Knowing Where You Are

- Given a map and a query point q specified by its coordinates, find the region of the map containing q.
- A map can be treated as a subdivision of the plane into regions, or planar subdivision.
- A map can be stored electronically for query preprocessing to answer point location query fast and display the map interactively.

Planar Point Location

- Let S be a planar subdivision with n edges. The planar point location problem is to store S in such a way that we can answer: given a query point q, report the face f of S that contains q. If q lies on an edge or a vertex, report so.

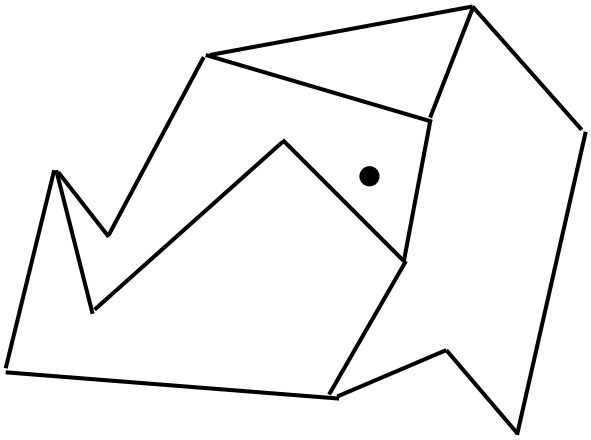
$\Rightarrow$ Partition the plane into vertical slabs. Store the x-coordinates of the vertices in the sorted order in an array. This makes it possible to determine in $O(\log n)$ time the slab contains a query point q.

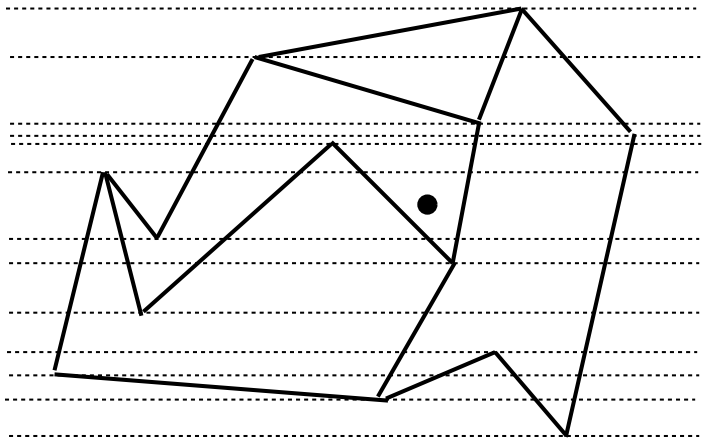
Location in Planar Subdivisions - Lower Bounds

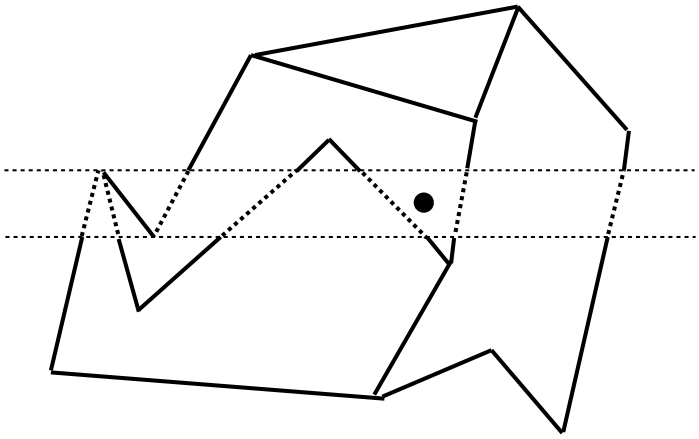
- Preprocessing $\Omega(n \log n)$
- Space $\Omega(n)$
- Query time $\Omega(\log n)$

Slab Method

- Use binary search to identify in $O(\log n)$ time the slab containing q.
- Use binary search to identify in $O(\log n)$ time the trapezoid containing q.

