Nearest Neighbor Search for Scientific Computing

Lecture 1

Foundations of Proximity Search

Jaehoon Chung

Research Fellow, Center for AI and Natural Sciences
Korea Institute for Advanced Study

sk7755@kias.re.kr

Environment Setup

Environment Setup

- **Use the shared conda environment** (conda, mamba, ...)

```
conda env create -f environment.yml
conda activate nns
```

- **If conda is not available**
  Use any Python `3.11` environment and install the core packages manually:

```
python -m pip install numpy scipy pandas matplotlib jupyterlab
```

- **Prepare datasets**

```
python data/build_data.py
```

- **GPU is optional** Most practices run on CPU. GPU-specific cells require CUDA/CuPy and can be skipped on machines without a CUDA GPU.

- **Recommended platform**
  This course assumes a Linux environment.

- **Code**

https://drive.google.com/file/d/1W24WxT1t49b5Vy0JxuAFCTKHdlKyoLX0/view?usp=sharing

Environment Setup

Environment Details

Item	Expected setup
Python	`3.11`
Package channel	`conda-forge`
Core stack	`numpy`, `scipy`, `pandas`, `matplotlib`, `jupyterlab`
Tree / metric experiments	course package `algorithms/`
Astronomy backend	`astropy`, `healpy`, `astropy-healpix`
Single-cell backend	`h5py`, `pynndescent`, `leidenalg`, `python-igraph`
GPU backend	optional: `cupy`
ANN backends	optional: `hnswlib`, `faiss-cpu`, `python-annoy`

- Start from `README.md`.
- Each practice notebook begins with setup notes and sanity checks.
- Optional GPU/ANN backends can be skipped unless explicitly needed.
- Please avoid using the shared `LEXICON` server during the lecture; some practices can be compute- or memory-intensive.

Contents

Lecture 1: Foundations of Proximity Search

- **What proximity means in science**
  From distance to relevance.
- **Nearest-Neighbor Problems**
  Formalize local relations as kNN queries, kNN graphs, and radius joins.
- **Algorithms and Analysis**
  What is an algorithm, how do we analyze its cost, and what do experiments measure on actual hardware?
- **Brute Force Baseline**
  Compute all pairwise distances, then select or filter the desired neighbors.

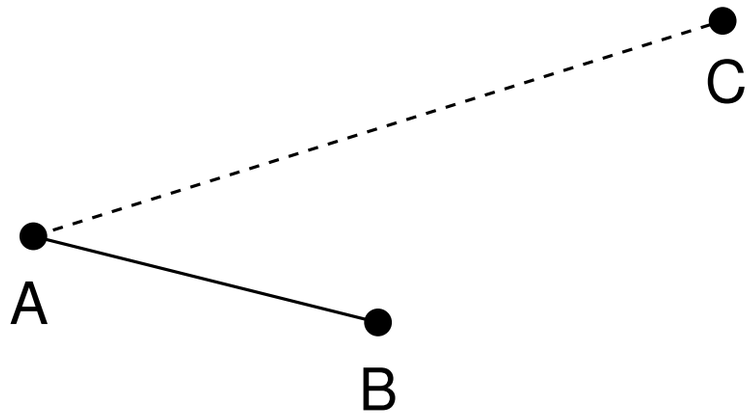
What proximity means in science

From Distance to Proximity

- Distance assigns values to pairs, such as $d(A, B)$ and $d(A, C)$.
- Comparing these values induces an order: $d(A, B) < d(A, C)$.
- Proximity ranks relations:

$$A-B \quad \text{before} \quad A-C$$

- Order is not yet scientific meaning.



B is closer to A than C

What proximity means in science

Proximity as Scientific Relevance

- We do not search for neighbors only to rank distances.
- We expect nearby objects to be more relevant to the question.
- For a scientific question Q:

$$d(A,B) < d(A,C) \quad \overset{\text{expect}}{\rightsquigarrow} \quad R_Q(A,B) > R_Q(A,C)$$

*$R_Q(A,B)$: relevance of A and B for scientific question Q.

- Proximity is used as a signal of relevance.

What proximity means in science

Proximity as Scientific Relevance

- We do not search for neighbors only to rank distances.
- We expect nearby objects to be more relevant to the question.
- For a scientific question Q:

$$d(A,B) < d(A,C) \quad \overset{\text{expect}}{\rightsquigarrow} \quad R_Q(A,B) > R_Q(A,C)$$

*$R_Q(A,B)$: relevance of A and B for scientific question Q.

- Proximity is used as a signal of relevance.

Domain	Question Q	Relevance $R_Q(A,B)$
Molecular dynamics	Will they interact?	interaction strength
Astronomy	Same object?	source-match probability
Single-cell genomics	Same cell state?	biological-state similarity
Information retrieval	Same meaning?	semantic relevance

What proximity means in science

Nearest Is Not Yet Meaningful

- Once a distance is defined, every finite dataset has a nearest neighbor.
- Even if distance is intended to reflect scientific relevance,

$$d(A,B) < d(A,C) \quad \rightsquigarrow \quad R_Q(A,B) > R_Q(A,C),$$

  the nearest neighbor is still only a distance rank.
- We must distinguish two statements:

$$B \text{ is nearest to } A \quad \neq \quad B \text{ is meaningfully near to } A.$$

When is a nearest neighbor not meaningful?

What proximity means in science

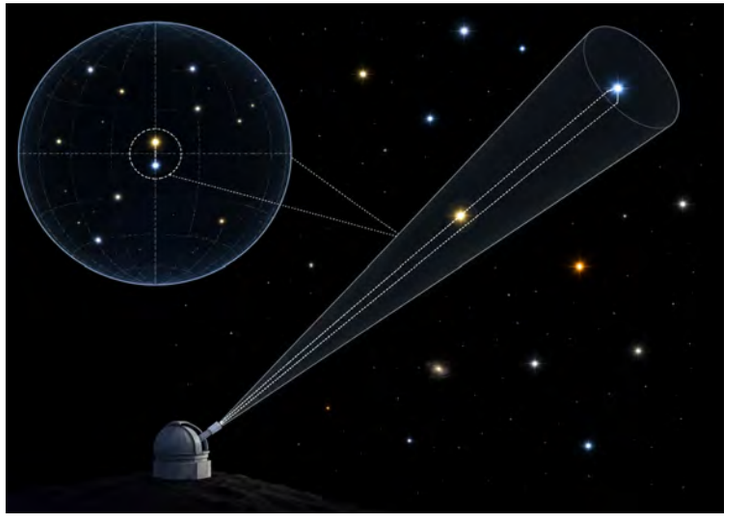
Let B be the nearest neighbor of A. Nearness can usually fail in two ways.

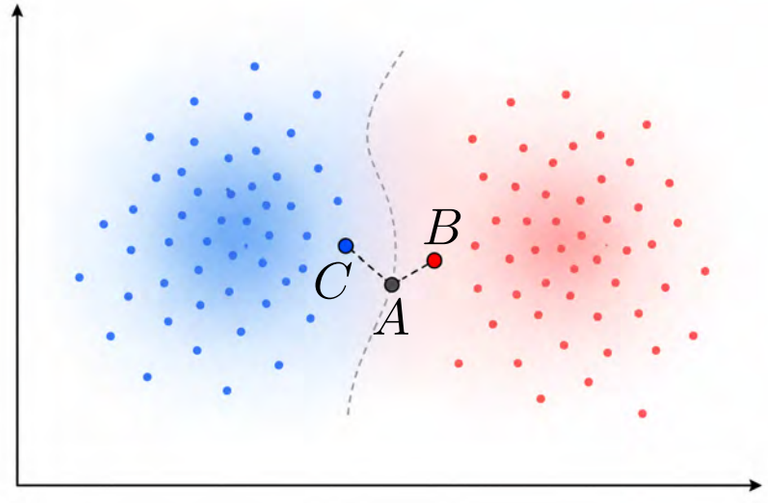
- **No relevant neighbor.**
  B is nearest, but still irrelevant to the question:

$$R_Q(A, B) \approx 0.$$

- **No reliable distinction.**
  B is nearest, but not clearly more relevant than alternatives:

$$R_Q(A, B) \not\gg R_Q(A, C).$$





What proximity means in science

Why Seek Meaningful Proximity in Science?

- To understand A through its relevant neighbors.
- A meaningful neighbor enables local inference:

$$\text{knowledge about } B \quad \rightsquigarrow \quad \text{understanding of } A.$$

Domain	Local inference from a neighbor
Single-cell genomics	infer the cell type of A
Astronomy	identify the source corresponding to A
Medicine	predict the response of patient A
Molecular dynamics	compute the force on particle A
Information retrieval	interpret the meaning of document A
...	...

What proximity means in science

From Scientific Relevance to Search

- Scientific relevance R_Q is not directly searchable.
- We choose a representation and a distance so that relevance appears as proximity.
- This is the modeling step behind nearest-neighbor search.

scientific question Q

↓

representation

↓

distance or similarity

↓

nearest-neighbor search

Nearness in this space is assumed to reflect scientific relevance.

