Computational Geometry 1st Set

Emmanouil-Thomas Chatzakis : 2021030061

1

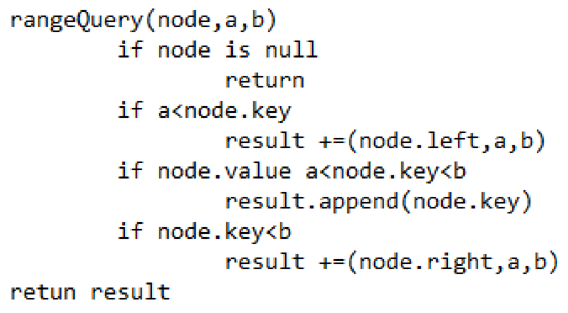
Assume that T is a binary search tree with n nodes and height h. Prove that a range query returning k elements (all keys x such that $a \leq x \leq b$) has cost $O(h + k)$. Then, give a pseudocode for such a range search procedure.

1.1 Solution

Properties of Binary Search Tree:

- The left subtree of a node contains only nodes with keys less than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.

In the worst case, performing a range query search in a Binary Search Tree (BST) takes O(h) time. The algorithm works as follows: starting from the root node, it compares the current key with the given range limits (a and b). Based on this comparison, the algorithm determines which subtree to explore further. The process is repeated recursively for each node, narrowing down the search to the relevant subtrees. If the key is less than the lower bound (a), the search proceeds to the right subtree; if it is greater than the upper bound (b), the search continues in the left subtree. If the key lies within the range, it is returned; otherwise, no result is returned. When the algorithm finds the keys of the range query it takes some constant time O(k) to report each value. So the complexity of the algorithm is O(h+k)



```
rangeQuery(node,a,b)
        if node is null
                return
        if a<node.key
                result +=(node.left,a,b)
        if node.value a<node.key<b
                result.append(node.key)
        if node.key<b
                result +=(node.right,a,b)
retun result
```

Figure 1:

2

Prove that, for any $p_1, p_2, q_1, q_2 \in R^2$, such that no three are collinear, segment $\overline{p_1p_2}$ intersects segment $\overline{q_1q_2}$ if and only if the closed curve defined by the sequence $[p_1, q_1, p_2, q_2]$ is a convex polygon.
Based on this, describe an algorithm that takes as input two line segments, defined by the pairs of endpoints (p_1, p_2) and (q_1, q_2) respectively, where $p_1, p_2, q_1, q_2 \in R^2$. The algorithm returns `TRUE` if the two segments intersect, and `FALSE` otherwise.

2.1 Solution

First of all we will prove that if segment $\overline{p_1p_2}$ intersect with segment $\overline{q_1q_2}$ then the closed curved defined by $[p_1, q_1, p_2, q_2]$ is a convex polygon.

When the segments intersect they divide the plane into four regions. Since no three points are collinear then each point p_1, p_2, q_1, q_2 or each vertex of the close curve must lie on a different region of the plane. Based on that if we traverse through the vertices $[p_1, q_1, p_2, q_2]$ the turns that we make are on the same direction.

This means that the cross product of three consecutive vertices

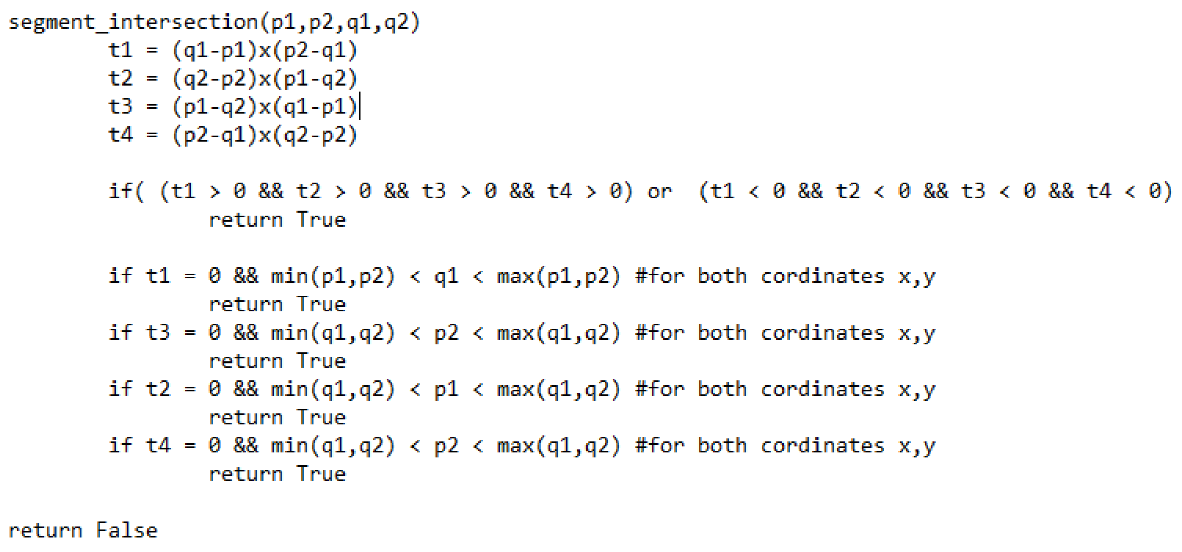
$$(q_1 - p_1) \times (p_2 - q_1)$$
$$(p_2 - q_1) \times (q_2 - p_2)$$
$$(q_2 - p_2) \times (p_1 - q_2)$$
$$(p_1 - q_2) \times (q_1 - p_1)$$

