ACT-22 MARCH, 1980

CSL COORDINATED SCIENCE LABORATORY

APPLIED COMPUTATION THEORY GROUP

SEGMENTS, RECTANGLES, CONTOURS

WITOLD LIPSKI, Jr.
FRANCO P. PREPARATA

REPORT R-876 UILU-ENG 80-2208

UNIVERSITY OF ILLINOIS - URBANA, ILLINOIS

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Segments, Rectangles, Contours		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER R-876 (ACT-22); UILUENG 80-2208
7. AUTHOR(s) Witold Lipski, Jr. Franco P. Preparata		8. CONTRACT OR GRANT NUMBER(s) NSF MCS-78-13642; N00014-79-C-0424
9. PERFORMING ORGANIZATION NAME AND ADDRESS Coordinated Science Laboratory University of Illinois Urbana, Illinois 61801		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS National Science Foundation Joint Services Electronics Program		12. REPORT DATE March, 1980 13. NUMBER OF PAGES 22
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

Approved for public release; distribution unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Computational geometry, segments, contours, rectangles, point-location, searching, segment-trees

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Two sets H and V of horizontal and vertical segments, respectively, determine a subdivision M of the plane into regions. A nontrivial region is one whose boundary contains an end-portion of nonzero length of at least one segment, and the nontrivial contour of M is the collection of the boundaries of nontrivial regions. In this paper we consider several problems pertaining to H and V, such as the construction of the nontrivial contour of M, of the external contour of M, and of a path between two points in the plane avoiding the segments (route-in-a-maze). We show that, if $|H| + |V| = n$, all of these problems are solved in

DD FORM 1 JAN 73 1473

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

time O(nlogn), by making use of a static data structure, called the adjacency map, which can be searched in time O(logn) and can be constructed in time O(nlogn). The algorithms for the nontrivial and external contour are shown to be optimal.

SEGMENTS, RECTANGLES, CONTOURS

by

Witold Lipski, Jr. and Franco P. Preparata

This work was supported in part by the National Science Foundation under Grant MCS 78-13642 and Joint Services Electronics Program (U.S. Army, U.S. Navy and U.S. Air Force) under Contract N00014-79-C-0424.

SEGMENTS, RECTANGLES, CONTOURS

Witold Lipski, Jr.* and Franco P. Preparata**

Abstract

Two sets H and V of horizontal and vertical segments, respectively, determine a subdivision M of the plane into regions. A nontrivial region is one whose boundary contains an end-portion of nonzero length of at least one segment, and the nontrivial contour of M is the collection of the boundaries of nontrivial regions. In this paper we consider several problems pertaining to H and V, such as the construction of the nontrivial contour of M, of the external contour of M, and of a path between two points in the plane avoiding the segments (route-in-a-maze). We show that, if $|H| + |V| = n$, all of these problems are solved in time O(nlogn), by making use of a static data structure, called the adjacency map, which can be searched in time O(logn) and can be constructed in time O(nlogn). The algorithms for the nontrivial and external contour are shown to be optimal.

Keywords and phrases: Computational geometry, segments, contours, rectangles, point-location, searching, segment-trees.

---

*Institute of Computer Science, Polish Academy of Sciences, P.O. Box 22, 00-901 Warsaw PKiN, Poland.

**Coordinated Science Laboratory, Department of Electrical Engineering, and Department of Computer Science, University of Illinois, Urbana, Illinois 61801, U.S.A.

This work was supported in part by the National Science Foundation under Grant MCS 78-13642 and in part by the Joint Services Electronics Program (U.S. Army, U.S. Navy and U.S. Air Force) under Contract N00014-79-C-0424.

1. Introduction

Several applications arising in diverse fields such as computer aided design, operations research, large-scale-integration, data base concurrency control, puzzles, etc., [1-4] call for the solutions of problems whose basic ingredients are two mutually orthogonal collections of parallel straight line segments in the plane. Although in many practical cases — typically, we may think of VLSI applications — these segments are constrained to form the boundaries of rectangles, there is some gain in generality, and applicability to a wider class of problems, if we consider the segments as unconstrained, except that no two parallel segments are allowed to overlap. In general, we shall view the two families H and V of segments as being respectively parallel to the x- and the y-axis of the plane (horizontal and vertical, respectively).

In this paper we shall consider a number of diverse geometric problems and we shall show how they can all be efficiently solved by the same technique. The common setting of all these problems are the two finite sets H and V of horizontal and vertical segments; denoting by E^2 the Euclidean plane, the point set $M = E^2 - H \cup V$ (relative complement of $H \cup V$) is a subdivision of the plane consisting of a finite number of connected regions, of which one is unbounded and is called external. We now describe the problems to be considered.

(1) NONTRIVIAL-CONTOUR. In the planar subdivision M we call nontrivial any region whose boundary contains an end-portion of nonzero length of at least one segment (that is, a segment endpoint and a nontrivial portion of that segment); all other regions are called trivial. (Notice that each trivial region is a rectangle, although the converse is not true.) The NONTRIVIAL-CONTOUR problem consists in producing the boundaries of all nontrivial regions in M. A problem reducible to NONTRIVIAL-CONTOUR is

(2) EXTERNAL-CONTOUR. Here the objective is the determination of the boundary of the external region (external contour). Notice that, in general, the external contour is a subset of the nontrivial contour.