Nearest-Neighbor Problems

Euclidean Space as a Working Model

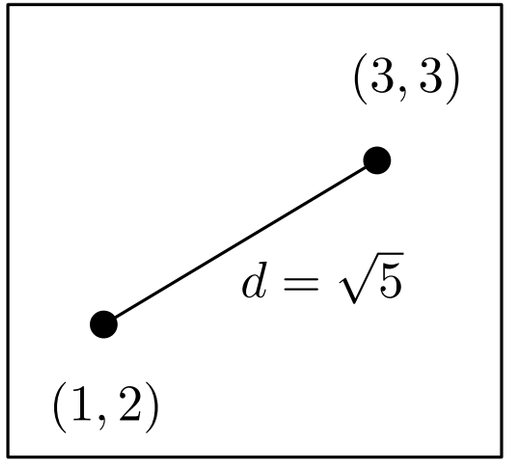
- We represent data as points:

$$P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^d.$$

- For now, we use Euclidean distance:

$$d(p, q) = \|p - q\|_2.$$

- The goal is to find local relations induced by this distance.



Setting	Space	Distance
Euclidean data	$\mathbb{R}^d$	ℓ_2 distance
Astronomy	sphere	angular distance
Molecular dynamics	periodic box	min-image Euclidean distance
Single-cell data	latent space	Euclidean / graph distance
Text embeddings	vector space	angular distance

Nearest-Neighbor Problems

Euclidean Space as a Working Model

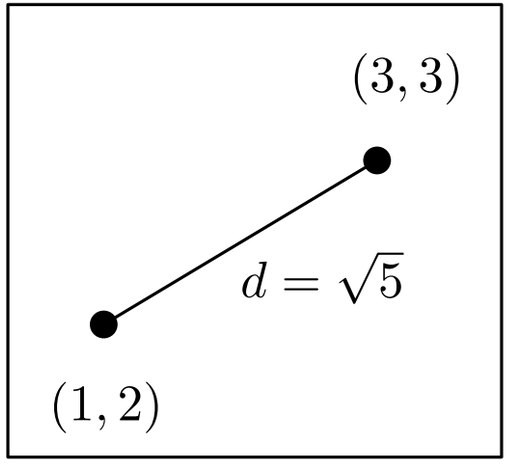
- We represent data as points:

$$P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^d.$$

- For now, we use Euclidean distance:

$$d(p, q) = \|p - q\|_2.$$

- The goal is to find local relations induced by this distance.

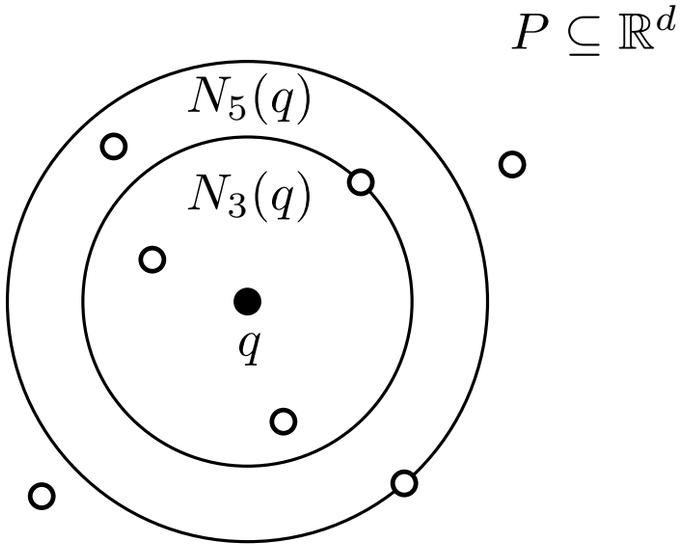


Setting	Space	Distance
Euclidean data	$\mathbb{R}^d$	ℓ_2 distance
Astronomy	sphere	angular distance
Molecular dynamics	periodic box	min-image Euclidean distance
Single-cell data	latent space	Euclidean / graph distance
Text embeddings	vector space	angular distance

Many cases can be reduced to this working model.

Nearest-Neighbor Problems

- Dataset: $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^d$
- Distance: $d(p, q) = \|p - q\|_2$
- $N_k(q; P)$: the k nearest points to q in P

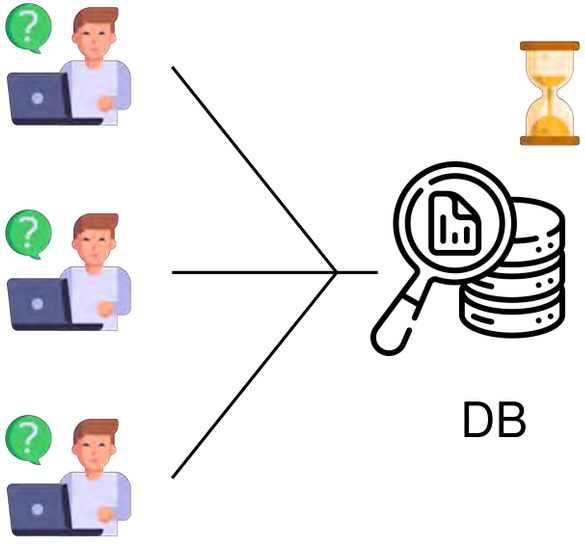


Problem	Input	Output
kNN query	query q	$N_k(q; P)$
kNN graph	dataset P	$N_k(p; P \setminus \{p\})$ for all $p \in P$
Radius join	radius r	all disjoint pairs with $d(p_i, p_j) \leq r$

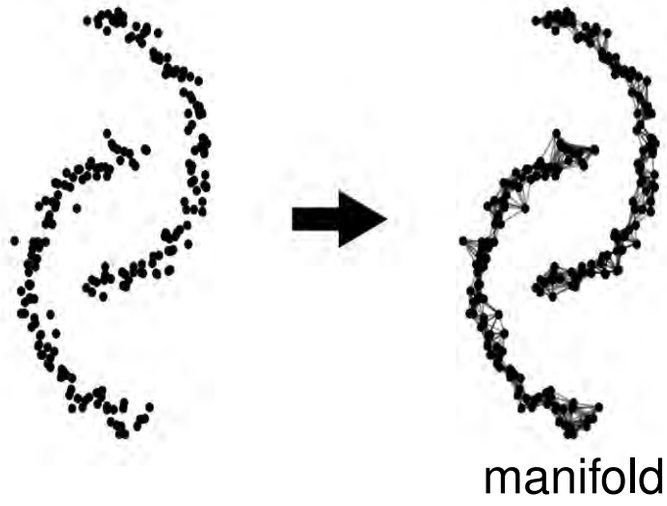
Nearest-Neighbor Problems

Different forms of the same message: finding relevant local relations.

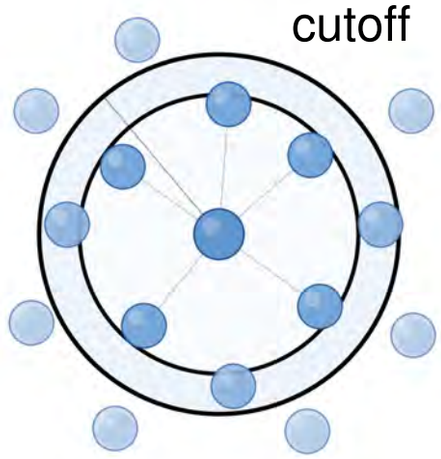
Problem	Input	Output
kNN query	query q	$N_k(q; P)$
kNN graph	dataset P	$N_k(p; P \setminus \{p\})$ for all $p \in P$
Radius join	radius r	all pairs with $d(p_i, p_j) \leq r$



kNN-queries



kNN-graph

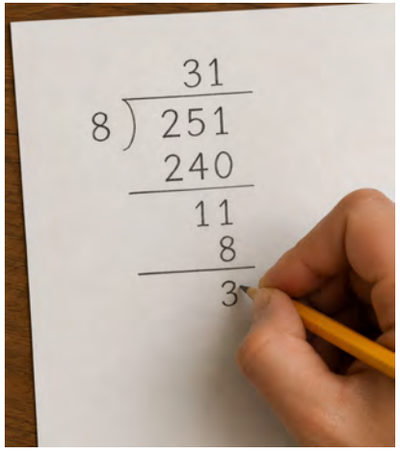


Radius join

Algorithms and Analysis

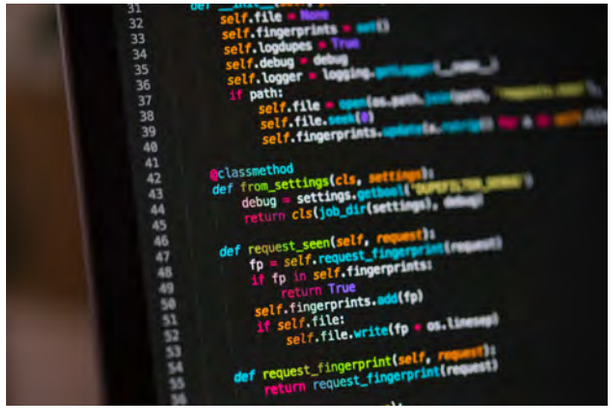
What Is an Algorithm?

