Portal

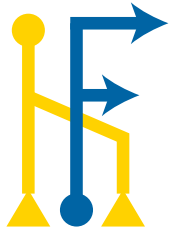
A High-Performance Language and Compiler for Parallel N-body Problems

Laleh Aghababaie Beni, Saikiran Ramanan

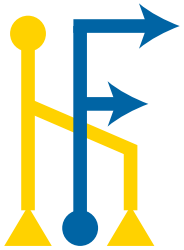
Prof. Aparna Chandramowlishwaran

University of California, Irvine

IPDPS 2019

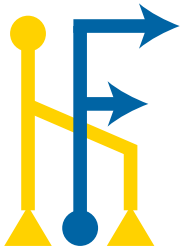


N-body Problems



N-body Problems

- General representation $op_1, \ldots, op_m \;\; K\left(x_1, \ldots, x_m\right)$
  - Nearest Neighbor $\forall q \in Q, \;\; \arg\min_{r \in R} \;\; \|x_q - x_r\|$



N-body Problems

- General representation

$$op_1, \ldots, op_m \;\; K\left(x_1, \ldots, x_m\right)$$

  - Nearest Neighbor

$$\forall q \in Q, \;\; \arg\min_{r \in R} \|x_q - x_r\|$$

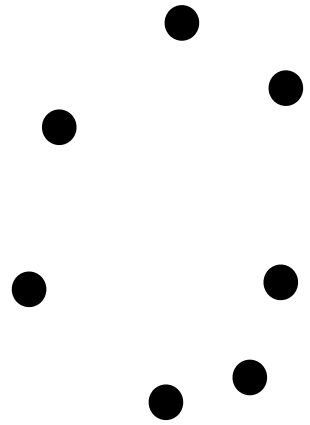
N-body Problems

- General representation

$$op_1, \ldots, op_m \quad K\left(x_1, \ldots, x_m\right)$$

  - Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \|x_q - x_r\|$$



Nearest Neighbor?

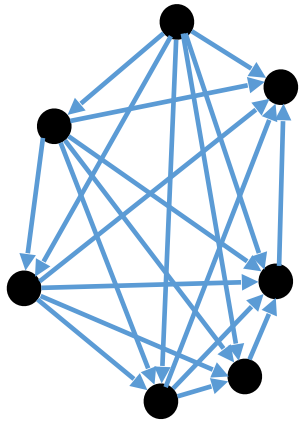
N-body Problems

- General representation

$$op_1, \ldots, op_m \;\; K\left(x_1, \ldots, x_m\right)$$

  - Nearest Neighbor

$$\forall q \in Q, \;\; \arg\min_{r \in R} \|x_q - x_r\|$$



Nearest Neighbor?

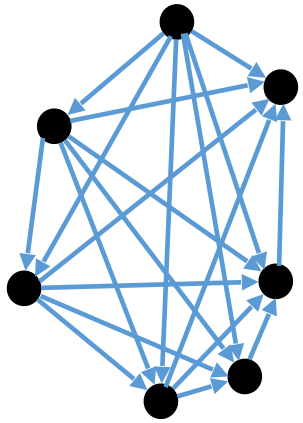
N-body Problems

- General representation

$$op_1, \ldots, op_m \quad K\left(x_1, \ldots, x_m\right)$$

  - Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \|x_q - x_r\|$$



Nearest Neighbor?

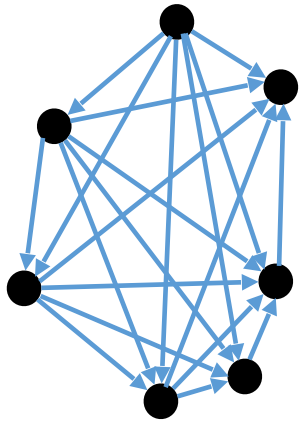
N-body Problems

- General representation

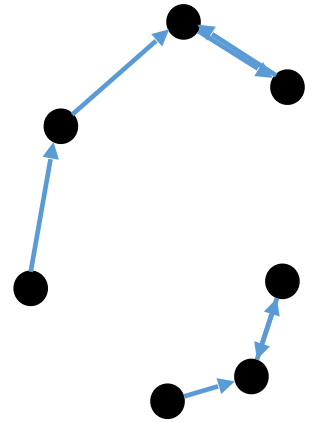
$$op_1, \ldots, op_m \quad K\left(x_1, \ldots, x_m\right)$$

  - Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \|x_q - x_r\|$$



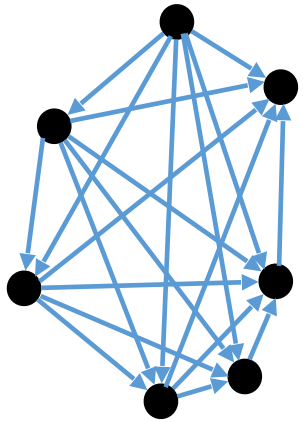
Nearest Neighbor?



Nearest Neighbors

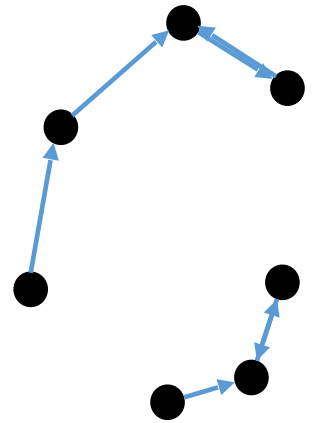
N-body Problems

- General representation $op_1, \ldots, op_m \;\; K(x_1, \ldots, x_m)$
  - Nearest Neighbor $\forall q \in Q, \; \arg\min_{r \in R} \|x_q - x_r\|$



Nearest Neighbor?

Complexity?



Nearest Neighbors

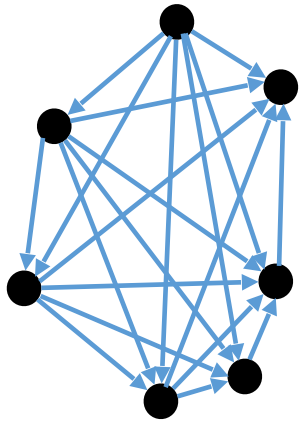
N-body Problems

- General representation

$$op_1, \ldots, op_m \quad K\left(x_1, \ldots, x_m\right)$$

  - Nearest Neighbor

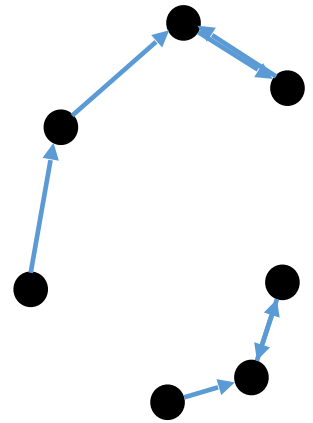
$$\forall q \in Q, \quad \arg\min_{r \in R} \|x_q - x_r\|$$



Nearest Neighbor?

Complexity?

O(N²)



Nearest Neighbors

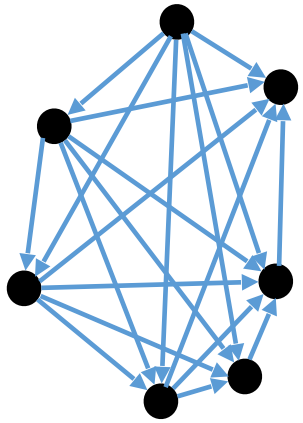
N-body Problems

- General representation

$$op_1, \ldots, op_m \quad K\left(x_1, \ldots, x_m\right)$$

  - Nearest Neighbor

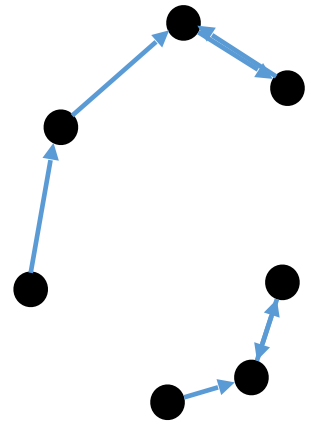
$$\forall q \in Q, \quad \arg\min_{r \in R} \|x_q - x_r\|$$



Nearest Neighbor?

Complexity?

O(N²)



Nearest Neighbors

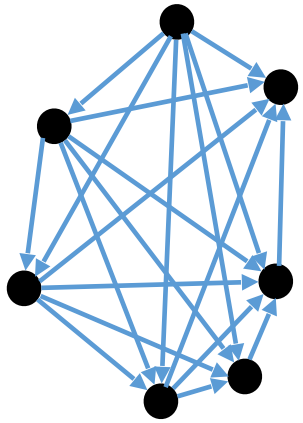
N-body Problems

- General representation

$$op_1, \ldots, op_m \quad K\left(x_1, \ldots, x_m\right)$$

  - Nearest Neighbor

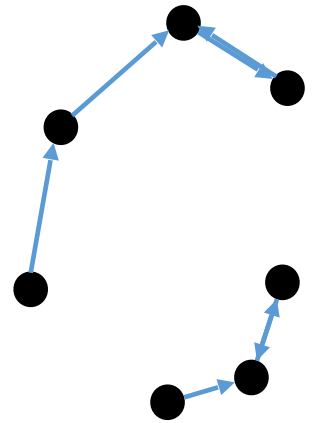
$$\forall q \in Q, \quad \arg\min_{r \in R} \|x_q - x_r\|$$



Nearest Neighbor?

Complexity?

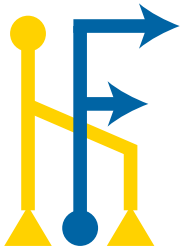
O(N²)



Nearest Neighbors

N-body Problems

- General representation $op_1, \ldots, op_m \;\; K\left(x_1, \ldots, x_m\right)$



N-body Problems

- General representation $op_1, \ldots, op_m \;\; K\left(x_1, \ldots, x_m\right)$
  - Kernel Density Estimation $\forall q \in Q, \quad \sum_{r \in R} K_\sigma \left(\frac{\|x_q - x_r\|}{\sigma} \right)$

N-body Problems

- General representation

$$op_1, \ldots, op_m \;\; K\left(x_1, \ldots, x_m\right)$$

  - Kernel Density Estimation

$$\forall q \in Q, \sum_{r \in R} K_\sigma\left(\frac{\|x_q - x_r\|}{\sigma}\right)$$

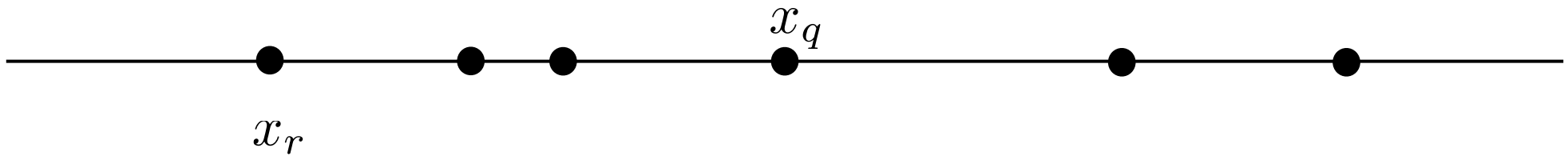
N-body Problems

- General representation

$$op_1, \ldots, op_m \;\; K\left(x_1, \ldots, x_m\right)$$

  - Kernel Density Estimation

$$\forall q \in Q, \sum_{r \in R} K_\sigma \left(\frac{\|x_q - x_r\|}{\sigma} \right)$$



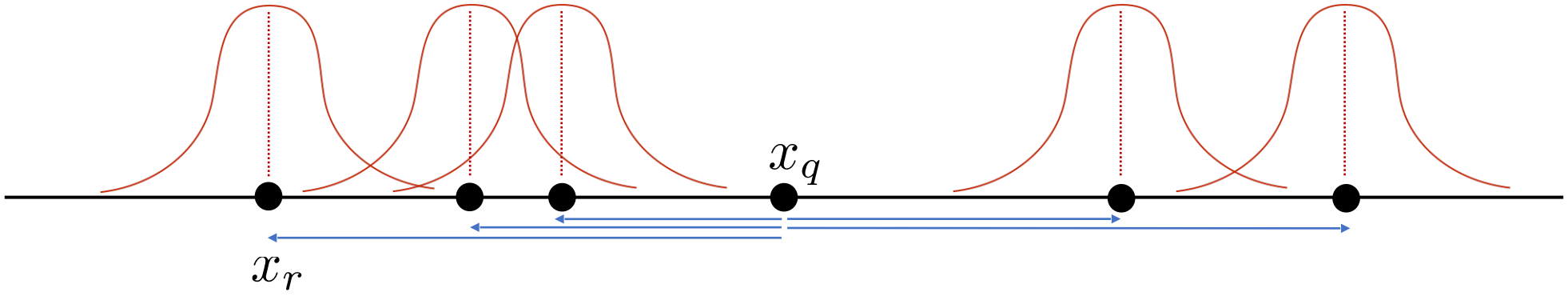
N-body Problems

- General representation

$$op_1, \ldots, op_m \quad K\left(x_1, \ldots, x_m\right)$$

  - Kernel Density Estimation

$$\forall q \in Q, \quad \sum_{r \in R} K_\sigma \left(\frac{\|x_q - x_r\|}{\sigma} \right)$$



N-body Problems

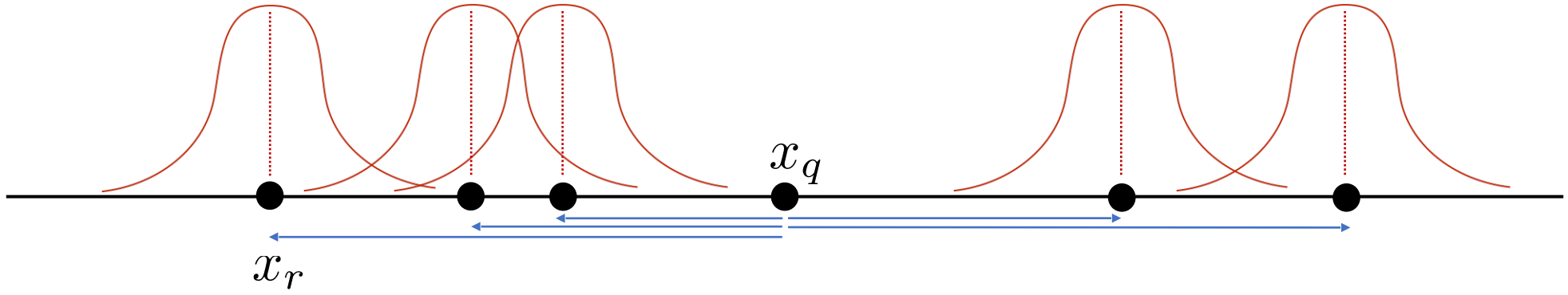
- General representation

$$op_1, \ldots, op_m \quad K(x_1, \ldots, x_m)$$

  - Kernel Density Estimation

$$\forall q \in Q, \quad \sum_{r \in R} K_\sigma \left(\frac{\|x_q - x_r\|}{\sigma} \right)$$

Complexity?



N-body Problems

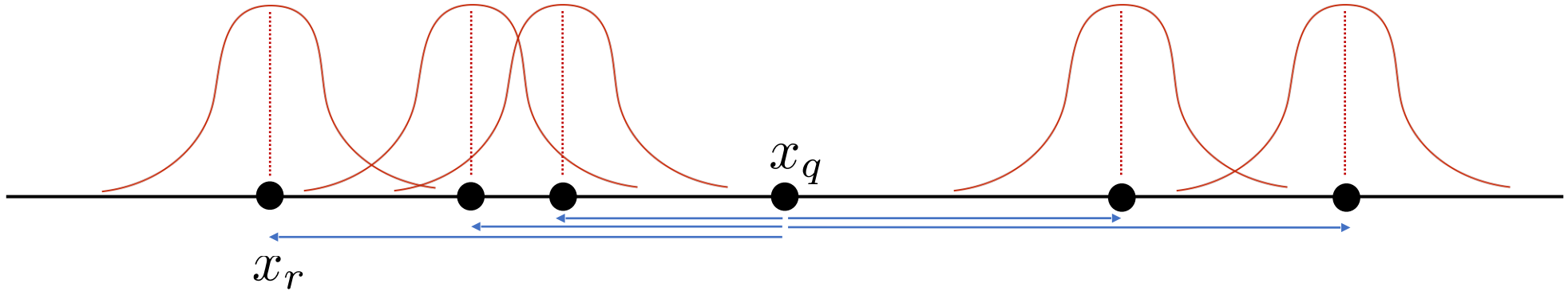
- General representation

$$op_1, \ldots, op_m \quad K(x_1, \ldots, x_m)$$

  - Kernel Density Estimation

$$\forall q \in Q, \quad \sum_{r \in R} K_\sigma \left(\frac{\|x_q - x_r\|}{\sigma} \right)$$

Complexity?



O(N²)

N-body Problems

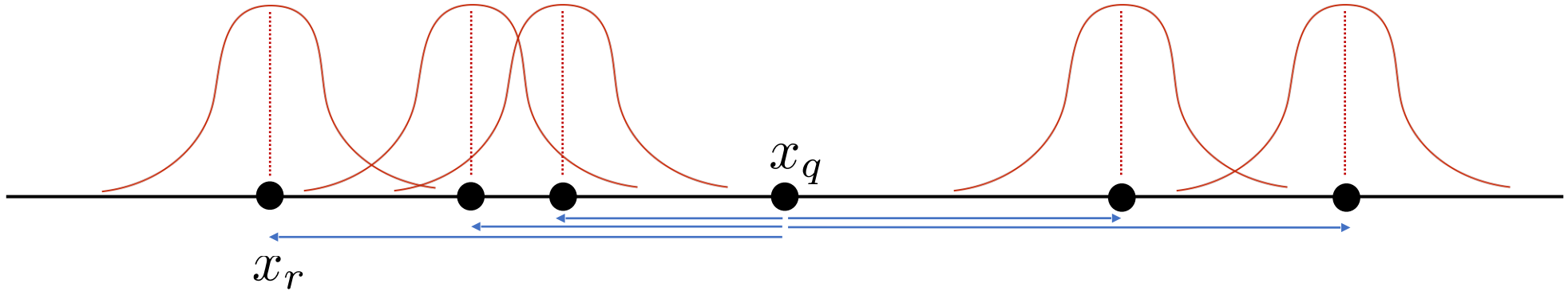
- General representation

$$op_1, \ldots, op_m \quad K(x_1, \ldots, x_m)$$

  - Kernel Density Estimation

$$\forall q \in Q, \quad \sum_{r \in R} K_\sigma \left(\frac{\|x_q - x_r\|}{\sigma} \right)$$

Complexity?



O(N²)

N-body Problems

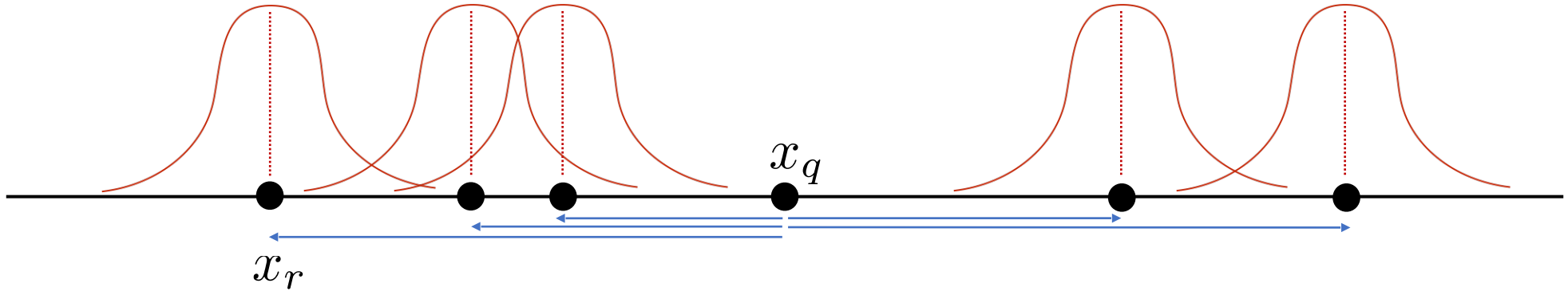
- General representation

$$op_1, \ldots, op_m \quad K(x_1, \ldots, x_m)$$

  - Kernel Density Estimation

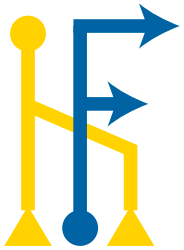
$$\forall q \in Q, \quad \sum_{r \in R} K_\sigma \left(\frac{\|x_q - x_r\|}{\sigma} \right)$$

Complexity?



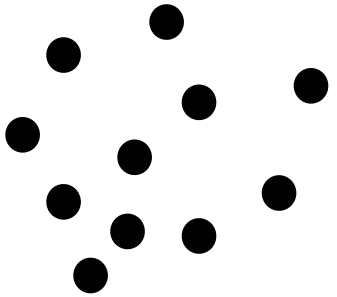
O(N²)

Complexity Reduction



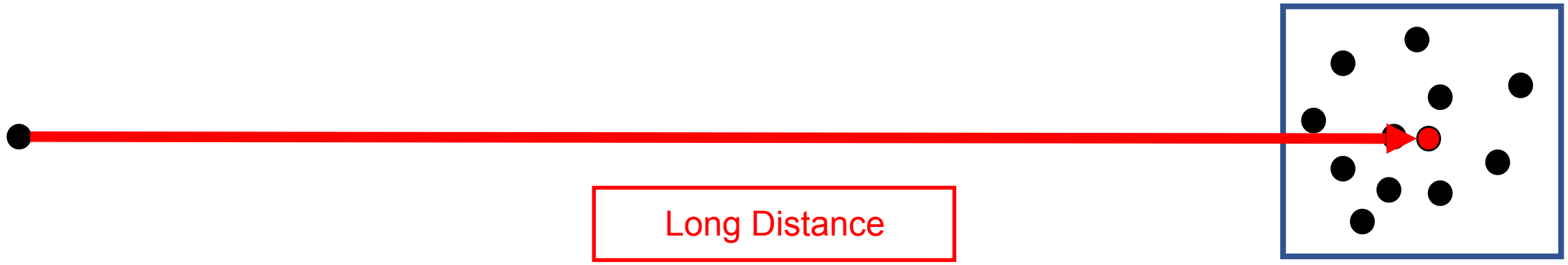
Complexity Reduction

- Barnes-Hut [Nature 1986] for force calculation



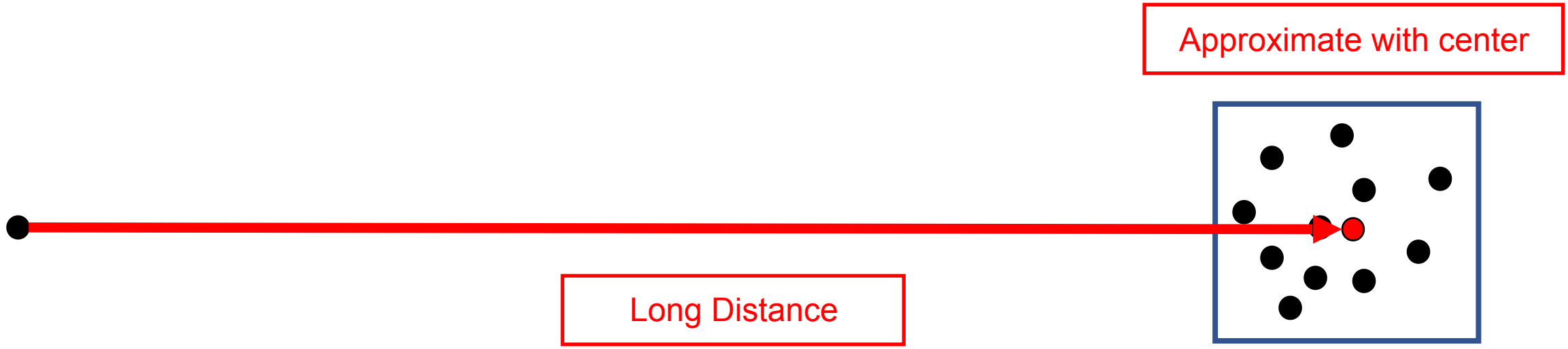
Complexity Reduction

- Barnes-Hut [Nature 1986] for force calculation



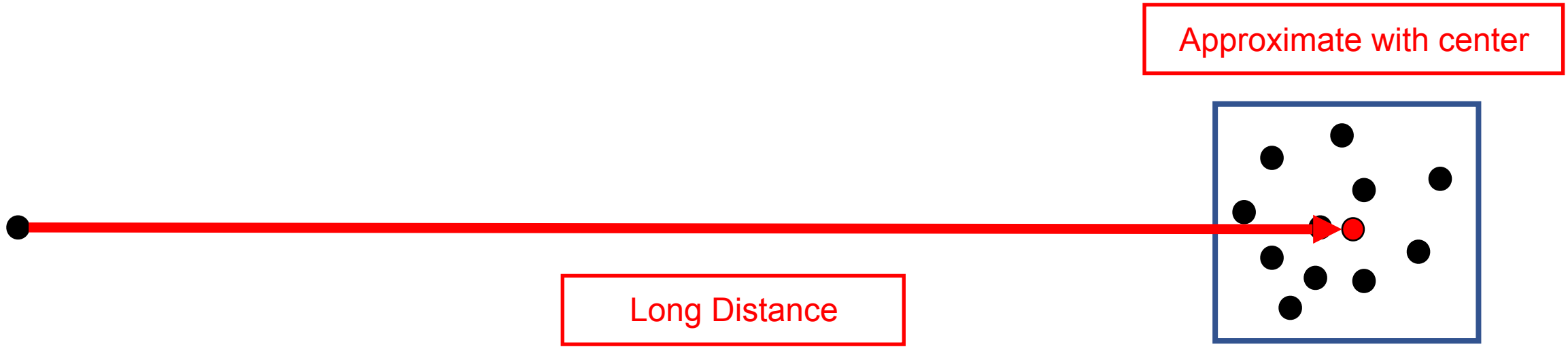
Complexity Reduction

- Barnes-Hut [Nature 1986] for force calculation



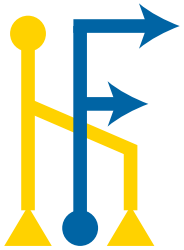
Complexity Reduction

- Barnes-Hut [Nature 1986] for force calculation



Asymptotically reduces the complexity: $O(N^2) \rightarrow O(N \log N)$

N-body Problems



N-body Problems

Problem	Operators	Kernel function
All Nearest Neighbor	$\forall, \arg\min$	$\|\|x_q - x_r\|$
All Range Search	$\forall, \cup \arg$	$I\left(h_{\min} < \|\|x_q - x_r\|\| < h_{\max}\right)$
All Range Count	$\forall, \sum$	$I\left(h_{\min} < \|\|x_q - x_r\|\| < h_{\max}\right)$
Naïve Bayes Classifier	$\forall, \arg\max$	$(1/\sqrt{2\pi\|\Sigma_k\|})e^{-\frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k)}P(C_k)$
Mixture Model E-step	$\forall, \forall$	$(1/\sqrt{2\pi\|\Sigma_k\|})e^{-\frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k)}$
K-means E-step	$\forall, \arg\min$	$\|\|x_q - x_r\|$
Mixture Model Log-likelihood	$\sum, \log\sum$	$(1/\sqrt{2\pi\|\Sigma_k\|})e^{-\frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k)}$
Kernel Density Estimation	$\forall, \sum$	$\phi(\frac{\|\|x_q - x_r\|\|}{h})$
Kernel Density Bayes Classifier	$\forall, \arg\max\sum$	$\phi(\frac{\|\|x_q - x_r\|\|}{h})P(C_k)$
2-Point Correlation	$\sum, \sum$	$I(\|\|x_q - x_r\|\| < h)$
Nadaraya-Watson Regression	$\forall, \sum$	$y_r\ \phi(\frac{\|\|x_q - x_r\|\|}{h})$
Thermodynamic Average	$\sum, \sum$	$\phi(\|\|x_q - x_r\|\|)$
Largest-Span Set	$\max, ..., \max$	$\sum(\|\|x_q - x_r\|\|)$
Closet Pair	$\max, ..., \max$	$\|\|x_q - x_r\|$
Minimum Spanning Tree	$\forall, \arg\min$	$\|\|x_q - x_r\|$
Coulombic Interaction	$\forall, \sum$	$\frac{\alpha_q\alpha_r}{\|\|x_q - x_r\|\|}$
Average Density	$\sum, \sum$	$I(\|\|x_q - x_r\|\| < h)$
Wave Function	$\forall, \prod$	$\phi(\|\|x_q - x_r\|\|)$
Hausdorff Distance	$\max, \min$	$\|\|x_q - x_r\|$
Intrinsic (Fractional) Dimension	$\sum, \sum$	$I(\|\|x_q - x_r\|\| < h)$

N-body Problems

Operators → decomposability property over the dataset

Problem	Operators	Kernel function
All Nearest Neighbor	$\forall, \arg\min$	$\|\|x_q - x_r\|$
All Range Search	$\forall, \cup \arg$	$I\left(h_{\min} < \|\|x_q - x_r\|\| < h_{\max}\right)$
All Range Count	$\forall, \sum$	$I\left(h_{\min} < \|\|x_q - x_r\|\| < h_{\max}\right)$
Naïve Bayes Classifier	$\forall, \arg\max$	$(1/\sqrt{2\pi\|\Sigma_k\|})e^{-\frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k)}P(C_k)$
Mixture Model E-step	$\forall, \forall$	$(1/\sqrt{2\pi\|\Sigma_k\|})e^{-\frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k)}$
K-means E-step	$\forall, \arg\min$	$\|\|x_q - x_r\|$
Mixture Model Log-likelihood	$\sum, \log\sum$	$(1/\sqrt{2\pi\|\Sigma_k\|})e^{-\frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k)}$
Kernel Density Estimation	$\forall, \sum$	$\phi(\frac{\|\|x_q - x_r\|\|}{h})$
Kernel Density Bayes Classifier	$\forall, \arg\max\sum$	$\phi(\frac{\|\|x_q - x_r\|\|}{h})P(C_k)$
2-Point Correlation	$\sum, \sum$	$I(\|\|x_q - x_r\|\| < h)$
Nadaraya-Watson Regression	$\forall, \sum$	$y_r\ \phi(\frac{\|\|x_q - x_r\|\|}{h})$
Thermodynamic Average	$\sum, \sum$	$\phi(\|\|x_q - x_r\|\|)$
Largest-Span Set	$\max, ..., \max$	$\sum(\|\|x_q - x_r\|\|)$
Closet Pair	$\max, ..., \max$	$\|\|x_q - x_r\|$
Minimum Spanning Tree	$\forall, \arg\min$	$\|\|x_q - x_r\|$
Coulombic Interaction	$\forall, \sum$	$\frac{\alpha_q\alpha_r}{\|\|x_q - x_r\|\|}$
Average Density	$\sum, \sum$	$I(\|\|x_q - x_r\|\| < h)$
Wave Function	$\forall, \prod$	$\phi(\|\|x_q - x_r\|\|)$
Hausdorff Distance	$\max, \min$	$\|\|x_q - x_r\|$
Intrinsic (Fractional) Dimension	$\sum, \sum$	$I(\|\|x_q - x_r\|\| < h)$

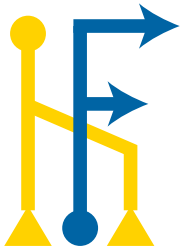
N-body Problems

Operators → decomposability property over the dataset

Problem	Operators	Kernel function
All Nearest Neighbor	$\forall, \arg\min$	$\|\|x_q - x_r\|$
All Range Search	$\forall, \cup \arg$	$I\left(h_{\min} < \|\|x_q - x_r\|\| < h_{\max}\right)$
All Range Count	$\forall, \sum$	$I\left(h_{\min} < \|\|x_q - x_r\|\| < h_{\max}\right)$
Naïve Bayes Classifier	$\forall, \arg\max$	$(1/\sqrt{2\pi\|\Sigma_k\|})e^{-\frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k)}P(C_k)$
Mixture Model E-step	$\forall, \forall$	$(1/\sqrt{2\pi\|\Sigma_k\|})e^{-\frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k)}$
K-means E-step	$\forall, \arg\min$	$\|\|x_q - x_r\|$
Mixture Model Log-likelihood	$\sum, \log\sum$	$(1/\sqrt{2\pi\|\Sigma_k\|})e^{-\frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k)}$
Kernel Density Estimation	$\forall, \sum$	$\phi(\frac{\|\|x_q - x_r\|\|}{h})$
Kernel Density Bayes Classifier	$\forall, \arg\max\sum$	$\phi(\frac{\|\|x_q - x_r\|\|}{h})P(C_k)$
2-Point Correlation	$\sum, \sum$	$I(\|\|x_q - x_r\|\| < h)$
Nadaraya-Watson Regression	$\forall, \sum$	$y_r\ \phi(\frac{\|\|x_q - x_r\|\|}{h})$
Thermodynamic Average	$\sum, \sum$	$\phi(\|\|x_q - x_r\|\|)$
Largest-Span Set	$\max, ..., \max$	$\sum(\|\|x_q - x_r\|\|)$
Closet Pair	$\max, ..., \max$	$\|\|x_q - x_r\|$
Minimum Spanning Tree	$\forall, \arg\min$	$\|\|x_q - x_r\|$
Coulombic Interaction	$\forall, \sum$	$\frac{\alpha_q\alpha_r}{\|\|x_q - x_r\|\|}$
Average Density	$\sum, \sum$	$I(\|\|x_q - x_r\|\| < h)$
Wave Function	$\forall, \prod$	$\phi(\|\|x_q - x_r\|\|)$
Hausdorff Distance	$\max, \min$	$\|\|x_q - x_r\|$
Intrinsic (Fractional) Dimension	$\sum, \sum$	$I(\|\|x_q - x_r\|\| < h)$