(3) POINT-LOCATION (in a segment-induced planar subdivision). Given M and a target point p, find the region of M which contains p. Of course, this problem is interesting if such searches have to be made in a repetitive mode, so that it pays to preprocess M in order to ease the search. A problem related to "point location" is

(4) ROUTE-IN-A-MAZE. Consider the maze created by the two collections of segments. A route between two selected points s and t in the plane is a curve connecting the two points without crossing any segment. Obviously a route is entirely contained in a region of M, so the existence of a route can be decided by POINT-LOCATION, since s and t must belong to the same region of M. The objective of ROUTE-IN-A-MAZE, however, is the actual construction of such route if it is possible.

The paper is organized as follows. In Section 2 we shall describe a basic data structure, called "adjacency map", and its search algorithm, which is efficiently applicable to the solutions of the above problems, and presumably of many others arising in similar contexts. In Section 3 we shall describe in detail the solutions of the selected problems, while in Section 4 we shall develop some efficiency considerations regarding the proposed methods.

2. The adjacency maps

As we mentioned earlier, our approach to the previously illustrated problems rests on the construction of two data structures, called the horizontal and vertical adjacency maps, and is an adaptation of a recently developed planar point-location technique [5]. We shall

confine our presentation to the horizontal adjacency map (HAM); the discussion is applicable with trivial changes to the vertical adjacency map (VAM).

Consider the set V of the vertical segments (figure 1a). Through each endpoint p of each member of V we trace a horizontal half-line to the right and one to the left; each of these half-lines either terminates on the vertical segment closest to p or, if no such intercept exists, the half-line continues to infinity. In this manner the plane is partitioned into regions, of which two are half-planes and all the others are rectangles, possibly unbounded in one or both horizontal directions (figure 1b). Each rectangle is an equivalence class of points of the plane with respect to their horizontal adjacency to vertical segments (whence the name "horizontal adjacency map"). A simple induction argument shows that

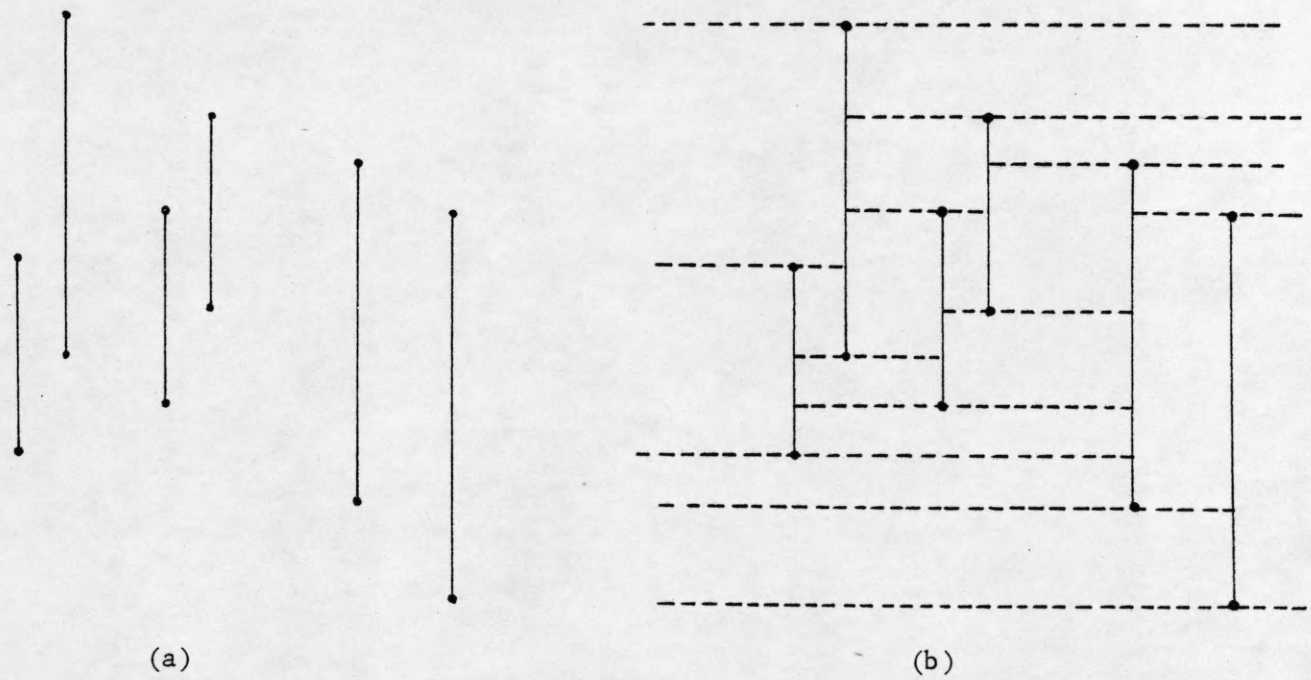


Figure 1. The set V of vertical segments (a) and the resulting horizontal adjacency map (b).

the total number of regions in the HAM is at most $3|V| + 1$.

The horizontal adjacency map is just a special case of a subdivision of the plane induced by an embedded planar graph; therefore the point-location problem in the HAM can be handled by one of the general techniques recently proposed to solve this problem [5 - 7]. Particularly suited to this instance is the techniques proposed in [5]; however, due to the specialization that the edges of the graph correspond either to vertical or to horizontal segments, there are some natural simplifications. For the benefit of the reader, we shall now describe the adaptation of the general technique to our instance.

