# Computational Geometry

cgvr.cs.uni-bremen.de

Exersice → Grad A
Fachgespräch → Grad B

$\min\{B, \frac{1}{2}(A+B)\}$

THE Better ONE

95% → 1.0
40% → 4.0

Exercise Group of 2 or alone

Importan PreProcessing

- Domain discretion (decompose)

- Do grid over it → Computational inefficient (spaces aren't used)
→ uniform Grid not good

- Non-uniform, comforming mesh that respects the input.

- Long & thin triangles, always bad

- quadtree quite nice

♡ used in simulation to eg. flow(air) around a vehicle or crashed test.

# Quadtrees

store geometry data

Coincidence, Incidence, adjacancy

same location


e is incidence
to point v

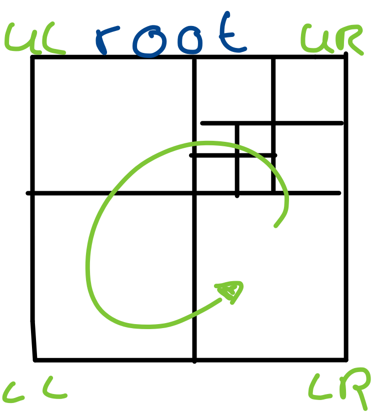

they are neighbors

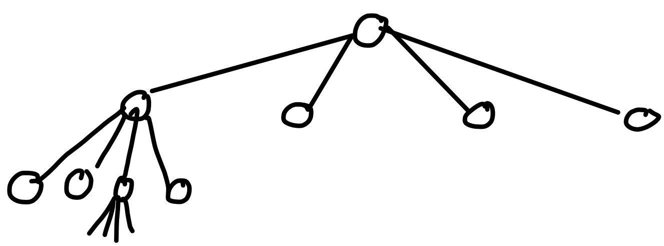
Points/vectors: p, q, v, w....

set of points, Polygons; ...: P, Q, S...

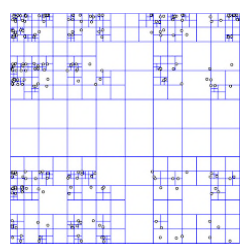
Segments: $\overline{qp}$

Quadtree = Tree, with
inner nodes corresponding to squares;
children of a node partition the node
into four quadrants



Children Direction

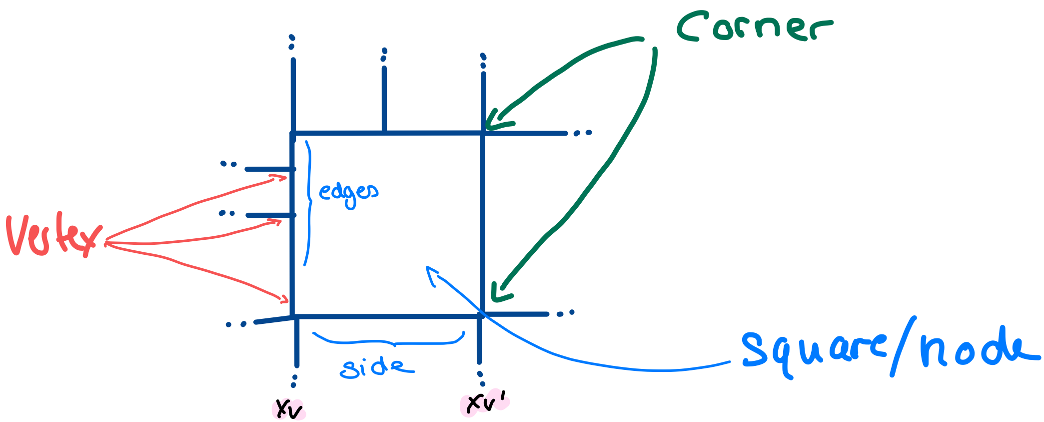


Note: quadtree induces partitioning of the domain

covers whole area, but does not overlay other members

! Complete quadtree is like a normal grid, called multi-level grid

## Terminology:




**Def:** nodes are adjacent :$\Longleftrightarrow$ (only if) their squares share an edge

**Def:** square of a node $v$

$$q(v) = [x_v, x_v'] \times [y_v, y_v']$$

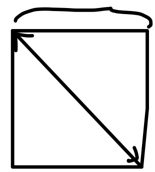
Given: Set of points $P \subseteq \mathbb{R}^2$

**Def:** Quadtree $Q$ over point set $P$

**Proof:** wlog $s = 1$

side length of nodes $v$ on level $i = \frac{1}{2^i}$

max distance inside $v = \frac{\sqrt{2}}{2^i}$.



