



Dyalog North America – 29-30 September 2025

What can Vectorised Trees Do for You?

Asher Harvey-Smith

Me

Freshly graduated 🎓

(Computer Science MEng, University of Warwick)



Me

Freshly graduated 🎓

(Computer Science MEng, University of Warwick)

Intern



Me

Freshly graduated 🎓

(Computer Science MEng, University of Warwick)

Intern

Senior Intern



Me

Freshly graduated 🎓

(Computer Science MEng, University of Warwick)

Intern

~~Senior Intern~~

Proper employee 🎓





Search Ctrl + K

Exploring Things with Dyalog APL

Trees

Introduction

Representing Trees

Parent Vectors

Relating the Depth and Parent Vectors

Forests

Deleting Nodes

Bottom-Up Accumulation

Working with `JSON`

Combinatorics

Enumerative Combinatorics



```
f(c''index:' depths:' data:'),(i#depths) depths data
```

index:	0	1	2	3	4	5	6	7	8	9
depths:	0	1	2	2	1	2	3	3	3	1
data:	a	b	e	f	c	g	h	i	j	d

At this point it's helpful to make a small mental shift. We are drawing a one-to-one correspondence between node and indices into these vectors, using the indices as unique identifiers. We will use this so frequently that making the distinction explicit becomes tiring, so from now on, we will often refer to 'the node associated with index i ' simply as 'node i '.

Labelling each node of our tree with its corresponding index reveals an interesting pattern.

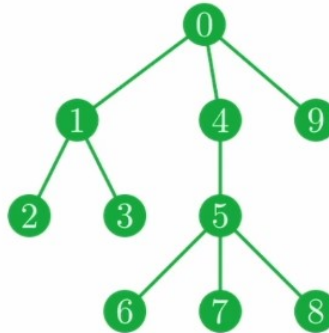


Fig. 4 The same tree, with nodes labelled with their index into the `depth` and `data` vectors.