The HAM will be represented by a search data structure $\mathcal{H}$, a binary tree; the construction is based on the normalization of coordinates and the partition of segments as induced by segment trees [8]. Normalization means that the members of each of the two sets of coordinates — abscissae and ordinates — are replaced by their rank in the set (ties are possible, i.e., there are $m \leq 2|V|$ distinct ordinates). Partition of segments is the subdivision of an interval $[e_1, e_2]$ (e_1 and e_2 are integers) into a collection of standard intervals, which are defined by a segment tree. We recall that, for an integer interval $[a, b]$ $(a < b)$, a segment tree $T(a, b)$ consists of a root v with INTERVAL[v] = [a,b], and if $b - a > 1$, of a left subtree $T(a, \lfloor (a+b)/2 \rfloor)$ and a right subtree $T(\lfloor (a+b)/2 \rfloor, b)$; if $b - a = 1$, then the left and right subtrees are empty. In figure 2 we illustrate the tree T(1,11), where each node v is labeled with INTERVAL[v]. It is well-known [5,8] that an interval $[e_1,e_2]$ (with $1 \leq e_1 < e_2 \leq m$) can be subdivided in a standard way into $O(\log m)$ segments, each of which equals INTERVAL[v] for some node v of T(1,m). The partition induced by T(1,11) (figure 2) on the members of V

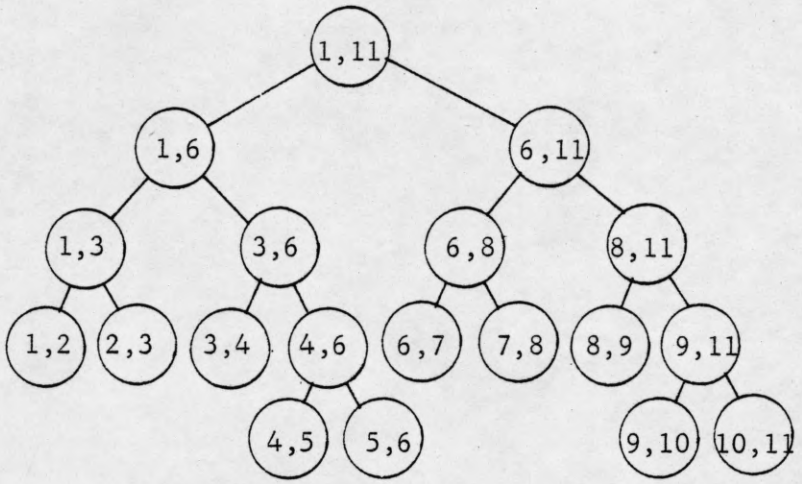


Figure 2. Illustration of T(1, 11).

of our current example (figure 1) is shown in figure 3.

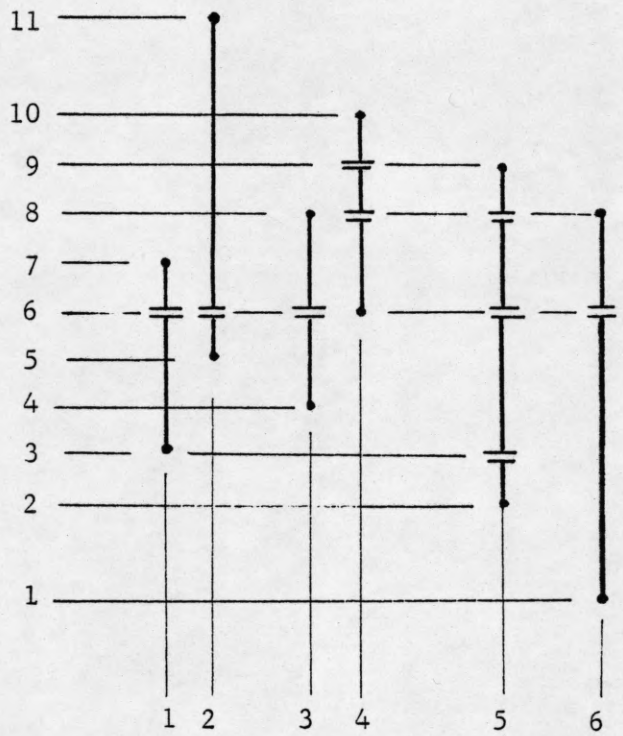


Figure 3. The partition of the members of V induced by T(1, 11).

In the tree $\mathcal{K}$ we shall use two types of nodes, with different pictorial representations: "∇", a ∇-node or "horizontal node", has as discriminant an ordinate; "O", an O-node or "vertical node," has as discriminant an abscissa. A slab [b,t] is the plane strip $b \leq y \leq t$. For any segment $e \in V$, L[e] and U[e] are respectively the lower and upper endpoints of e.

