6.851 ADVANCED DATA STRUCTURES

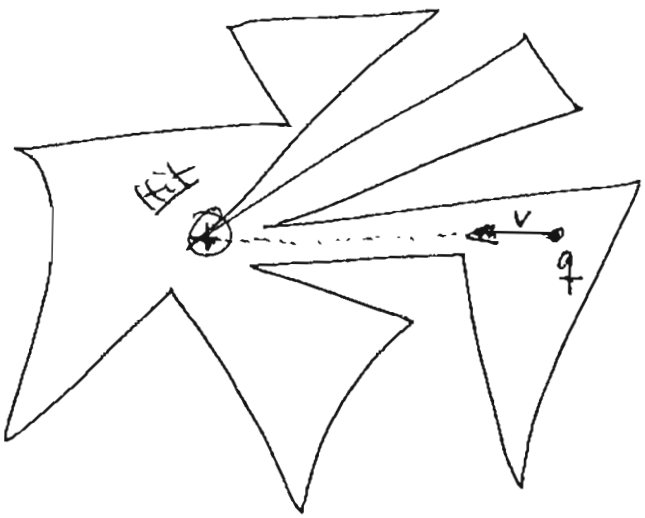
(ANDRÉ SCHULZ' NOTES)

Lecture 5.2

Ray shooting

Problem: Set of geometric objects (lines, polygons ...)
query point q + direction v
Q: What is the *first* object hit by the ray emanating from q at v ?

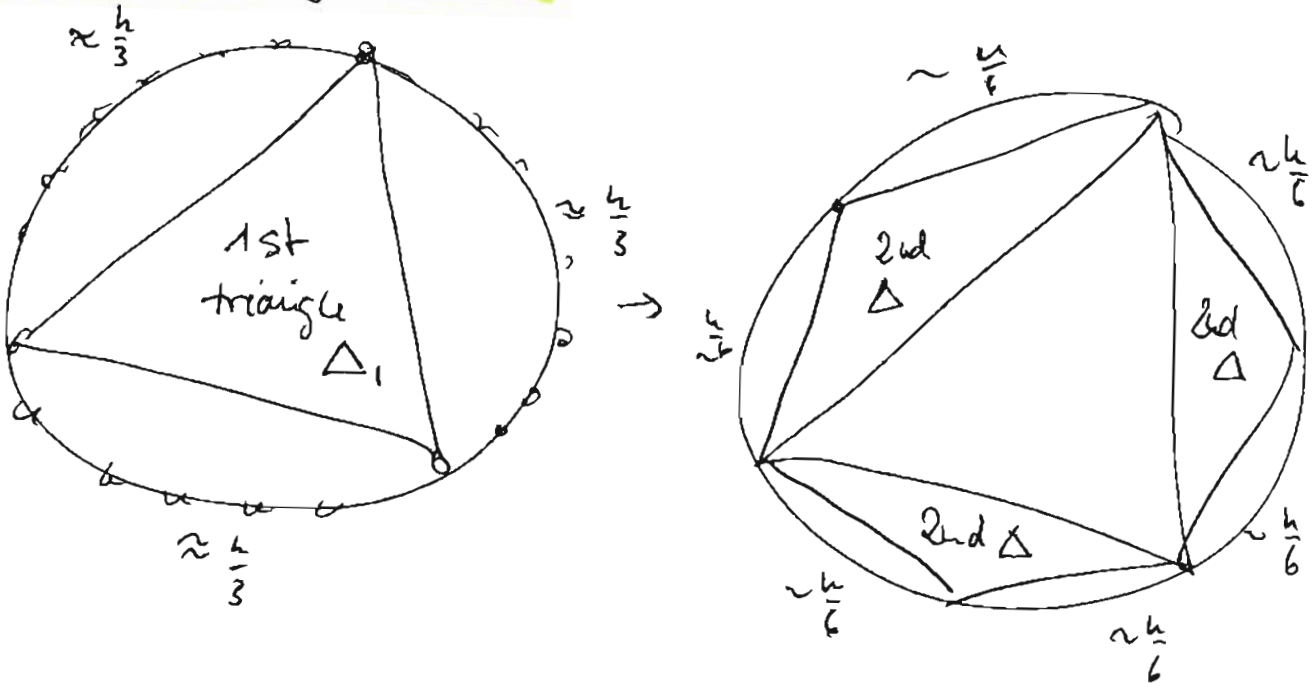
Classic Ray shooting in simple polygons
[Chazelle et al. 94]



Algorithm & Data Structure by Chazelle et al. '92
Simplification Hershberger & Suri '95

