Geometric Interpretation of BSTs

Matt Asnes and Harrison Kaiser

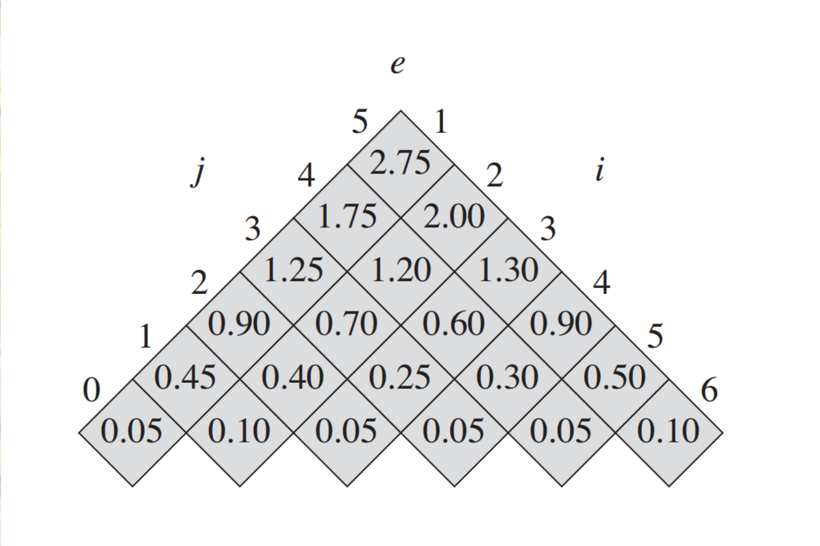
Part I: Some Review

The Model

- BSTs are data structures reacting to a series of *operations*
  - Insert, search, delete, merge, etc.
  - Each operation has an *argument*
  - e.g. insert 10, delete 5, search 10, …
- Each operation starts at the root
- At each step:
  - Move between adjacent vertices
  - Perform a rotation
- Eventually we *access* to argument of the operation
- Access: find & report back, insert, delete

BST Optimality

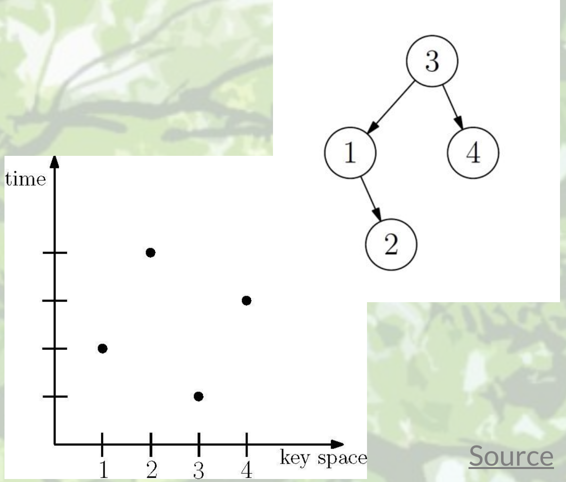
- Given a known set of operations, we can calculate
- But for an unknown set of operations, what tree reacts the best?
- Can be expressed mathematically…



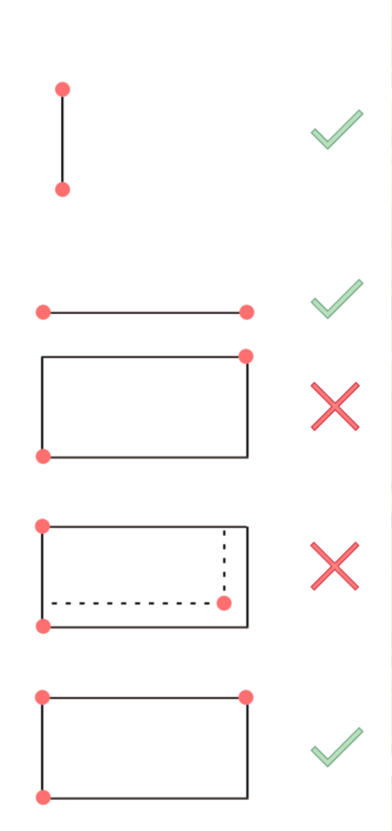
Source: CLRS, ch. 15.6

Arborally Satisfied Point Sets

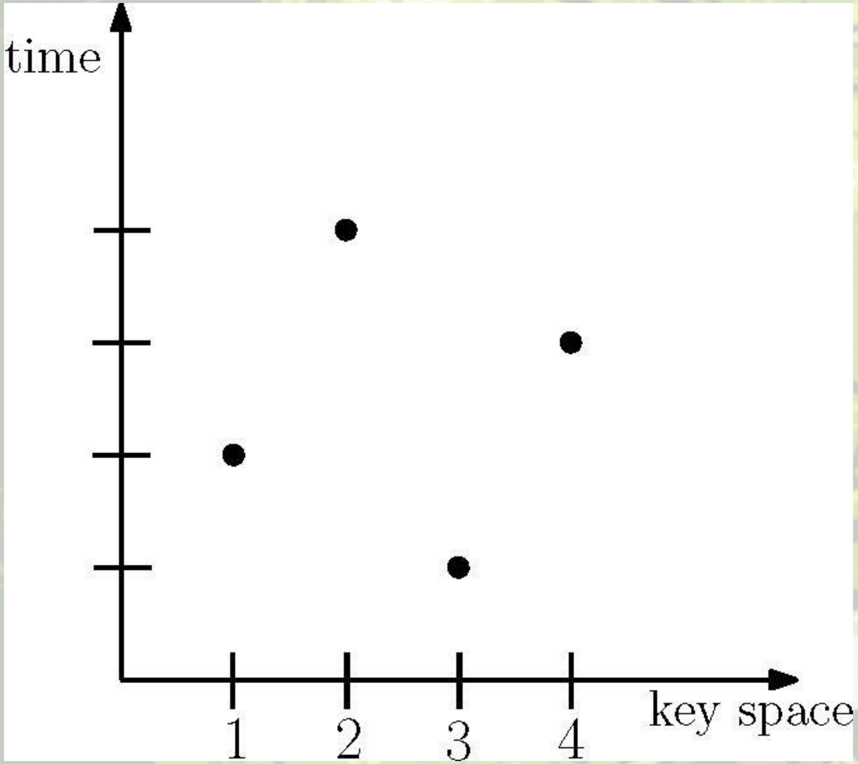
- For any pair of points:
  - If the two points form a rectangle:
    - There exists a third point inside or on the boundary of that rectangle
- Translation to BSTs:
  - In 2D
  - x-axis: values being accessed
  - y-axis: operation # of access