The tree $\mathcal{K}$ is constructed by a recursive procedure. Each recursive call processes one slab, that is, it accepts a slab and a left-to-right sequence S of segments which have a nonempty intersection with the slab. The search tree $\mathcal{K}$ is built by TREE(S*, 1, m), where S* is the left-to-right ordering of the members of V (structured as a queue) and TREE(S, b,t) is the recursive procedure[1] given below. Notice that the procedure implicitly performs the segment partition as induced by the segment tree.

```
procedure TREE(S, b,t)
begin if S = ∅ then TREE ← Λ
    else begin S1 ← S2 ← U ← ∅ (* queues S1, S2, and U are local to
                this procedure*)
                repeat e ⇐ S (*e =Λ if S = ∅*)
                    if (e ≠ Λ) and ((b < L[e]) or (U[e] < t)) then
                      (*in this case [L[e],U[e]] ⊉ [b,t]*)
                      begin if L[e] < ⌊(b + t)/2⌋ then S1 ⇐ e
                            if ⌊(b + t )/2⌋ < U[e] then S2 ⇐ e
                      end
```

[1] If S is queue "S⇐" and "⇐S" denote the "add to" and "remove from" operations, respectively.

```
                else (*in this e = Λ or [L[e],U[e]] ⊇ [b,t]*)
                    begin new (w) (* create a new ∇-node w *)
                        Y[w] ← ⌊(b + t)/2⌋
                        LTREE[w] ← TREE(S1, b, Y[w])
                        RTREE[w] ← TREE(S2, Y[w], t)
                        U ⇐ w
                        if (e ≠ Λ) then U ⇐ X[e]
                    end
            until e = Λ
            𝒰 ← BALANCE (U)
            (* procedure BALANCE takes the alternating sequence U
             of trees and abscissae and arranges these items
             into a balanced tree *)
            return 𝒰
        end
end
```

It has been shown in [5] that BALANCE can be designed so that the total depth of $\mathcal{K}$ is $O(\log|V|)$; moreover, the analysis presented in [5] shows that both the running time and the storage requirement of TREE are $O(|V| \log|V|)$. The tree $\mathcal{K}$ pertaining to our running example (figures 1 and 3) is shown in figure 4 (O-nodes and ∇-nodes are labeled with abscissae and ordinates, respectively). To each node, with less than two offsprings, we append one or two "leaves" so that each shown node has exactly two offsprings; a leaf is simply to be viewed as the termination of a path from the root.

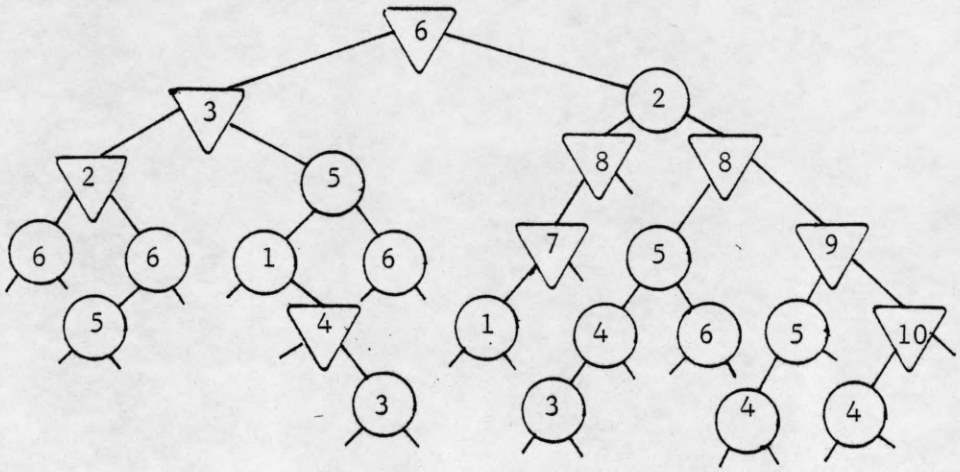


Figure 4. The search tree $\mathcal{K}$ corresponding to the examples of figures 1 and 3.

The horizontal adjacency problem for a given point $p_0 = (x_0, y_0)$ in the plane slab $[1, m]$ is solved by locating p_0 in the planar subdivision described by $\mathcal{K}$. This corresponds to tracing a path in $\mathcal{K}$ from the root to a leaf and recording on this path either (i) the smallest abscissa larger than x_0 for the segment adjacent to the right or (ii) the largest abscissa smaller than x_0 for the segment adjacent to the left; obviously, if on the path the set of abscissae larger than x_0 is empty, then there is no segment adjacent to the right, and analogously for the other case.

3. Applications

3.1 NONTRIVIAL-CONTOUR. Let $|H| + |V| = n$. The number t of edges forming the boundaries of all regions of M may be, in general, $\Omega(n^2)$; however we shall see that the number of edges in the boundaries of all nontrivial regions (see Section 1) is only $O(n)$.

Each segment in H is partitioned into edges by the members of V which intersect it; similarly a segment in V with respect to H (the endpoints of an edge are naturally called vertices). We regard each edge as being lined by two arcs with opposite direction and lying on either side of the edge, as illustrated in figure 5. The boundary of any region in M consists of (directed) circuit(s) of arcs; a circuit is called external or internal depending upon whether it is clockwise or

Figure 5. Conventions on the directions of the lining arcs.

counterclockwise. An arc is said to be terminated if it contains an endpoint of the segment to which it belongs. A circuit is said to be trivial or nontrivial depending upon whether or not it contains a terminated arc. Notice that the nontrivial contour of M - defined as the collection of the boundaries of all nontrivial regions - contains all the nontrivial circuits, but it may contain in addition "trivial" rectangles containing in their interior one or more nontrivial circuits; in any case the number of the latter is less than the number of nontrivial circuits.

A very interesting fact, given as an appendix to this paper, is embodied by the following proposition:

Proposition: The total number of arcs in the nontrivial circuits of M is O(n).