- A problem specifies what output is required for each valid input.
- An algorithm specifies how to produce that output.
- It is a finitely described, unambiguous squence of effective steps.
- For every valid input, it should halt and return the required output.



Division

jacquard loom



Modern computer

https://youtube.com/shorts/KirgYU6aga0?si=ilSVM5XcknpMCvtp

Appendix: Hilbert's decision problem

Can logical reasoning be formalized so that validity can be decided mechanically?

Who	Formalism	View of computation
Church	λ-calculus	expression reduction
Turing	Turing machines	mechanical symbol manipulation

Different formalisms were shown to define the same class of computable functions

Algorithms and Analysis

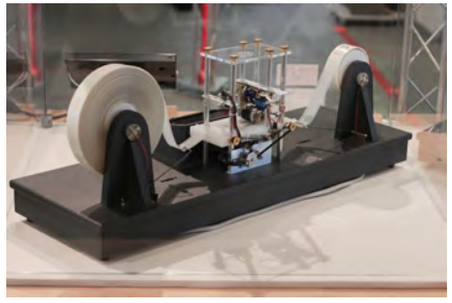
Appendix: Hilbert's decision problem

Can logical reasoning be formalized so that validity can be decided mechanically?

Who	Formalism	View of computation
Church	λ-calculus	expression reduction
Turing	Turing machines	mechanical symbol manipulation

They said NO!

Different formalisms were shown to define the same class of computable functions



Turing machine

Parallel computing

GPU

Quantum computer

What Does Theoretical Computer Science Study?

Question	Area
What is a computational procedure?	models of computation
What can or cannot be computed?	computability theory
How many computational resources are required?	complexity theory
How do we solve concrete problems efficiently?	algorithms and data structures

TCS seeks machine-independent explanations and guarantees, complementing measurements on particular implementations and machines.

Algorithms and Analysis

What Does Theoretical Computer Science Study?

Question	Area
What is a computational procedure?	models of computation
What can or cannot be computed?	computability theory
How many computational resources are required?	complexity theory
How do we solve concrete problems efficiently?	algorithms and data structures

TCS seeks machine-independent explanations and guarantees, complementing measurements on particular implementations and machines.

How should we express this difficulty?

difficulty $\approx$ growth of required resources

$\rightarrow$ time, space, samples, ...

Asymptotic Bounds: Big-O Notation

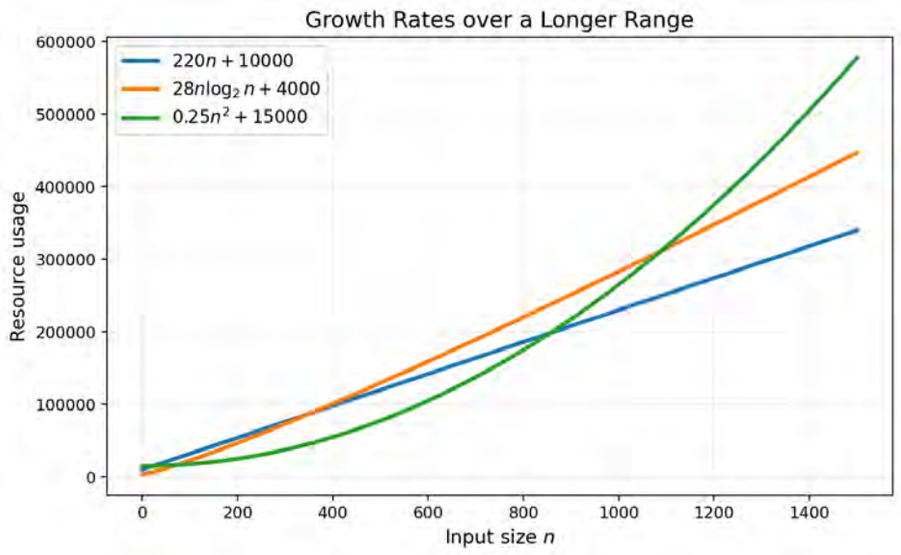
Let $T(n)$ be the resource usage for inputs of size n.

$$T(n) = O(g(n)) \iff \exists c > 0,\ n_0 \in \mathbb{N} :\ T(n) \leq cg(n),\ \forall n \geq n_0 \iff \limsup_{n\to\infty} \frac{T(n)}{g(n)} < \infty.$$

Examples:

- $220n + 10000 = O(n)$
- $28n \log_2 n + 4000 = O(n \log n)$
- $0.25n^2 + 15000 = O(n^2)$

Big-O gives upper bounds on resource growth.



Simplified word-RAM model

Count each primitive operation as unit cost:

- arithmetic operation
- comparison / branch
- memory read
- memory write

```python
def counter(n):
    count = 0

    for _ in range(n):
        count += 1

    return count
```

$T_A(n) = O(n), \quad S_A(n) = O(1)$

$$T_A(x) = \text{number of primitive operations performed by } A \text{ on } x.$$

Space is measured differently:

$$S_A(x) = \text{maximum number of memory cells used during the computation.}$$

$$\text{primitive operations} \longrightarrow \text{time complexity}$$

$$\text{stored data} \longrightarrow \text{space complexity}$$

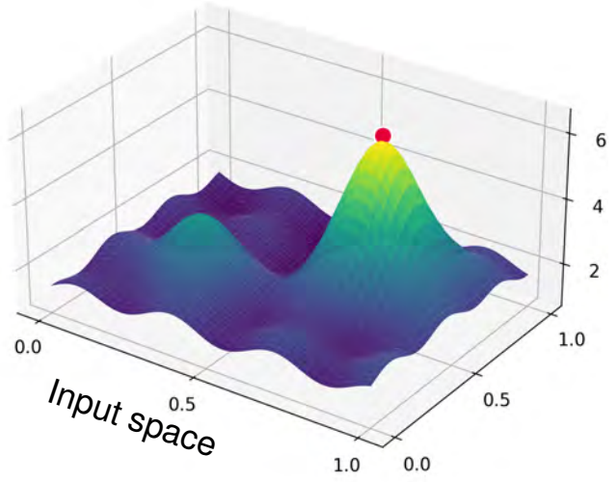
Algorithms and Analysis

Worst, Average, and Amortized Analysis

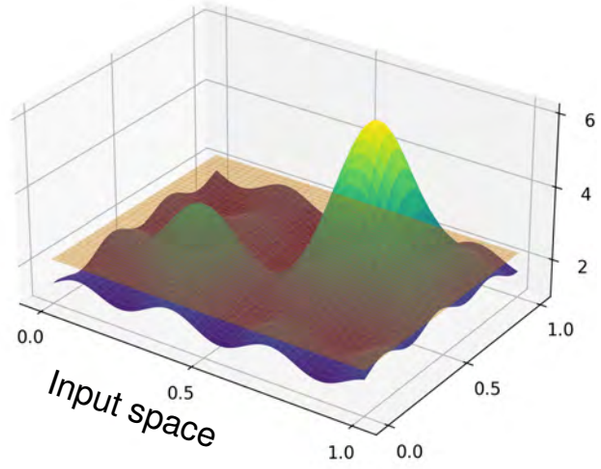
For an algorithm A and input x, let $C_A(x) =$ cost of running A on x.