must have the same sign.

Now we will prove the opposite direction. Let $[p_1, q_1, p_2, q_2]$ be a convex polygon and lets assume that the line segments $\overline{p_1p_2}$ and $\overline{q_1q_2}$ do not intersect. IF the segments do not intersect then at least one diagonal of the polygon must lie outside of it. This is means that the polygon has one vertex that "points inward" making it concave. This contradicts our assumption that the polygon is convex. Therefore, if the polygon is convex,the line line segments $\overline{p_1p_2}$ and $\overline{q_1q_2}$ must intersect.

The algorithm that checks if two given line segments intersect must:

- Check the direction of the turn formed by three consecutive vertices of the polygon ($[p_1, q_1, p_2, q_2]$) for all triplets of vertices
- Check special cases such:
  1. q_1 lies on $\overline{p_1p_2}$
  2. q_2 lies on $\overline{p_1p_2}$
  3. p_1 lies on $\overline{q_1q_2}$
  4. p_2 lies on $\overline{q_1q_2}$



```
segment_intersection(p1,p2,q1,q2)
        t1 = (q1-p1)x(p2-q1)
        t2 = (q2-p2)x(p1-q2)
        t3 = (p1-q2)x(q1-p1)
        t4 = (p2-q1)x(q2-p2)

        if( (t1 > 0 && t2 > 0 && t3 > 0 && t4 > 0) or  (t1 < 0 && t2 < 0 && t3 < 0 && t4 < 0)
                return True

        if t1 = 0 && min(p1,p2) < q1 < max(p1,p2) #for both cordinates x,y
                return True
        if t3 = 0 && min(q1,q2) < p2 < max(q1,q2) #for both cordinates x,y
                return True
        if t2 = 0 && min(q1,q2) < p1 < max(q1,q2) #for both cordinates x,y
                return True
        if t4 = 0 && min(q1,q2) < p2 < max(q1,q2) #for both cordinates x,y
                return True

return False
```

Figure 2:

3

Describe an $O(n \log n)$-time algorithm for the following problem:

INPUT: A collection S of n vertical planar segments, described as triples $(a_i, b_i, c_i) \in R^3$ such that $b_i < c_i$, corresponding to the vertical segment with endpoints

$$\begin{bmatrix} a_i \\ b_i \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} a_i \\ c_i \end{bmatrix}.$$

OUTPUT: `TRUE` if there exists a line $y = px+q$, intersecting every vertical segment (i.e., for all i, $b_i < pa_i + q < c_i$), and `FALSE` otherwise.

Hint: Hyperplane separation theorem

3.1 Solution

1. Sort the vertical planar segments lexicographically by (a_i). If two segments have the same a_i use (b_i, c_i) to break ties.
   Time Complexity: $\mathcal{O}(n \log n)$.

2. Split the sorted list of segments into two halves: The algorithm recursively solves for each subset. If at least one of the subsets fails (no valid line exists), return FALSE.

3. For each segment (a_i, b_i, c_i), the following constraint must be satisfied for the line $y = px + q$ to intersect the segment :

$$b_i < p \cdot a_i + q < c_i.$$

   This inequality defines a set of valid (p, q) pairs in the (p,q)-space bounded by two parallel lines:

$$p \cdot a_i + q > b_i \qquad \text{(Lower bound)}$$
$$p \cdot a_i + q < c_i \qquad \text{(Upper bound)}$$

These bounds create a region in the (p, q)-plane where any point (p, q) represents a line that intersects the given segment. For a subset of segments we combine these regions by taking their intersection. This intersection forms a convex set in the (p, q)-space, representing all lines that intersect all segments in the subset.

