CSE520 Computational Geometry
Lecture 24
Compressed Quadtrees

Antoine Vigneron

Ulsan National Institute of Science and Technology

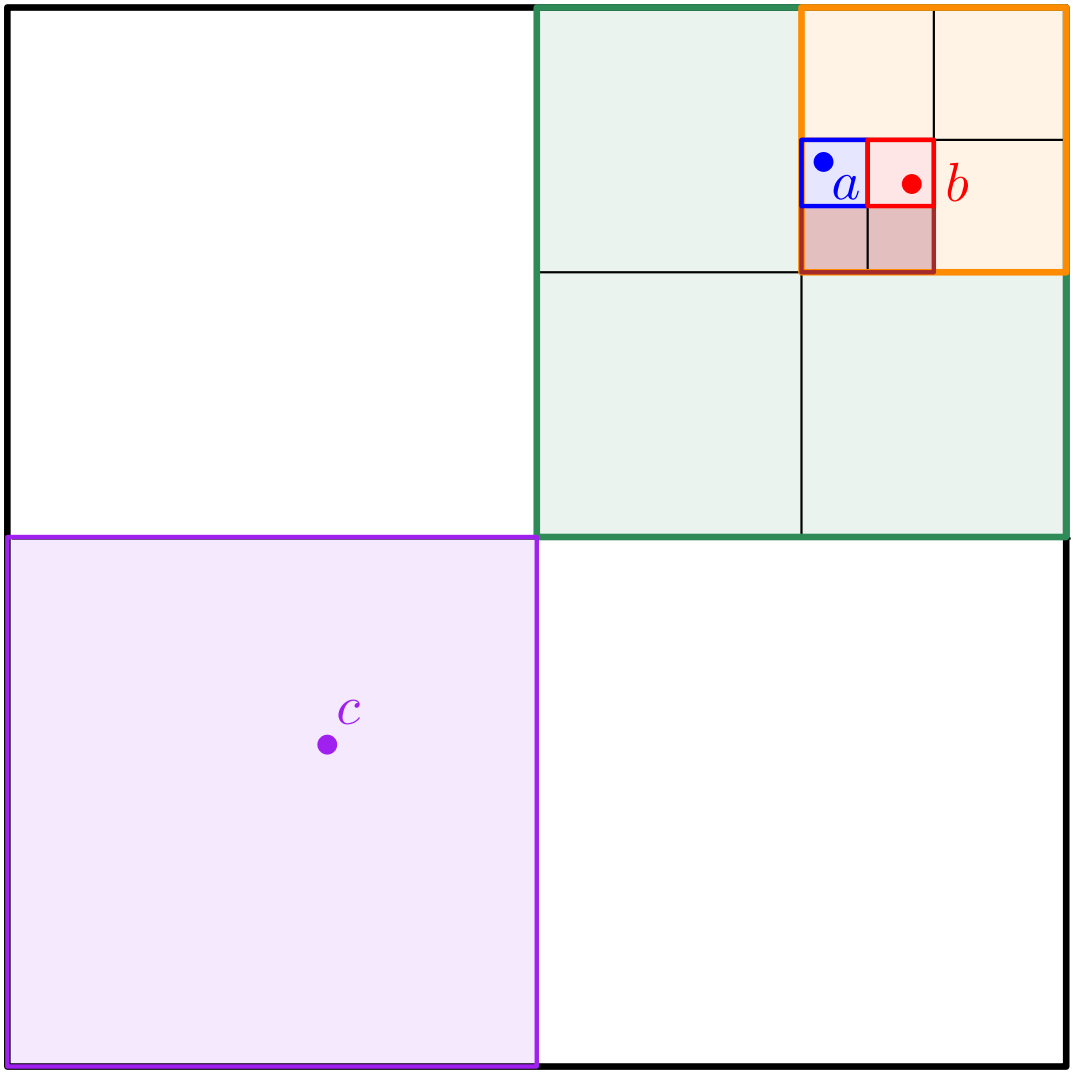
June 15, 2020

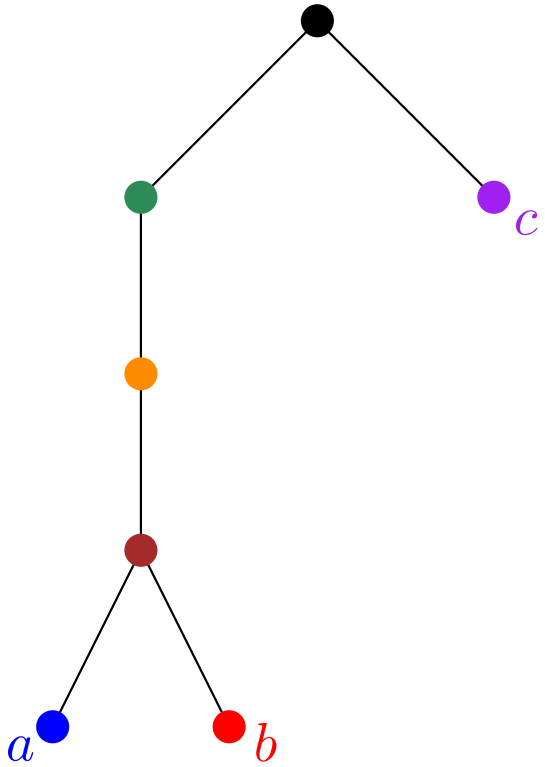
Course Organization

- Today, I present *compressed quadtrees*, which are a modified version of the quadtrees presented in last lecture that use less space.
- I will also show how to apply them to near-neighbor searching.