We shall now describe the method to obtain the nontrivial contour. Its central constituent is a procedure for constructing arc-by-arc each nontrivial circuit. The advancing step runs as follows. Starting from the current vertex v_1 (see figure 6) we march along the current segment ℓ_1 in the assigned direction, and one of the following three cases occurs:

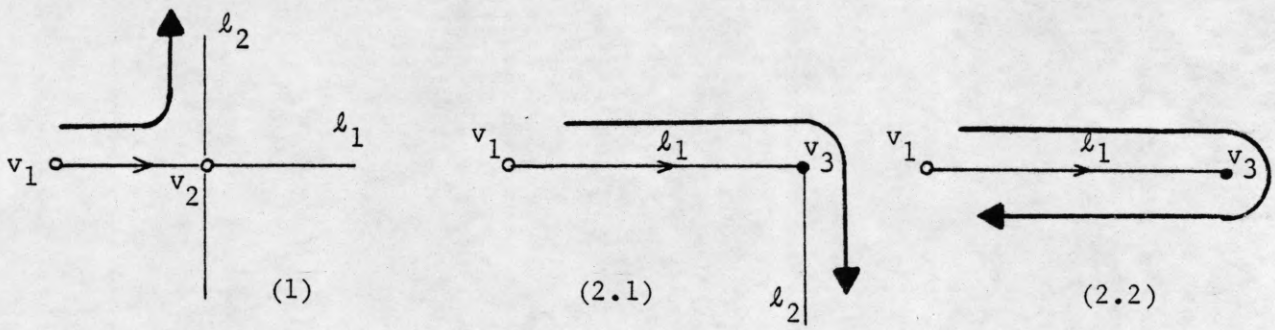


Figure 6. Illustration of the advancing step of the circuit construction procedure.

1. There is a segment ℓ_2, closest to v_1, which intersects ℓ_1 and crosses the region to the left of ℓ_1. In this case we make a left-turn, i.e. the intersection v_2 becomes the current vertex and ℓ_2 becomes the current segment;
2. We reach the endpoint v_3 of ℓ_1, without finding any segment which intersects ℓ_1 and crosses the region to the left of ℓ_1. We now check:
   2.1 v_3 is also the endpoint of a segment ℓ_2 (notice that ℓ_2 can only be in the region to the right of ℓ_1). In this case we make a right-turn, i.e. v_3 becomes the current vertex and ℓ_2 becomes the current segment;
   2.2 v_3 is just the remaining endpoint of ℓ_1. In this case we make a

U-turn, i.e. v_3 becomes the current vertex and ℓ_2, with reversed direction, becomes the current segment.

The implementation of the advancing step is quite simple with the use of the adjacency maps. In fact, suppose that $v_1 = (x_1, y_1)$ and that ℓ_1 belongs to the line $y = y_1$ in the interval $[x_1, x_3]$. We locate in the HAM the point $(x_1, y_1 + \varepsilon)$ (where $\varepsilon > 0$ is arbitrarily small[2]) and we obtain the abscissa x_2 of the closest vertical segment to the right of $(x_1, y_1 + \varepsilon)$. If $x_2 \leq x_3$, then we have a left-turn; if $x_2 > x_3$, then we check whether (x_3, y_1) is also the endpoint of a vertical segment, and resolve between cases 2.1 and 2.2. (There is more than one way to perform the latter check: for example, by locating $(x_1 - \varepsilon, y_1 - \varepsilon)$ in the HAM). The other three possible cases for the current segment (to the left, above, and below the current vertex) are handled in exactly analogous ways. Thus the addition of one arc to a nontrivial circuit costs one (or two) interrogation(s) of either adjacency map; i.e., a computational work bounded by $O(\log n)$.[3]

It should be evident that all nontrivial circuits are generated by initializing the above method to a segment endpoint and a direction on the segment (i.e., either "away" from the endpoint or "toward" it). Therefore, each of the endpoints of members of $H \cup V$ (left, right, bottom, and top)

---

[2] Note that ε need not be explicitly defined; it simply gives a rule on how to break ties in comparisons.

[3] Strictly speaking, a single advancing step may produce an arc which is the union of several consecutive arcs (according to our definition); this happens when several parallel segments terminate on a single orthogonal segment and all lie on one side of it.

gives rise to two "circuit seeds" (of nontrivial circuits), which are naturally called centrifugal (away from the endpoint) and centripetal (toward the endpoint). For example, given segment ℓ - corresponding to the interval $[x_1,x_2]$ $(x_1 < x_2)$ on the line $y = y_1$ - consider endpoint (x_1,y_1) of ℓ: the corresponding centrifugal seed is given by point $(x_1,y_1+\epsilon)$ and direction of increasing x, while the centripetal seed is given by point $(x_1+\epsilon,y_1-\epsilon)$ and direction of decreasing x.

Initially, all endpoints are placed in a pool and are untagged. They are extracted one by one from the pool, to generate all the nontrivial circuits. In the course of this construction an endpoint v of a segment ℓ is tagged with the label OUT if we traverse ℓ from v, while it is tagged with the label IN when we reach v along ℓ. Whenever an endpoint is tagged with both labels OUT and IN, it is excised from the pool of endpoints. Obviously, whenever the currently considered endpoint is tagged OUT only the centripetal seed is to be used; similarly, when it is tagged IN only the centrifugal seed is to be used. The process is concluded when the pool becomes empty.