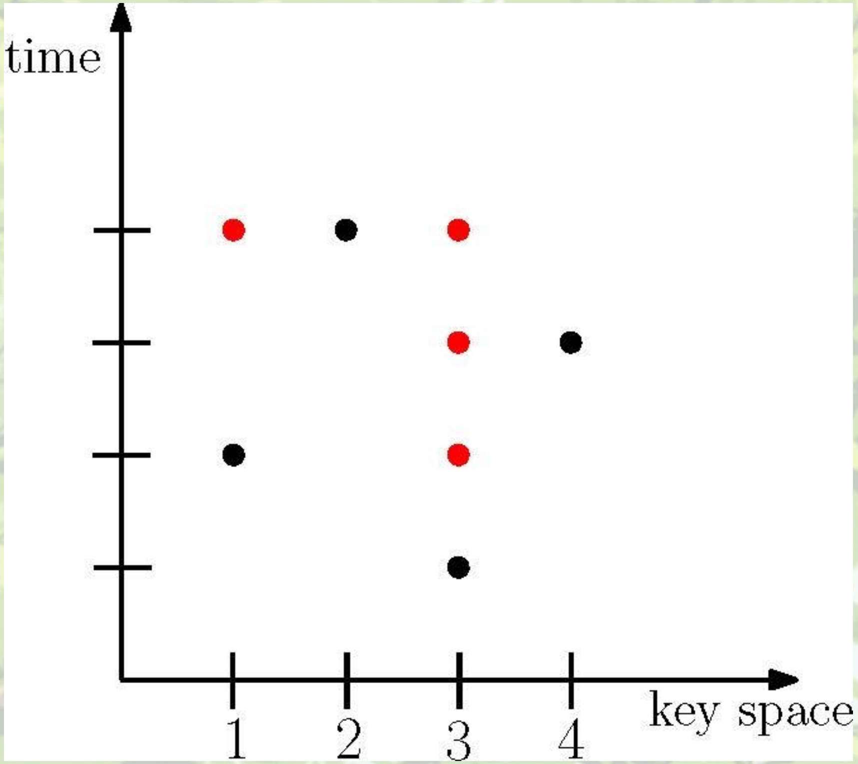


Source

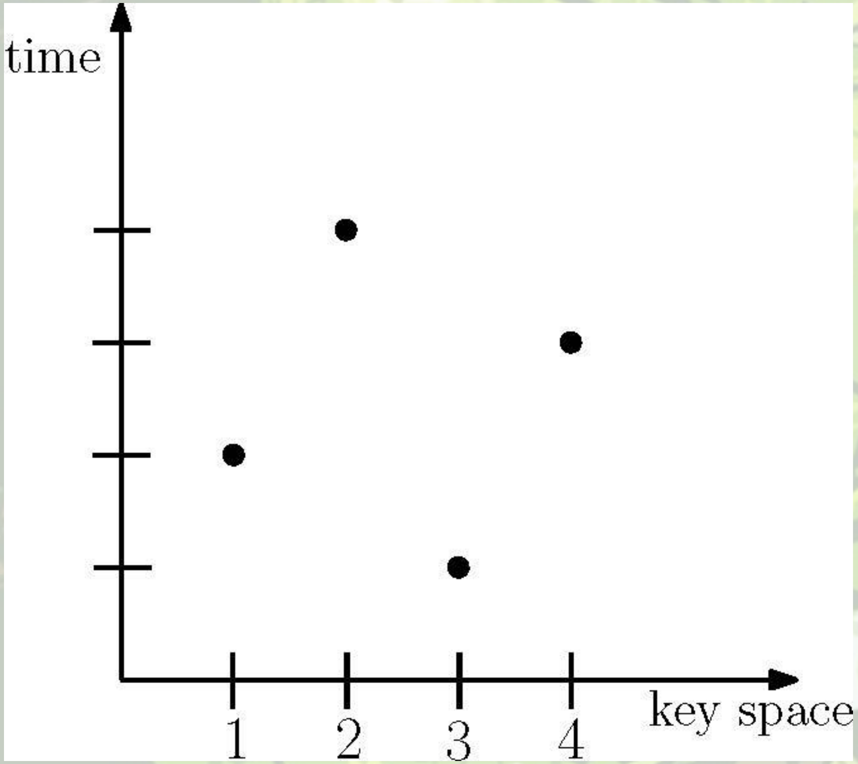


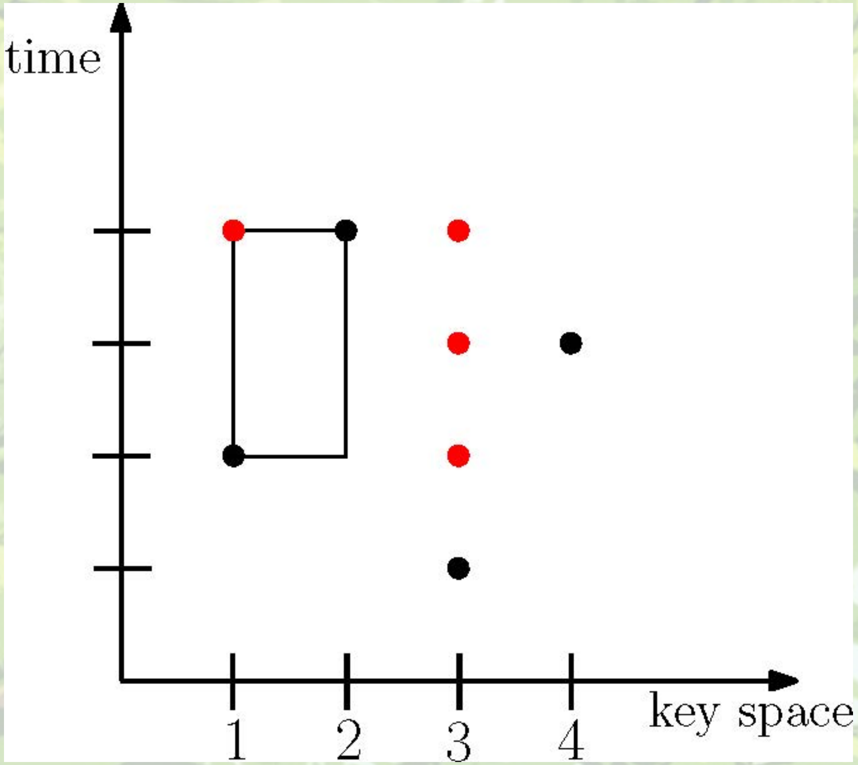
Arborally Satisfied Point Sets



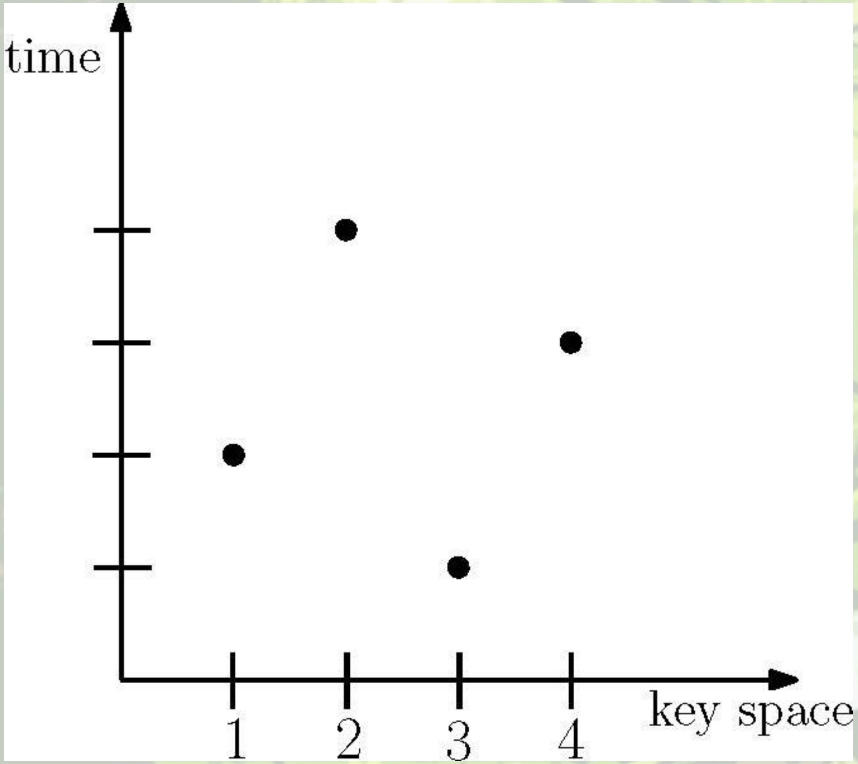


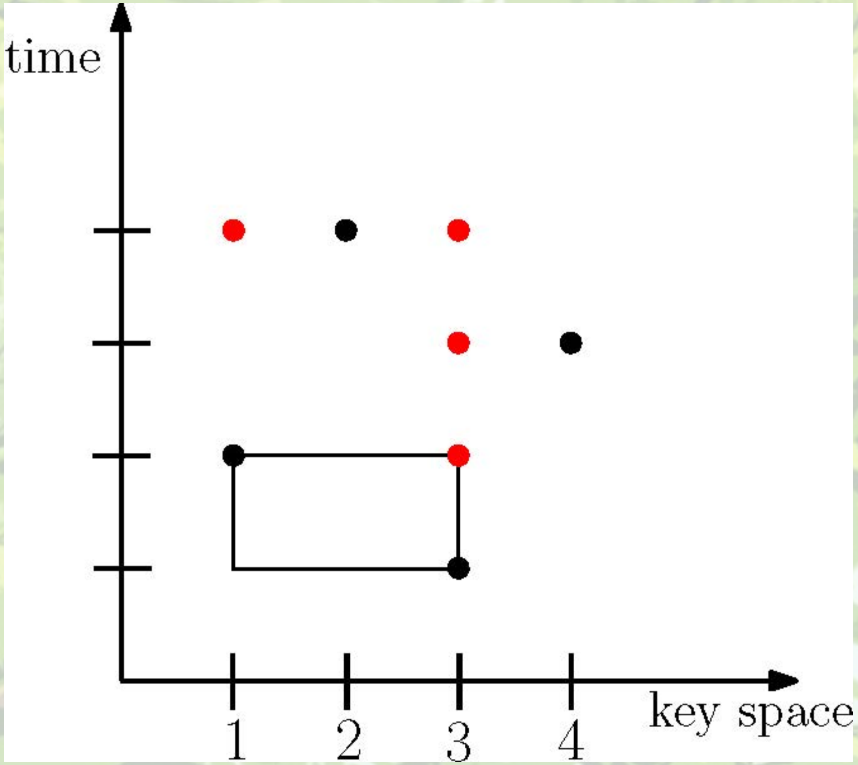
Arborally Satisfied Point Sets



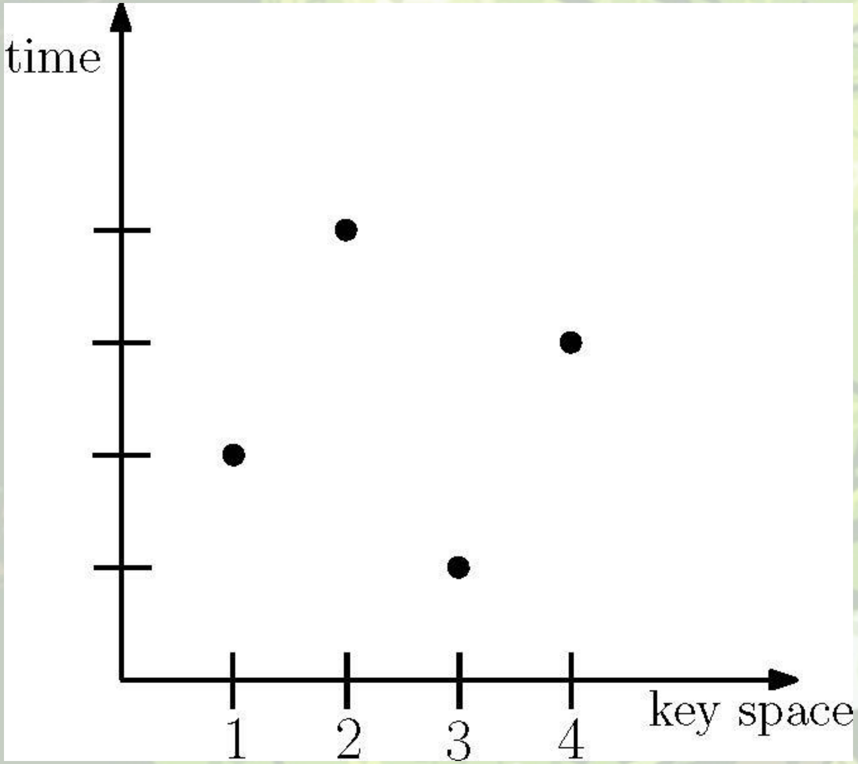


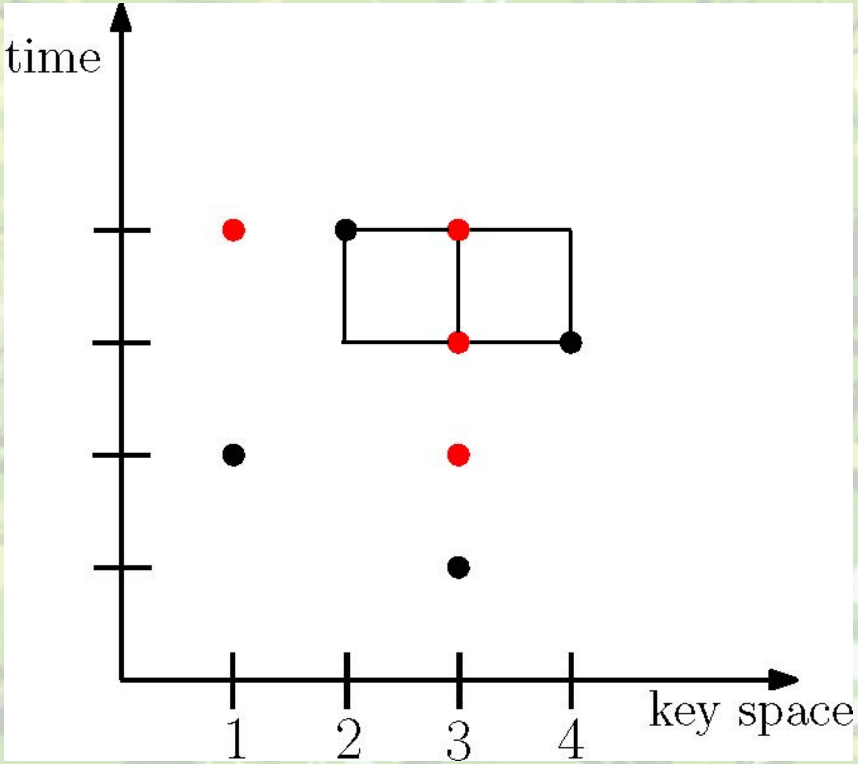
Arborally Satisfied Point Sets





Arborally Satisfied Point Sets





Dynamic Optimality

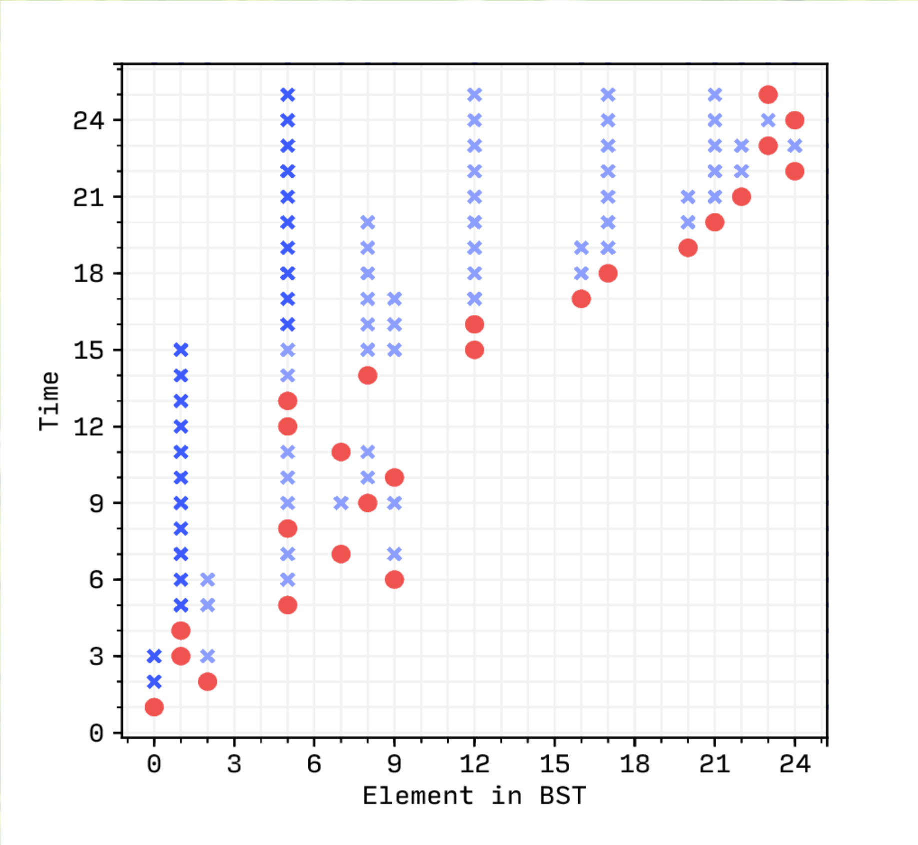
- A BST algorithm behaves deterministically on a sequence of operations
- A sequence of operations defines a set of points in a 2D space
- Plotting the accesses a BST makes along with the operations creates an arborally satisfied point set
- Then optimality is equivalent to a minimization problem

