Lokalizacja punktu w przestrzeni dwuwymiarowej – metoda trapezowa

Geometria obliczeniowa – Kamil Borowiec

Opis zagadnienia

Dany jest obszar z podziałem poligonowym. Zadawany jest punkt P na płaszczyźnie. Należy zaimplementować algorytm lokalizacji punktu metodą trapezową, który odpowie na pytanie, w którym elemencie znajduje się dany punkt. Program powinien w sposób graficzny prezentować etapy algorytmu dla wybranych przykładów (w celu objaśnienia działania algorytmu). Program ma służyć jako narzędzie dydaktyczne do objaśnienia działania algorytmu.

Lokalizowania punktu 2D

Dane wejściowe:

- Punkt 2D
- Obszar poligonowy podziału płaszczyzny

Dane wyjściowe :

- Wielokąt zawierający zadany punkt

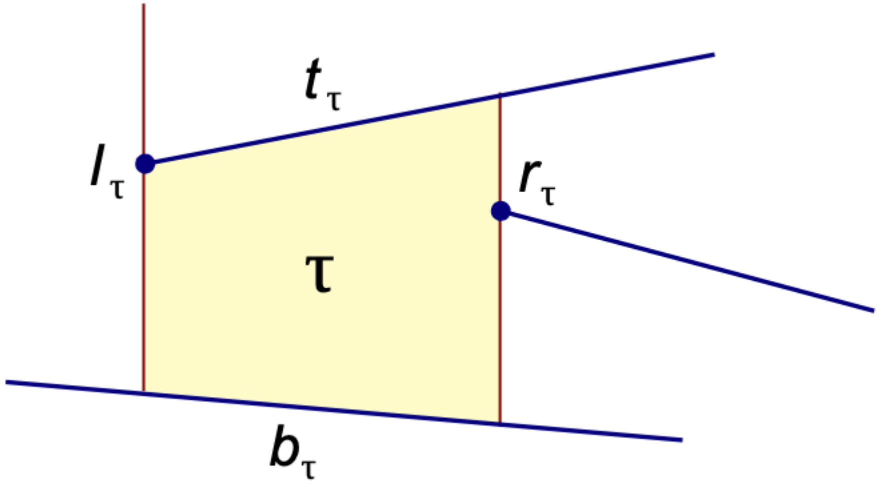
Tworzenie trapezoid map - algorytm

1. Determine a bounding box, that contains every segment in S. Initialize the Trapezoidal Map and the corresponding searchstructure for it.
2. Compute a random permutation of the elements in S.
3. for i = 1 to n do
4. Find the set of trapezoids that are intersected when the segment s_i is added.
5. Remove the intersected trapezoids that arise because of the insertion of s_i .
6. Remove the leaves for the intersected trapezoids from the search structure and create leaves for the new trapezoids. Link the created leaves to the existing nodes in the search structure.

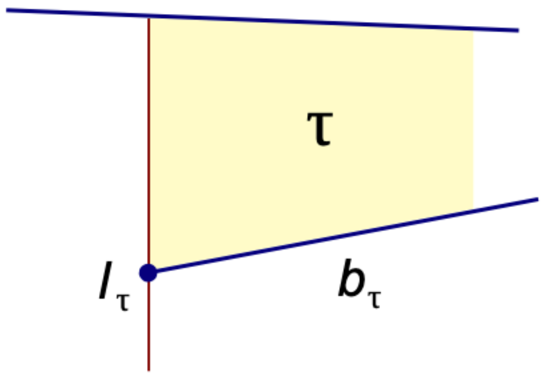
Definicja trapezu na potrzeby naszego algorytmu

Trapezoid τ :

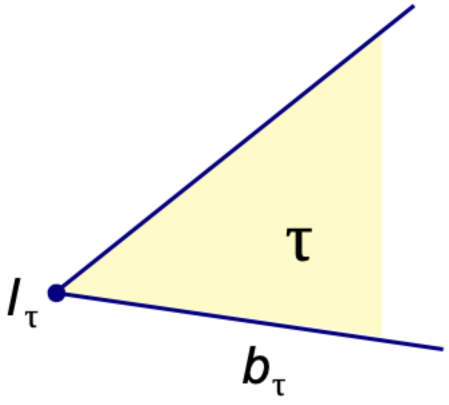
- Top edge: t_τ
- Bottom edge: b_τ
- Left vertex: l_τ
- Right vertex: r_τ
- (possibly horizontal walls / vertices of the bounding box)



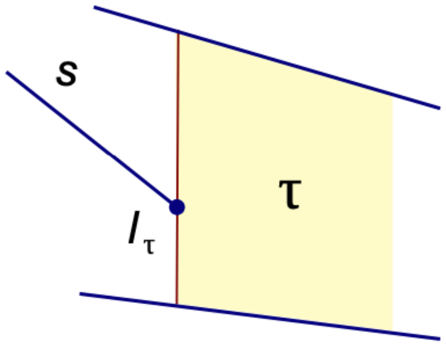
Przypadki lewego boku



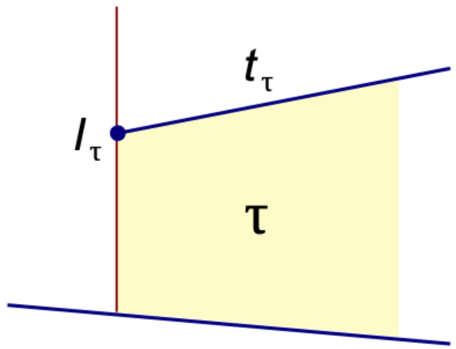
upper vertical extension
of original endpoint l_τ



meeting point l_τ
of two original segments



whole vertical extension
of original endpoint l_τ



lower vertical extension
of original endpoint l_τ

Algorithm steps

- Initially the map contains only the *bounding box*
- $\rightarrow$ one-node DAG
- For each edge $e \in S$ in randomized order...
  - remove the trapezoids $\mathcal{T}_1, \mathcal{T}_2, \ldots \mathcal{T}_k$ in *conflict* with e
  - replace them with the new trapezoids determined by e
  - remove the DAG's leaves linked to $\mathcal{T}_1, \mathcal{T}_2, \ldots \mathcal{T}_k$
  - replace these leaves with x-/y-nodes as appropriate
  - create and link leaves for the new trapezoids

Finding trapezoids in conflict with a new edge

- Point location of e's left endpoint (current DAG)
- $\rightarrow$ leftmost trapezoid $\mathcal{T}_1$ in conflict with e
- Follow right-neighbor links from $\mathcal{T}_1$ to the trapezoid $\mathcal{T}_k$ which contains e's right endpoint (edges do not cross)
- The correct neighbor $\mathcal{T}_{i+1}$ of $\mathcal{T}_i$ is identified by testing where $r_{\mathcal{T}_i}$ lies relative to e