In summary, since there are O(n) arcs in the collection of nontrivial circuits and the construction of one such arc can be done in time O(logn), the set of nontrivial circuits can be constructed in time O(nlogn), including the preprocessing required to construct the search trees of the adjacency maps (see Section 2).

The construction of the nontrivial contour, however, is not yet complete. Indeed, we must still link together different circuits forming the boundary of the same region. Nontrivial region boundaries consisting of more than one circuit are of two types (see figure 7):

either a collection of clockwise nontrivial circuits (the unique external region) (figure 7a), or a collection of clockwise nontrivial circuits enclosed by a single counterclockwise circuits, either nontrivial or trivial (a rectangle) (figure 7b,c); obviously the enclosing trivial circuit is not generated by the process described above.

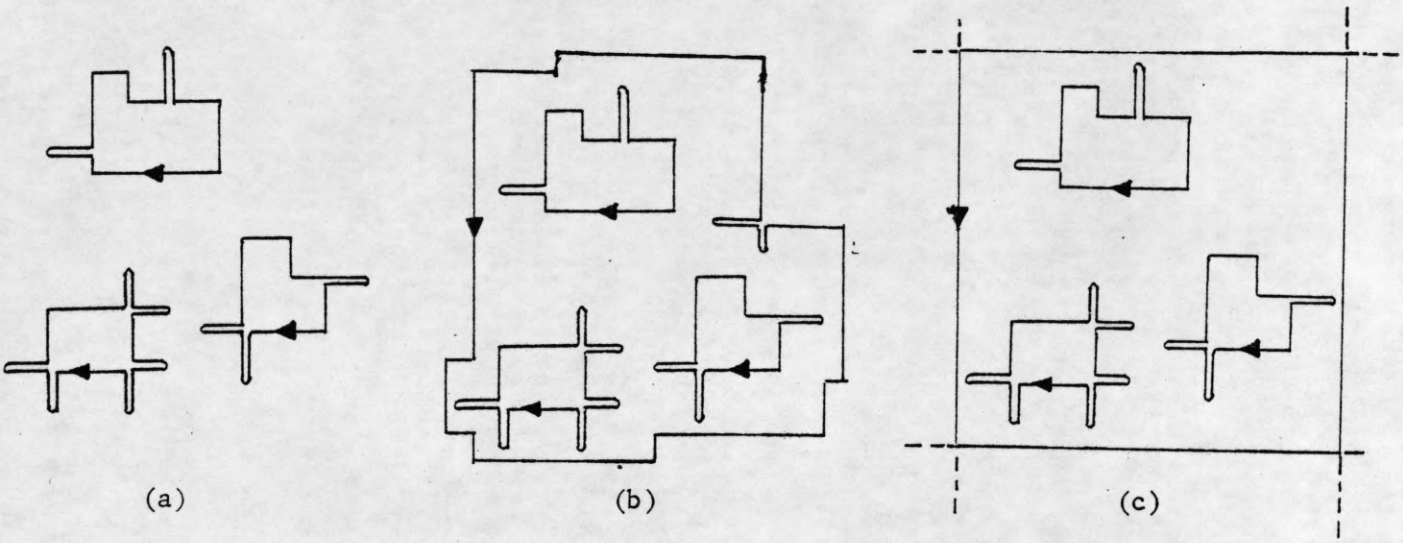


Figure 7. Types of boundaries of nontrivial regions

The above task can be carried out as follows. Each arc of a nontrivial circuit is a portion of an original segment; it is now useful to construct a map which for each point on a segment gives the arc, or arcs, which contain that point. Specifically, extending the set of arcs of nontrivial circuits with the empty arc Λ, for each point on a segment the points-of-segments → arcs map provides two arcs - one or both possibly empty - with opposite directions and containing that point. This map is easily constructed by scanning the set of arcs of the nontrivial circuits and by distributing over the set of segments the vertices which are origins and termini of arcs; next, for each segment, the vertices lying on it are organized as a search tree. The construction of this map can be accomplished in time $O(n\log n)$ in a straightforward manner.

Next we label the clockwise nontrivial circuits, by assigning the same distinguishing label to all arcs of a given circuit; moreover in each circuit a vertex, say,with lowest ordinate, is selected as the "representative" of the circuit.

At this point, region boundaries can be assembled together by a UNION-FIND approach. The set of nontrivial clockwise circuits is scanned in order of increasing ordinate of their representative vertices. For each such vertex v we seek in the vertical adjacency map its closest horizontal segment below it (this corresponds to drawing a vertical line passing through v and seeking a closest intercept p). If no such segment exists, then the circuit being considered belongs to the unbounded region. Otherwise, using the previously constructed map, we locate point p in the found segment. Now, two cases are possible: p belongs to an arc e of a nontrivial circuit C or otherwise. In the first case , "FIND" consists in obtaining the label of (the circuit containing) e, and "UNION" consists in identifying the labels of the two circuits under consideration (notice that this is correct, irrespective of whether e belongs to a clockwise or a counterclockwise nontrivial circuit); in the second case, e is an arc of a trivial enclosing rectangle (figure 7c), which can therefore be constructed in a straightforward manner, and also added to the points-of-segments → arcs map. Referring to the properties of the adjacency maps, of the standard UNION-FIND techniques [9], and to the fact that there are O(n) nontrivial circuits, it is easily realized that the just described taks can be completed in time O(nlogn).