Contents

Preliminary Definitions

The Nested Representation

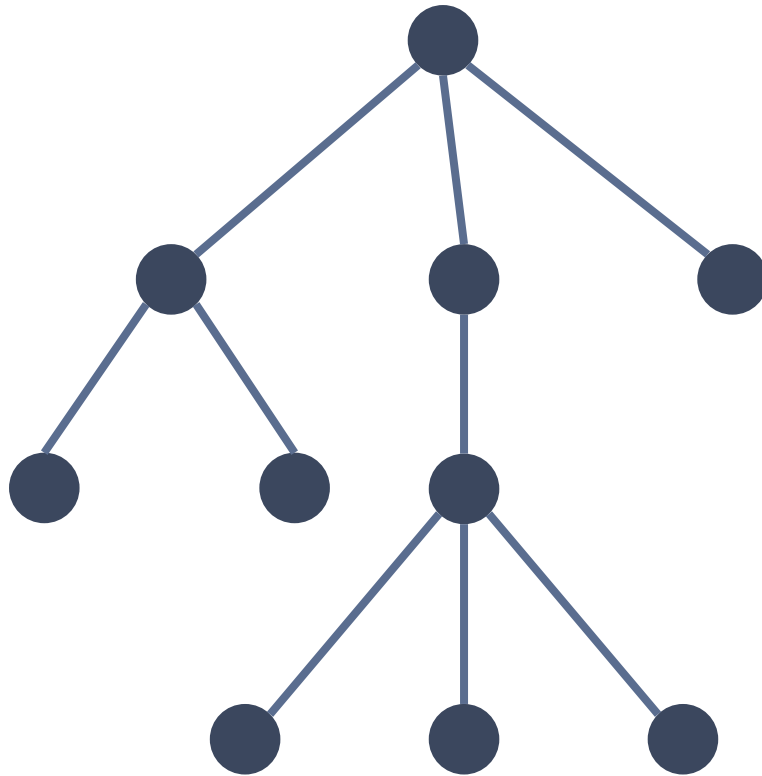
The Depth Vector Representation

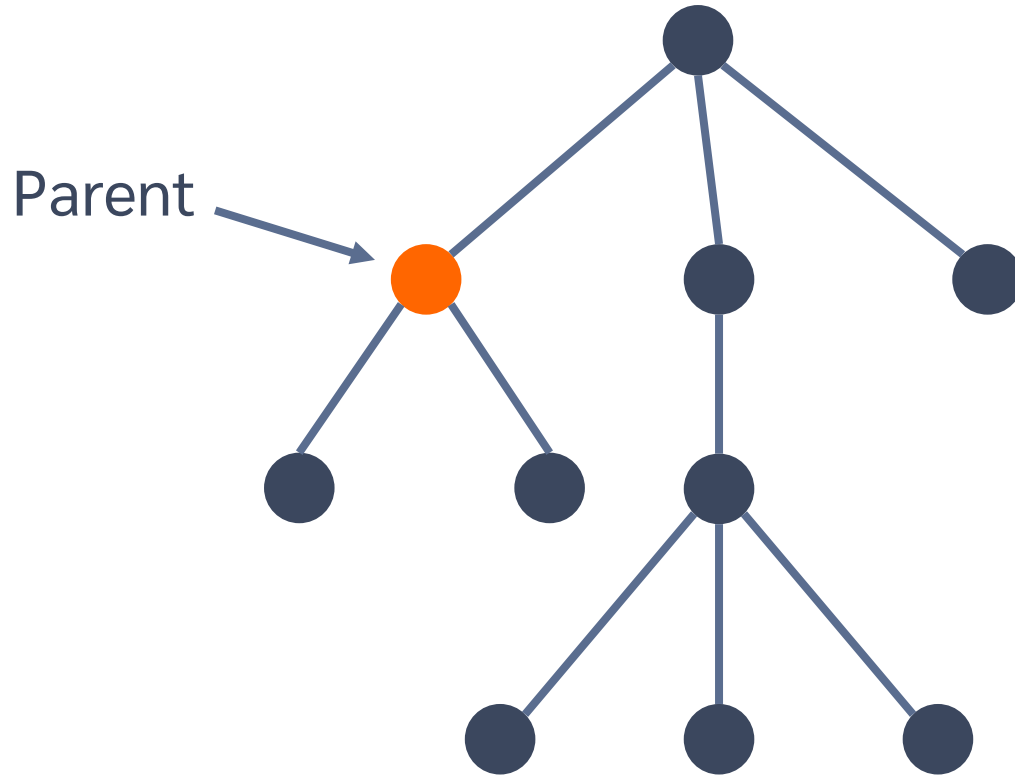
The Path Matrix Representation

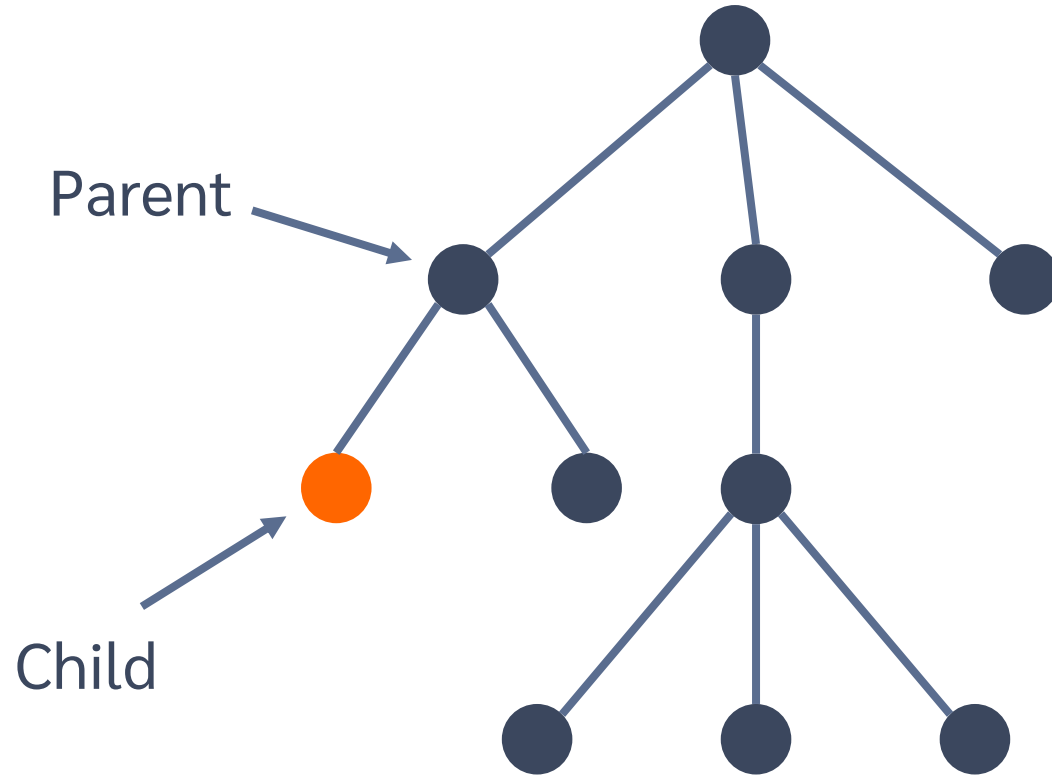
The Parent Vector Representation

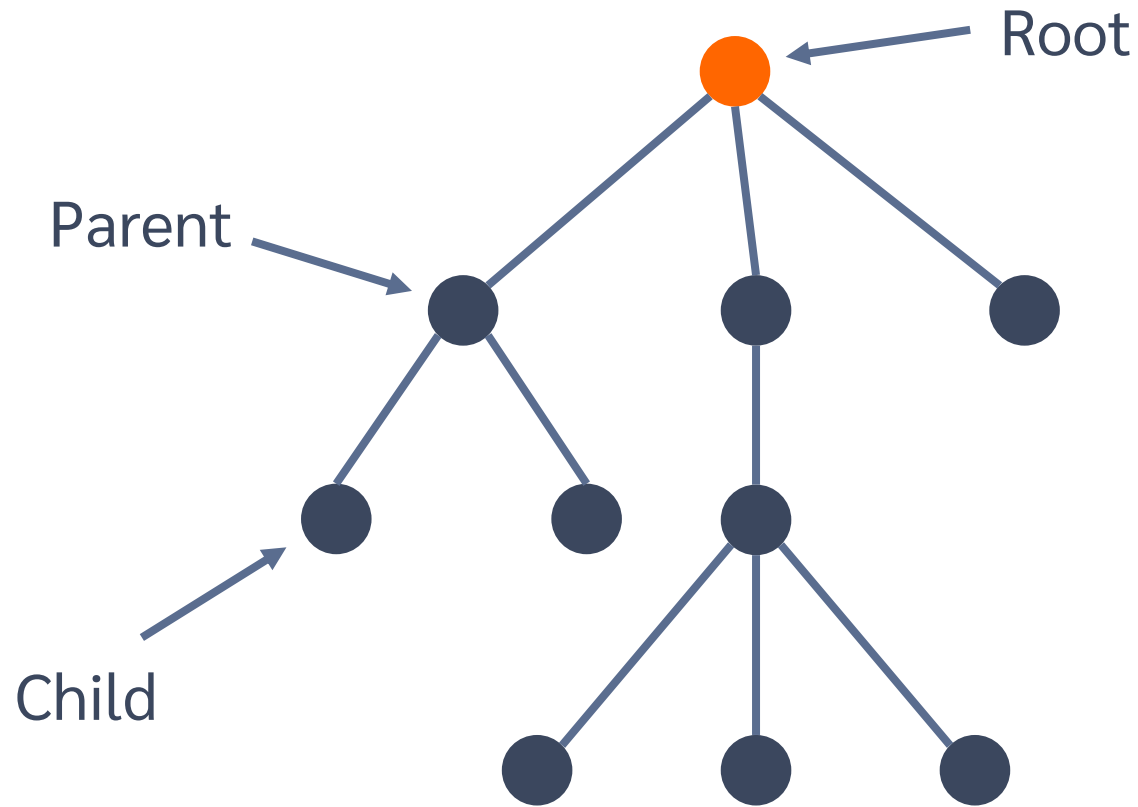
Challenge

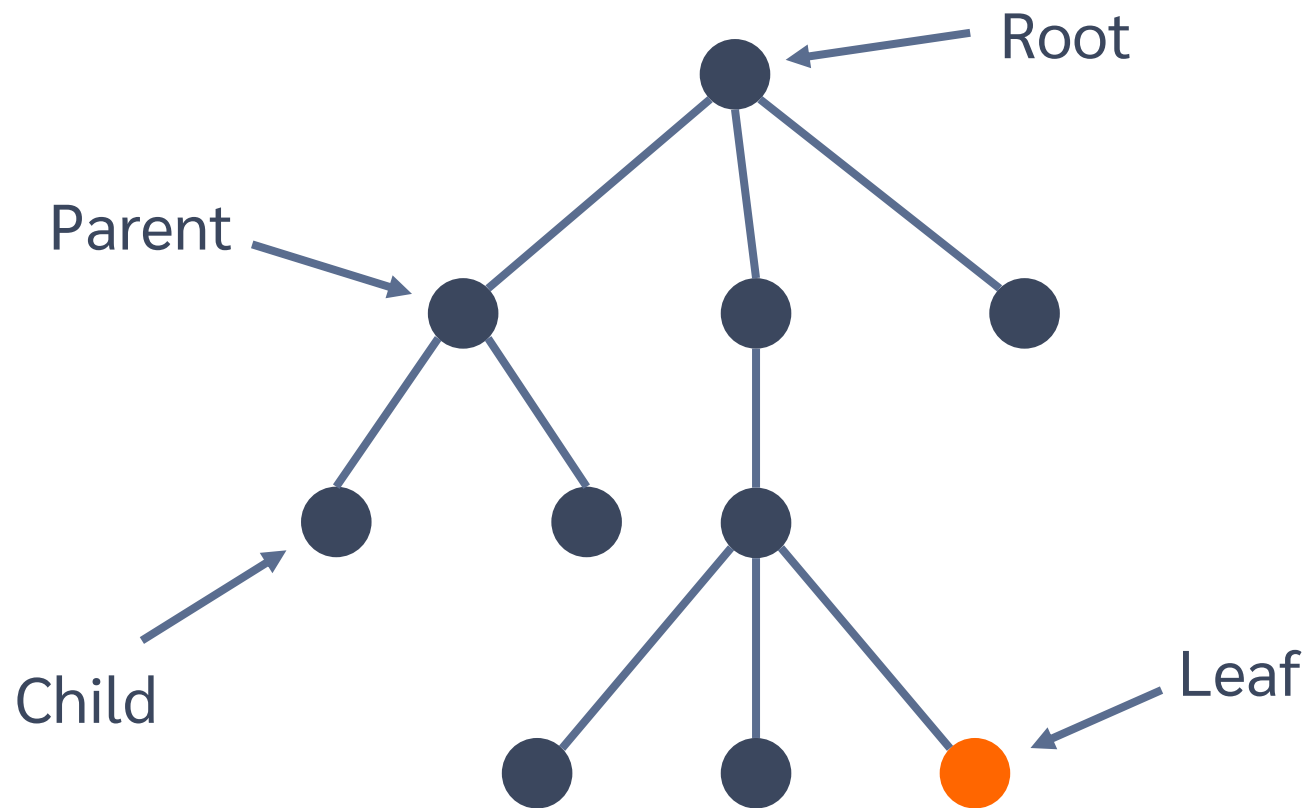


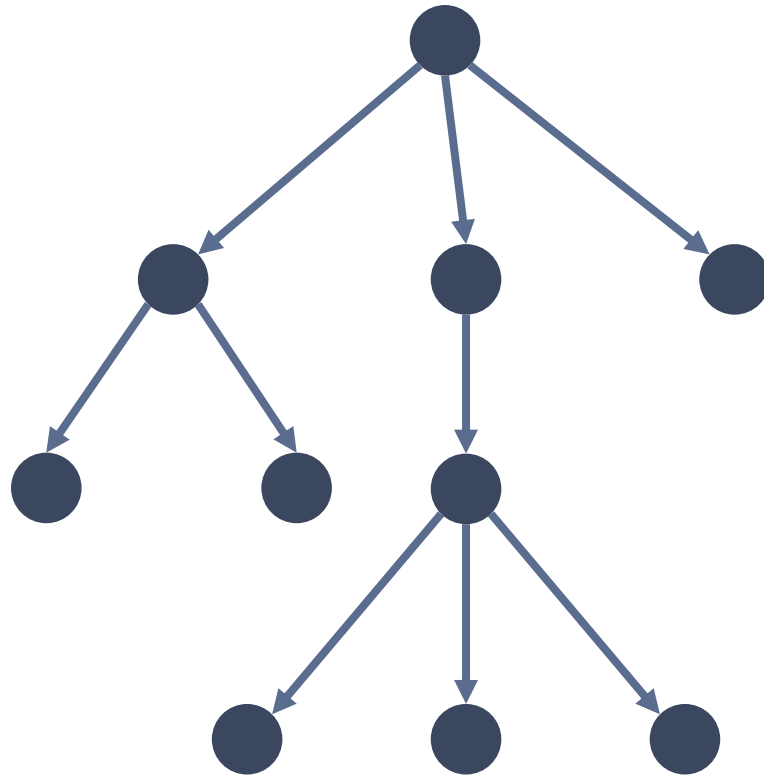


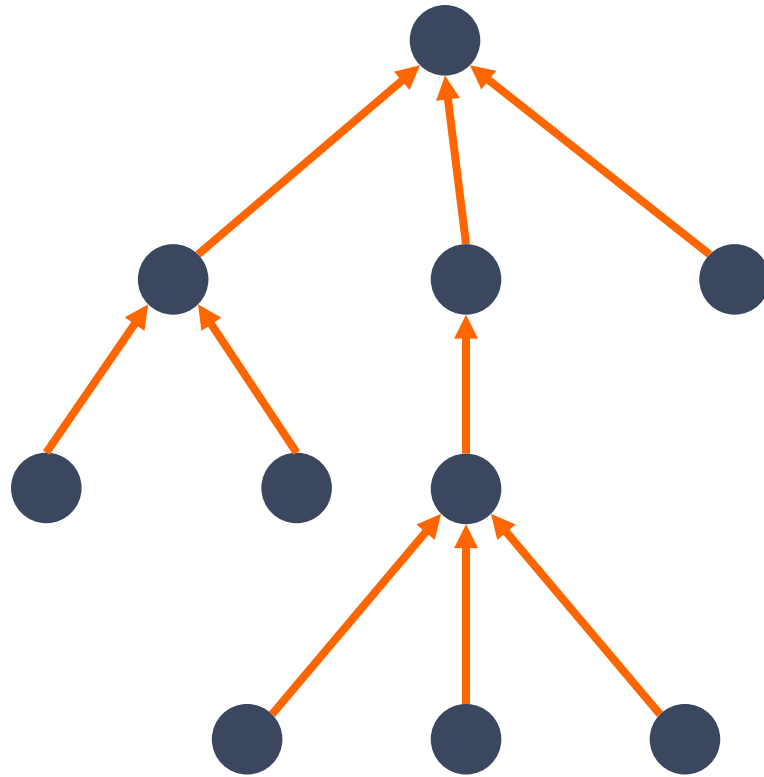


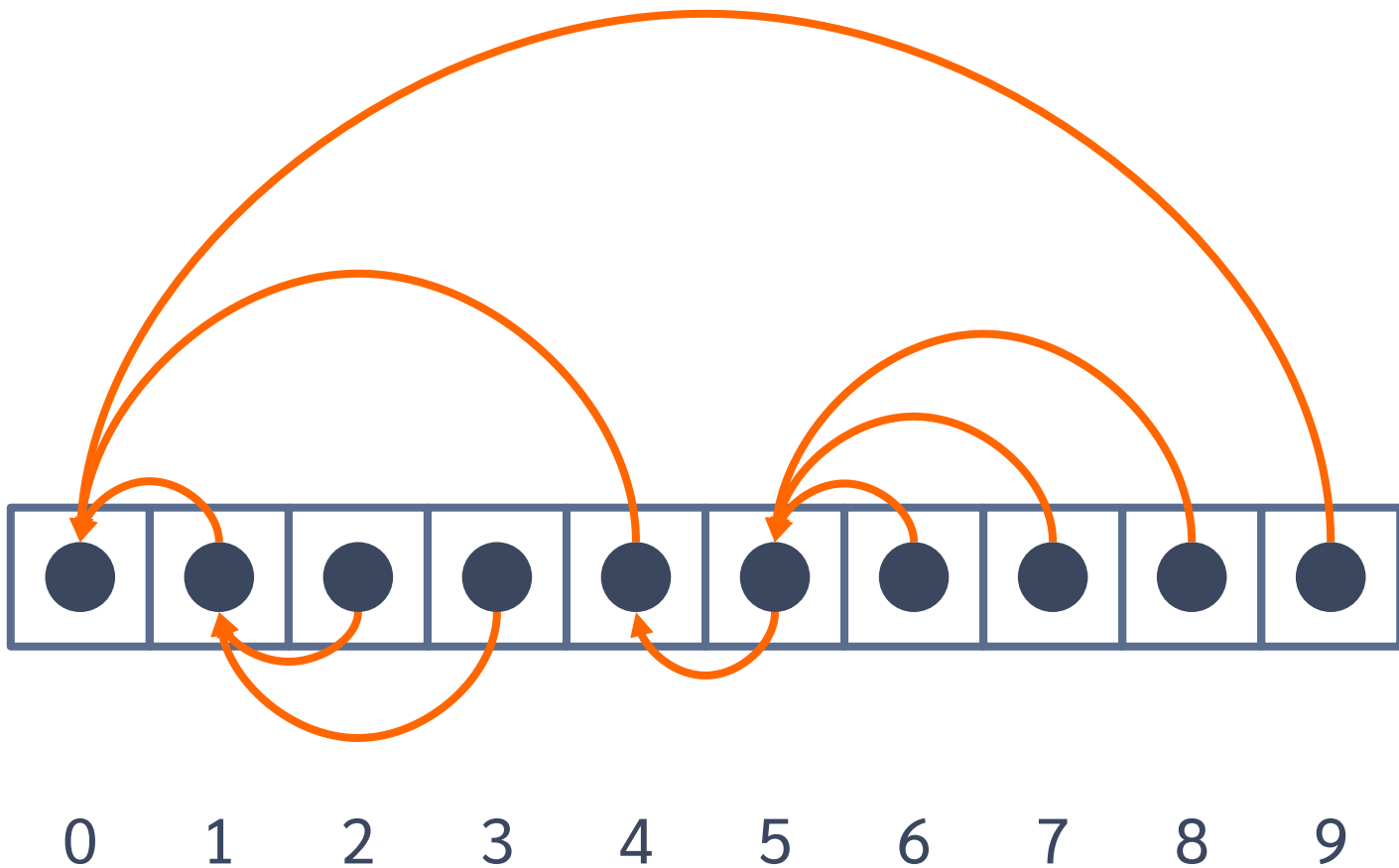


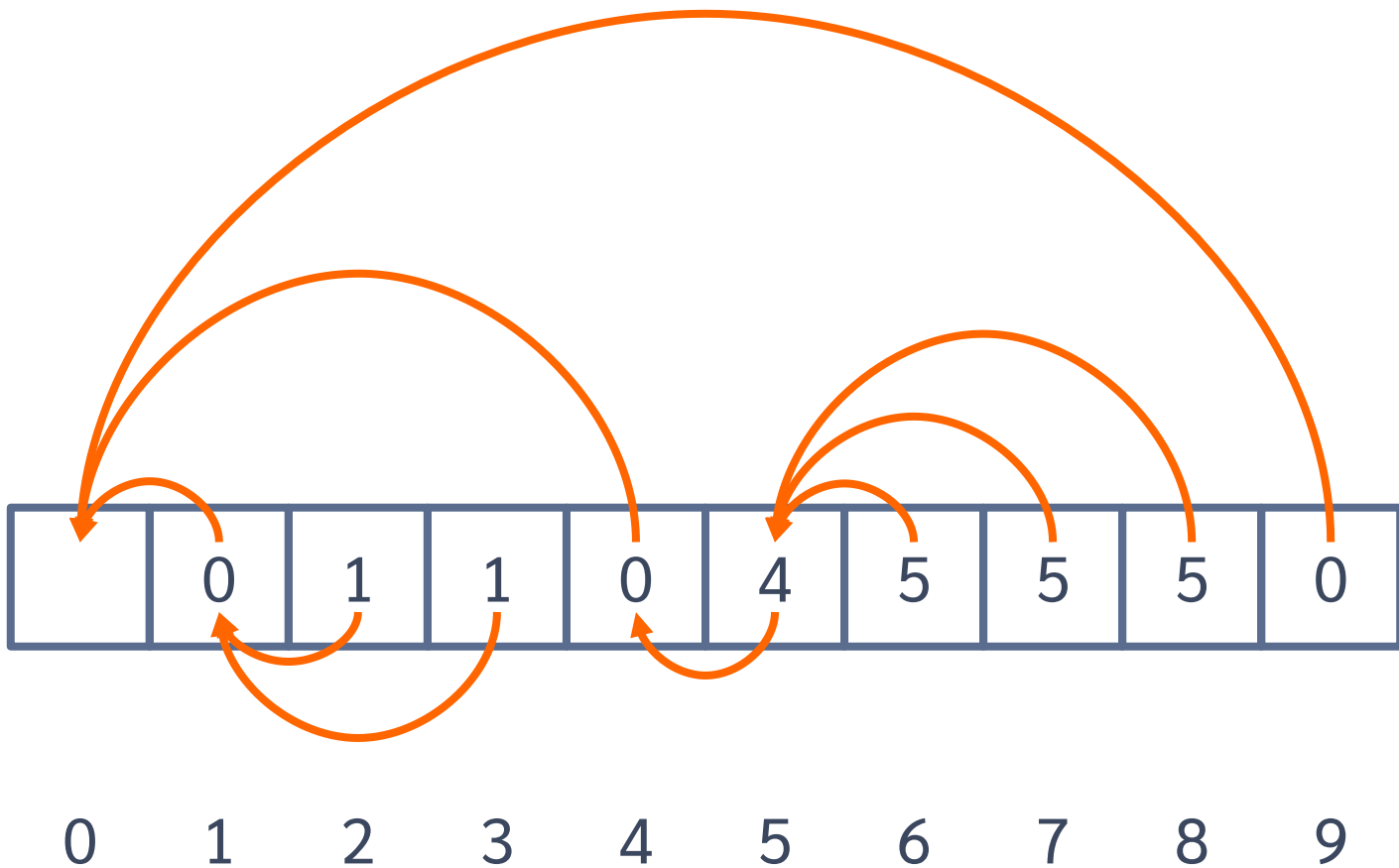


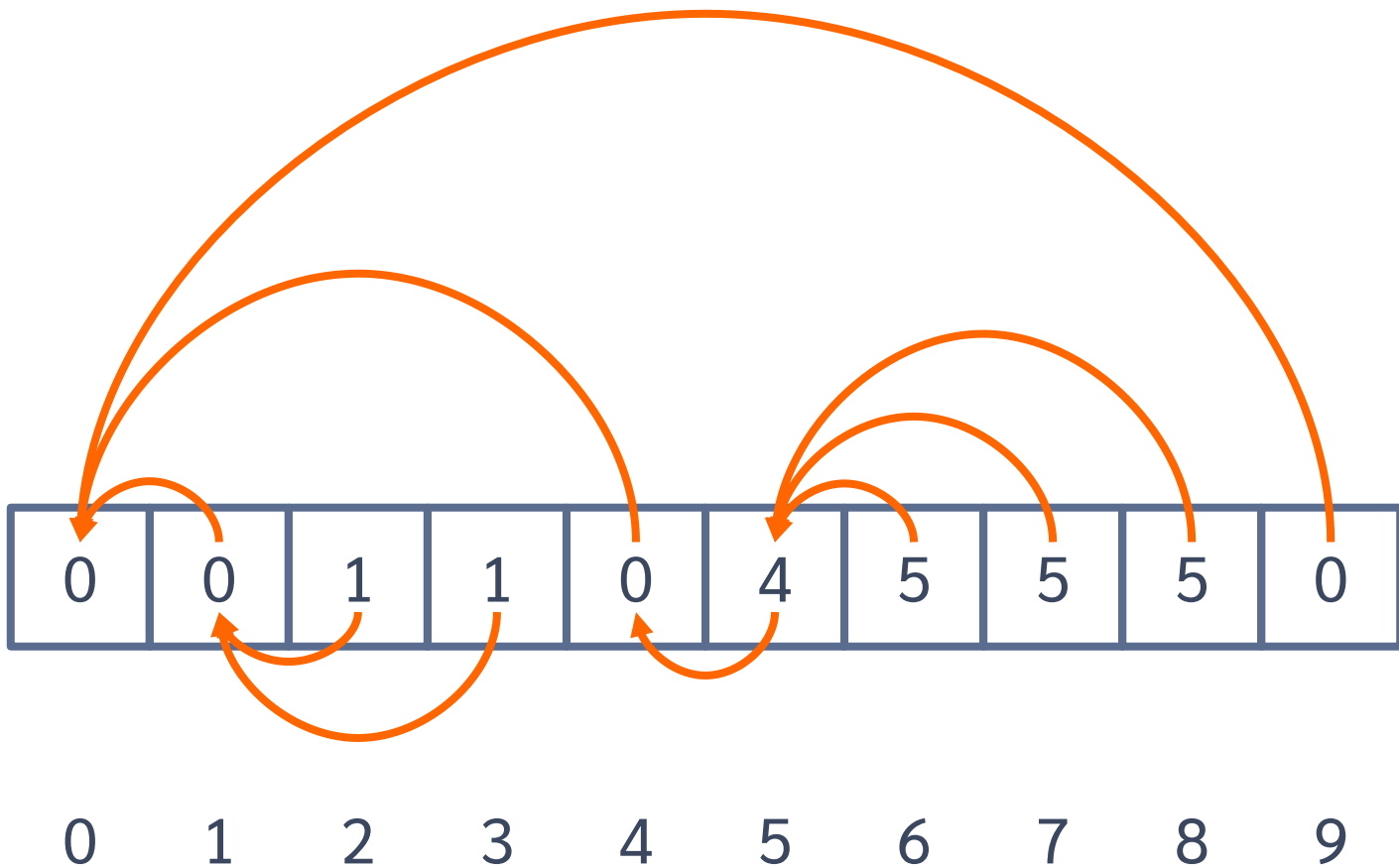




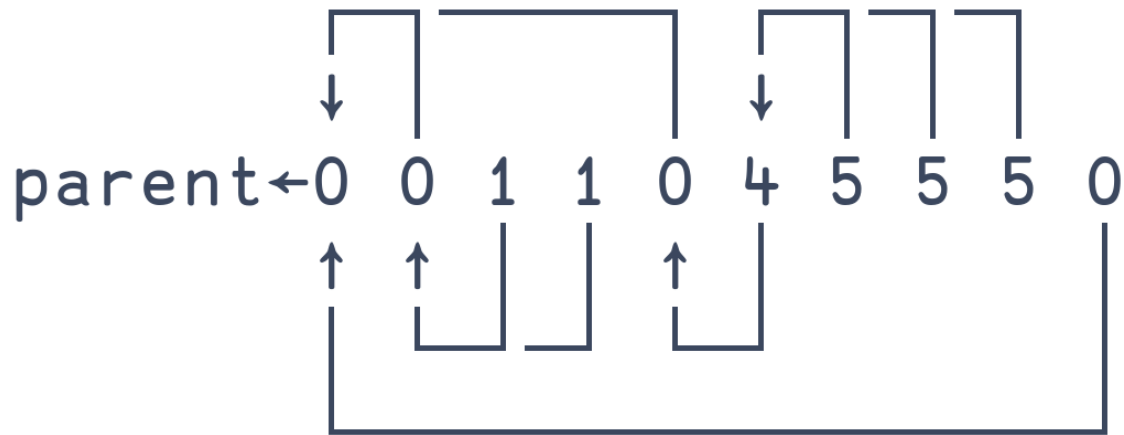
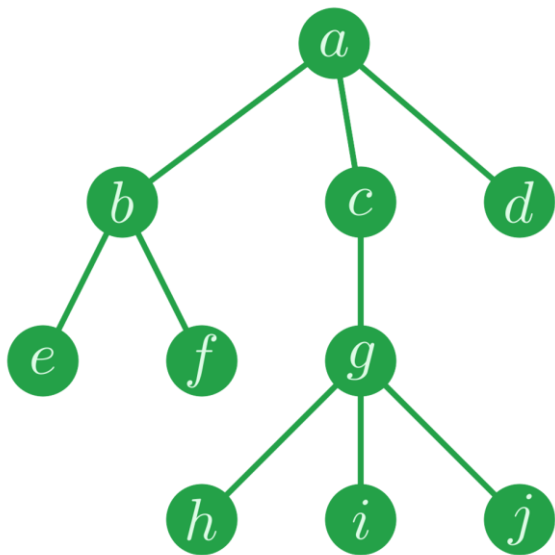




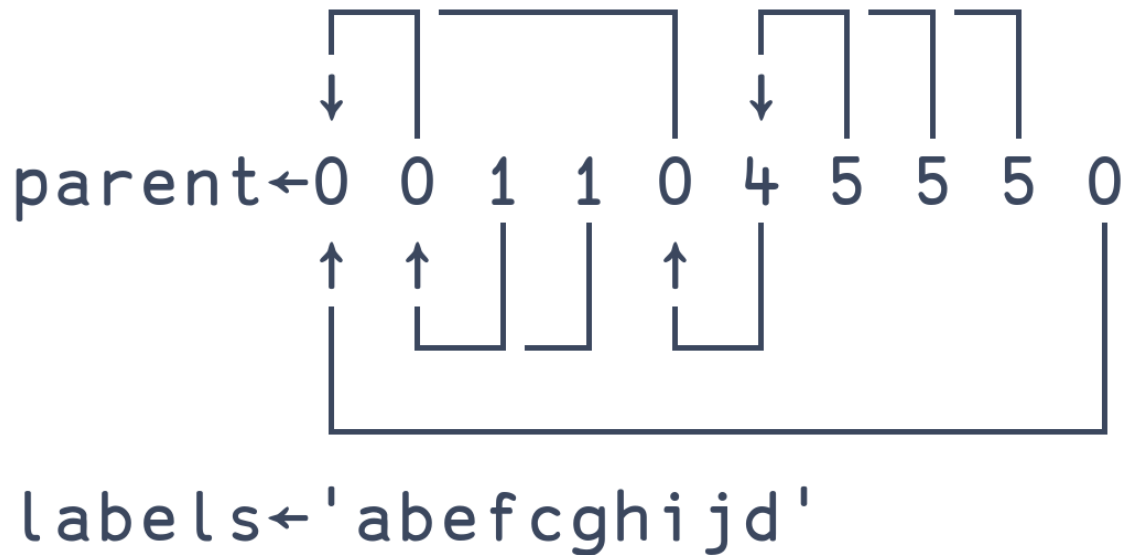
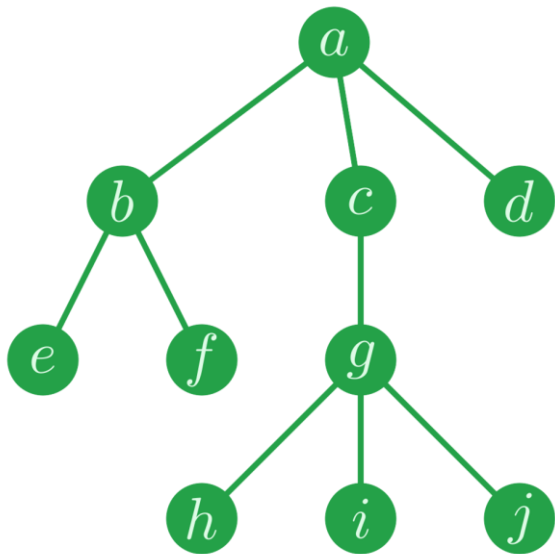




Parent Vectors



Parent Vectors



Parent Vectors

Let's see some code

When Are Parent Vectors Useful?

- Large ✓
 - Larger trees have more performance gains from vectorisation.

When Are Parent Vectors Useful?

- ❖ Large ✓
 - ❖ Larger trees have more performance gains from vectorisation.
- ❖ Irregular ✓
 - ❖ For trees with a predictable shape, you can get away with ad-hoc approaches.

When Are Parent Vectors Useful?

- ❖ Large ✓
 - ❖ Larger trees have more performance gains from vectorisation.
- ❖ Irregular ✓
 - ❖ For trees with a predictable shape, you can get away with ad-hoc approaches.
- ❖ Traversing bottom-up ✓
 - ❖ Parent pointers take us up the tree.
 - ❖ You can go bottom-up more often than you think!

Case Study: Co-dfns & Static Analysis

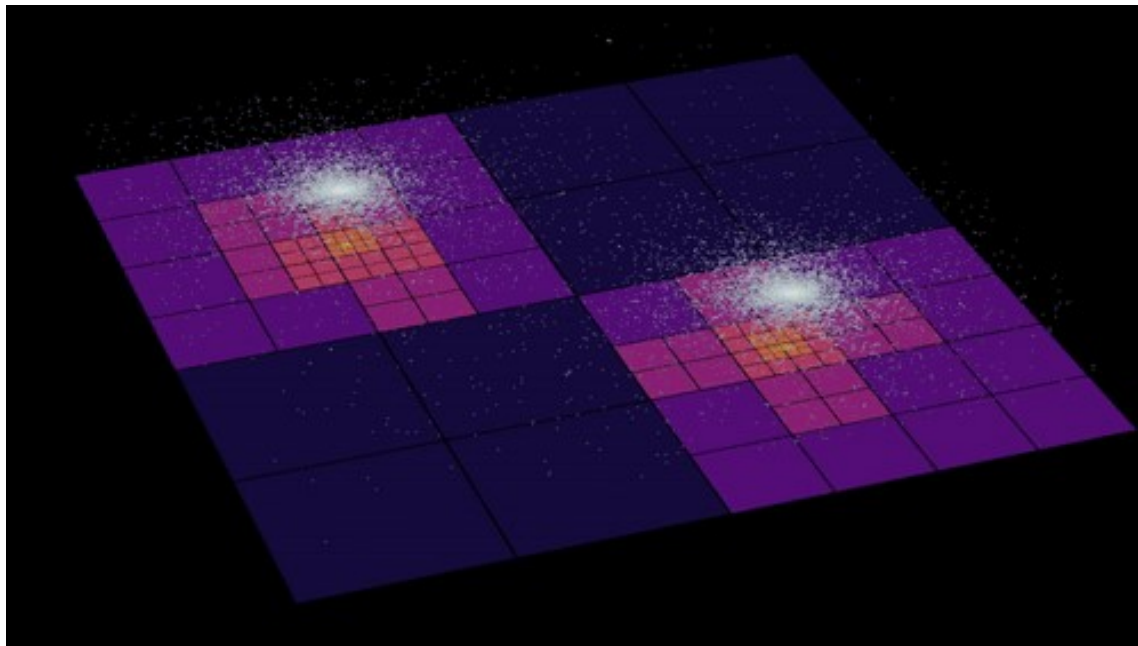
- ✧ Syntax tree is represented by a parent vector
 - ✧ Large ✓
 - ✧ Irregular ✓
 - ✧ Bottom-up traversals ✓
- ✧ Largest serious projects using the technique (that I know of)

Case Study: File System Queries

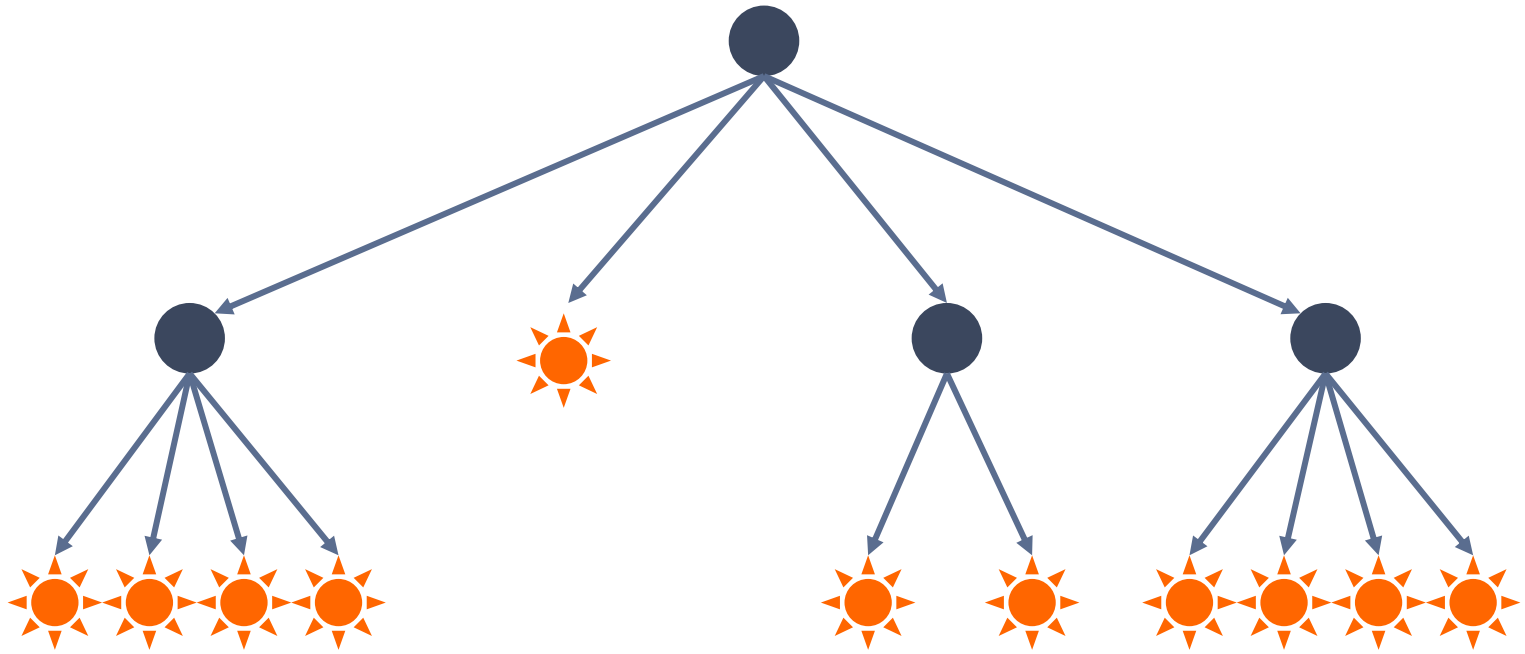
- File systems are* trees
 - *pedants will bring up file system links, we will ignore the pedants
- They can be large ✓ and irregular ✓
- You can do all sorts of queries very easily on a parent vector
 - Using bottom-up traversals ✓

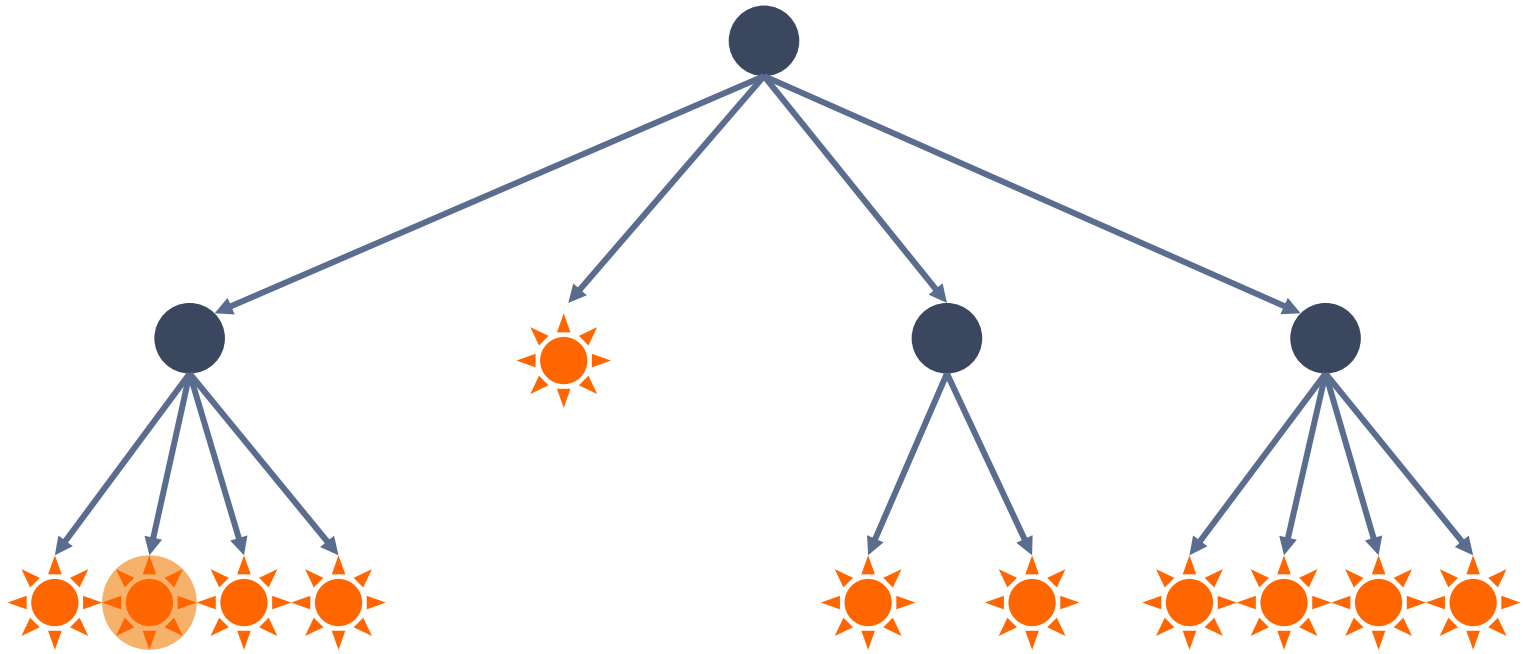
```
desktop/  
├── drawer  
│   ├── document.md  
│   └── notebook.ipynb  
├── empty-mug  
├── my-wallet  
│   ├── card-slot  
│   │   └── id  
│   └── coin-pouch  
│       ├── cent  
│       ├── dime  
│       ├── nickel  
│       └── quarter  
├── pen.cil  
├── plant-pot  
│   └── aloe.vera  
├── .secret-compartment  
│   ├── chocolate.bar  
│   └── ufo.evidence  
└── todo.txt
```

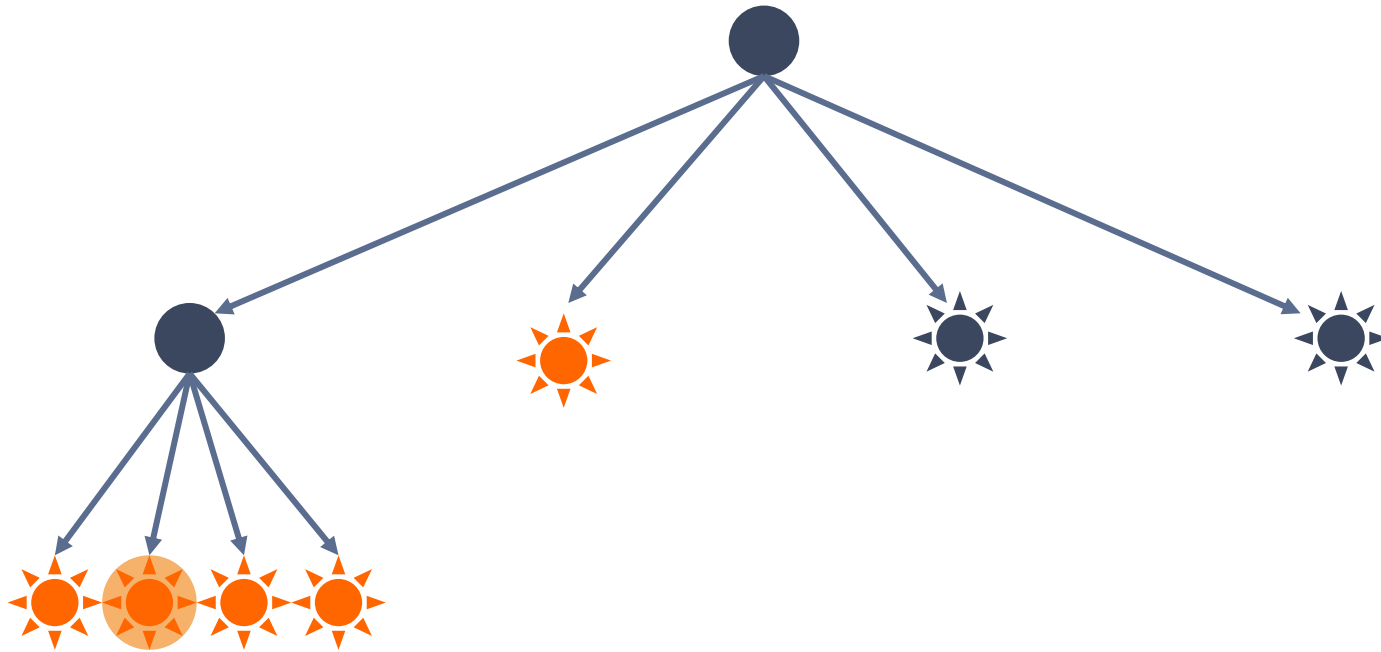
Case Study: Barnes-Hut Simulation

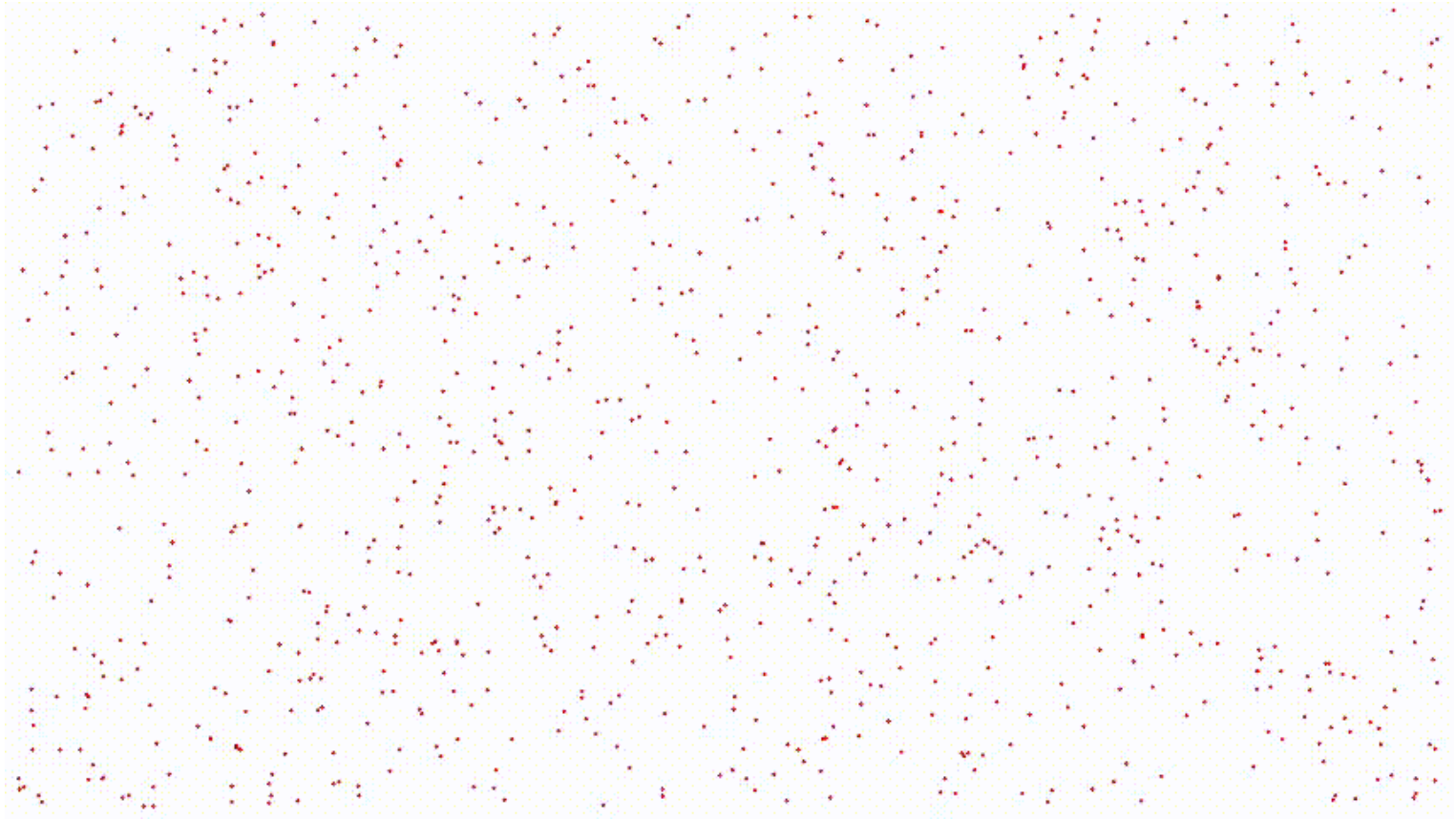


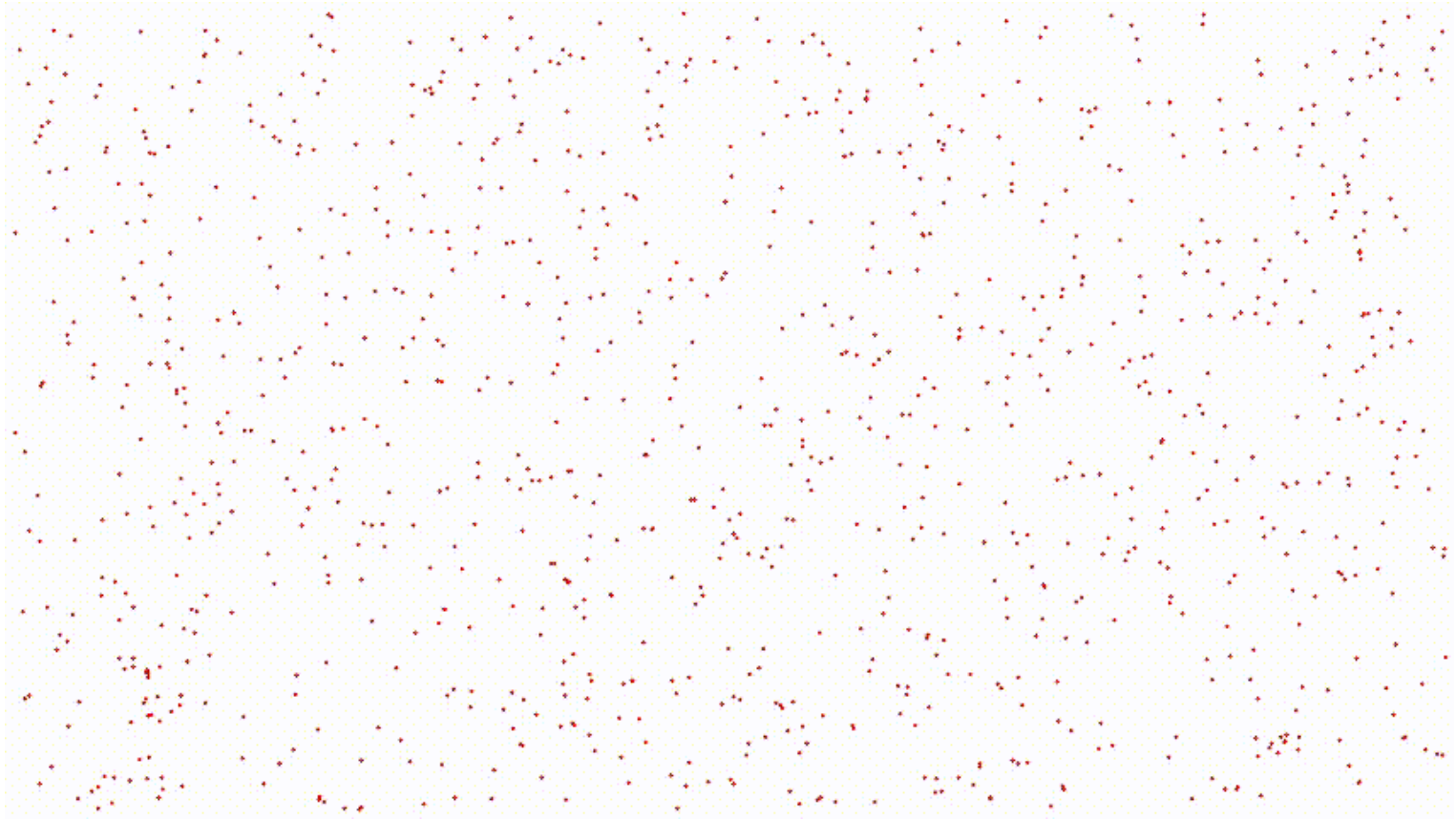
By Perossello - Own work, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=147365688>











Case Study: Barnes-Hut Simulation

- ◆ Possibility 1: I did it wrong

Case Study: Barnes-Hut Simulation

- ◆ Possibility 1: I did it wrong
- ◆ Possibility 2: Interpreter overhead is overwhelming the speedup
 - ◆ Top down traversal for each particle, without any sharing

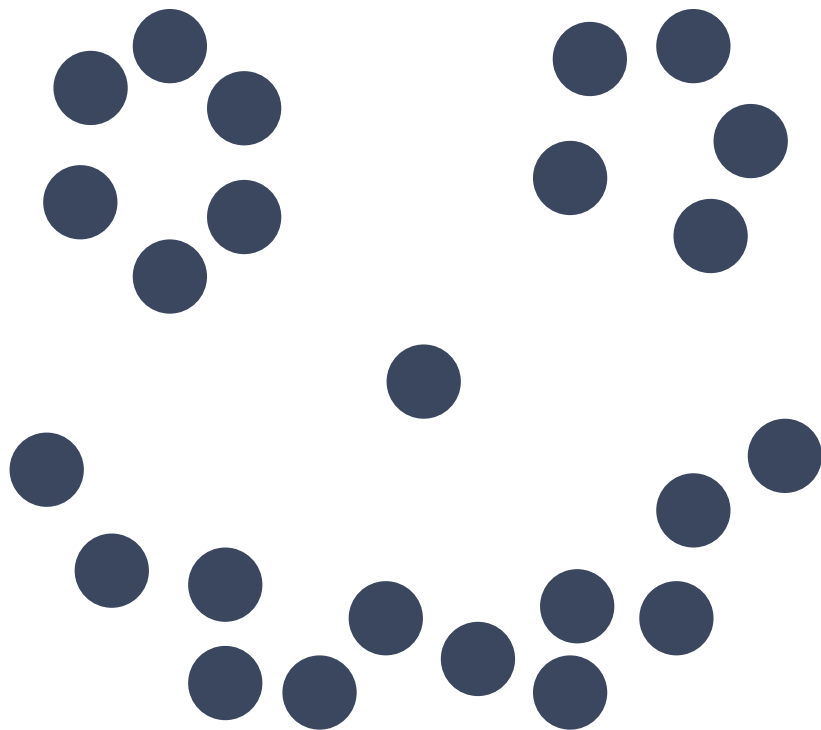
Case Study: Barnes-Hut Simulation

- ◆ Possibility 1: I did it wrong
- ◆ Possibility 2: Interpreter overhead is overwhelming the speedup
 - ◆ Top down traversal for each particle, without any sharing
- ◆ Maybe with more particles it is faster
 - ◆ My laptop is not strong enough to try

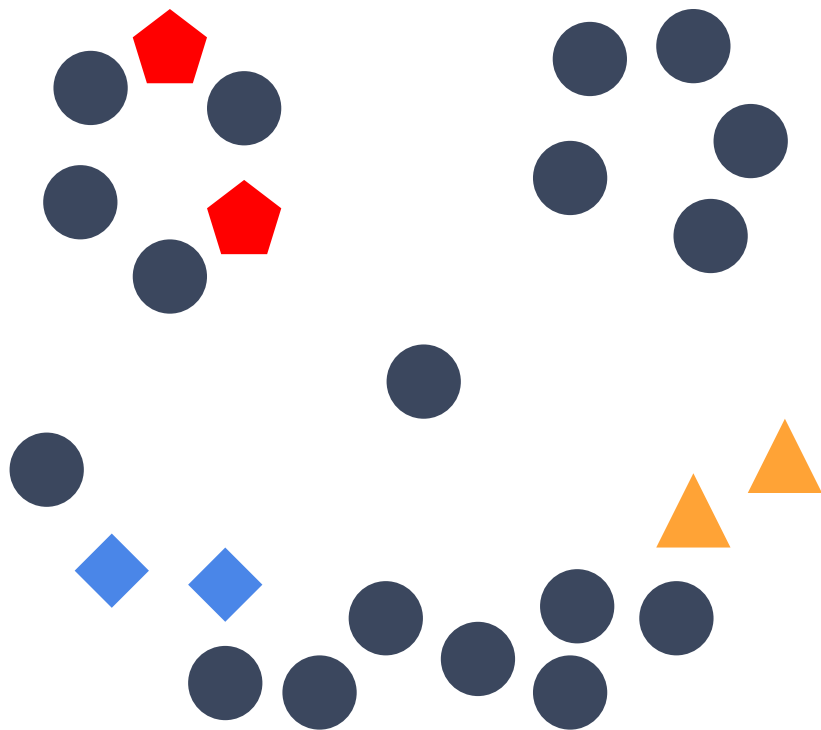
Case Study: Barnes-Hut Simulation

- ◆ Possibility 1: I did it wrong
- ◆ Possibility 2: Interpreter overhead is overwhelming the speedup
 - ◆ Top down traversal for each particle, without any sharing
- ◆ Maybe with more particles it is faster
 - ◆ My laptop is not strong enough to try
- ◆ The point: parent vectors don't help us with top-down traversal!