If $v$ is inner node $\Rightarrow$

$$c \le \frac{\sqrt{2}}{2^i}$$

$$\Rightarrow i \le \log \frac{\sqrt{2}}{c} = \log \frac{1}{c} + \frac{1}{2} \Rightarrow \text{Lemma}$$

for leaves: $i \le \underbrace{\log \frac{1}{c} + \frac{1}{2}}_{\text{level of parent}} + 1$

## Lemma: Complexity of quadtrees

A quatrees of depth $d$ over $n$ points, takes $O(n(d+1))$ nodes and takes $O(n(d+1))$ to construct.
↳ can get rid of it

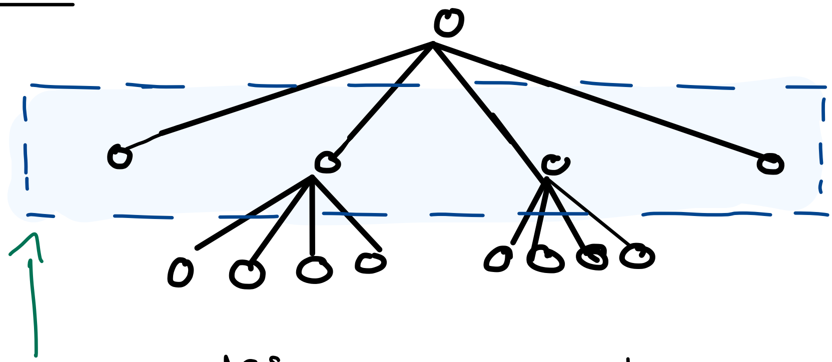
**Proof:**

$\#$ leaves $= (\#$ inner nodes$) \cdot 3 + 1$ (by induction)

Number of

$\Rightarrow$ upper bounds on inner nodes suffice.

## Part 1



\# inner nodes on one level

$\leq n$ (in each inner node there are at least 2 points)

$\Rightarrow$ # inner nodes over all levels $\leq n(d-1)$

$\Rightarrow$ # nodes $\leq n(d-1) + 2n$
because in each quadtupel, 2 leaves must contain a point.

## Part 2

For each node $v$, we time $T(v)$

$T(v) = O(m)$, $m$ = # points in $v$.

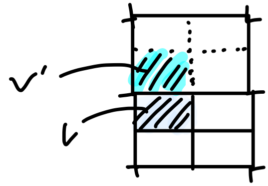
Sum of all points on Level $i \leq n$

$$\sum_{v \text{ is node on level } i} T(v) \in O(n)$$

$\Rightarrow$ time $O(n \cdot d)$ or $(n \cdot d) + 1$

# Find north neighbor

Given: node v




Sought: v' = north neighbor of v, such that depth(v') ≤ depth(v)

# Algorithmus getNorthNeighbor(v)

If v is root → return nil

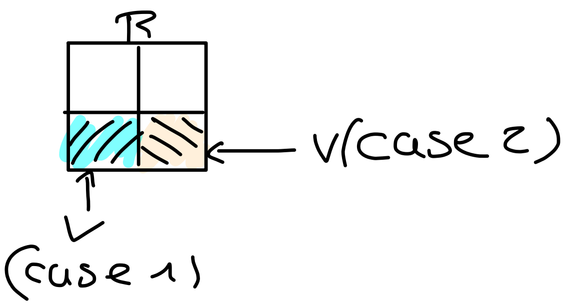
let P := parent(v)

(1) If v is lowerleft (LL) child of P → return UL child of P

$\underbrace{\text{UL child of P}}_{\text{2 sibling of v}}$

(2) If v is LR child of P → return UR child of P

## Case 1 & 2



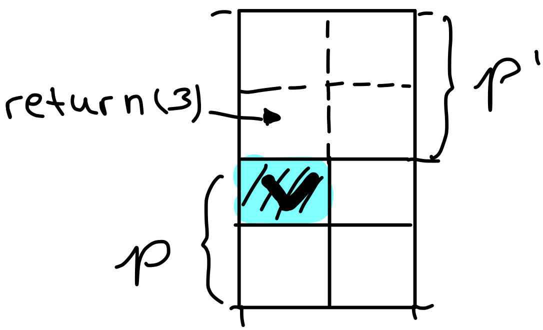
$p' = \text{getNorthNeighbor}(p)$

If $p'$ is nil or $p'$ is leaf → return $p'$

(3) If v is UL child of p → return LL child of $p'$

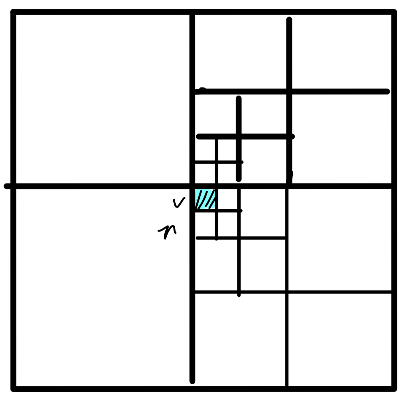
(4) If v is UR -"- → return LR -"-

## Case 3 & 4




worst case

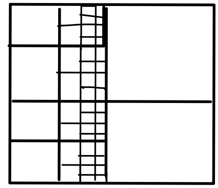
**Running Time:** $O(d)$




Worst case to get parent go all the way up in the hirachy and all the way down again.