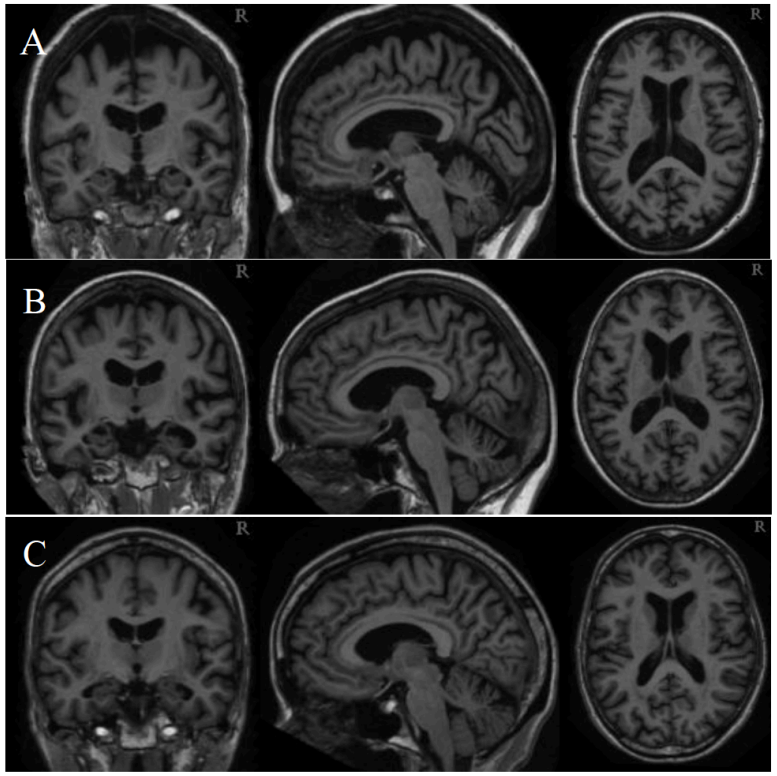
Kernel function → decrease monotonically with distance

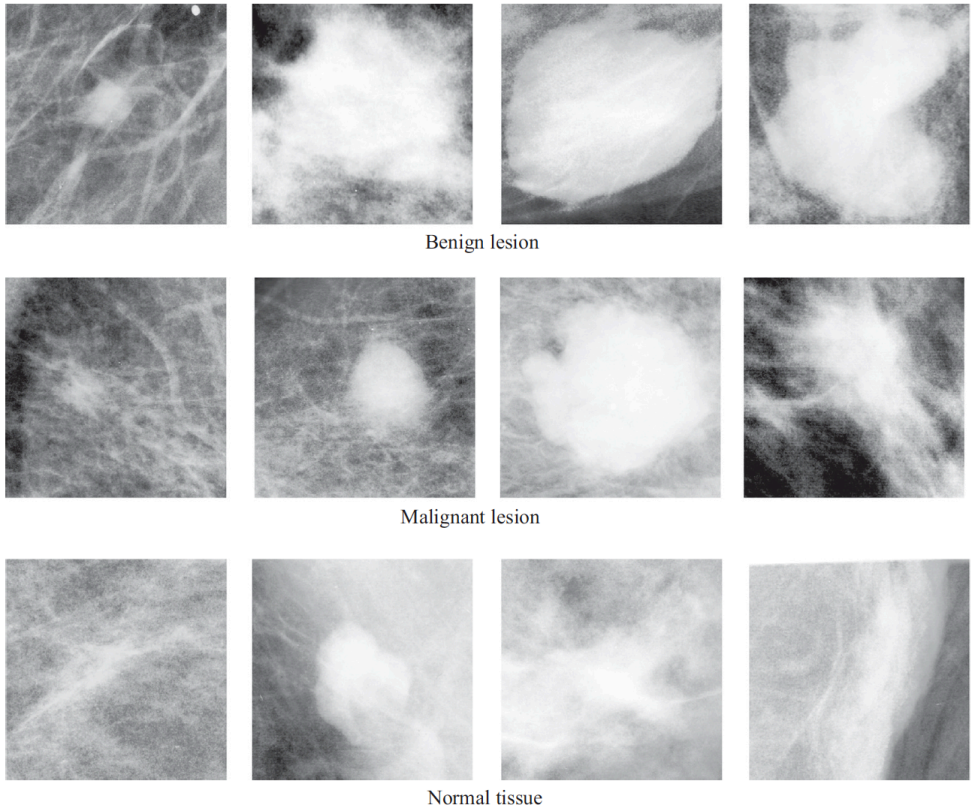
Why N-body Problems?



Why N-body Problems?

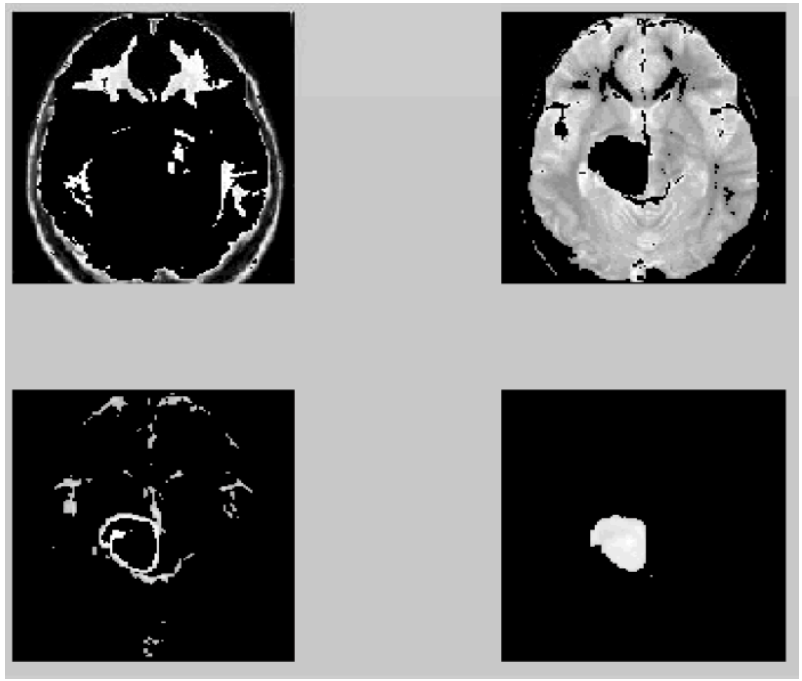
- K-Nearest Neighbors → Alzheimer disease [IJISAE 2016] & breast cancer diagnosis [CBM 2015]





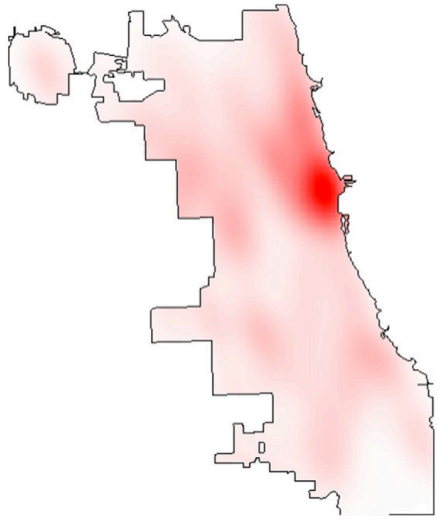
Why N-body Problems?

- K-Nearest Neighbors → Alzheimer disease [IJISAE 2016] & breast cancer diagnosis [CBM 2015]
- K-Means → detecting brain tumors [ICCSP 2013]



Why N-body Problems?

- K-Nearest Neighbors → Alzheimer disease [IJISAE 2016] & breast cancer diagnosis [CBM 2015]
- K-Means → detecting brain tumors [ICCSP 2013]
- Kernel Density Estimation → predicting crime [DSS 2015]



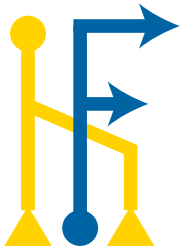
Why N-body Problems?

- K-Nearest Neighbors → Alzheimer disease [IJISAE 2016] & breast cancer diagnosis [CBM 2015]
- K-Means → detecting brain tumors [ICCSP 2013]
- Kernel Density Estimation → predicting crime [DSS 2015]
- K-Nearest Neighbor, Expectation Maximization, K-Means, and Naïve Bayes → in **top 10 algorithms** for data mining research [CRC 2009]

Why N-body Problems?

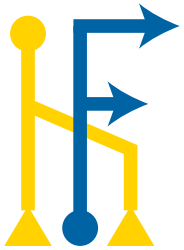
- K-Nearest Neighbors → Alzheimer disease [IJISAE 2016] & breast cancer diagnosis [CBM 2015]
- K-Means → detecting brain tumors [ICCSP 2013]
- Kernel Density Estimation → predicting crime [DSS 2015]
- K-Nearest Neighbor, Expectation Maximization, K-Means, and Naïve Bayes → in **top 10 algorithms** for data mining research [CRC 2009]
- Applications: machine learning, computer vision, computational geometry, database, scientific computing, etc.

Related Work



Related Work

- N-body algorithms
  - Physics
    - Barnes-Hut → O(N log N), FMM → O(N)
  - Machine Learning
    - MLPACK [JMLR 2014]
    - Weka [SIGKDD 2009]
    - Scikit-learn [JMLR 2011]



Related Work

- N-body algorithms
  - Physics
    - Barnes-Hut → O(N log N), FMM → O(N)
  - Machine Learning
    - MLPACK [JMLR 2014]
    - Weka [SIGKDD 2009]
    - Scikit-learn [JMLR 2011]
- Domain-specific language (DSL) and compilers
  - Embedded
    - OptiML → machine learning
    - DeepDSL → deep learning
    - Halide → image processing
  - Stand-alone
    - SCOPE → data analysis

Related Work

- N-body algorithms
  - Physics
    - Barnes-Hut → O(N log N), FMM → O(N)
  - Machine Learning
    - MLPACK [JMLR 2014]
    - Weka [SIGKDD 2009]
    - Scikit-learn [JMLR 2011]
- Domain-specific language (DSL) and compilers
  - Embedded
    - OptiML → machine learning
    - DeepDSL → deep learning
    - Halide → image processing
  - Stand-alone
    - SCOPE → data analysis

- Lack efficient optimal algorithms
- Lack parallelism and scalability

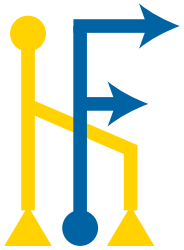
Related Work

- N-body algorithms
  - Physics
    - Barnes-Hut → O(N log N), FMM → O(N)
  - Machine Learning
    - MLPACK [JMLR 2014]
    - Weka [SIGKDD 2009]
    - Scikit-learn [JMLR 2011]
- Domain-specific language (DSL) and compilers
  - Embedded
    - OptiML → machine learning
    - DeepDSL → deep learning
    - Halide → image processing
  - Stand-alone
    - SCOPE → data analysis

- Lack efficient optimal algorithms
- Lack parallelism and scalability

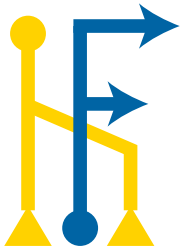
DSLs provide terse and extensible programs while achieving state-of-the-art performance

Portal



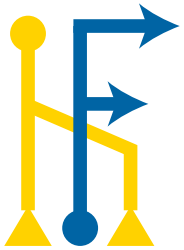
Portal

- Optimal, parallel, and scalable solution for N-body problems



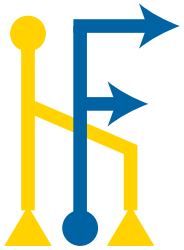
Portal

- Optimal, parallel, and scalable solution for N-body problems
- Embedded language in C++



Portal

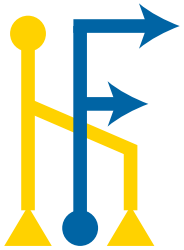
- Optimal, parallel, and scalable solution for N-body problems
- Embedded language in C++
- Built on top of the *algorithmic framework PASCAL* [Euro-Par 2017]



Portal

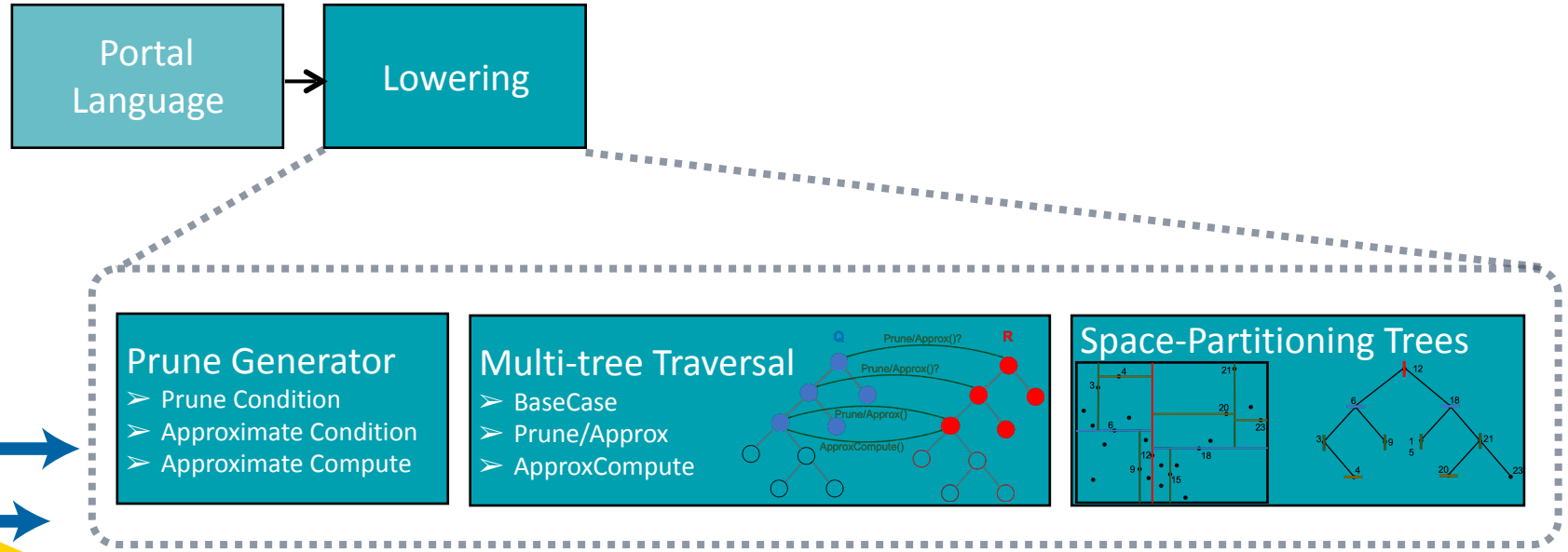
- Optimal, parallel, and scalable solution for N-body problems
- Embedded language in C++
- Built on top of the *algorithmic framework PASCAL* [Euro-Par 2017]

Portal Language



Portal

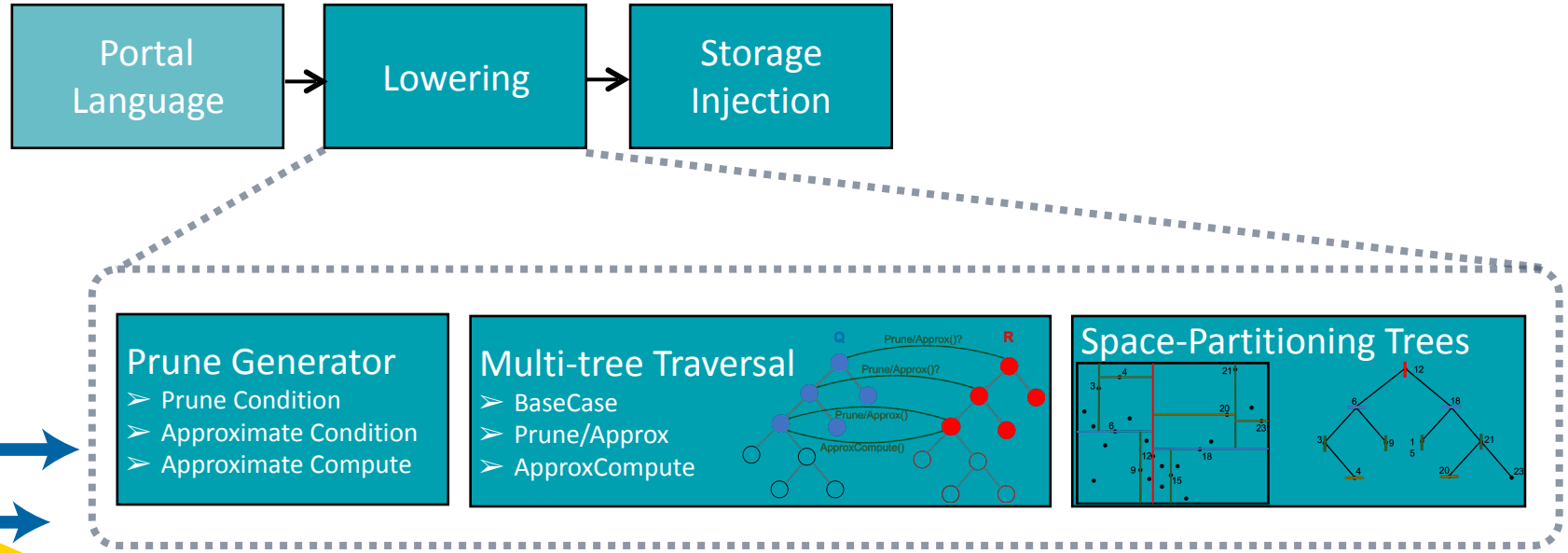
- Optimal, parallel, and scalable solution for N-body problems
- Embedded language in C++
- Built on top of the *algorithmic framework PASCAL* [Euro-Par 2017]



Extended from PASCAL

Portal

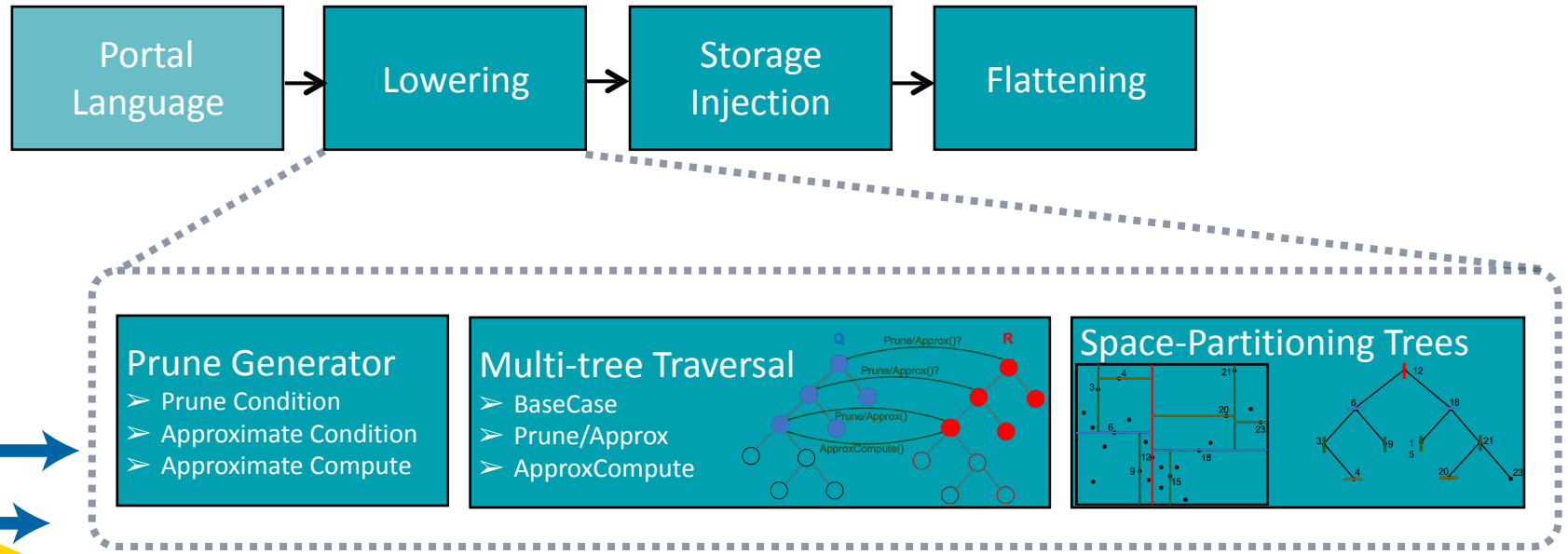
- Optimal, parallel, and scalable solution for N-body problems
- Embedded language in C++
- Built on top of the *algorithmic framework PASCAL* [Euro-Par 2017]



Extended from PASCAL

Portal

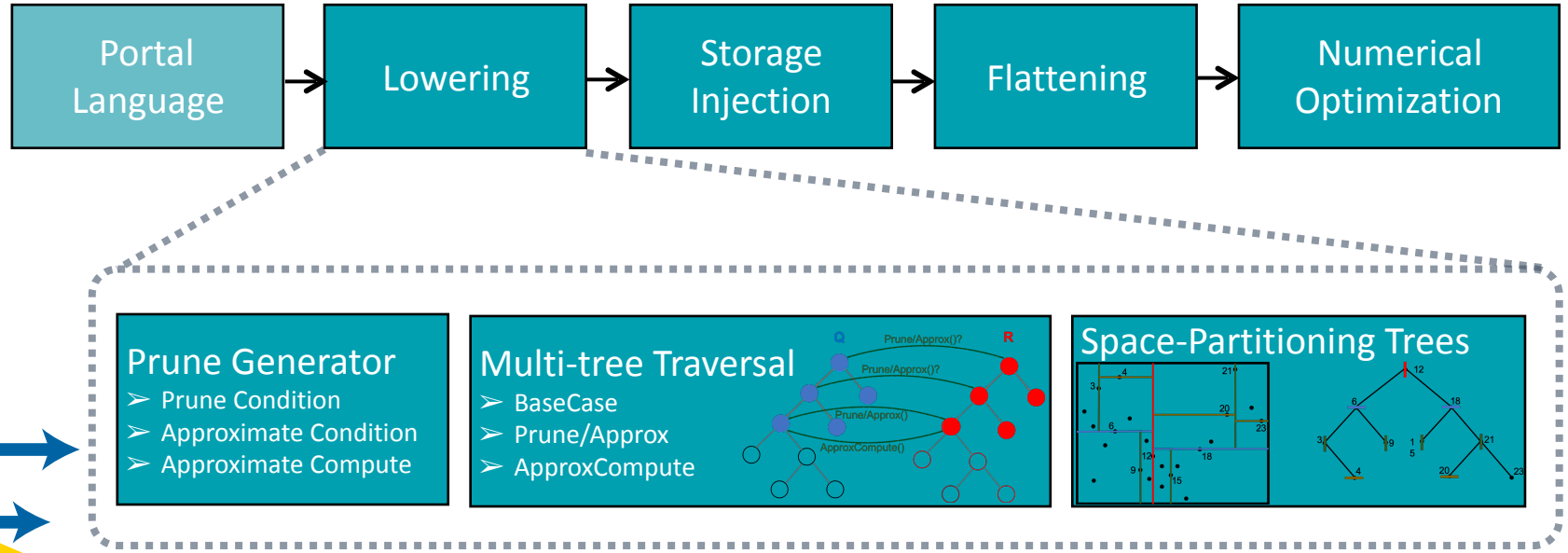
- Optimal, parallel, and scalable solution for N-body problems
- Embedded language in C++
- Built on top of the *algorithmic framework PASCAL* [Euro-Par 2017]



Extended from PASCAL

Portal

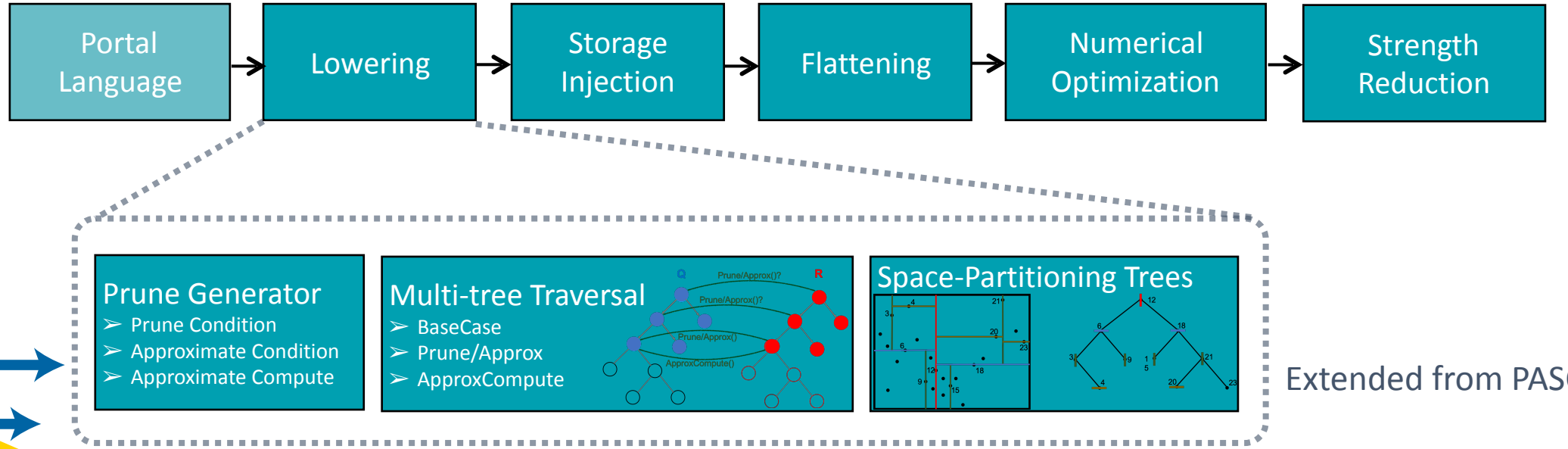
- Optimal, parallel, and scalable solution for N-body problems
- Embedded language in C++
- Built on top of the *algorithmic framework PASCAL* [Euro-Par 2017]



Extended from PASCAL

Portal

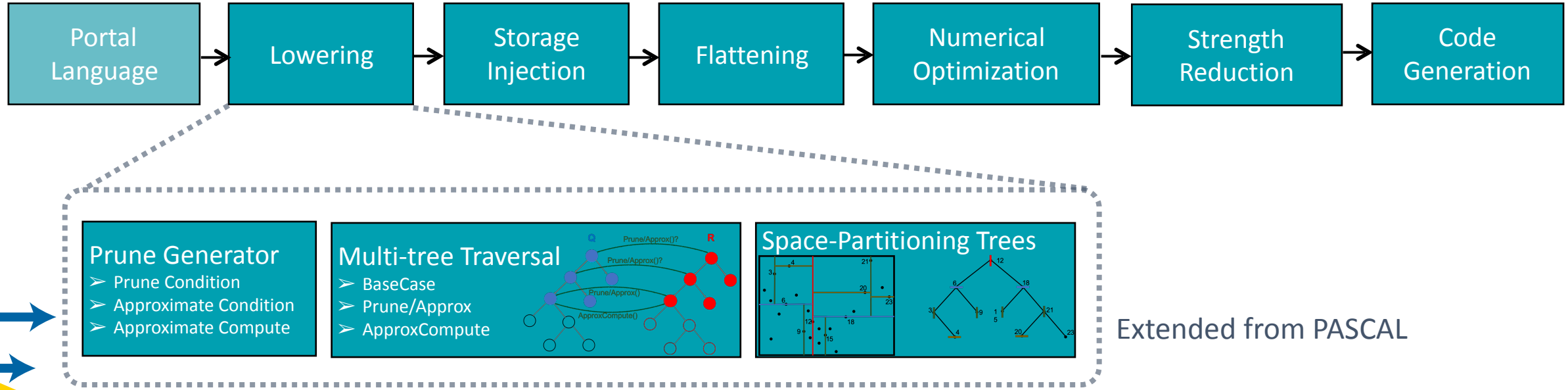
- Optimal, parallel, and scalable solution for N-body problems
- Embedded language in C++
- Built on top of the *algorithmic framework PASCAL* [Euro-Par 2017]



Extended from PASCAL

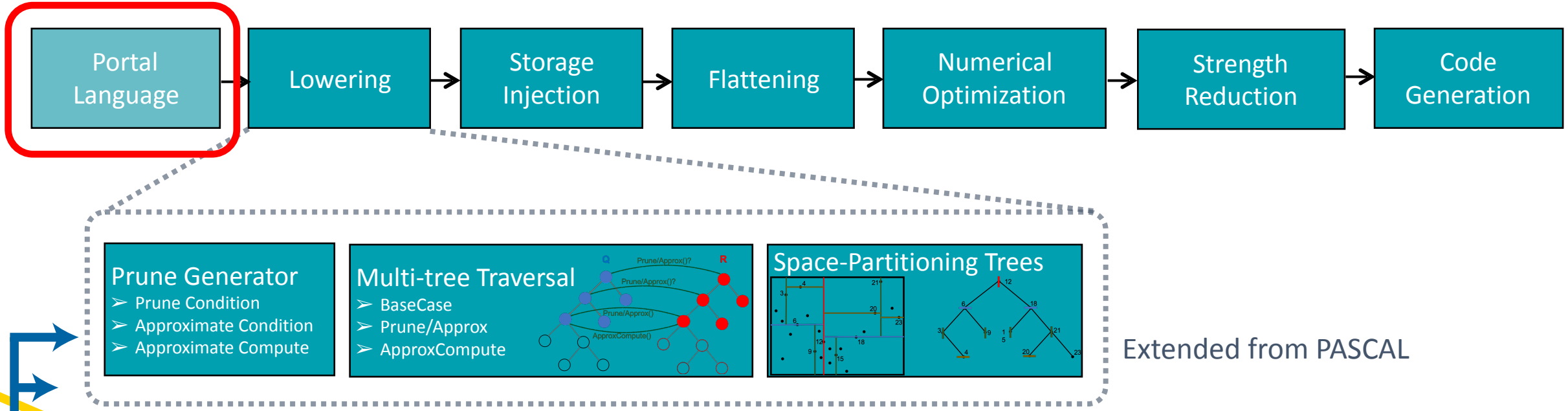
Portal

- Optimal, parallel, and scalable solution for N-body problems
- Embedded language in C++
- Built on top of the *algorithmic framework PASCAL* [Euro-Par 2017]

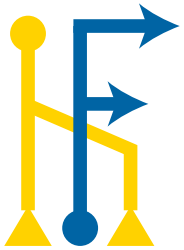


Portal

- Optimal, parallel, and scalable solution for N-body problems
- Embedded language in C++
- Built on top of the *algorithmic framework PASCAL* [Euro-Par 2017]



Portal DSL



Portal DSL

- Inspired by mathematical formulation

$$op_1, \ldots, op_m \;\; K\left(x_1, \ldots, x_m\right)$$ [Applied to m datasets (D_1, …, D_m)]

- Build up an N-body problem by chaining Layers

Portal DSL

- Inspired by mathematical formulation

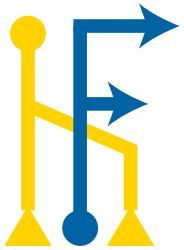
$$op_1, \ldots, op_m \;\; K\left(x_1, \ldots, x_m\right)$$ [Applied to m datasets (D_1, …, D_m)]

- Build up an N-body problem by chaining Layers
- Each Layer includes:
  - Operator
  - Dataset
  - Kernel/Modifying function

Portal DSL

- Example: Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \quad \|x_q - x_r\|$$



Portal DSL

- Example: Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \quad \|x_q - x_r\|$$