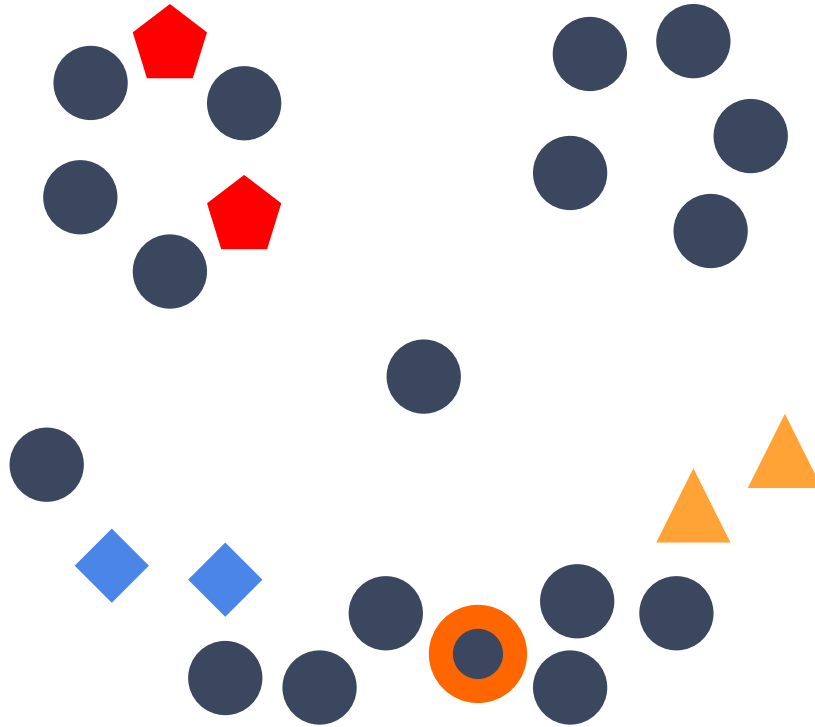
Case Study: Disjoint Set Data Structure



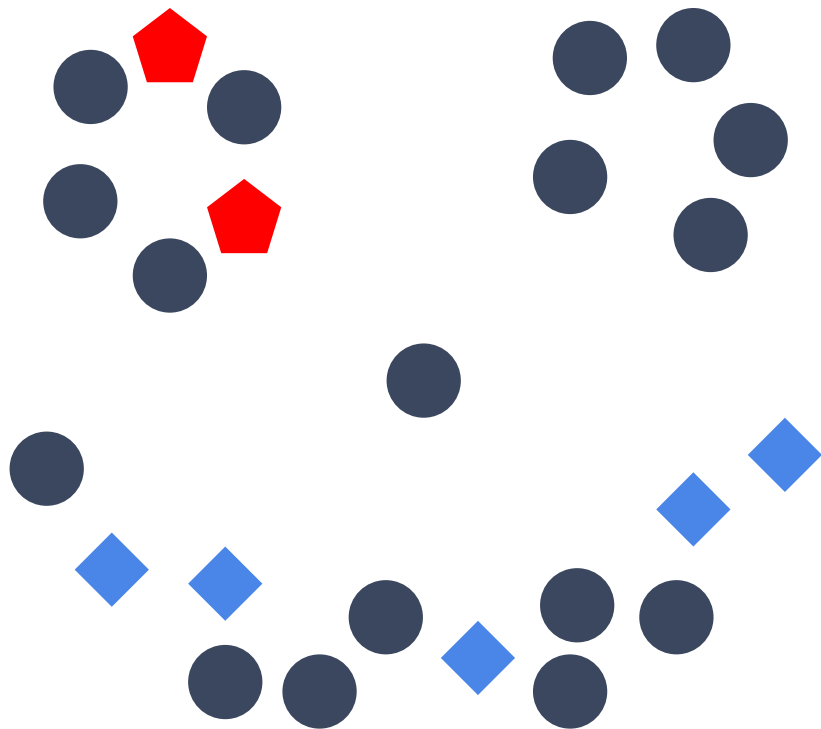
Case Study: Disjoint Set Data Structure



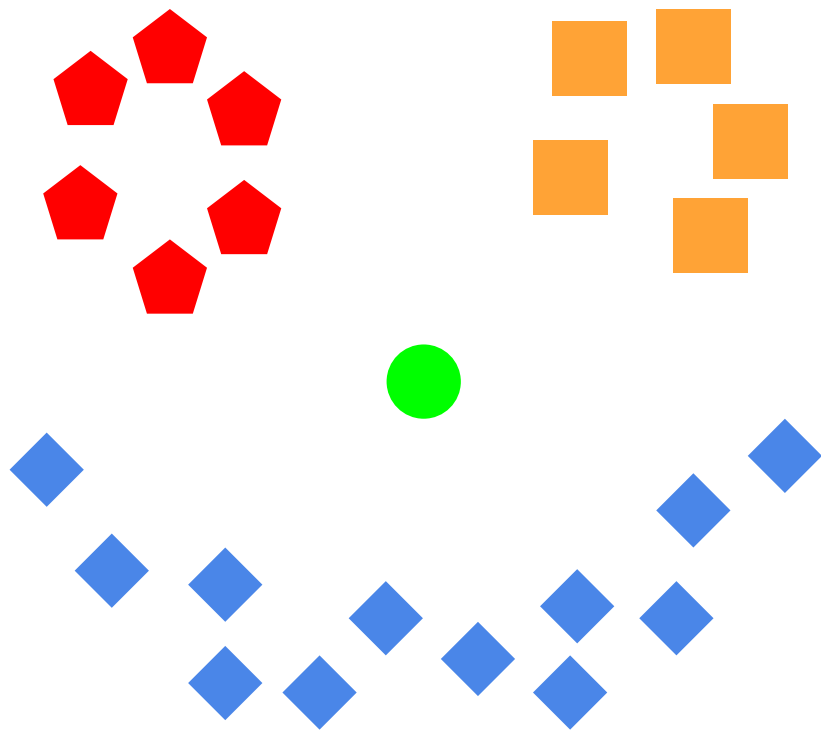
Case Study: Disjoint Set Data Structure



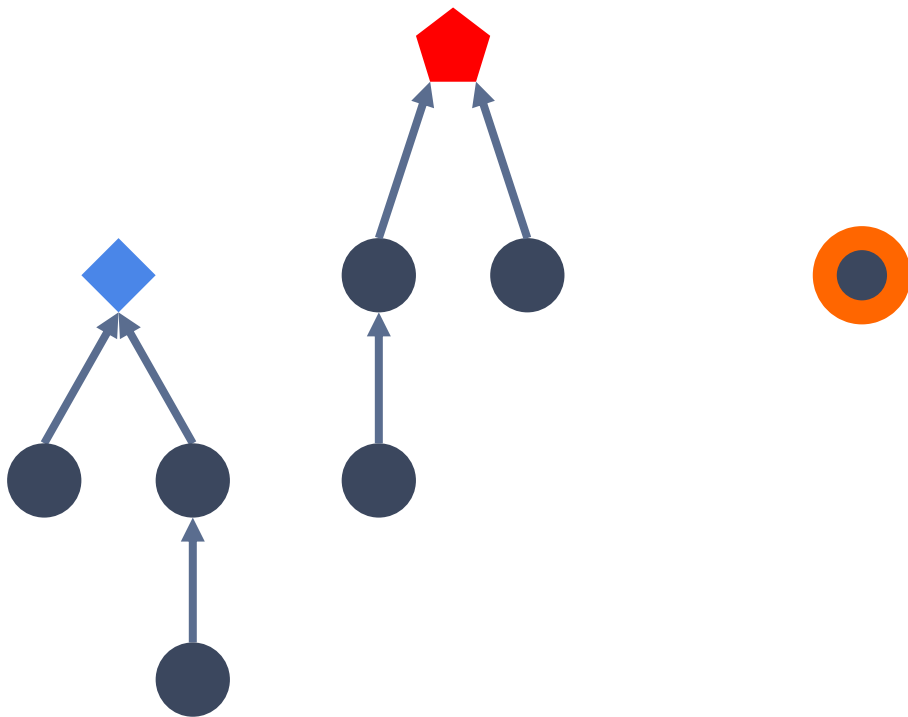
Case Study: Disjoint Set Data Structure



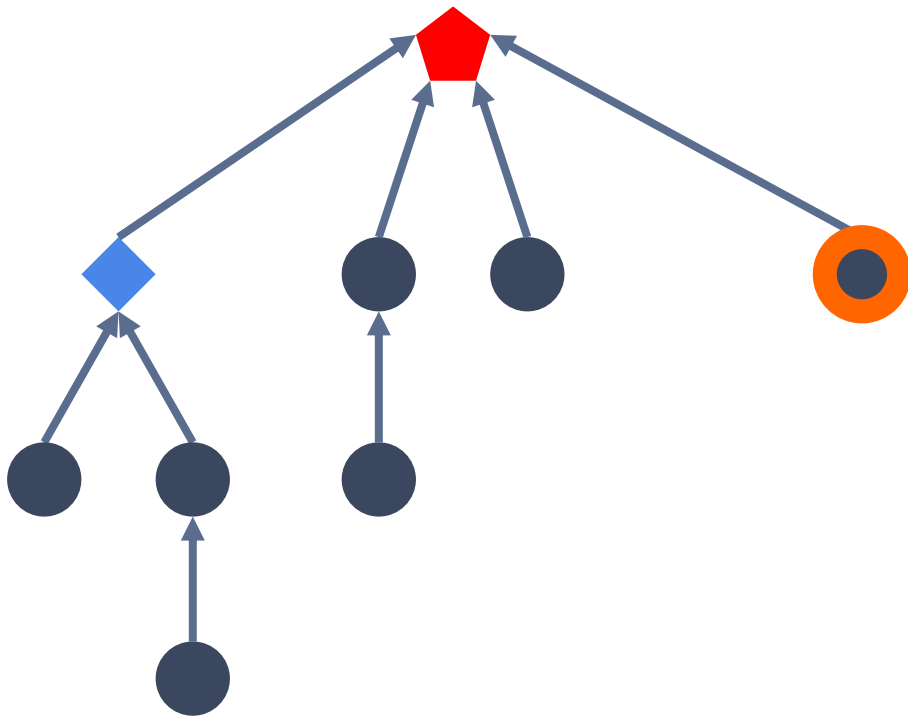
Case Study: Disjoint Set Data Structure



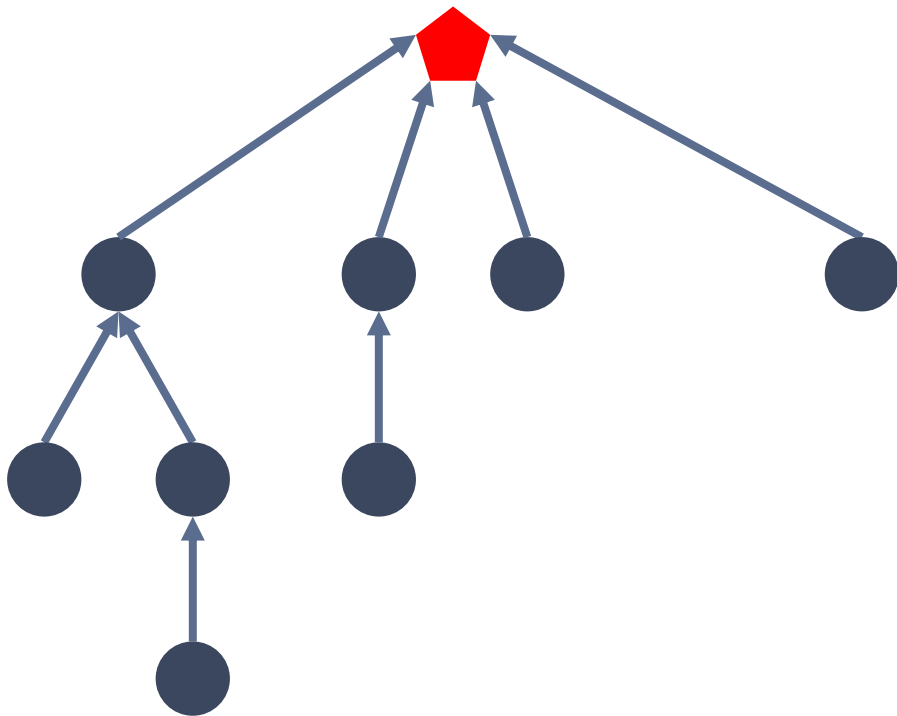
Case Study: Disjoint Set Data Structure



Case Study: Disjoint Set Data Structure



Case Study: Disjoint Set Data Structure



Case Study: Disjoint Set Data Structure

- ✧ Finding contiguous regions of an image
 - ✧ I used it for this
- ✧ Finding connected components
- ✧ Finding minimum spanning trees
- ✧ Solving symbolic expressions by unification
- ✧ Probably more!

The Root of It

❖ Data

- ❖ Syntax trees ✓

- ❖ File systems ✓

- ❖ ...

❖ Algorithms

- ❖ Barnes-Hut simulation ✗

- ❖ Disjoint sets ✓

- ❖ ...

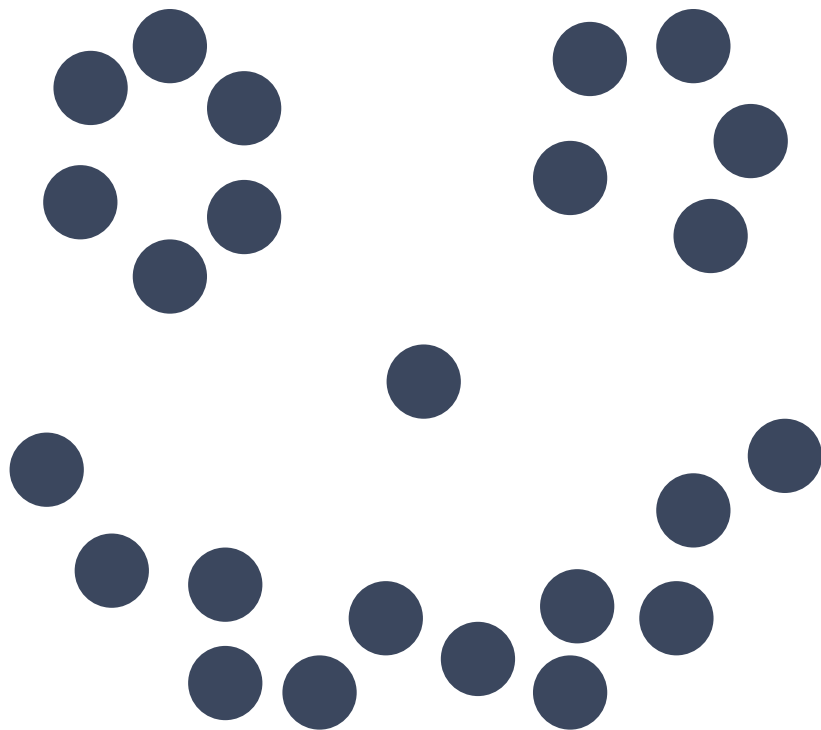
The Root of It

- ◆ You can do trees the APL way!
- ◆ Consider using parent vectors if you have trees which
 - ◆ Are large
 - ◆ Are irregular
 - ◆ Are often traversed bottom-up
- ◆ You could* speed up your program and/or make it easier to understand
 - ◆ *your mileage may vary, always profile

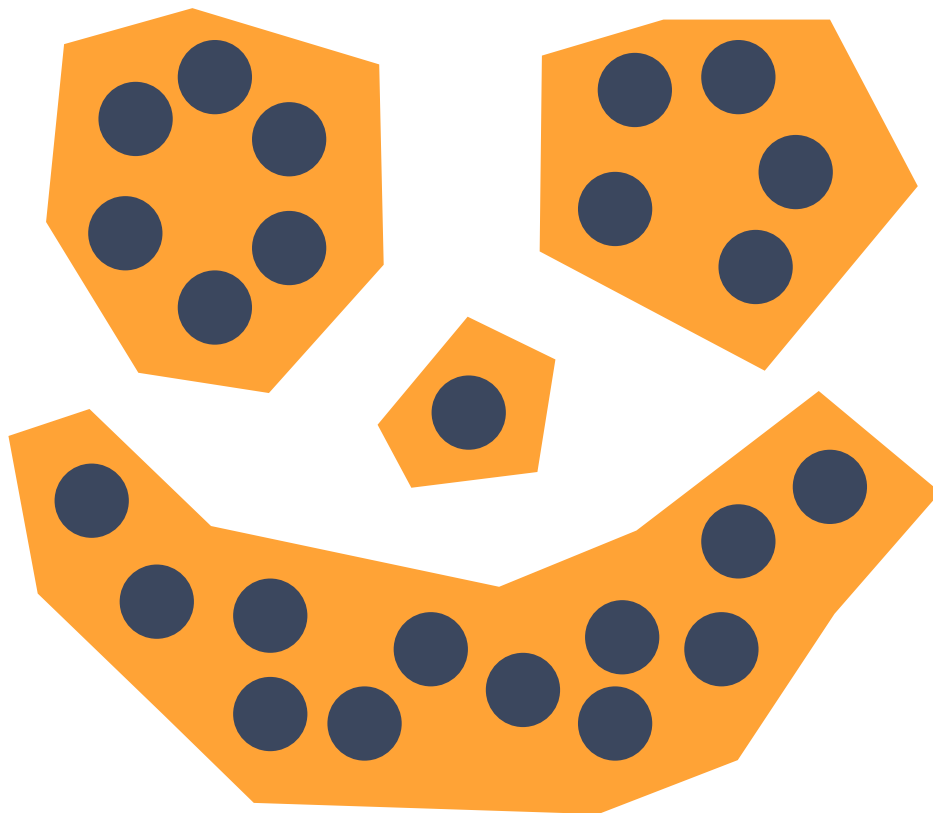
Read My Tutorial!