Analysis	Cost function	Meaning
Worst case	$\max_{\lvert x\rvert=n} C_A(x)$	guarantee for every input
Average case	$\mathbb{E}_{x\sim D_n}[C_A(x)]$	expectation over a specified input distribution
Amortized case	$\frac{1}{m}\max_{\sigma:\,\lvert\sigma\rvert=m}\sum_{i=1}^{m} C_i$	average over operations, worst over sequences

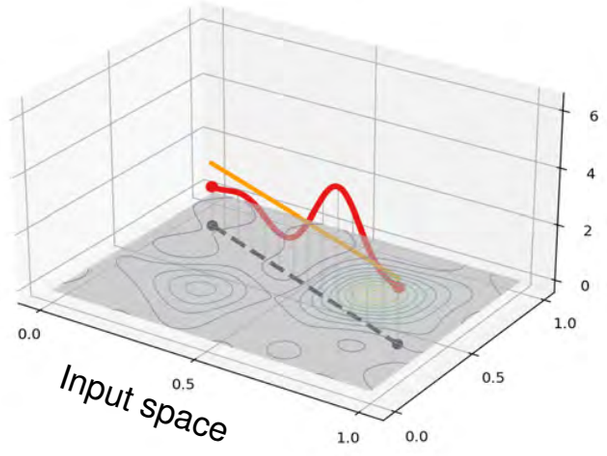
cost landscapes $z = C_A(x, y)$



Worst



Average



Amortized

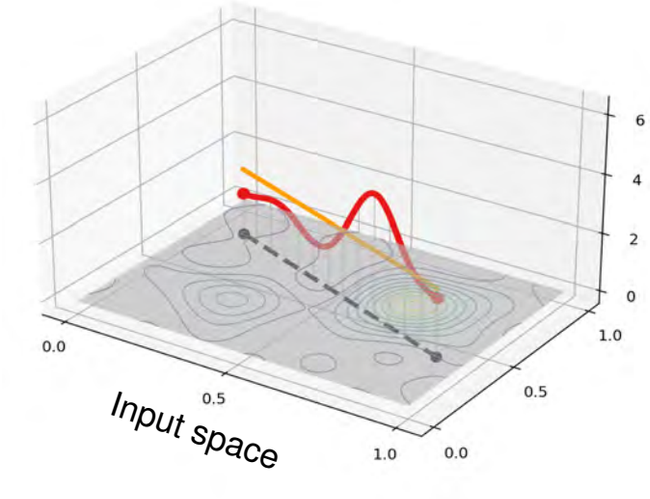
Algorithms and Analysis

Example: k-bit Binary Counter

Cost of increment $=$ # bit flips. Bits are indexed from 0.

Analysis	Counting idea	Cost
Worst case	$011\cdots1 \to 100\cdots0$ $\max$ #flips $= k$	$O(k)$
Average case	$\Pr[\text{bit } i \text{ flips}] = 2^{-i}$ $\mathbb{E}[\#\text{flips}] = \sum_{i=0}^{k-1} 2^{-i} < 2$	$O(1)$
Amortized case	#flips of bit $i \le m/2^i$ $\sum_{i=0}^{k-1} m/2^i < 2m$	$O(1)$ per increment

https://www.youtube.com/watch?v=zELAfmp3fXY



Trajectory of m increments

Algorithms and Analysis

Experimental Analysis: what happens on actual hardware?

We measure performance at three levels.

Level	What we measure
End-to-end behavior	wall-clock time, memory usage, recall
Algorithmic work	distances evals, # candidates, # visited nodes
Machine efficiency	bandwidth, cache misses, GPU occupancy

We then vary the problem size and the available hardware resources, and profile the runtime to understand where the bottlenecks appear.

Algorithms and Analysis

Experimental Analysis: what happens on actual hardware?

We measure performance at three levels.

Level	What we measure
End-to-end behavior	wall-clock time, memory usage, recall
Algorithmic work	distances evals, # candidates, # visited nodes
Machine efficiency	bandwidth, cache misses, GPU occupancy

We then vary the problem size and the available hardware resources, and profile the runtime to understand where the bottlenecks appear.

Empirical scaling.
Observed growth of an implementation on a fixed workload family and machine.

$$C(n) \approx an^b$$

The exponent b is the empirical scaling rate.

Algorithms and Analysis

Appendix: Simplex method.
Linear programming with m constraints and n variables:

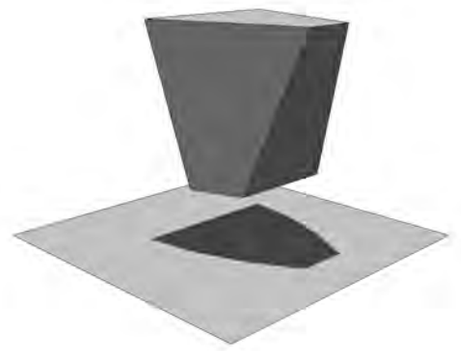
$$\max_x \; c^\top x \quad \text{subject to} \quad Ax \le b. \qquad A \in \mathbb{R}^{m \times n}.$$

Let $P(m, n)$ be the number of pivots.

Worst case.

$$P(m, n) \ge 2^n - 1$$

Dantzig's rule on Klee–Minty cubes (Klee–Minty, 1972).



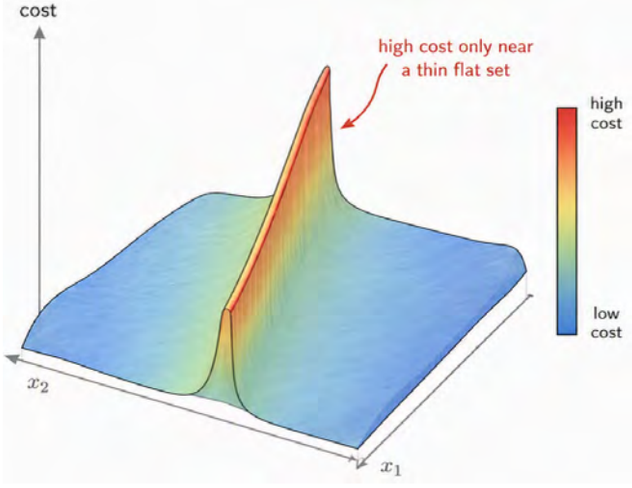
Klee–Minty cube

Empirical behavior. On many standard benchmark families, practical pivot counts can be far smaller than the exponential worst case

Smoothed analysis.

$$\max \mathbb{E}_{\text{noise}}[P(m, n)] \le \text{poly}(m, n, 1/\sigma)$$

for shadow-vertex simplex after small random perturbations (Spielman–Teng, 2004).



Cost landscape

Algorithms and Analysis

Why Theoretical Analysis of Algorithms?

Experiments show performance for a particular implementation, input set, and machine.

Theory asks what is structural.

Question	What theory adds
How does the cost grow?	Scaling with input size
Why is it fast?	The structure exploited by the algorithm
Under what conditions?	Assumptions where the behavior is expected
What is unavoidable?	Lower bounds or inherent bottlenecks

Theory separates accidental speed from structural efficiency.

Brute Force Baseline

Brute force means: **compare everything with everything.**

(1) all pairwise distances $\longrightarrow$ (2) select / filter outputs

For queries $Q \in \mathbb{R}^{m \times d}$ and data $P \in \mathbb{R}^{n \times d}$,
Distance computation:

$$d(q_i, p_j) = \|q_i - p_j\|_2$$

Task	Query set	Distance phase	Selection / output
kNN query	$Q = \{q\}$	$\|q - p_j\|_2$ for all $p_j \in P$	top-k nearest points
kNN graph	$Q = P$	$\|p_i - p_j\|_2$ for all $i \neq j$	top-k neighbors of each p_i
Radius join	$\|Q\| = m$	$\|p_i - p_j\|_2$ for all $i < j$	pairs with $\|p_i - p_j\|_2 \leq r$

Brute Force Baseline

Brute force means: **compare everything with everything.**

(1) all pairwise distances $\longrightarrow$ (2) select / filter outputs

For queries $Q \in \mathbb{R}^{m \times d}$ and data $P \in \mathbb{R}^{n \times d}$,
Distance computation:

$$d(q_i, p_j) = \|q_i - p_j\|_2 \longrightarrow \begin{array}{c} q_i = (x_1, \ldots, x_d), \\ p_j = (y_1, \ldots, y_d) \\ \|q_i - p_j\|_2 \to \text{cost } O(d) \end{array}$$

Task	Query set	Distance phase	Selection / output
kNN query	$Q = \{q\}$	$\|q - p_j\|_2$ for all $p_j \in P$	top-k nearest points
kNN graph	$Q = P$	$\|p_i - p_j\|_2$ for all $i \neq j$	top-k neighbors of each p_i
Radius join	$\|Q\| = m$	$\|p_i - p_j\|_2$ for all $i < j$	pairs with $\|p_i - p_j\|_2 \leq r$

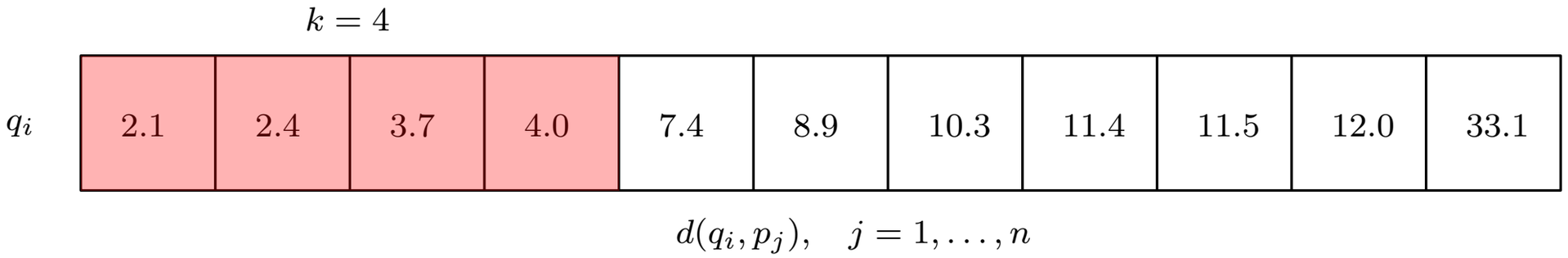
Compute all pairwise distances

Brute Force Baseline

Selection Cost after Brute-force Distances

- A simple baseline sorts all distances and takes the first k elements.
- Using a heap can further reduce the cost.