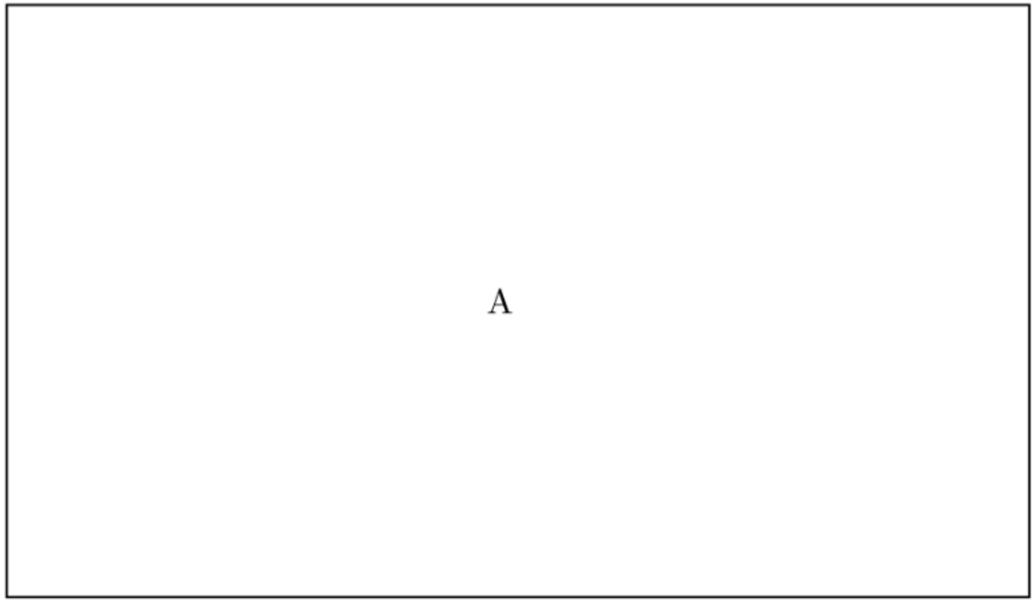
Updating the map

- $\mathcal{T}_1$ and $\mathcal{T}_k$ are partitioned in three parts (four if $\mathcal{T}_1 = \mathcal{T}_k$)
- $\mathcal{T}_2, \mathcal{T}_3, \ldots \mathcal{T}_{k-1}$ are split
- Whenever possible, the resulting trapezoids bounded by e are merged
- All operations can be done in $O(k)$ (in constant time for each involved trapezoid)

Updating the DAG

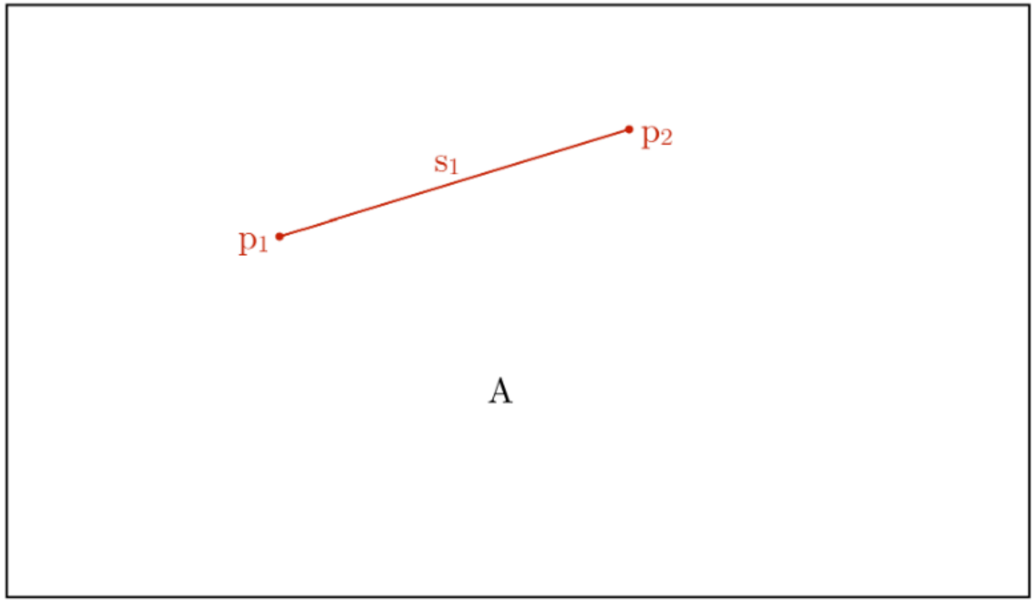
- Cross links between leaf nodes and trapezoids
- At most three new x-/y-nodes for each removed trapezoid
- Several nodes are linked to a new “merged” trapezoid
- All arrangements can be done in $O(k)$