4. Let:

   - R_1: The convex region defined by all constraints from the left half (subset).
   - R_2: The convex region defined by all constraints from the right half (subset).

   1. Now lets represent each convex region (R_1, R_2) as a set of linear inequalities formed by their regions (step 3)

   $$R_1 = \bigcap_{i=1}^{n} (b_i < pa_i + q < c_i), \quad R_2 = \bigcap_{j=1}^{m} (b_j < pa_j + q < c_j).$$

   2. Check whether these regions overlap.

      (a) If $R_1 \cap R_2 = \emptyset$, then no valid line exists (return FALSE).

      (b) If $R_1 \cap R_2 \neq \emptyset$, then there exists a valid line (return TRUE).

5. Finaly If regions overlap at every merge step, return TRUE. Otherwise, return FALSE.

The algorithm runs in $\mathcal{O}(n \log n)$ time complexity:

- Sorting: $\mathcal{O}(n \log n)$
- Divide-and-Conquer Recursion: $\log n$ levels
- Merging Constraints at Each Level: $\mathcal{O}(n)$ per level

4

Prove that the convex hull of a finite set of points in R^2 is the minimum-perimeter simple polygon containing the set.

4.1 Solution

The convex hull $CH(S)$ of a set $S \subseteq R^2$ is defined as:

$$CH(S) = \left\{ \lambda_1 x_1 + \cdots + \lambda_n x_n : x_i \in S,\, \lambda_i \geq 0,\, \sum_{i=1}^{n} \lambda_i = 1 \right\}.$$

Alternatively, it can also be defined as:

$$CH(S) = \bigcap \{C \supseteq S : C \text{ is convex}\}.$$

Or equivalently:

$$CH(S) = \bigcap \{H \supseteq S : H \text{ is a halfspace (open or closed)}\}.$$

From these definitions, $CH(S)$ satisfies the following properties:

- $S \subseteq CH(S)$,
- $CH(S)$ is convex,
- $CH(S)$ is the smallest convex set containing S.

The convex hull is surrounded by supporting hyperplanes that each of them define a vertex of the convex hull. We know that the boundary of S intersects the supporting hyperplanes and that CH(S) is the smallest convex set containing S, thus the supporting hyperplanes surrounding the convex hull form the minimal enclosing boundary.

Lets assume P is a simple polygon that contains all points of S and that P is not convex. Therefore exists at least one concave turn defined by some consecutive points p_1, p_2, p_3. By replacing the line segments $\overline{p_1p_2}$ and $\overline{p_2p_3}$ with the line segment $\overline{p_1p_3}$, we can reduce or maintain the perimeter of P as the triangle inequality suggests. Thereupon any simple polygon containing all points of S must be convex to minimize its perimeter.

Finally we will prove that the convex hull is the simple convex polygon with the smallest perimeter.

The convex hull is the convex set enclosing all point of S. It only containts vertices that are extreme points. Any other convex polygon that contains all points of S must have vertices that lie outside of the minimal enclosing boundary defined by the convex hull. For that reason the perimeter of these convex polygon is increased. **Therefore the convex hull is the simple convex polygon with the smallest perimeter.**

5

Let $p_i \in R^3$, $i = 1, \ldots, 3$ be the vertices of a 3-d triangle. Consider the ray (half line) $a + \lambda D$, where $a, D \in R^3$ and $\lambda \geq 0$. With p_i, a, D given, write a predicate to compute whether the ray intersects the interior of the triangle.Also, write a formula that returns the intersection point (when it exists and is unique).

5.1 Solution

First we will find the perpedicular vector to the plane containing the triangle. To find this vector (let us name it G), we use the cross product.

$$N = (p_2 - p_1) \times (p_3 - p_1) \tag{1}$$

A point P is on the plane if and only if

$$(P - p_1) \cdot N \;=\; 0 \tag{2}$$

Given the ray $a + \lambda D$ we will check if its parallel to the plane containing the triangle. if

$$D \cdot N \;=\; 0 \tag{3}$$

then the line it's perpedicular to the vector G. From this result there are two cases:

1. The line lies entirely in the plane:
$$(a - p_1) \cdot N \;=\; 0$$
2. The line is outside the plane:
$$(a - p_1) \cdot N \neq 0$$