Basic idea: Divide and conquer DECOMPOSE

For a convex polygon we can define a balanced triangulation



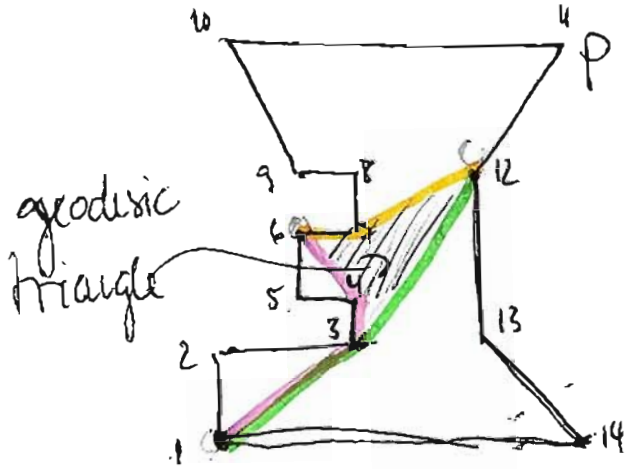
and so on.

Property: the dual is a tree (balanced if (~~rooted~~ deg 3) looked at Δ_1)

$\Rightarrow$ depth $\leq O(\log n)$

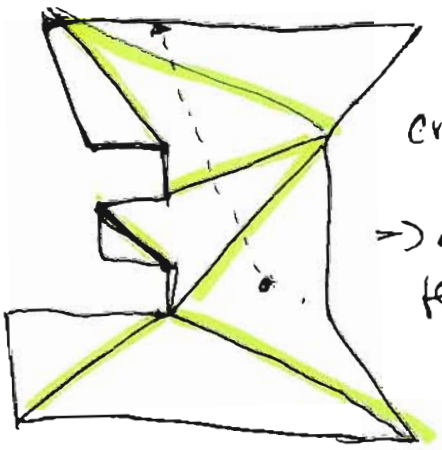
We define in a similar fashion the balanced geodesic triangulation for a simple polygon (instead of line segments we use shortest paths

Example



geodesic triangle

geodesics 1-6
6-12
12-1



crossing lines

=> up/down in the dual tree

Observation: A line intersecting (P) hits the geodesic triangles in an order that reflects a "traversal" in the dual tree

=> At most $O(\log n)$ geodesic triangles are crossed by a line in int(P)

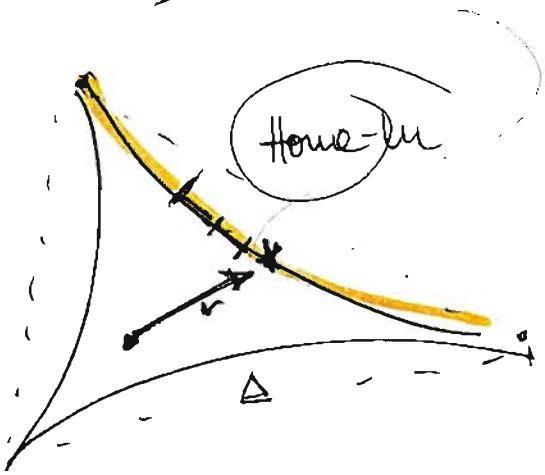
=> So we reduced the problem to a ~~see~~ answer a ray shooting query for a geodesic triangle $O(\log^2)$ time

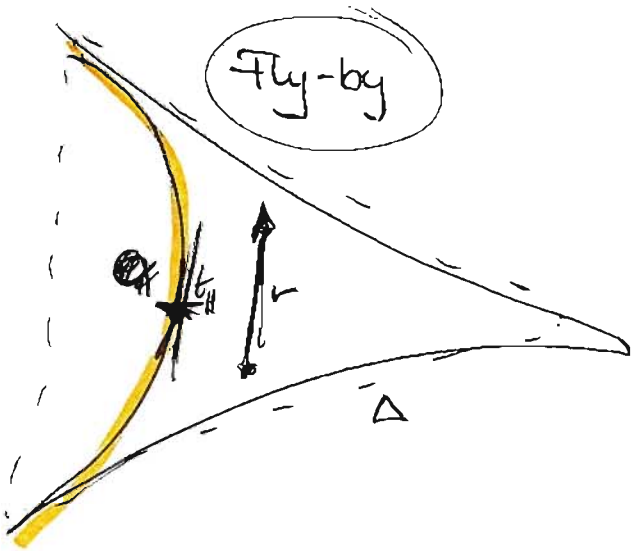
too much! (leads to $O(\log^2 n)$ Query time