Step 1: Initialization of Trapezoidal Map and History

TrapezoidalMap	History
	

Step 3: First segment to be added

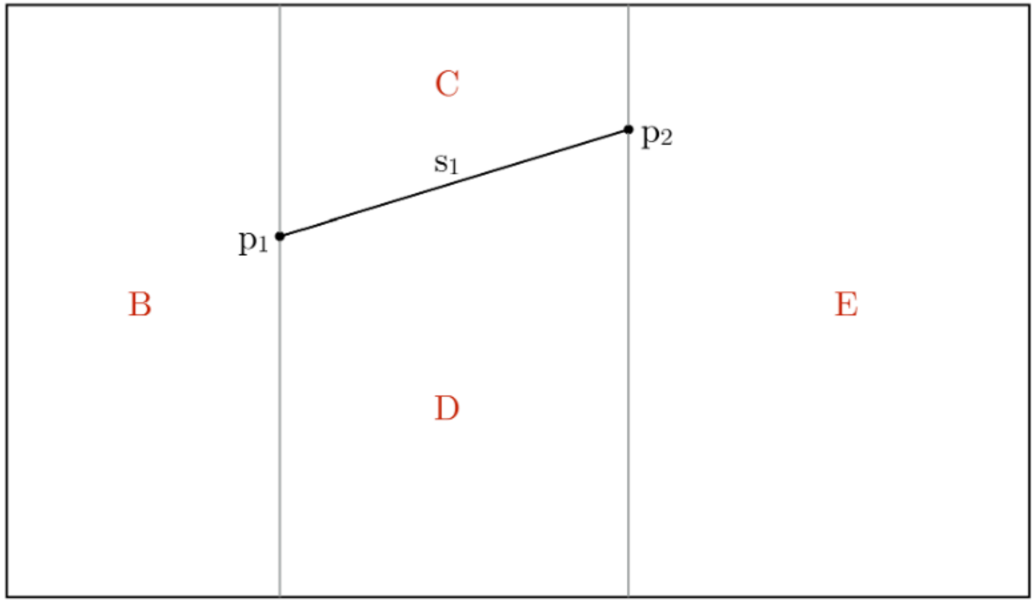
TrapezoidalMap



History

Step 5: Vertical extension of the segment's endpoints

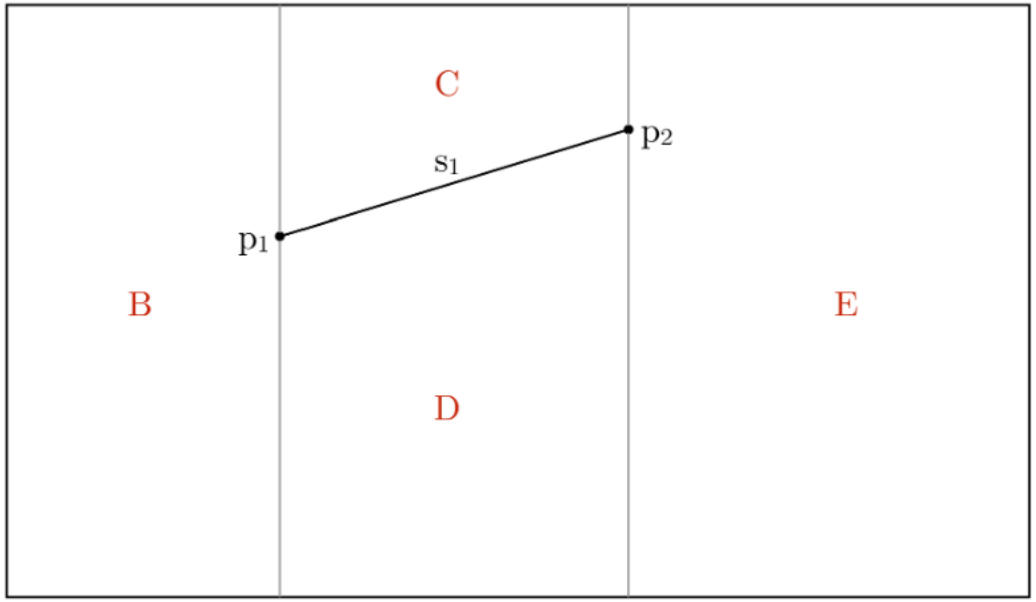
TrapezoidalMap



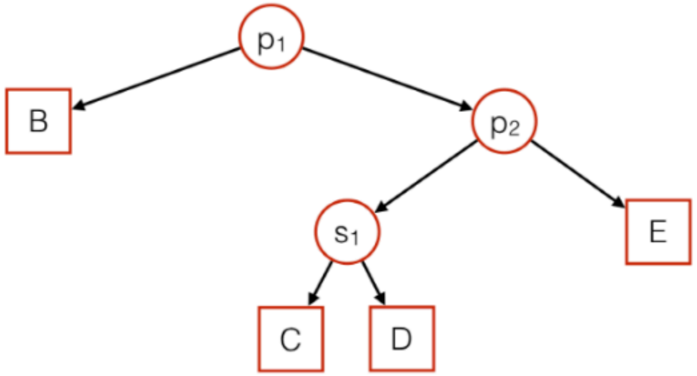
History

Step 6: The History is updated with new nodes for the new trapzoids

TrapezoidalMap

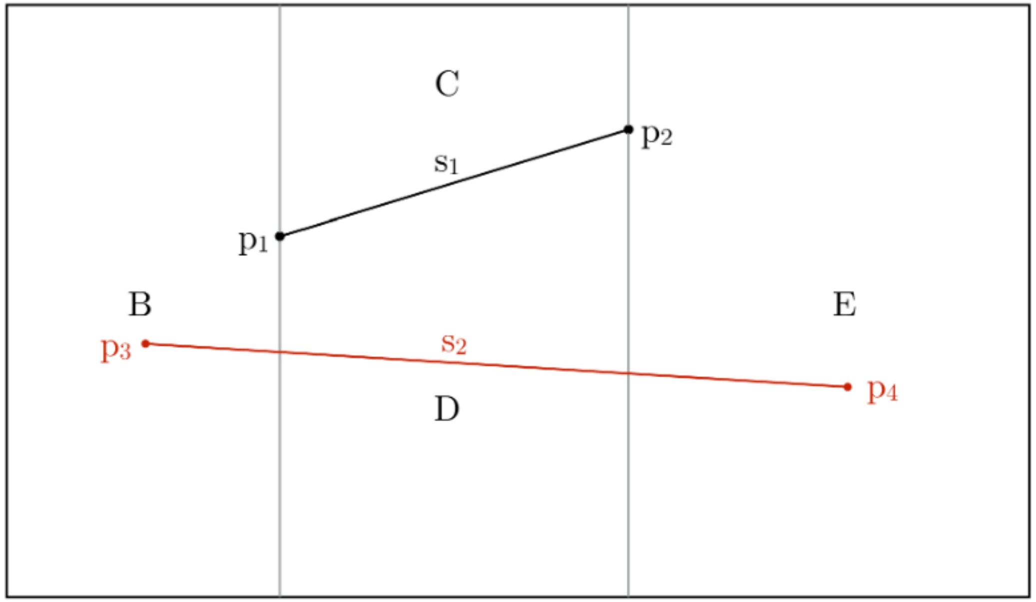


History

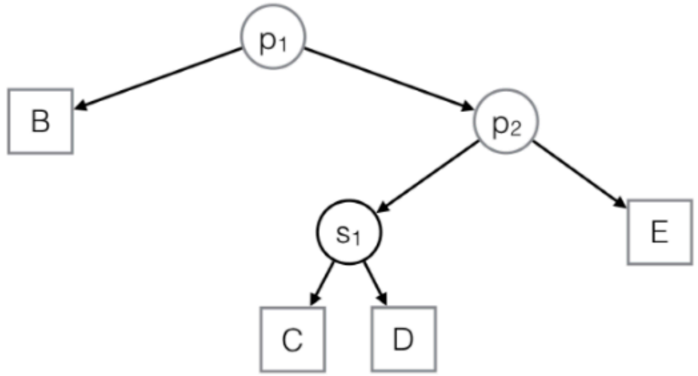


Step 3: Second segment to be added

TrapezoidalMap

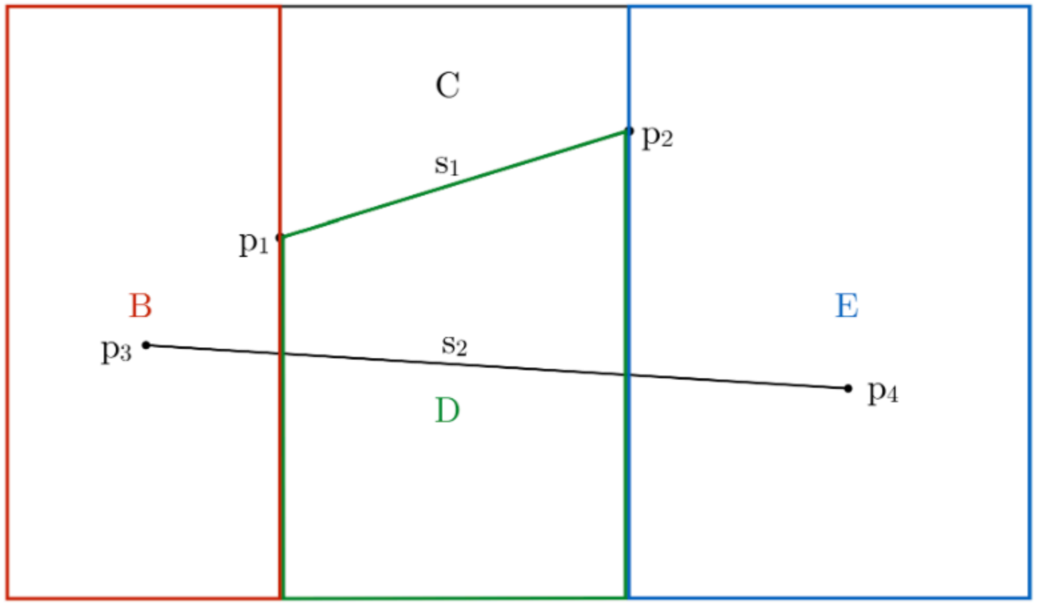


