is concave).

Why This Algorithm Works

- **Consistency of Cross Products:** Convex polygons maintain a consistent turning direction (either always left or always right) when walking along the vertices. The cross product method is a robust way to detect this.
- **Skipping the Shared Edge:** By avoiding the shared edge during vertex collection, the algorithm ensures the union polygon is represented correctly without artifacts from the removed edge.
- **Handling Edge Cases:** The algorithm naturally handles degenerate cases, such as collinear vertices or minimal polygons (triangles), by relying on the geometric properties of cross products.

Conclusion

The proposed algorithm provides an efficient method to determine whether the union of two adjacent faces in a DCEL forms a convex or concave polygon. By leveraging the properties of cross products and maintaining a consistent traversal of the DCEL structure, the method is robust and ensures the validity of the resulting polygon.

Question 4

Take a look at the below figures that use the same set of points $S = \{A = (0,0), B = (0,10)C = (2,10), D = (5,1), E = (2,0)\}$ with different edges. The resulting polygons are not the same. Thus, they **do not** have to be unique.

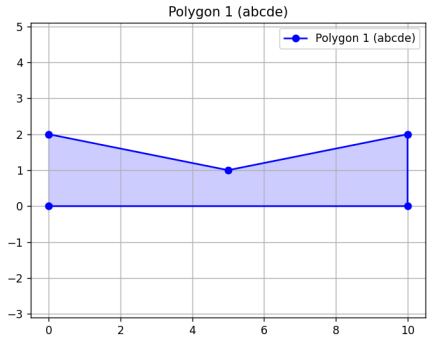

Figure 1: Polygon 1 ($ABCDE$)

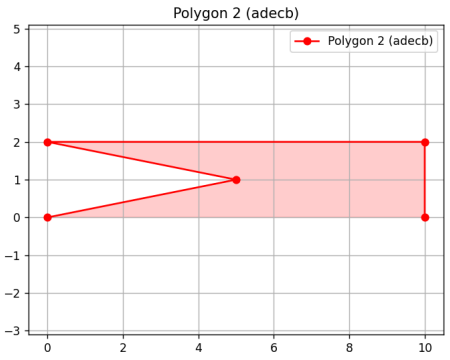

Figure 2: Polygon 2 ($ADECB$)

Question 5

Assuming each vertex has degree of at least 3.

1. Prove $v \leq \frac{2}{3}e$

Since each vertex has a degree of at least 3:

$$v \cdot 3 \leq 2e \implies v \leq \frac{2e}{3}$$

Proved.

2. Prove $e \leq 3v - 6$

Using Euler's formula:

$$f = e - v + 2$$

Each face in a planar graph is bounded by at least 3 edges:

$$3f \leq 2e$$

Substitute $f = e - v + 2$ into the inequality:

$$3(e - v + 2) \leq 2e$$

Expand and simplify:

$$3e - 3v + 6 \leq 2e$$

$$e \leq 3v - 6$$

Proved.

3. Prove $e \leq 3f - 6$

From the inequality $3f \leq 2e$:

$$f \leq \frac{2e}{3}$$

Rearrange to express in terms of e:

$$e \leq \frac{3}{2}f$$

Using Euler's formula $f = e - v + 2$ and $e \leq 3v - 6$:

$$e \leq 3f - 6$$

Proved.

4. Prove $f \leq \frac{2}{3}e$

Using the face degree assumption that each face is bounded by at least 3 edges:

$$3f \leq 2e \implies f \leq \frac{2e}{3}$$

Proved.

5. Prove $v \leq 2f - 4$

Using Euler's formula:

$$v = e - f + 2$$

Since:

$$e \leq 3f - 6$$

Substitute:

$$v \leq (3f - 6) - f + 2$$

Simplify:

$$v \leq 2f - 4$$

Proved.

6. Prove $f \leq 2v - 4$

From Euler's formula:

$$f = e - v + 2$$

Since:

$$e \leq 3v - 6$$

Substitute:

$$f \leq (3v - 6) - v + 2$$

Simplify:

$$f \leq 2v - 4$$

Proved.

Proving $e = 3v - 6$ for Triangulated PSLG

For a triangulated Planar Straight Line Graph (PSLG), every face is a triangle. This implies that each face is bounded by exactly 3 edges. Therefore, the total sum of the degrees of all faces is:

$$3f = 2e \implies f = \frac{2e}{3}$$

Using Euler's formula for planar graphs:

$$v - e + f = 2$$

Substitute $f = \frac{2e}{3}$ into Euler's formula:

$$v - e + \frac{2e}{3} = 2$$

Clear the fraction by multiplying by 3:

$$3v - 3e + 2e = 6$$

Simplify:

$$3v - e = 6$$

Rearrange to get the final result:

$$e = 3v - 6$$

Proved.