Task	Candidates	Selection rule	Selection cost
kNN query	n distances	top-k	$O(dn \log n)$ by sorting
kNN graph	n rows, each with $n-1$ distances	top-k per row	$O(dn^2 \log n)$ by sorting
Radius join	nm distances	threshold filter	$O(dnm)$



Brute Force Baseline

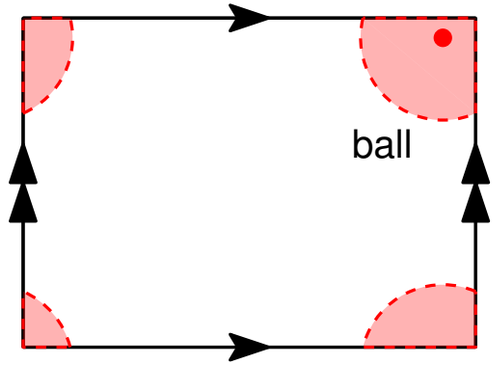
Appendix: Analysis of reporting number z.

Let $z = \#\{(a, b) \in A \times B : \|a - b\|_2 \leq r\}$, with $0 < r < 1/2$. Worst case: $z = O(nm)$.
All pairs may be within distance r.

Periodic cube

Assume $A, B \subset [0, 1)^d$ are uniformly distributed, with periodic boundary conditions, where $|A| = n$ and $|B| = m$.

$$[0, 1)^d / \mathbb{Z}^d \qquad \text{flat torus}$$



For a fixed $a \in A$,

$$\Pr[\|a - b\|_2 \leq r] = \mathrm{vol}(B_d(r)) = V_d r^d,$$

where

$$V_d = \mathrm{vol}(B_d(1)) = \frac{\pi^{d/2}}{\Gamma(d/2 + 1)}.$$

For radius join between A and B,

$$\mathbb{E}[z] = nmV_d r^d = O(nmV_d r^d). \qquad \because r < 1, \text{ it goes to 0 as } d \to \infty$$

Brute Force Baseline

Practical Distance Computation

Let $Q \in \mathbb{R}^{m \times d}$ and $P \in \mathbb{R}^{n \times d}$. For selection, squared distances are enough:

$$D_{ij} = \|q_i - p_j\|_2^2 = \|q_i\|_2^2 + \|p_j\|_2^2 - 2q_i^\top p_j.$$

Let

$$q_i^{(2)} = \|q_i\|_2^2, \qquad p_j^{(2)} = \|p_j\|_2^2.$$

Then the full distance matrix is

$$D = q^{(2)} \mathbf{1}_n^\top + \mathbf{1}_m (p^{(2)})^\top - 2QP^\top.$$

The norm terms are row-wise reductions; the dominant term is $QP^\top$, a matrix multiplication.

Naive vectorization creates an $m \times n \times d$ intermediate.
The GEMM formulation avoids this tensor and computes distances through $QP^\top$.

Brute Force Baseline

Practical Distance Computation

Let $Q \in \mathbb{R}^{m \times d}$ and $P \in \mathbb{R}^{n \times d}$. For selection, squared distances are enough:

$$D_{ij} = \|q_i - p_j\|_2^2 = \|q_i\|_2^2 + \|p_j\|_2^2 - 2q_i^\top p_j.$$

Let

$$q_i^{(2)} = \|q_i\|_2^2, \qquad p_j^{(2)} = \|p_j\|_2^2.$$

Then the full distance matrix is

$$D = q^{(2)}\mathbf{1}_n^\top + \mathbf{1}_m(p^{(2)})^\top - \boxed{2QP^\top}.$$

highly optimized primitive

The norm terms are row-wise reductions; the dominant term is $QP^\top$, a matrix multiplication.

Naive vectorization creates an $m \times n \times d$ intermediate.
The GEMM formulation avoids this tensor and computes distances through $QP^\top$.

But is this always beneficial?

Brute Force Baseline

Appendix: Numerical Precision

Recall the GEMM-based distance computation:

$$D = q^{(2)}\mathbf{1}_n^\top + \mathbf{1}_m(p^{(2)})^\top - 2QP^\top,$$

Floating point: FP32 vs. FP64

- **FP32**: 32-bit; often faster, less precise $\approx 10^{-7}$ machine epsilon.
- **FP64**: 64-bit; often slower, more precise $\approx 10^{-16}$ machine epsilon.

Catastrophic cancellation

If q_i and p_j are very close, then D_{ij} is small.

$$D_{ij} = \underbrace{\|q_i\|^2 + \|p_j\|^2}_{\text{large}} - \underbrace{2q_i^\top p_j}_{\text{large}}$$

small distance $=$ large number $-$ nearly equal large number.

Small relative errors in the large terms can become large relative errors in D_{ij}.

In FP32, be careful!
$x = 1.0000001$,
$y = 1.0000000$
$x^2 - y^2 \to (x+y)(x-y)$

Nearest Neighbor Search for Scientific Computing

Lecture 2

Beyond Brute Force: Pruning and Parallel Search

Jaehoon Chung

Research Fellow, Center for AI and Natural Sciences
Korea Institute for Advanced Study

sk7755@kias.re.kr

Contents

Lecture 2: Beyond Brute Force: Pruning and Parallel Search

- **From Distance to Geometry**
  View Euclidean distance as spatial structure, then use geometry to avoid comparisons.
- **kd-tree and Ball Tree**
  Prune with boxes in Euclidean space and balls in metric space.
- **Curse of Dimensionality**
  Understand why pruning collapses when distances become similar.
- **Intrinsic Dimension**
  Explain when real data are easier than their ambient dimension suggests.
- **Parallel Search on Hardware**
  Split queries, subtrees, cells, or distance blocks across CPU/GPU execution units.
- **Hardware Bottlenecks and Profiing**
  Memory movement, locality, warp divergence, and cold/warm profiling.
- **Agentic Coding for Optimization**
  Use LLM agents and profiling data for performance engineering.

From Distance to Geometry

From Distance to Geometry

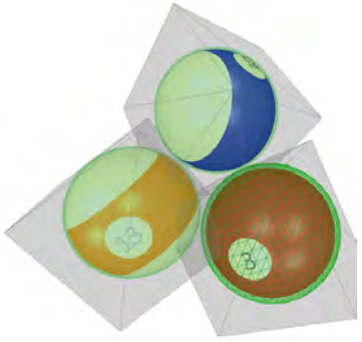
$$x, y \in \mathbb{R}^d \qquad \|x - y\|_2$$

Not just an abstract model.
A geometric relationship between points.

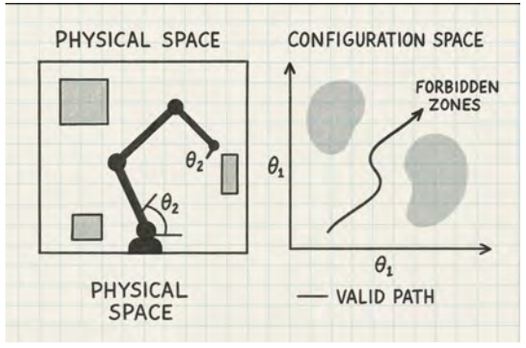
Computational geometry	uses	spatial structure
	studies	distance, intersection, containment, neighborhood
	motivations	graphics, CAD/CAM, robotics, GIS, VLSI

Even non-geometric problems may become easier after geometric representation.