> “ Finding the best BST execution for a sequence of operations is equivalent to finding the minimum cardinality set of points that is arborally satisfied ”

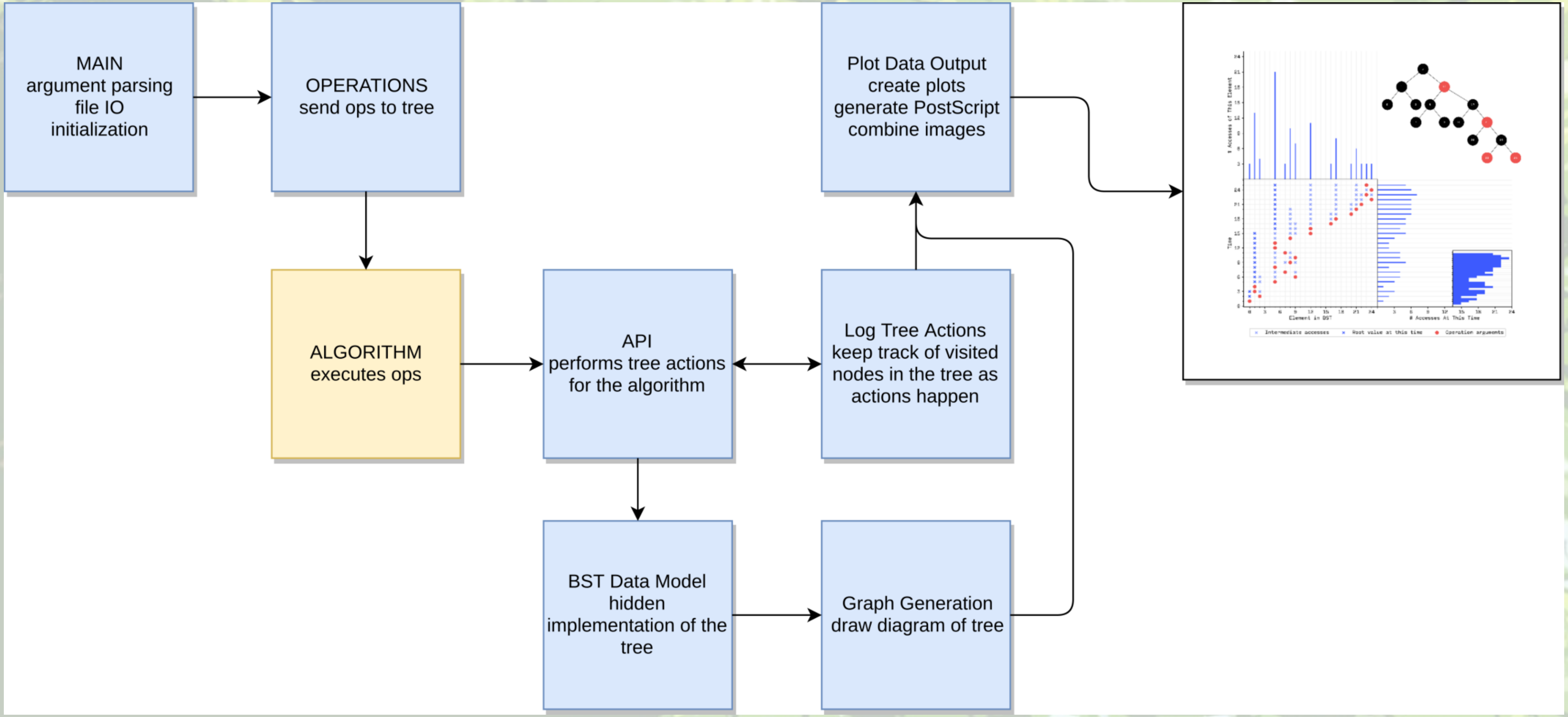
Part II: Our Project

Generation of BST Plots

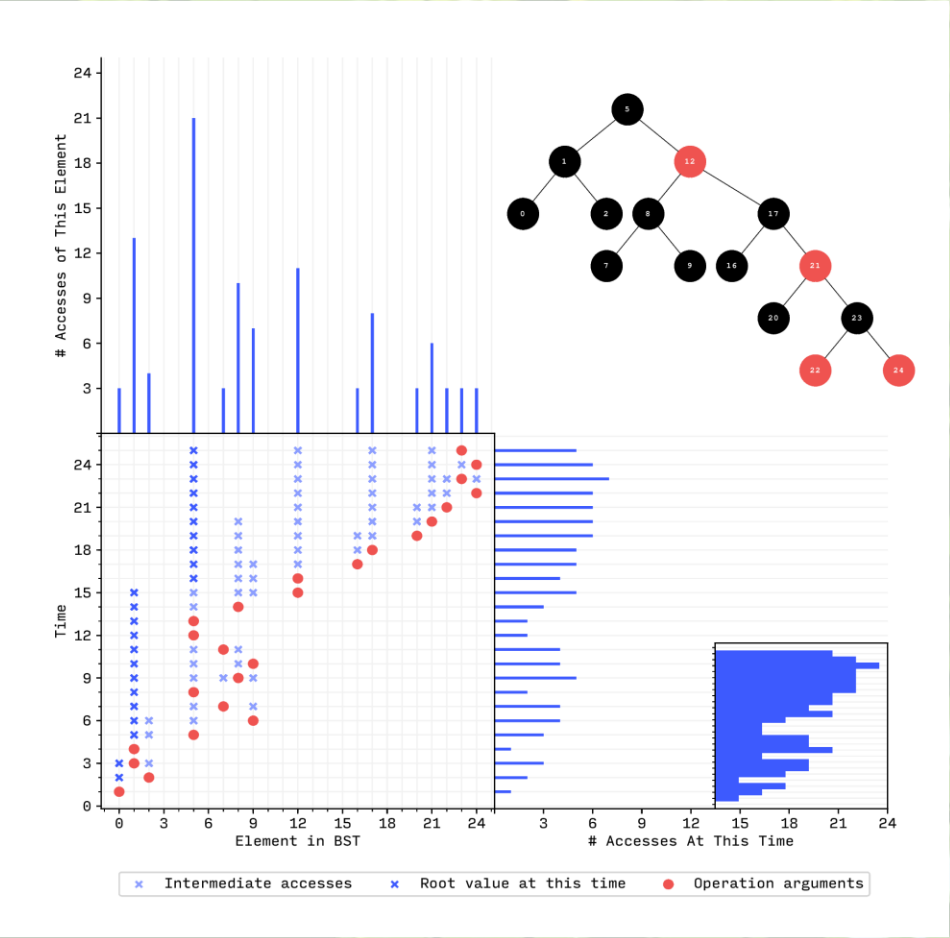
- Program to generate point set plots of BST operations & accesses
- Input: operations and their arguments
- Input: a BST algorithm
- Output: plots & analysis
- Simple, RB, Splay, AVL, WAVL, OPT, Greedy



Architecture

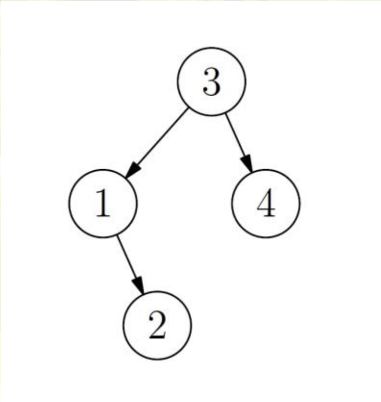


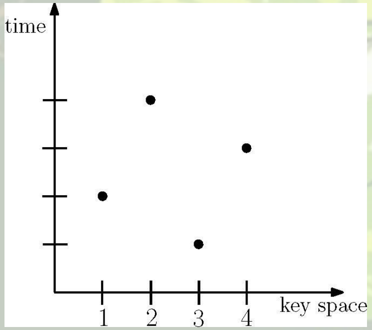
Output Format

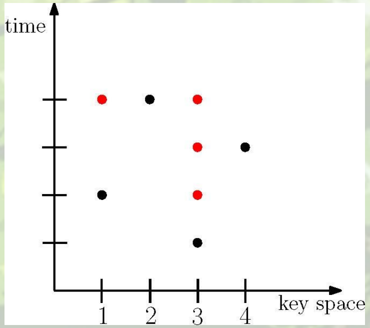


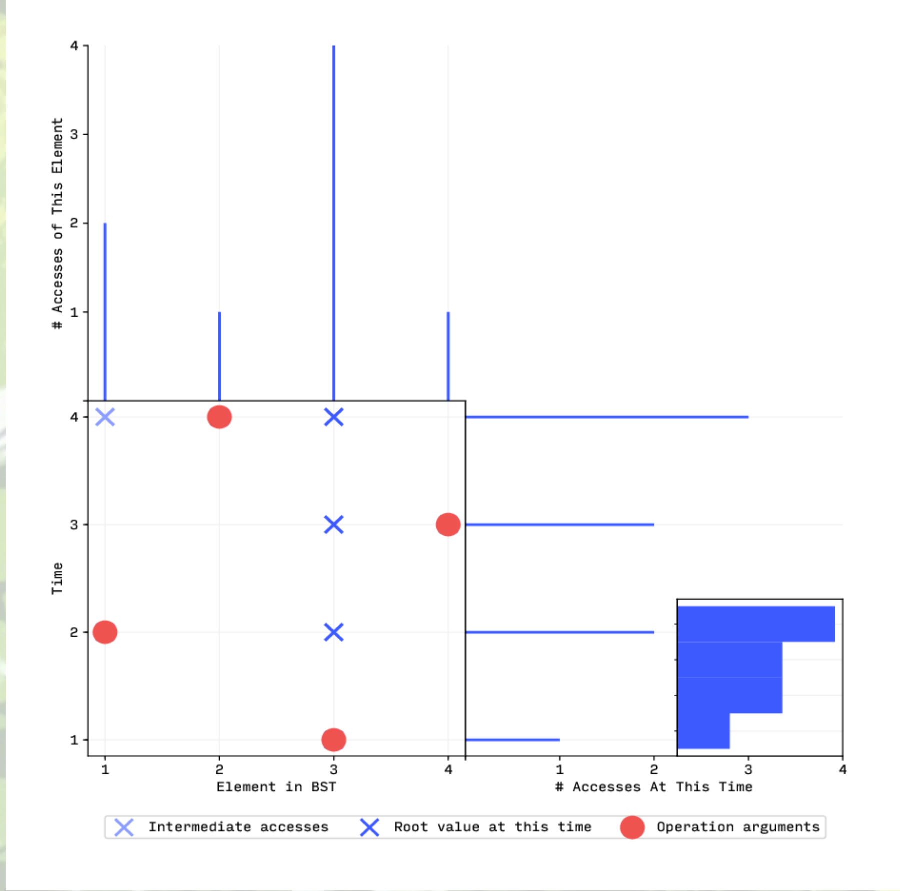
Arboral Satisfaction

```
# input
ins 3
ins 4
ins 1
ins 2
```