Layer1
Operator: $\forall$
Dataset: Q
Kernel: --

Portal DSL

- Example: Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \|x_q - x_r\|$$

Layer1
Operator: $\forall$
Dataset: Q
Kernel: --

Layer2
Operator: $\arg\min$
Dataset: R
Kernel: $\|x_q - x_r\|$

Portal DSL

- Example: Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \quad \|x_q - x_r\|$$

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, query);
expr.addLayer(PortalOp::ARGMIN, reference, PortalFunc::EUCLIDEAN);
expr.execute();
Storage output = expr.getOutput();
```

Portal DSL

- Example: Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \quad \|x_q - x_r\|$$

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, query);
expr.addLayer(PortalOp::ARGMIN, reference, PortalFunc::EUCLIDEAN);
expr.execute();
Storage output = expr.getOutput();
```

Defining storage for each data set

Portal DSL

- Example: Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \quad \|x_q - x_r\|$$

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, query);
expr.addLayer(PortalOp::ARGMIN, reference, PortalFunc::EUCLIDEAN);
expr.execute();
Storage output = expr.getOutput();
```

Portal DSL

- Example: Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \quad \|x_q - x_r\|$$

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, query);
expr.addLayer(PortalOp::ARGMIN, reference, PortalFunc::EUCLIDEAN);
expr.execute();
Storage output = expr.getOutput();
```

Defining the N-body structure

Portal DSL

- Example: Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \quad \|x_q - x_r\|$$

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, query);
expr.addLayer(PortalOp::ARGMIN, reference, PortalFunc::EUCLIDEAN);
expr.execute();
Storage output = expr.getOutput();
```

Portal DSL

- Example: Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \quad \|x_q - x_r\|$$

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, query);
expr.addLayer(PortalOp::ARGMIN, reference, PortalFunc::EUCLIDEAN);
expr.execute();
Storage output = expr.getOutput();
```

Adding Layer 1 for NN

Portal DSL

- Example: Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \quad \|x_q - x_r\|$$

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, query);
expr.addLayer(PortalOp::ARGMIN, reference, PortalFunc::EUCLIDEAN);
expr.execute();
Storage output = expr.getOutput();
```

Portal DSL

- Example: Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \quad \|x_q - x_r\|$$

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, query);
expr.addLayer(PortalOp::ARGMIN, reference, PortalFunc::EUCLIDEAN);
expr.execute();
Storage output = expr.getOutput();
```

Adding Layer 2 for NN

Portal DSL

- Example: Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \quad \|x_q - x_r\|$$

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, query);
expr.addLayer(PortalOp::ARGMIN, reference, PortalFunc::EUCLIDEAN);
expr.execute();
Storage output = expr.getOutput();
```

Portal DSL

- Example: Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \quad \|x_q - x_r\|$$

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, query);
expr.addLayer(PortalOp::ARGMIN, reference, PortalFunc::EUCLIDEAN);
expr.execute();
Storage output = expr.getOutput();
```

Executing the N-body computation

Portal DSL

- Example: Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \quad \|x_q - x_r\|$$

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, query);
expr.addLayer(PortalOp::ARGMIN, reference, PortalFunc::EUCLIDEAN);
expr.execute();
Storage output = expr.getOutput();
```

Portal DSL

- Example: Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \quad \|x_q - x_r\|$$

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, query);
expr.addLayer(PortalOp::ARGMIN, reference, PortalFunc::EUCLIDEAN);
expr.execute();
Storage output = expr.getOutput();
```

Returning the output

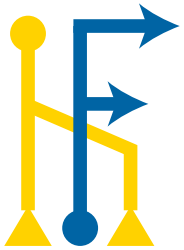
Portal DSL

- Example: Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \quad \|x_q - x_r\|$$

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, query);
expr.addLayer(PortalOp::ARGMIN, reference, PortalFunc::EUCLIDEAN);
expr.execute();
Storage output = expr.getOutput();
```

Portal DSL



Portal DSL

- Portal operators
  - Single variable reduction
  - Multi variable reduction
  - All

Category	Math Operator	Portal Operator
Single	$\sum$	SUM
Single	$\prod$	PROD
Single	$\texttt{argmin}$	ARGMIN
Single	$\texttt{argmax}$	ARGMAX
Single	$\texttt{min}$	MIN
Single	$\texttt{max}$	MAX
Multi	$\bigcup$	UNION
Multi	$\bigcup \texttt{arg}$	UNIONARG
Multi	$\texttt{argmin}^k$	KARGMIN
Multi	$\texttt{argmax}^k$	KARGMAX
Multi	$\texttt{min}^k$	KMIN
Multi	$\texttt{max}^k$	KMAX
All	$\forall$	FORALL

Portal DSL

- Portal operators
  - Single variable reduction
  - Multi variable reduction
  - All
- Storage
  - From file
  - From C++ data-structure

Category	Math Operator	Portal Operator
Single	$\sum$	SUM
Single	$\prod$	PROD
Single	$\texttt{argmin}$	ARGMIN
Single	$\texttt{argmax}$	ARGMAX
Single	$\texttt{min}$	MIN
Single	$\texttt{max}$	MAX
Multi	$\bigcup$	UNION
Multi	$\bigcup \texttt{arg}$	UNIONARG
Multi	$\texttt{argmin}^k$	KARGMIN
Multi	$\texttt{argmax}^k$	KARGMAX
Multi	$\texttt{min}^k$	KMIN
Multi	$\texttt{max}^k$	KMAX
All	$\forall$	FORALL

Portal DSL

- Portal operators
  - Single variable reduction
  - Multi variable reduction
  - All
- Storage
  - From file
  - From C++ data-structure

Category	Math Operator	Portal Operator
Single	$\sum$	SUM
Single	$\prod$	PROD
Single	$\mathtt{argmin}$	ARGMIN
Single	$\mathtt{argmax}$	ARGMAX
Single	$\mathtt{min}$	MIN
Single	$\mathtt{max}$	MAX
Multi	$\bigcup$	UNION
Multi	$\bigcup \mathtt{arg}$	UNIONARG
Multi	$\mathtt{argmin}^k$	KARGMIN
Multi	$\mathtt{argmax}^k$	KARGMAX
Multi	$\mathtt{min}^k$	KMIN
Multi	$\mathtt{max}^k$	KMAX
All	$\forall$	FORALL

Portal DSL

- Portal operators
  - Single variable reduction
  - Multi variable reduction
  - All
- Storage
  - From file
  - From C++ data-structure

Category	Math Operator	Portal Operator
Single	$\sum$	SUM
Single	$\prod$	PROD
Single	$\texttt{argmin}$	ARGMIN
Single	$\texttt{argmax}$	ARGMAX
Single	$\texttt{min}$	MIN
Single	$\texttt{max}$	MAX
Multi	$\bigcup$	UNION
Multi	$\bigcup \texttt{arg}$	UNIONARG
Multi	$\texttt{argmin}^k$	KARGMIN
Multi	$\texttt{argmax}^k$	KARGMAX
Multi	$\texttt{min}^k$	KMIN
Multi	$\texttt{max}^k$	KMAX
All	$\forall$	FORALL

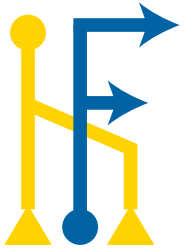
Portal DSL

- Portal operators
  - Single variable reduction
  - Multi variable reduction
  - All
- Storage
  - From file

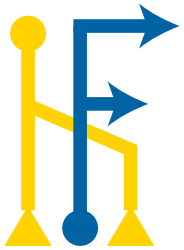
```
// Construct Storage CVS file
Storage file("query_file.csv");
```

  - From C++ data-structure

```
// Construct Storage from C++ data-structure
std::vector<std::vector<float>> input;
// Fill in the input data structure with values
Storage query(input);
```



Portal DSL



Portal DSL

- Kernel function
  - Pre-defined

```
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::MANHATTAN));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::CHEBYSHEV));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::MAHALANOBIS));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::SQREUCDIST));
```

  - User-defined (nearest neighbor example)

Portal DSL

- Kernel function
  - Pre-defined

```
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::MANHATTAN));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::CHEBYSHEV));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::MAHALANOBIS));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::SQREUCDIST));
```

Name of the predefined kernel function

  - User-defined (nearest neighbor example)

Portal DSL

- Kernel function
  - Pre-defined

```
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::MANHATTAN));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::CHEBYSHEV));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::MAHALANOBIS));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::SQREUCDIST));
```

  - User-defined (nearest neighbor example)

Portal DSL

- Kernel function
  - Pre-defined

```
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::MANHATTAN));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::CHEBYSHEV));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::MAHALANOBIS));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::SQREUCDIST));
```

  - User-defined (nearest neighbor example)

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
Var q,r;
Expr EuclidDist = sqrt(pow((q-r), 2));
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, q, query);
expr.addLayer(PortalOp::ARGMIN, r, reference, EuclidDist);
expr.execute();
Storage output = expr.getOutput();
```

Portal DSL

- Kernel function
  - Pre-defined

```
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::MANHATTAN));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::CHEBYSHEV));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::MAHALANOBIS));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::SQREUCDIST));
```

  - User-defined (nearest neighbor example)

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
Var q,r;
Expr EuclidDist = sqrt(pow((q-r), 2));
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, q, query);
expr.addLayer(PortalOp::ARGMIN, r, reference, EuclidDist);
expr.execute();
Storage output = expr.getOutput();
```

Keyword for defining variables

Portal DSL

- Kernel function
  - Pre-defined

```
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::MANHATTAN));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::CHEBYSHEV));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::MAHALANOBIS));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::SQREUCDIST));
```

  - User-defined (nearest neighbor example)

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
Var q,r;
Expr EuclidDist = sqrt(pow((q-r), 2));
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, q, query);
expr.addLayer(PortalOp::ARGMIN, r, reference, EuclidDist);
expr.execute();
Storage output = expr.getOutput();
```

Portal DSL

- Kernel function
  - Pre-defined

```
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::MANHATTAN));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::CHEBYSHEV));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::MAHALANOBIS));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::SQREUCDIST));
```

  - User-defined (nearest neighbor example)

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
Var q,r;
Expr EuclidDist = sqrt(pow((q-r), 2));
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, q, query);
expr.addLayer(PortalOp::ARGMIN, r, reference, EuclidDist);
expr.execute();
Storage output = expr.getOutput();
```

User-defined kernel function

Portal DSL

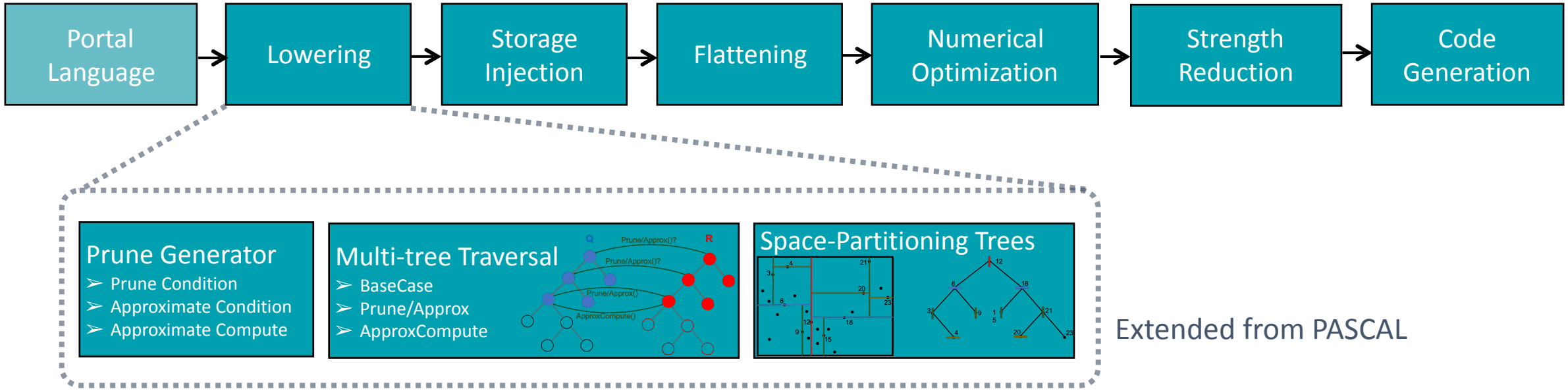
- Kernel function
  - Pre-defined

```
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::MANHATTAN));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::CHEBYSHEV));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::MAHALANOBIS));
expr.addLayer(PortalOp::FORALL, ref, PortalFunc::SQREUCDIST));
```

  - User-defined (nearest neighbor example)

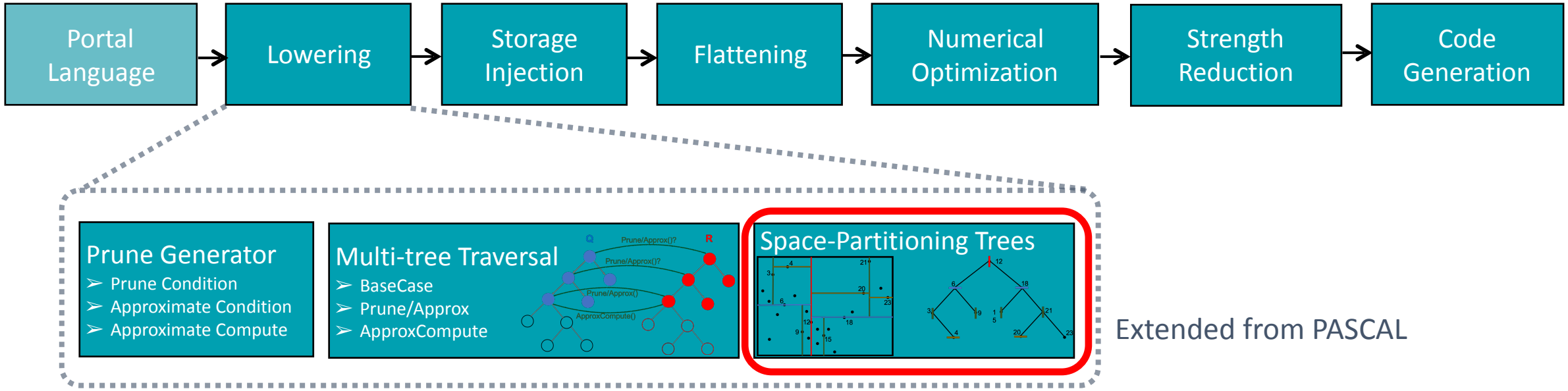
```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
Var q,r;
Expr EuclidDist = sqrt(pow((q-r), 2));
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, q, query);
expr.addLayer(PortalOp::ARGMIN, r, reference, EuclidDist);
expr.execute();
Storage output = expr.getOutput();
```

Portal



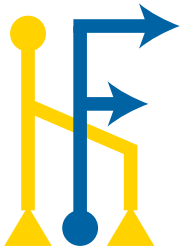
Extended from PASCAL

Portal



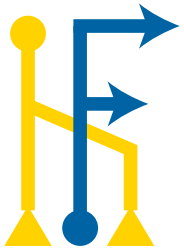
Extended from PASCAL

Space-partitioning Tree



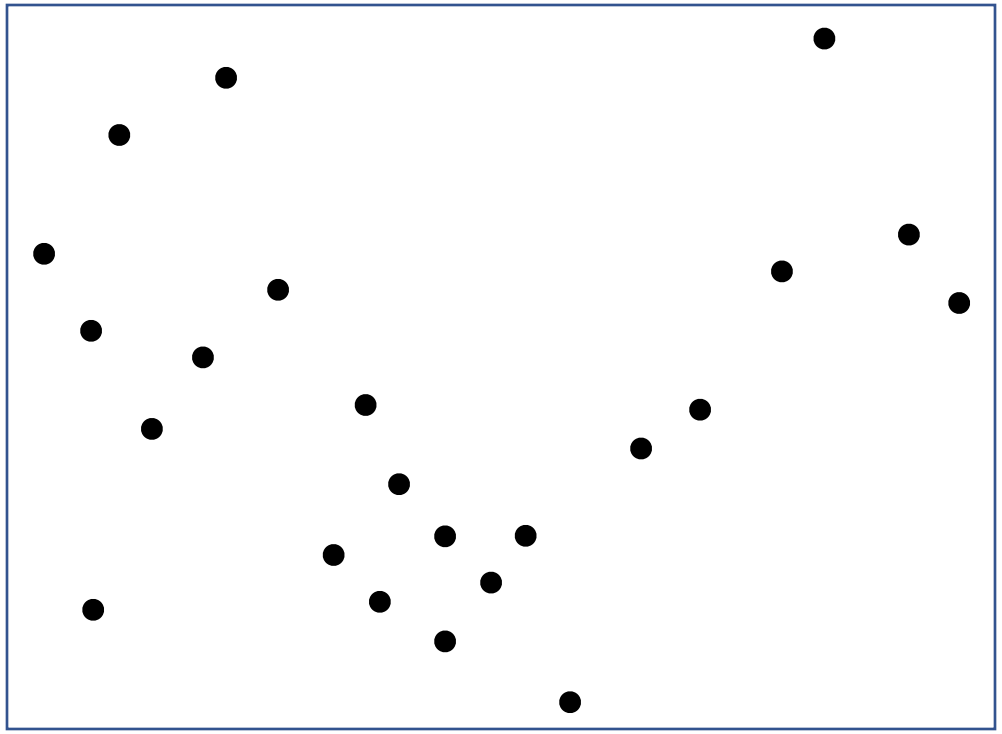
Space-partitioning Tree

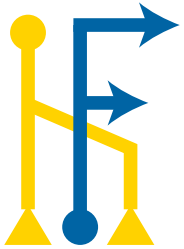
- K-d tree



Space-partitioning Tree

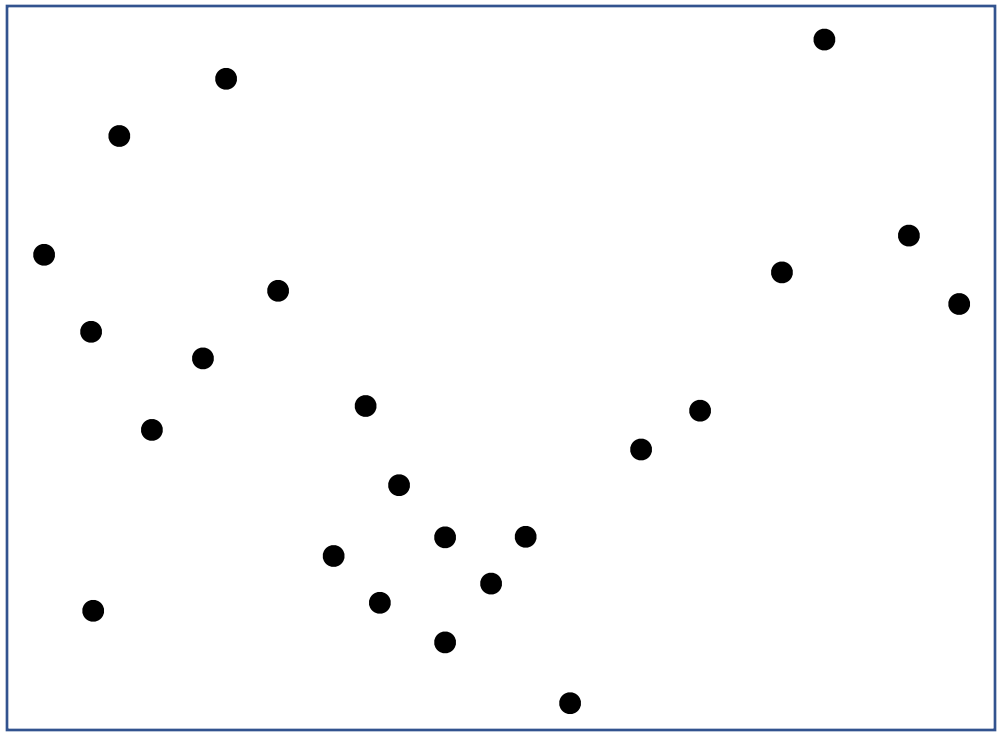
- K-d tree

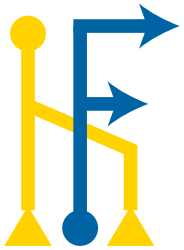




Space-partitioning Tree

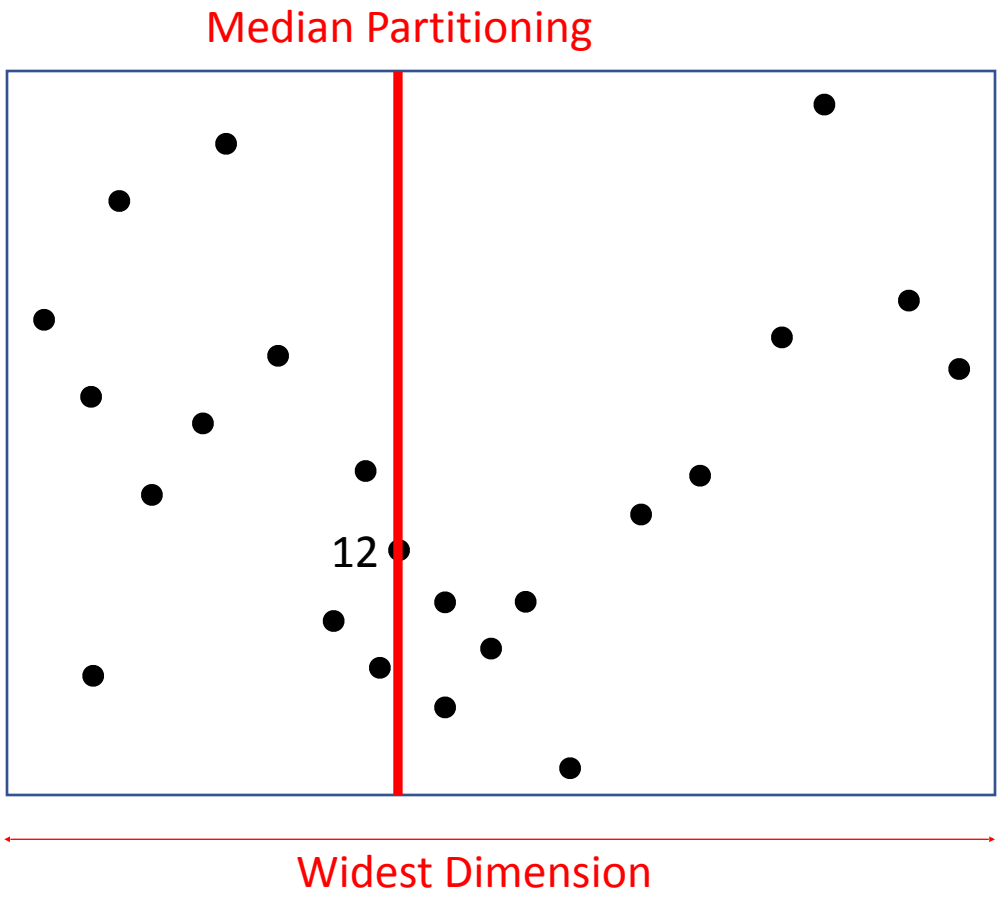
- K-d tree

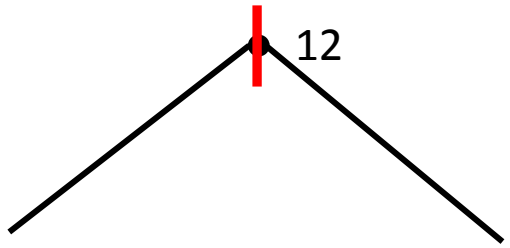




Space-partitioning Tree

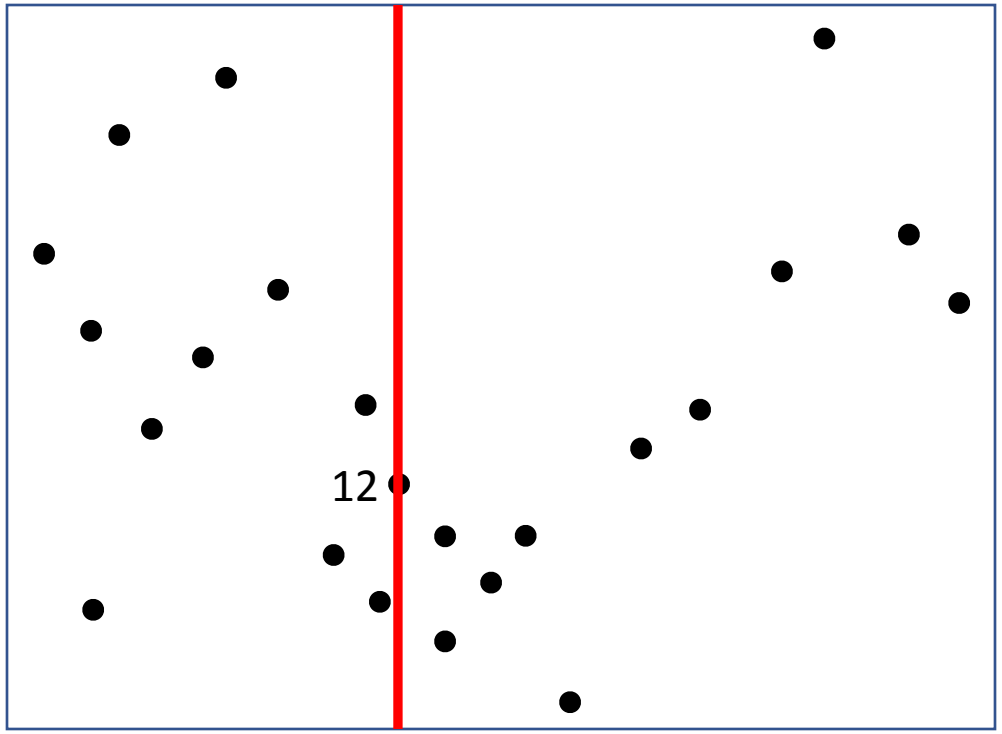
- K-d tree

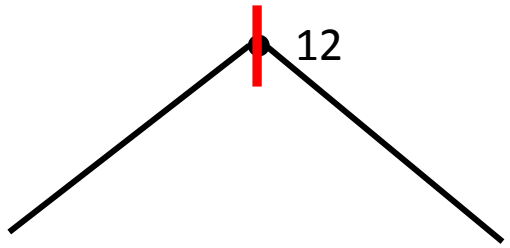


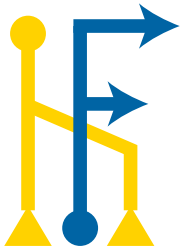


Space-partitioning Tree

- K-d tree

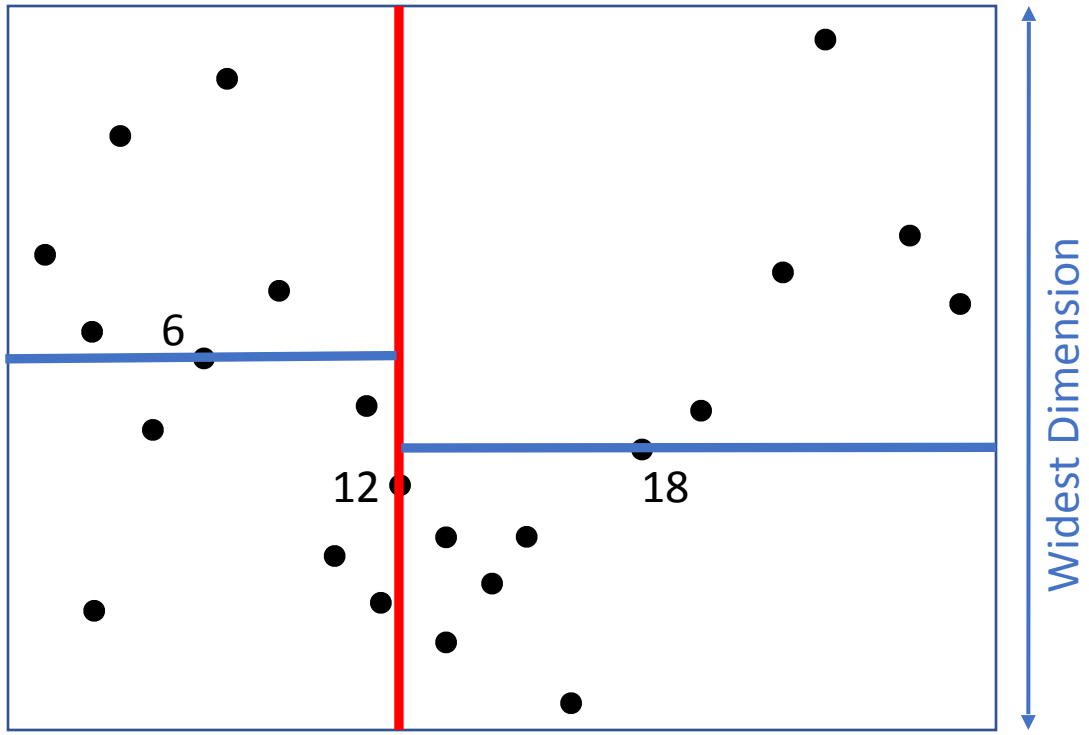


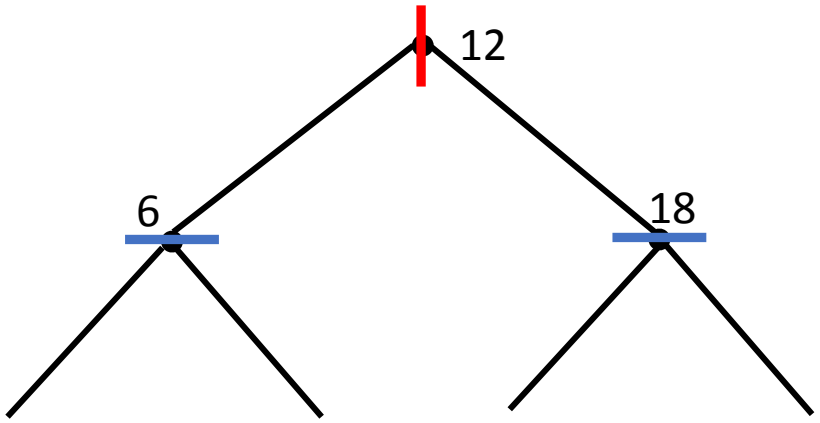




Space-partitioning Tree

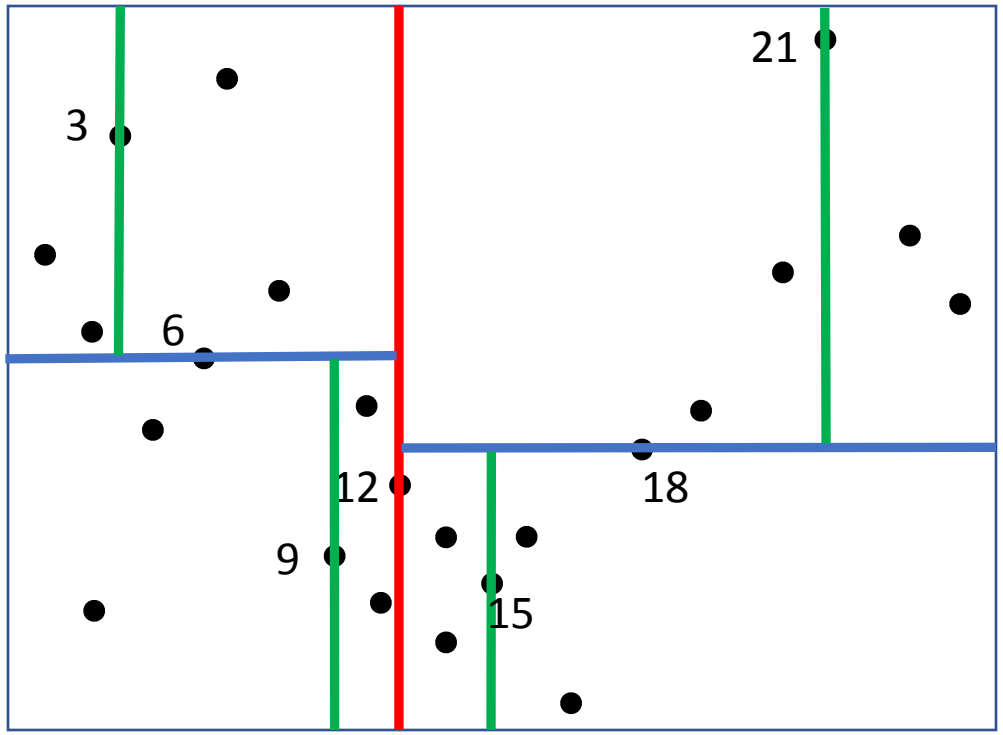
- K-d tree

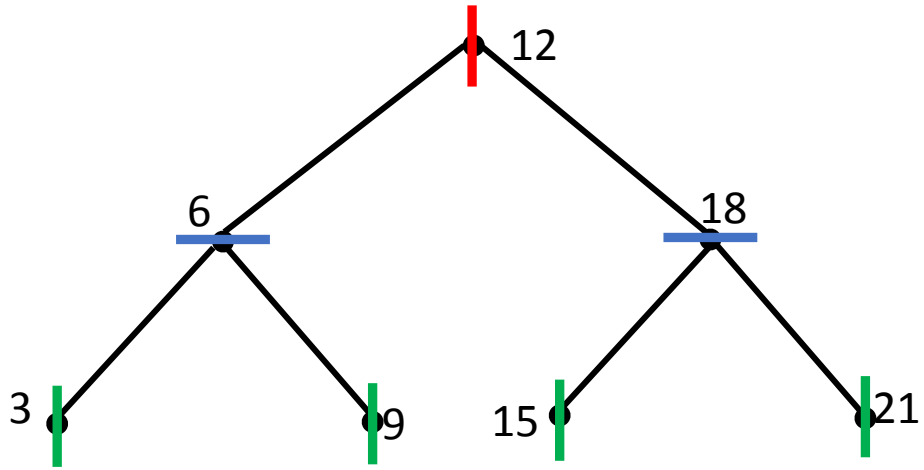




Space-partitioning Tree

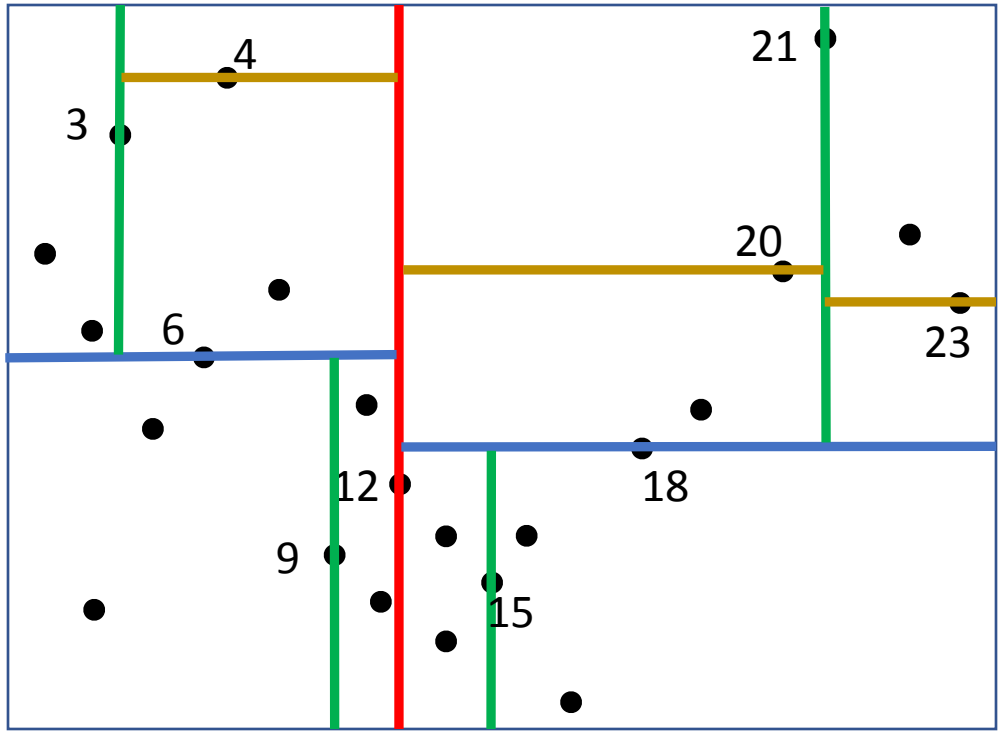
- K-d tree

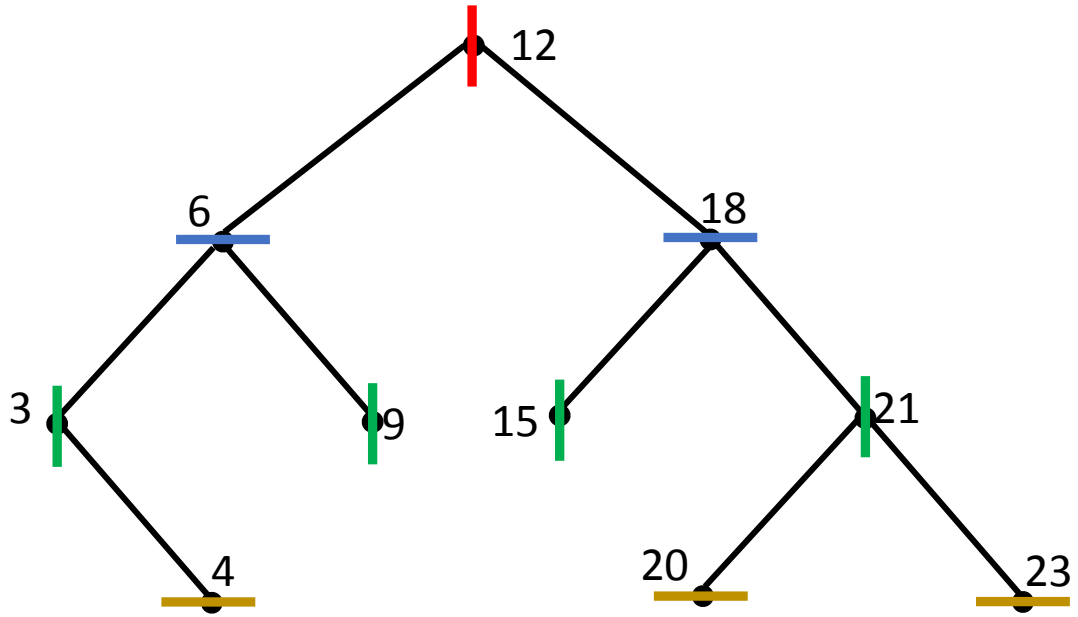




Space-partitioning Tree

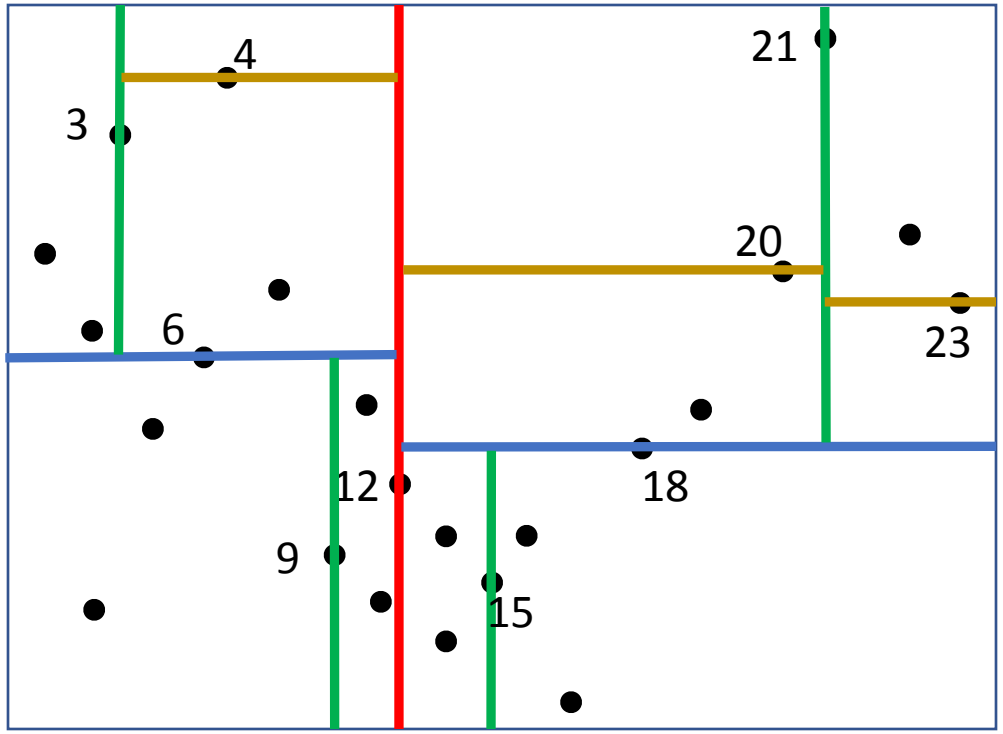
- K-d tree

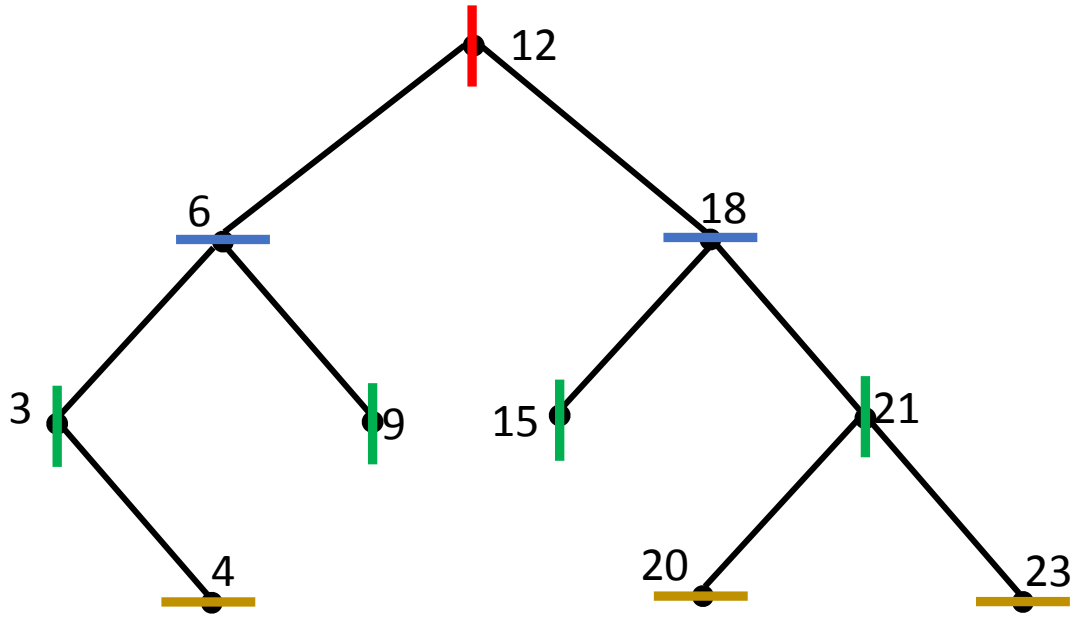




Space-partitioning Tree

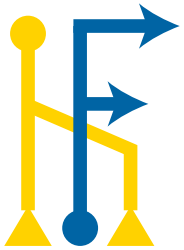
- K-d tree





Recursively divide space until each box has at most q data point

Classification of N-body Problems



Classification of N-body Problems

- Approximation

Classification of N-body Problems

- Approximation → Kernel Density Estimation

$$\forall q \in Q, \quad \sum_{r \in R} K_\sigma \left(\frac{\|x_q - x_r\|}{\sigma} \right)$$

Classification of N-body Problems

- Approximation → Kernel Density Estimation

$$\forall q \in Q, \quad \sum_{r \in R} K_\sigma \left(\frac{\|x_q - x_r\|}{\sigma} \right)$$

Arithmetic operator

Non-comparative kernel

Classification of N-body Problems

- Approximation → Kernel Density Estimation

$$\forall q \in Q, \quad \sum_{r \in R} K_\sigma \left(\frac{\|x_q - x_r\|}{\sigma} \right)$$

Arithmetic operator

Non-comparative kernel

- Pruning

Classification of N-body Problems

- Approximation → Kernel Density Estimation

$$\forall q \in Q, \quad \sum_{r \in R} K_\sigma \left(\frac{\|x_q - x_r\|}{\sigma} \right)$$

Arithmetic operator

Non-comparative kernel

- Pruning → Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \|x_q - x_r\|$$

Classification of N-body Problems

- Approximation → Kernel Density Estimation

$$\forall q \in Q, \quad \sum_{r \in R} K_\sigma \left(\frac{\|x_q - x_r\|}{\sigma} \right)$$

Arithmetic operator

Non-comparative kernel

- Pruning → Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \|x_q - x_r\|$$

Comparative operator

Classification of N-body Problems

- Approximation → Kernel Density Estimation

$$\forall q \in Q, \quad \sum_{r \in R} K_\sigma \left(\frac{\|x_q - x_r\|}{\sigma} \right)$$

Arithmetic operator

Non-comparative kernel

- Pruning → Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \|x_q - x_r\|$$

Comparative operator

Classification of N-body Problems

- Approximation → Kernel Density Estimation

$$\forall q \in Q, \quad \sum_{r \in R} K_\sigma \left(\frac{\|x_q - x_r\|}{\sigma} \right)$$

Arithmetic operator