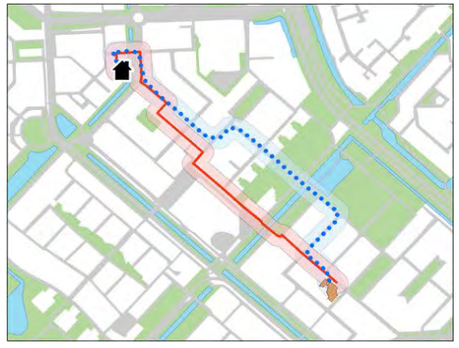
Can geometry help us avoid comparing everything?



collision detection



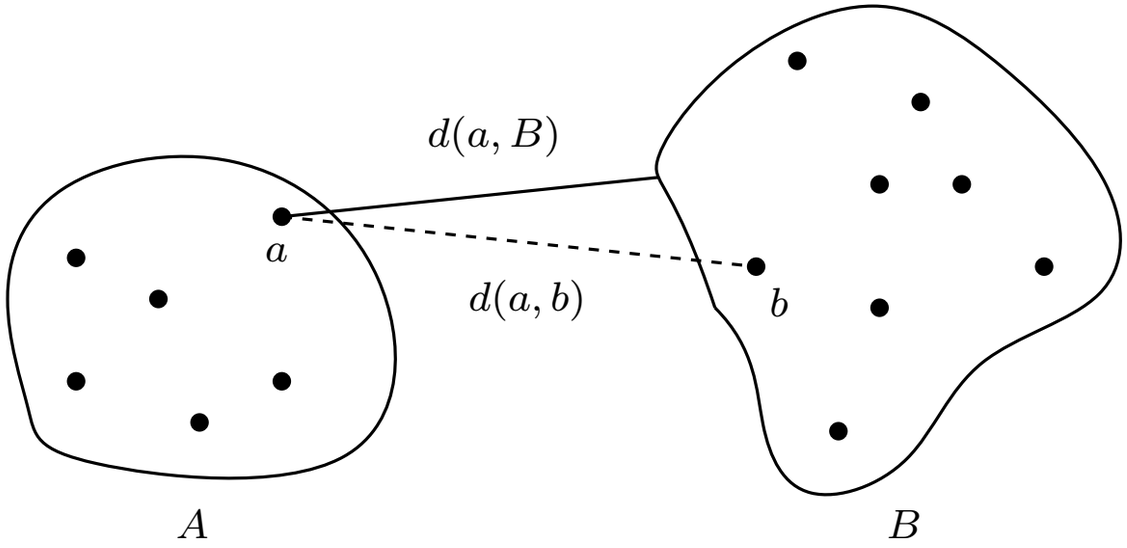
motion planning



shortest path

From Distance to Geometry

Basic pruning rule: $d(a, B) \leq d(a, b)$



- Current best: $\tau \geq 0$.
- Lower bound for B: $d(a, B) = \min_{y \in B} d(a, y)$.
- If $d(a, B) > \tau$, prune B.

This pruning rule is not geometric by itself.

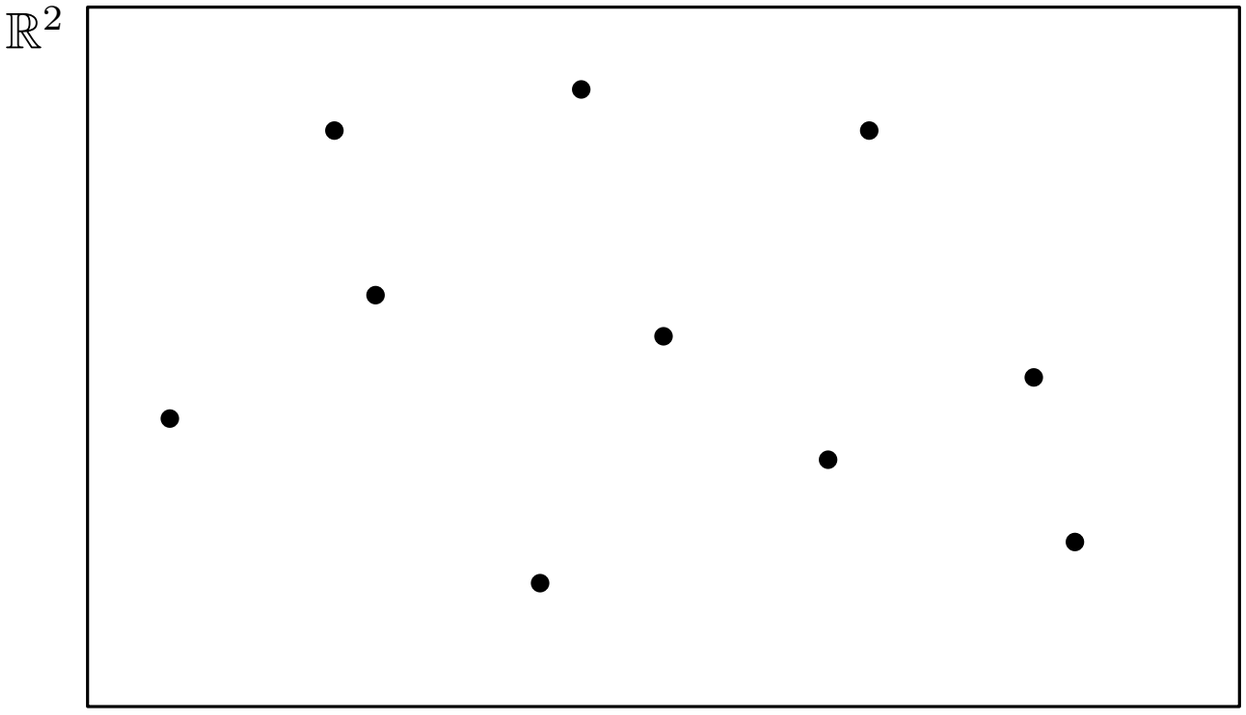
Key features of fast pruning

- Cells should be simple to compute with.
- Point pairs should be quickly decomposed into cell pairs. → Euclidean space provides a particularly convenient setting!
- Small point groups should occupy small cells.

kd-tree and Ball Tree

Tree-based indexing: kd-tree

- Choose one coordinate axis.
- Cut the points into two almost equal-sized groups.
- Repeat this recursively.
- Each node corresponds to a simple spatial cell.

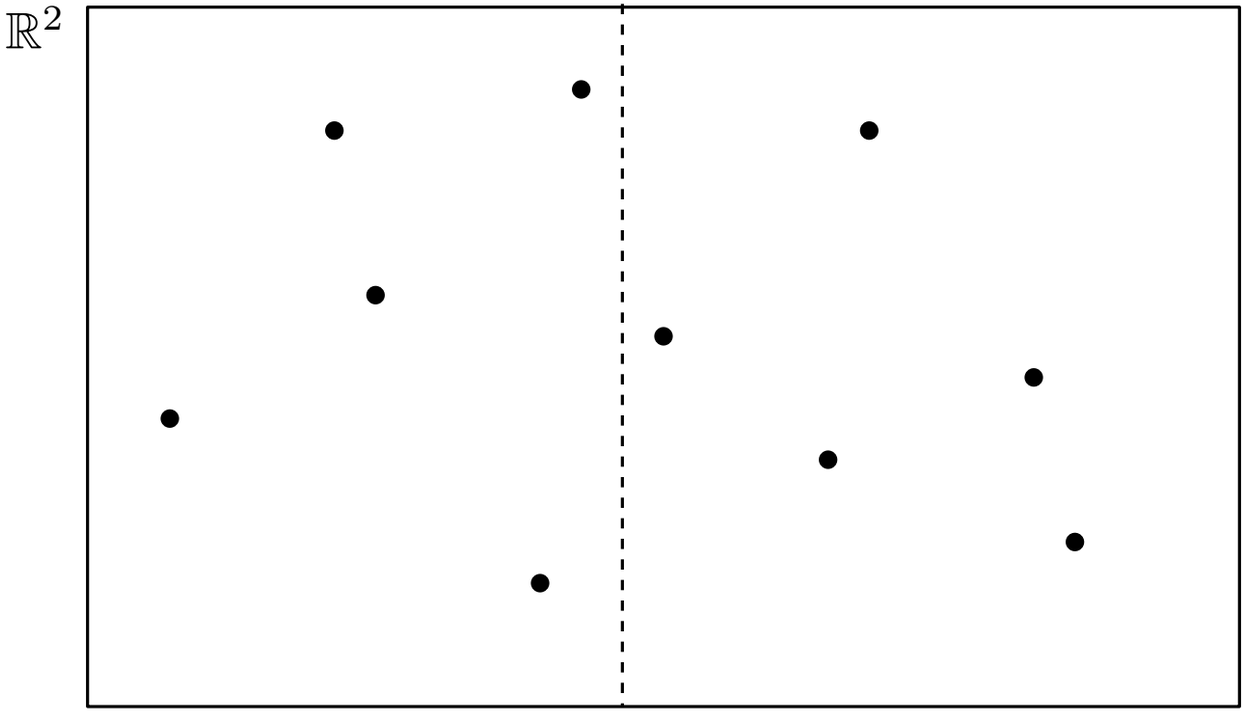


kd-tree

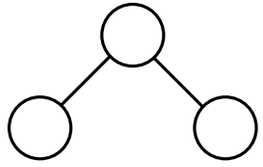
kd-tree and Ball Tree

Tree-based indexing: kd-tree

- Choose one coordinate axis.
- Cut the points into two almost equal-sized groups.
- Repeat this recursively.
- Each node corresponds to a simple spatial cell.