! Exam: Sketch this for get west neighbors and why is it so complex + worst case?

# Balanced Quadtrees




Def: A quadtree is "balanced"

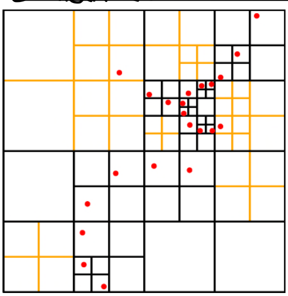
$\iff$

$\forall$ neighbors $v, v'$: $|\text{depth}(v) - \text{depth}(v')| \leq 1$

Corollary

If $Q$ is balanced $\Rightarrow$ size of neighbors differs by factor 2 at most.

Balanced Quadtree



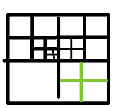
## Algo for constructing balanced quadtrees:

Maintain: List $L$ of Leaves

while there are still nodes $v$ in $L$:

1. check wether $v$ needs to split (neighbor finding algo)

2. If $v$ had to split, check wether neighbors need splitting, too



## Lemma:

Let $Q$ be a quadtree with $m$ nodes,
$\hat{Q}$ = balanced quadtree from $Q$.

Then $\hat{Q}$ has $O(m)$ nodes, and it can be constructed in time $O(m(d+1))$.

## Proof

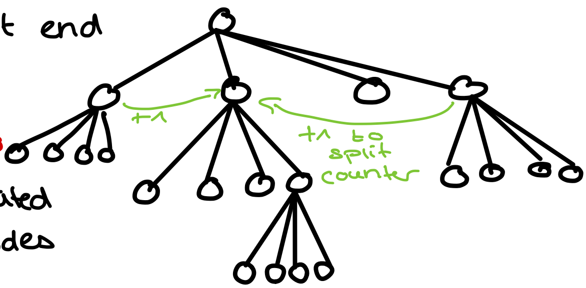
**Part 1**: We prove that there are $O(m)$ splitting operations
($\Rightarrow$ Lemma follows, b/c each split
generate 4 additional nodes.

**Define split counter**

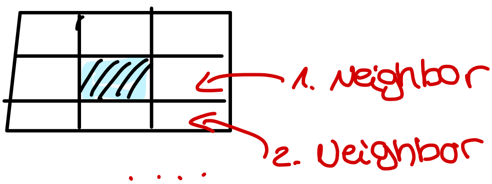
- Only for old nodes (from origin quadtree) :=
  how many times did the old node cause split

- Split counter at end of balancing $\leq 8$

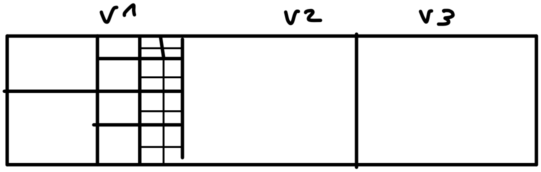
#neighbors




- Each old node generated at most $8 \cdot 4$ new nodes



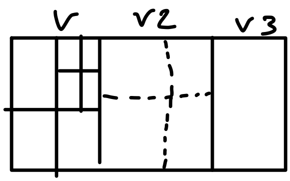
## Assertion (to be proven):




⇒ No matter how deep subtree under v1 is, v3 never has to split because of v1.

**Def:** $D(v)$ = depth of subtree under $v$.

Base case:

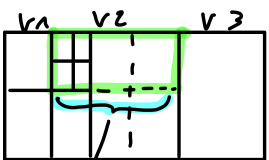



$D(v_2) = D(v_3) = 0$
$D(v_1) = 2$

Inductive step: Lemma is true for $D < d$

$D(v_1) = d > 2$

$D$(UR Child of $v$) $= d-1$. $v_2$ is split at least once




situation for which Lemma holds, b/c depth of UR child of $v \leq D$
⇒ UR child of $v_2$ will not be split

## Part 2:

Time per node $\in O(d+1)$, b/c of const number of neighbor finding operations (ops.).
Each node will be considered only once ⇒ Lemma