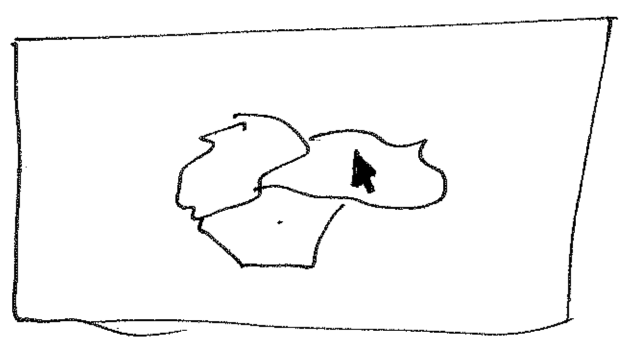
Problem: Point Location in a planar subdivision (PSLG)

Input: A PSLG.
Preprocess: Build a D.S. to support P.L. queries in $O(n)$ time
Query: For $q \in \mathbb{R}^2$, return face containing q.
Goal: $O(\log n)$ time queries

Most common setting:



where did you click?

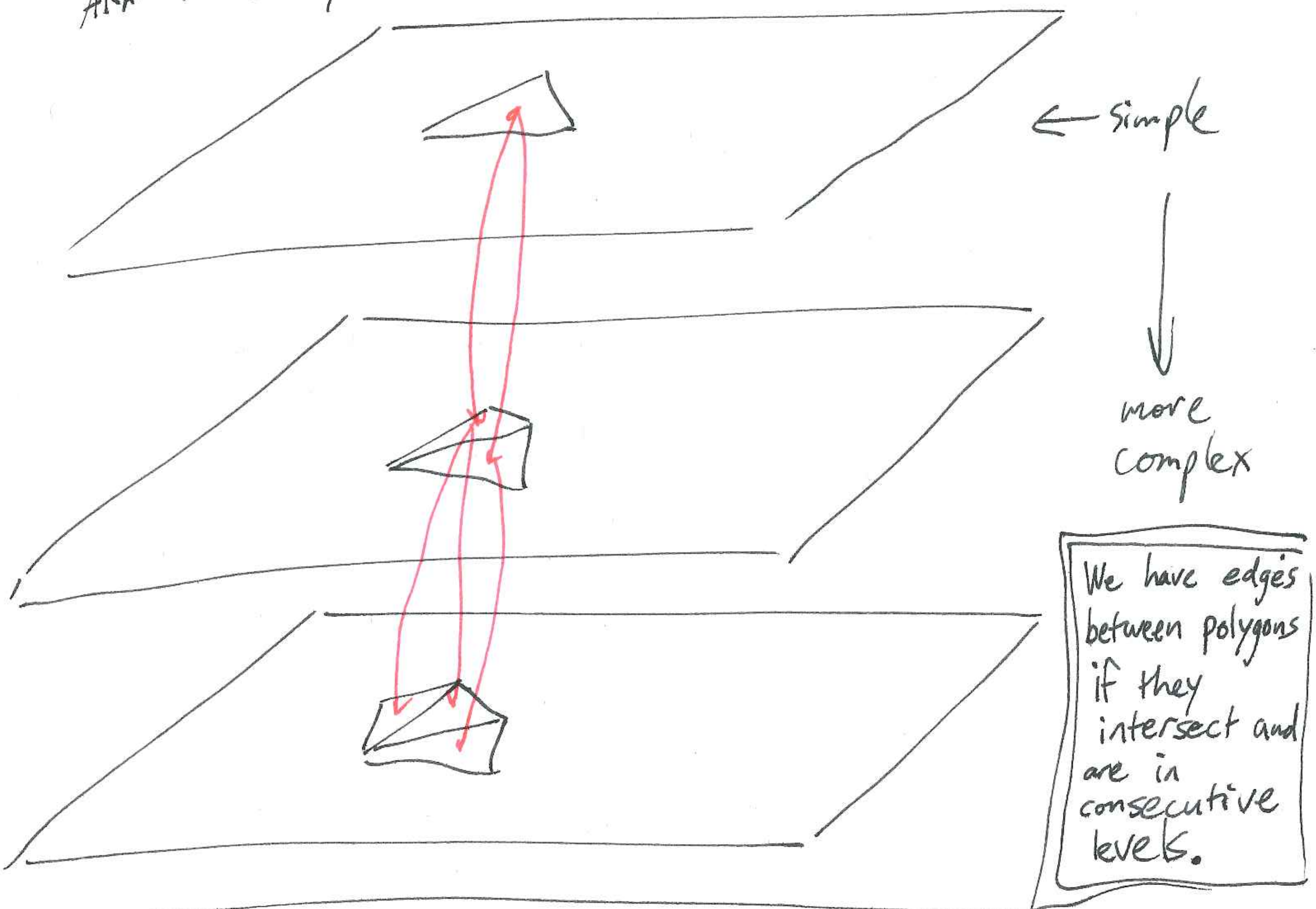
(mouseX, mouseY)

$n = \#$ vertices

$\#$ faces $= O(n)$

[Kirkpatrick's Algorithm '81]

AKA The History DAG.