History

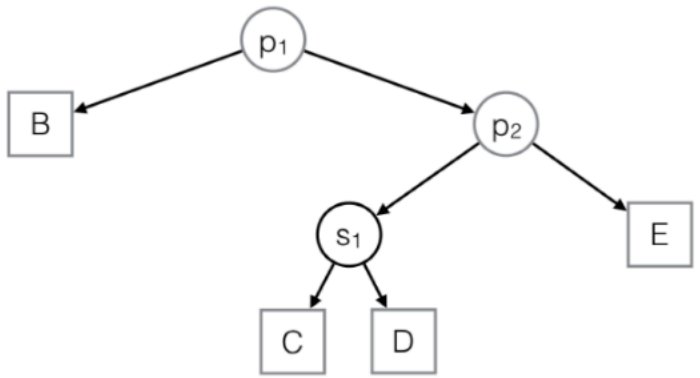


Step 4: Intersected trapezoids are highlighted

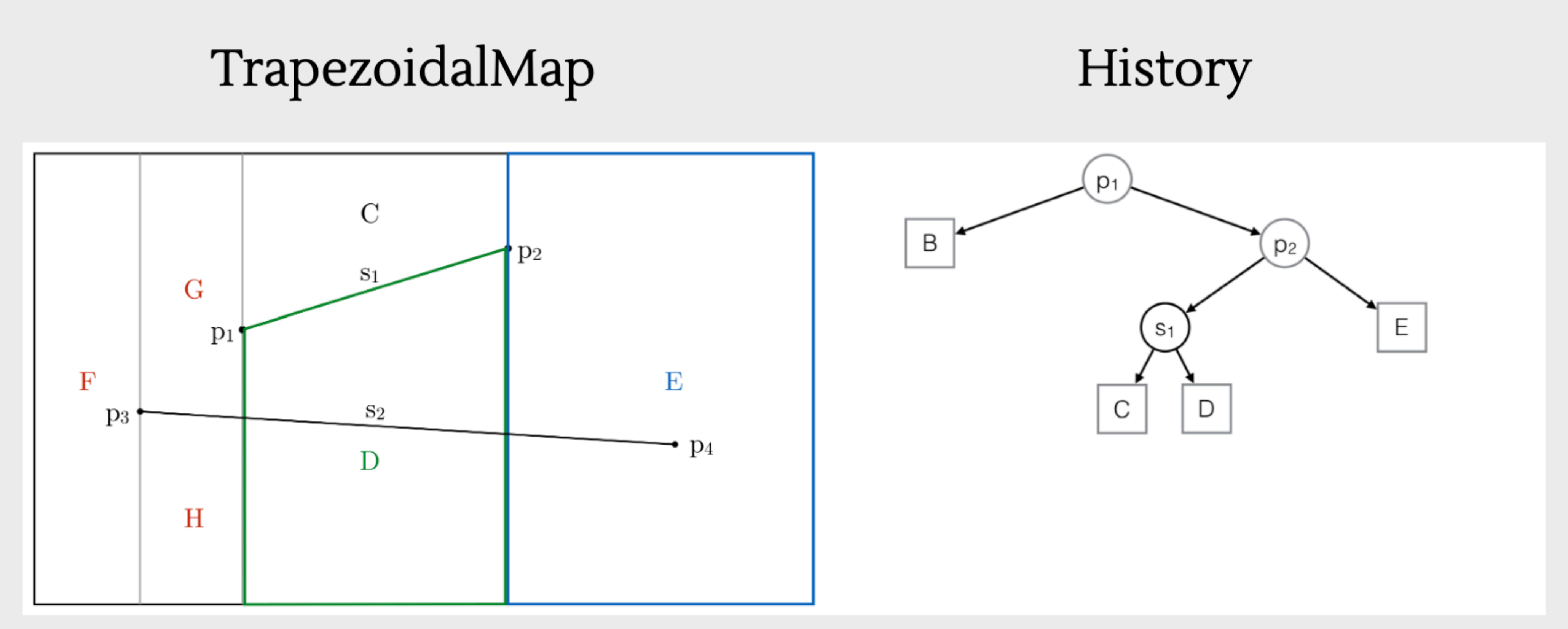
TrapezoidalMap



History

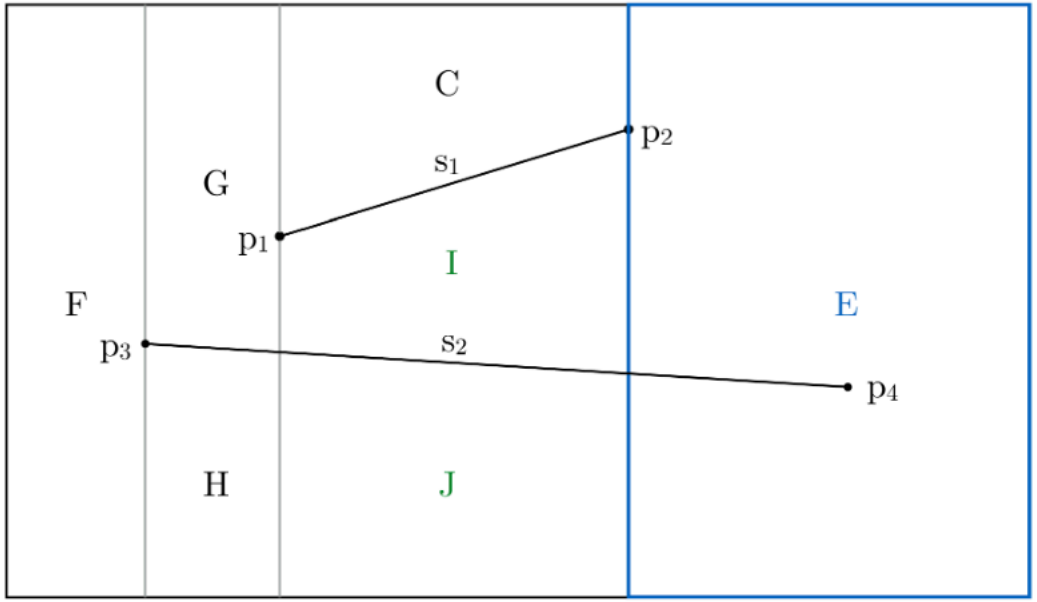


Step 5: The first intersected trapezoid is replaced by three new trapezoids

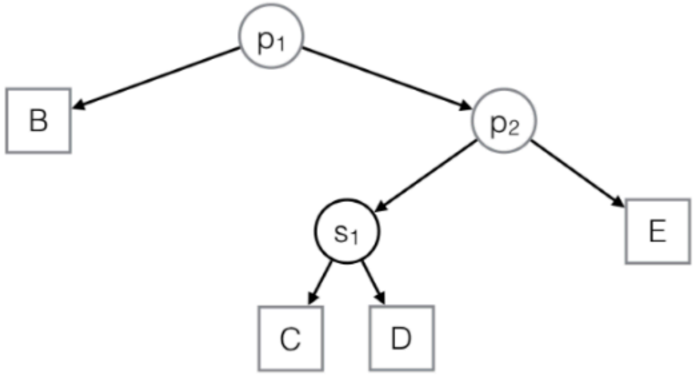


Step 5: The second intersected trapezoid is replaced by two new trapezoids

TrapezoidalMap

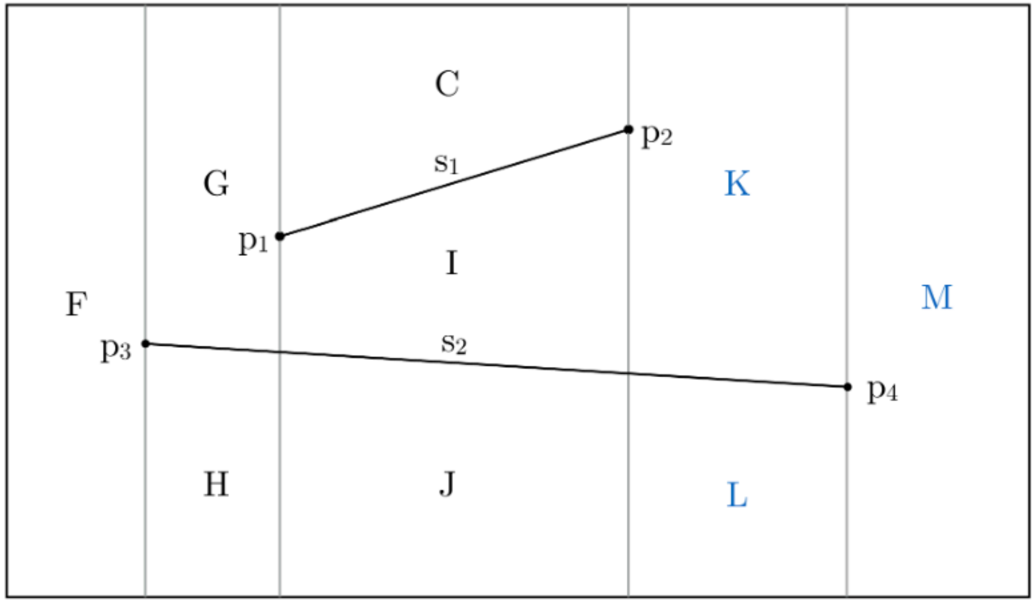


History

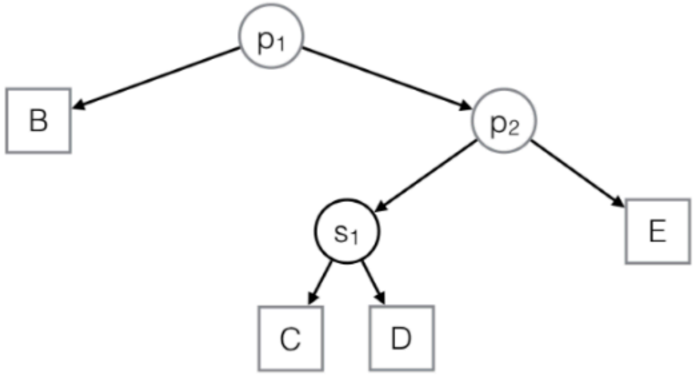


Step 5: The third intersected trapezoid is replaced by three new trapezoids