The point P that the ray intersects with the plane containing the triangle is given by the formula:

$$P(\lambda) = a + \lambda D \tag{4}$$

if we substitute in (2) the we get:

$$(P(\lambda) - p_1) \cdot N = 0 \Rightarrow$$

$$\Rightarrow (a + \lambda D) \cdot N = 0 \Rightarrow$$

$$\Rightarrow a \cdot N + \lambda D \cdot N - p_1 \cdot N = 0 \Rightarrow$$

$$\Rightarrow (a - p_1) \cdot N + \lambda D \cdot N = 0 \Rightarrow$$

$$\Rightarrow \lambda = \frac{(p_1 - a) \cdot N}{D \cdot N} \tag{5}$$

If $\lambda < 0$ then points are behind the starting point a, which are outside the domain of the ray.

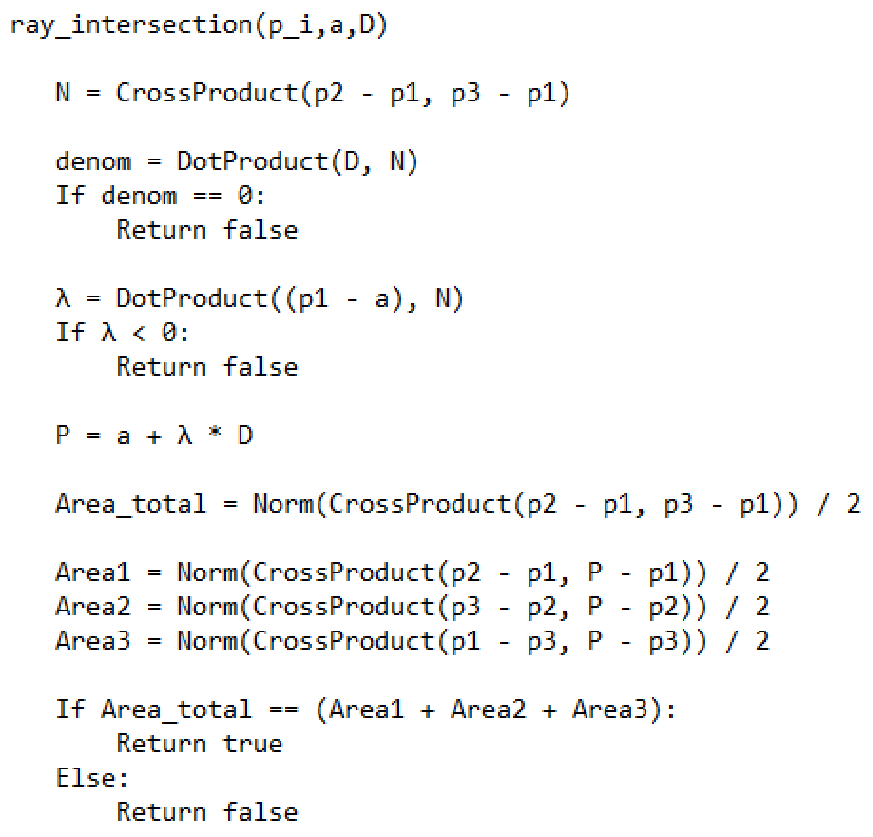
Finally, to find if the point P intersects with the interior of the triangle, we can use a geometric property : A point inside a triangle can create three smaller triangles whose areas are equal to the starting triangle. So for the point to be inside the triangle the equation below must be true:

$$Area_{p_1,p_2,P} + Area_{p_2,p_3,P} + Area_{p_1,p_3,P} = Area_{p_1,p_2,p_3} \tag{6}$$

If an area of the smaller triangles is zero, that means that the point P lies on a vertex, so we want all the areas to be greater than zero.

The formula of the intersection is

$$P(\lambda) = a + \frac{(p_1 - a) \cdot N}{D \cdot N} D \tag{7}$$



```
ray_intersection(p_i,a,D)

   N = CrossProduct(p2 - p1, p3 - p1)

   denom = DotProduct(D, N)
   If denom == 0:
       Return false

   λ = DotProduct((p1 - a), N)
   If λ < 0:
       Return false

   P = a + λ * D

   Area_total = Norm(CrossProduct(p2 - p1, p3 - p1)) / 2

   Area1 = Norm(CrossProduct(p2 - p1, P - p1)) / 2
   Area2 = Norm(CrossProduct(p3 - p2, P - p2)) / 2
   Area3 = Norm(CrossProduct(p1 - p3, P - p3)) / 2

   If Area_total == (Area1 + Area2 + Area3):
       Return true
   Else:
       Return false
```

Figure 3: