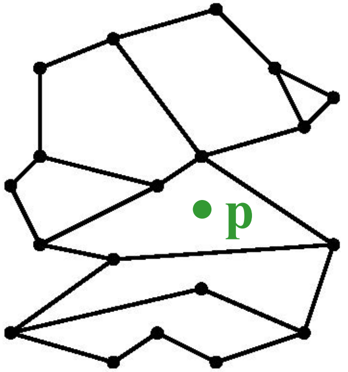
Computational Geometry



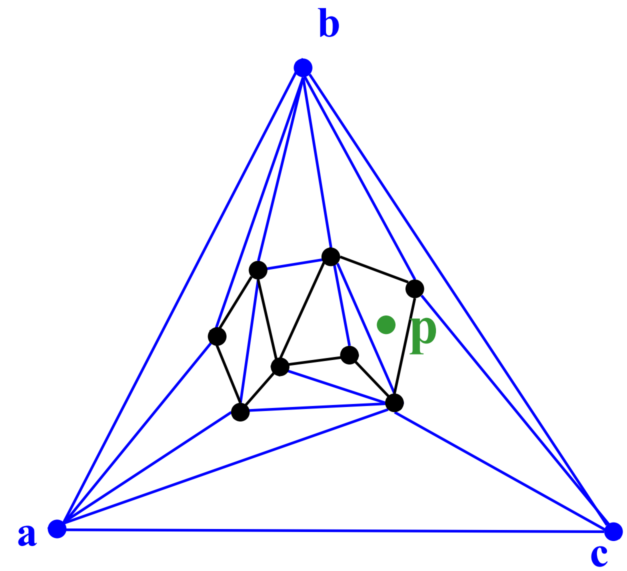
Kirkpatrick's Point Location Algorithm

Michael Goodrich

with slides from Carola Wenk

Kirkpatrick's Point Location Algorithm

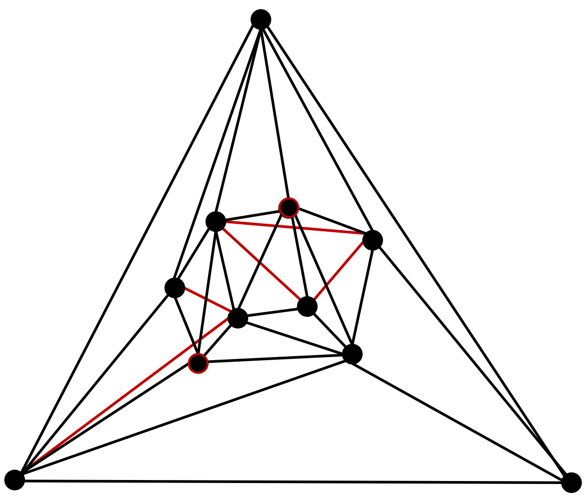
- Needs a triangulation as input.
- One can convert a planar subdivision with n vertices into a triangulation:
  - Triangulate each face, keep same label as original face.
  - If the outer face is not a triangle:
    - Compute the convex hull of the subdivision.
    - Triangulate pockets between the subdivision and the convex hull.
    - Add a large triangle (new vertices **a**, **b**, **c**) around the convex hull, and triangulate the space in-between.



- The size of the triangulated planar subdivision is still O(n), by Euler's formula.
- The conversion can be done in O(n) time.
- Given ***p***, if we find a triangle containing ***p*** we also know the (label of) the original subdivision face containing ***p***.

Kirkpatrick's Hierarchy

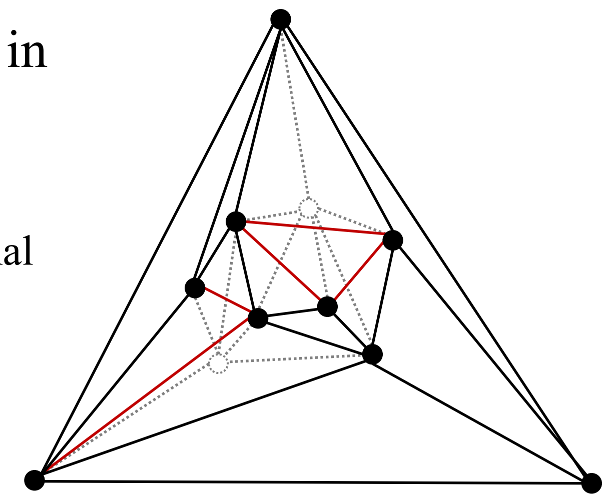
- Compute a sequence $T_0, T_1, \ldots, T_k$ of increasingly coarser triangulations such that the last one has constant complexity.
- The sequence $T_0, T_1, \ldots, T_k$ should have the following properties:
  - T_0 is the input triangulation, T_k is the outer triangle
  - $k \in O(\log n)$
  - Each triangle in T_{i+1} overlaps $O(1)$ triangles in T_i
- How to build such a sequence?
  - Need to delete vertices from T_i .
  - Vertex deletion creates holes, which need to be re-triangulated.
- How do we go from T_0 of size $O(n)$ to T_k of size $O(1)$ in k=$O(\log n)$ steps?
  - In each step, delete a constant fraction of vertices from T_i .
- We also need to ensure that each new triangle in T_{i+1} overlaps with only $O(1)$ triangles in T_i .



Vertex Deletion and Independent Sets

When creating T_{i+1} from T_i , delete vertices from T_i that have the following properties:

- **Constant degree:**
  Each vertex **v** to be deleted has O(1) degree in the graph T_i .
  - If **v** has degree d, the resulting hole can be re-triangulated with d-2 triangles
  - Each new triangle in T_{i+1} overlaps at most d original triangles in T_i
- **Independent sets:**
  No two deleted vertices are adjacent.
  - Each hole can be re-triangulated independently.

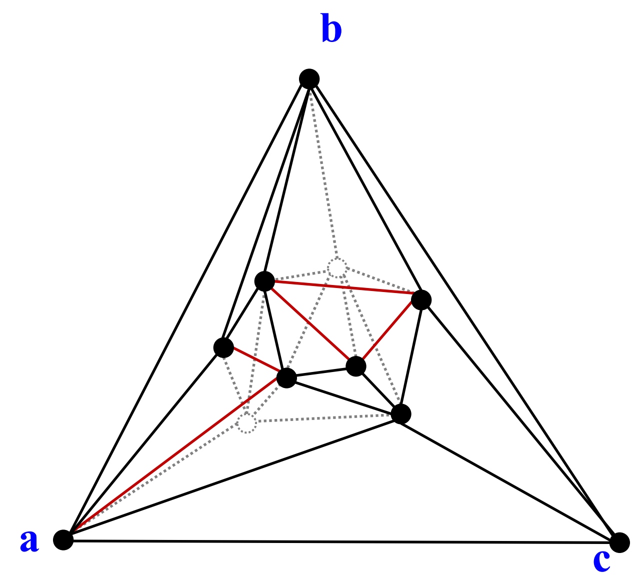


Independent Set Lemma

Lemma: Every triangulated planar graph on $n \geq 4$ vertices for contains an independent vertex set of size $n/18$ in which each vertex has degree at most 8. Such a set can be computed in $O(n)$ time.

Use this lemma to construct Kirkpatrick's hierarchy:

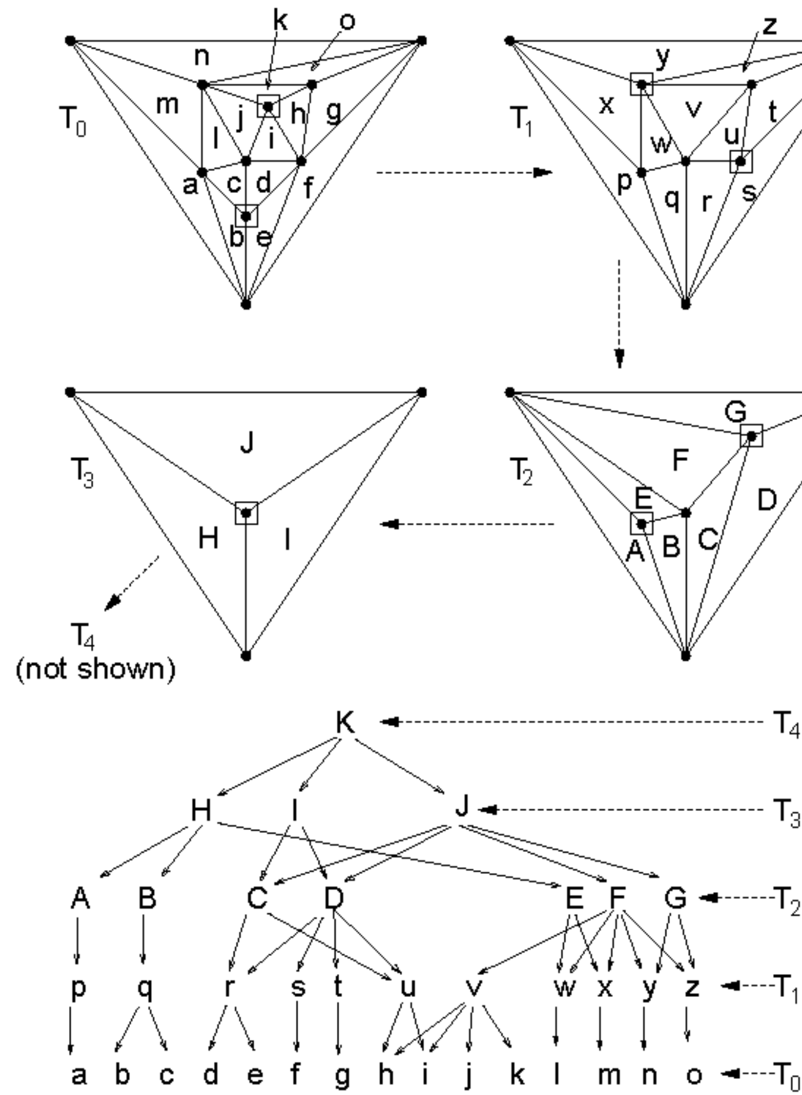
- Start with T_0, and select an independent set S of size $n/18$ in which each vertex has maximum degree 8. [Never pick the outer triangle vertices **a**, **b**, **c**.]
- Remove vertices of S, and re-triangulate holes.
- The resulting triangulation, T_1, has at most $17/18n$ vertices.
- Repeat the process to build the hierarchy, until T_k equals the outer triangle with vertices **a**, **b**, **c**.
- The depth of the hierarchy is $k = \log_{18/17} n$



Hierarchy Example

Use this lemma to construct Kirkpatrick's hierarchy:

- Start with T_0, and select an independent set S of size $n/18$ in which each vertex has maximum degree 8. [Never pick the outer triangle vertices **a**, **b**, **c**.]
- Remove vertices of S, and re-triangulate holes.
- The resulting triangulation, T_1, has at most $17/18n$ vertices.
- Repeat the process to build the hierarchy, until T_k equals the outer triangle with vertices **a**, **b**, **c**.
- The depth of the hierarchy is $k = \log_{18/17} n$



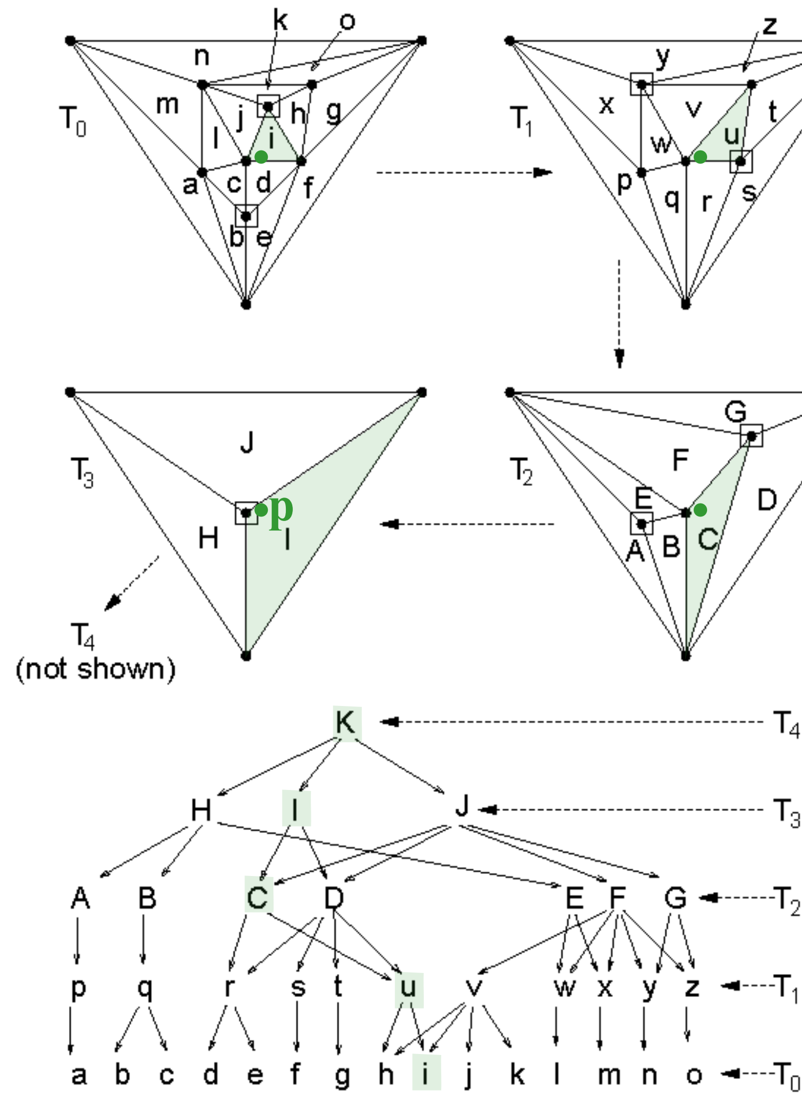
Hierarchy Data Structure

Store the hierarchy as a DAG:

- The root is T_k .
- Nodes in each level correspond to triangles T_i .
- Each node for a triangle in T_{i+1} stores pointers to all triangles of T_i that it overlaps.

How to locate point ***p*** in the DAG:

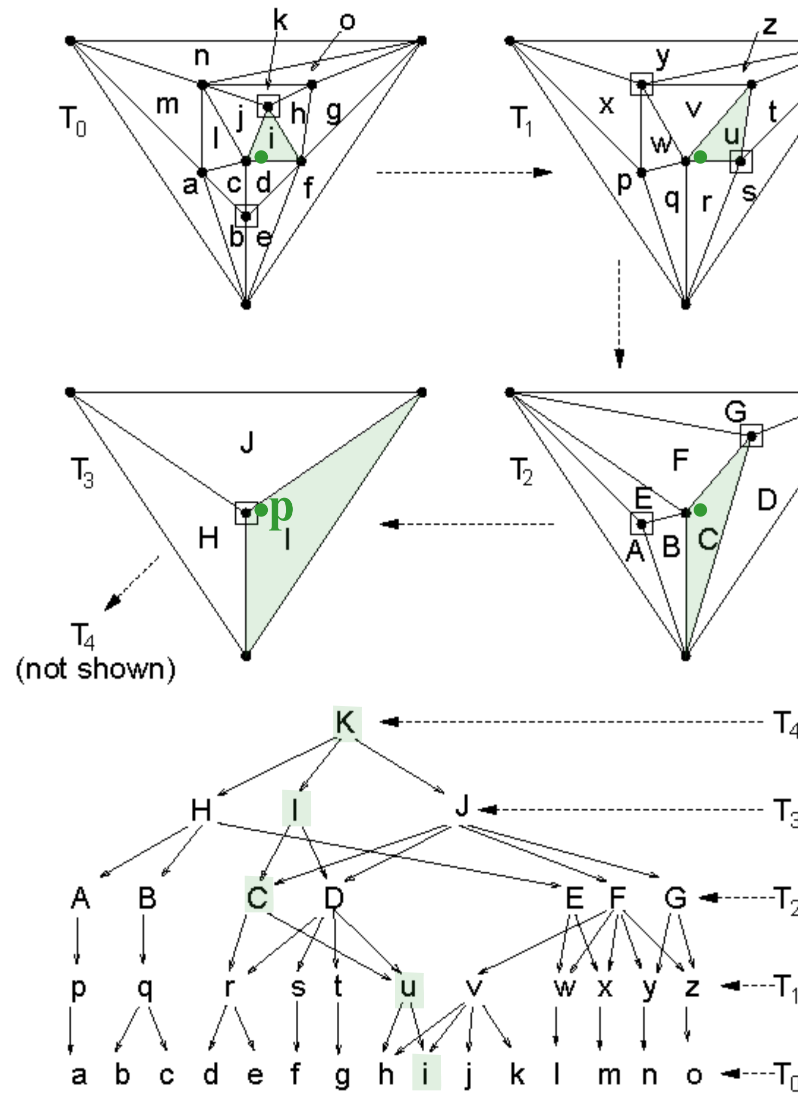
- Start at the root. If ***p*** is outside of T_k then ***p*** is in exterior face; done.
- Else, set Δ to be the triangle at the current level that contains ***p***.
- Check each of the at most 6 = 8-2 triangles of T_{k-1} that overlap with Δ, whether they contain ***p***. Update Δ and descend in the hierarchy until reaching T_0 .
- Output Δ .