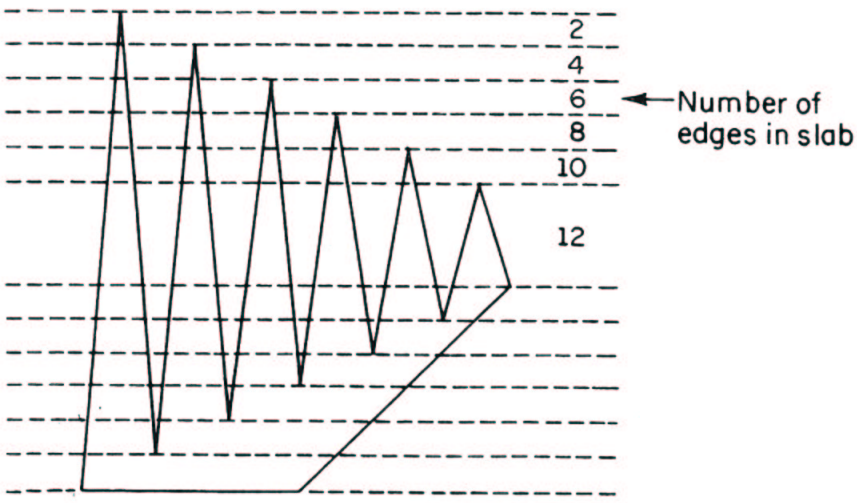






- Preprocessing: Slabs in a sorted list. Trapezoids within the same slab in a sorted list. Sorting in $O(n)$ slabs, each slab with $O(n)$ elements, takes $O(n^2 \log n)$ time.
- Time complexity can be improved to $O(n^2)$.

Nothing can be done (in this algorithm) to reduce the storage used since there exist PSLGs that need quadratic space

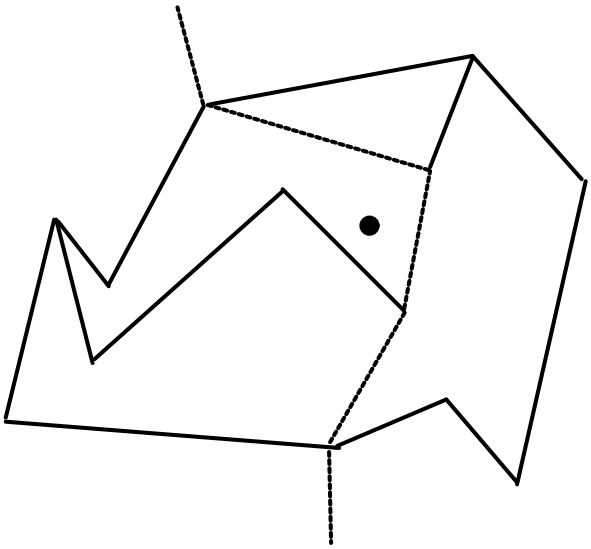



```
        procedure PREPROCESSING-FOR-PLANAR-POINT-
                  LOCATION
begin VERTEX[1:2N] := Sort the vertices of G by increasing y;
      L := ∅;
      for i := 1 until N do
        begin DELETE(B[i]);
              INSERT(A[i]);
              Output L
        end
end.
```

Thus we have

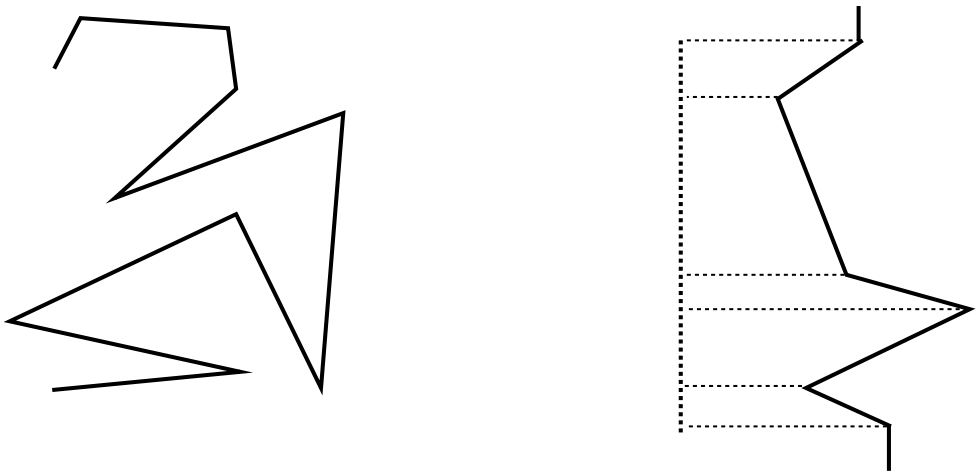
Theorem 2.4. *Point-location in an N-vertex planar subdivision can be effected in* $O(\log N)$ *time using* $O(N^2)$ *storage, given* $O(N^2)$ *preprocessing time.*

Chain Method



- How to determine a "halving" chain?
- How to decide that a query point is to the left of a chain?
- Deciding whether a point is to the left of an arbitrary chain is as difficult as deciding whether a point is inside a simple polygon.

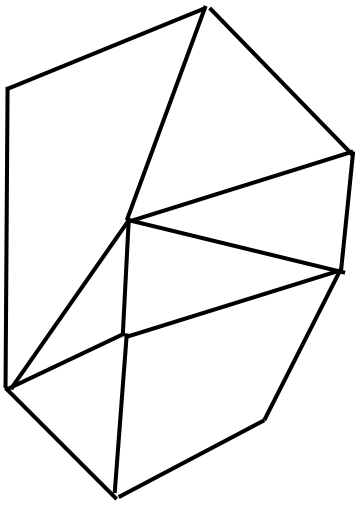
Monotone Chains

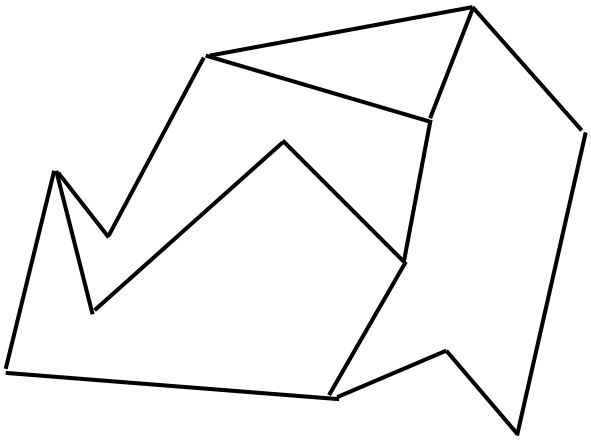


- A chain is *monotone* if there is a line so that the order of corners on the chain is preserved when projected on the line.
- It is possible to decide whether a point is to the left of a monotone chain in $O(\log n)$ time.

Chain Method - Continued

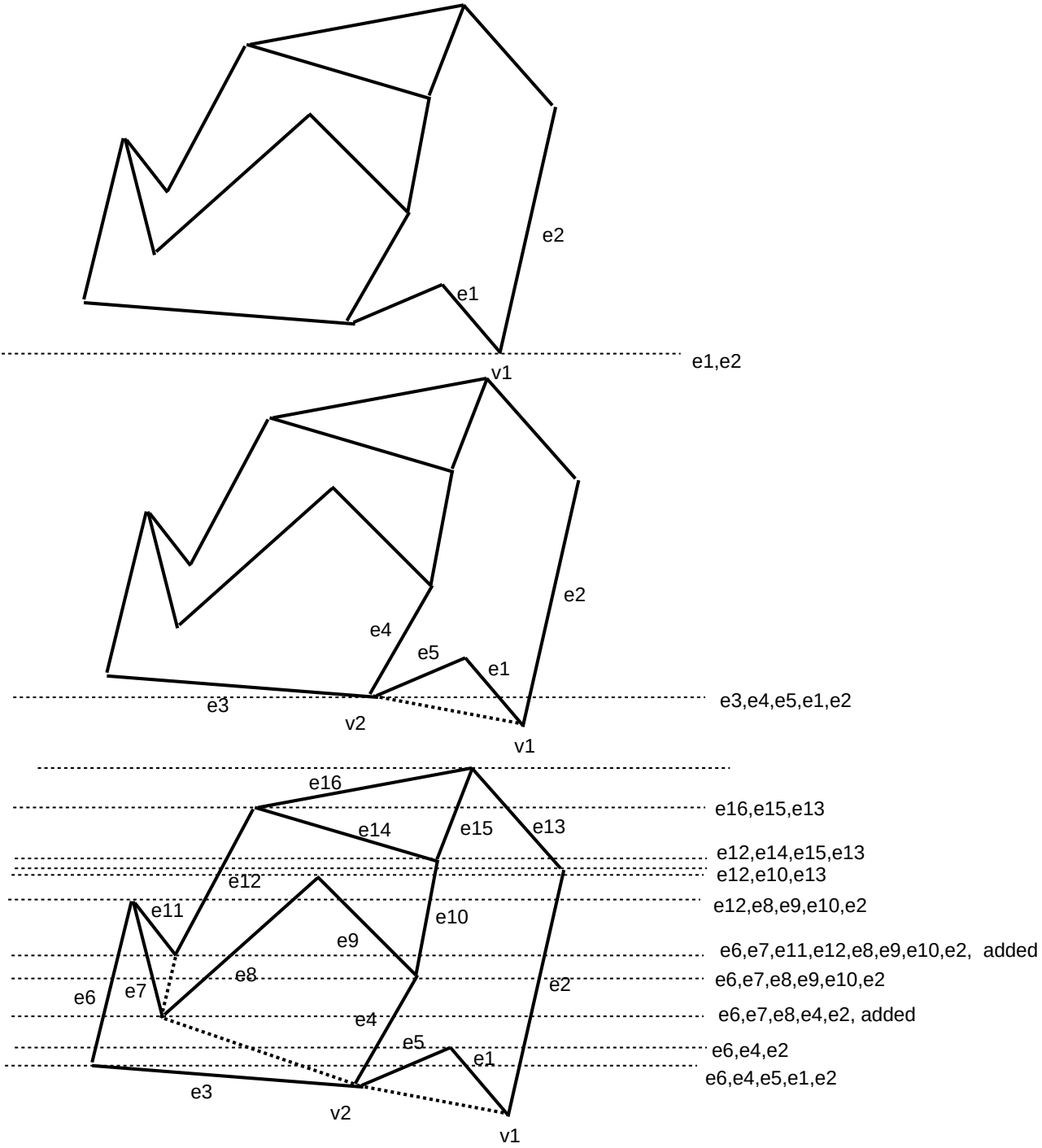
- A plane graph is *monotone* if it is possible to cover its edges by non-crossing monotone chains w.r.t. y-axis.