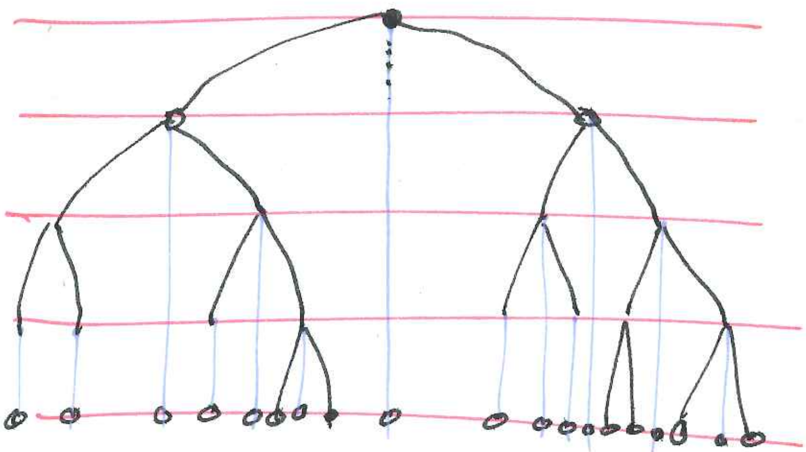


Difference w/ Rand. Inc. Del.

(1) Don't know queries in advance

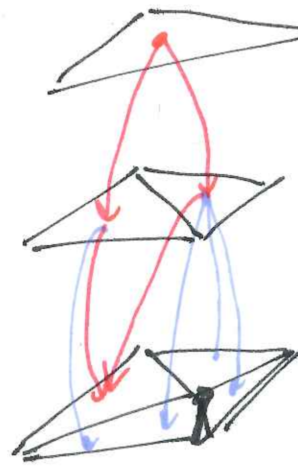
(2) We want worstcase performance

Related to BSTs (Binary Search Trees)



In 1D, a BST really is a history DAG.

In 2D, we have a DAG, NOT a tree.



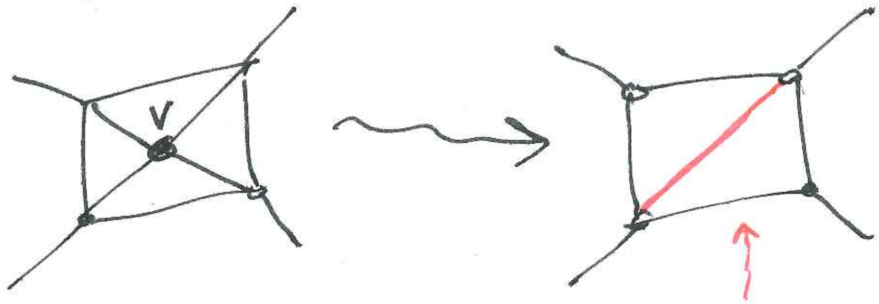
We care about:

(1) Degree (goal: $O(1)$)

(2) Depth (goal: $O(\log n)$)

Idea 1: Think Backwards

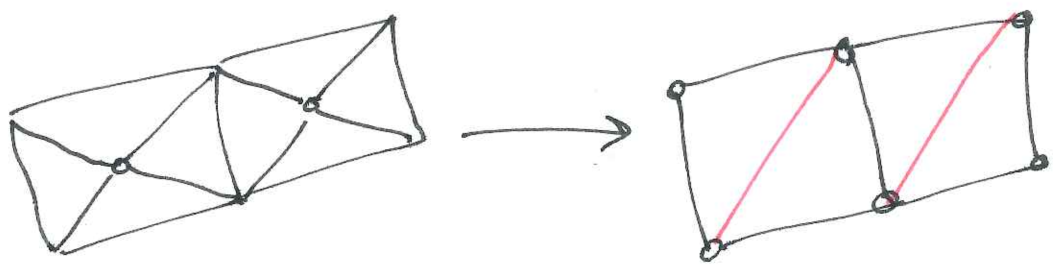
Instead of building up the PSLG, imagine breaking it down, i.e. simplifying it.



remove v. resulting square is cavity(v)