~~Next idea: triangulate the geodesic triangles~~

How to decide where we "leave" a geodesic Δ

2 situations





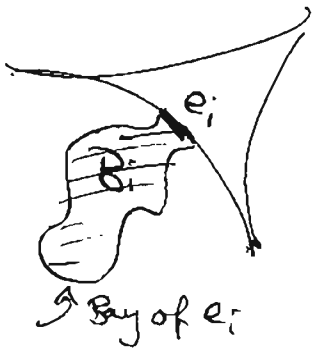
Usually: 1 Chain : Home-in (Case A)
1 Chain : Fly-By (Case B)

Case A

We can find the point $r \cap \Delta$ with binary search

$\Rightarrow$ we use **weight balanced binary search trees**

WBST



B_i = # of edes in the Bay of e_i

↖ weights for the tree

$\Rightarrow$ Query time $O\left(1 + \log\left(\frac{\sum_{e \in C} B_j}{B_i}\right)\right)$

Lemma Searches for all Home-Ins here $O(\log n)$ ti

Proof: HW

Case B

1st Step: determine the edge/vertex parallel to the ray
(binary search over the slopes of all 3 chains)

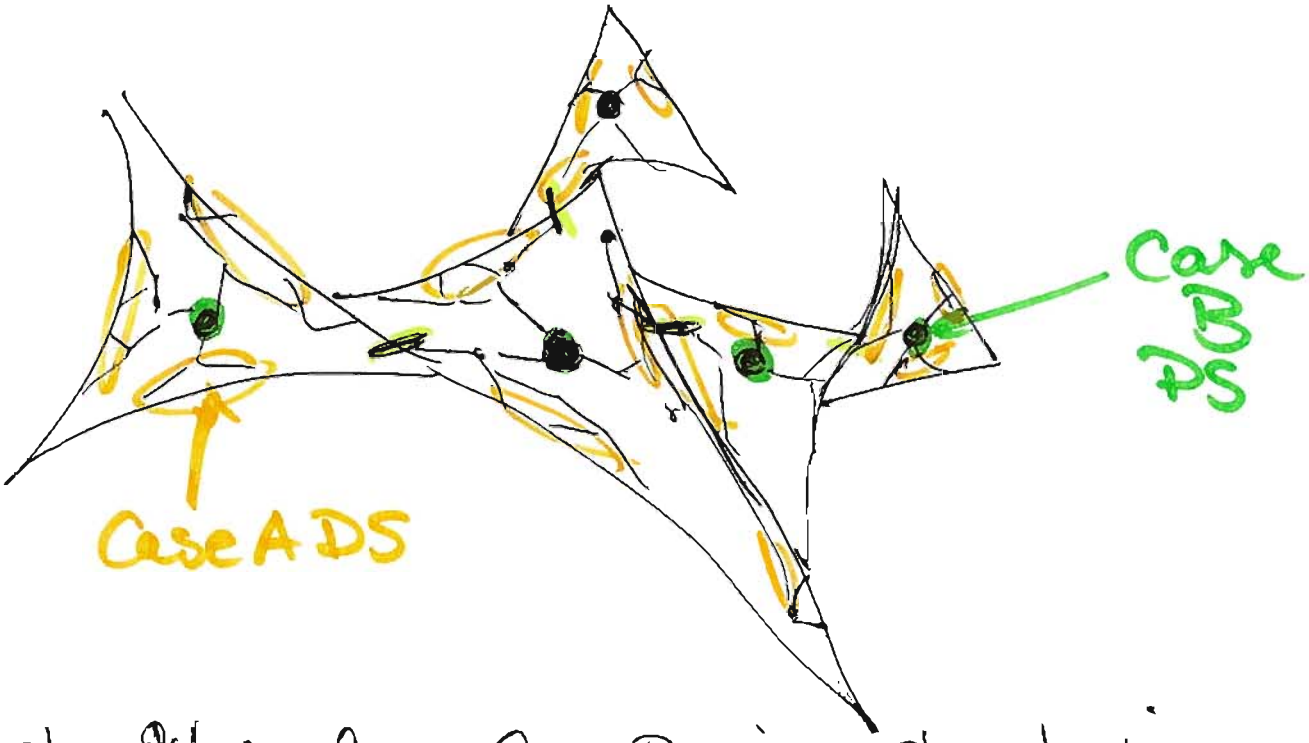
2nd Step: Use the weight balanced trees to determine the exit point (if there is any)

Notice: step 1 is necessary to perform a binary search in step 2, because we know in which part of C to search

=> $O(\log^2 n)$ query time

Homework

=> Speed-up: Fractional Cascading



Data structure for Case B is stored in Nodes in the grad. △

- Link all Case A DS (WBST) by there common edges
- Link alle Case A DS (WBST) with the "Case B nodes"

⇒ Apply Fractional Cascading [Guibas Chazelle '86] in its general form

(The slope we search in the case B DS is always the same!)

⇒ $O(\log n)$ Query time