3.2 EXTERNAL CONTOUR. As we noted, the method to obtain the nontrivial contour presented above implicitly produces the contour of the external region, i.e., the external contour. However, some efficiency can be gained by simplifying the process of linking nontrivial circuits. Specifically, the "UNION" task is omitted any time the labels to be identified are not those of the external region. Obviously, this minute modification does not change the order of the running time. In addition, the presented method is entirely applicable to the determination of both external and nontrivial contour of a collection of rectangles (a special case). It is interesting to contrast this method with the one presented in a related paper [10] to obtain the boundary of a union of n rectangles, not just the nontrivial boundary. In that case, if t is the total number of arcs in the boundary, the algorithm proposed in [10] runs in time $O(n\log n + t \log(2n^2/t))$.

3.3 POINT-LOCATION. Assume that the nontrivial contour of M be available. We assign to each nontrivial region a distinguishing label and associate with each arc of a nontrivial circuit the label of the region lying to the left of it. This labeling is completed in time O(n). Assume also that the map points-of-segments → arcs be available.

Location of a target point p in M can be carried out in the following manner. We locate p in one of the two adjacency maps, say the HAM; this search operation returns two (possibly empty) abscissae x_1 and x_2 (with $x_1 < x_2$), which are respectively the intercepts, closest to p and on opposite sides of p, of a horizontal line through p with vertical segments. We now have one of the following cases:

(i) either x_1 or x_2 (or both) belong to an arc (this can be tested in O(logn) time with the aid of the points-of-segments → arcs map); then in constant time we obtain the name of the nontrivial region containing p.

(ii) neither x_1 nor x_2 belongs to an arc (in this case p lies in a trivial rectangle); using the HAM and the VAM we obtain in O(logn) time two pairs of segments, respectively vertical and horizontal, which define the trivial rectangle containing p.

In all cases, point location is completed in O(logn) time on data structures which use O(nlogn) space (the space needed to store the adjacency maps), although - as we noted earlier - M may contain $\Omega(n^2)$ regions

3.4 ROUTE-IN-A-MAZE. Given two points s and t in the plane, using the point-location technique just described, we can readily ascertain the existence of a route between s and t by simply verifying if both points belong to the same region of M. However, this preliminary point location is not necessary; we shall try to construct a route from s to t, and the constructions fails if and only if s and t do not lie in the same region.

In view of the present application, we slightly modify the method for the assembly of boundaries of nontrivial regions described in Section 3.1. Specifically, we keep track of the "history" of label-identifications by setting up a directed forest F, whose nodes correspond to circuits of M and whose arcs correspond to label-identifications (the arc is directed from the node of the current circuit to that of a previously considered circuit): thus, in F there is a tree for each nontrivial region. With this information at our disposal, we locate in the vertical adjacency map the circuits C(s) and C(t) which are respectively adjacent to s and t from below. Next in F we determine the roots of the trees to which the nodes corresponding to C(s) and C(t) belong: if these roots are distinct, then there is no route

between s and t, otherwise we obtain the two paths from the two nodes to the common root. These two paths contain all the information necessary for constructing the route: indeed, starting from s, we use a vertical edge to reach C(s) and proceed on C(s) to its "representative" vertex (lowermost), use another vertical edge to reach another circuit (specified by one of the paths in F), and so on in a straightforward manner. This construction is done in time O(nlogn).

An interesting related problem — not be be discussed in this paper — is the construction of a shortest route between two points under the L_1-metric.

4. Performance Analysis

We observe that EXTERNAL-CONTOUR is transformable in time O(n) to NONTRIVIAL-CONTOUR: in fact no additional operation is needed on the input, but the output of the latter must be inspected, in O(n) time, to extract the external contour from the nontrivial contour.

We now show that sorting of n numbers $x_1,\ldots,x_n$ can be transformed in O(n) time into EXTERNAL-CONTOUR. This transformation is quite simple and is illustrated in figure 8. Indeed, let $m' = \min_{1\le i\le n} x_i$ and $m'' = \max_{1\le i\le n} x_i$. For each x_i we construct two segments H_i and V_i, where H_i lies on the line $y = m'' - x_i$ and spans the interval $[m', x_i]$ while V_i lies on $x = x_i$ and spans the interval $[0, m'' - x_i]$. Obviously, H_i and V_i share the endpoint $(x_i, m'' - x_i)$ (also if $x_{j_1} = m'$ and $x_{j_2} = m''$ segments, H_{j_1} and V_{j_2} are empty). It is clear that from the external contour of the set of segments $\{H_1,\ldots,H_n\} \cup \{V_1,\ldots,V_n\}$ we obtain the sorted sequence of the x_i in O(n) time[4]. In the comparison tree model, sorting requires Ω(nlogn) time, whence the proposed

[4] This reduction is identical to that used in [10].

algorithm for EXTERNAL-CONTOUR and NONTRIVIAL-CONTOUR are both optimal under this model.