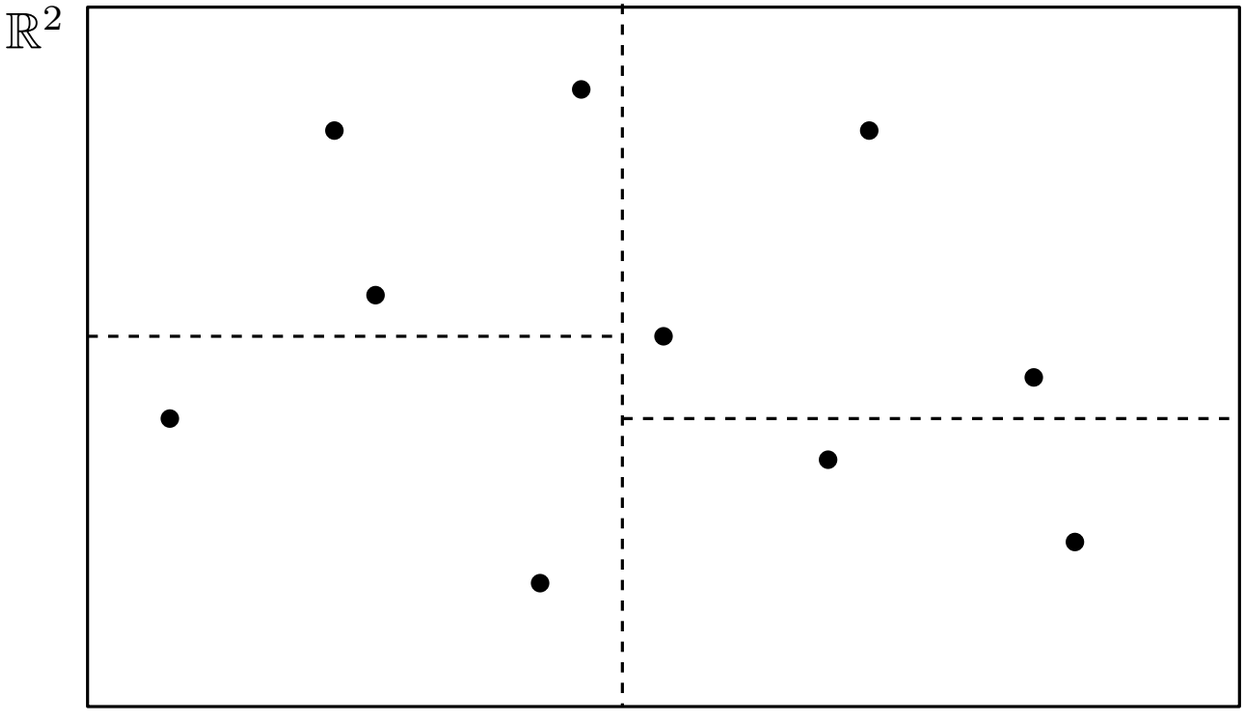
kd-tree



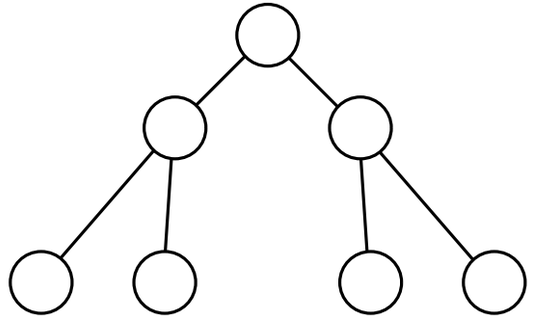
kd-tree and Ball Tree

Tree-based indexing: kd-tree

- Choose one coordinate axis.
- Cut the points into two almost equal-sized groups.
- Repeat this recursively.
- Each node corresponds to a simple spatial cell.



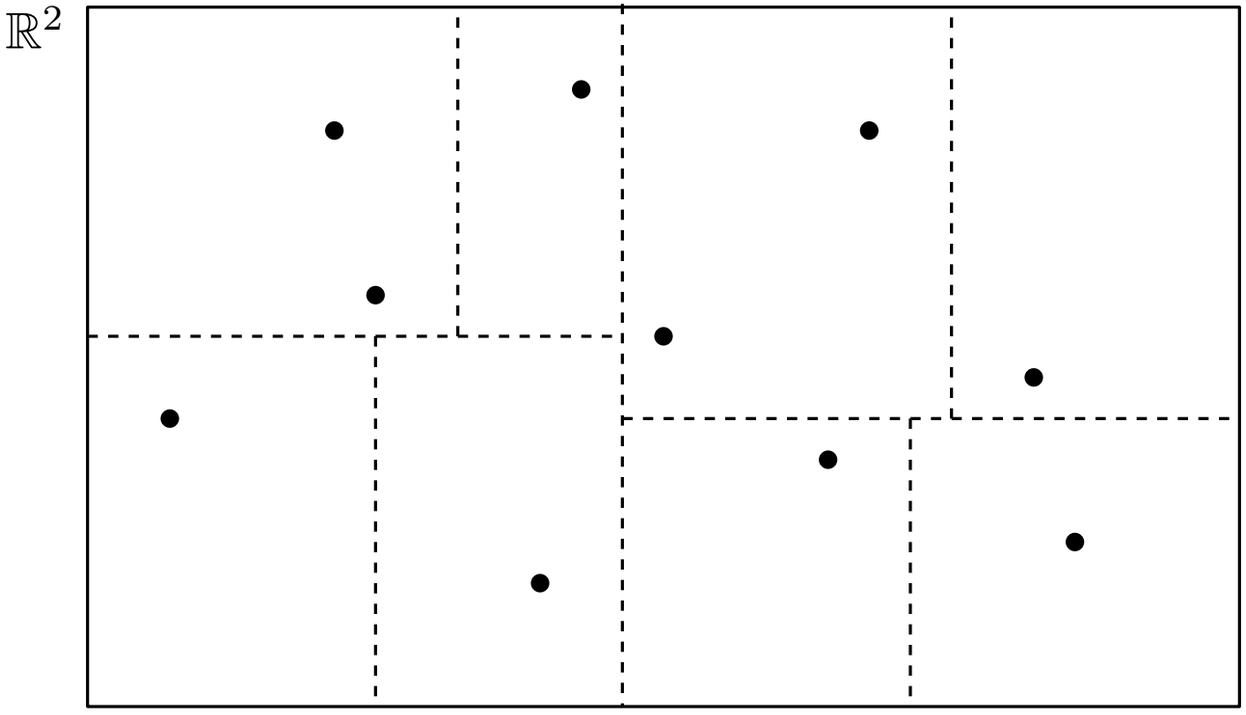
kd-tree



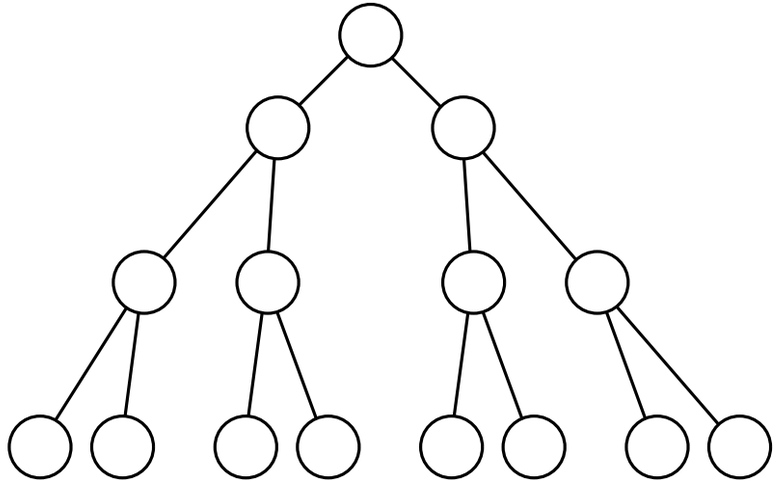
kd-tree and Ball Tree

Tree-based indexing: kd-tree

- Choose one coordinate axis.
- Cut the points into two almost equal-sized groups.
- Repeat this recursively.
- Each node corresponds to a simple spatial cell.