Analysis

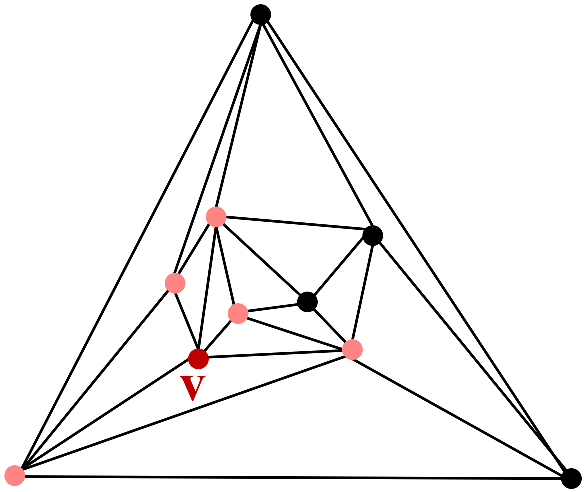
- **Query time** is O(log n): There are O(log n) levels and it takes constant time to move between levels.
- **Space complexity** is O(n):
  - Sum up sizes of all triangulations in hierarchy.
  - Because of Euler's formula, it suffices to sum up the number of vertices.
  - Total number of vertices:

    $n + 17/18\ n + (17/18)^2\ n + (17/18)^3\ n + \ldots$
    $\leq 1/(1\text{-}17/18)\ n = 18\ n$
- **Preprocessing time** is O(n log n):
  - Triangulating the subdivision takes O(n log n) time.
  - The time to build the DAG is proportional to its size.



Independent Set Lemma

Lemma: Every triangulated planar graph on $n \geq 4$ vertices contains an independent vertex set of size $n/18$ in which each vertex has degree at most 8. Such a set can be computed in $O(n)$ time.



Proof:

Greedy algorithm to construct an independent set:

- Mark all vertices of degree ≥ 9
- While there is an unmarked vertex
  - Let **v** be an unmarked vertex
  - Add **v** to the independent set
  - Mark **v** and all its neighbors
- Can be implemented in $O(n)$ time: Keep list of unmarked vertices, and store the triangulation in a data structure (DCEL) that allows finding neighbors in $O(1)$ time.

Independent Set Lemma

Still need to prove existence of a *large* independent set.

- Euler's formula for a triangulated planar graph on n vertices:
  #edges = $3n - 6$
- Sum over vertex degrees:
  $\sum_v \deg(v) = 2\ \#\text{edges} = 6n - 12 < 6n$
- **Claim:** At least $n/2$ vertices have degree ≤ 8.
  Proof: By contradiction. So, suppose otherwise.
  $\rightarrow$ $n/2$ vertices have degree ≥ 9. The remaining have degree ≥ 3.
  $\rightarrow$ The sum of the degrees is $\geq 9\ n/2 + 3\ n/2 = 6n$. Contradiction.

- In the beginning of the algorithm, at least $n/2$ nodes are unmarked. Each picked vertex **v** marks ≤ 8 other vertices, so including itself 9.
- Therefore, the while loop can be repeated at least $n/18$ times.
- This shows that there is an independent set of size at least $n/18$ in which each node has degree ≤ 8.

□

Summing Up

- Kirkpatrick's point location data structure needs $O(n)$ preprocessing time, $O(n)$ space, and has $O(\log n)$ query time.
- It involves high constant factors though. So this algorithm, while asymptotically optimal, is mostly of theoretical interest.