TrapezoidalMap

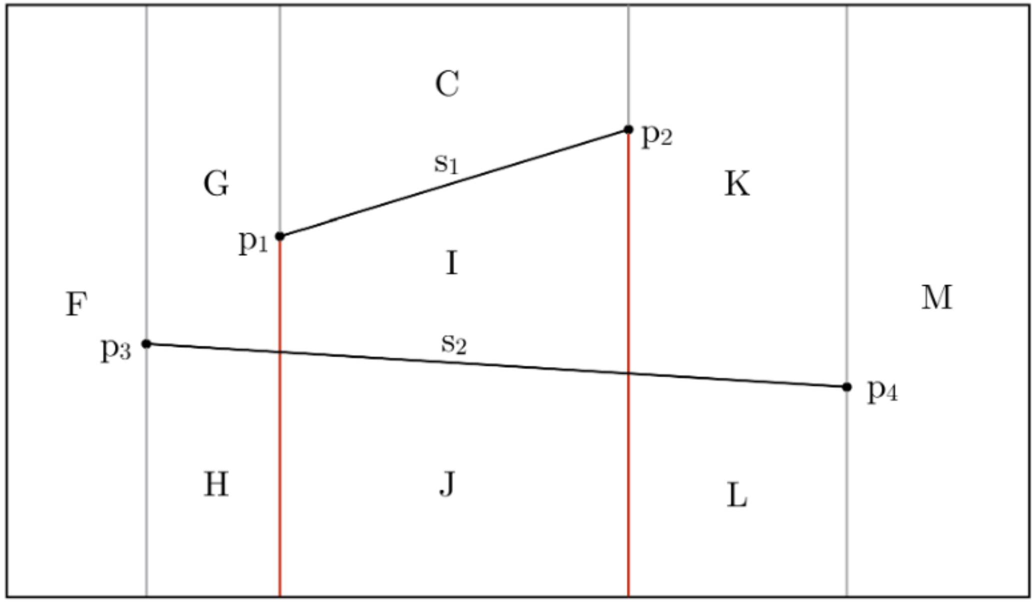


History

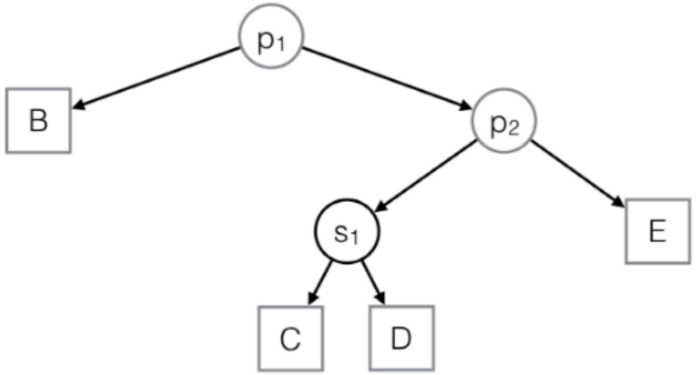


Step 5: Highlighted vertical extensions of the first segment are too long

TrapezoidalMap

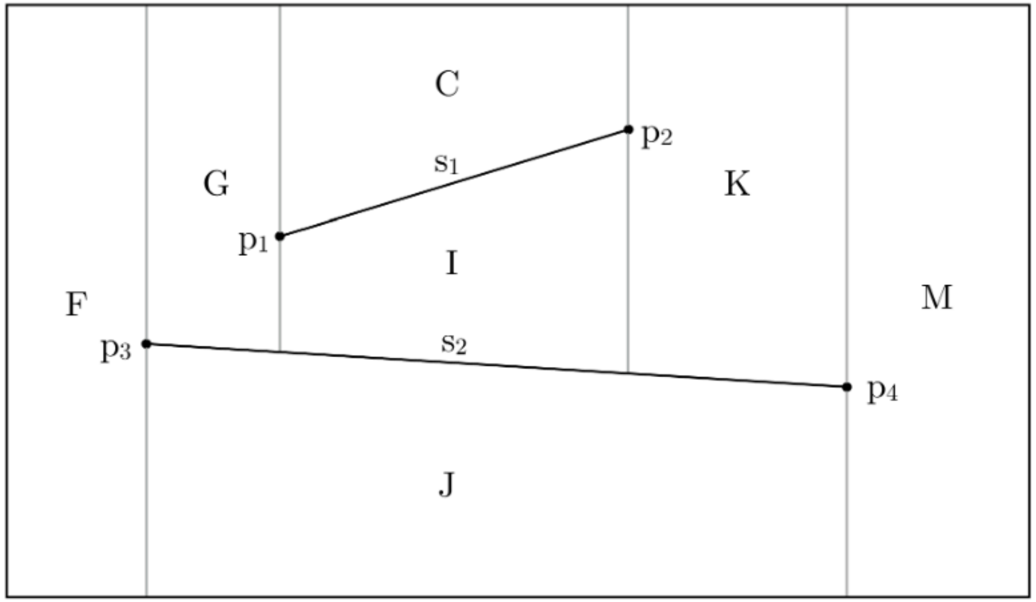


History

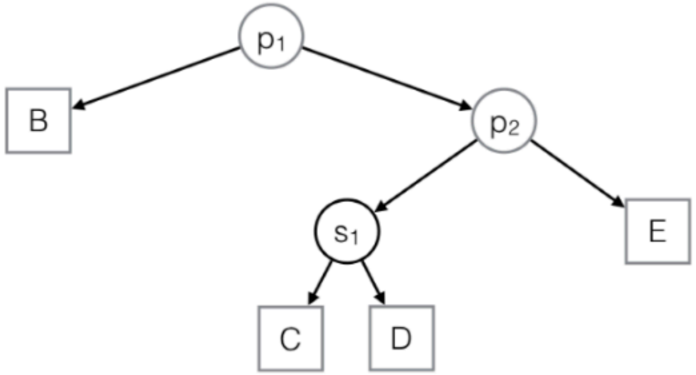


Step 5: Shortening of the vertical extensions

TrapezoidalMap

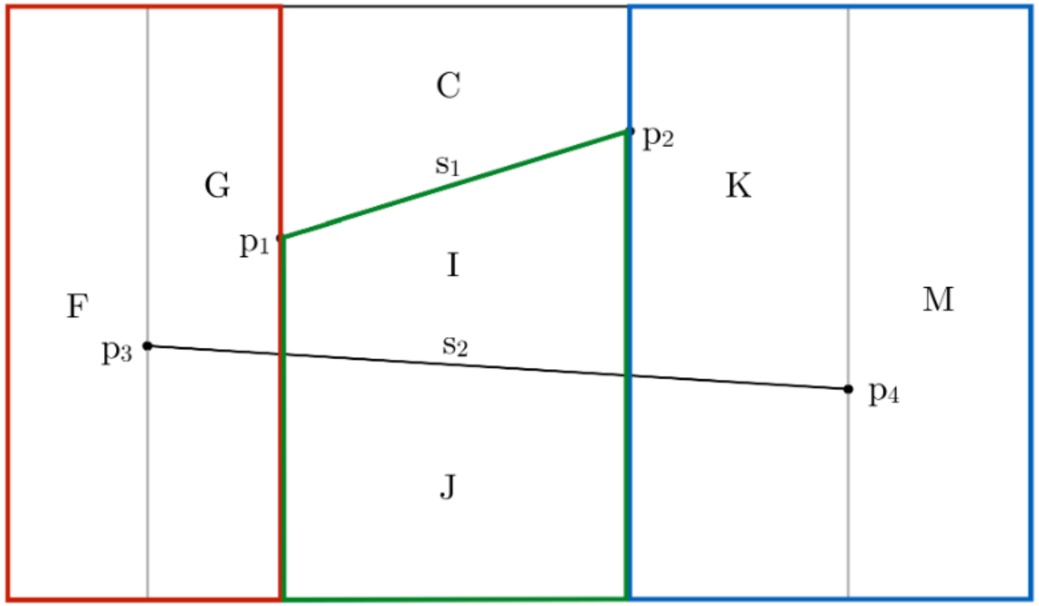


History

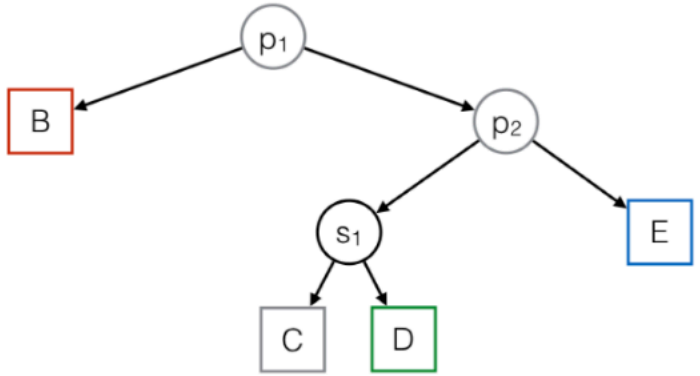


Intersected trapezoids are highlighted in the Trapezoidal Map and the History

TrapezoidalMap

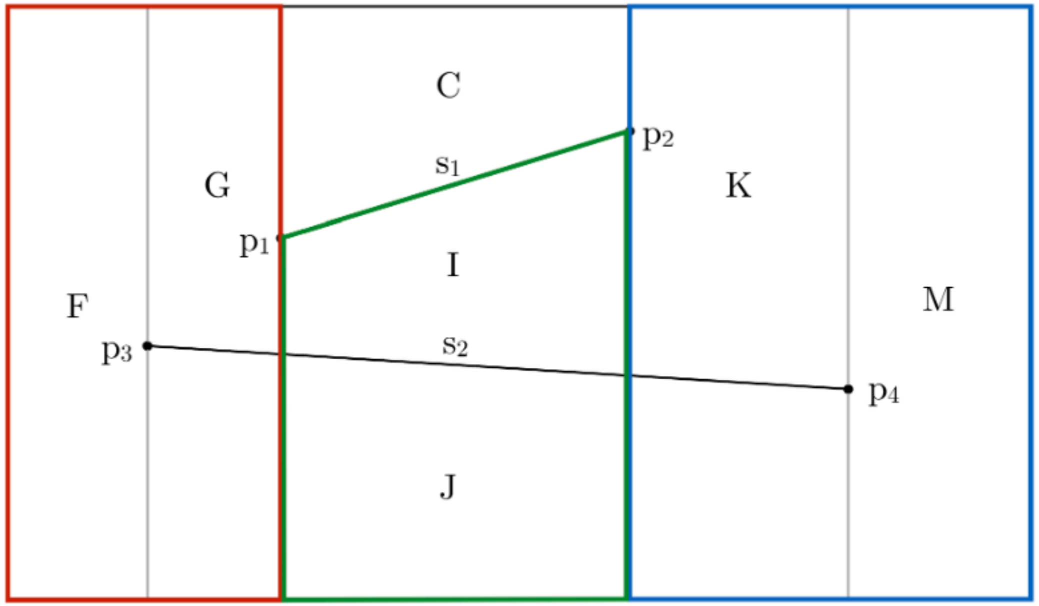


History