asherbhs.github.io/apl-site/trees/intro.html

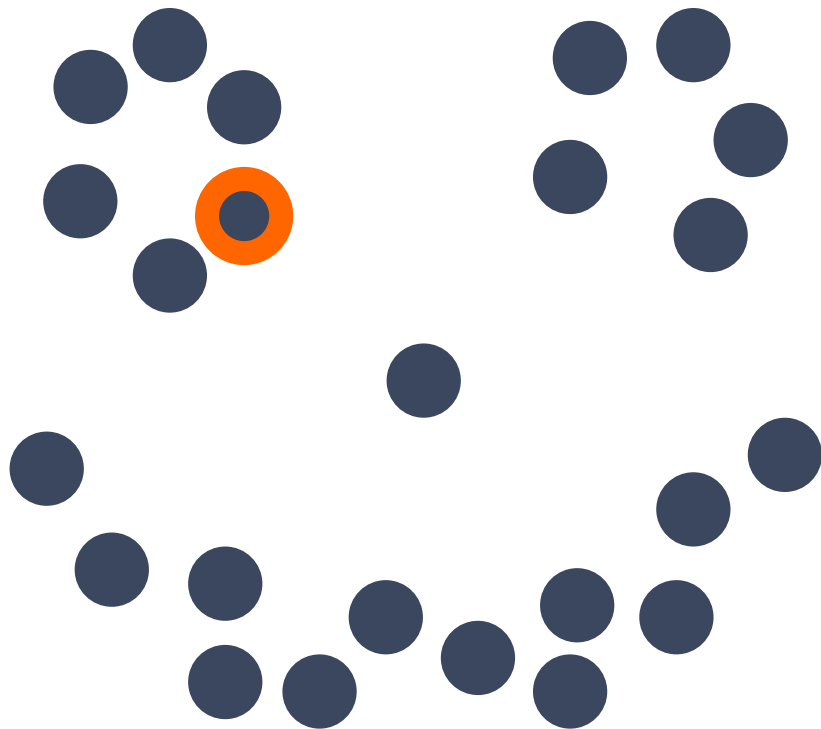
Case Study: Disjoint Set Data Structure



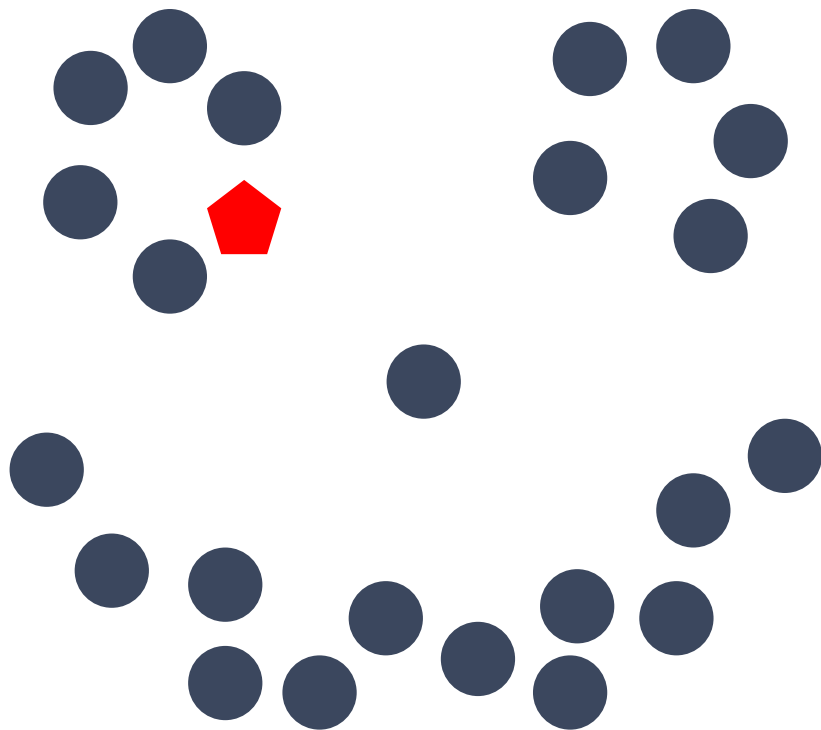
Case Study: Disjoint Set Data Structure



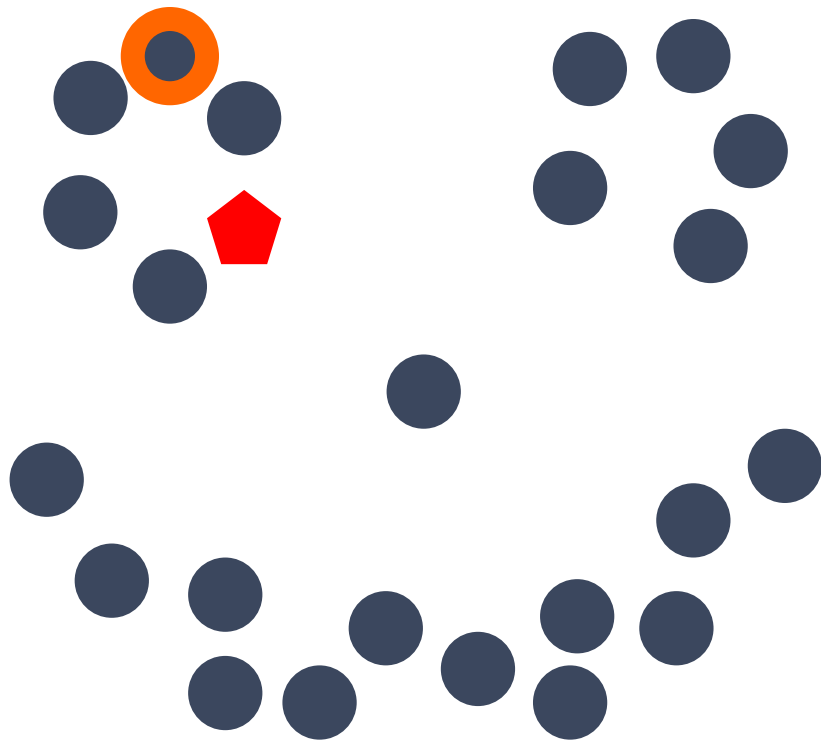
Case Study: Disjoint Set Data Structure



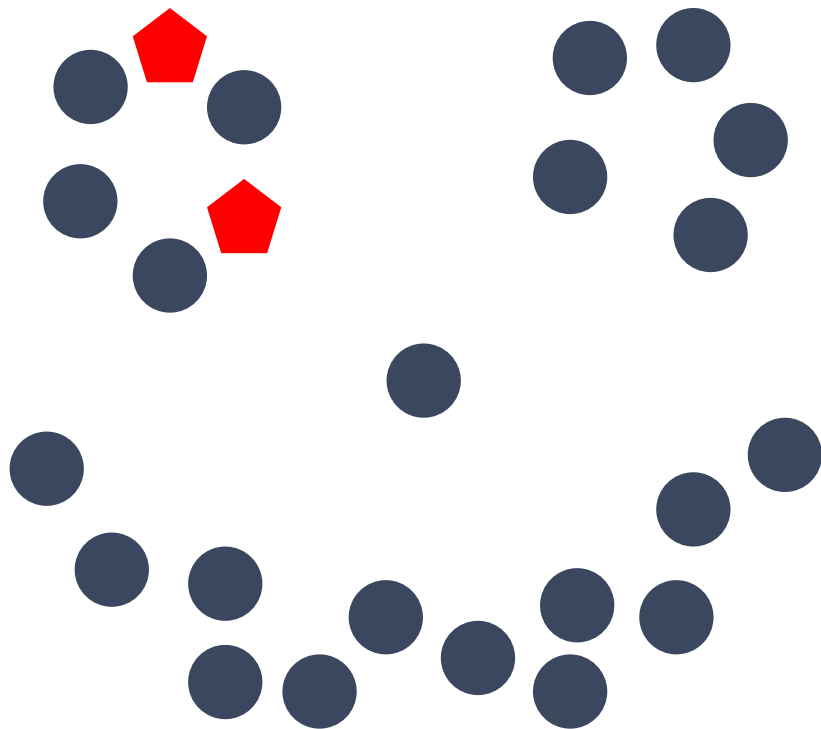
Case Study: Disjoint Set Data Structure



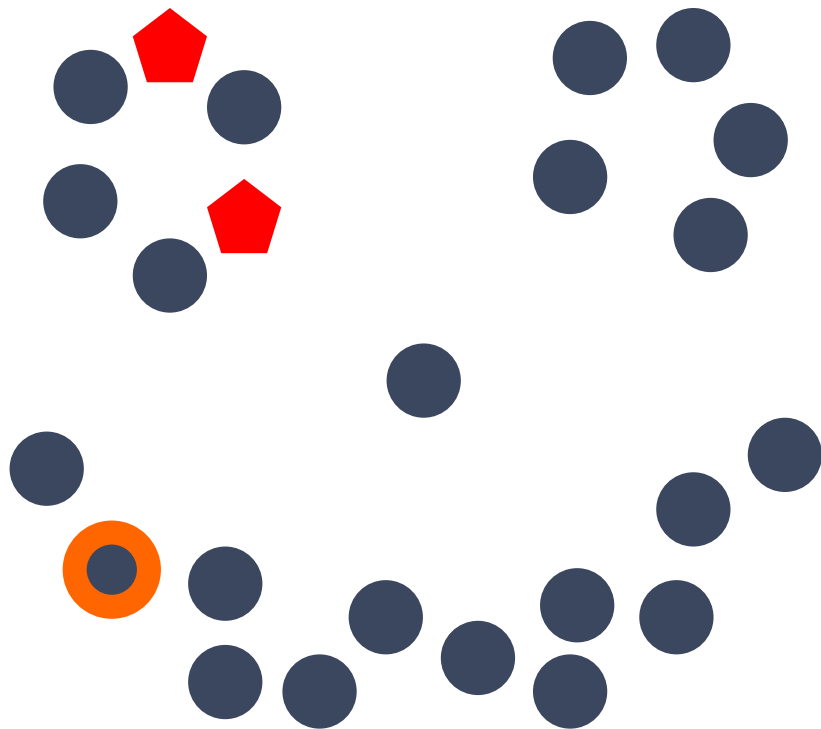
Case Study: Disjoint Set Data Structure



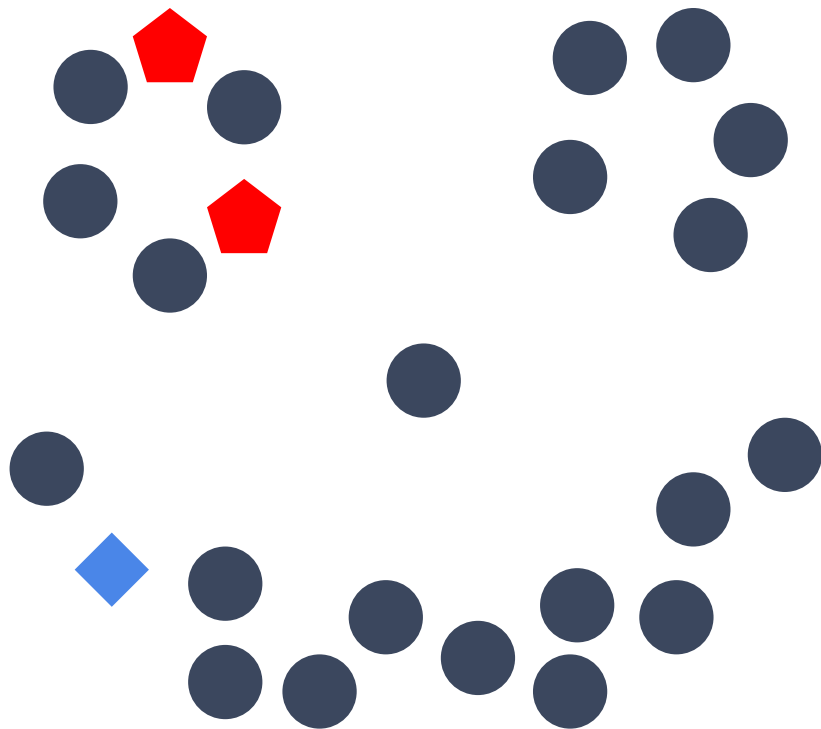
Case Study: Disjoint Set Data Structure



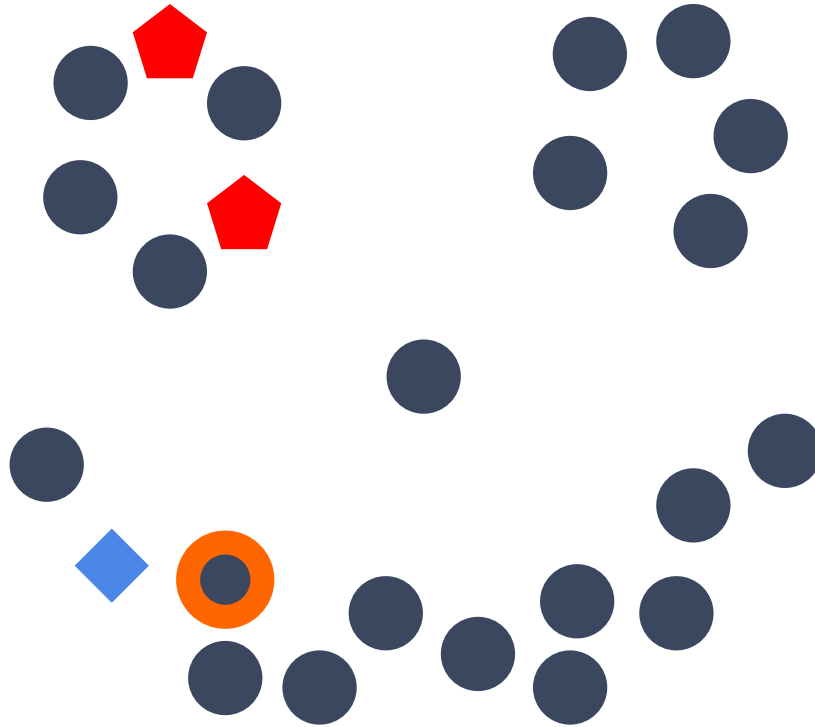
Case Study: Disjoint Set Data Structure



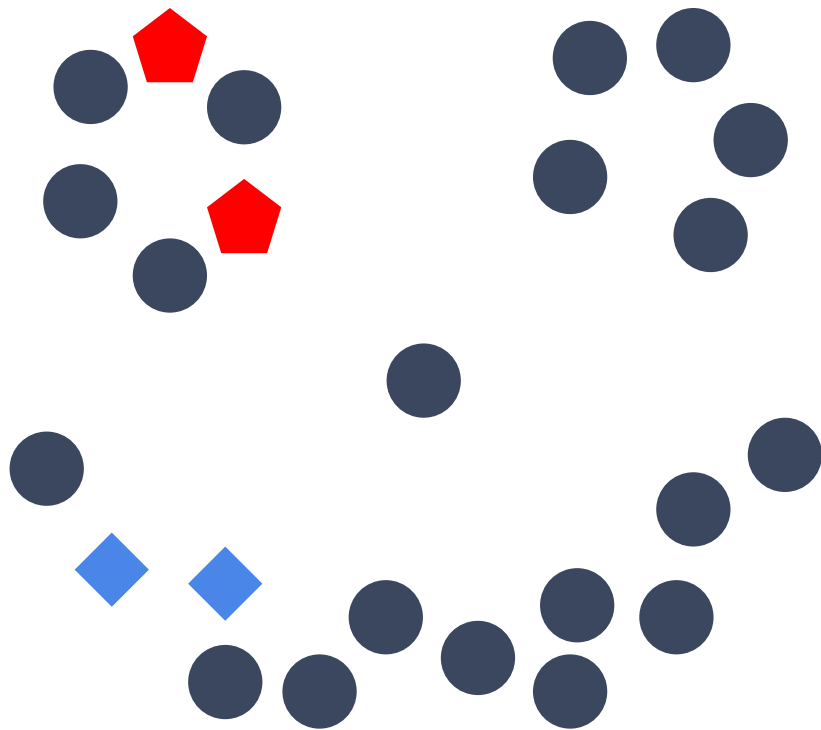
Case Study: Disjoint Set Data Structure



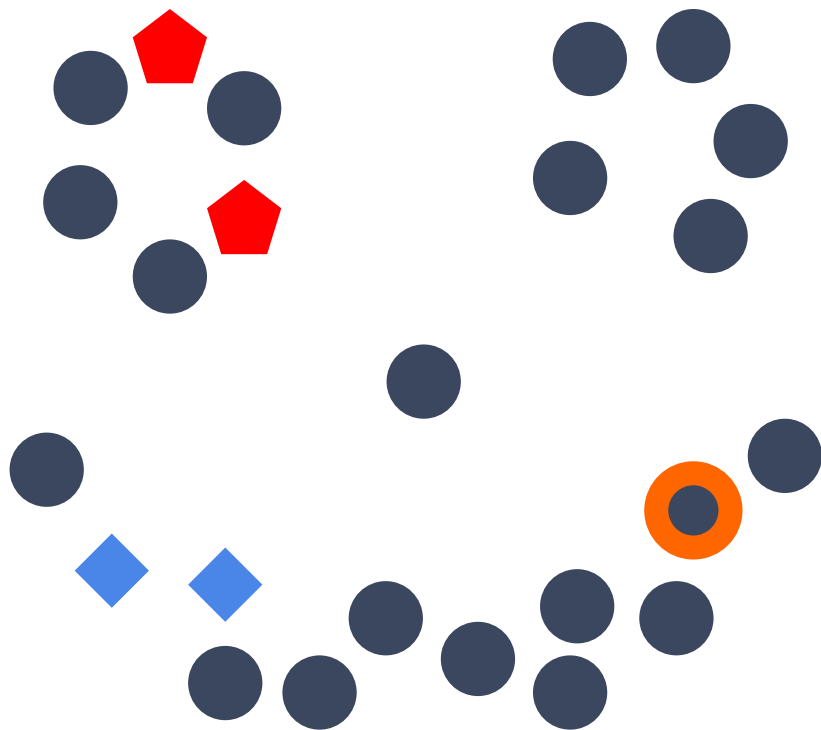
Case Study: Disjoint Set Data Structure



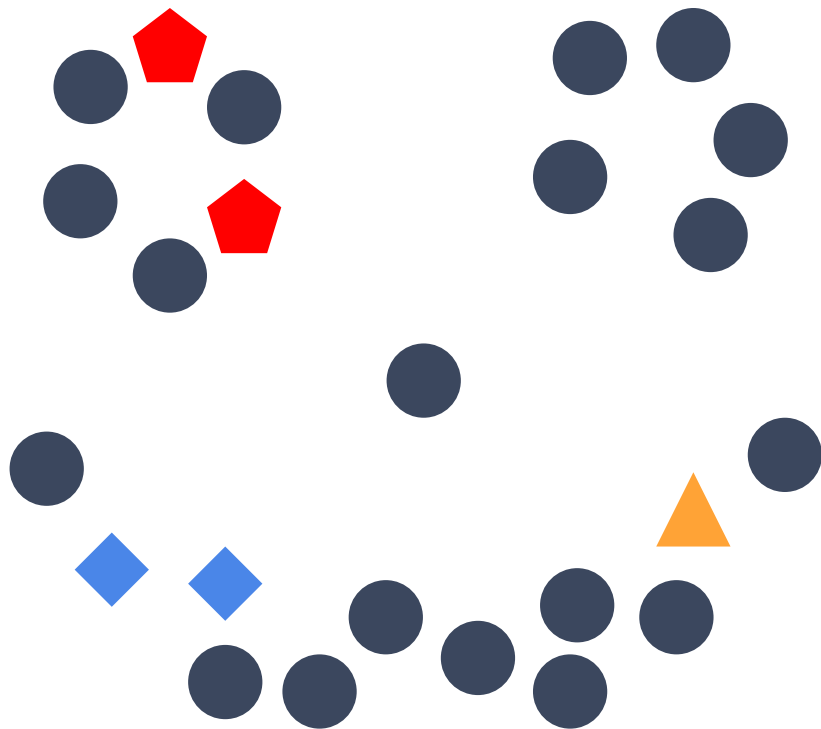
Case Study: Disjoint Set Data Structure



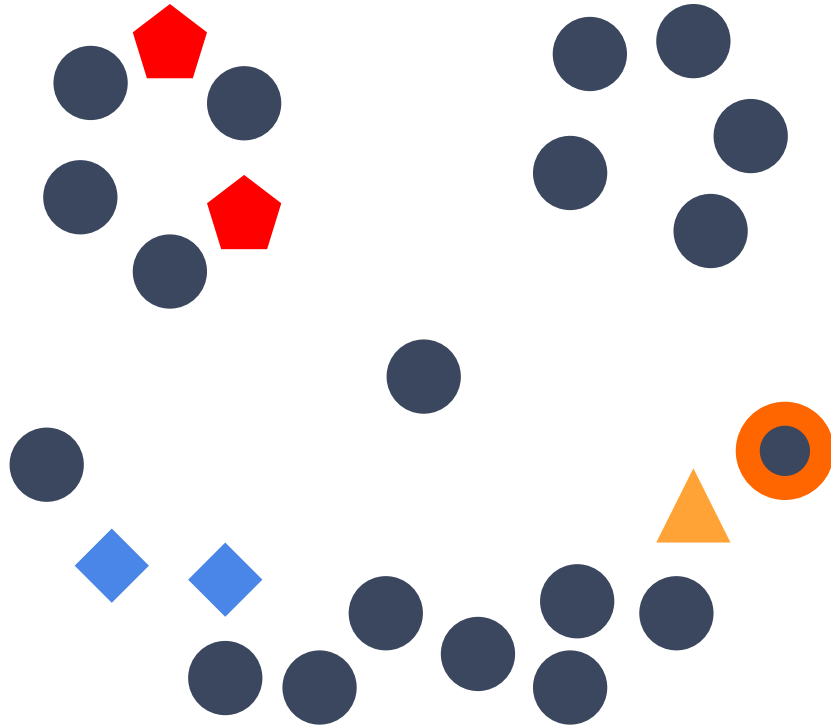
Case Study: Disjoint Set Data Structure



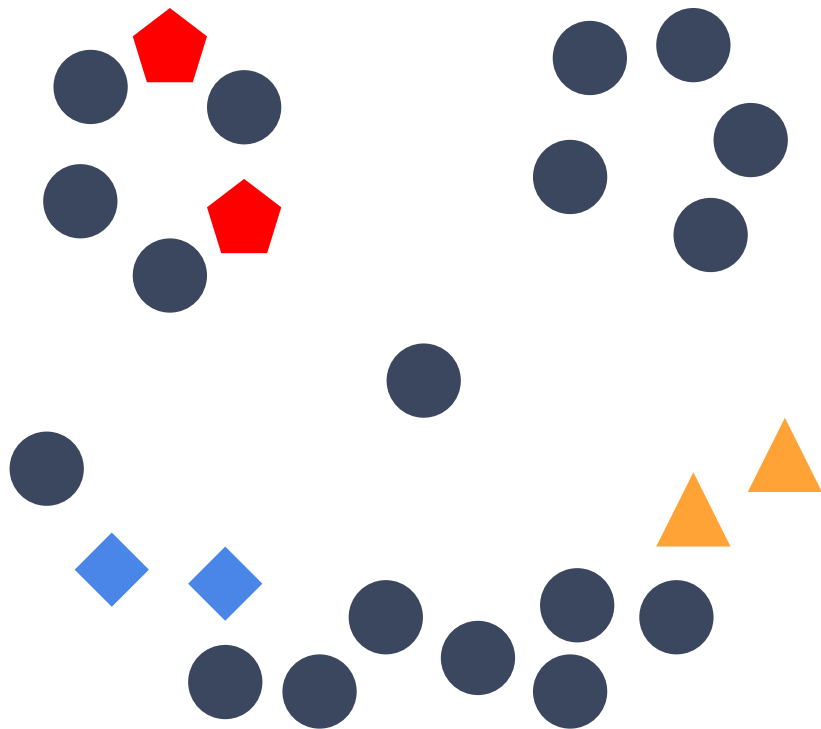
Case Study: Disjoint Set Data Structure



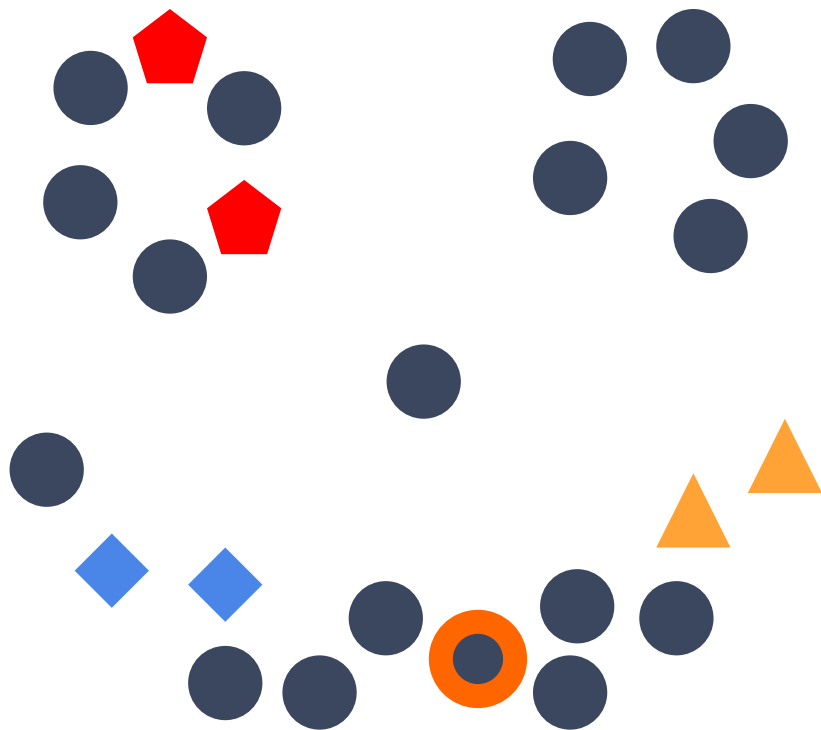
Case Study: Disjoint Set Data Structure



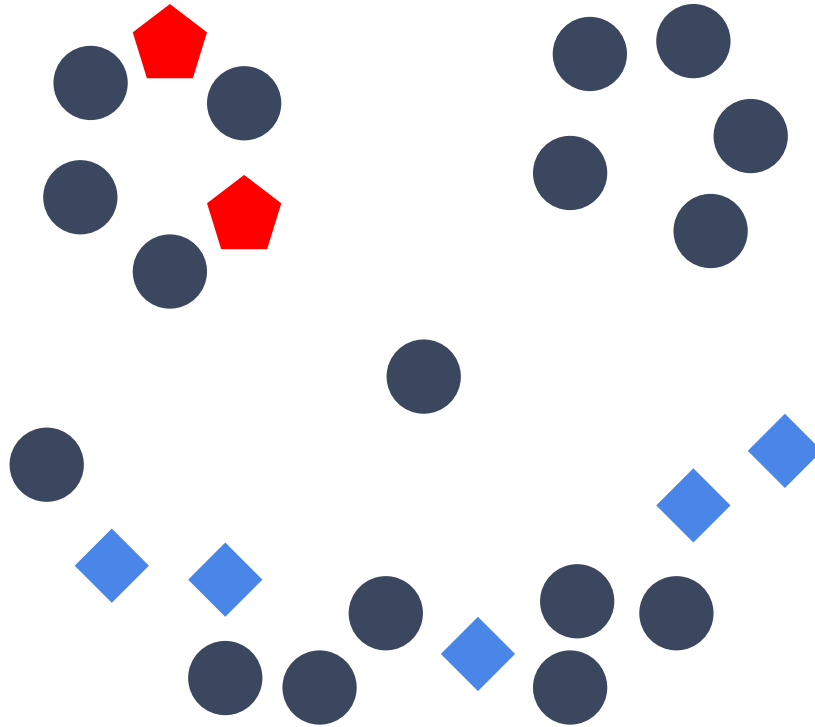
Case Study: Disjoint Set Data Structure



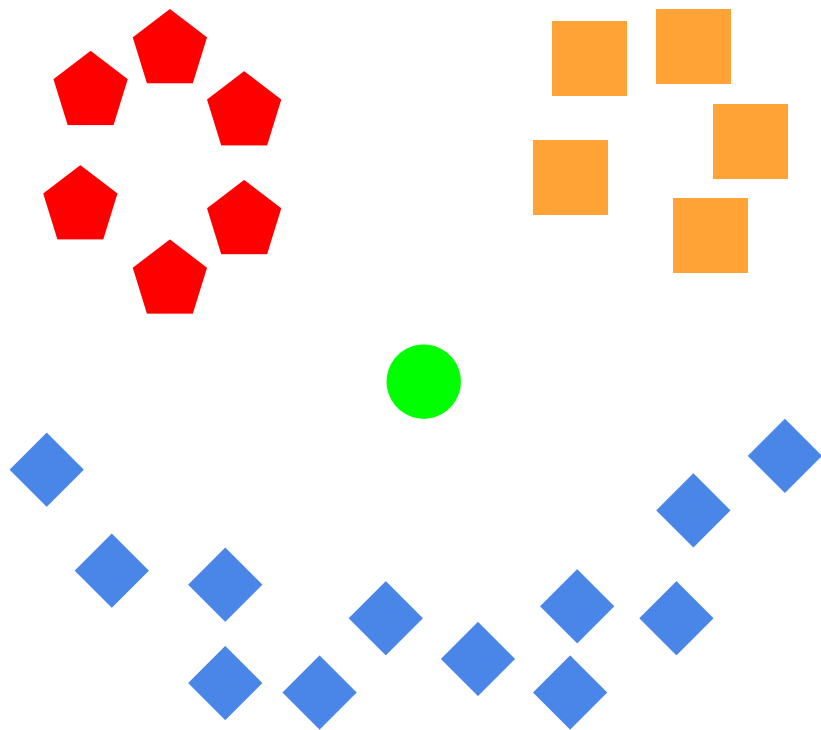
Case Study: Disjoint Set Data Structure



Case Study: Disjoint Set Data Structure



Case Study: Disjoint Set Data Structure