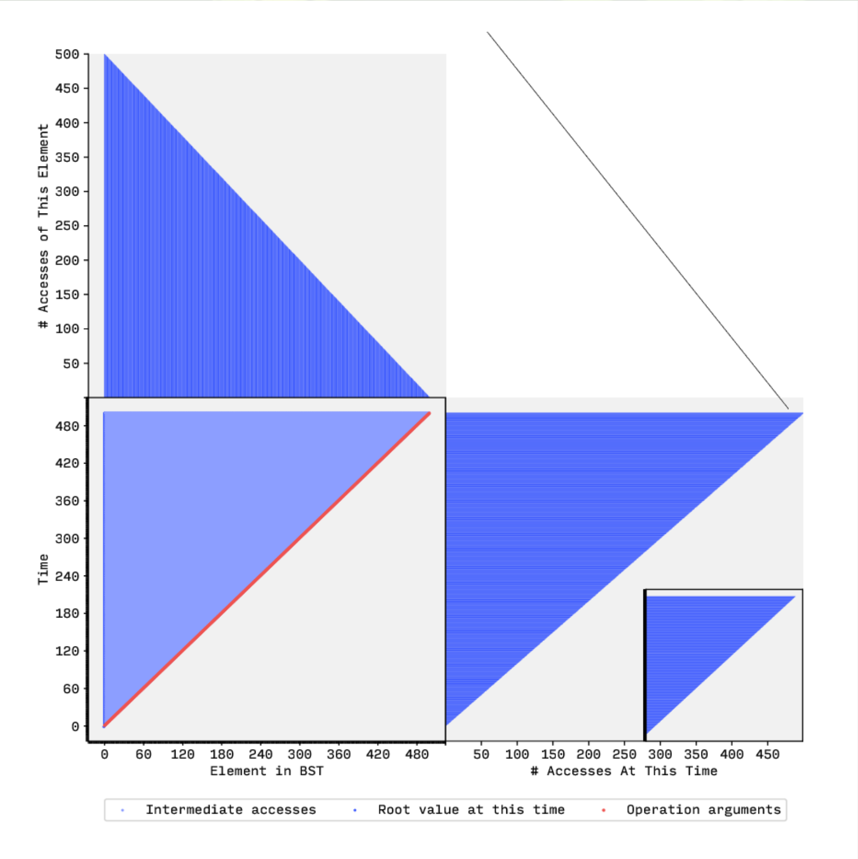


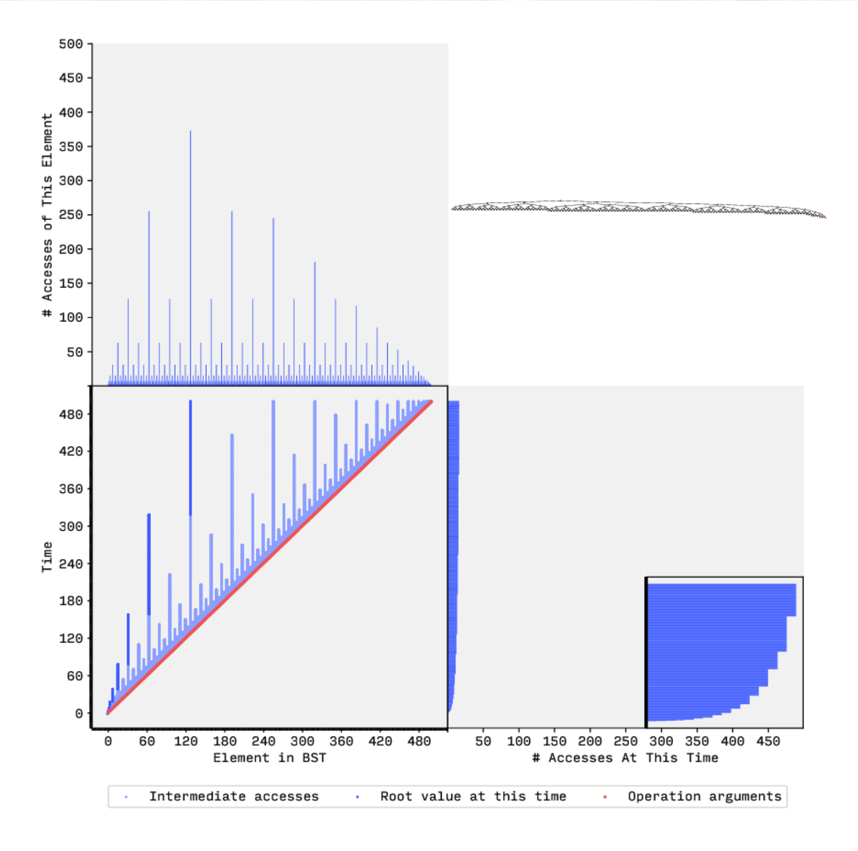




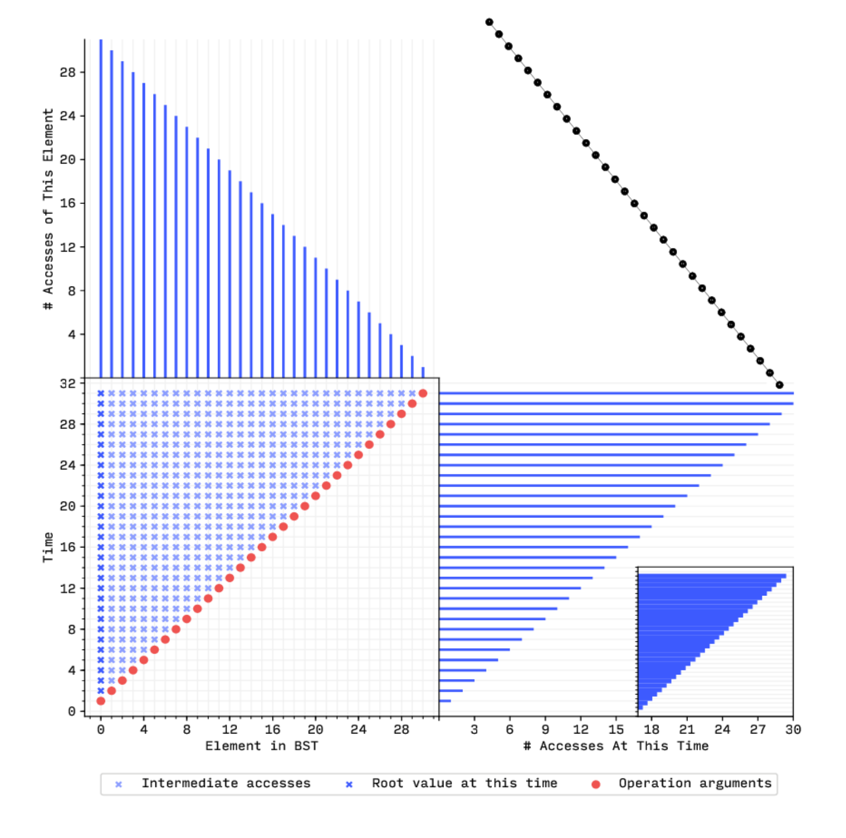


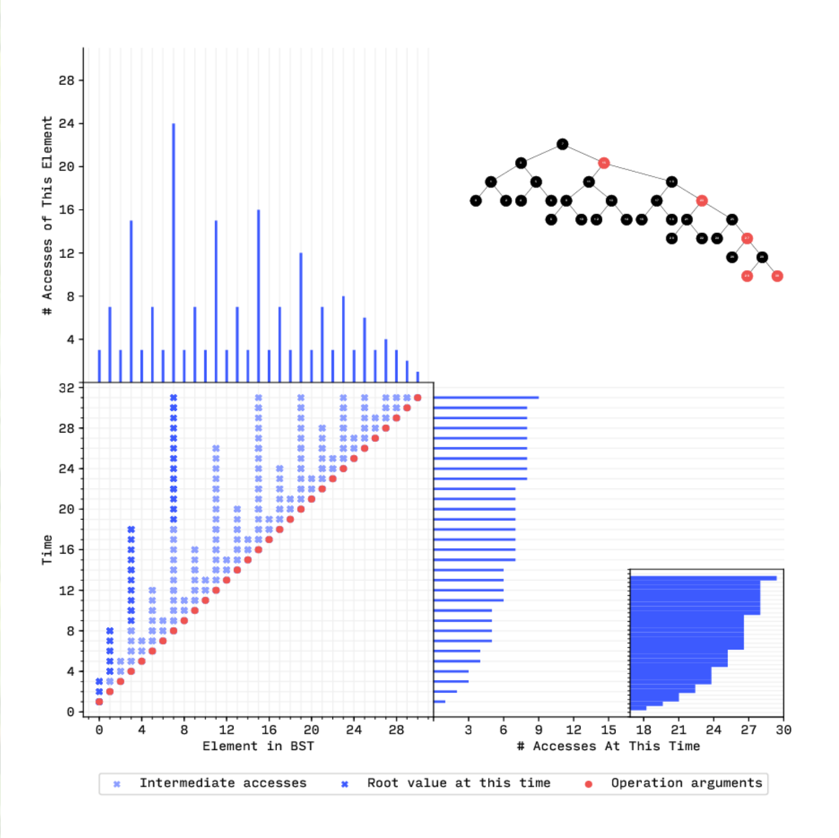
Time Complexity



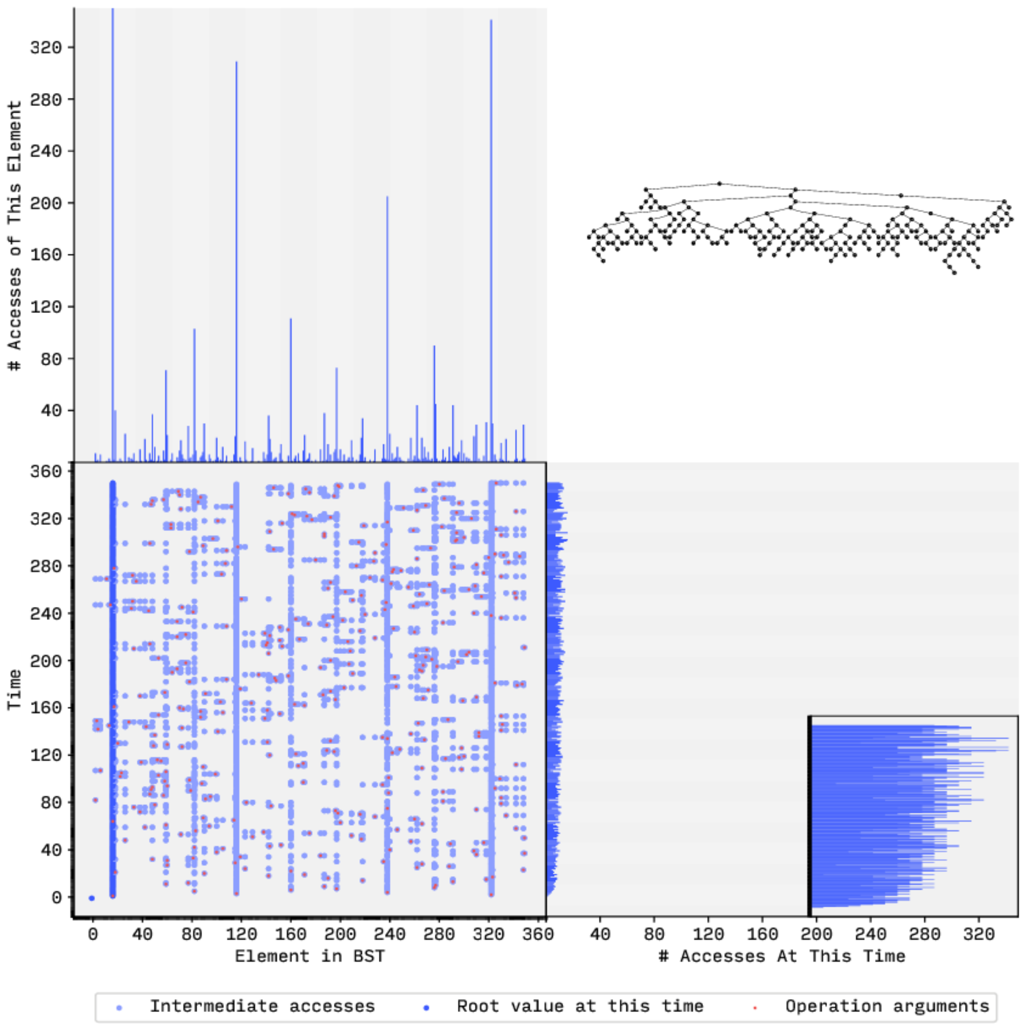


Time Complexity

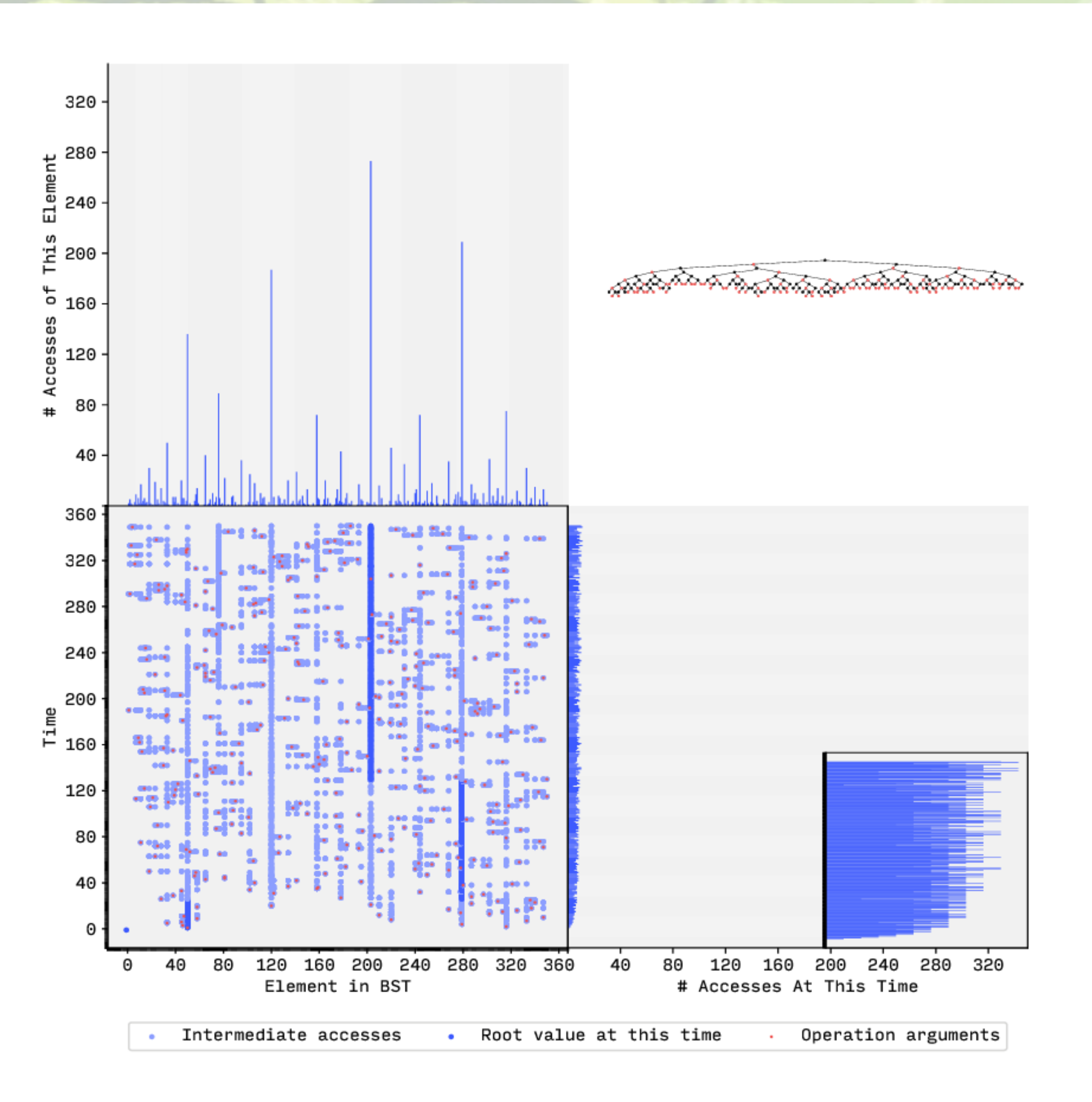




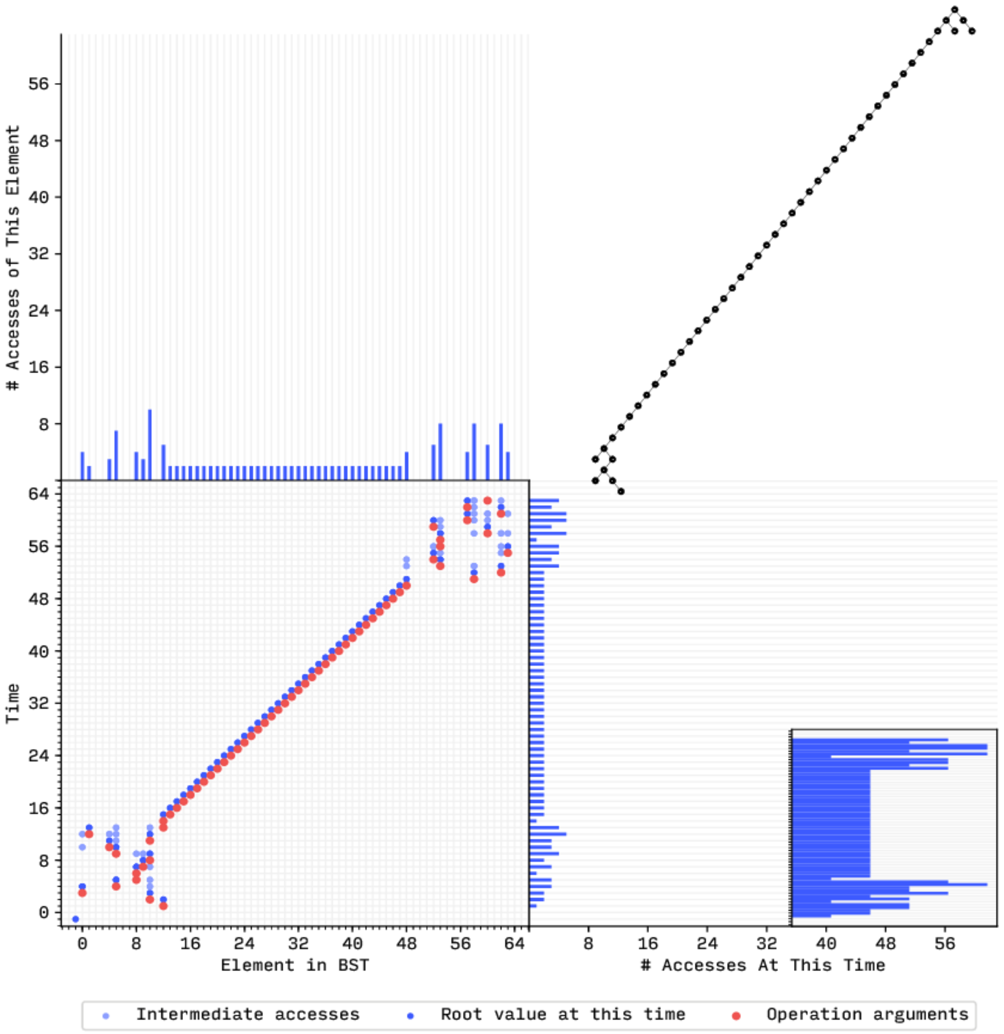
Simple BST, Random Inserts



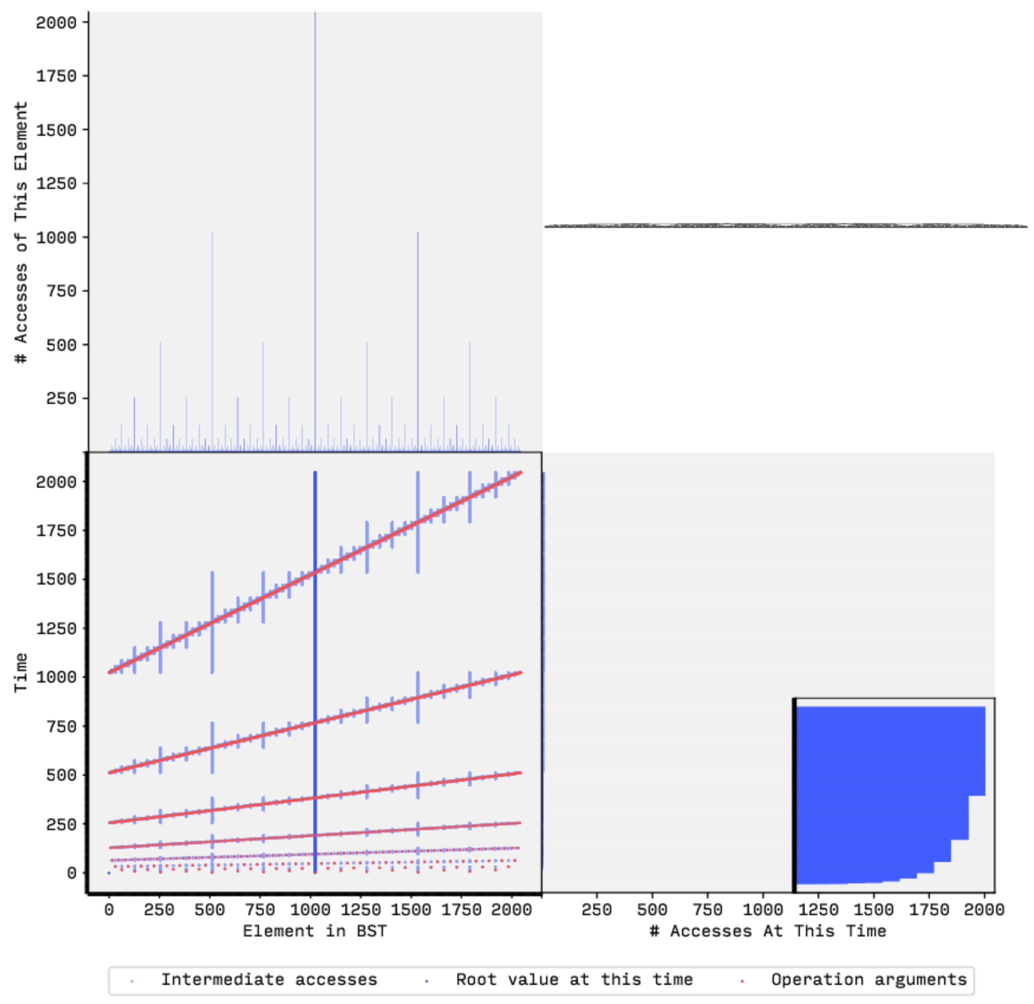
Red–Black Tree, Random Inserts



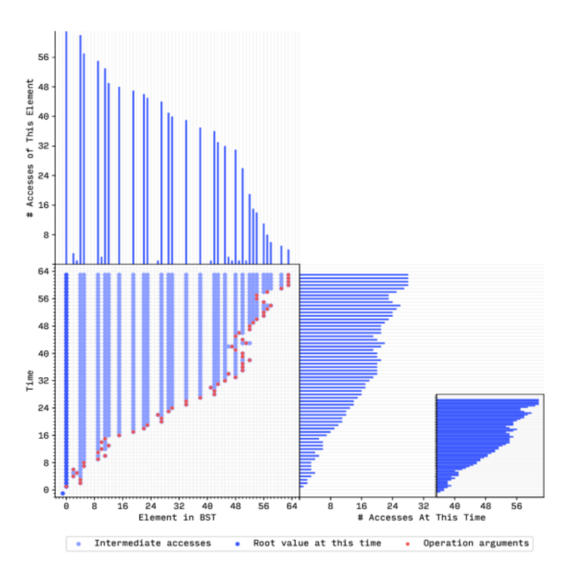
Splay Tree: Random Inserts, Then Increasing Inserts, Then Random

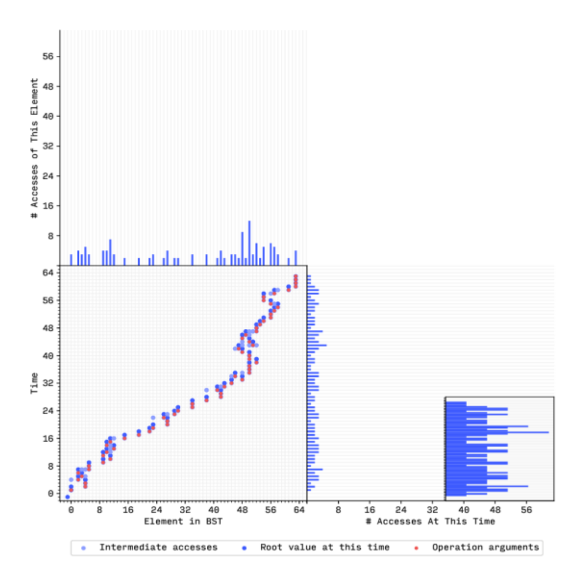


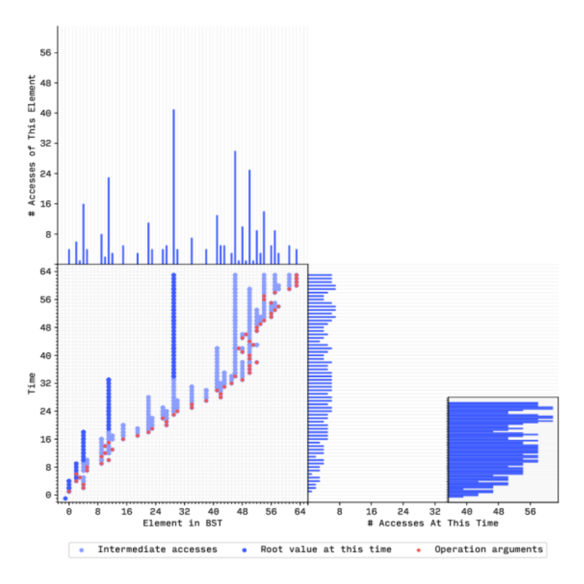
Simple BST: Perfectly Balanced Inserts

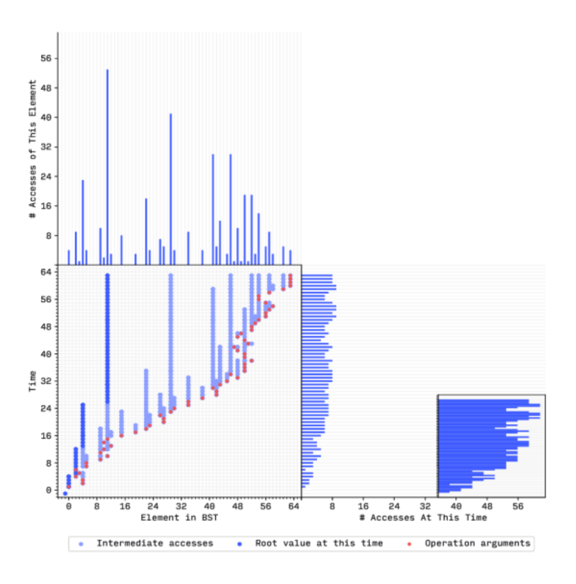


A Game: AVL, Splay, Simple, Red–Black

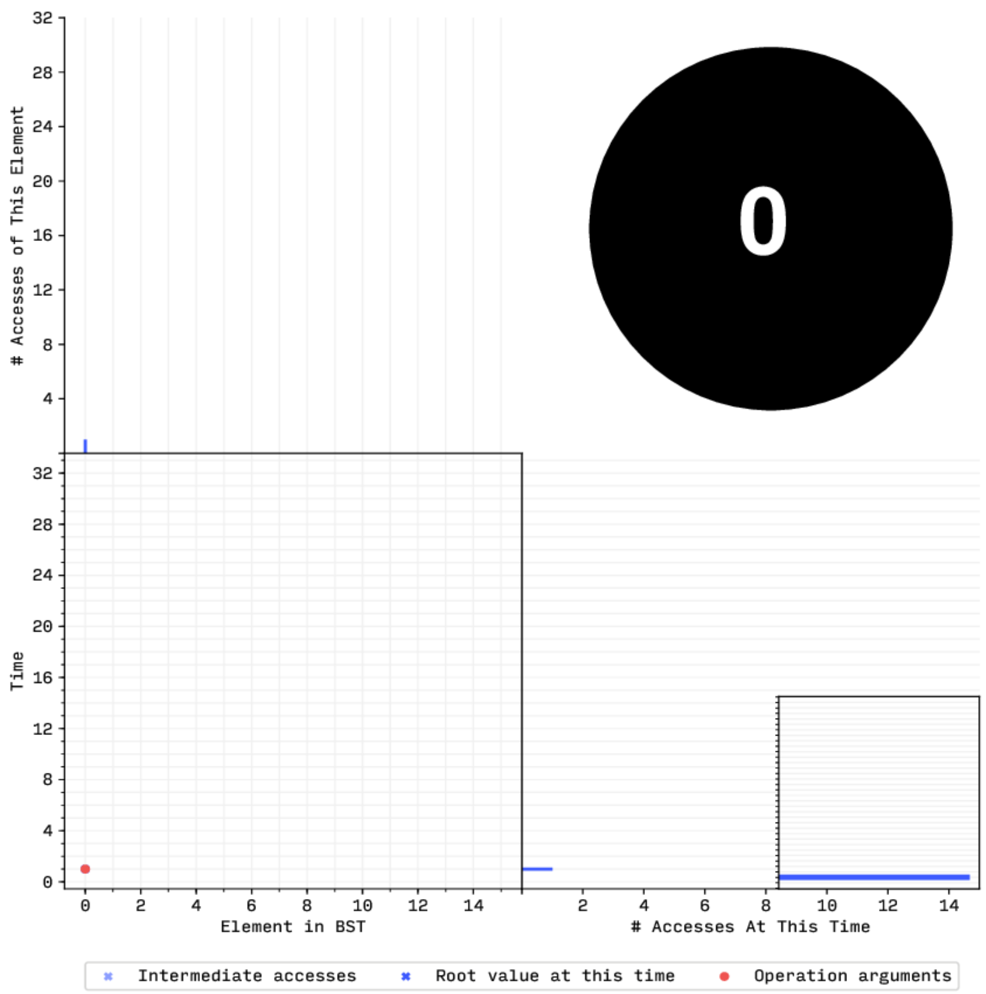




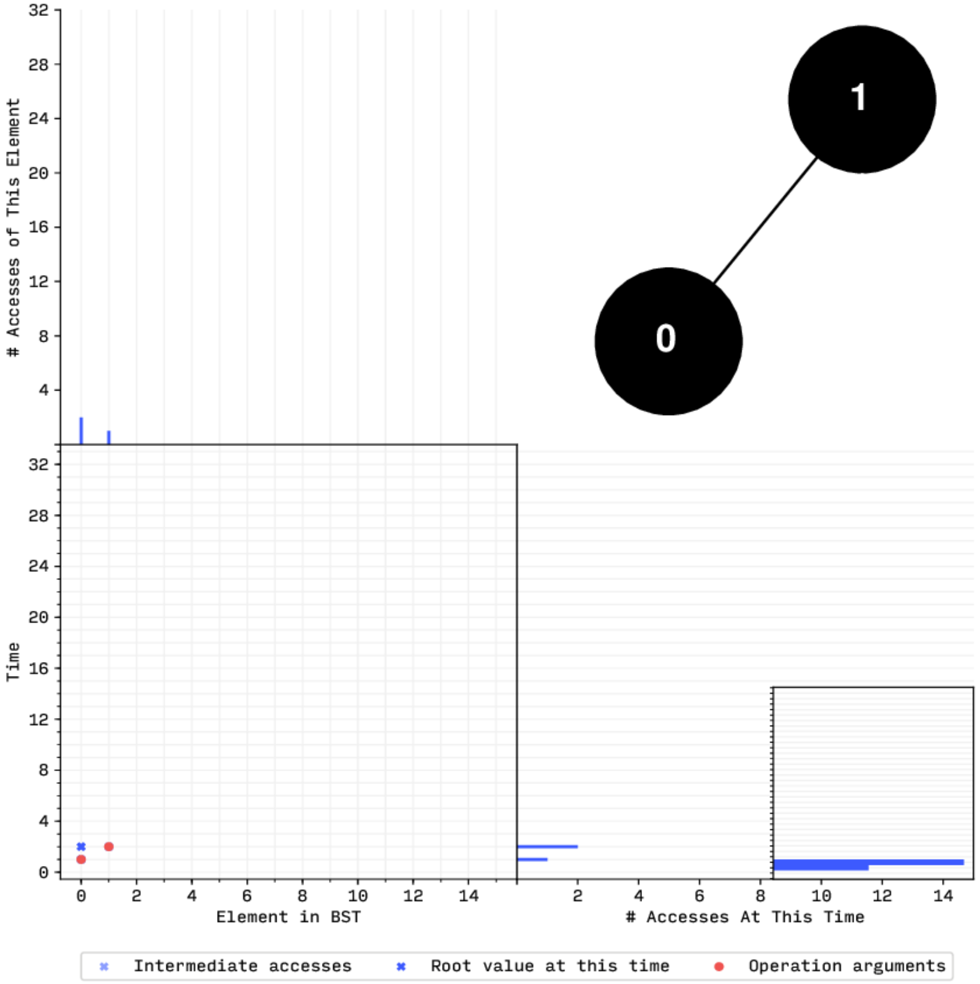




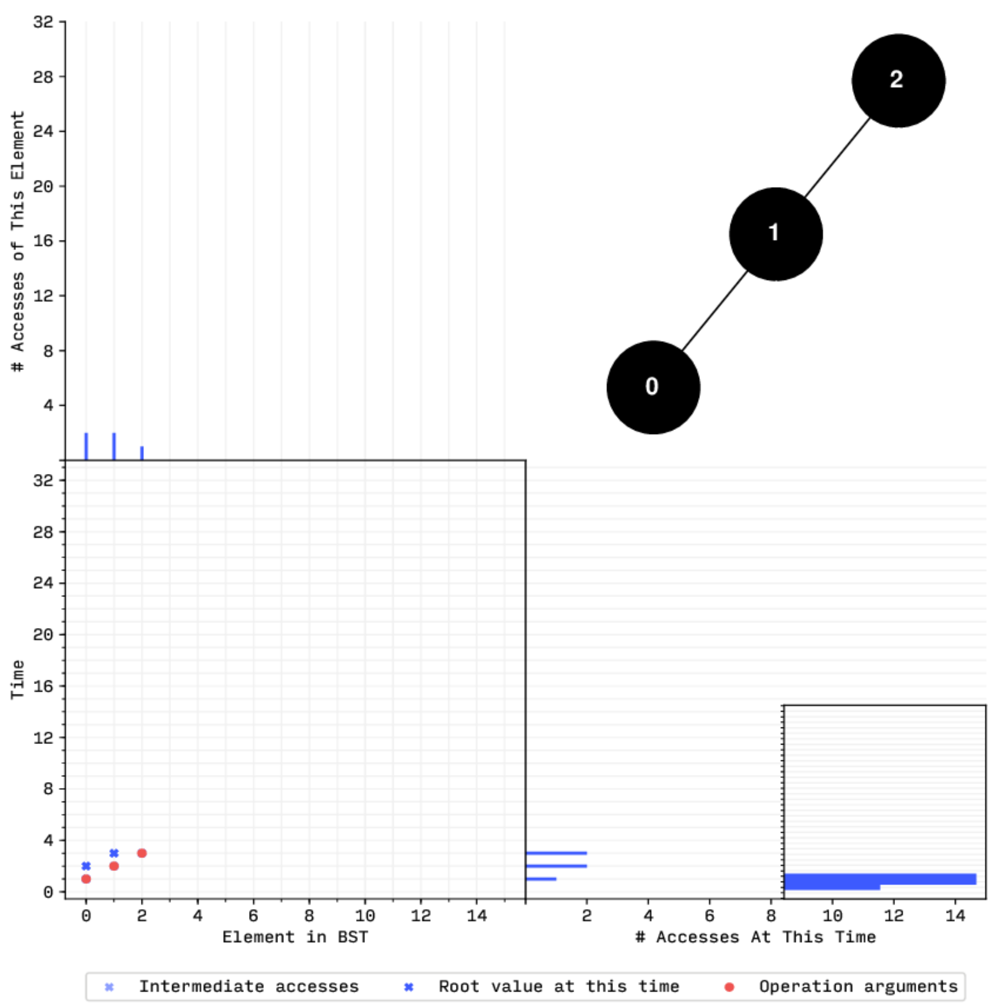
Splay Tree: Increasing Inserts, Decreasing Searches



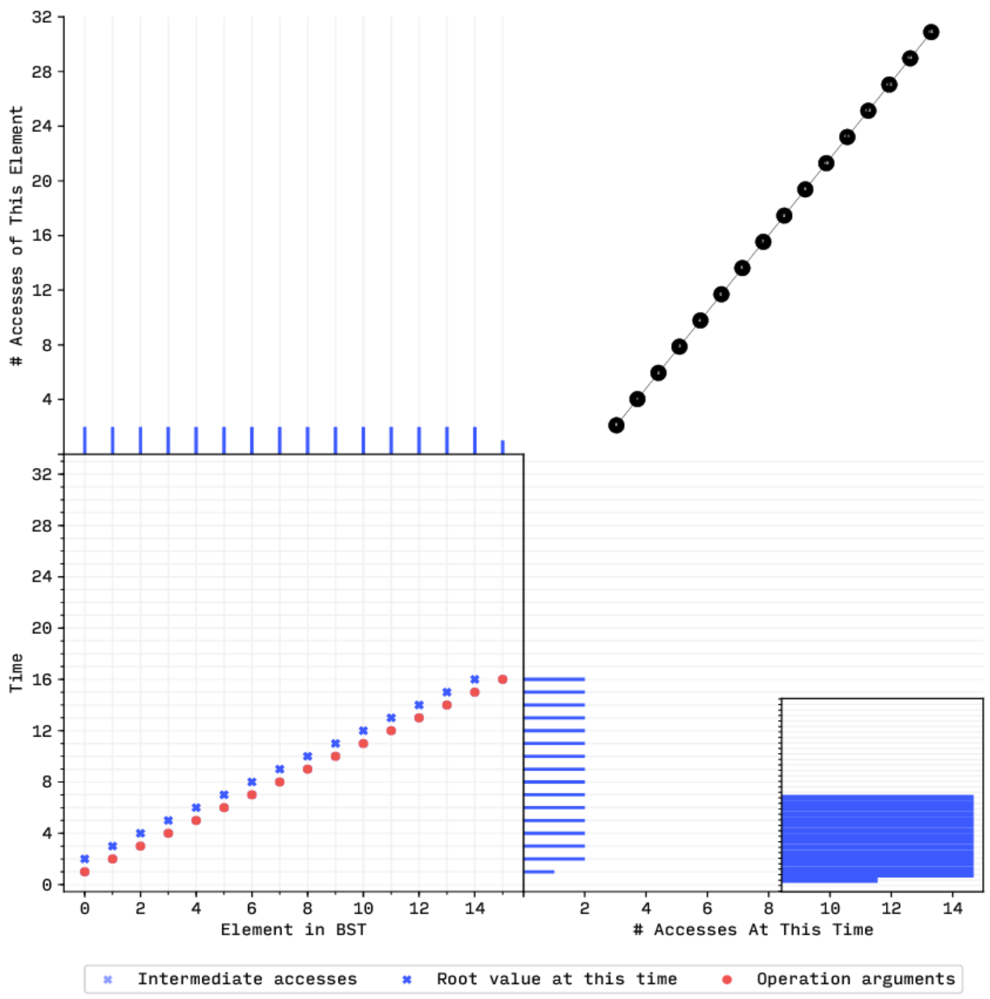
Splay Tree: Increasing Inserts, Decreasing Searches



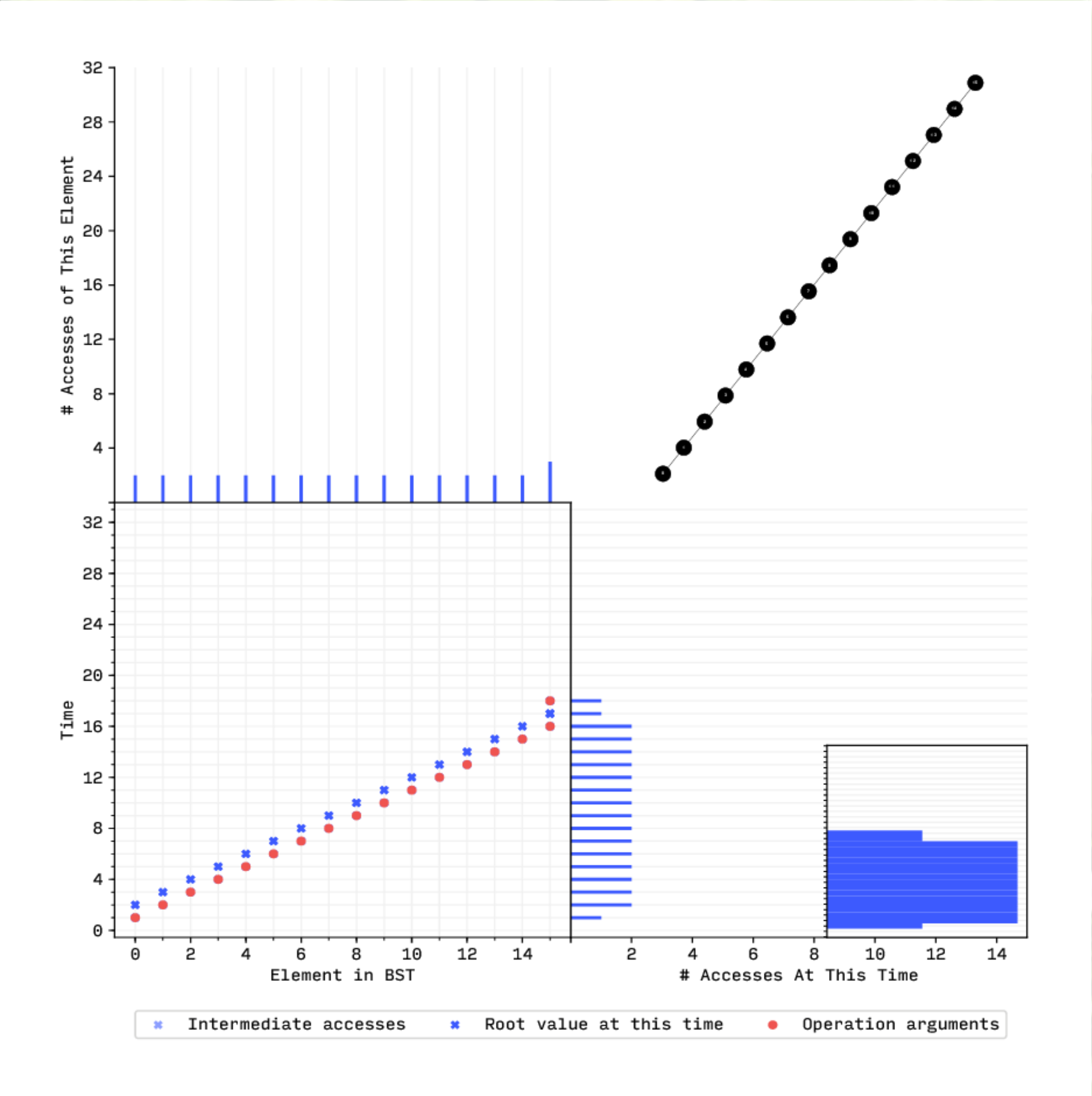
Splay Tree: Increasing Inserts, Decreasing Searches



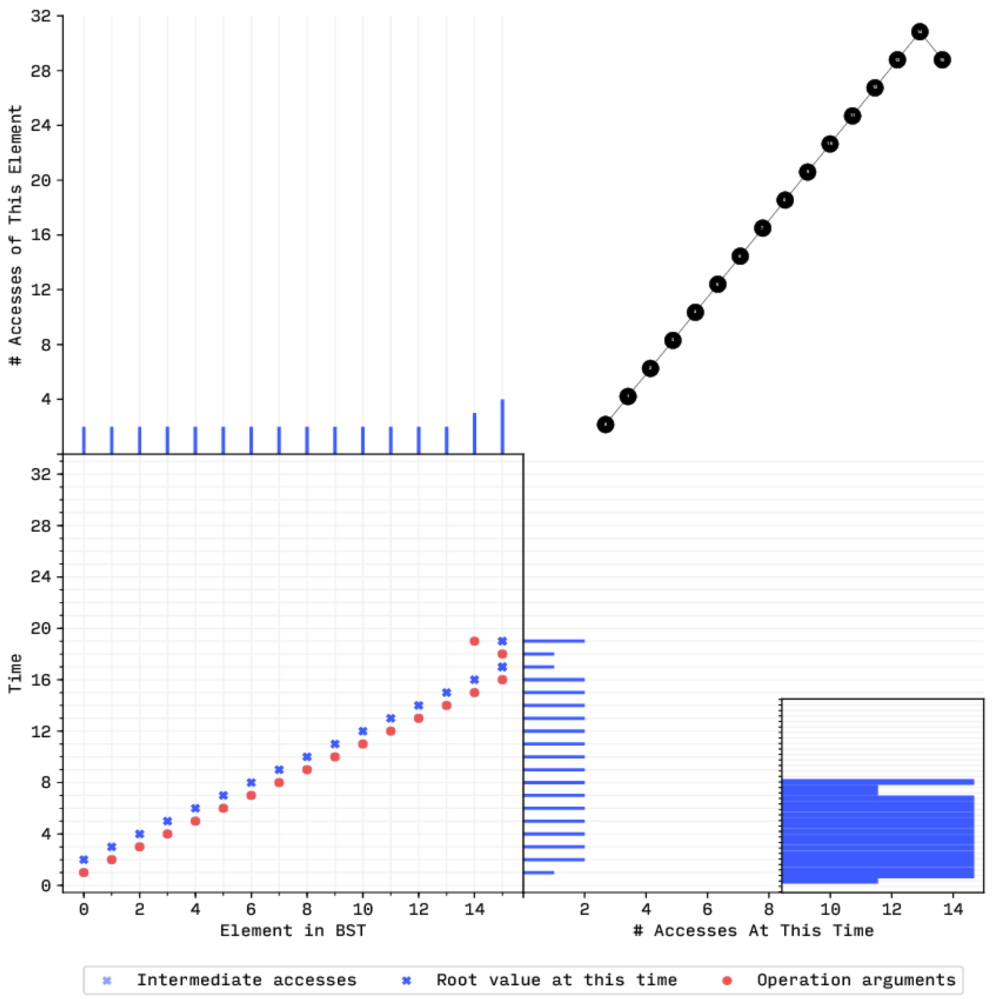
Splay Tree: Increasing Inserts, Decreasing Searches