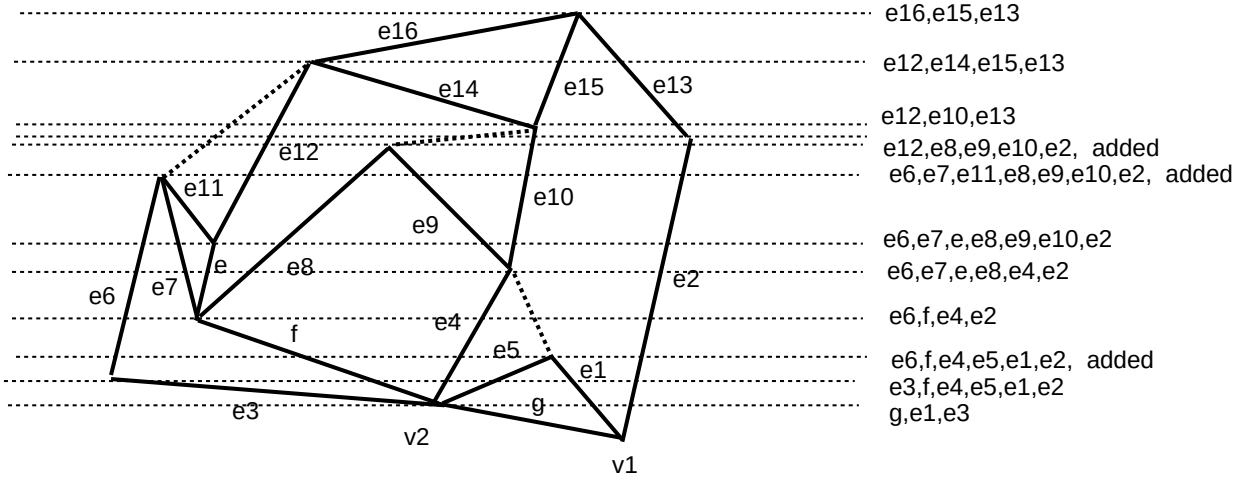


- Not all plane graphs are monotone.
- A plane graph is *regular* if each node has at least one edge coming from above and at least one from below (apart from the top and bottom node).
- Every regular graph is monotone.