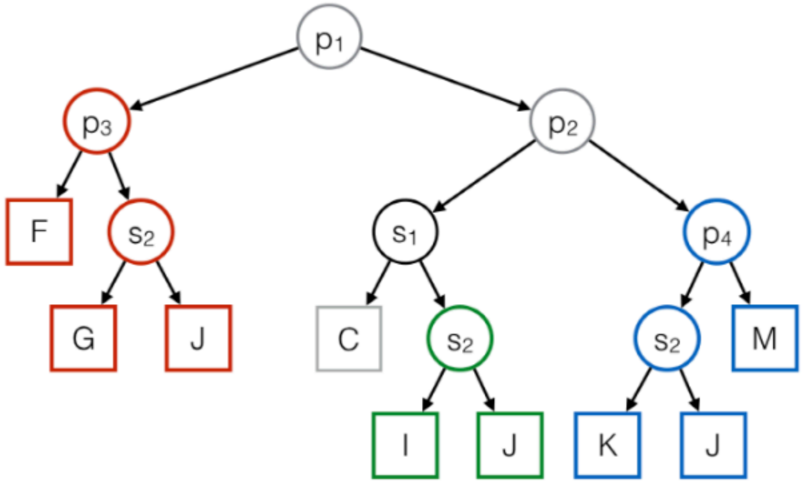


Step 6: Nodes of intersected trapezoids are replaced by new ones

TrapezoidalMap

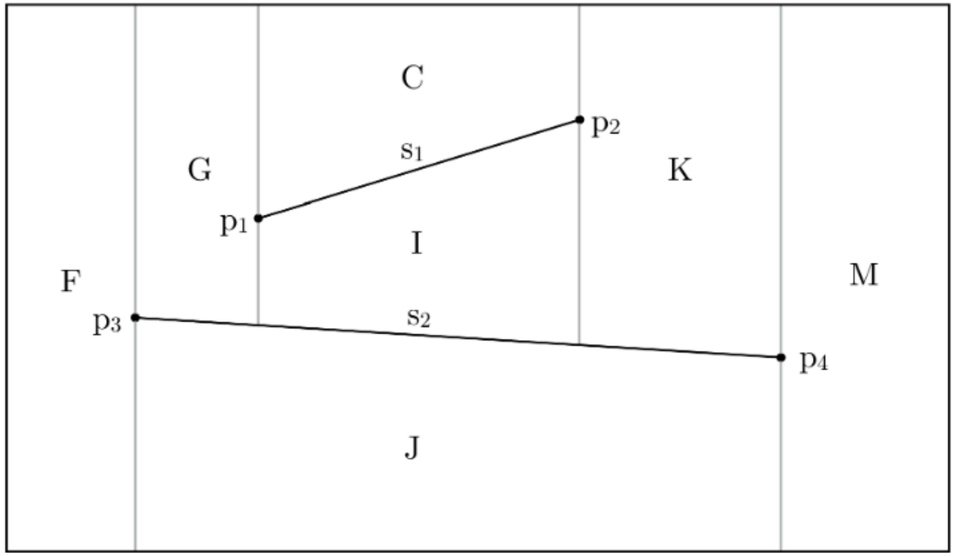


History

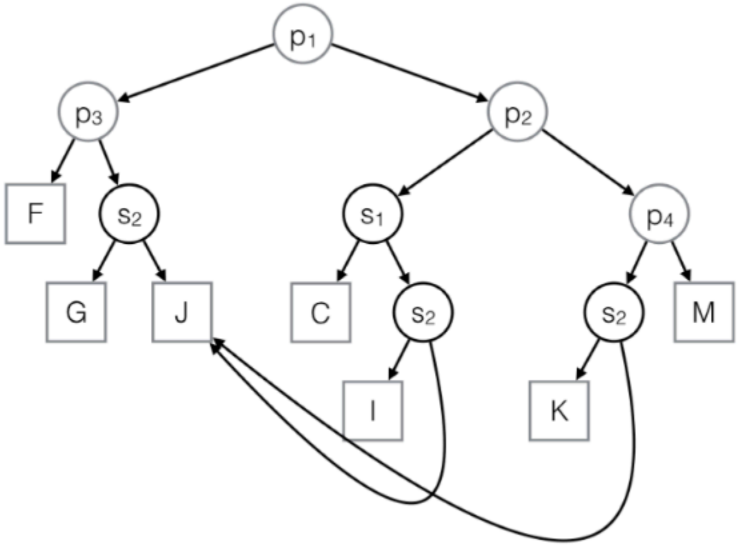


Same trapezoid nodes are linked together

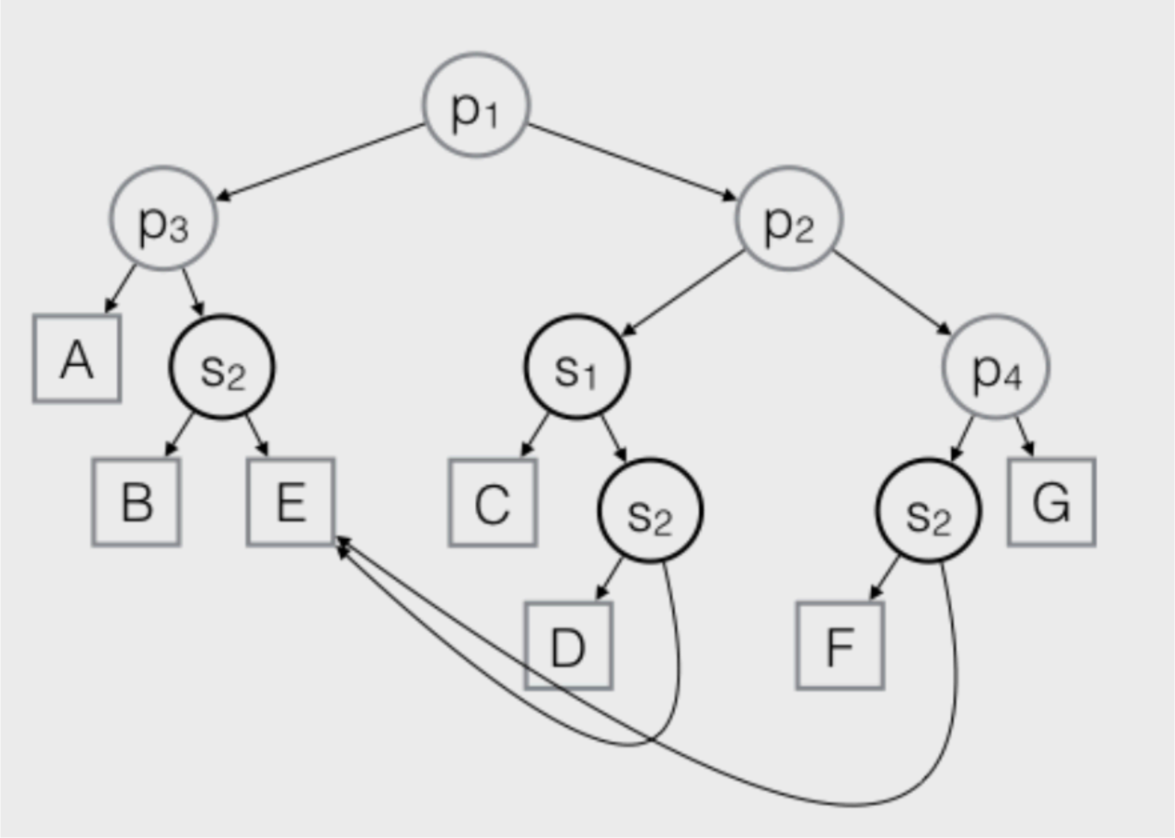
TrapezoidalMap



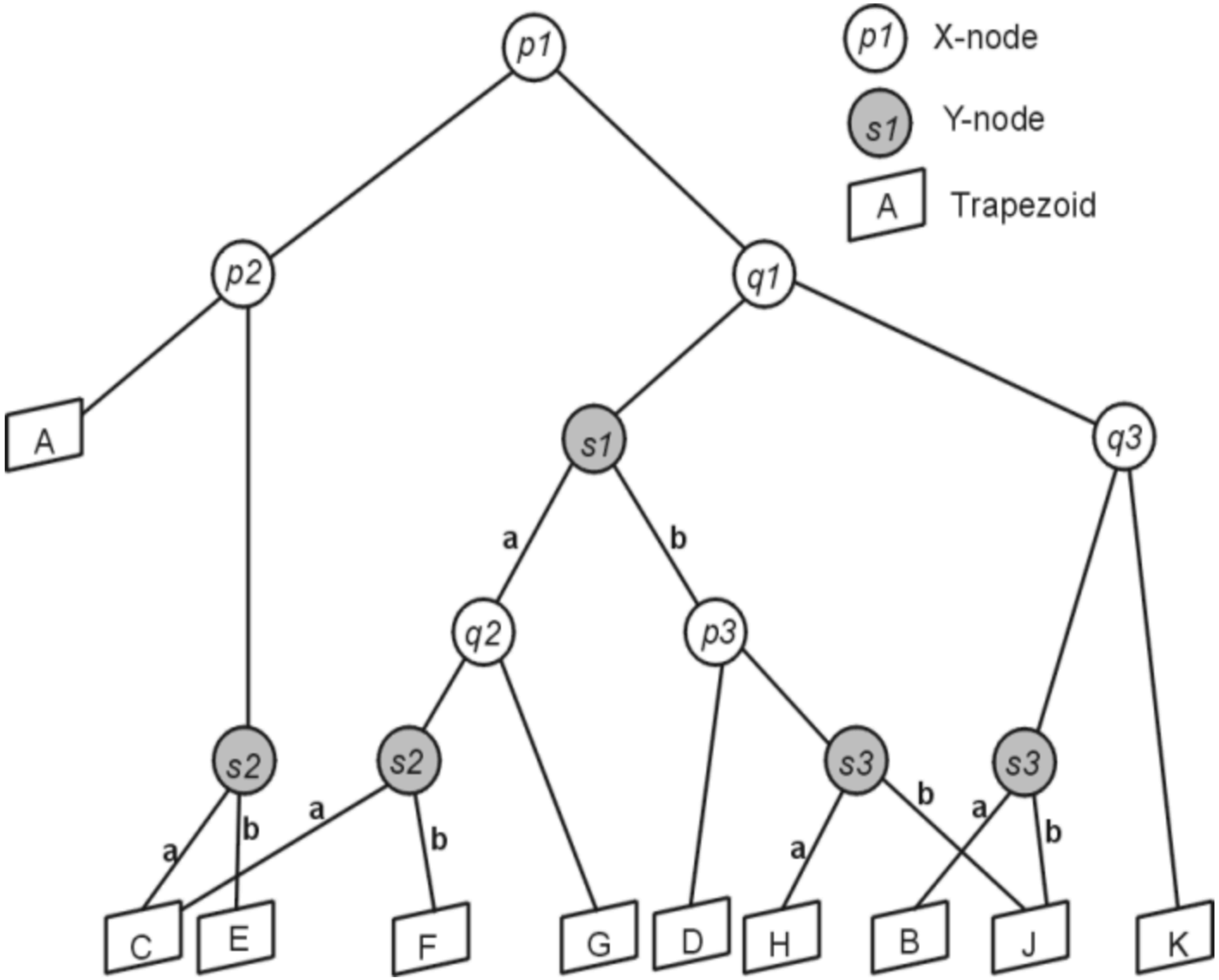
History



Wyszukiwanie wielokąta, w którym znajduje się punkt wg struktury history



- X-nodes: Inner node storing an endpoint of a segment
- Y-nodes: Inner node storing a segment
- Leaf-nodes: Representing trapezoids



Algorithm **QueryTrapezoidMap(** ***D*****,** ***n*****,** ***p*****)**