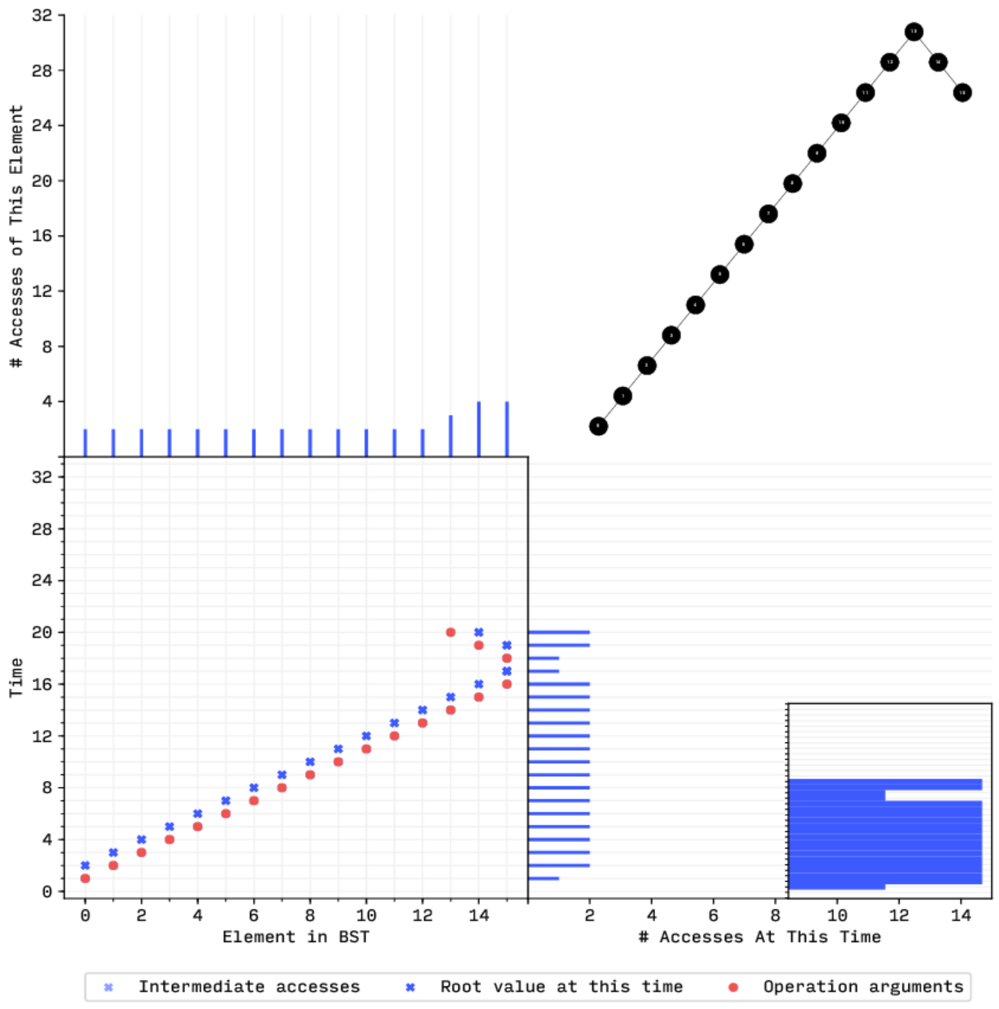
Splay Tree: Increasing Inserts, Decreasing Searches



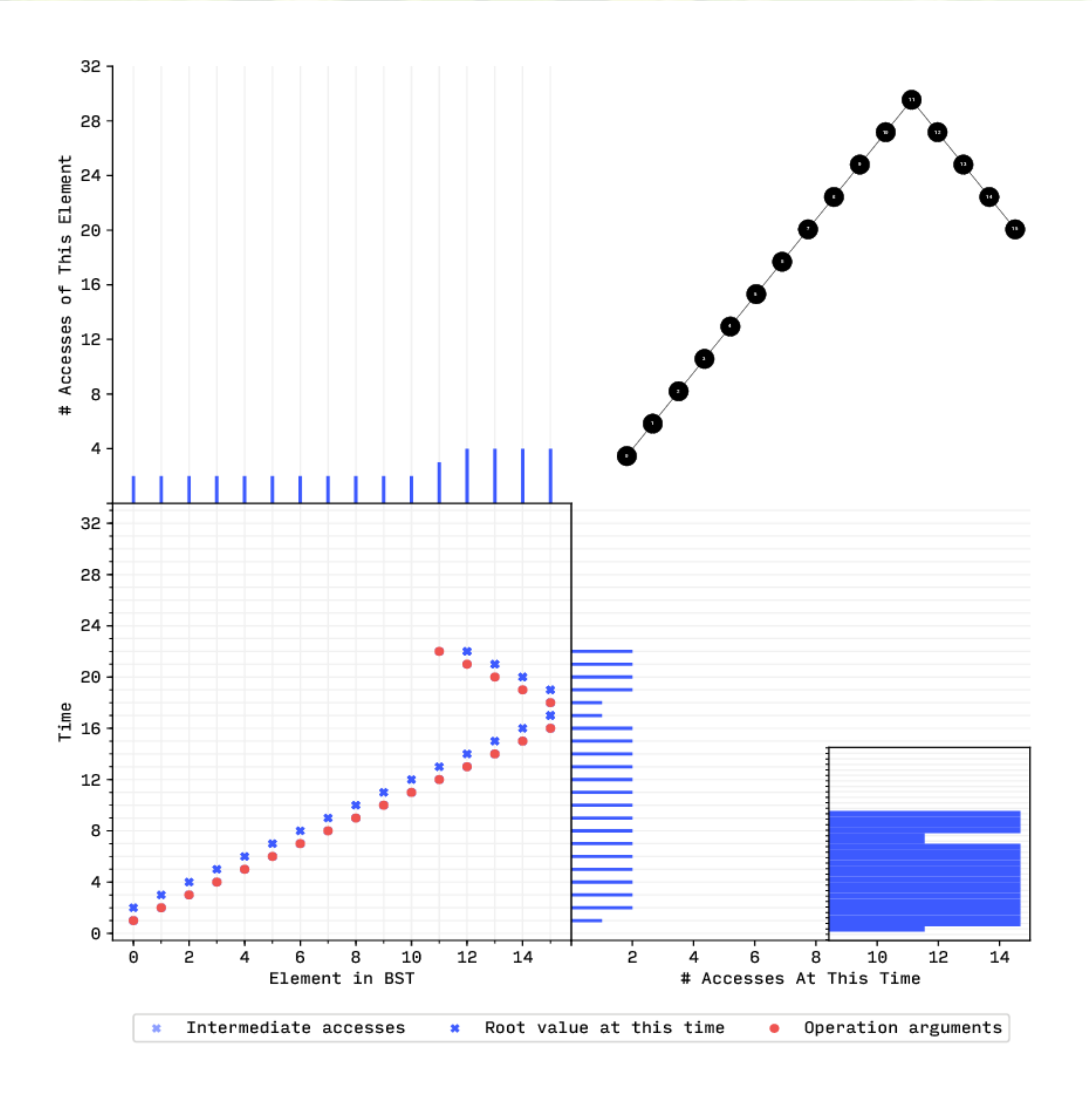
Splay Tree: Increasing Inserts, Decreasing Searches



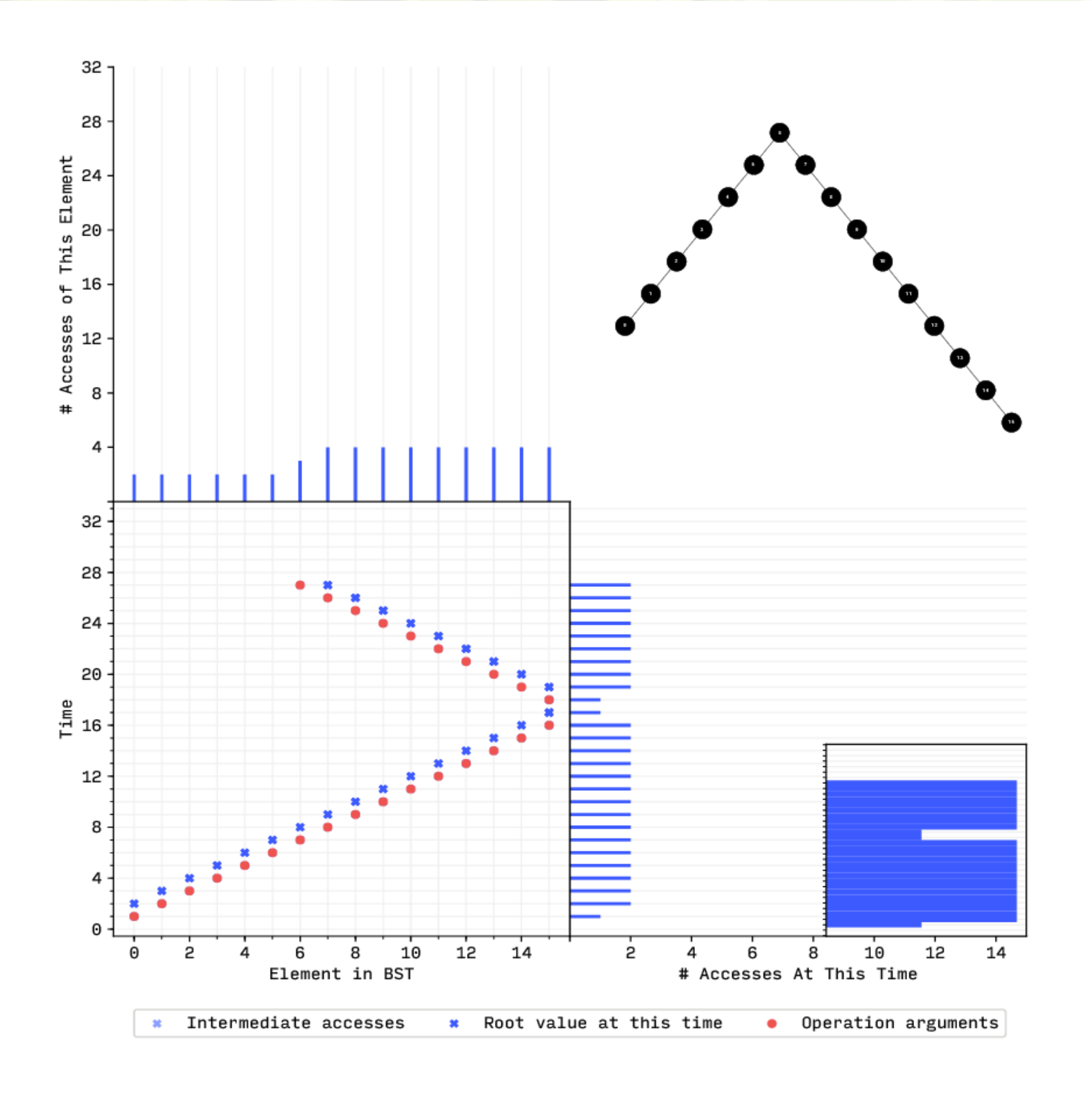
Splay Tree: Increasing Inserts, Decreasing Searches



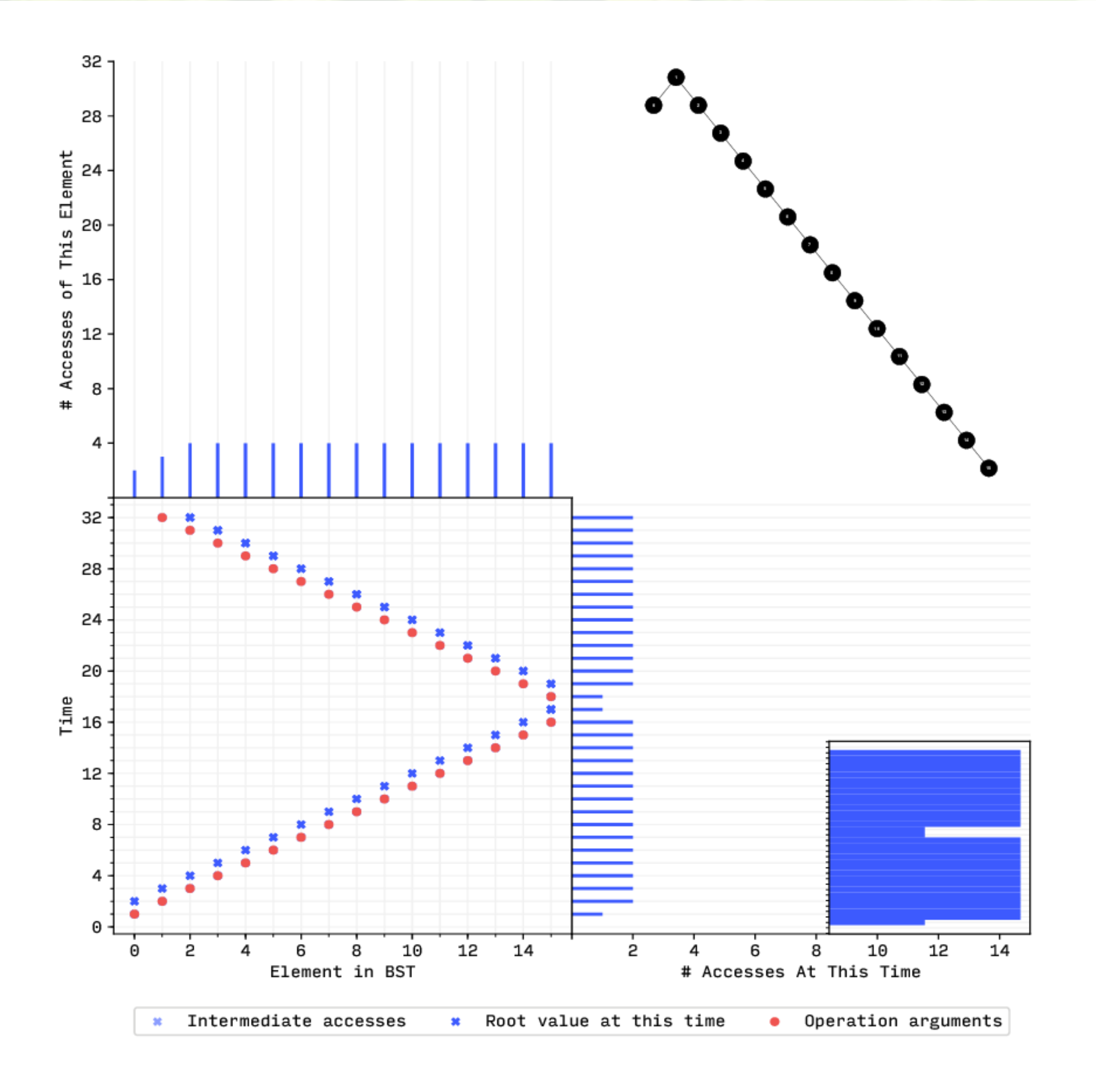
Splay Tree: Increasing Inserts, Decreasing Searches



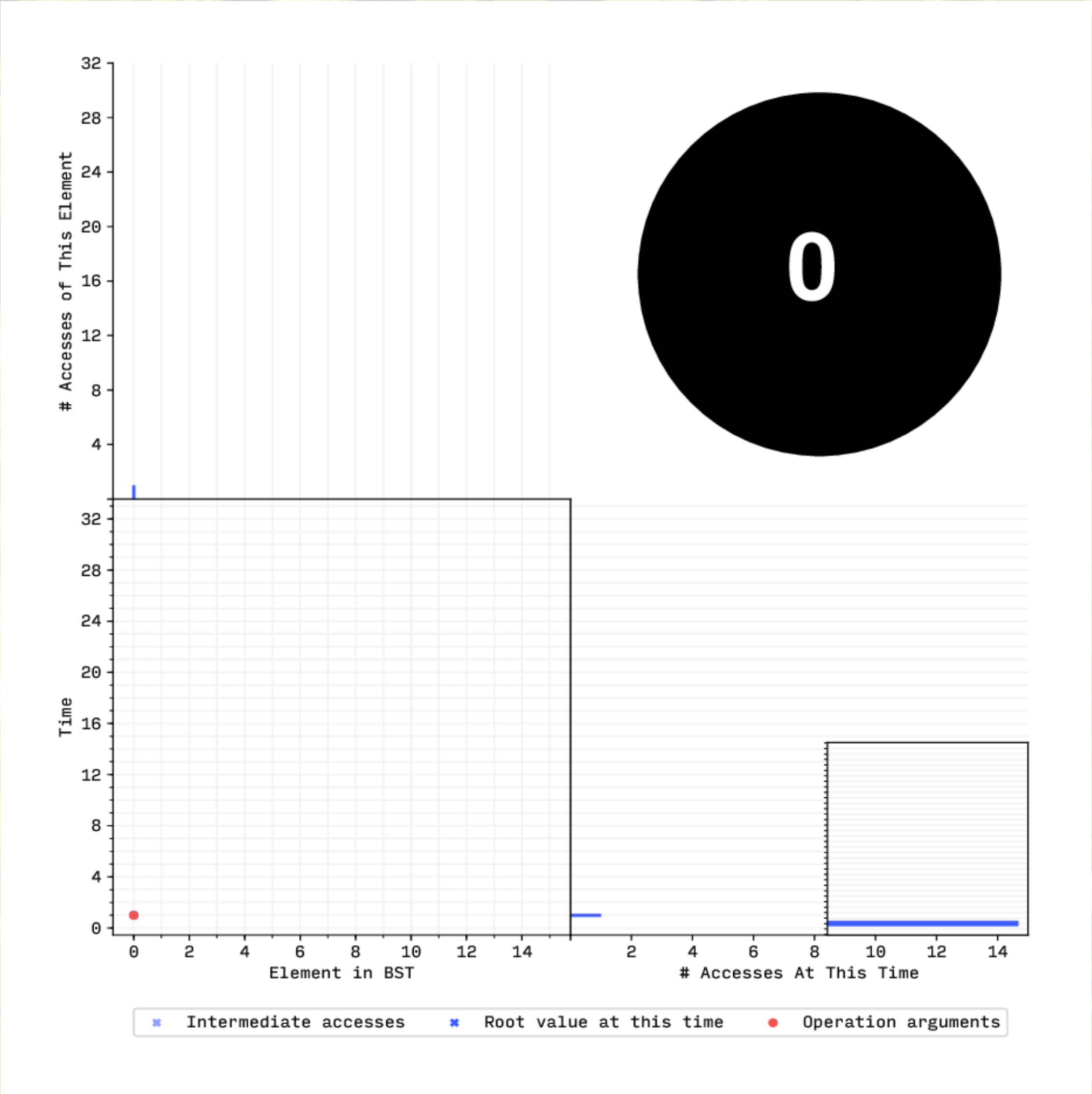
Splay Tree: Increasing Inserts, Decreasing Searches