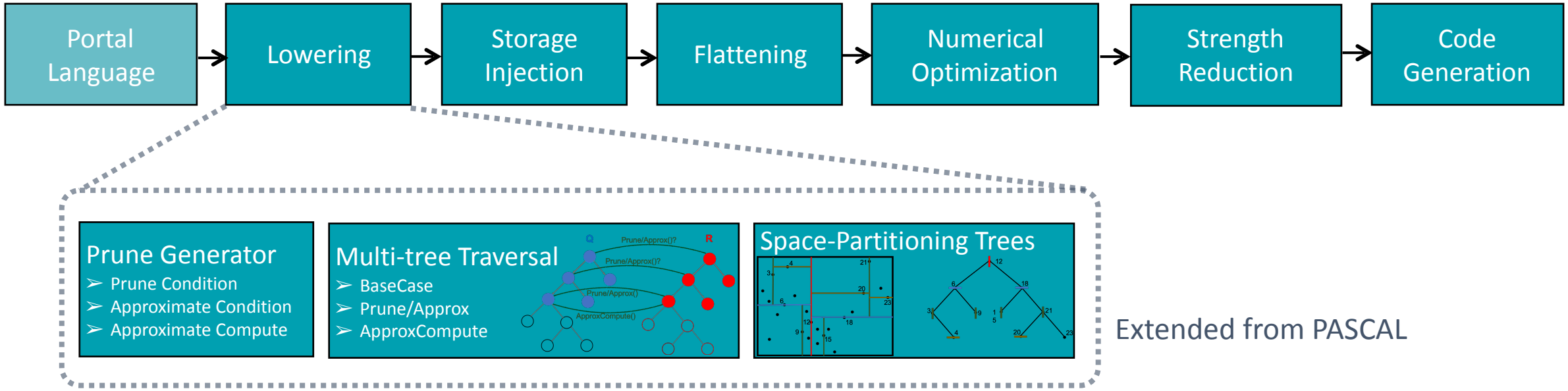
Non-comparative kernel

- Pruning → Nearest Neighbor

$$\forall q \in Q, \quad \arg\min_{r \in R} \|x_q - x_r\|$$

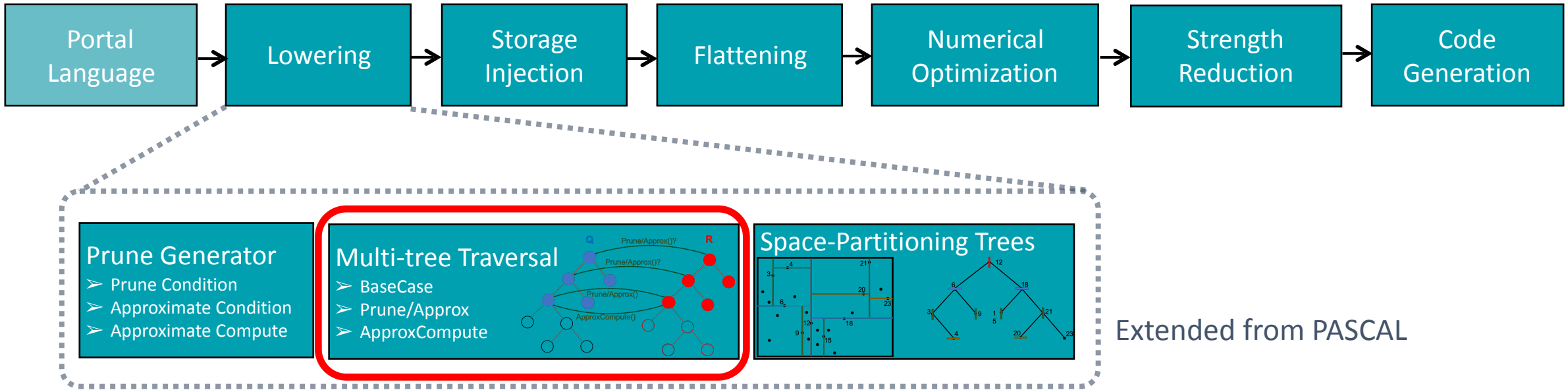
Comparative operator

Portal



Extended from PASCAL

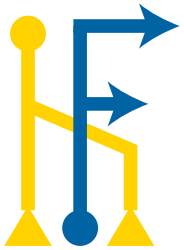
Portal



Extended from PASCAL

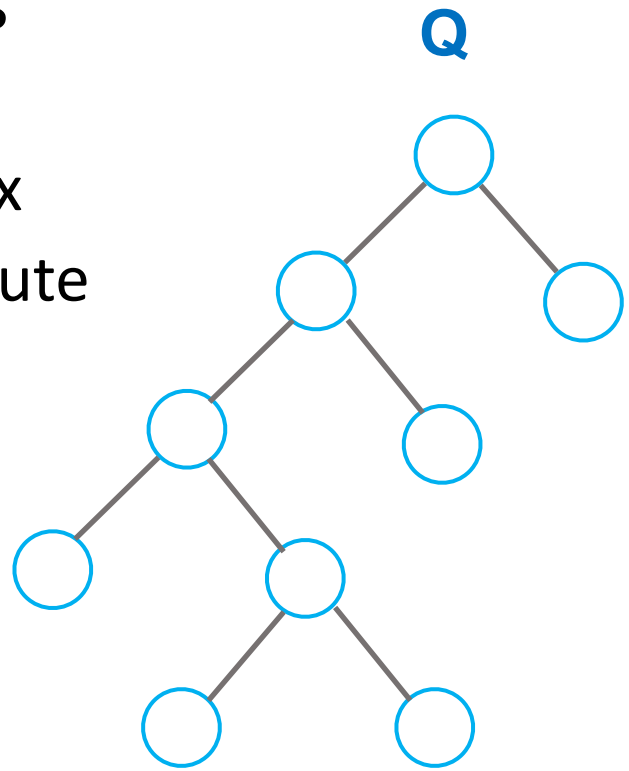
Multi-tree Traversal

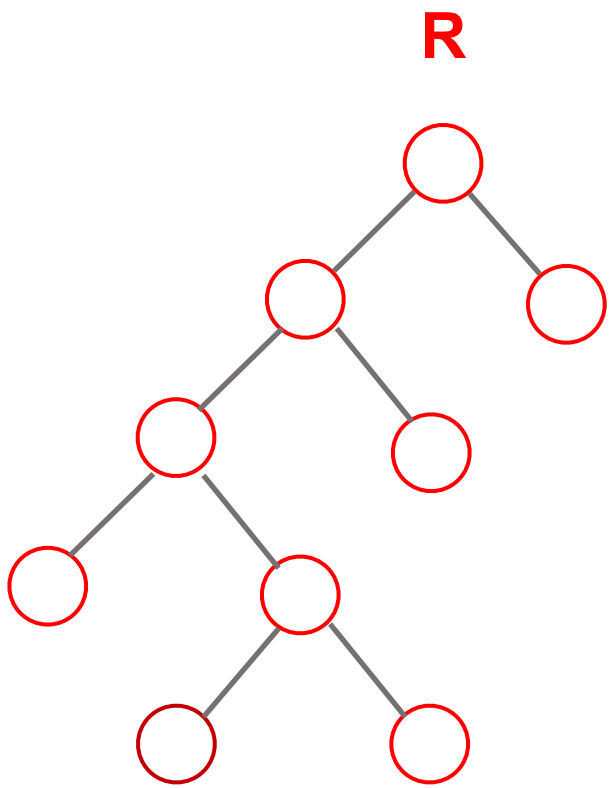
- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute



Multi-tree Traversal

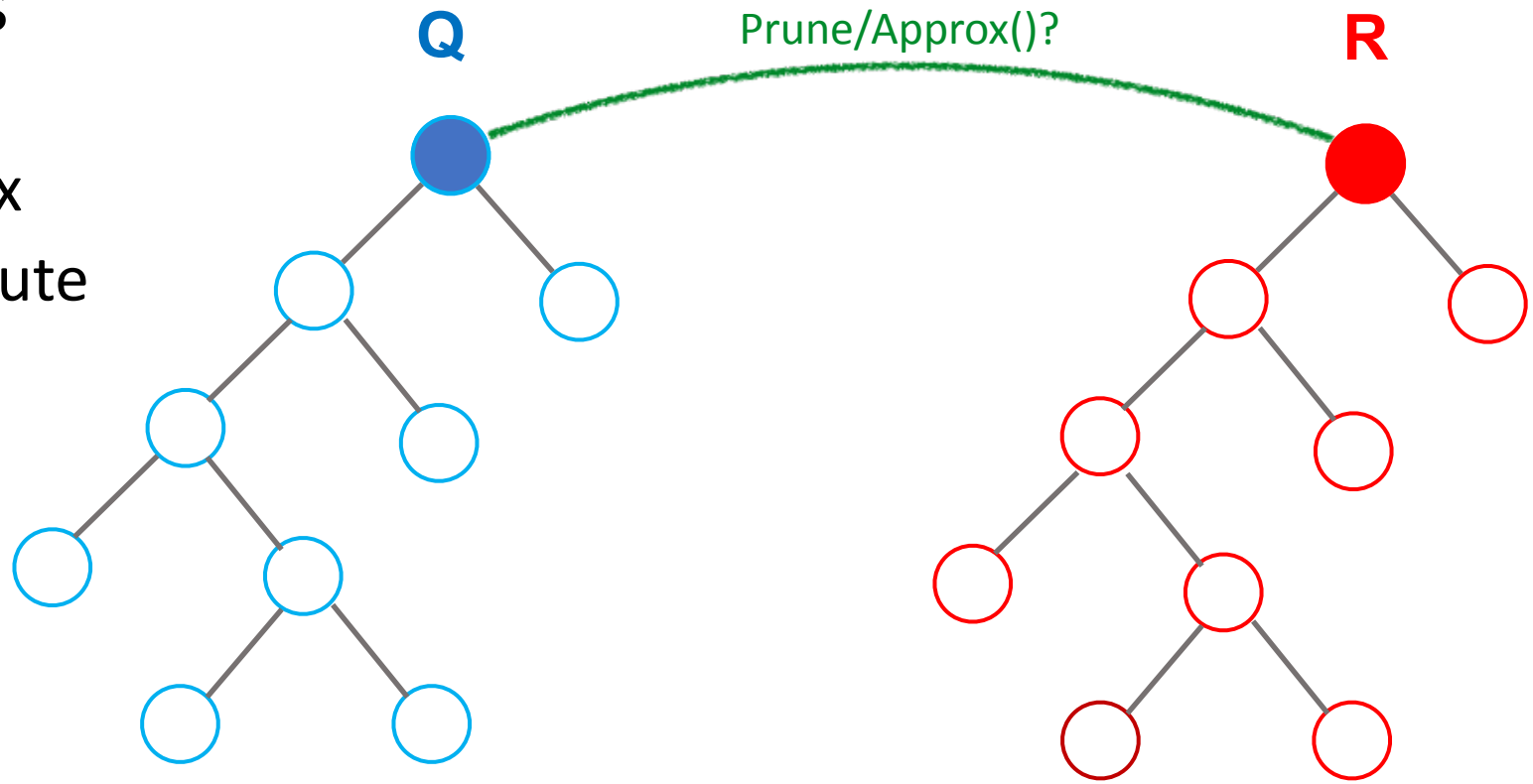
- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute





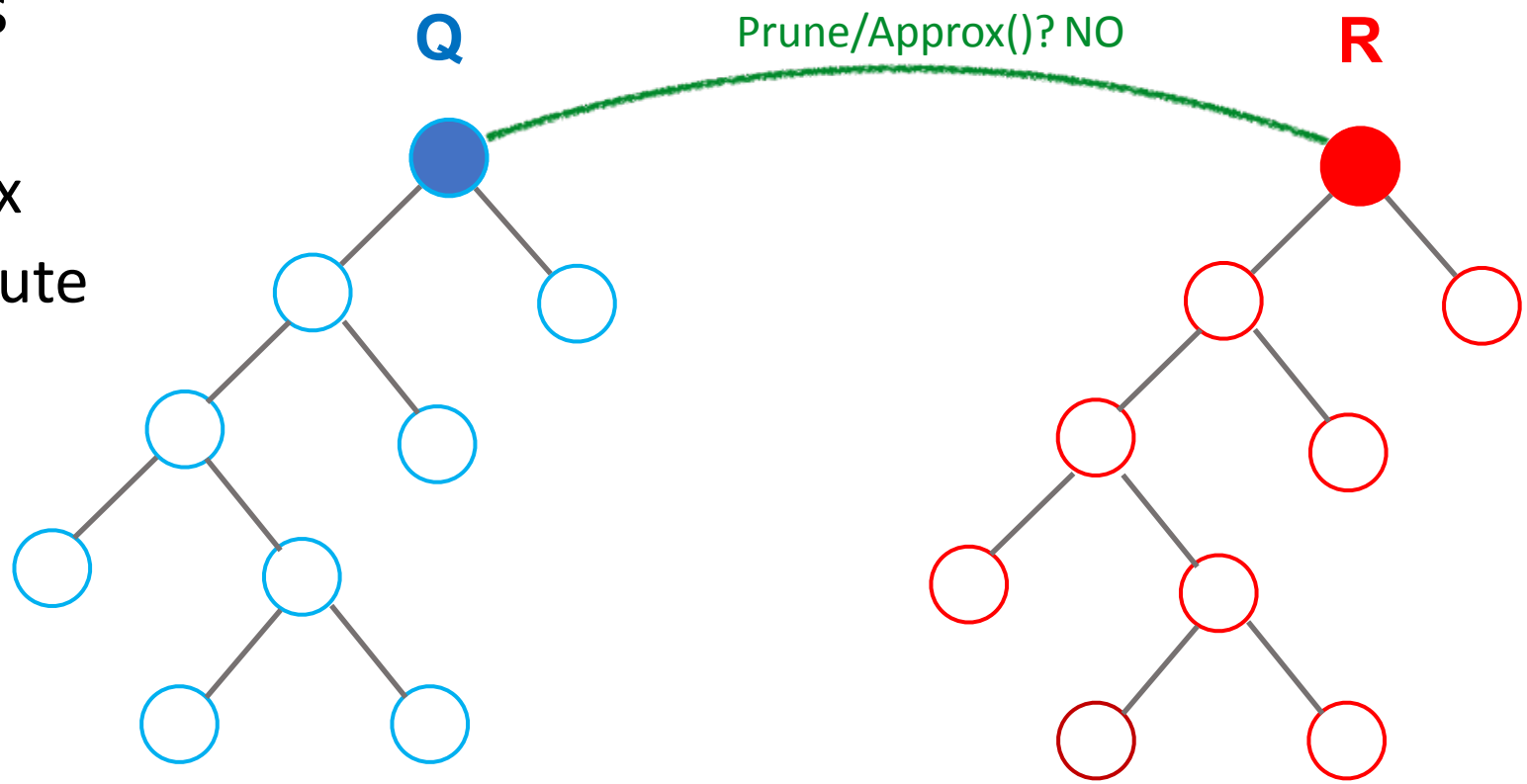
Multi-tree Traversal

- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute



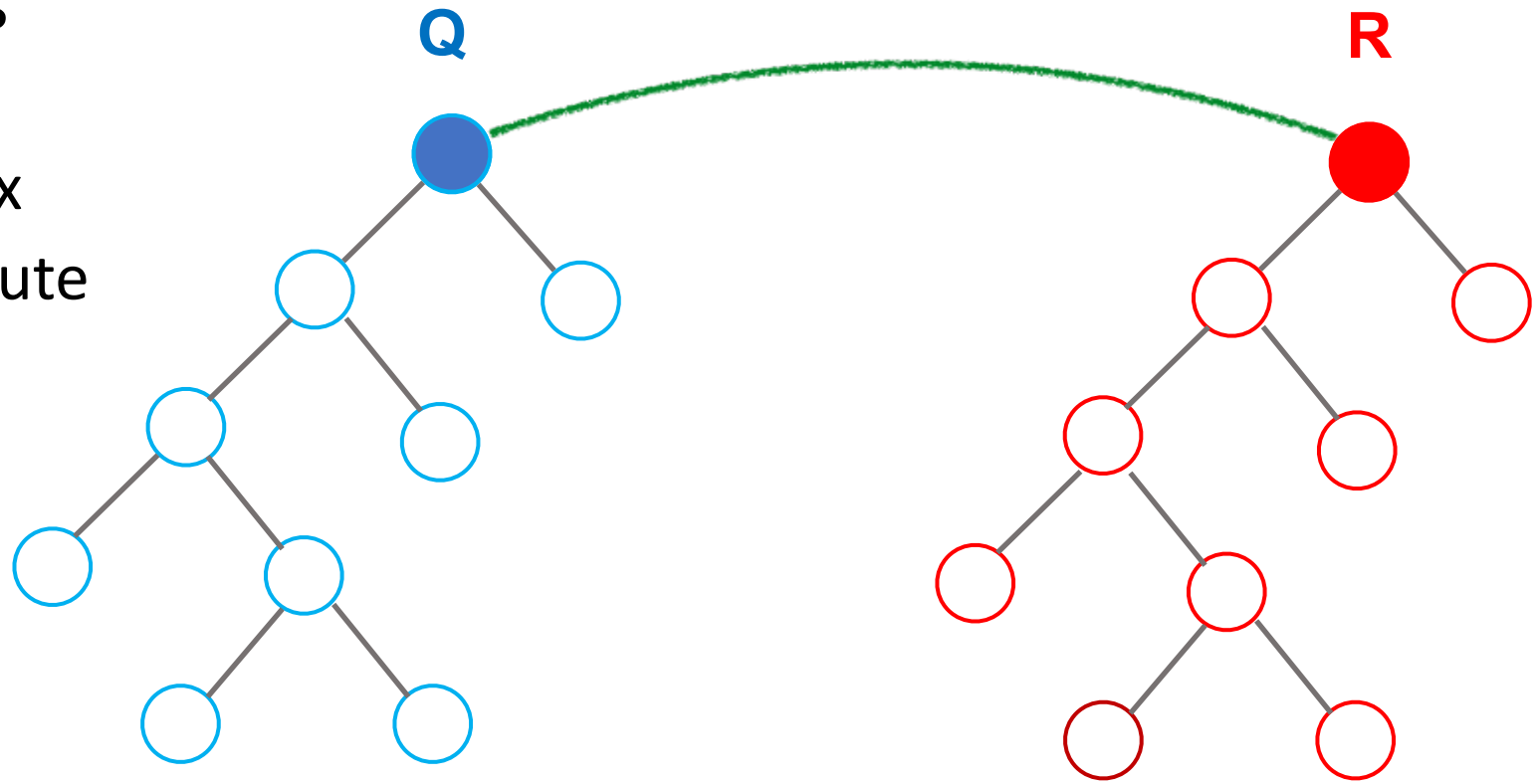
Multi-tree Traversal

- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute



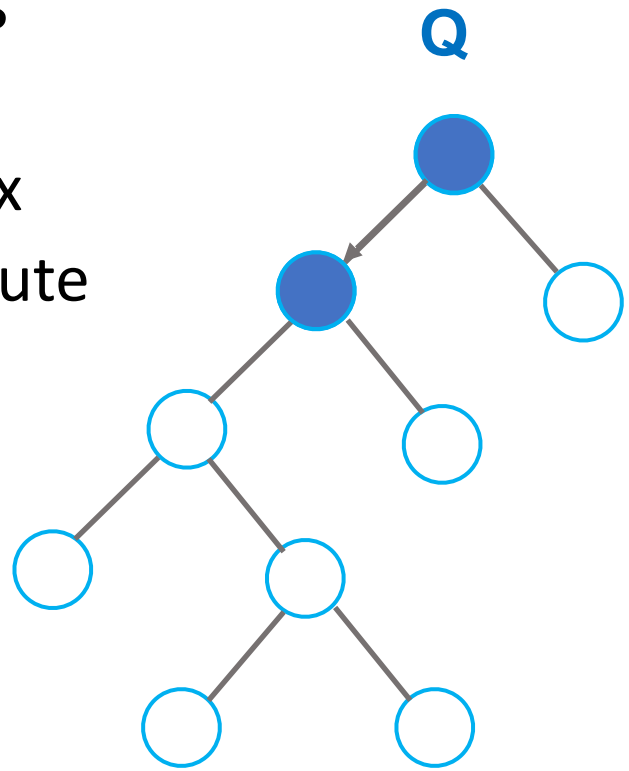
Multi-tree Traversal

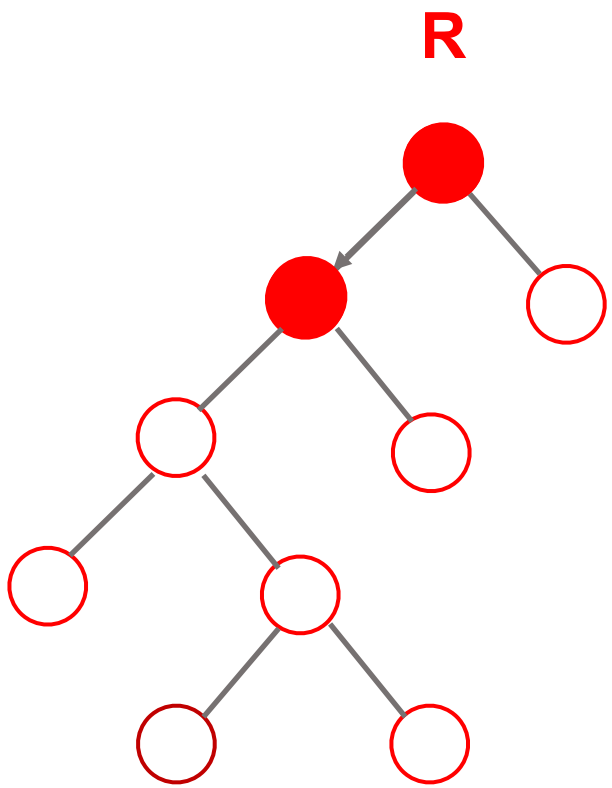
- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute



Multi-tree Traversal

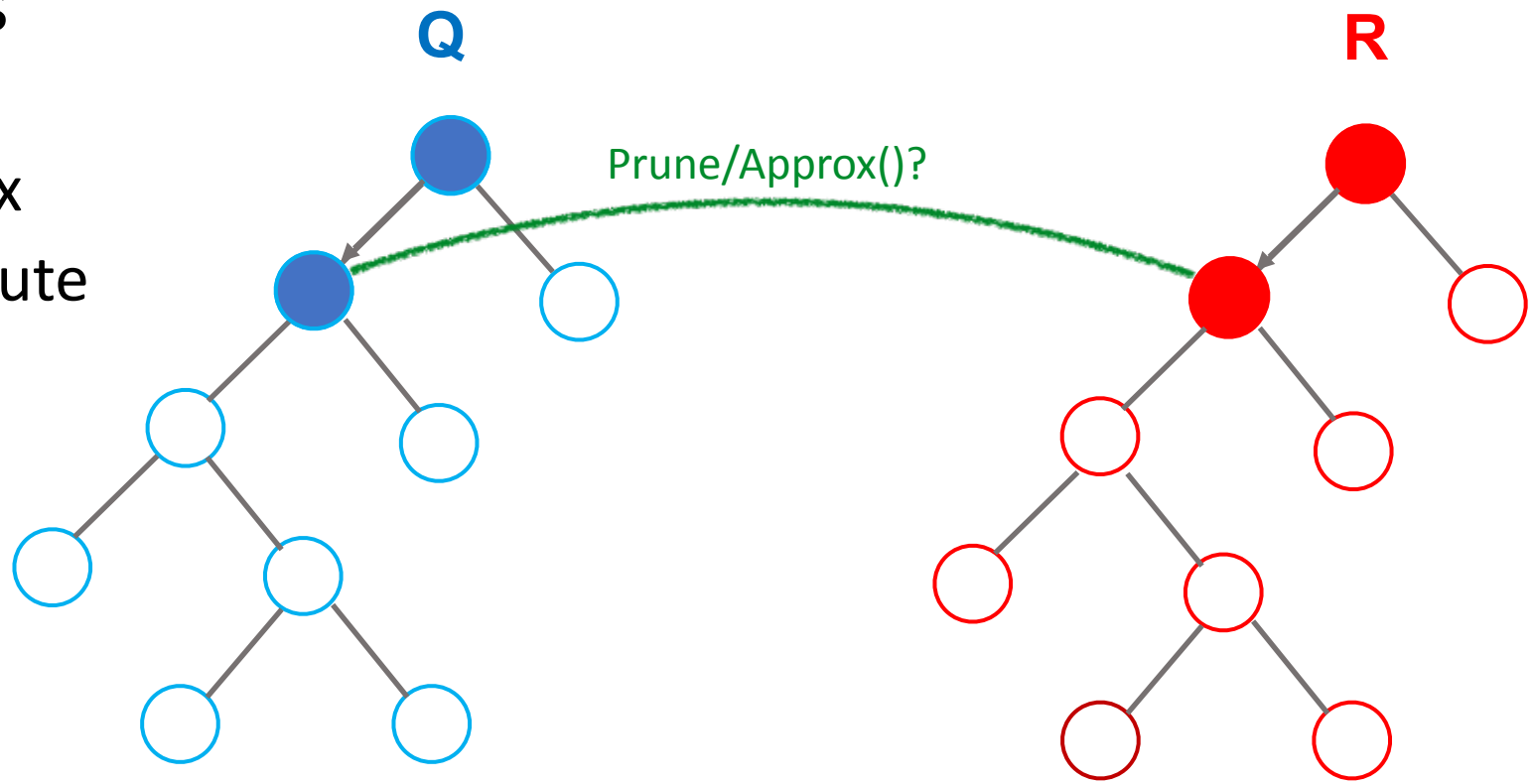
- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute





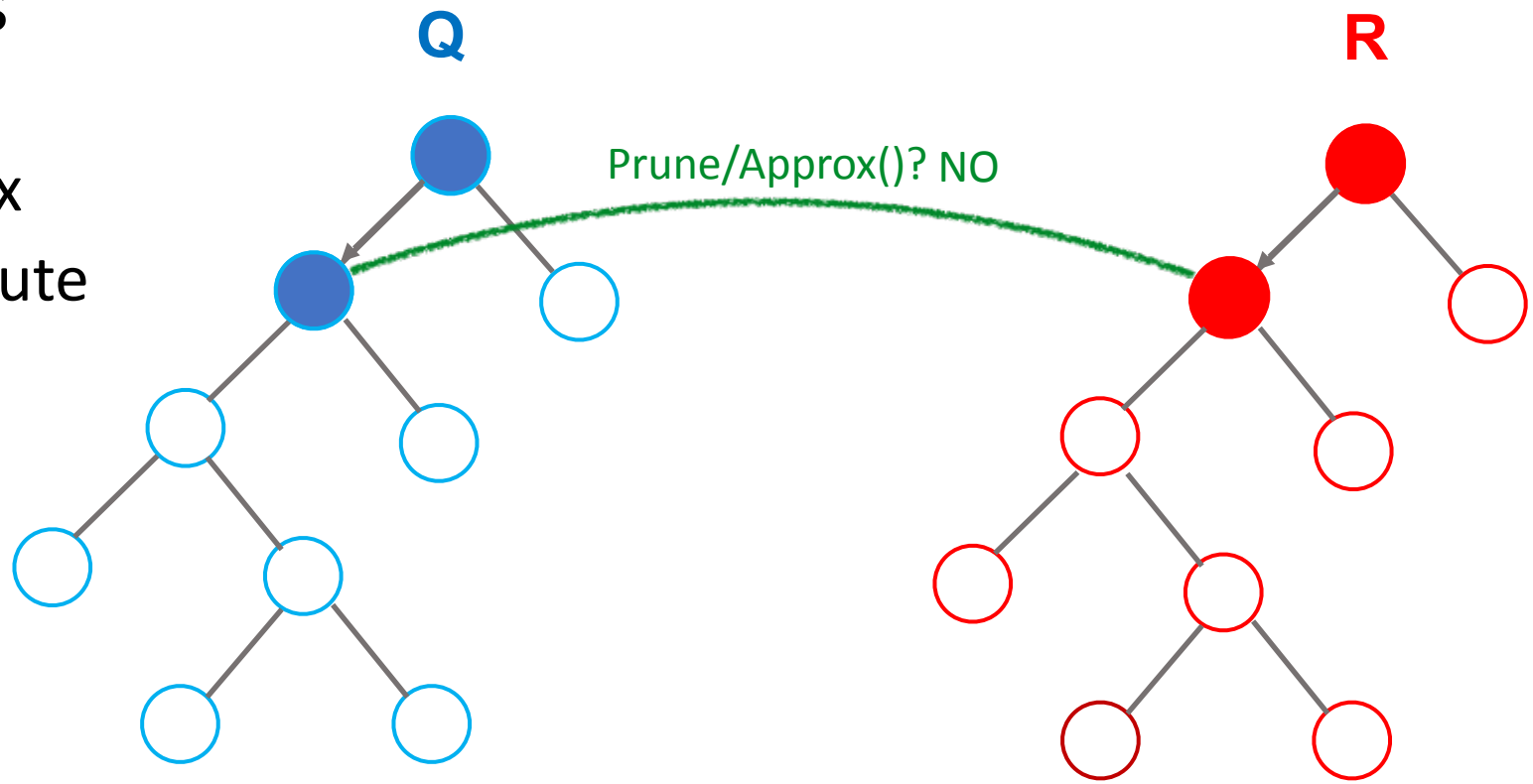
Multi-tree Traversal

- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute



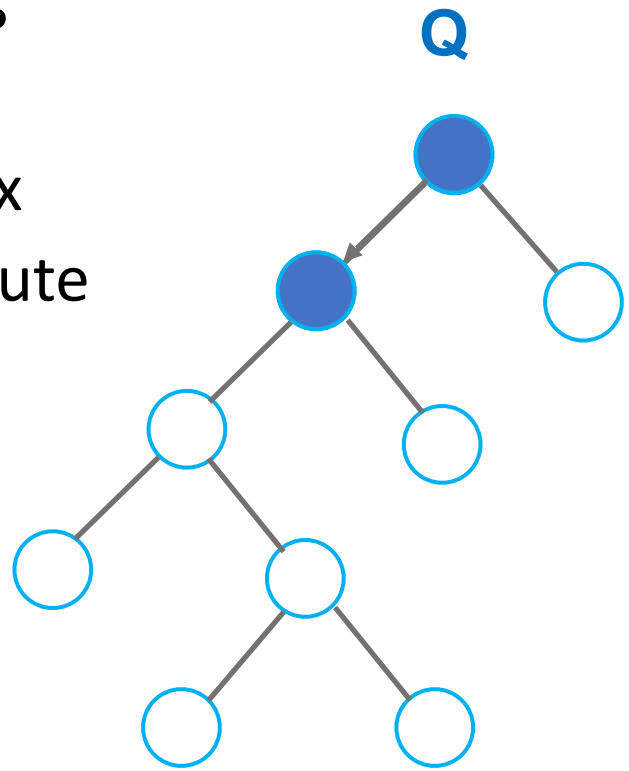
Multi-tree Traversal

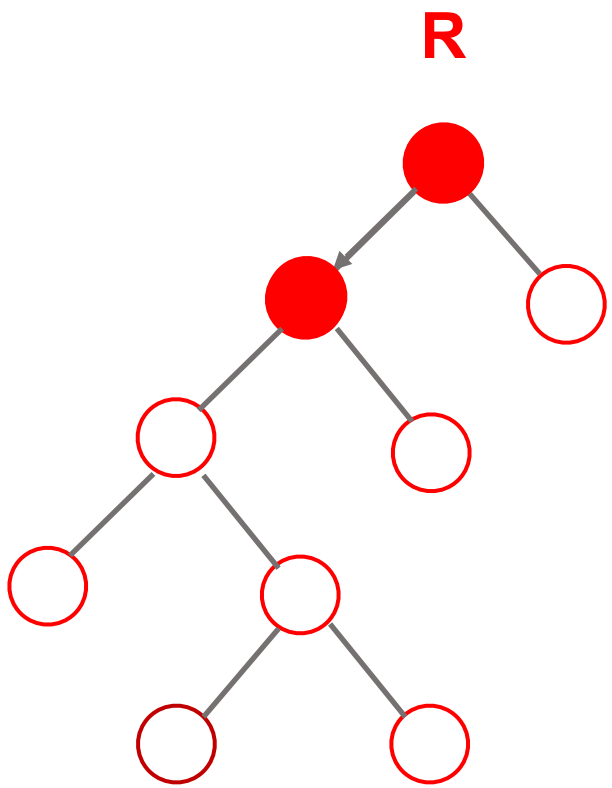
- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute



Multi-tree Traversal

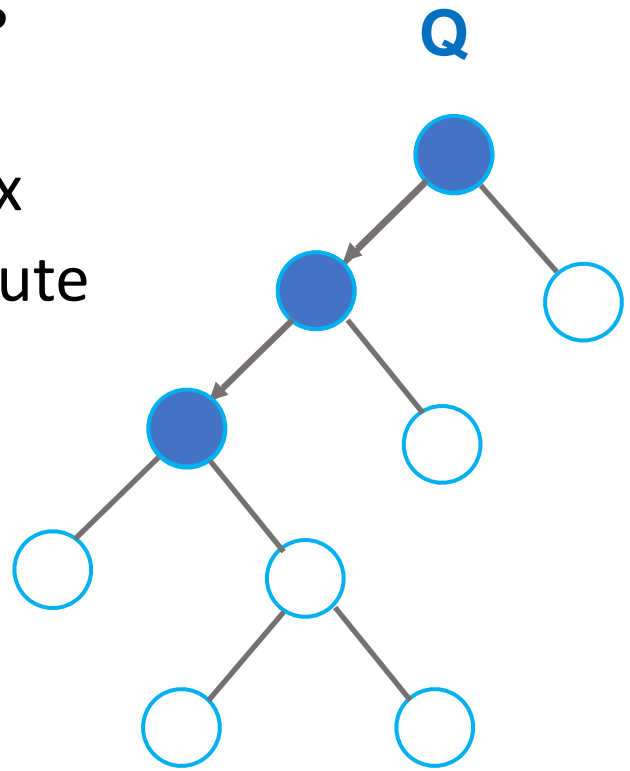
- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute

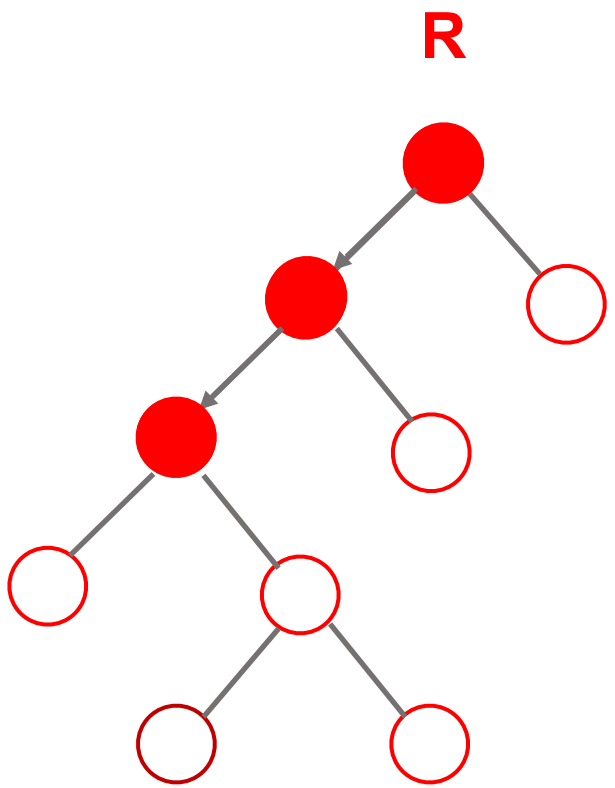




Multi-tree Traversal

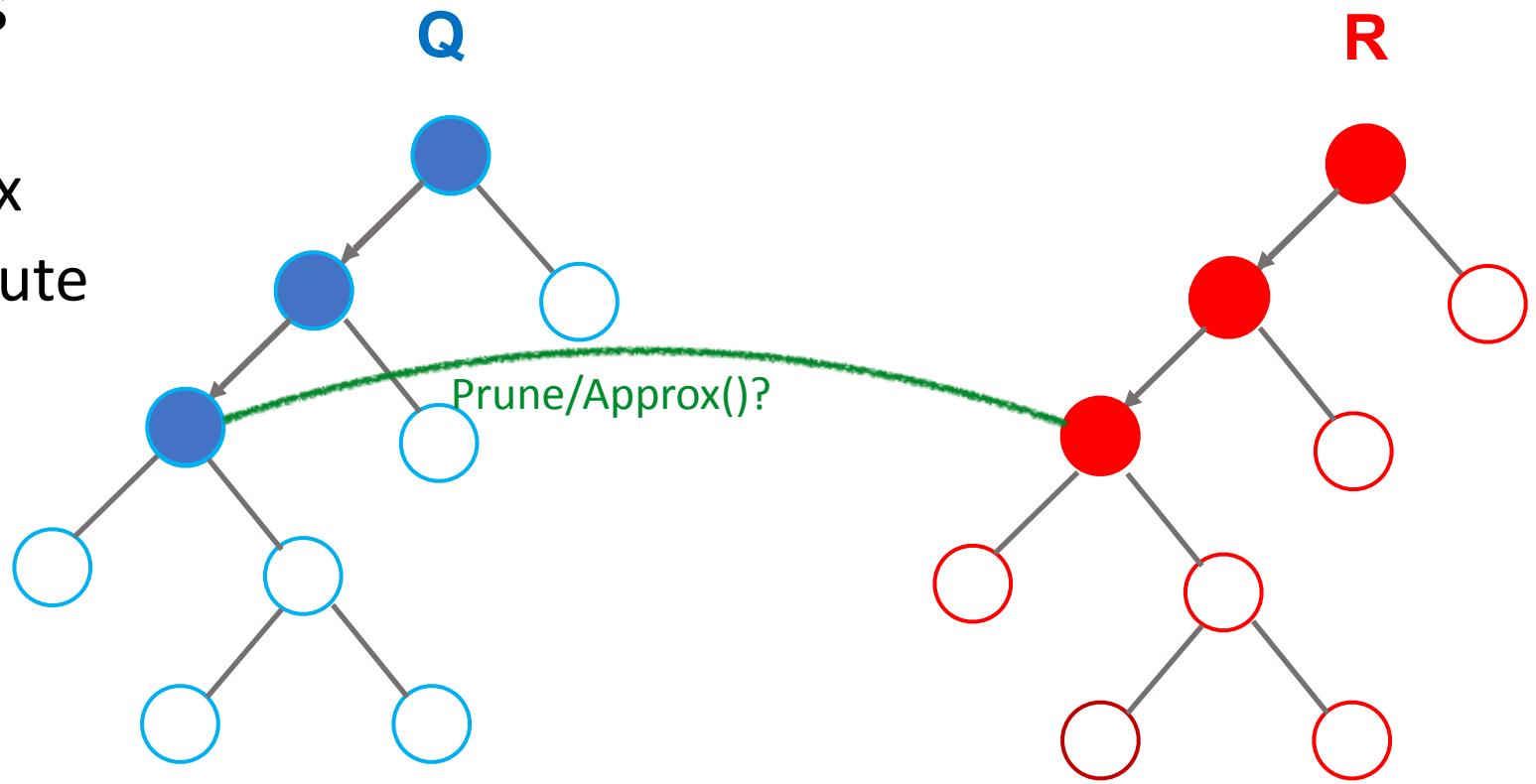
- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute





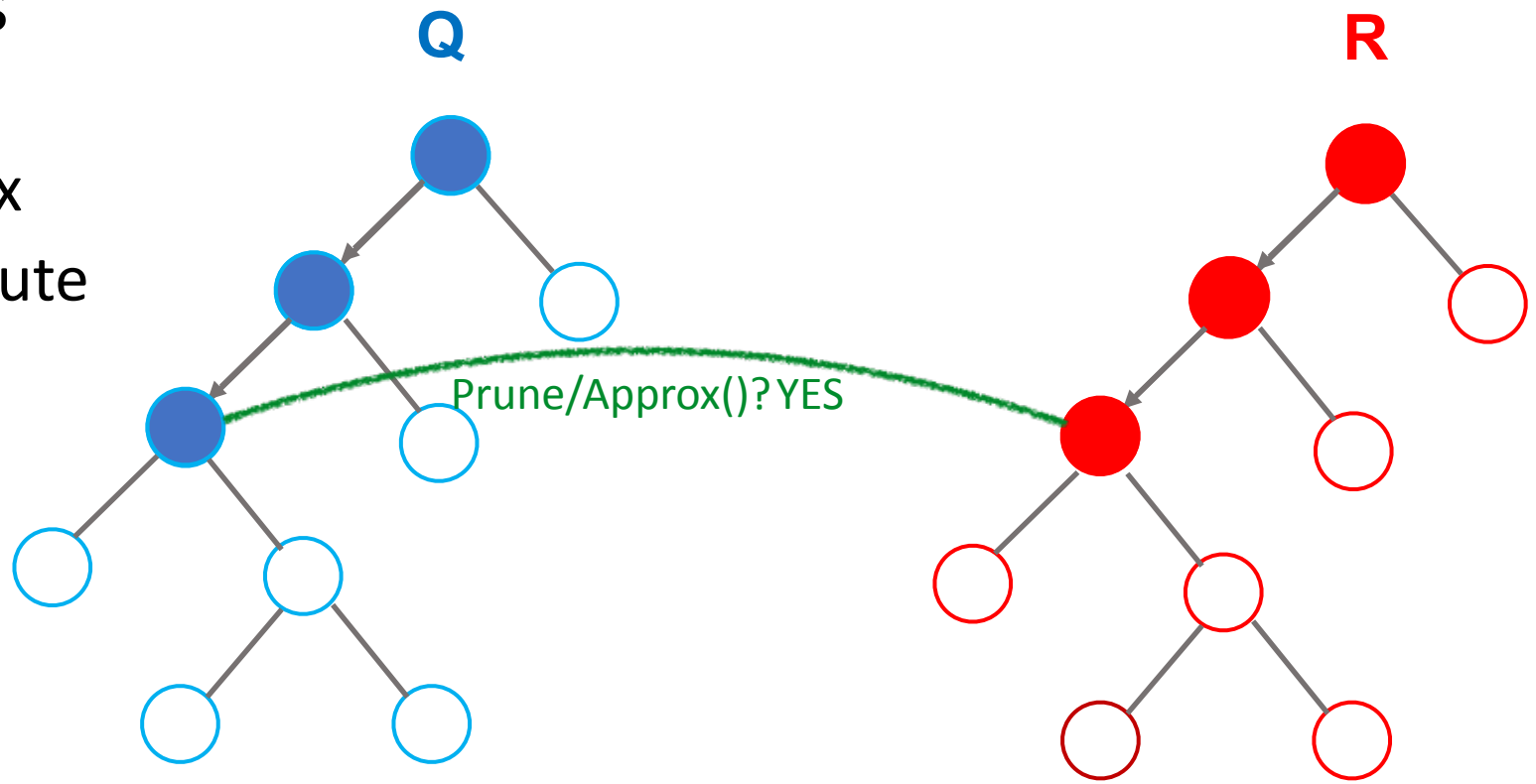
Multi-tree Traversal

- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute



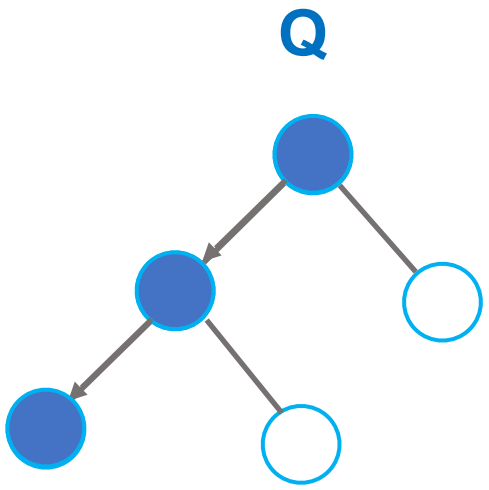
Multi-tree Traversal

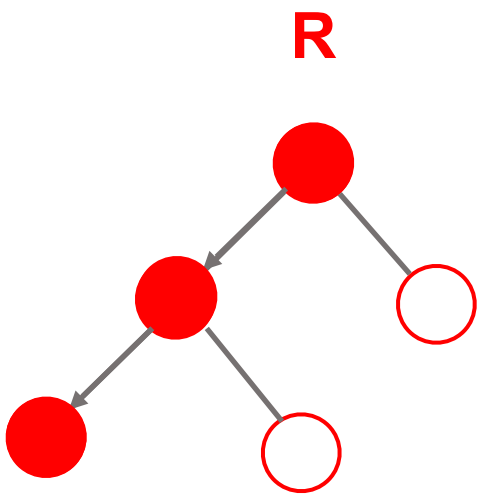
- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute



Multi-tree Traversal

- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute

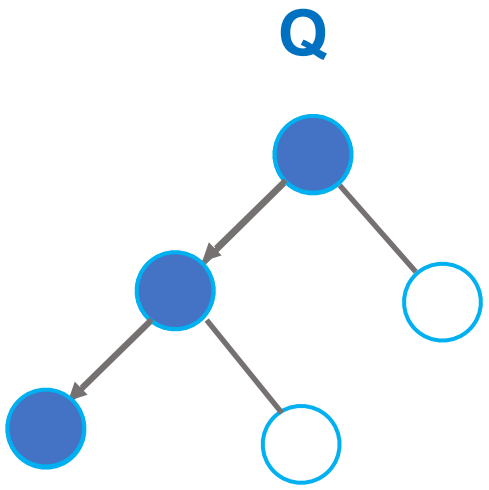


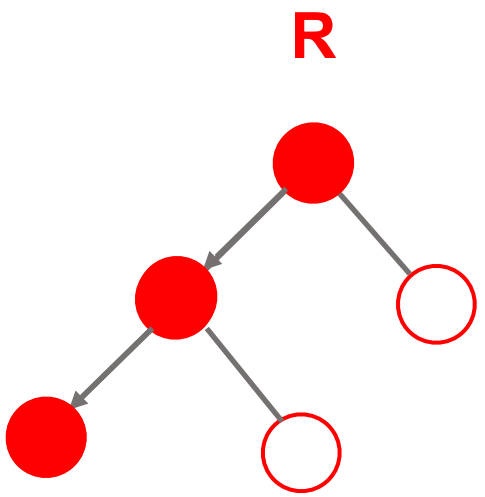


For pruning problems: if Prune/Approx() is true, discard the entire subtree

Multi-tree Traversal

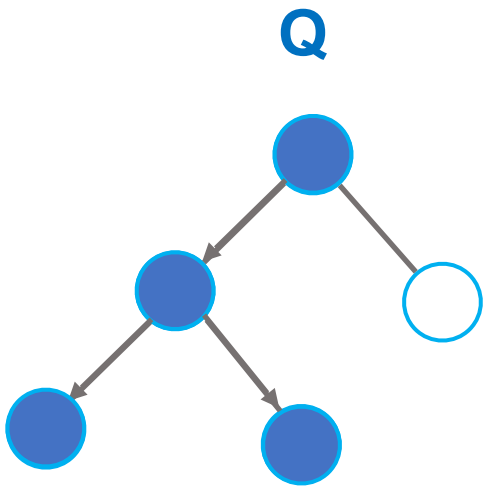
- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute

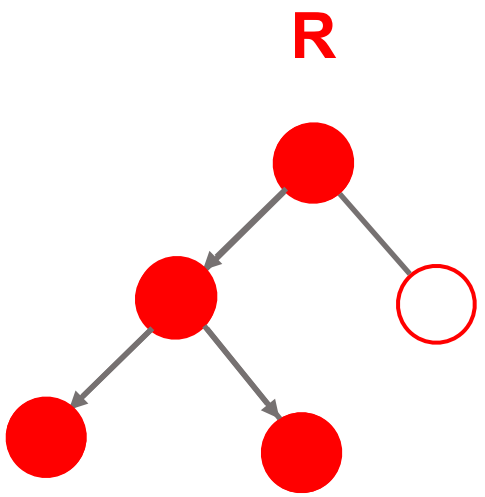




Multi-tree Traversal

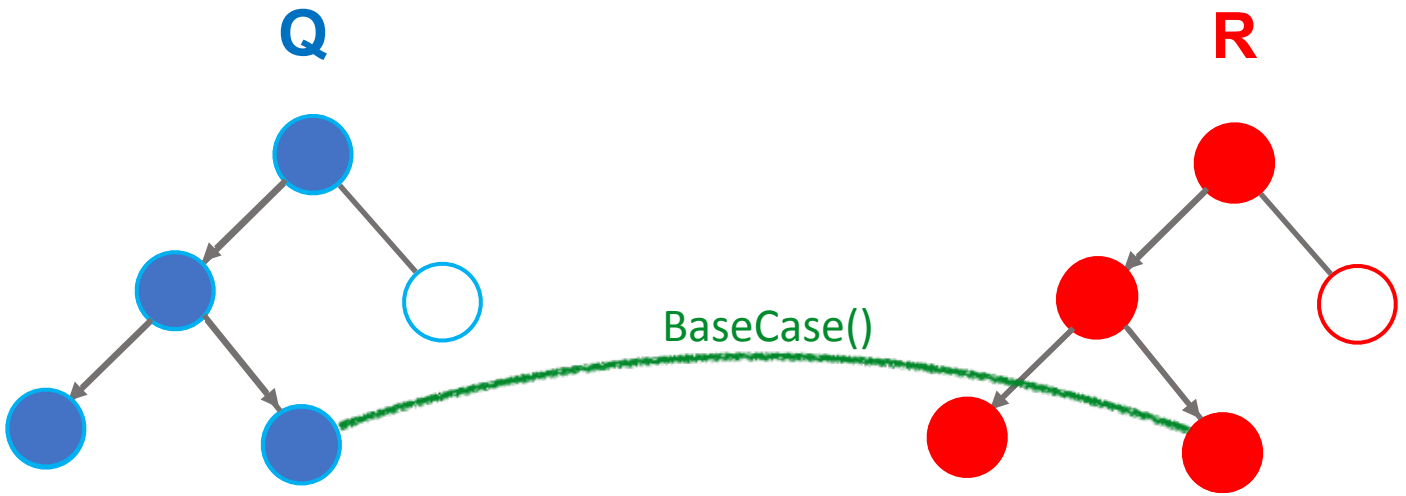
- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute





Multi-tree Traversal

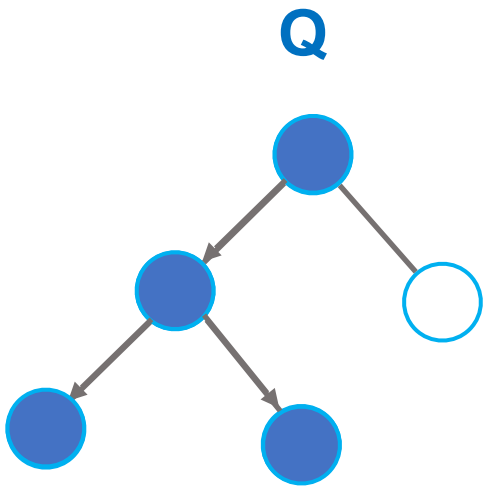
- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute

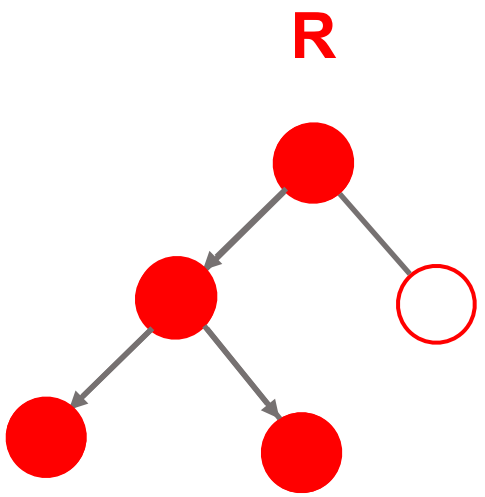


Direct computation $O(q^2)$

Multi-tree Traversal

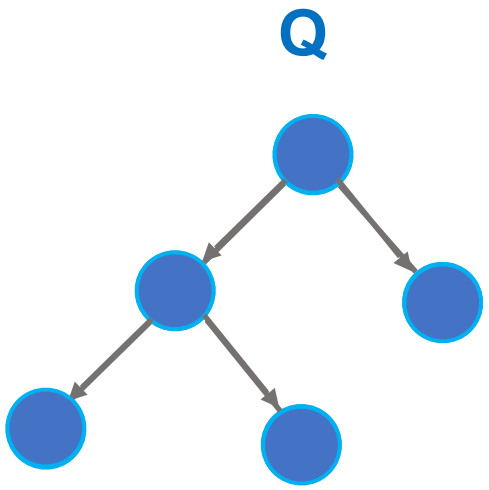
- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute

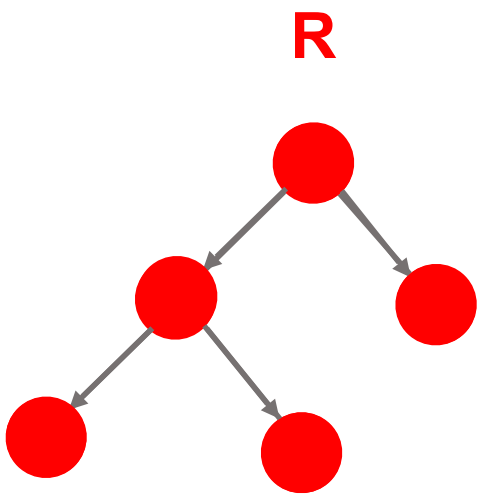




Multi-tree Traversal

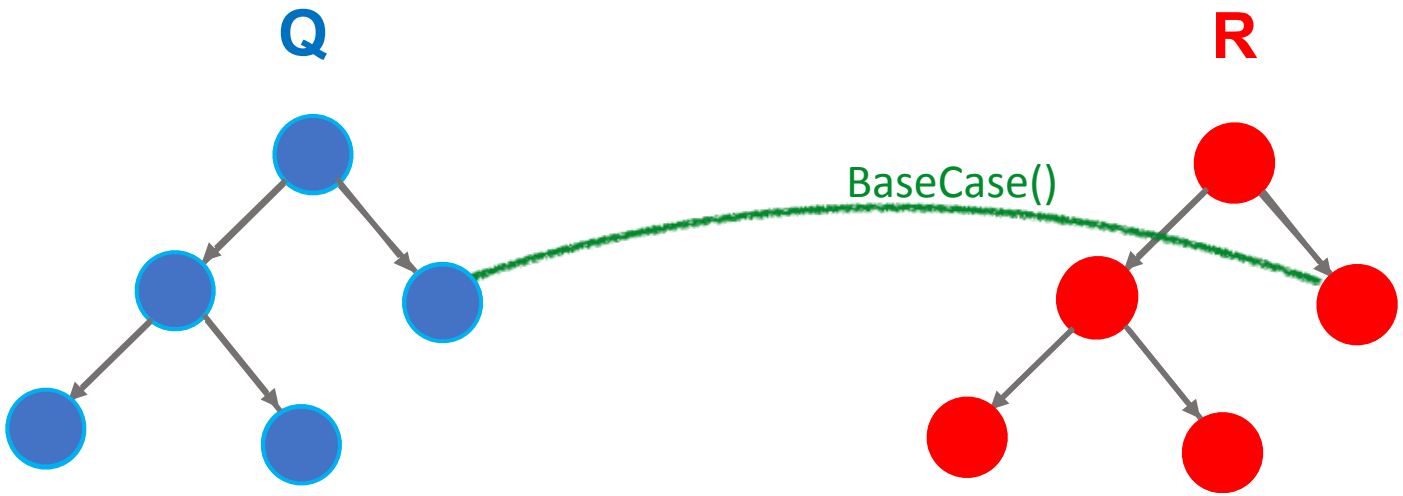
- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute





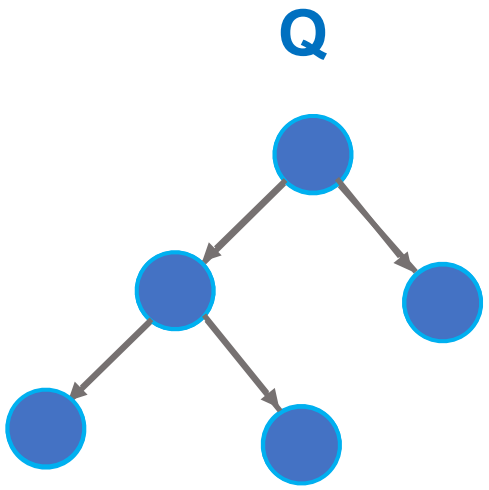
Multi-tree Traversal

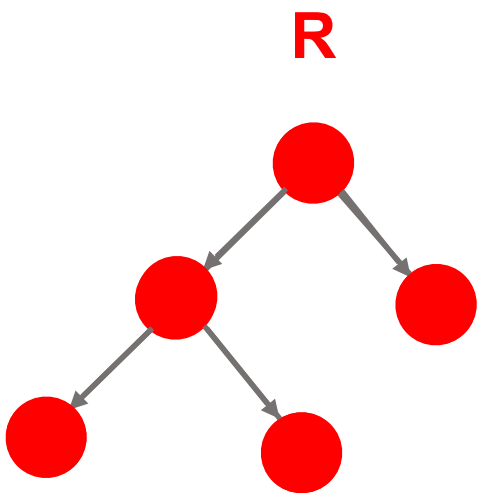
- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute



Multi-tree Traversal

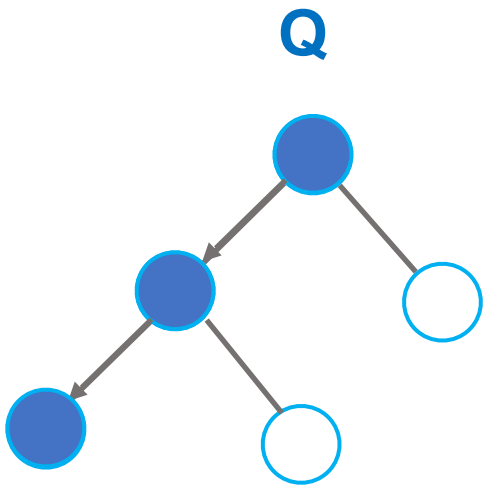
- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute

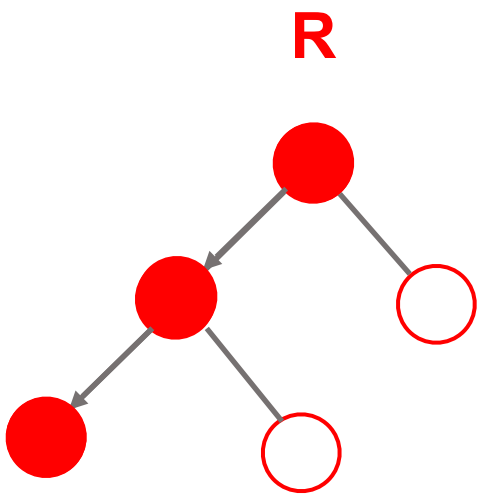




Multi-tree Traversal

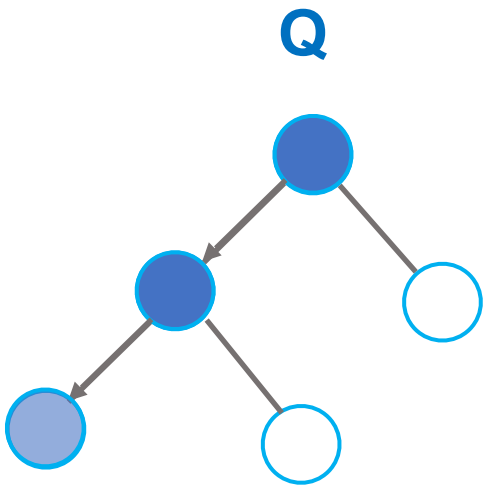
- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute

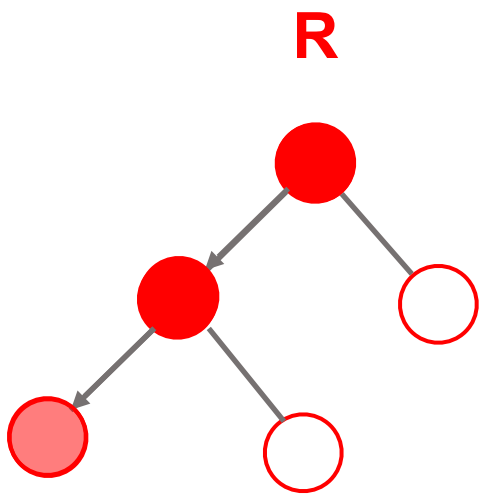




Multi-tree Traversal

- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute

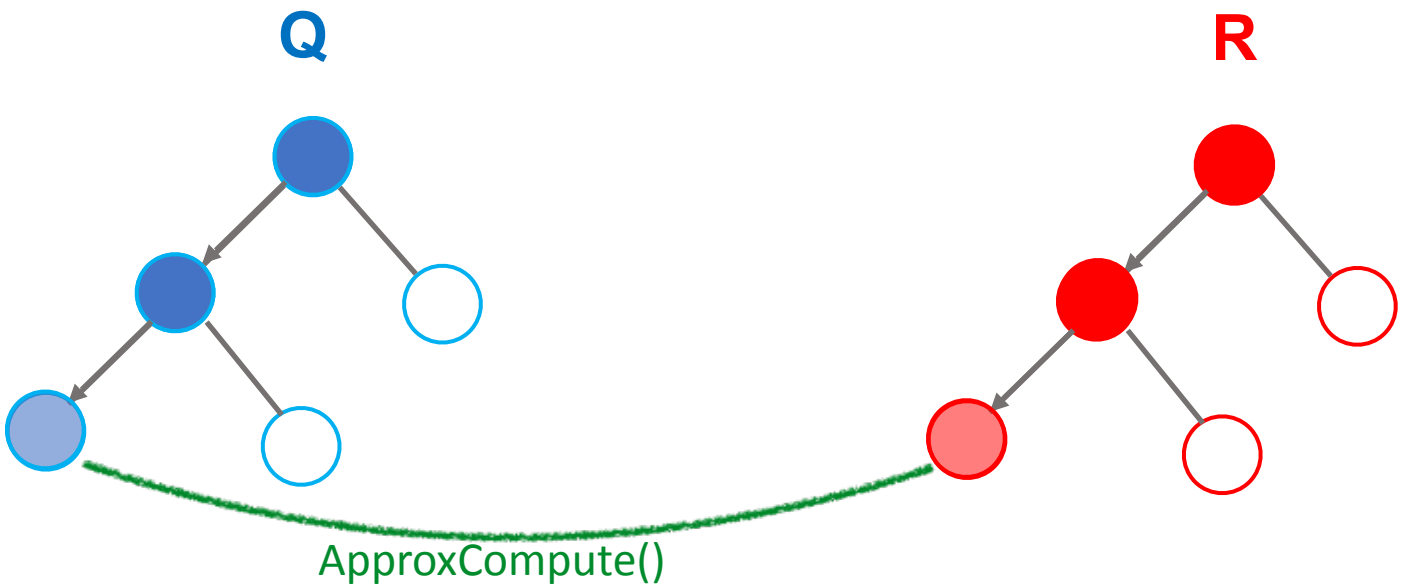




For Approximation problems: if Prune/Approx() is true, replace the subtree with the centroid

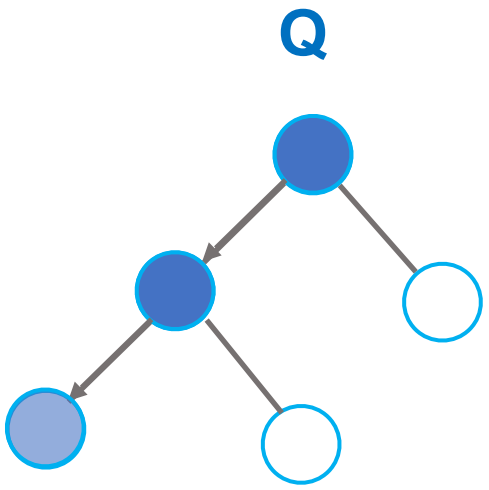
Multi-tree Traversal

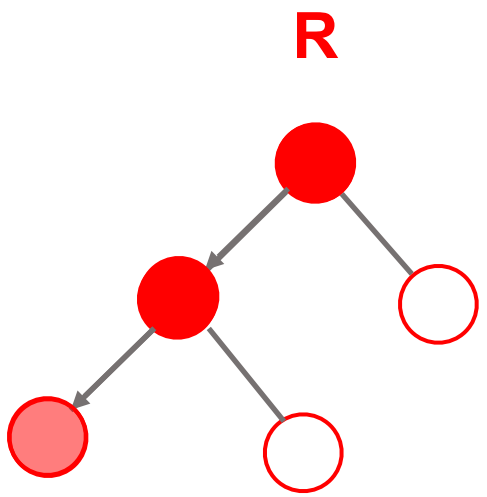
- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute



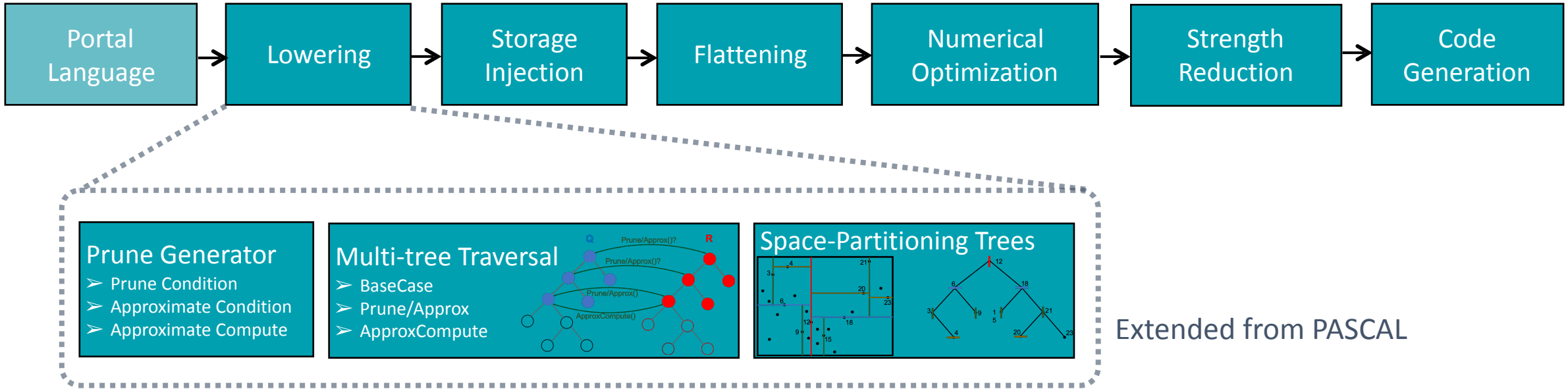
Multi-tree Traversal

- Functionalities
  - BaseCase
  - Prune/Approx
  - ApproxCompute



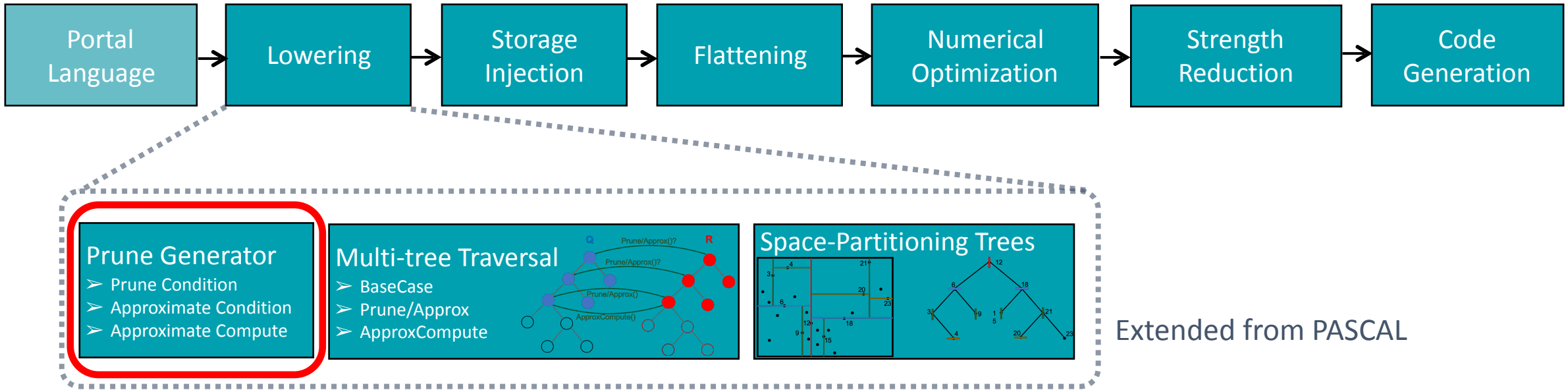


Portal



Extended from PASCAL

Portal



Extended from PASCAL

Prune/Approximate Generator

- Nearest Neighbor (Pruning class)

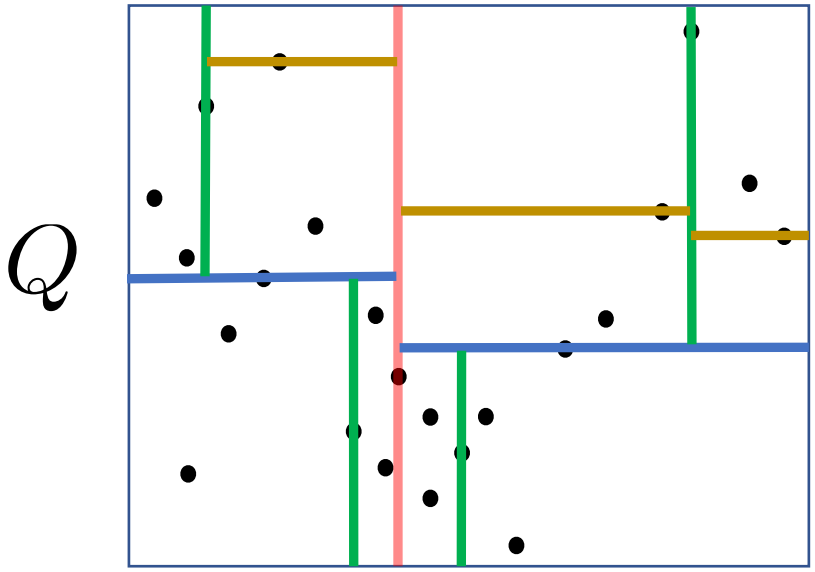
$$\forall q \in Q, \quad \arg\min_{r \in R} \quad \|x_q - x_r\|$$

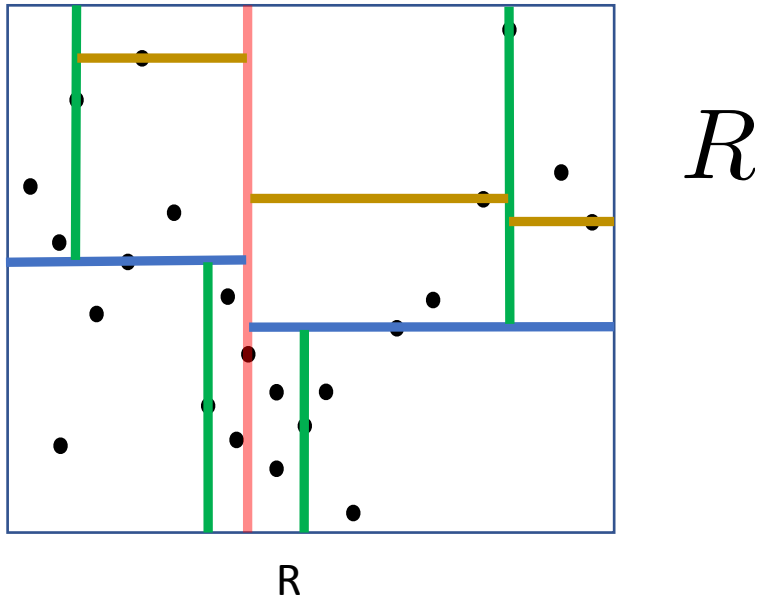
R

Prune/Approximate Generator

- Nearest Neighbor (Pruning class)

$$\forall q \in Q, \quad \arg\min_{r \in R} \ \|x_q - x_r\|$$

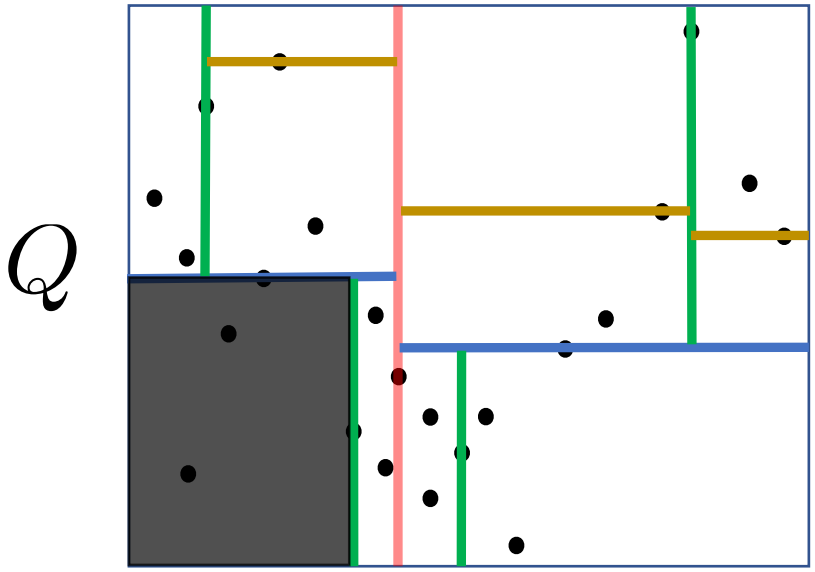


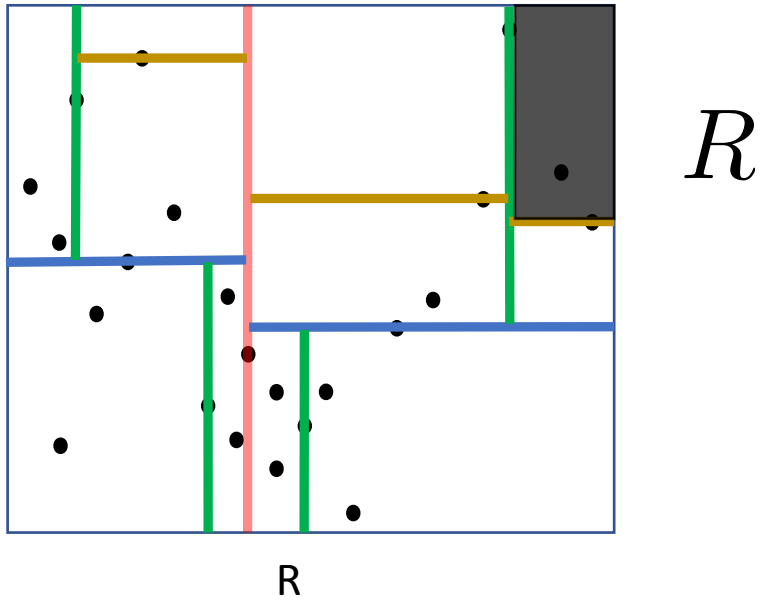


Prune/Approximate Generator

- Nearest Neighbor (Pruning class)

$$\forall q \in Q, \quad \arg\min_{r \in R} \ \|x_q - x_r\|$$

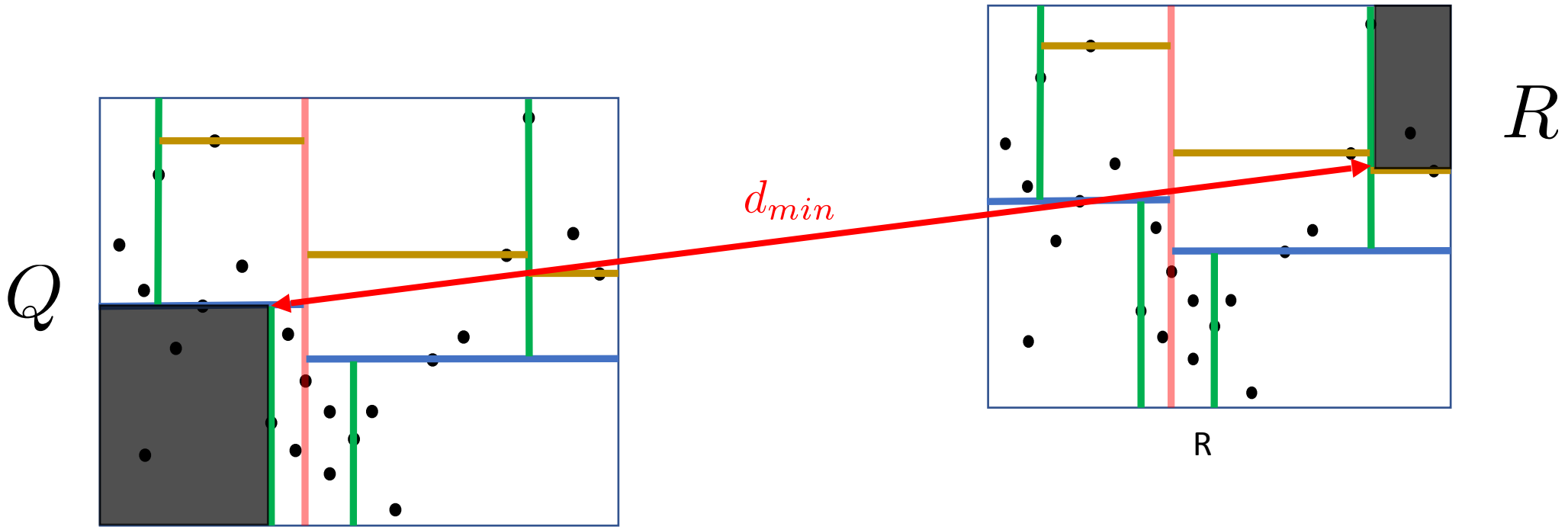




Prune/Approximate Generator

- Nearest Neighbor (Pruning class)

$$\forall q \in Q, \quad \arg\min_{r \in R} \ \|x_q - x_r\|$$

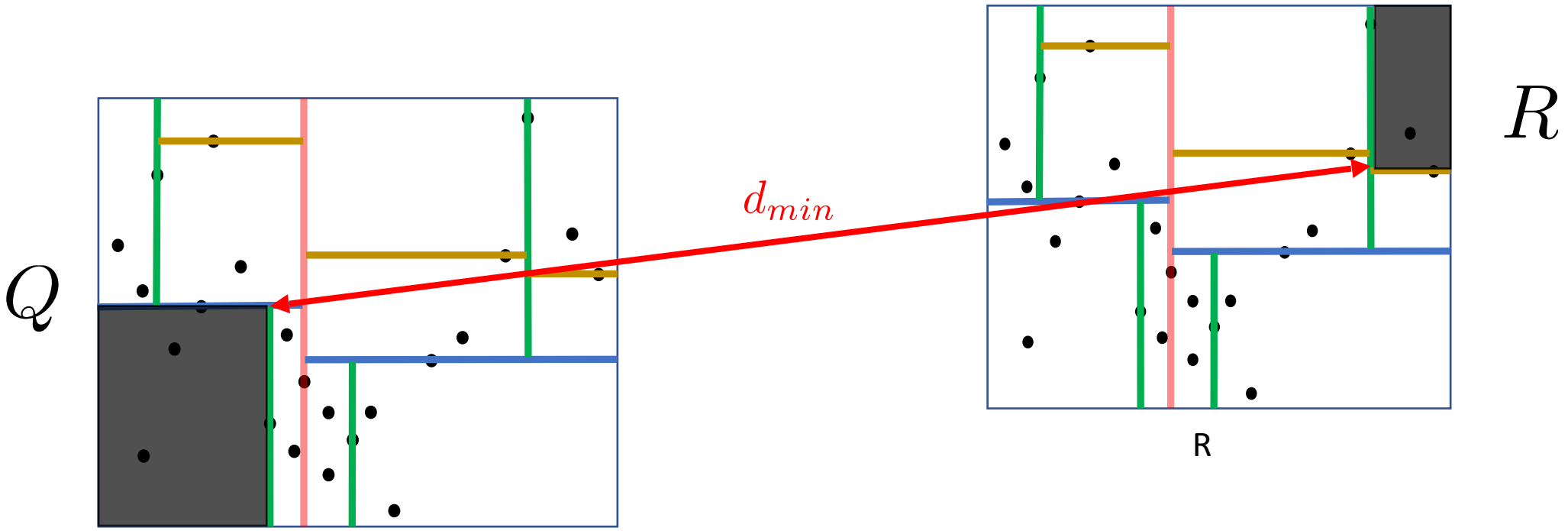


Prune/Approximate Generator

- Nearest Neighbor (Pruning class)
  - Prune condition

$$\forall q \in Q, \quad \arg\min_{r \in R} \ \|x_q - x_r\|$$

$$d_{sofar} < d_{min}$$



Prune/Approximate Generator

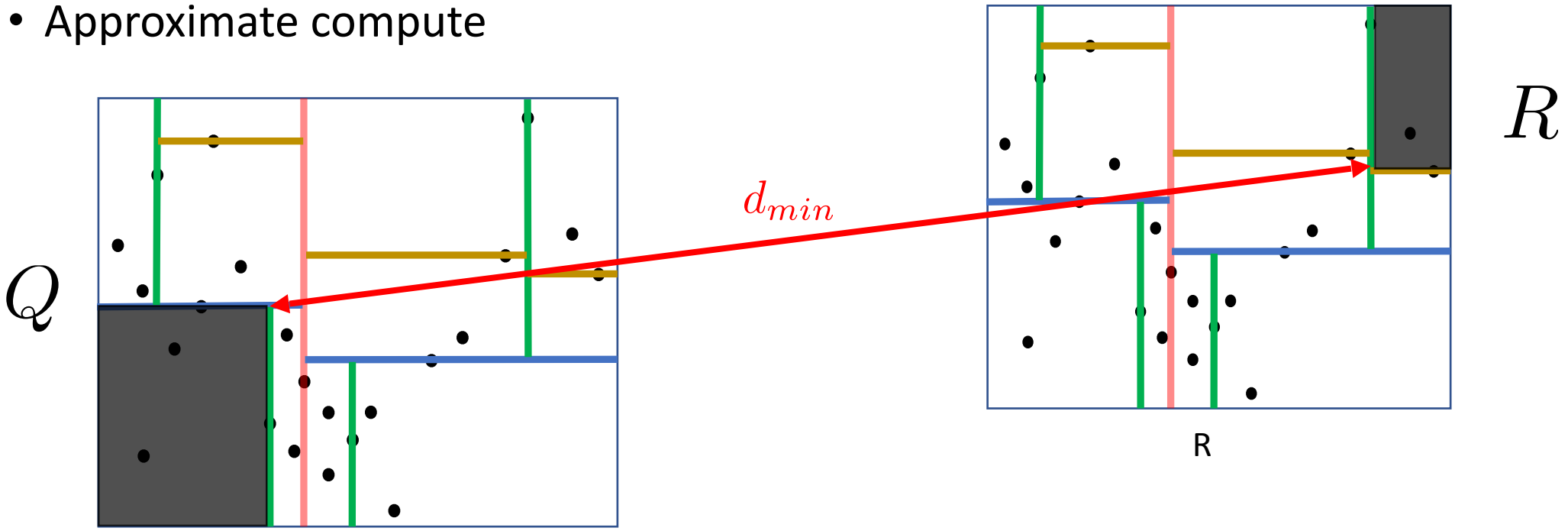
- Nearest Neighbor (Pruning class)

$$\forall q \in Q, \quad \arg\min_{r \in R} \ \|x_q - x_r\|$$

  - Prune condition

$$d_{sofar} < d_{min}$$

  - Approximate compute



Prune/Approximate Generator

- Nearest Neighbor (Pruning class)

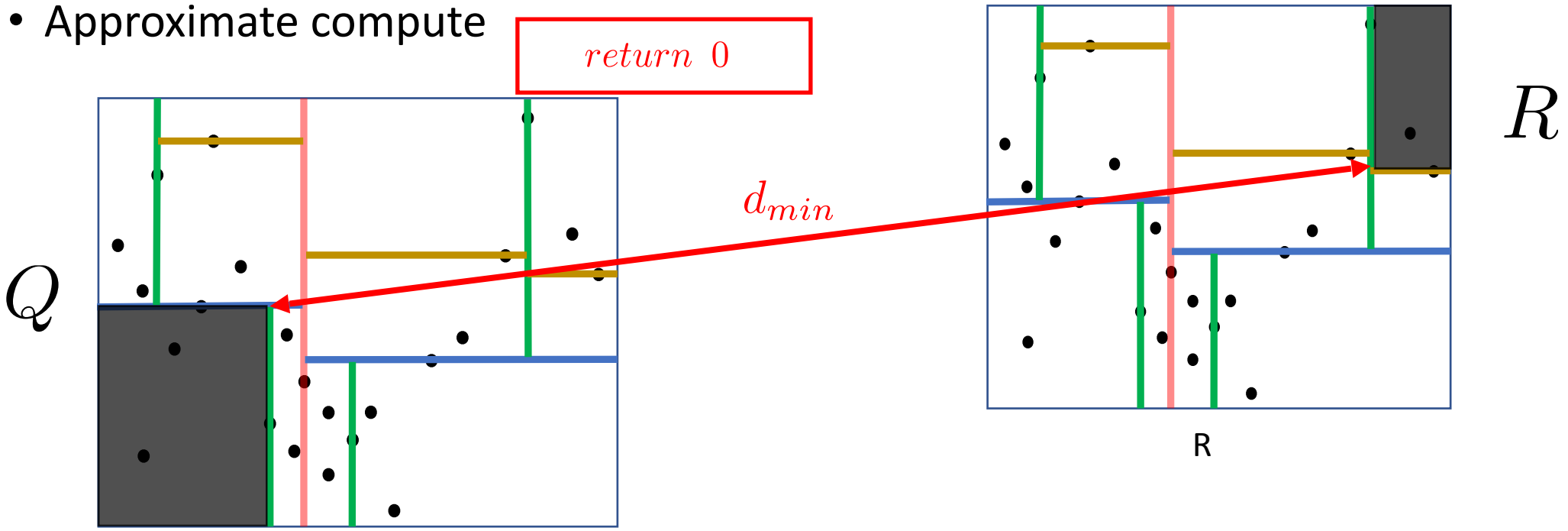
$$\forall q \in Q, \quad \arg\min_{r \in R} \ \|x_q - x_r\|$$

  - Prune condition

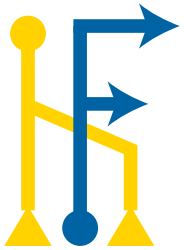
$$d_{sofar} < d_{min}$$

  - Approximate compute

$$return\ 0$$



Prune/Approximate Generator



Prune/Approximate Generator

- Kernel Density Estimation (Approximation class)
  - Approximate condition

$$\forall q \in Q, \quad \sum_{r \in R} K_\sigma \left(\frac{\|x_q - x_r\|}{\sigma} \right)$$

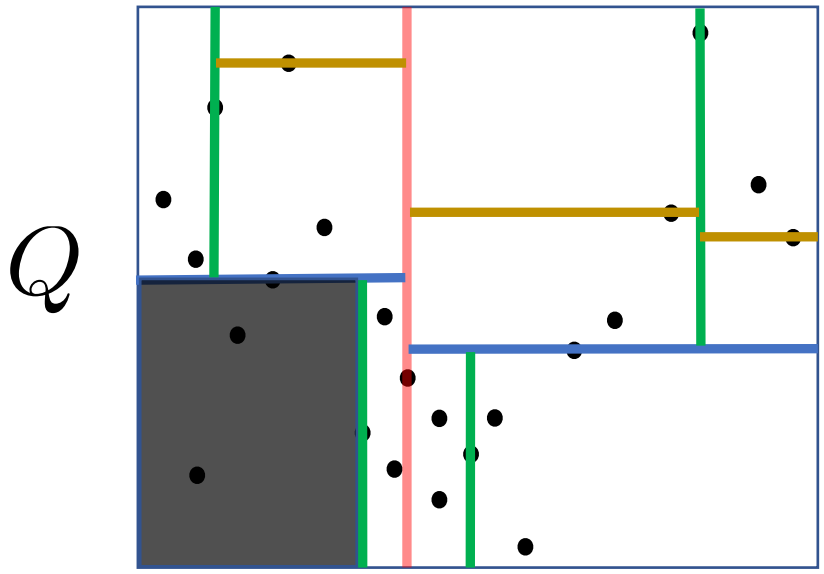
  - Approximate compute

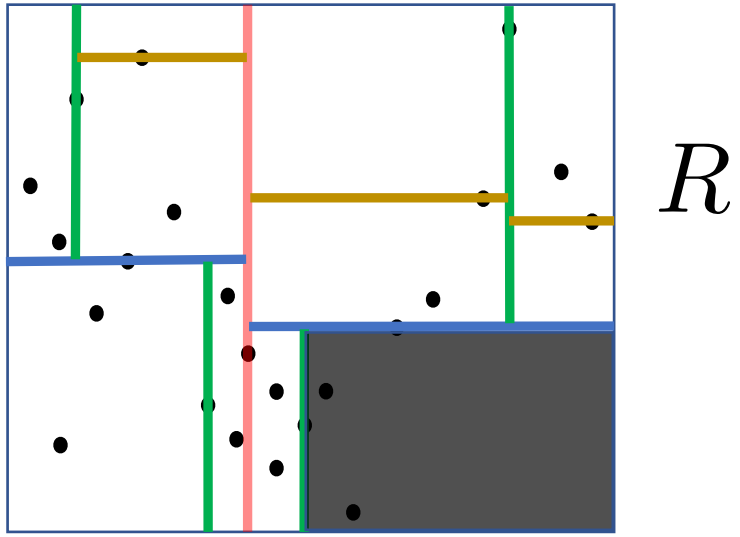
Prune/Approximate Generator

- Kernel Density Estimation (Approximation class)
  - Approximate condition

$$\forall q \in Q, \quad \sum_{r \in R} K_\sigma \left(\frac{\|x_q - x_r\|}{\sigma} \right)$$

  - Approximate compute





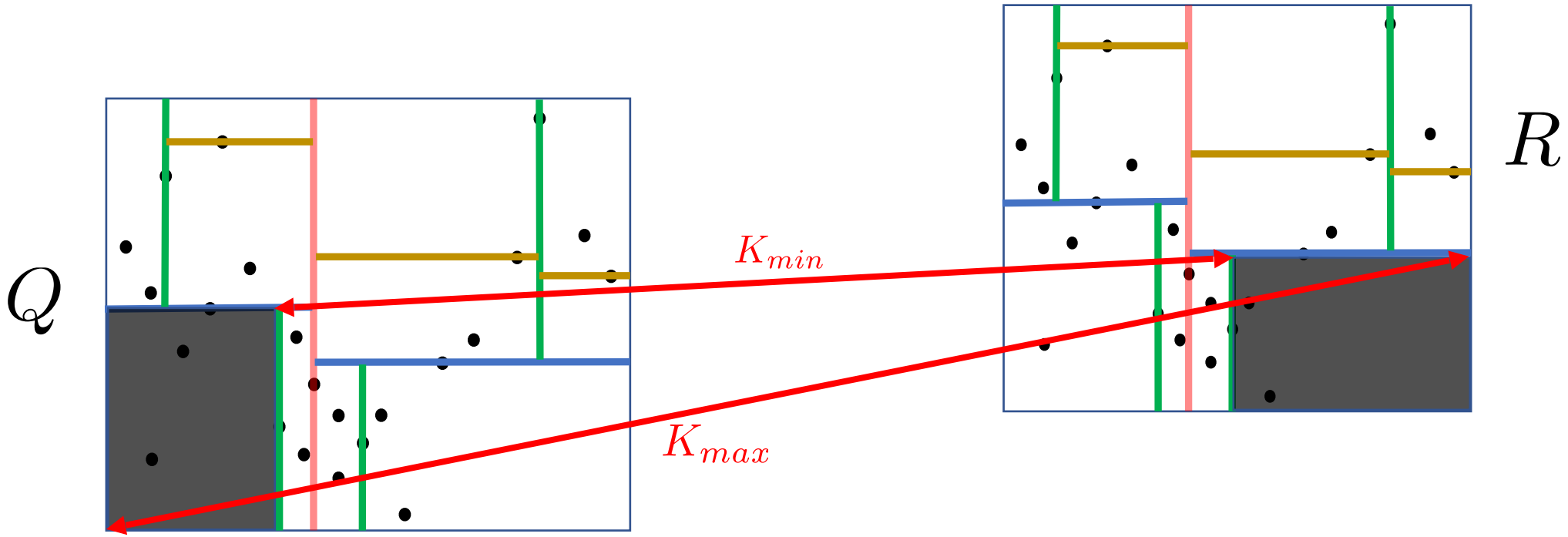
Prune/Approximate Generator

- Kernel Density Estimation (Approximation class)
  - Approximate condition

$$K_{\max} - K_{\min} < \beta \times K_{center}$$

$$\forall q \in Q, \quad \sum_{r \in R} K_\sigma \left(\frac{\|x_q - x_r\|}{\sigma} \right)$$

  - Approximate compute



Prune/Approximate Generator

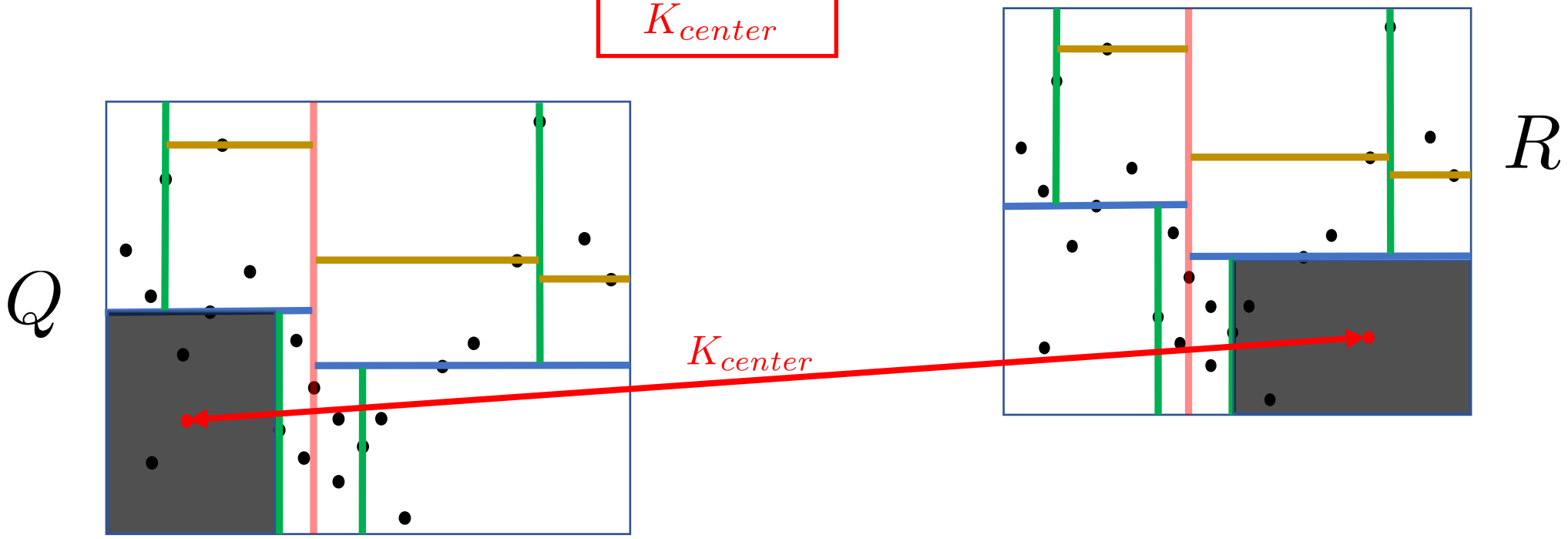
- Kernel Density Estimation (Approximation class)
  - Approximate condition

$$K_{\max} - K_{\min} < \beta \times K_{center}$$

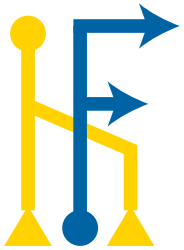
$$\forall q \in Q, \quad \sum_{r \in R} K_\sigma \left(\frac{\|x_q - x_r\|}{\sigma} \right)$$

  - Approximate compute

$$K_{center}$$

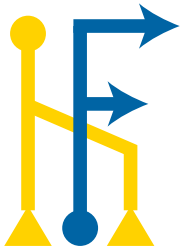


Portal Compiler



Portal Compiler

Nearest Neighbor



Portal Compiler

Nearest Neighbor

Mathematical Equation

$$\forall q \in Q, \ \arg\min_{r \in R} \ \|x_q - x_r\|$$

Portal Compiler

Portal Language

Nearest Neighbor

Mathematical Equation

$$\forall q \in Q, \ \arg\min_{r \in R} \ \|x_q - x_r\|$$

Portal Compiler

Portal Language

Nearest Neighbor

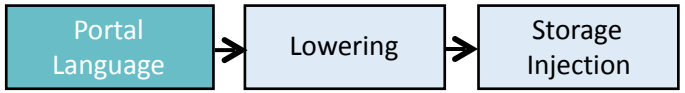
Mathematical Equation

$$\forall q \in Q, \ \arg\min_{r \in R} \ \|x_q - x_r\|$$

Portal Language

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
Var q,r;
Expr EuclidDist = sqrt(pow((q-r), 2));
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, q, query);
expr.addLayer(PortalOp::ARGMIN, r, reference,
EuclidDist);
expr.execute();
Storage output = expr.getOutput();
```

Portal Compiler



Nearest Neighbor

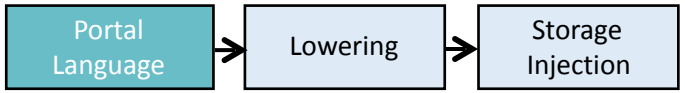
Mathematical Equation

$$\forall q \in Q, \ \arg\min_{r \in R} \ \|x_q - x_r\|$$

Portal Language

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
Var q,r;
Expr EuclidDist = sqrt(pow((q-r), 2));
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, q, query);
expr.addLayer(PortalOp::ARGMIN, r, reference,
EuclidDist);
expr.execute();
Storage output = expr.getOutput();
```

Portal Compiler



Nearest Neighbor

Mathematical Equation

$$\forall q \in Q, \ \arg\min_{r \in R} \ \|x_q - x_r\|$$

Portal Language

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
Var q,r;
Expr EuclidDist = sqrt(pow((q-r), 2));
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, q, query);
expr.addLayer(PortalOp::ARGMIN, r, reference,
EuclidDist);
expr.execute();
Storage output = expr.getOutput();
```

```
/* Storage injection for outer layer */
alloc storage0[q.size]
for q in query.start ... query.end
  /* Storage injection for inner layer */
  alloc storage1 = max numeric limit
  for r in reference.start ...reference.end
    alloc t = 0
    /* Lowering the kernel function */
    for d in 0 ... dim
      t += pow((load(q,d)-load(r,d)),2)
    t = sqrt(t)
    /* Lowering the min functionality */
    storage1 = storage1<t?storage1:t
  storage0[q] = storage1
```

```
/* Prune/Approximate condition for the two tree nodes
N1 (from query) and N2 (from reference) */
for d in 0 ... dim
   t += pow((load(N1min,d)-load(N2min,d)),2)
t = sqrt(t)
return (t>current_min_distance)
```

```
/* Nearest Neighbor is a pruning problem, hence there
is no approximation */
return 0;
```

Portal Compiler

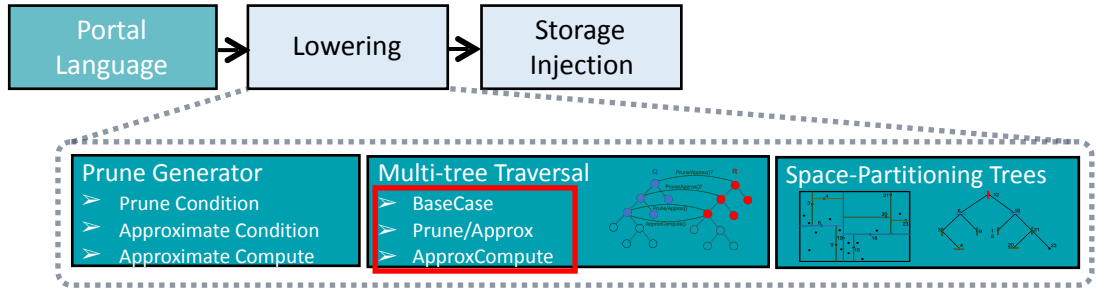
Nearest Neighbor

Mathematical Equation

$$\forall q \in Q, \ \arg\min_{r \in R} \ \|x_q - x_r\|$$

Portal Language

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
Var q,r;
Expr EuclidDist = sqrt(pow((q-r), 2));
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, q, query);
expr.addLayer(PortalOp::ARGMIN, r, reference,
EuclidDist);
expr.execute();
Storage output = expr.getOutput();
```



BaseCase

```
/* Storage injection for outer layer */
alloc storage0[q.size]
for q in query.start ... query.end
  /* Storage injection for inner layer */
  alloc storage1 = max numeric limit
  for r in reference.start ...reference.end
    alloc t = 0
    /* Lowering the kernel function */
    for d in 0 ... dim
      t += pow((load(q,d)-load(r,d)),2)
    t = sqrt(t)
    /* Lowering the min functionality */
    storage1 = storage1<t?storage1:t
  storage0[q] = storage1
```

Prune/Approx

```
/* Prune/Approximate condition for the two tree nodes
N1 (from query) and N2 (from reference) */
for d in 0 ... dim
   t += pow((load(N1min,d)-load(N2min,d)),2)
t = sqrt(t)
return (t>current_min_distance)
```

Approx Compute

```
/* Nearest Neighbor is a pruning problem, hence there
is no approximation */
return 0;
```

Portal Compiler

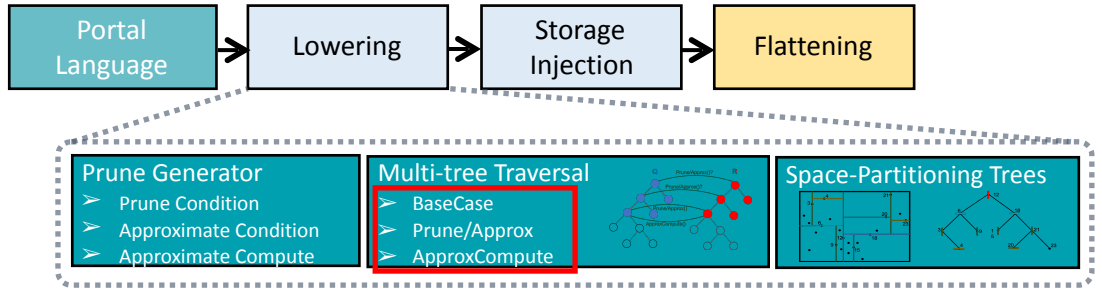
Nearest Neighbor

Mathematical Equation

$$\forall q \in Q, \ \arg\min_{r \in R} \ \|x_q - x_r\|$$

Portal Language

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
Var q,r;
Expr EuclidDist = sqrt(pow((q-r), 2));
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, q, query);
expr.addLayer(PortalOp::ARGMIN, r, reference,
EuclidDist);
expr.execute();
Storage output = expr.getOutput();
```



BaseCase

```
/* Storage injection for outer layer */
alloc storage0[q.size]
for q in query.start ... query.end
  /* Storage injection for inner layer */
  alloc storage1 = max numeric limit
  for r in reference.start ...reference.end
    alloc t = 0
    /* Lowering the kernel function */
    for d in 0 ... dim
      t += pow((load(q,d)-load(r,d)),2)
    t = sqrt(t)
    /* Lowering the min functionality */
    storage1 = storage1<t?storage1:t
  storage0[q] = storage1
```

Prune/Approx

```
/* Prune/Approximate condition for the two tree nodes
N1 (from query) and N2 (from reference) */
for d in 0 ... dim
   t += pow((load(N1min,d)-load(N2min,d)),2)
t = sqrt(t)
return (t>current_min_distance)
```

Approx Compute

```
/* Nearest Neighbor is a pruning problem, hence there
is no approximation */
return 0;
```

Portal Compiler

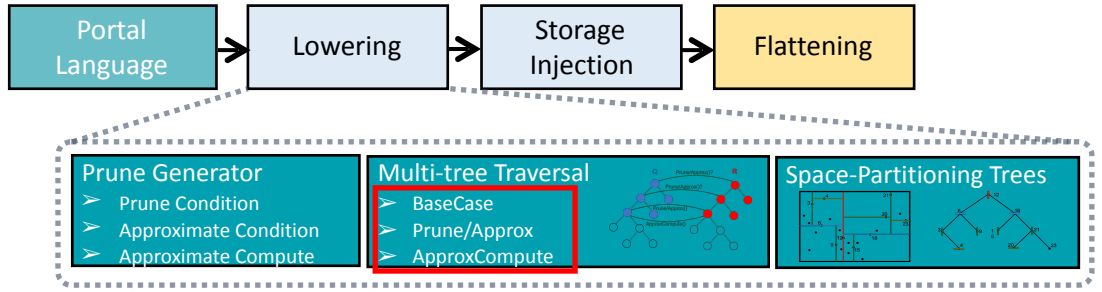
Nearest Neighbor

Mathematical Equation

$$\forall q \in Q, \ \arg\min_{r \in R} \ \|x_q - x_r\|$$

Portal Language

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
Var q,r;
Expr EuclidDist = sqrt(pow((q-r), 2));
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, q, query);
expr.addLayer(PortalOp::ARGMIN, r, reference,
EuclidDist);
expr.execute();
Storage output = expr.getOutput();
```



BaseCase

```
/* Storage injection for outer layer */
alloc storage0[q.size]
for q in query.start ... query.end
  /* Storage injection for inner layer */
  alloc storage1 = max numeric limit
  for r in reference.start ...reference.end
    alloc t = 0
    /* Lowering the kernel function */
    for d in 0 ... dim
      t += pow((load(q,d)-load(r,d)),2)
    t = sqrt(t)
    /* Lowering the min functionality */
    storage1 = storage1<t?storage1:t
  storage0[q] = storage1
```

```
t += pow((load((q×q.stride)-
     (q.start×q.stride)+d)-
     load((r×r.stride)-
     (r.start×r.stride)+d)),2)
```

```
storage0[(q×q.stride)-
(q.start×q.stride)] = storage1
```

Prune/Approx

```
/* Prune/Approximate condition for the two tree nodes
N1 (from query) and N2 (from reference) */
for d in 0 ... dim
   t += pow((load(N1min,d)-load(N2min,d)),2)
t = sqrt(t)
return (t>current_min_distance)
```

```
t += pow((load((N1min×N1min.stride)-
     N1min.start×N1min.stride)+d)-
     load((N2min×N2min.stride)-
    (N2min.start×N2min.stride)+d)),2)
```

Approx Compute

```
/* Nearest Neighbor is a pruning problem, hence there
is no approximation */
return 0;
```

Portal Compiler

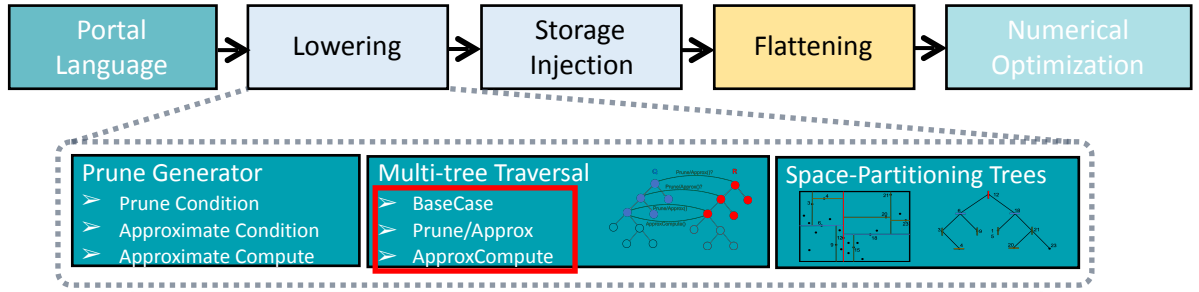
Nearest Neighbor

Mathematical Equation

$$\forall q \in Q, \ \arg\min_{r \in R} \ \|x_q - x_r\|$$

Portal Language

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
Var q,r;
Expr EuclidDist = sqrt(pow((q-r), 2));
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, q, query);
expr.addLayer(PortalOp::ARGMIN, r, reference,
EuclidDist);
expr.execute();
Storage output = expr.getOutput();
```



BaseCase

```
/* Storage injection for outer layer */
alloc storage0[q.size]
for q in query.start ... query.end
  /* Storage injection for inner layer */
  alloc storage1 = max numeric limit
  for r in reference.start ...reference.end
    alloc t = 0
    /* Lowering the kernel function */
    for d in 0 ... dim
      t += pow((load(q,d)-load(r,d)),2)
    t = sqrt(t)
    /* Lowering the min functionality */
    storage1 = storage1<t?storage1:t
  storage0[q] = storage1
```

```
t += pow((load((q×q.stride)-
     (q.start×q.stride)+d)-
     load((r×r.stride)-
     (r.start×r.stride)+d)),2)
```

```
storage0[(q×q.stride)-
(q.start×q.stride)] = storage1
```

Prune/Approx

```
/* Prune/Approximate condition for the two tree nodes
N1 (from query) and N2 (from reference) */
for d in 0 ... dim
   t += pow((load(N1min,d)-load(N2min,d)),2)
t = sqrt(t)
return (t>current_min_distance)
```

```
t += pow((load((N1min×N1min.stride)-
     N1min.start×N1min.stride)+d)-
     load((N2min×N2min.stride)-
    (N2min.start×N2min.stride)+d)),2)
```

Approx Compute

```
/* Nearest Neighbor is a pruning problem, hence there
is no approximation */
return 0;
```

Portal Compiler

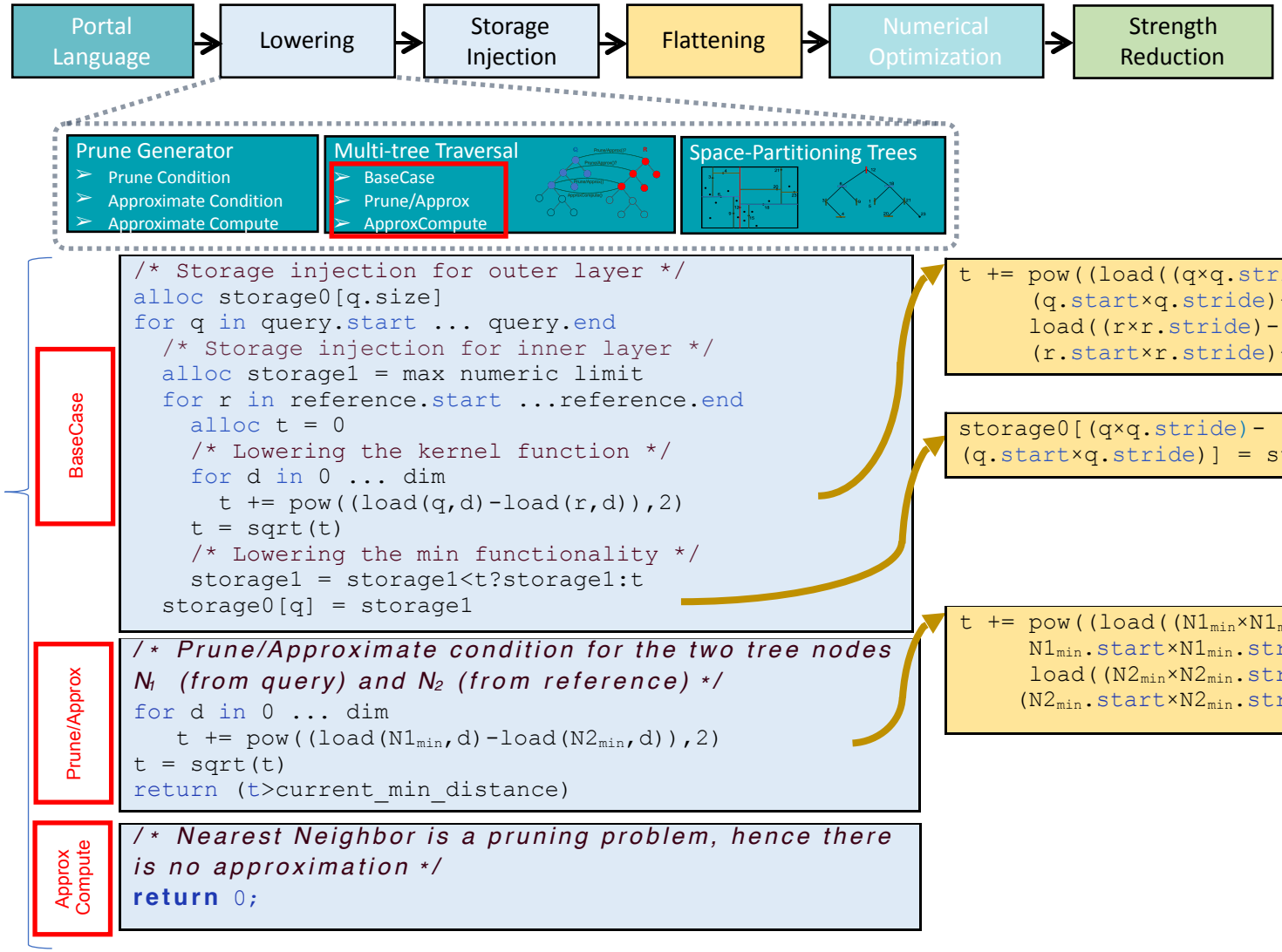
Nearest Neighbor

Mathematical Equation

$$\forall q \in Q, \ \arg\min_{r \in R} \ \|x_q - x_r\|$$

Portal Language

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
Var q,r;
Expr EuclidDist = sqrt(pow((q-r), 2));
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, q, query);
expr.addLayer(PortalOp::ARGMIN, r, reference,
EuclidDist);
expr.execute();
Storage output = expr.getOutput();
```



Portal Compiler

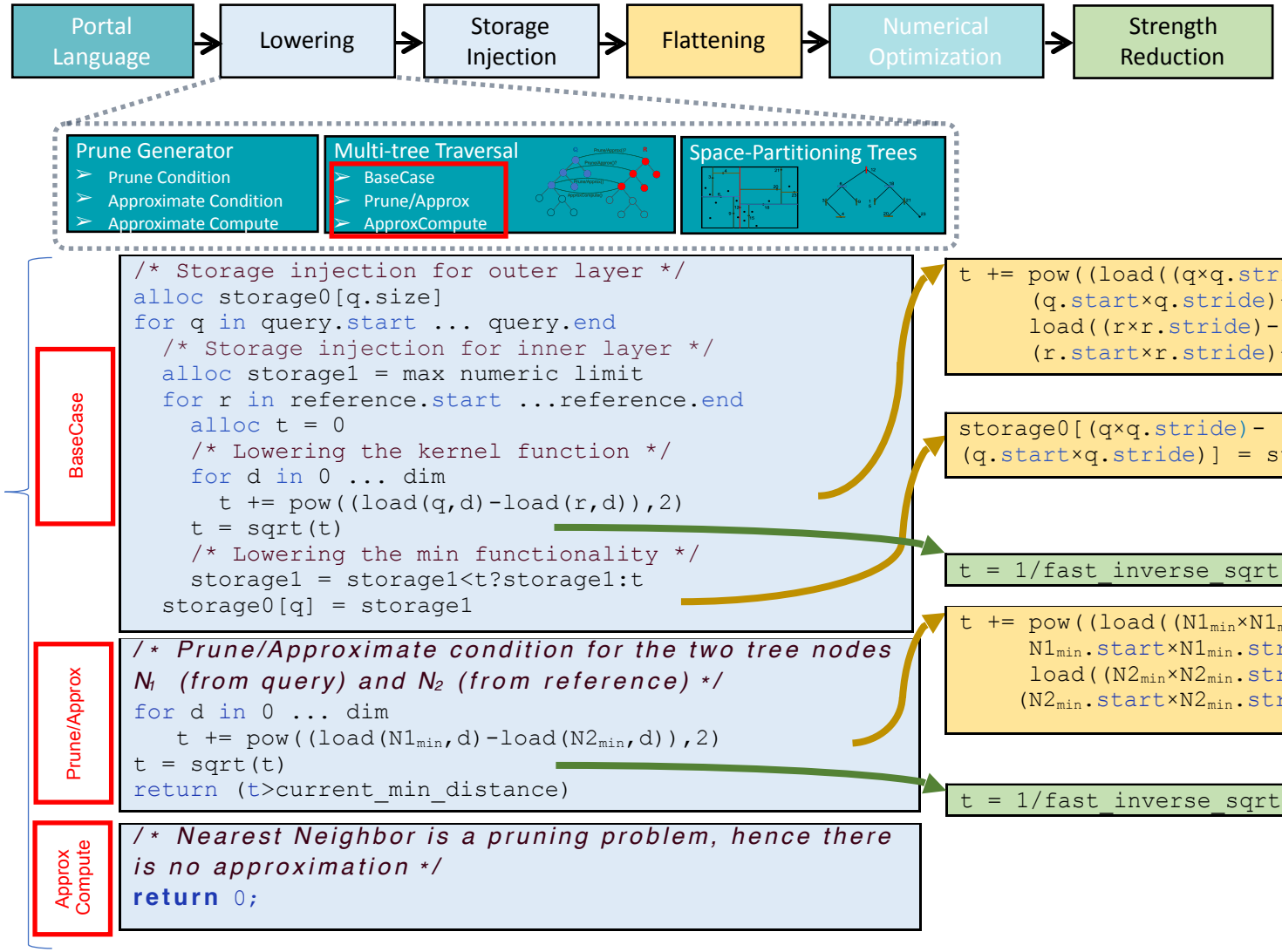
Nearest Neighbor

Mathematical Equation

$$\forall q \in Q, \ \arg\min_{r \in R} \ \|x_q - x_r\|$$

Portal Language

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
Var q,r;
Expr EuclidDist = sqrt(pow((q-r), 2));
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, q, query);
expr.addLayer(PortalOp::ARGMIN, r, reference,
EuclidDist);
expr.execute();
Storage output = expr.getOutput();
```