Input: *T* is the trapezoid map search structure, *n* is a node in the search structure and *p* is a query point.

Output: A pointer to the node in *D* for the trapezoid containing the point *p*.

1. **if** (n is a Trapezoid Node)
   return *n*;
2. **if** (n is an X-node)
   a. **if** the x-coordinate of p the is less than the x-coordinate of the point stored at this node then
      return *QueryTrapezoidMap(T, leftChild(n), p)*.
   else
      return *QueryTrapezoidMap(T, rightChild(n), p)*
3. **if** (n is a Y-node)
   a. **if** p is above the segment stored at n then
      return *QueryTrapezoidMap(T, aboveChild(n), p)*.
   else
      return *QueryTrapezoidMap(T, belowChild(n), p)*

Zalety podejścia

- Otrzymujemy strukturę trapezoid map. Złożoność czasowa operacji jej budowy wynosi O(n log n)
- Lokacja punktu dzięki strukturze history ma złożoność czasową O(log n)

Bibliografia

https://www.ti.inf.ethz.ch/ew/lehre/CG12/lecture/Chapter%209.pdf

http://cglab.ca/~cdillaba/comp5008/trapezoid.html

https://users.dimi.uniud.it/~claudio.mirolo/teaching/geom_comput/presentations/trapezoidal_map.pdf

https://janrollmann.de/projects/thesis/

https://isotropic.org/papers/point-location.pdf