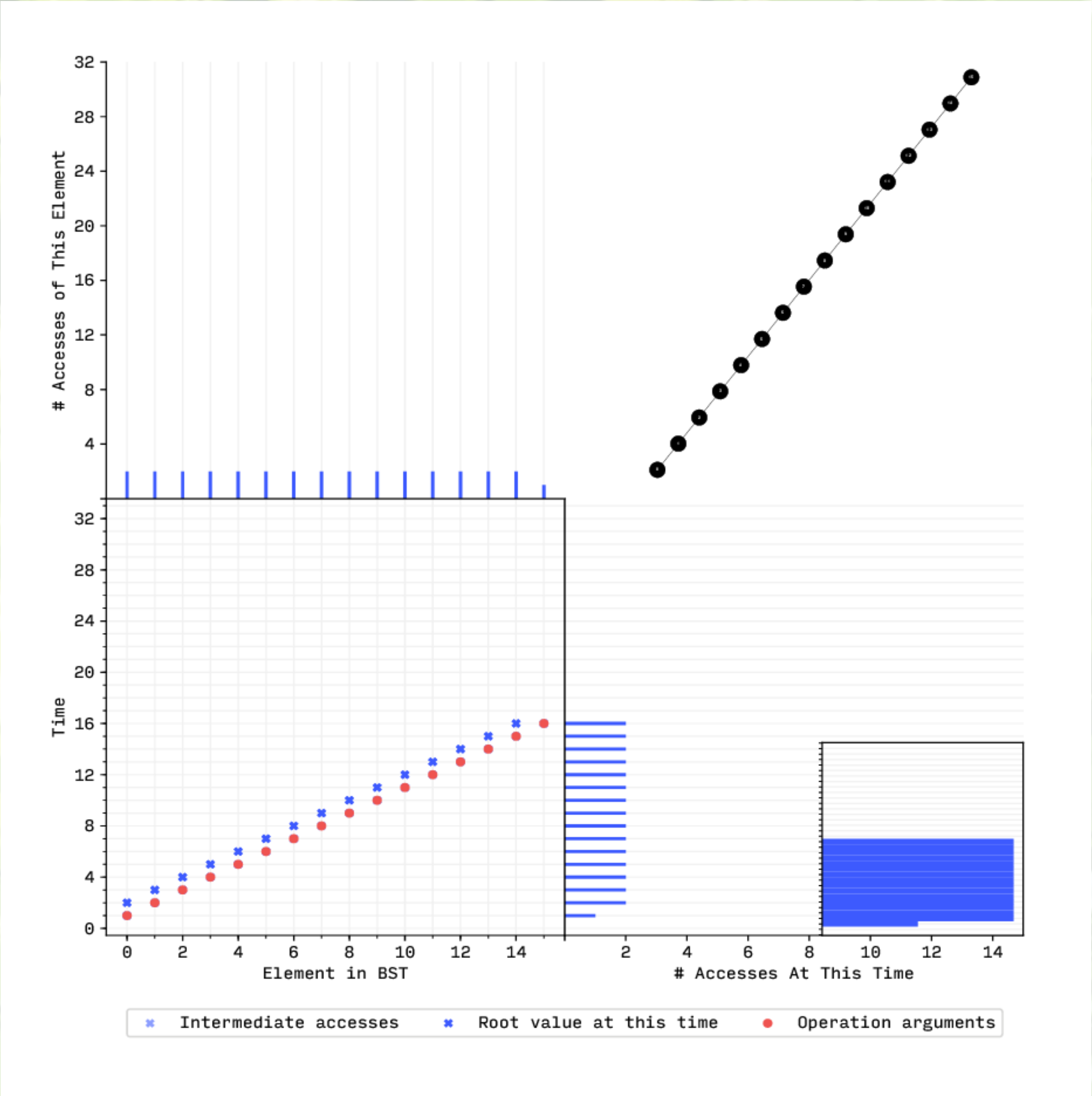
Splay Tree: Increasing Inserts, Decreasing Searches



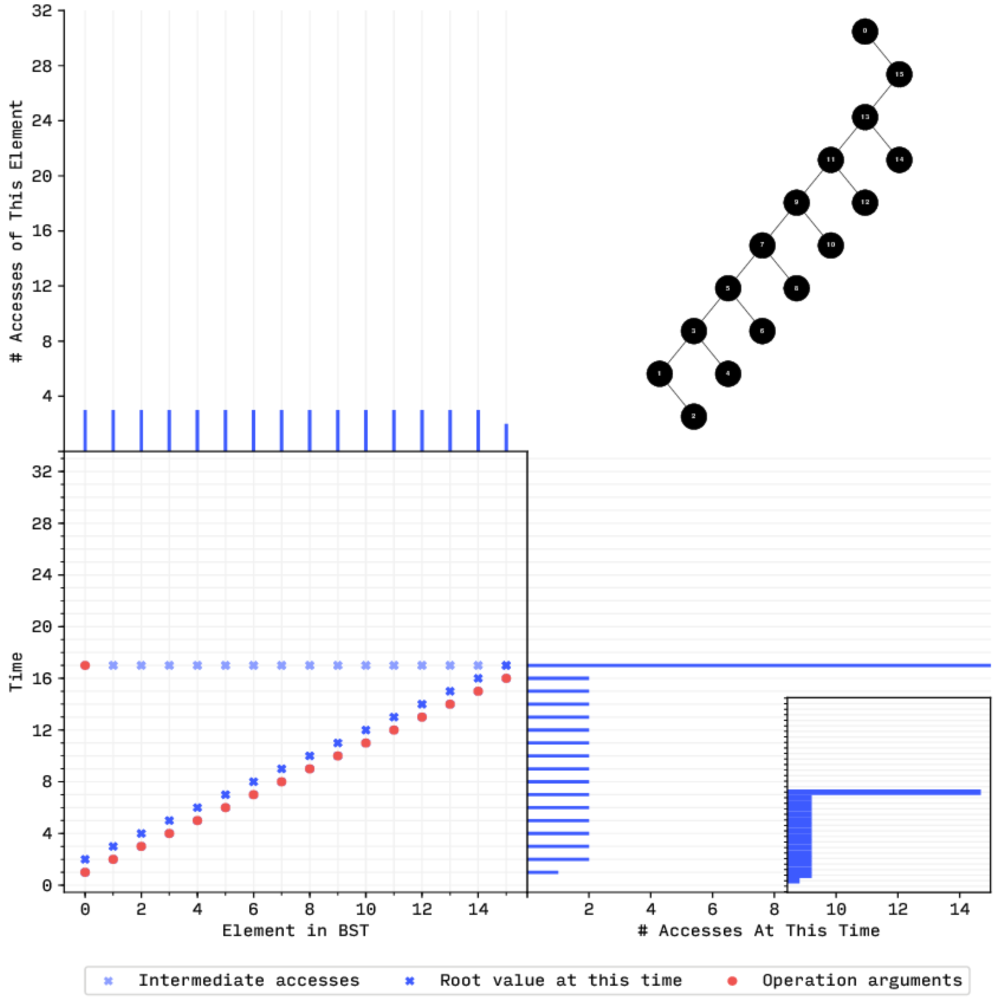
Splay Tree: Increasing Inserts, Increasing Searches



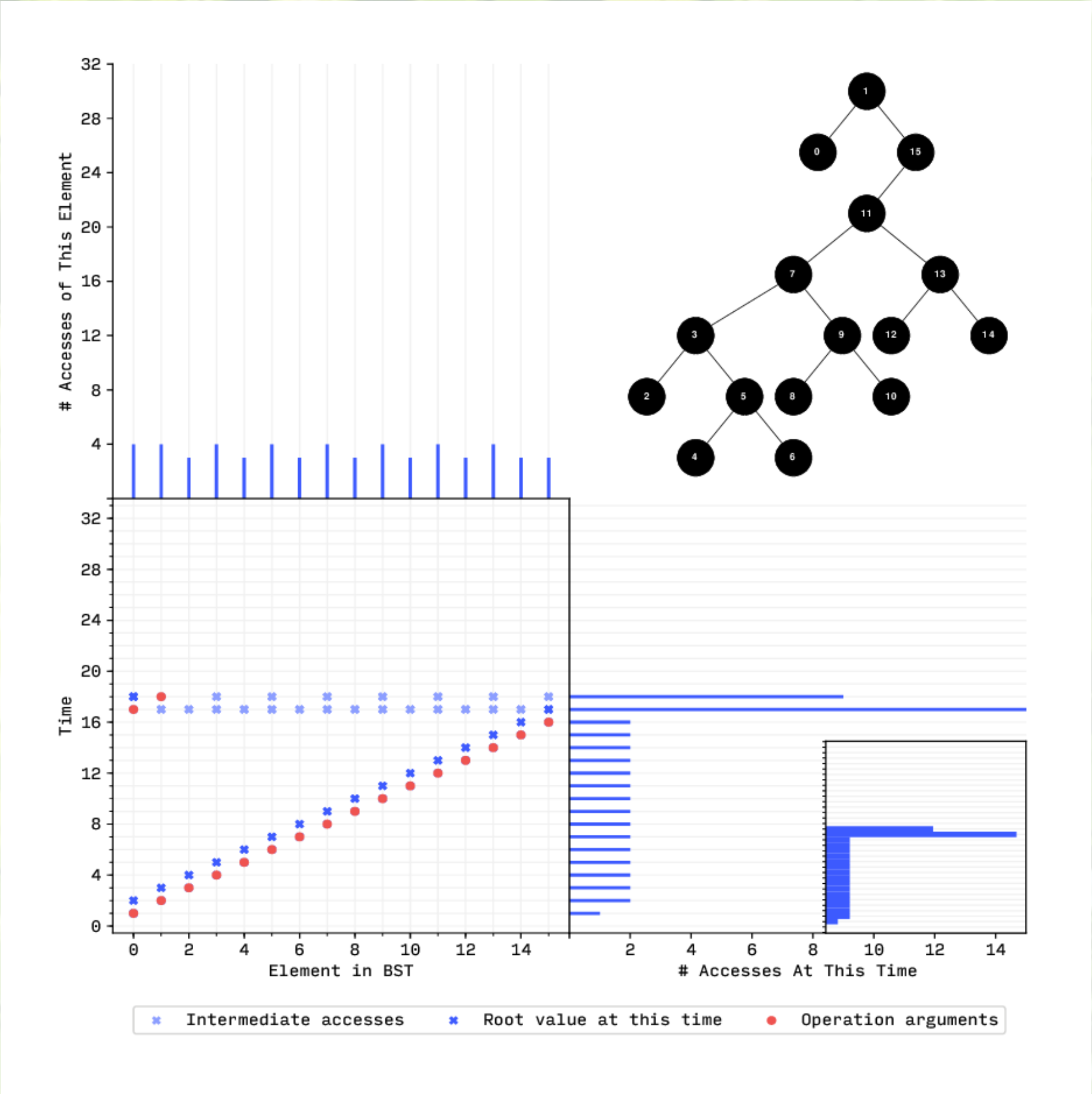
Splay Tree: Increasing Inserts, Increasing Searches



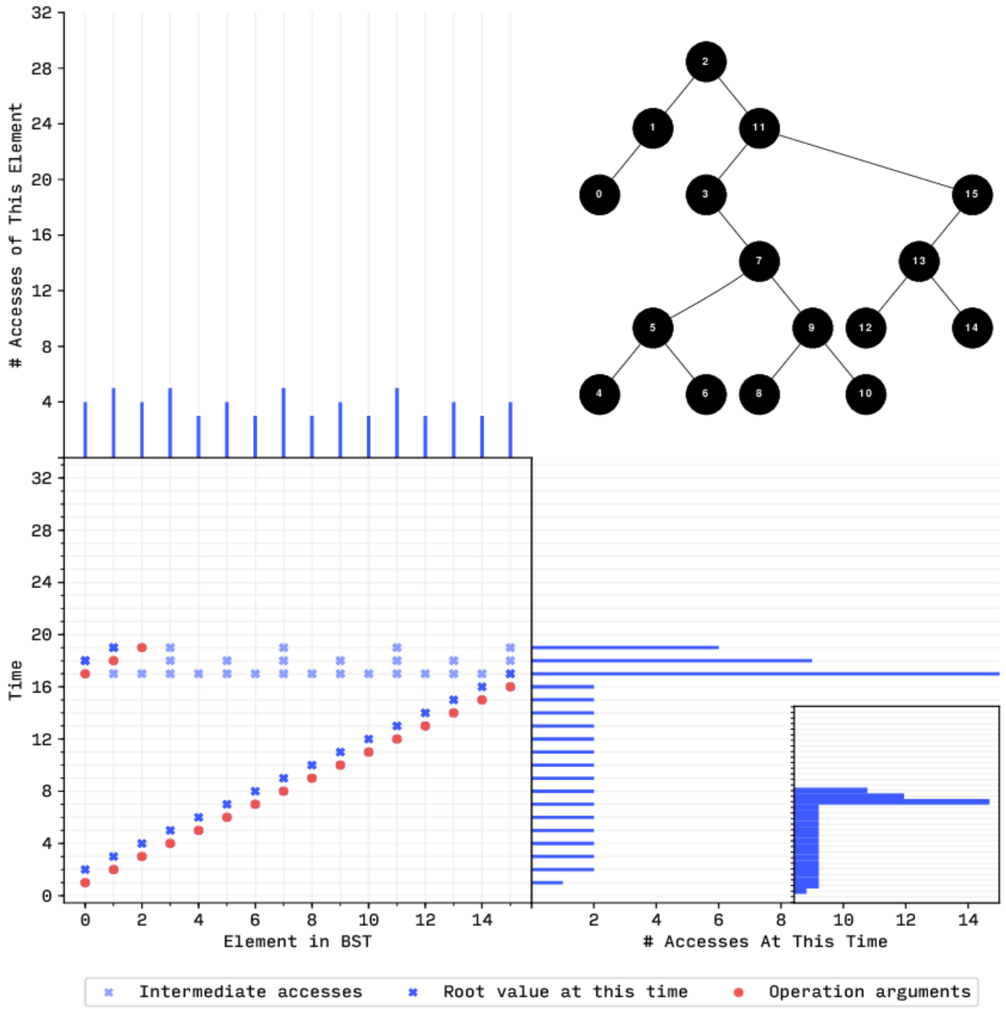
Splay Tree: Increasing Inserts, Increasing Searches



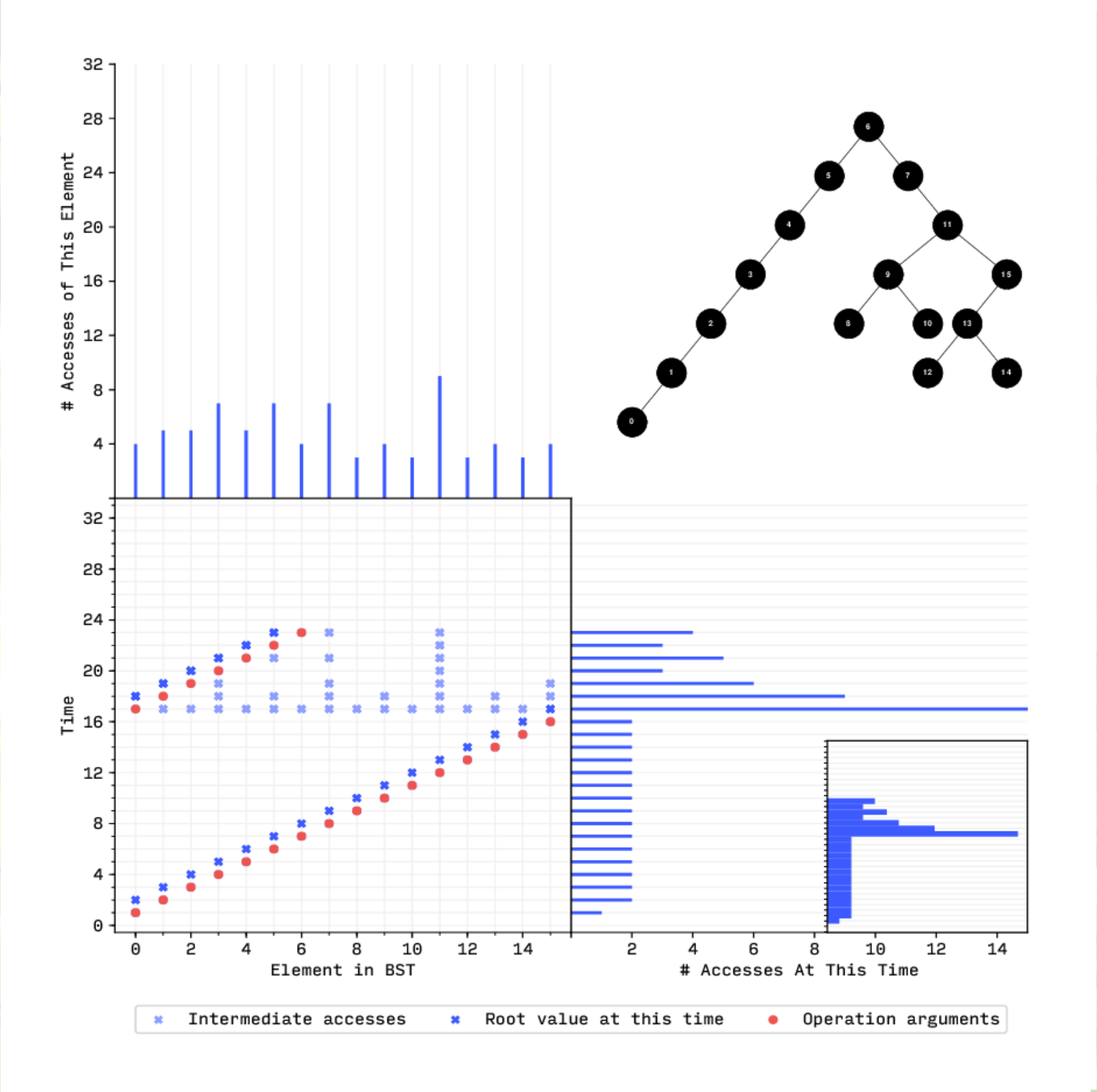
Splay Tree: Increasing Inserts, Increasing Searches