Portal Compiler

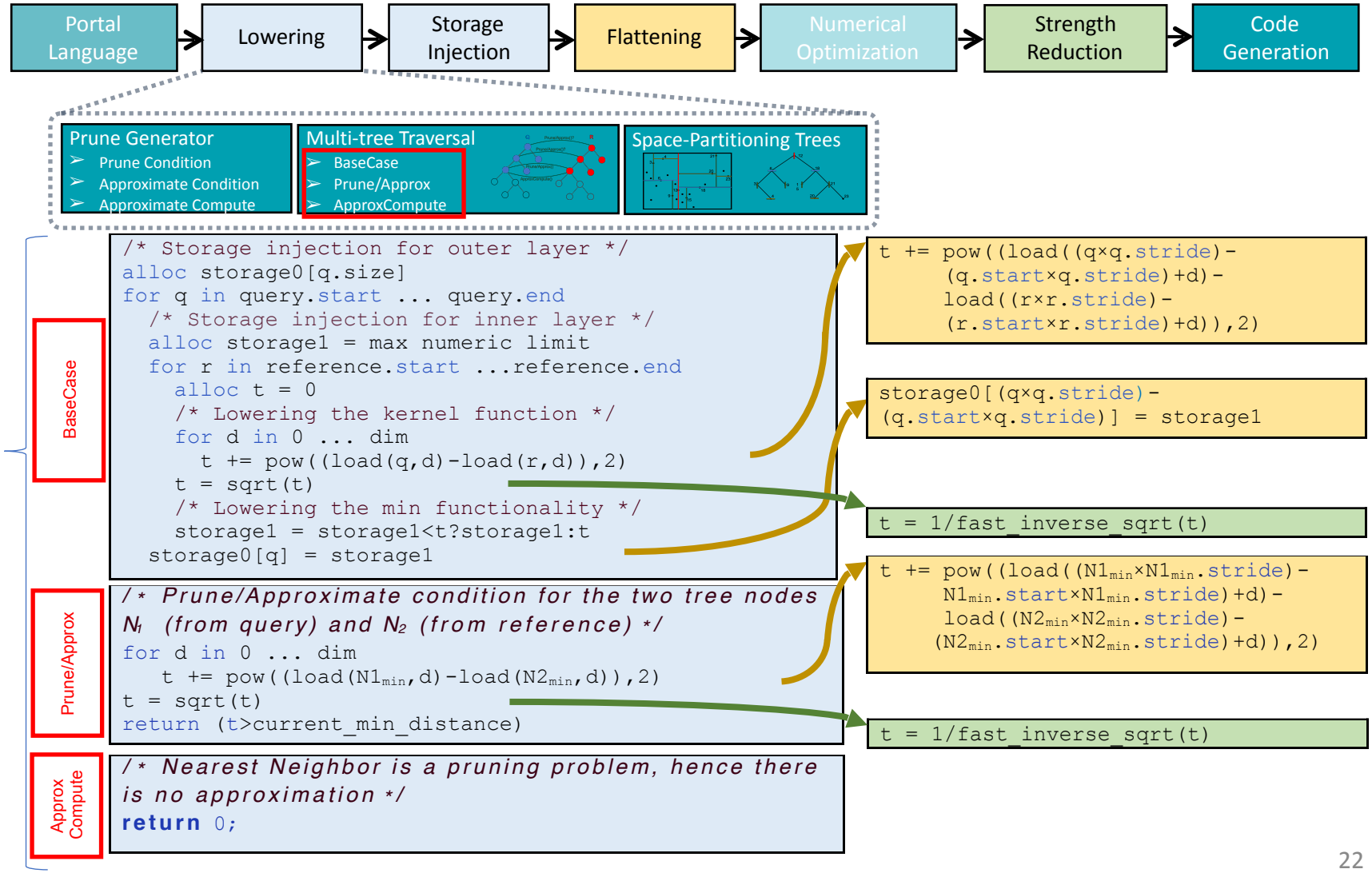
Nearest Neighbor

Mathematical Equation

$$\forall q \in Q, \ \arg\min_{r \in R} \ \|x_q - x_r\|$$

Portal Language

```
Storage query("query_file.csv");
Storage reference("reference_file.csv");
Var q,r;
Expr EuclidDist = sqrt(pow((q-r), 2));
PortalExpr expr;
expr.addLayer(PortalOp::FORALL, q, query);
expr.addLayer(PortalOp::ARGMIN, r, reference,
EuclidDist);
expr.execute();
Storage output = expr.getOutput();
```



Portal Compiler

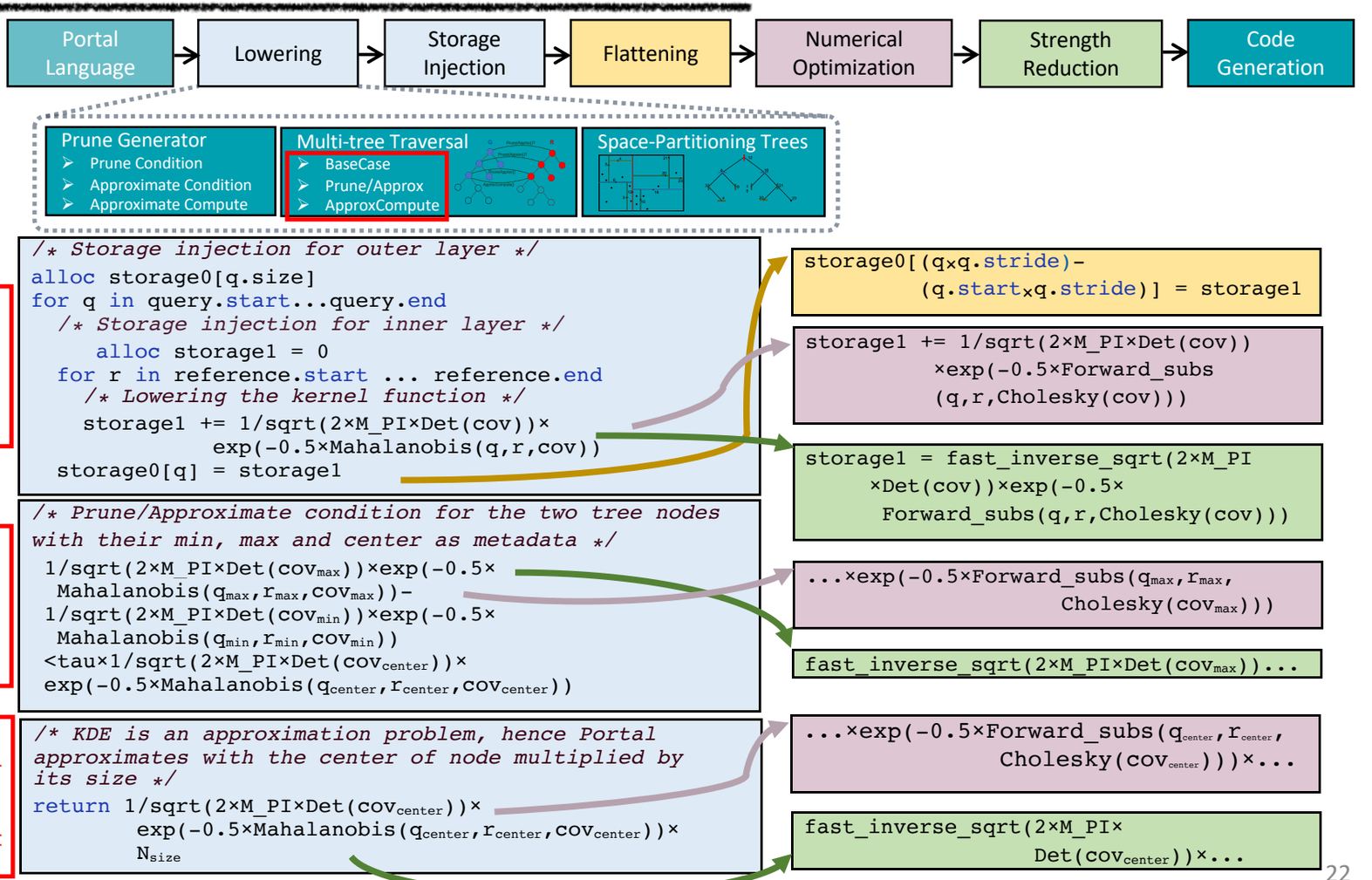
- Kernel Density Estimation

Mathematical Equation

$$\forall q \in Q, \quad \sum_r K_\sigma\left(\frac{\|x_q - x_r\|}{\sigma}\right)$$

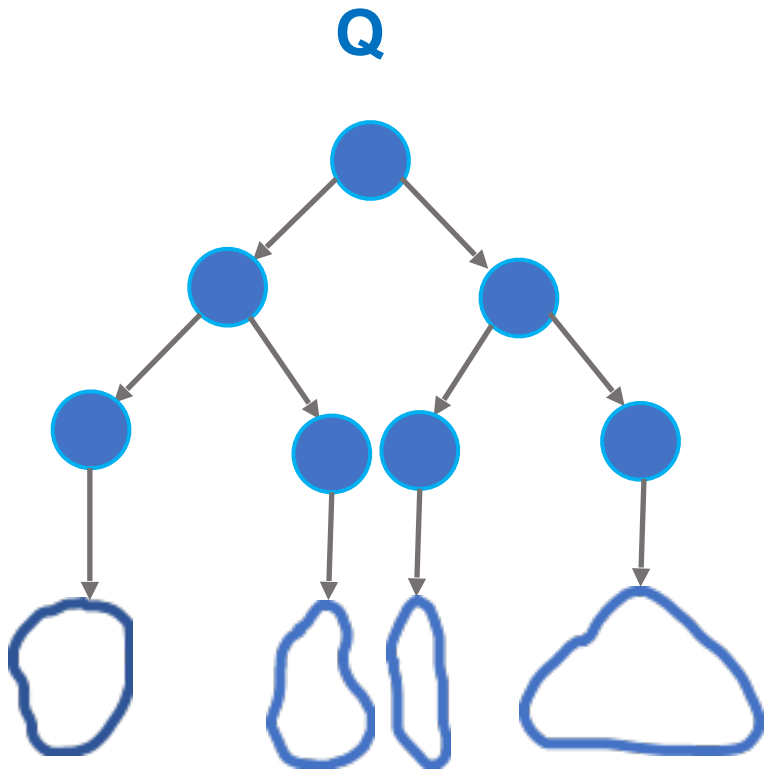
Portal Language

```
Storage query("query_file.csv");
Storage reference("ref_file.csv");
Var q,r,cov;
Expr Kernel = 1/sqrt(2×M_PI×Det(cov))
      ×exp(-0.5×Mahalanobis(q,r,cov));
PortalExpr expr;
expr.addLayer(PortalOp::FORALL,q,query);
expr.addLayer(PortalOp::SUM,r,
reference,Kernel);
expr.execute();
Storage output = expr.getOutput();
```



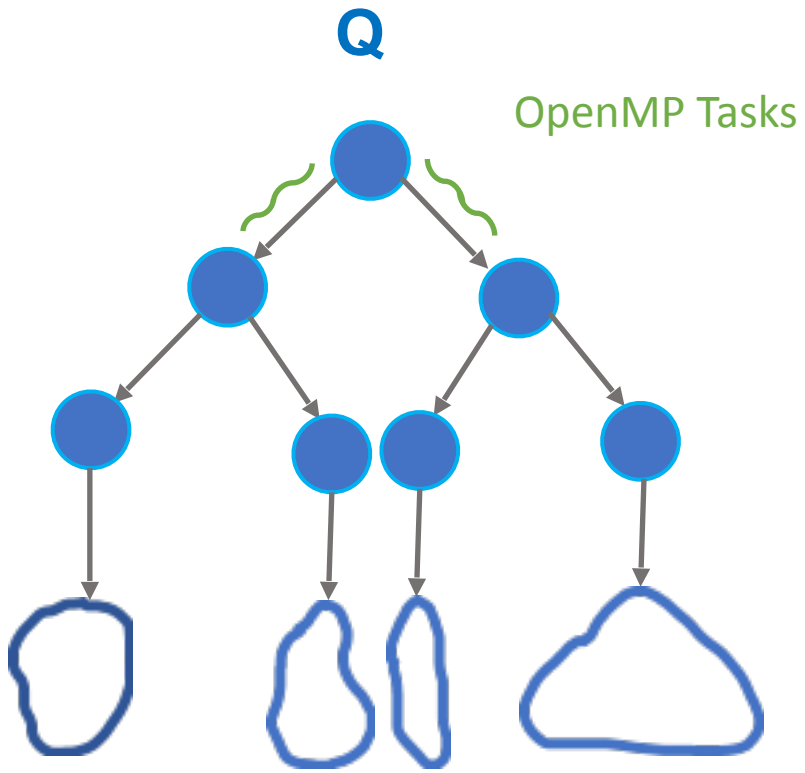
Portal Compiler

- Code Generation
  - Using LLVM for X86
  - Parallelization on multi-tree traversal
    - Using OpenMP



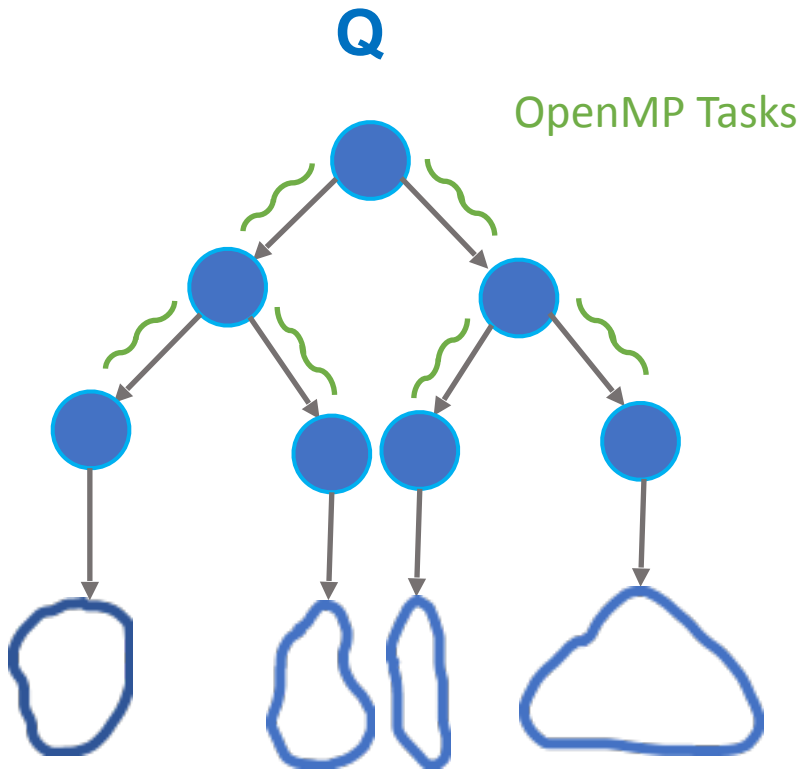
Portal Compiler

- Code Generation
  - Using LLVM for X86
  - Parallelization on multi-tree traversal
    - Using OpenMP



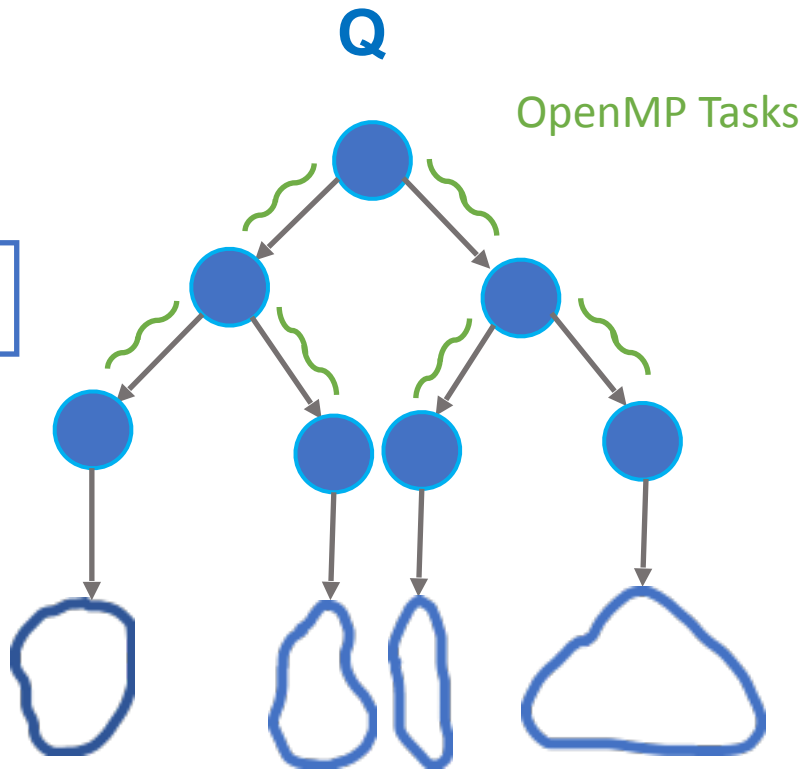
Portal Compiler

- Code Generation
  - Using LLVM for X86
  - Parallelization on multi-tree traversal
    - Using OpenMP



Portal Compiler

- Code Generation
  - Using LLVM for X86
  - Parallelization on multi-tree traversal
    - Using OpenMP



Experimental Setup

- Architecture and compiler
  - Dual-socket AMD EPYC 7551, each socket 64 cores
  - Clang compiler and LLVM version 6.0.0
  - Python v3.7.0 for scikit-learn v0.20.0 and MLPACK 3.0.3
- Benchmarks

Dataset	N	d
Yahoo!	41904293	11
IHEPC	2075259	9
HIGGS	11000000	28
Census	2458285	68
KDD	4898431	42
Elliptical	10000000	3

Case Studies

Portal Performance Test (6 examples)

Problem	Operators	Kernel function
All Nearest Neighbor	$\forall, \arg\min$	$\|\|x_q - x_r\|\|$
All Range Search	$\forall, \cup \arg$	$I\left(h_{\min} < \|\|x_q - x_r\|\| < h_{\max}\right)$
All Range Count	$\forall, \Sigma$	$I\left(h_{\min} < \|\|x_q - x_r\|\| < h_{\max}\right)$
Naïve Bayes Classifier	$\forall, \arg\max$	$(1/\sqrt{2\pi\|\Sigma_k\|})e^{-\frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k)}P(C_k)$
Mixture Model E-step	$\forall, \forall$	$(1/\sqrt{2\pi\|\Sigma_k\|})e^{-\frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k)}$
K-means E-step	$\forall, \arg\min$	$\|\|x_q - x_r\|\|$
Mixture Model Log-likelihood	$\sum, \log\sum$	$(1/\sqrt{2\pi\|\Sigma_k\|})e^{-\frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k)}$
Kernel Density Estimation	$\forall, \Sigma$	$\phi(\frac{\|\|x_q - x_r\|\|}{h})$
Kernel Density Bayes Classifier	$\forall, \arg\max\Sigma$	$\phi(\frac{\|\|x_q - x_r\|\|}{h})P(C_k)$
2-Point Correlation	Σ, Σ	$I(\|\|x_q - x_r\|\| < h)$
Nadaraya-Watson Regression	$\forall, \Sigma$	$y_r\ \phi(\frac{\|\|x_q - x_r\|\|}{h})$
Thermodynamic Average	Σ, Σ	$\phi(\|\|x_q - x_r\|\|)$
Largest-Span Set	$\max, ..., \max$	$\Sigma(\|\|x_q - x_r\|\|)$
Closet Pair	$\max, ..., \max$	$\|\|x_q - x_r\|\|$
Minimum Spanning Tree	$\forall, \arg\min$	$\|\|x_q - x_r\|\|$
Gravitation Interaction (Barnes-Hut)	$\forall, \Sigma$	$\frac{\alpha_q\alpha_r}{\|\|x_q - x_r\|\|}$
Average Density	Σ, Σ	$I(\|\|x_q - x_r\|\| < h)$
Wave Function	$\forall, \Pi$	$\phi(\|\|x_q - x_r\|\|)$
Hausdorff Distance	$\max, \min$	$\|\|x_q - x_r\|\|$
Intrinsic (Fractional) Dimension	Σ, Σ	$I(\|\|x_q - x_r\|\| < h)$

Portal Validation Test (3 examples)

Portal Performance

	Census			Yahoo!			IHEPC			HIGGS			KDD		
	Expert	Portal	% Diff	Expert	Portal	% Diff	Expert	Portal	% Diff	Expert	Portal	% Diff	Expert	Portal	% Diff
K-NN	22.8	23.9	4	84.6	85.2	2	8.7	9.1	4	186	191	3	21.4	22.6	5
KDE	1087	1129	3	133.7	139.5	4	39.2	41.7	6	411.9	430.9	4	926.5	949	2
RS	42.2	44.4	5	214.5	223.1	4	15	16.1	7	130.1	130.1	6	20.1	21.1	4
MST	374.1	391.7	4	918.4	946.1	3	200.8	211	5	478.3	486.2	2	273.6	281	3
EM	76.3	82.6	8	224.5	242.8	8	78.6	85.3	8	198.8	216.7	9	32.4	35.3	8
HD	40.9	43.1	5	122.8	129.7	5	38.4	40.1	4	236.6	243.8	3	36.2	38.3	5

LOC		
Expert	Portal	X shorter
867	13	67
626	15	42
673	13	52
956	54	17
1681	104	16
689	13	53

Times in second

K-NN: K-Nearest Neighbor KDE: Kernel Density Estimation. RS: Range Search
MST: Minimum Spanning Tree EM: Expectation Maximization HD: Hausdorff distance

Portal Performance

Datasets

	Census			Yahoo!			IHEPC			HIGGS			KDD		
	Expert	Portal	% Diff	Expert	Portal	% Diff	Expert	Portal	% Diff	Expert	Portal	% Diff	Expert	Portal	% Diff
K-NN	22.8	23.9	4	84.6	85.2	2	8.7	9.1	4	186	191	3	21.4	22.6	5
KDE	1087	1129	3	133.7	139.5	4	39.2	41.7	6	411.9	430.9	4	926.5	949	2
RS	42.2	44.4	5	214.5	223.1	4	15	16.1	7	130.1	130.1	6	20.1	21.1	4
MST	374.1	391.7	4	918.4	946.1	3	200.8	211	5	478.3	486.2	2	273.6	281	3
EM	76.3	82.6	8	224.5	242.8	8	78.6	85.3	8	198.8	216.7	9	32.4	35.3	8
HD	40.9	43.1	5	122.8	129.7	5	38.4	40.1	4	236.6	243.8	3	36.2	38.3	5

LOC		
Expert	Portal	X shorter
867	13	67
626	15	42
673	13	52
956	54	17
1681	104	16
689	13	53

Times in second

K-NN: K-Nearest Neighbor KDE: Kernel Density Estimation. RS: Range Search
MST: Minimum Spanning Tree EM: Expectation Maximization HD: Hausdorff distance

Portal Performance

Datasets

	Census			Yahoo!			IHEPC			HIGGS			KDD		
	Expert	Portal	% Diff	Expert	Portal	% Diff	Expert	Portal	% Diff	Expert	Portal	% Diff	Expert	Portal	% Diff
K-NN	22.8	23.9	4	84.6	85.2	2	8.7	9.1	4	186	191	3	21.4	22.6	5
KDE	1087	1129	3	133.7	139.5	4	39.2	41.7	6	411.9	430.9	4	926.5	949	2
RS	42.2	44.4	5	214.5	223.1	4	15	16.1	7	130.1	130.1	6	20.1	21.1	4
MST	374.1	391.7	4	918.4	946.1	3	200.8	211	5	478.3	486.2	2	273.6	281	3
EM	76.3	82.6	8	224.5	242.8	8	78.6	85.3	8	198.8	216.7	9	32.4	35.3	8
HD	40.9	43.1	5	122.8	129.7	5	38.4	40.1	4	236.6	243.8	3	36.2	38.3	5

LOC		
Expert	Portal	X shorter
867	13	67
626	15	42
673	13	52
956	54	17
1681	104	16
689	13	53

Algorithms

K-NN: K-Nearest Neighbor KDE: Kernel Density Estimation. RS: Range Search
MST: Minimum Spanning Tree EM: Expectation Maximization HD: Hausdorff distance

Times in second

Portal Performance

Datasets

	Census			Yahoo!			IHEPC			HIGGS			KDD		
	Expert	Portal	% Diff	Expert	Portal	% Diff	Expert	Portal	% Diff	Expert	Portal	% Diff	Expert	Portal	% Diff
K-NN	22.8	23.9	4	84.6	85.2	2	8.7	9.1	4	186	191	3	21.4	22.6	5
KDE	1087	1129	3	133.7	139.5	4	39.2	41.7	6	411.9	430.9	4	926.5	949	2
RS	42.2	44.4	5	214.5	223.1	4	15	16.1	7	130.1	130.1	6	20.1	21.1	4
MST	374.1	391.7	4	918.4	946.1	3	200.8	211	5	478.3	486.2	2	273.6	281	3
EM	76.3	82.6	8	224.5	242.8	8	78.6	85.3	8	198.8	216.7	9	32.4	35.3	8
HD	40.9	43.1	5	122.8	129.7	5	38.4	40.1	4	236.6	243.8	3	36.2	38.3	5

LOC		
Expert	Portal	X shorter
867	13	67
626	15	42
673	13	52
956	54	17
1681	104	16
689	13	53

Algorithms

K-NN: K-Nearest Neighbor KDE: Kernel Density Estimation. RS: Range Search
MST: Minimum Spanning Tree EM: Expectation Maximization HD: Hausdorff distance

Times in second

Portal Performance

Datasets

Line-Of-Code

	Census			Yahoo!			IHEPC			HIGGS			KDD		
	Expert	Portal	% Diff	Expert	Portal	% Diff	Expert	Portal	% Diff	Expert	Portal	% Diff	Expert	Portal	% Diff
K-NN	22.8	23.9	4	84.6	85.2	2	8.7	9.1	4	186	191	3	21.4	22.6	5
KDE	1087	1129	3	133.7	139.5	4	39.2	41.7	6	411.9	430.9	4	926.5	949	2
RS	42.2	44.4	5	214.5	223.1	4	15	16.1	7	130.1	130.1	6	20.1	21.1	4
MST	374.1	391.7	4	918.4	946.1	3	200.8	211	5	478.3	486.2	2	273.6	281	3
EM	76.3	82.6	8	224.5	242.8	8	78.6	85.3	8	198.8	216.7	9	32.4	35.3	8
HD	40.9	43.1	5	122.8	129.7	5	38.4	40.1	4	236.6	243.8	3	36.2	38.3	5

LOC		
Expert	Portal	X shorter
867	13	67
626	15	42
673	13	52
956	54	17
1681	104	16
689	13	53

Algorithms

K-NN: K-Nearest Neighbor KDE: Kernel Density Estimation. RS: Range Search
MST: Minimum Spanning Tree EM: Expectation Maximization HD: Hausdorff distance

Times in second

Portal Performance

Datasets

Line-Of-Code

	Census			Yahoo!			IHEPC			HIGGS			KDD		
	Expert	Portal	% Diff	Expert	Portal	% Diff	Expert	Portal	% Diff	Expert	Portal	% Diff	Expert	Portal	% Diff
K-NN	22.8	23.9	4	84.6	85.2	2	8.7	9.1	4	186	191	3	21.4	22.6	5
KDE	1087	1129	3	133.7	139.5	4	39.2	41.7	6	411.9	430.9	4	926.5	949	2
RS	42.2	44.4	5	214.5	223.1	4	15	16.1	7	130.1	130.1	6	20.1	21.1	4
MST	374.1	391.7	4	918.4	946.1	3	200.8	211	5	478.3	486.2	2	273.6	281	3
EM	76.3	82.6	8	224.5	242.8	8	78.6	85.3	8	198.8	216.7	9	32.4	35.3	8
HD	40.9	43.1	5	122.8	129.7	5	38.4	40.1	4	236.6	243.8	3	36.2	38.3	5

LOC		
Expert	Portal	X shorter
867	13	67
626	15	42
673	13	52
956	54	17
1681	104	16
689	13	53

Algorithms

K-NN: K-Nearest Neighbor KDE: Kernel Density Estimation. RS: Range Search
MST: Minimum Spanning Tree EM: Expectation Maximization HD: Hausdorff distance

Times in second

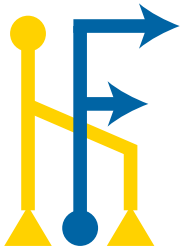
Portal Validation

Portal Validation

- Compare against State-of-the-art (S-O-A)

Portal Validation

- Compare against State-of-the-art (S-O-A)
  - 2-point correlation (2-PC): MLPACK [JMLR 2014]



Portal Validation

- Compare against State-of-the-art (S-O-A)
  - 2-point correlation (2-PC): MLPACK [JMLR 2014]
  - Naïve Bayes Classifier (NBC): Scikit-learn [JMLR 2011]

Portal Validation

- Compare against State-of-the-art (S-O-A)
  - 2-point correlation (2-PC): MLPACK [JMLR 2014]
  - Naïve Bayes Classifier (NBC): Scikit-learn [JMLR 2011]
  - Barnes-Hut (BH): FDPS[PASJ 2016]

Portal Validation

- Compare against State-of-the-art (S-O-A)
  - 2-point correlation (2-PC): MLPACK [JMLR 2014]
  - Naïve Bayes Classifier (NBC): Scikit-learn [JMLR 2011]
  - Barnes-Hut (BH): FDPS[PASJ 2016]

	Census			Yahoo!			IHEPC			HIGGS			KDD			Elliptical		
	S-O-A	Portal	Speedup	S-O-A	Portal	Speedup	S-O-A	Portal	Speedup	S-O-A	Portal	Speedup	S-O-A	Portal	Speedup	S-O-A	Portal	Speedup
2-PC	3529	53	66	37043	250	148	4281	26	162	17823	151	117	5134	31	165	5412	94	57
NBC	1337	87	15	3629	198	18	1699	88	19	5231	261	20	981	47	21	1026	194	5
BH	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	473	278	1.7

Times in second

Portal Validation

- Compare against State-of-the-art (S-O-A)
  - 2-point correlation (2-PC): MLPACK [JMLR 2014]
  - Naïve Bayes Classifier (NBC): Scikit-learn [JMLR 2011]
  - Barnes-Hut (BH): FDPS[PASJ 2016]

	Census			Yahoo!			IHEPC			HIGGS			KDD			Elliptical		
	S-O-A	Portal	Speedup	S-O-A	Portal	Speedup	S-O-A	Portal	Speedup	S-O-A	Portal	Speedup	S-O-A	Portal	Speedup	S-O-A	Portal	Speedup
2-PC	3529	53	66	37043	250	148	4281	26	162	17823	151	117	5134	31	165	5412	94	57
NBC	1337	87	15	3629	198	18	1699	88	19	5231	261	20	981	47	21	1026	194	5
BH	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	473	278	1.7

Times in second

Summary

- Portal
  - Portal Language
    - Inspired by mathematical formulation
    - Chaining layers
    - Reduces line of code
  - Portal Compiler
    - Provides asymptotically optimal tree-based algorithms → *O(N log N) & O(N)*
    - Applies domain-specific optimizations
    - Easily extensible
    - Similar performance to hand-tuned expert code
- Portal Validation
  - Validated on a variety of problems
- Portal is open-source! [https://gitlab.com/Nbody-Portal/Code]

Thank you!

Portal Compiler

- Numerical Optimizations
  - Mahalanobis distance

$$\mathcal{N}(x_i|\theta_k) = \frac{1}{\sqrt{2\pi\,|\Sigma_k|}} e^{-\frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k)}$$

  - Cholesky decomposition($\Sigma = LL^T$)

$$(x_q-\mu_r)^T\Sigma^{-1}(x_q-\mu_r) = (x_q-\mu_r)^T(LL^T)^{-1}(x_q-\mu_r)$$

  - Forward substitution

$$Y = x_i - \mu_j$$

$$Y^T\left(LL^T\right)^{-1}Y = \left(L^{-1}Y\right)^T L^{-1}Y$$

Why a Compiler?

- There are hundreds of N-body problems and its practically impossible to generate hand-optimized code for every single of them
- Hand-tuning is tedious and highly machine-specific
- Domain scientist expert in particular domain but not in parallel computing
- Scientist prefer to program in a high-level language which allows concise expression of their problem