Idea 2: Only consider triangulations

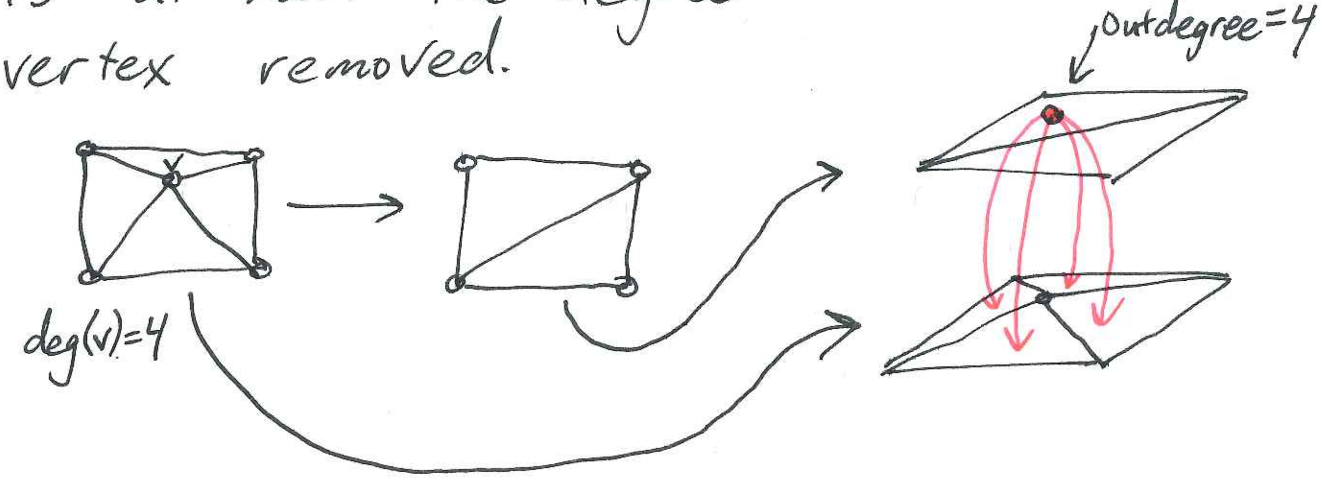
Idea 3: Think Parallel



We want to remove many "independent" vertices at once. Hopefully, $O(\log n)$ such rounds will suffice.

Observation 1: If u, v are not adjacent the $\text{cavity}(u) \cap \text{cavity}(v) = \emptyset$.

Observation 2: The degree in the DAG is at most the degree of the vertex removed.



Lem In a Δ^n, at least $n/2$ vertices have degree less than 12.

pf Suppose for contradiction that $n/2$ vertices have degree 12 or more. Then,

$$|E| = \frac{1}{2} \sum_{v \in V} \deg(v) \geq \frac{1}{2}\left(\frac{n}{2} \cdot 12\right) = \underbrace{3n > |E|}_{\text{Euler's Formula.}}$$

Contradiction $\Rightarrow$ done.

Lem: Given a Δ_1^n, we can find an independent set of at least $n/24$ vertices of degree less than 12 in $O(n)$ time.

pf Apply the following Greedy algorithm.

```
set r_i = false ∀i=1..n
For i = 1 to n
{ if deg(v_i) < 12 and r_i == false
    { output ≪ v_i
      for each v_j adjacent to v_i, set r_j = true.
    }
}
```

reject all neighbors of V_i by setting r_j.

By prev. lemma, there are $n/2$ low degree vertices. The above algorithm throws out at most 11 for every one found. So, we will return $\frac{1}{12}\left(\frac{n}{2}\right) = \frac{n}{24}$ vertices at least.

Thm A query takes $O(\log n)$ time.

pf

Let n_i be the number of vertices at level i.

$n_0 = n$, all the points appear in bottom level.

$n_{i+1} \le \left(\frac{23}{24}\right) n_i$ because we remove at least $n/24$ vertices in each round.

So, the total depth is the minimum d s.t. $n_d \le$ constant.

$$n_d \le \left(\frac{23}{24}\right)^d n \Rightarrow d \le \log_{24/23}(n) = O(\log n).$$

So, the depth is $O(\log n)$.

In a query, going from level i to level $i-1$ requires checking if the query is in at most 11 possible triangles in the next level. This takes $O(1)$ time.

So, the total time is $O(\log n)$.

Thm Given a Δ^n, the preprocessing time is ~~$O(n \log n)$~~ $\rightarrow O(n)$.

pf At each level i

- Finding independent set $[O(n_i)]$
- Retriangulating $[O(n_i)]$
- Checking HDAG edges $[O(n_i)]$

$O(\log n)$ levels

Counting precisely.

Cost of level i is cn_i for some c.

$n_i \leq \left(\frac{23}{24}\right)^i n$.

So, $\text{cost} \leq \sum_{i=0}^{\log_{24/23}(n)} cn_i \leq cn \sum_{i=0}^{\log_{(24/23)}(n)} \left(\frac{23}{24}\right)^i = O(n)$.