Chain Method - Regularization in $O(n \log n)$ Time

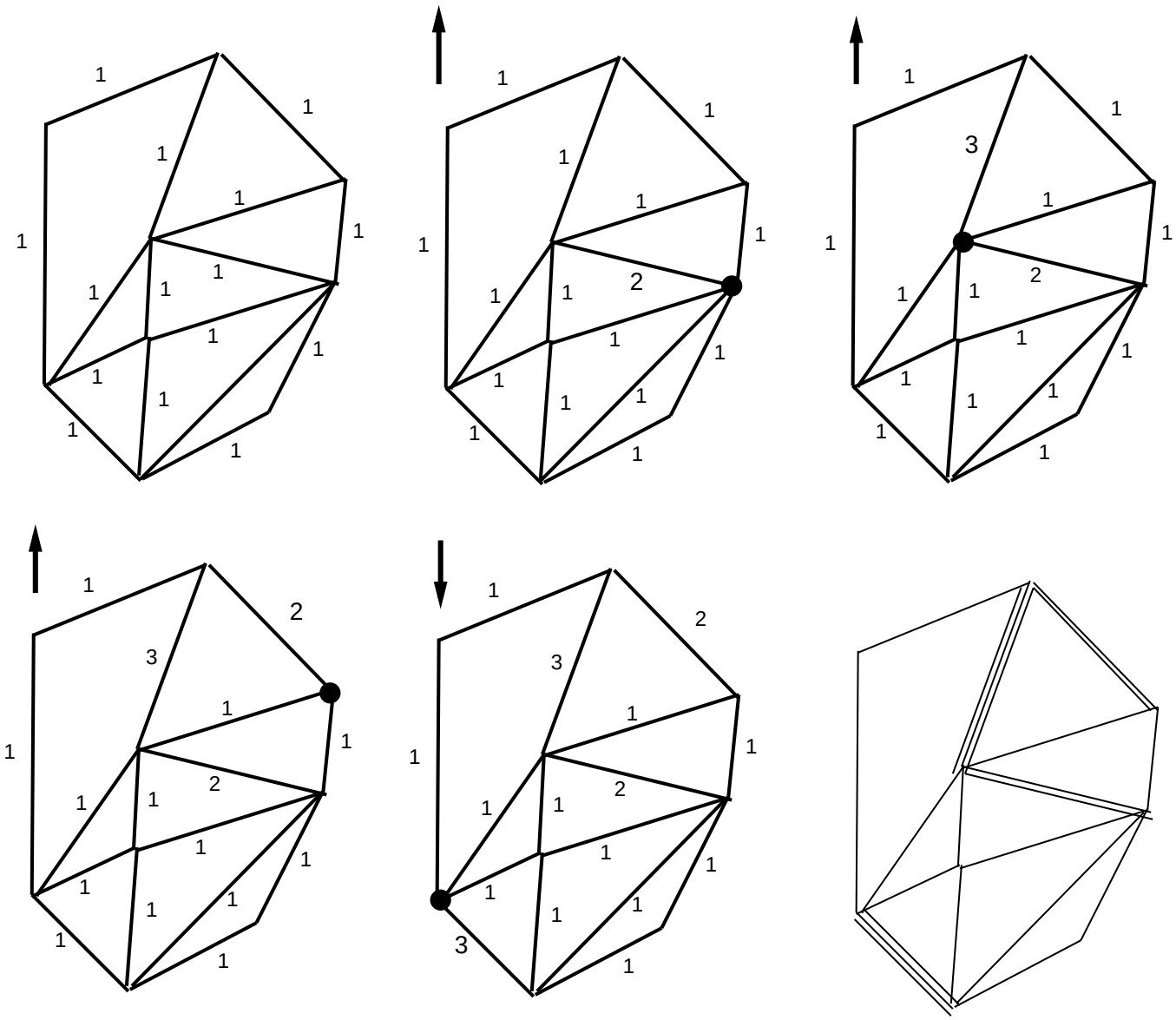


Chain Method - Regularization Continued



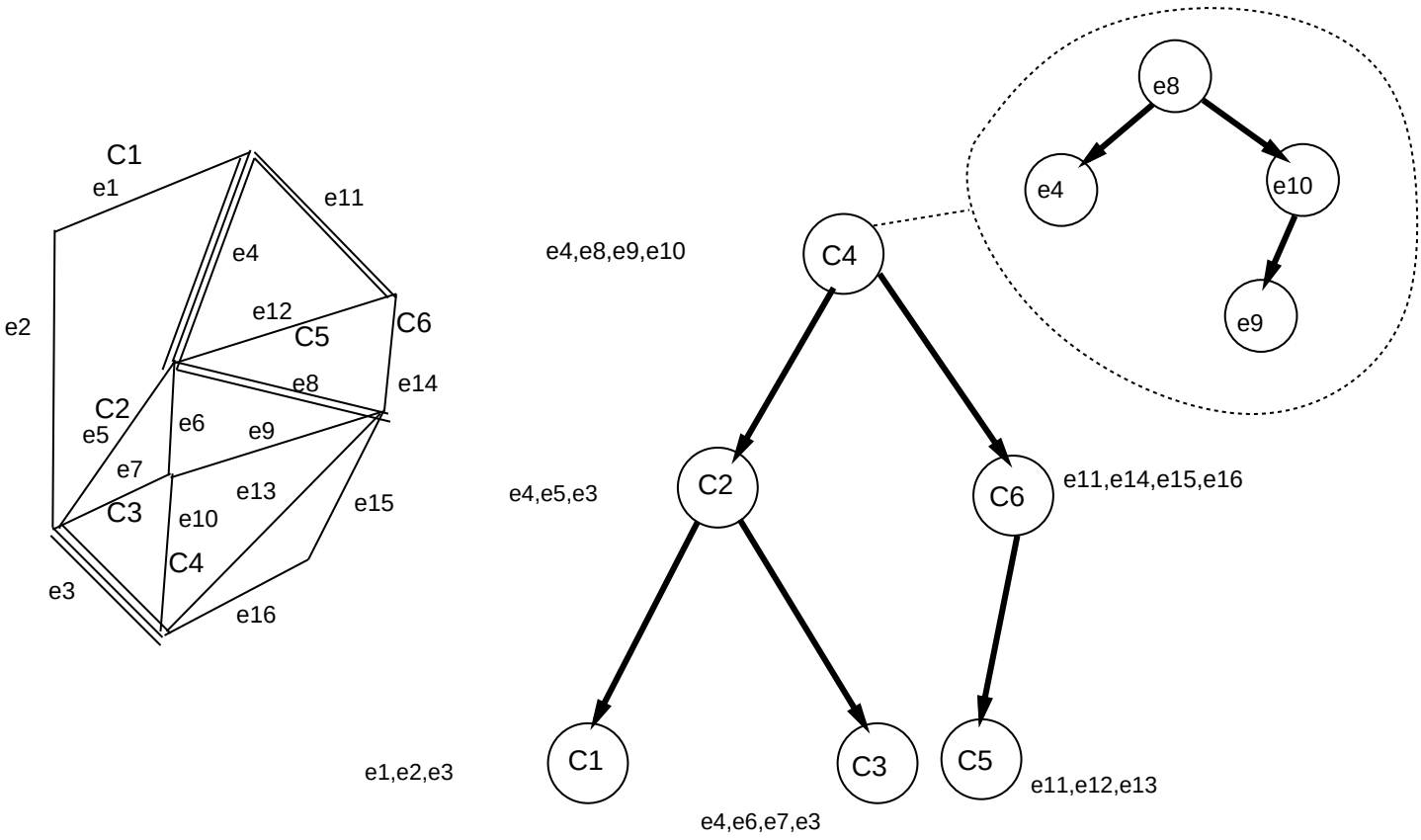
Determination of Monotone Chains

- Monotone chains in a regularized graph can be determined in $O(n)$ time.



Chain Method - Complexity

- Location of a point in a regularized graph with r chains and p nodes per chain requires $O(\log p \log r)$ time.
- There are regular graphs with $n/2$ chains and $n/2$ nodes per chain. Location requires then $O(\log n \log n)$ time.
- Preprocessing requires:
  - Regularization: $O(n \log n)$.
  - Monotone chains: $O(n)$.
- Space: $O(n^2)$.



Summary

Method	Prep. Time	Prep. Space	Query Time
Slab Method	$O(n^2)$	$O(n^2)$	$O(\log n)$
Chain Method	$O(n \log n)$	$O(n)$	$O(\log n)$
Triangulation	$O(n \log n)$	$O(n)$	$O(\log n)$
Trapezoid	$O(n \log n)$	$O(n \log n)$	$O(\log n)$

$\log^2 n$ (W
ha