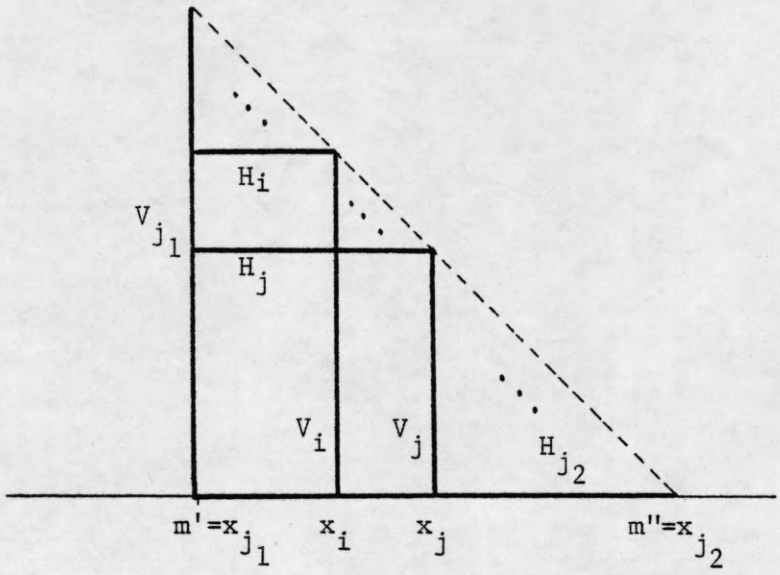


Figure 8. To show that SORTING is transformable to EXTERNAL CONTOUR.

References

1. J. L. Bentley and D. Wood,"An optimal worst-case algorithm for reporting intersections of rectangles", Carnegie-Mellon University, 1979.

2. U. Lauther,"4-Dimensional binary search trees as a means to speed up associative searches in design rule verification of integrated circuits", Journal of Design Automation and Fault-Tolerant Computing 2, 1978, pp. 241-247.

3. E. Lodi, F. Luccio, C. Mugnai, and L. Pagli,"On two dimensional data organization I", Fundamenta Informaticae, 2, 1979, pp. 211-226.

4. J. L. Bentley and T. Ottmann, "Algorithms for reporting and counting geometric intersections", IEEE Transactions on Computers, vol. 28, no. 9, pp. 643-647, September 1979.

5. F. P. Preparata, "A new approach to planar point location", Techn. Rept. ACT-11, Coordinated Science Laboratory, University of Illinois, September 1978; to appear in the SIAM Journal on Computing.

6. D. T. Lee and F. P. Preparata, "Location of a point in a planar subdivision and its applications", SIAM Journal on Computing, Vol. 6, N. 3, pp. 594-606, September 1977.

7. R. J. Lipton and R. E. Tarjan,"Applications of a planar separator theorem", Proc. of the 18th Symp. on Found. of Comp. Sci., Providence, R.I., October 1977, pp. 162-170.

8. J. L. Bentley,"Solutions to Klee's rectangle problems", unpublished notes, Carnegie-Mellon University, 1977.

9. A. V. Aho, J. E. Hopcroft, and J. D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass. 1974.

10. W. Lipski, Jr. and F. P. Preparata, "Finding the contour of a union of iso-oriented rectangles", The Journal of Algorithms, to appear.

Appendix

Since a segment endpoint can belong to at most two nontrivial circuit (this occurs when this endpoint lies in the interior of some other orthogonal segment), and there are 2n endpoints, we obtain the following straightforward lemma:

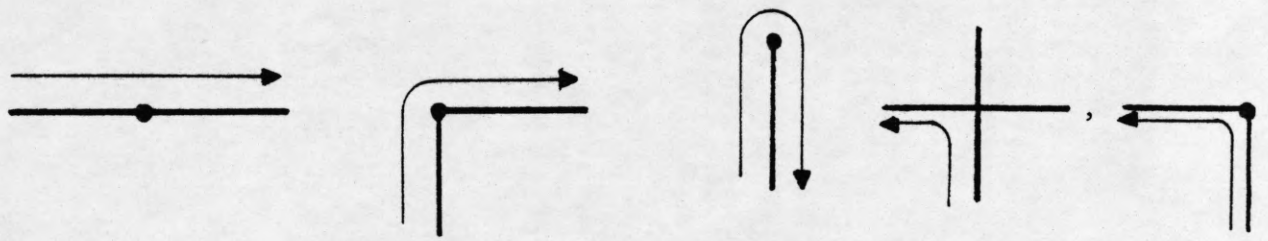
Lemma 1. The number of nontrivial circuits of M does not exceed 4n.

But a stronger statement can be proved.

Lemma 2. The total number of arcs in the nontrivial circuits of M is O(n).

Proof. We say that a vertex is of type i (i = 0,1,2,3) if the clockwise angle formed by the arcs meeting at that vertex (ordered according to the direction on the circuit) is $i\pi/2$ (see figure 9). We also let ν_i denote the number of vertices of type i on a given circuit. Then, for any circuit we have the following relation:

$$\nu_3 - \nu_1 - 2\nu_2 = \begin{cases} 4 & \text{for an internal (counterclockwise) circuit} \\ -4 & \text{for an external (clockwise) circuit} \end{cases}$$



Type 0 Type 1 Type 2 Type 3

Figure 9. Types of vertices.

Consequently, the total number ν of vertices (i.e., of arcs) is expressible as

$$\nu = \nu_0 + \nu_1 + \nu_2 + \nu_3 = \nu_0 + 2\nu_1 + 3\nu_2 \pm 4 \leq \nu_0 + 2\nu_1 + 3\nu_2 + 4$$

By summing ν over all nontrivial circuits we obtain (the summations are extended over all such circuits) for the total number of arcs t:

$$t \leq \Sigma\nu_0 + 2\Sigma\nu_1 + 3\Sigma\nu_2 + 4\Sigma 1$$

$$\leq 2n + 2n + 3 \cdot 2n + 4 \cdot 4n \leq 26n.$$

□