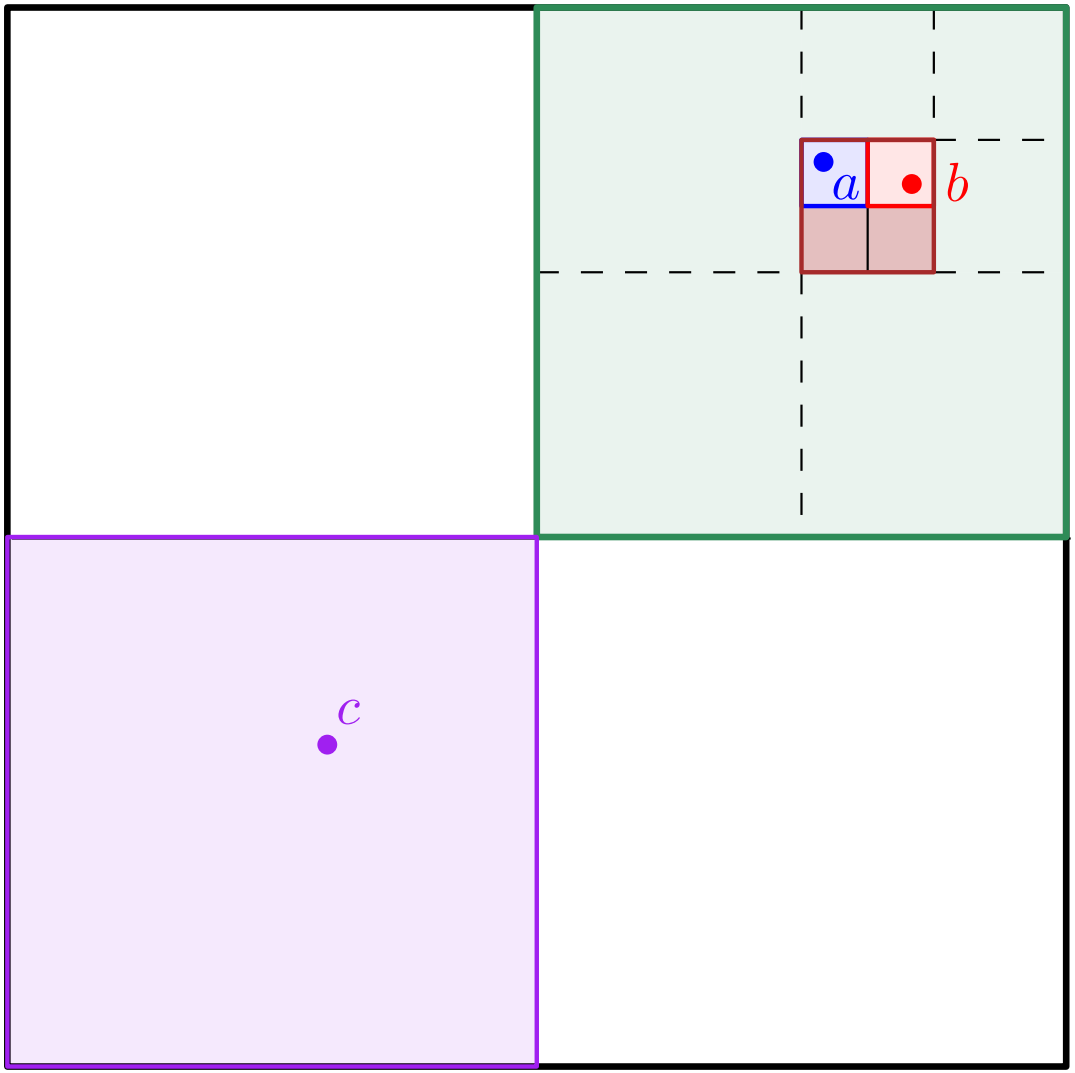
- References:
- Sariel Har Peled's book, chapters 2 and 17.

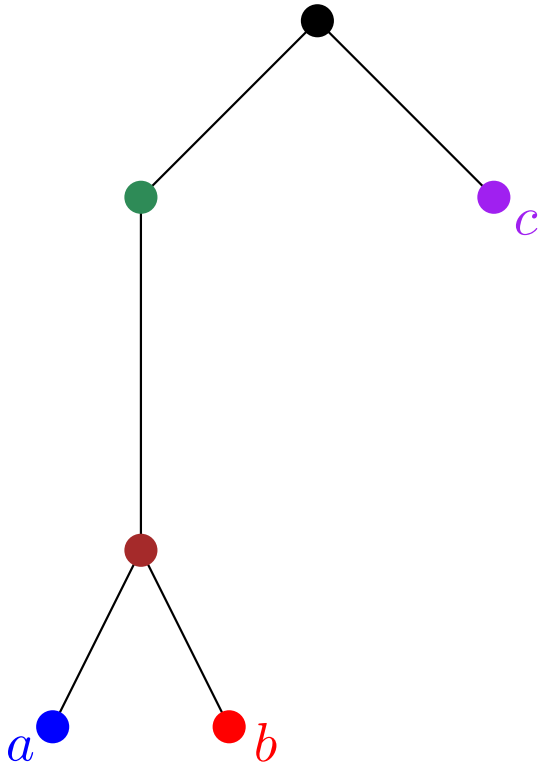
An Ordinary Quadtree



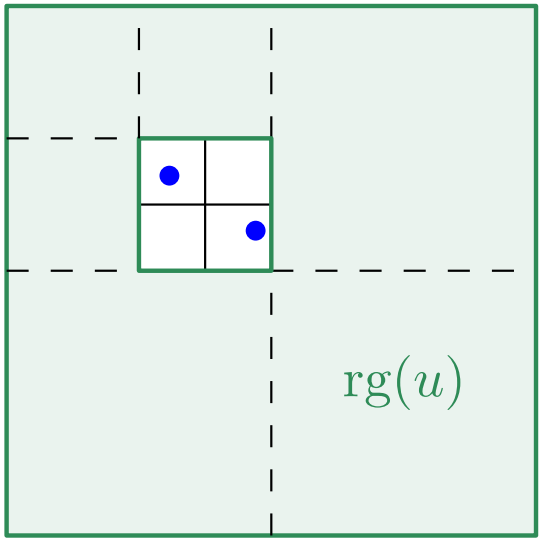


Compressed Version



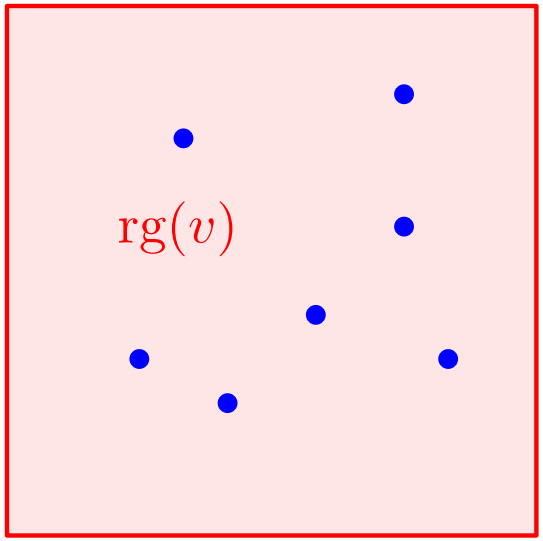


Compressed Quadtrees



compressed node u

$\mathrm{rg}(u)$ is an annulus

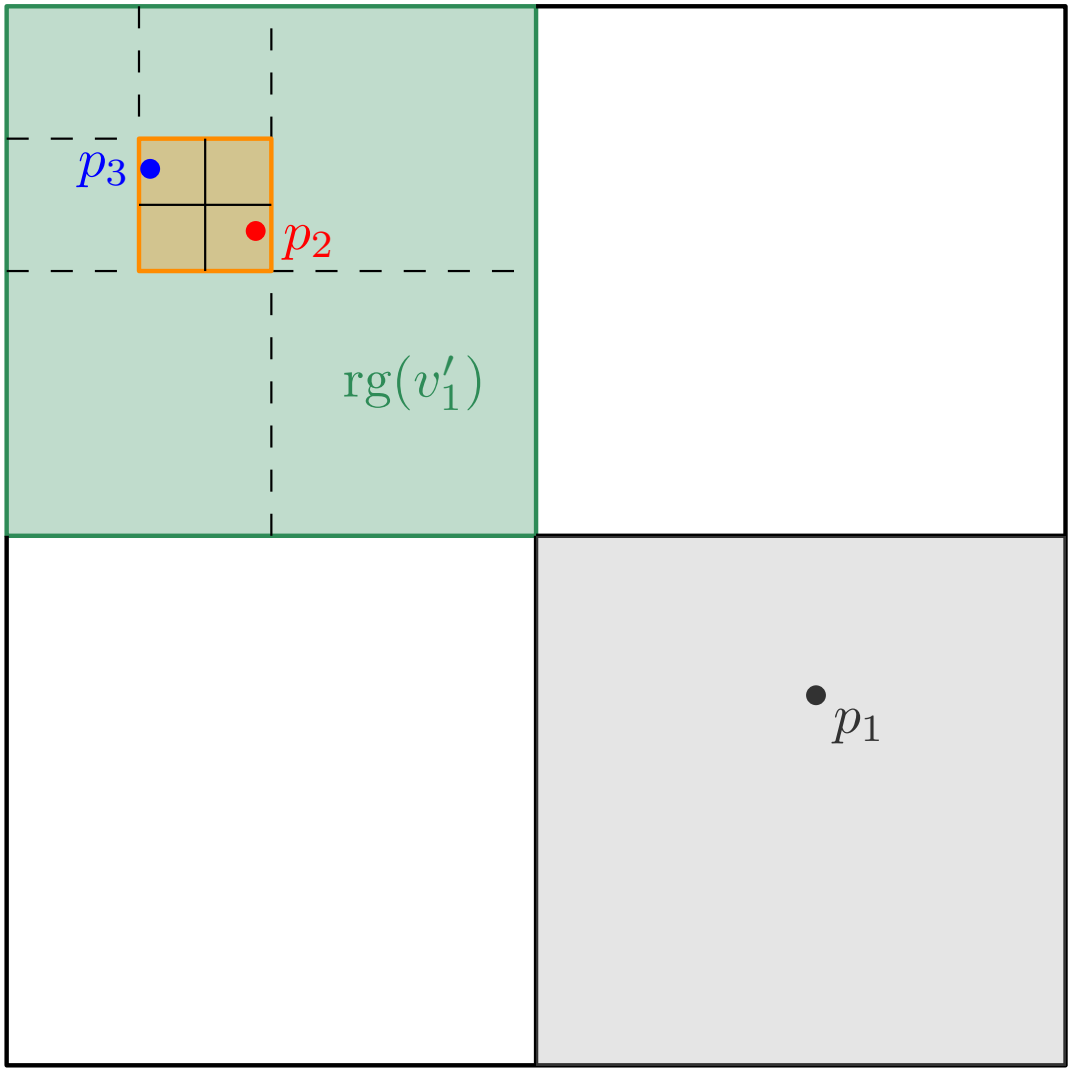


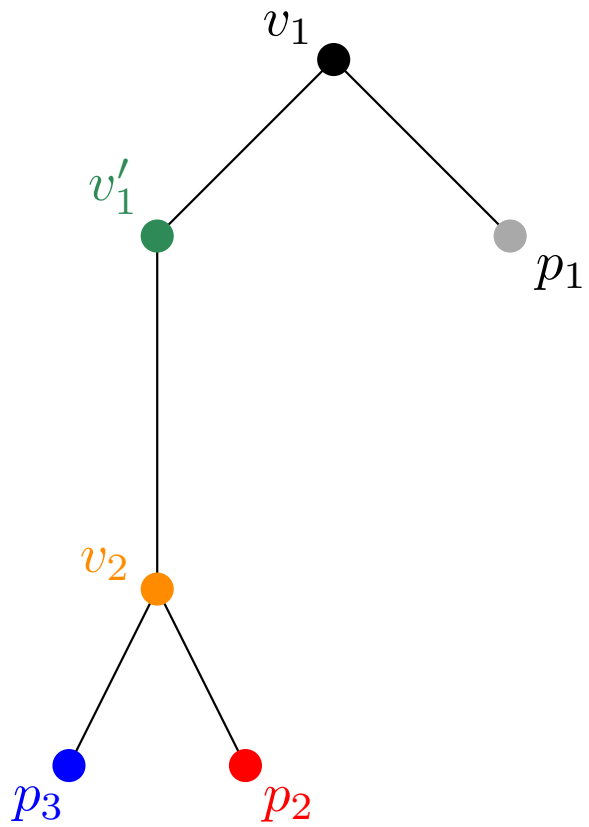
ordinary node v

$\mathrm{rg}(v) = \mathrm{box}(v)$

- In the quadtree, we replace each path of length $\geqslant 2$ by 2 nodes connected with an edge.
- Now the region $\mathrm{rg}(v)$ associated with a node v is not necessarily a box: it can also be an empty annulus, which is the difference of two quadtree boxes.

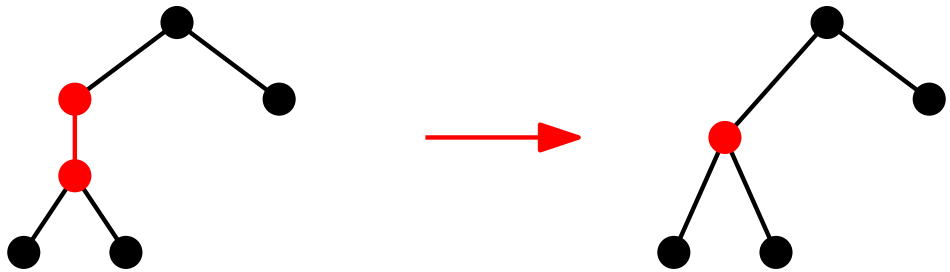
Compressed Quadtrees





Size of a Compressed Quadtree

- The *degree* of a node in a rooted tree is its number of children.
- Only compressed nodes have degree 1.
- Suppose I replace each each compressed node and its child with a single node.



- I obtain a rooted tree with n leaves, and such that each internal node has degree $\geqslant 2$.
  - n leaves because each leaf records an input point.
- This new tree has at most $n-1$ *internal* nodes.
- So the new tree has $2n-1$ nodes, and the original compressed quadtree has less than $4n$ nodes.

Size of a Compressed Quadtree

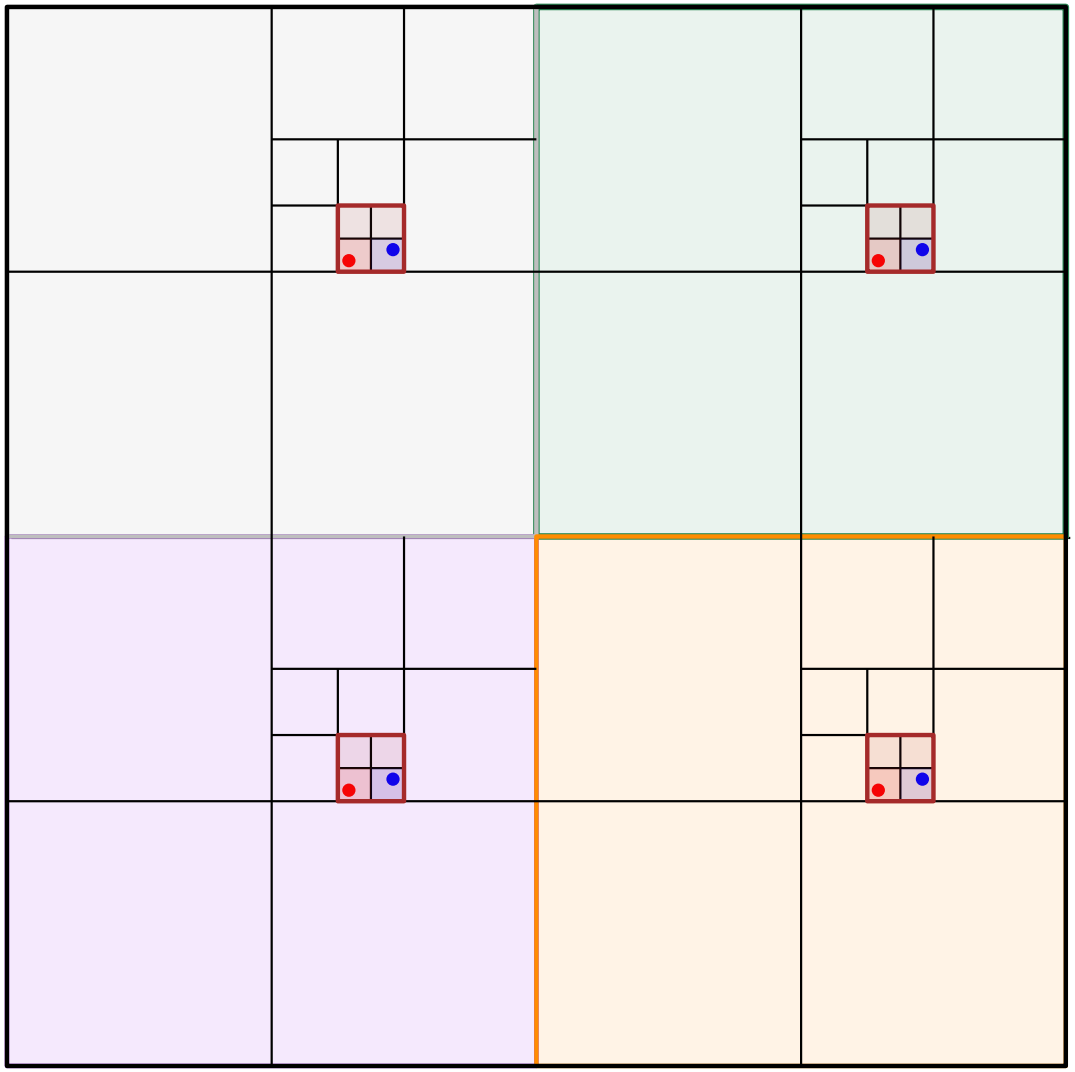
- I just proved the following:

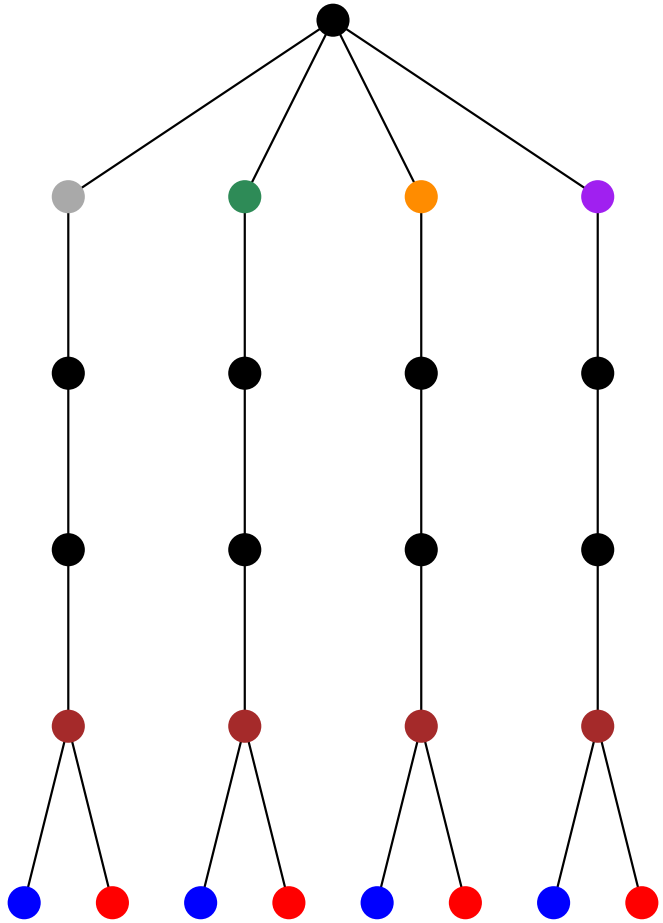
Lemma

The compressed quadtree $\mathcal{T}$ recording a set P of n points has less than $4n$ nodes.

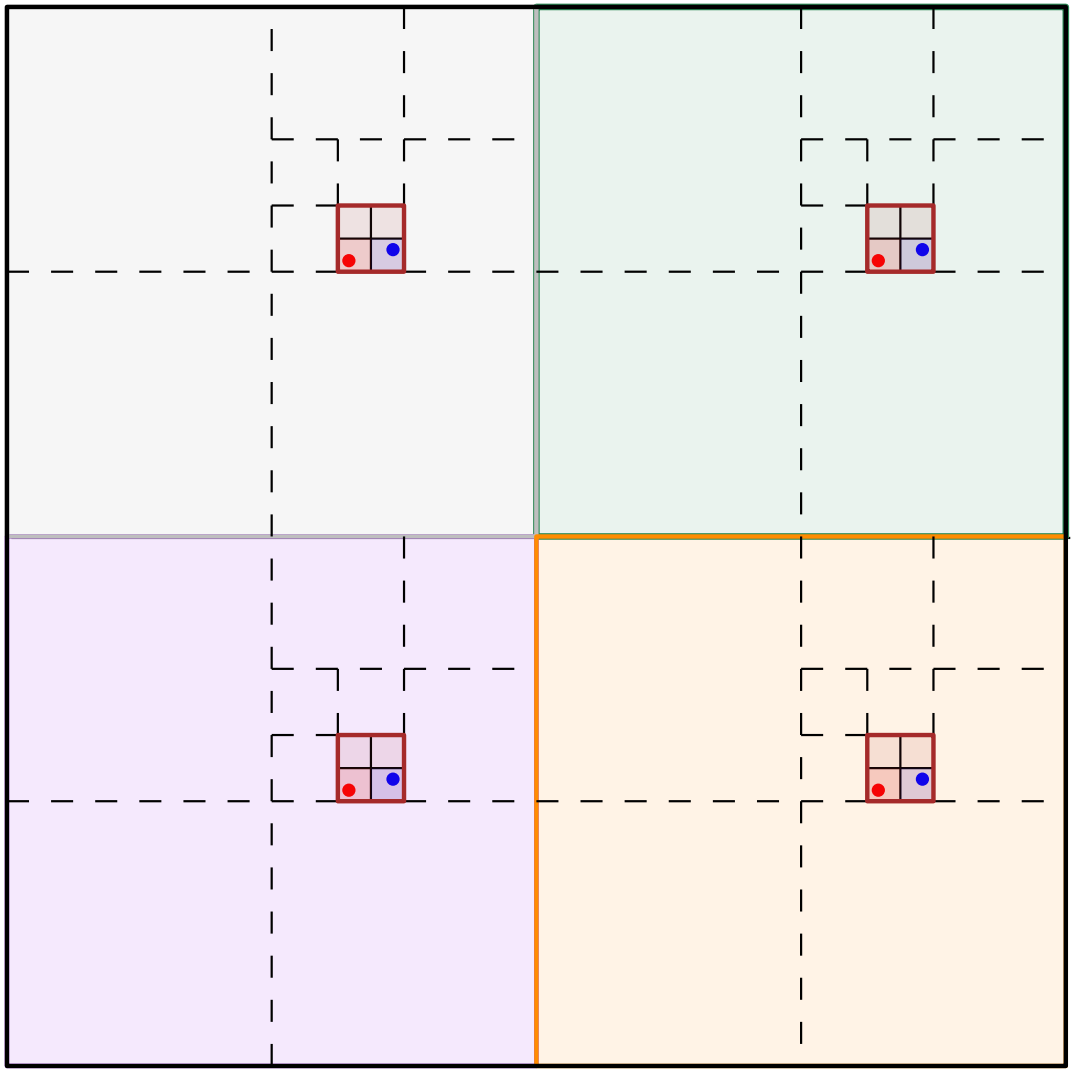
- Difference with a (non-compressed) quadtree:
- The bound was $O(n \log \Phi)$ nodes.
- It goes to infinity when Φ goes to infinity.
- See next 2 slides.

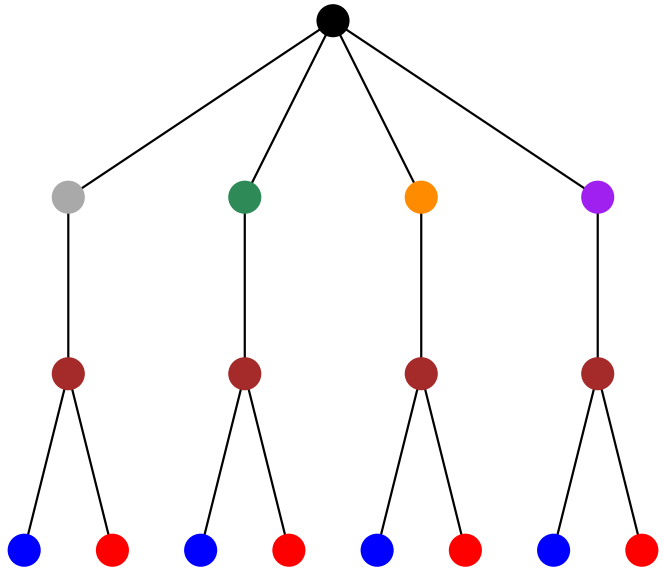
Size of a Quadtree





Size of a Compressed Quadtree





Construction

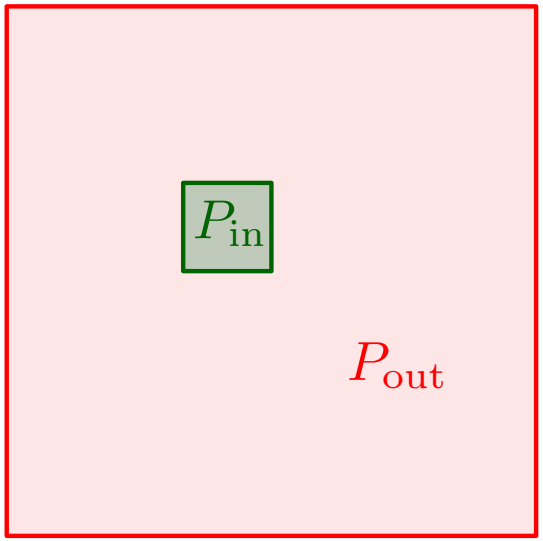
- We can construct a compressed quadtree in time $O(n^2)$ in a top down manner, each new node taking $O(n)$ time.
- A better time bound is achievable.

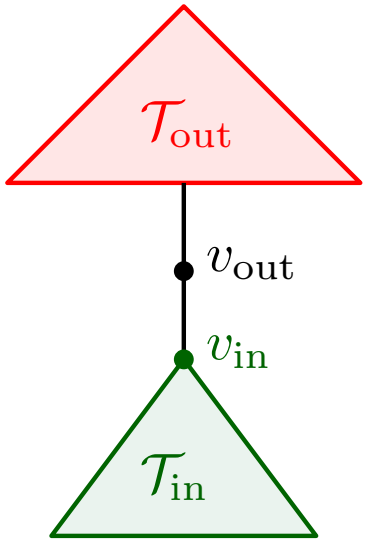
Lemma

The compressed quadtree recording a set of n points in $\mathbb{R}^d$ can be constructed in $O(n \log n)$ time, when $d = O(1)$.

- The idea of the construction is given on next slide.
- I will not present it in details as it would require to cover another chapter of the textbook.

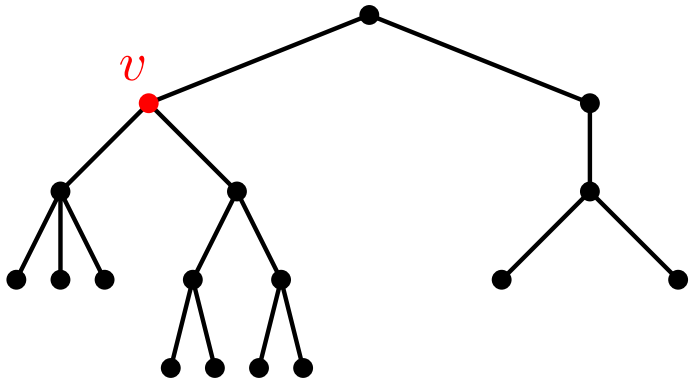
Construction



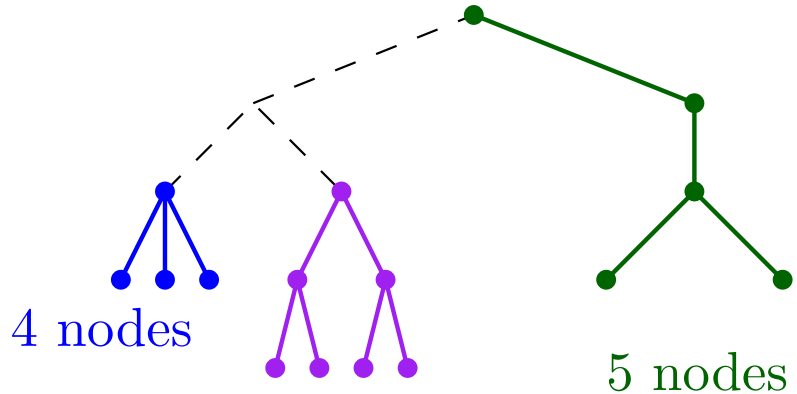


- It is a divide-and-conquer approach.
- We can find in $O(n)$ time a quadtree box such that the set P_{in} of pints inside it, and the set P_{out} of points outside it, have size $\Omega(n)$.
- We separately compute compressed quadtrees for these two sets and connect them using a new vertex v_{out}.

Tree Separator



$\mathcal{T}$ has 18 nodes



Definition (Separator)

Let $\mathcal{T}$ be a tree with n nodes. A separator is a node v such that $\mathcal{T} - v$ is a forest, each tree in this forest having at most $n/2$ vertices.

Lemma

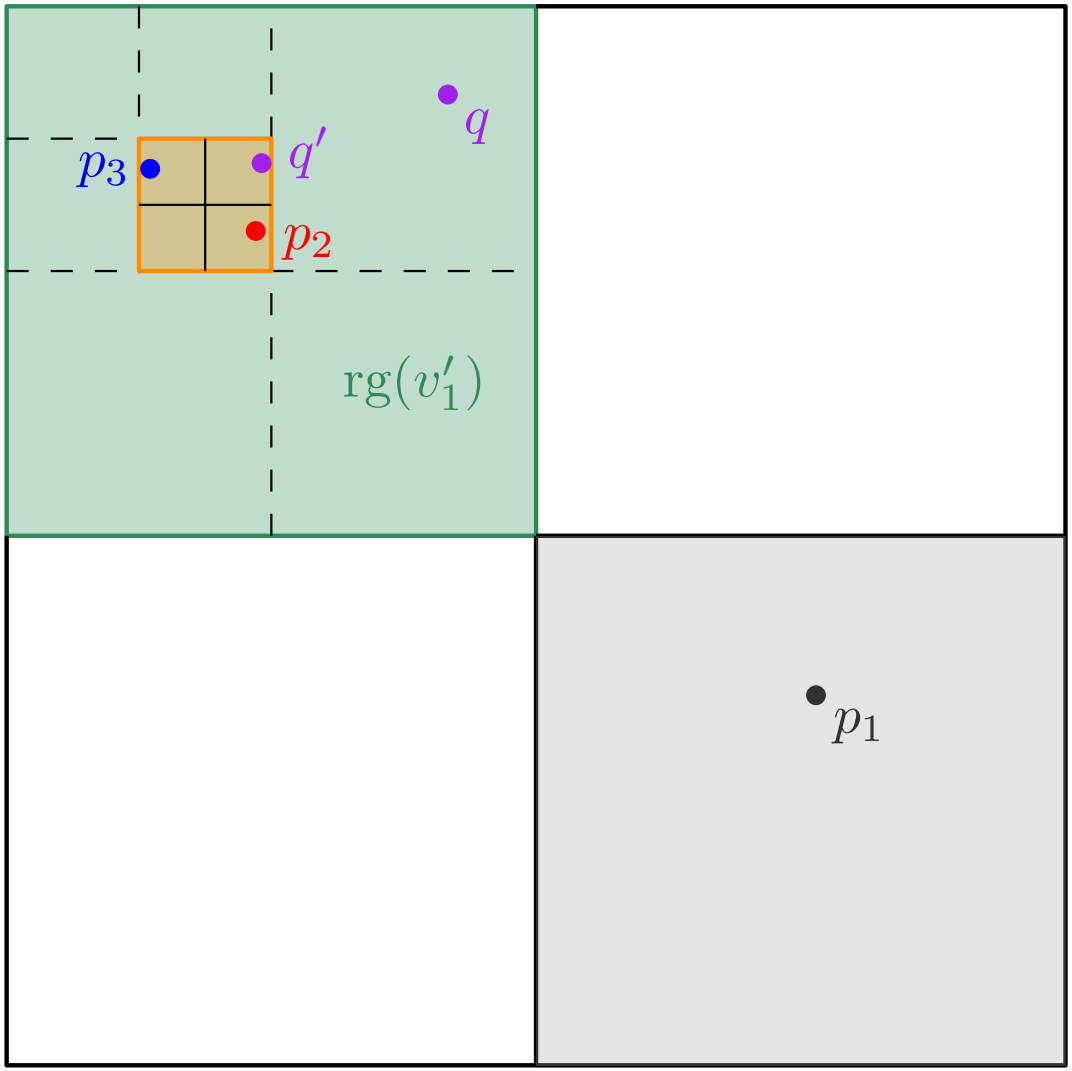
Every tree has a separator, and it can be computed in linear time.

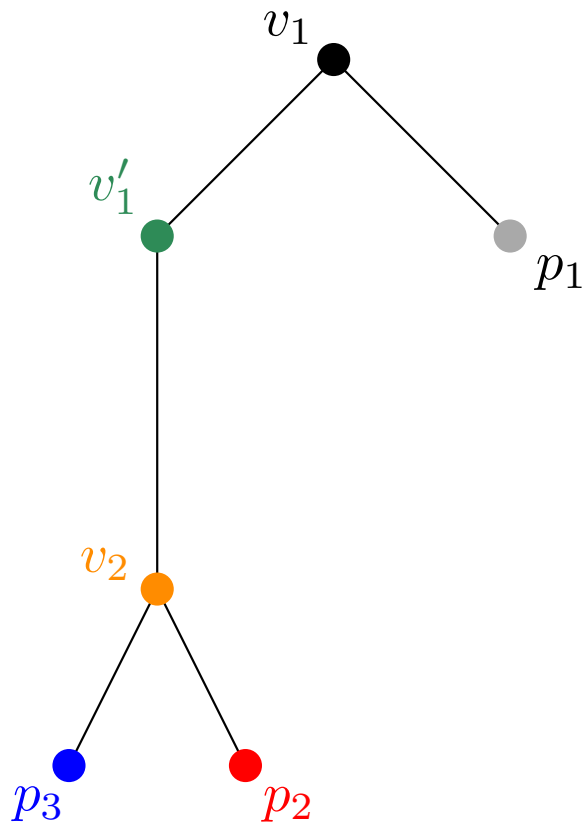
Tree Separator: Proof

- Take the lowest node v such that the subtree rooted at v has size at least $n/2$. This is a separator.

- Construction time:
- Proceed in a bottom-up manner, starting from the last level. While going up, maintain the size of the current subtree. When it becomes at least $n/2$, we have found a separator.
- This can be done in linear time. (For instance, do a breadth-first search, and proceed in the opposite order.)

Point Location in a Compressed Quadtree





Point Location in a Compressed Quadtree

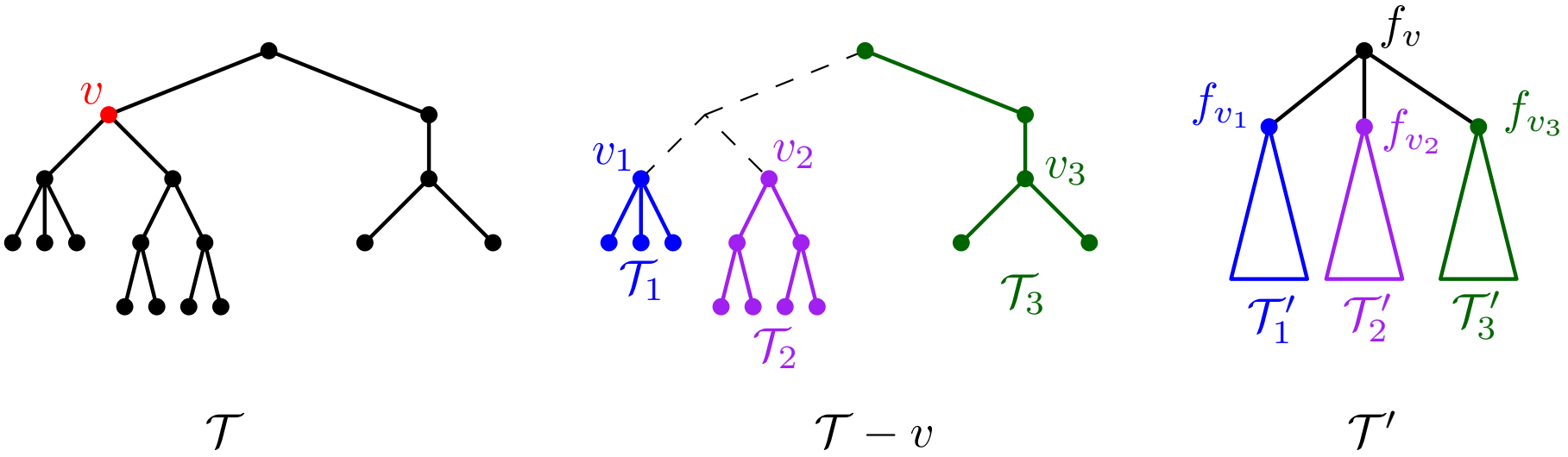
Lemma

Let $\mathcal{T}$ be a compressed quadtree over n points in $\mathbb{R}^d$, where $d = O(1)$. We can preprocess $\mathcal{T}$ in $O(n \log n)$ time such that, for any query point q, we can find in $O(\log n)$ time the lowest node in $\mathcal{T}$ that contains it.

- Example in previous slide:
- For query q, this node is v_1'.
- For query q', this node is v_2.

- Approach: We construct a new tree $\mathcal{T}'$.
- Its root is a node f_v, where v is a separator of $\mathcal{T}$.
- Recurse on each tree of the forest $\mathcal{T} - v$.

Point Location in a Compressed Quadtree



- In order to perform point location:
- Check whether q, belongs to the region covered by each tree of $\mathcal{T} - v$, or $\mathrm{rg}(v)$.
  - Choose the one that yields the lowest node in $\mathcal{T}$.
- Recurse on the corresponding subtree of $\mathcal{T}'$.
- Each step is done in constant time, and the size of the current subtree of $\mathcal{T}'$ is divided by at least 2.
- So it takes $O(\log n)$ time.