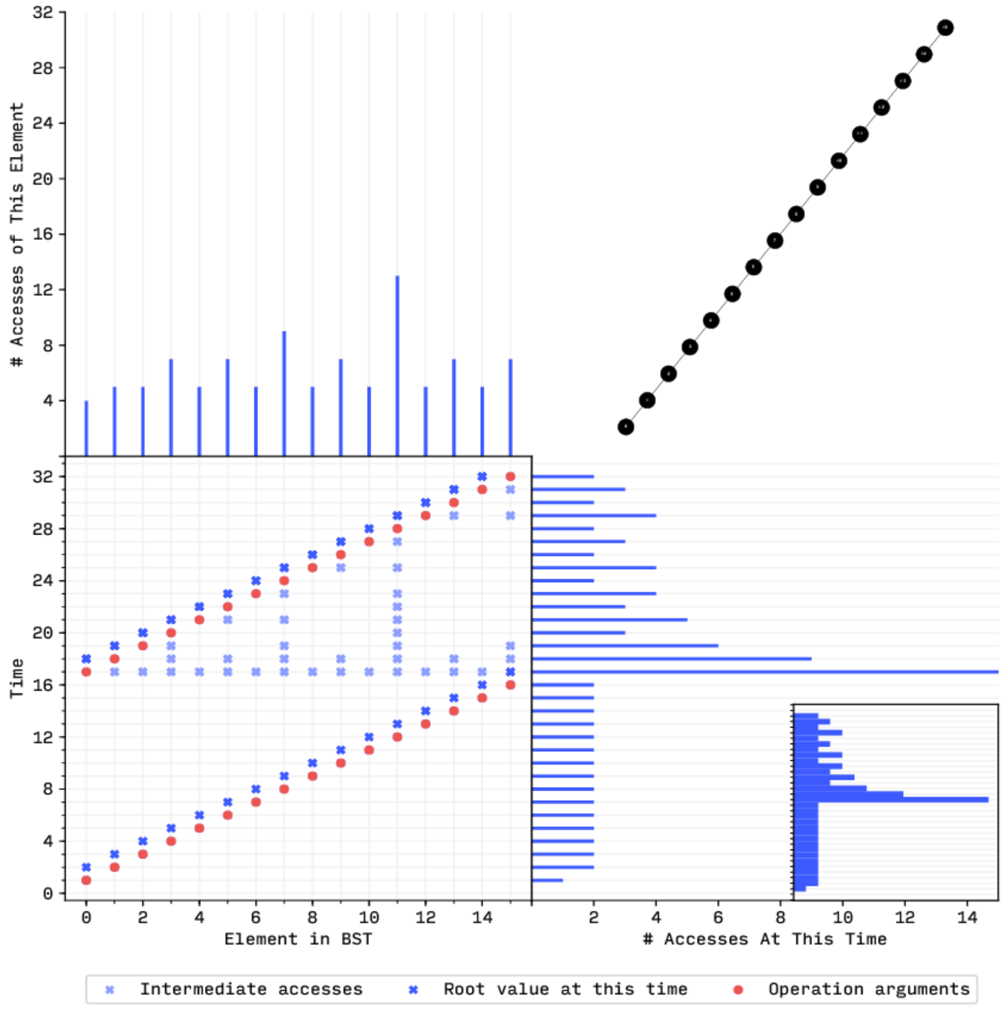
Splay Tree: Increasing Inserts, Increasing Searches



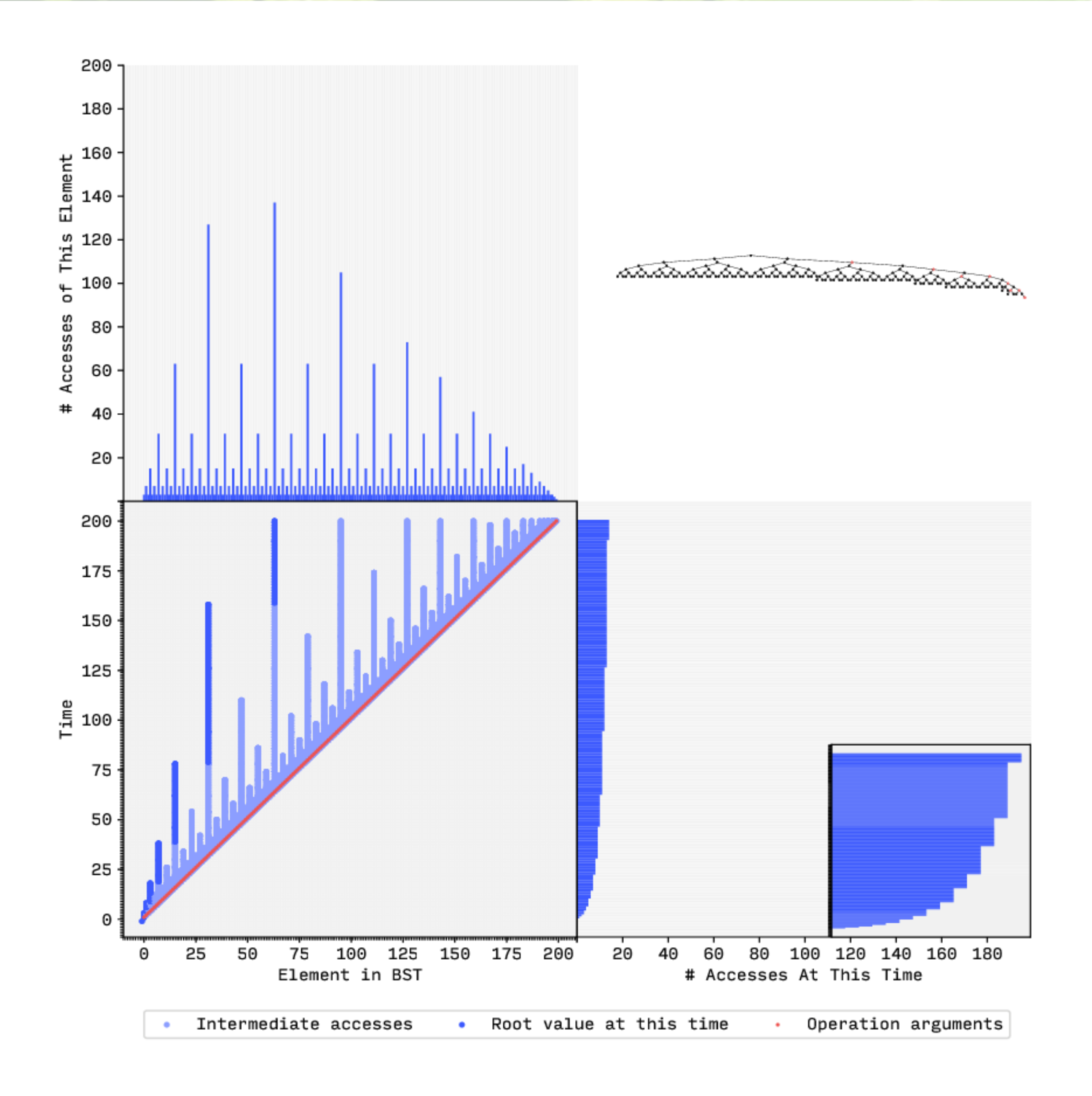
Splay Tree: Increasing Inserts, Increasing Searches



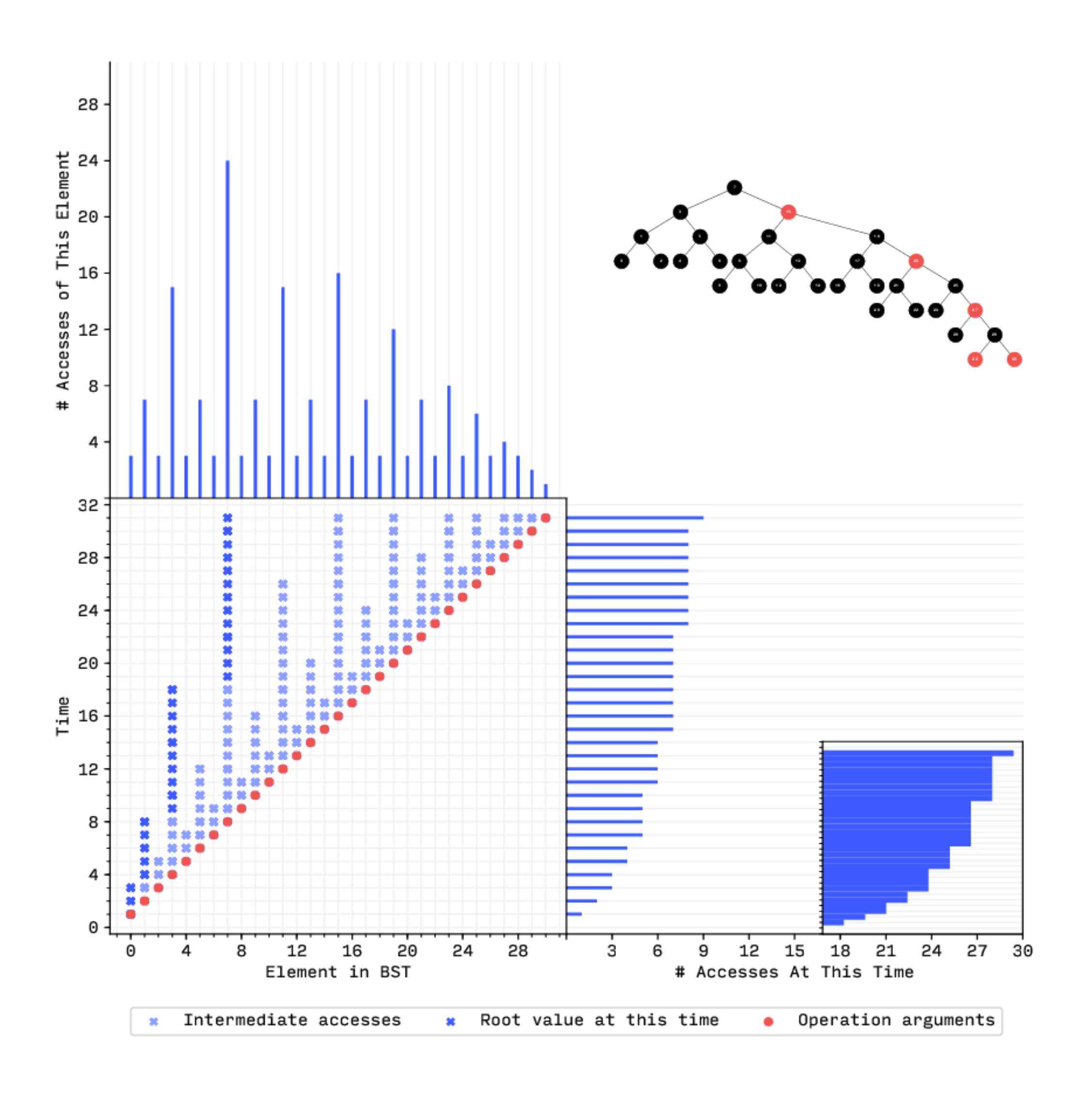
Splay Tree: Increasing Inserts, Increasing Searches



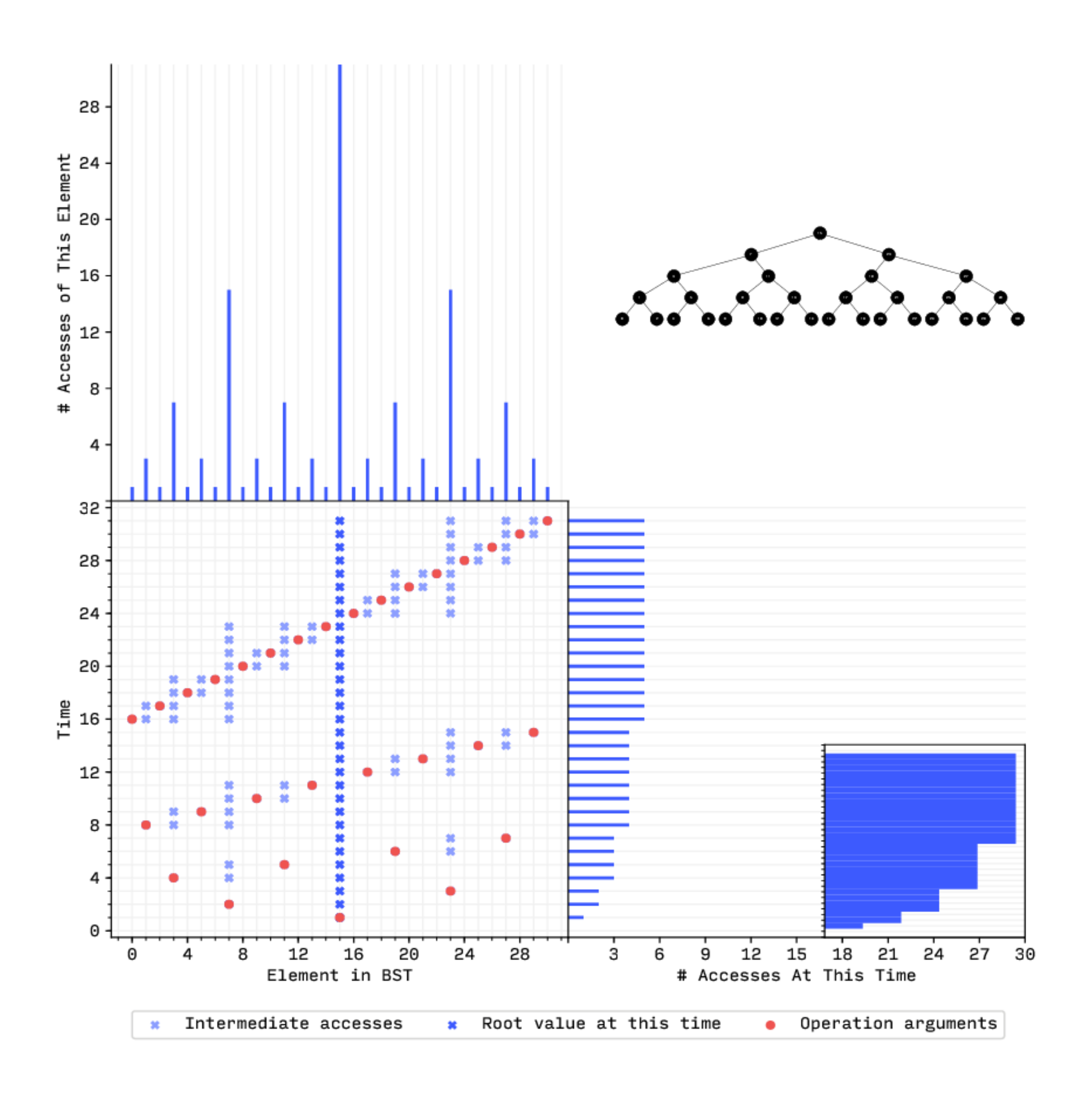
Red–Black Tree: Increasing Inserts



Red–Black Tree: Increasing Inserts



Simple BST: Balanced Inserts



Thank You!

GitHub Repository:

https://github.com/forsooth/BST-analysis