Case Study: Disjoint Set Data Structure



Case Study: Disjoint Set Data Structure



Case Study: Disjoint Set Data Structure



Case Study: Disjoint Set Data Structure



Case Study: Disjoint Set Data Structure



Case Study: Disjoint Set Data Structure



Case Study: Disjoint Set Data Structure



Case Study: Disjoint Set Data Structure



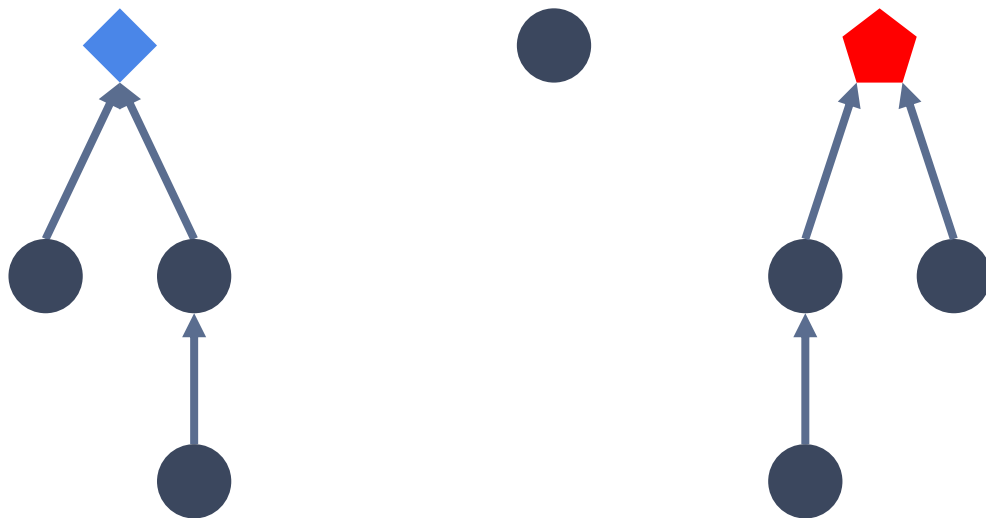
Case Study: Disjoint Set Data Structure



Case Study: Disjoint Set Data Structure



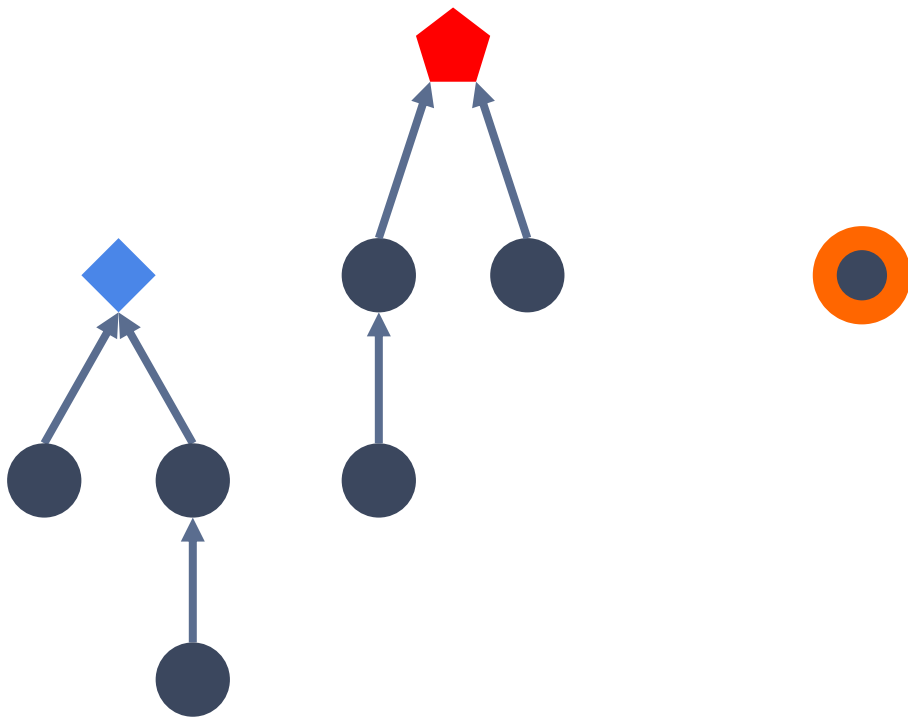
Case Study: Disjoint Set Data Structure



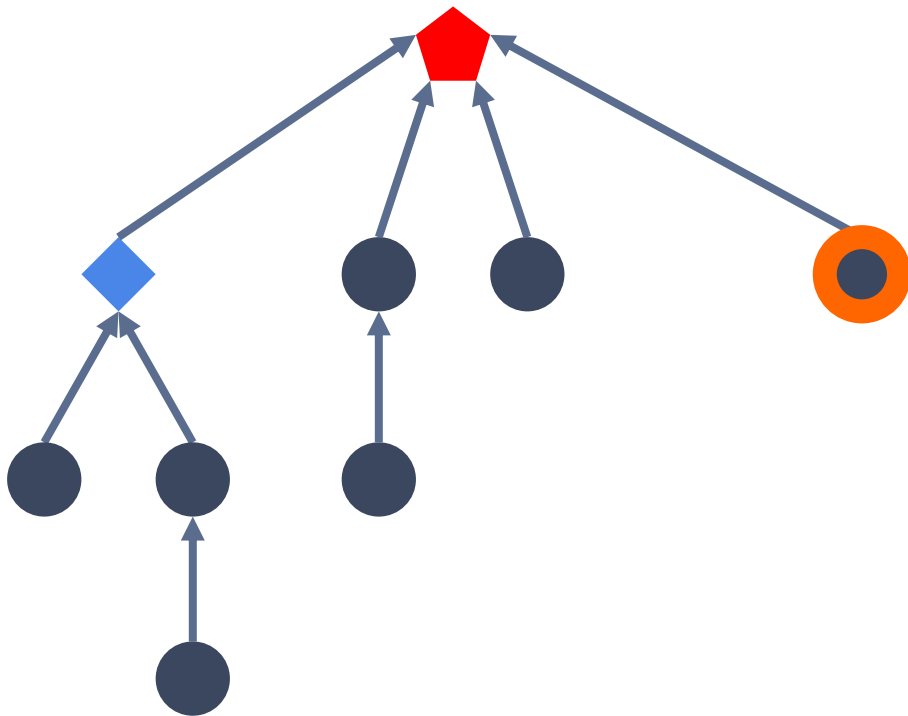
Case Study: Disjoint Set Data Structure



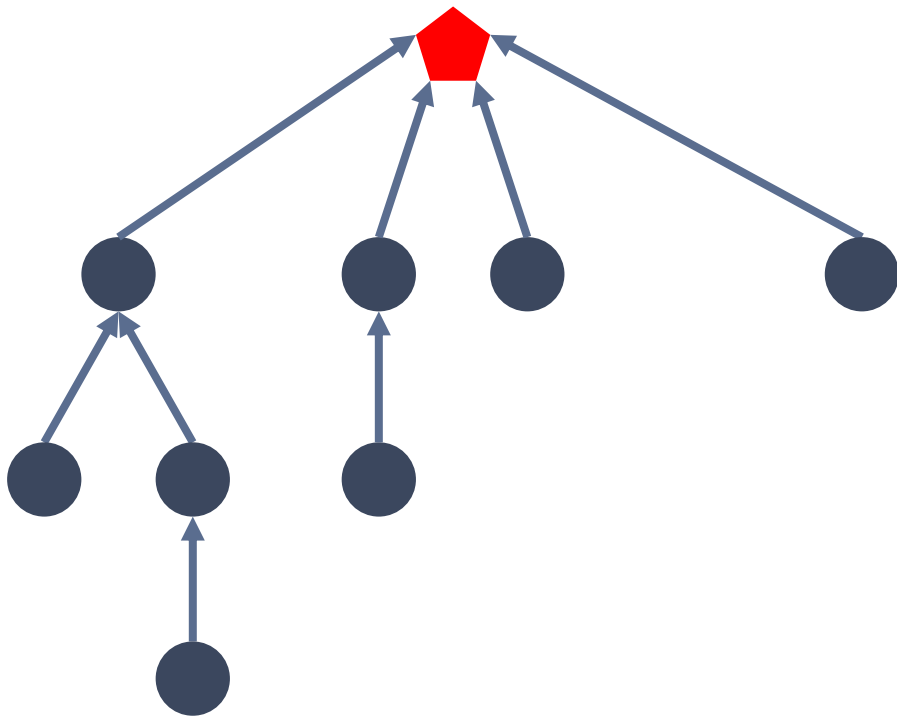
Case Study: Disjoint Set Data Structure

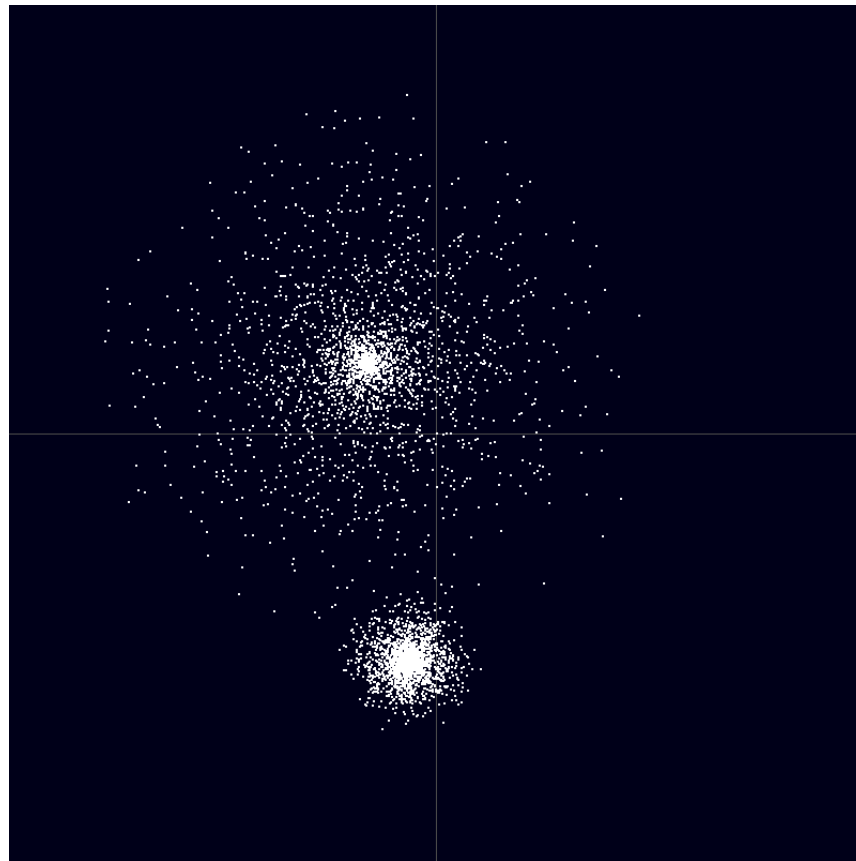


Case Study: Disjoint Set Data Structure

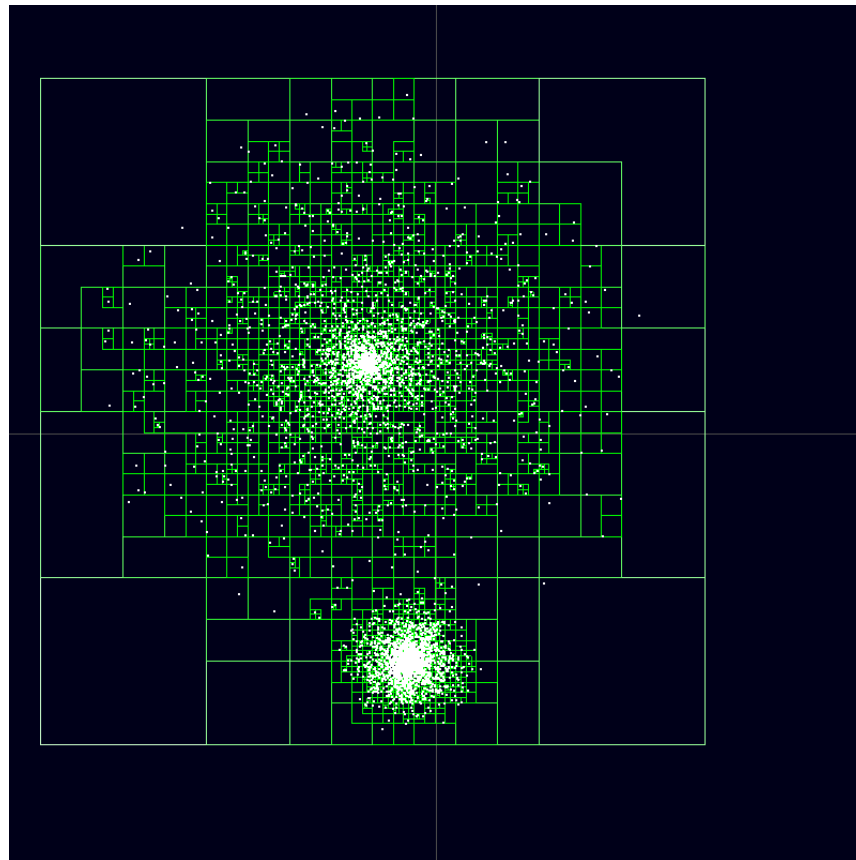


Case Study: Disjoint Set Data Structure

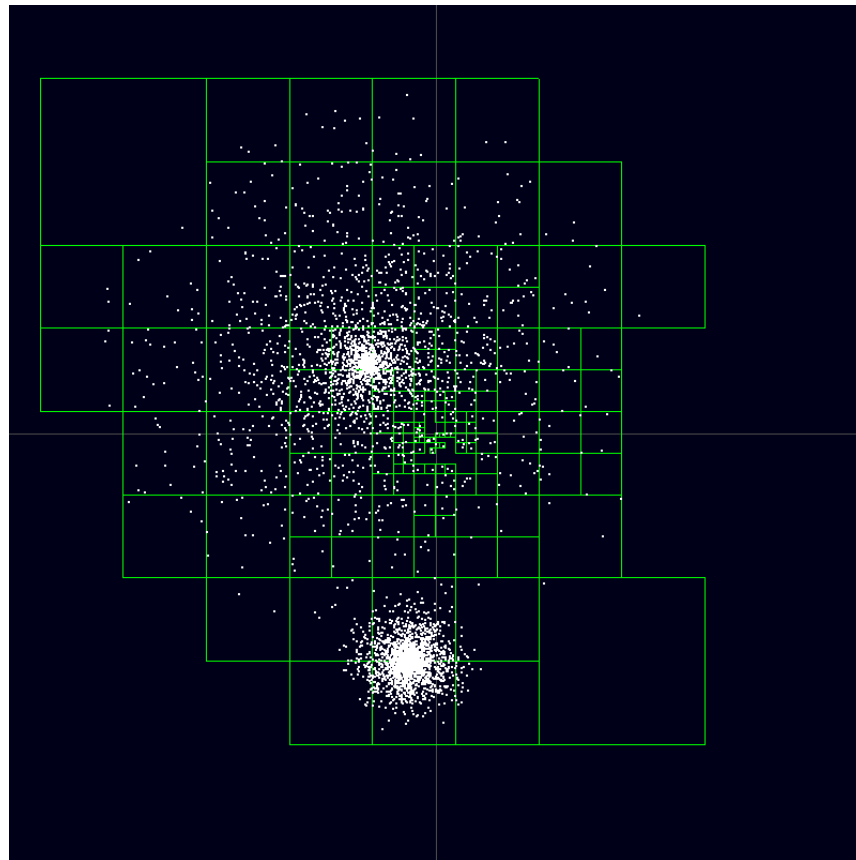




By Eclipse.sx - Own work, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=7388664>



By Eclipse.sx - Own work by uploader; created with: http://www.beltoforion.de/barnes_hut/barnes_hut_de.html, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=7388668>



By Eclipse.sx - Own work by uploader; created with: http://www.beltoforion.de/barnes_hut/barnes_hut_de.html, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=7388671>