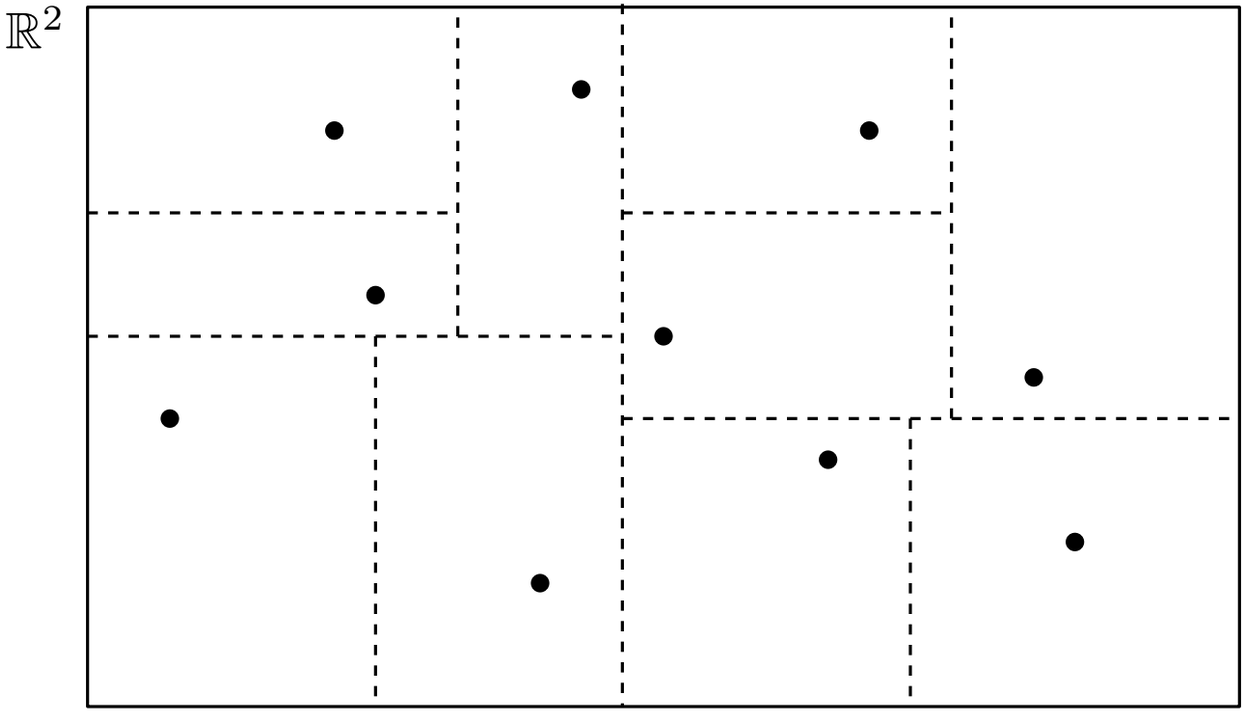
kd-tree

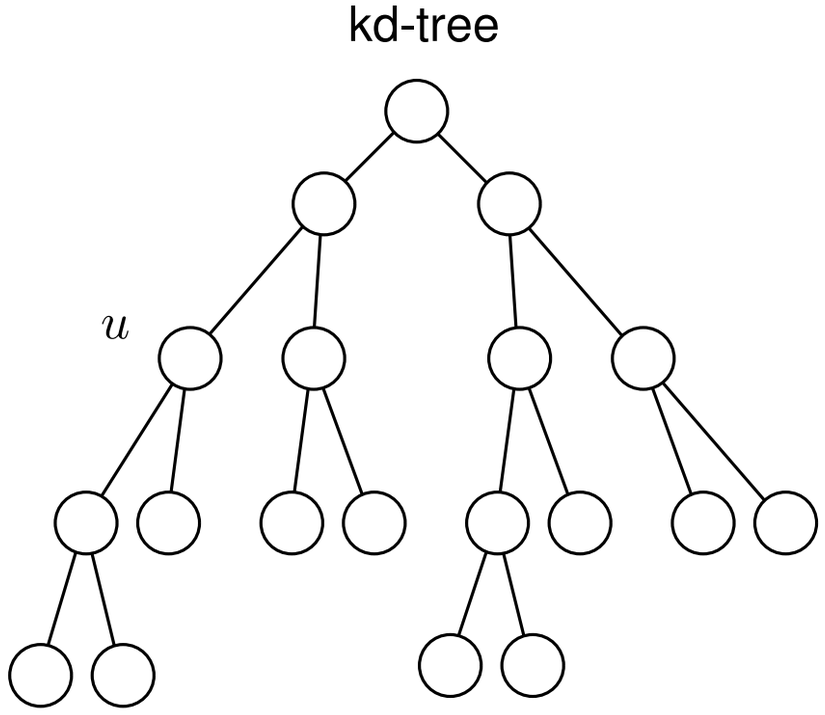


kd-tree and Ball Tree

Tree-based indexing: kd-tree

- Choose one coordinate axis.
- Cut the points into two almost equal-sized groups.
- Repeat this recursively.
- Each node corresponds to a simple spatial cell.

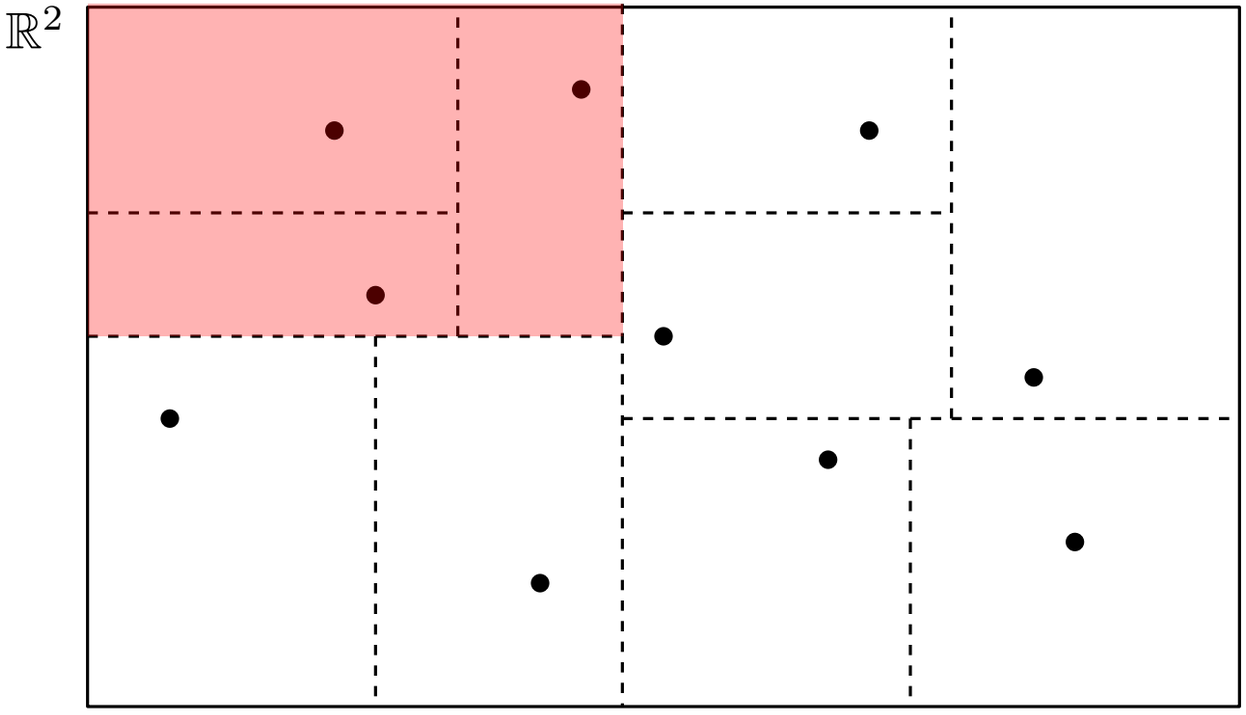


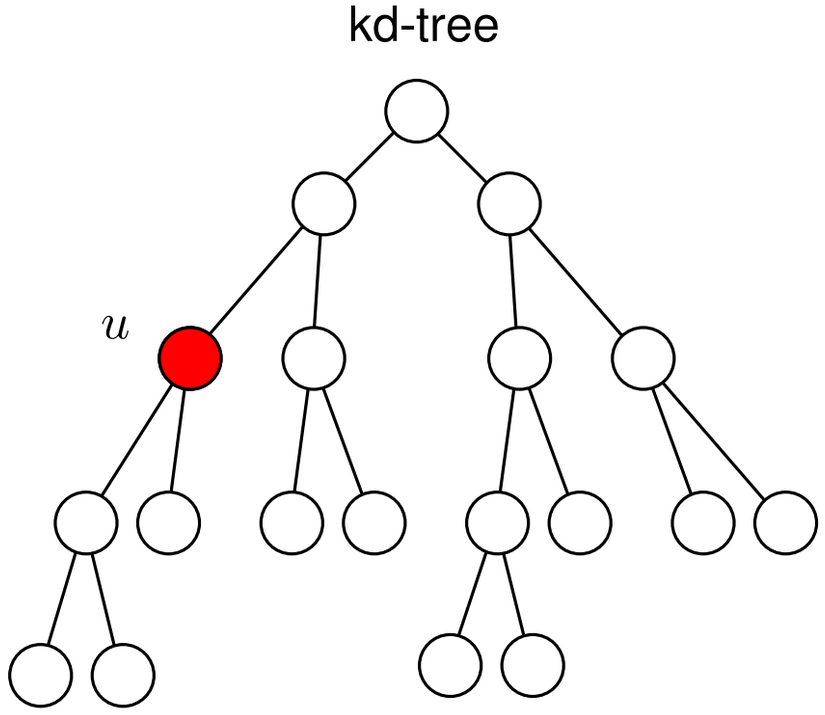


kd-tree and Ball Tree

Tree-based indexing: kd-tree

- Choose one coordinate axis.
- Cut the points into two almost equal-sized groups.
- Repeat this recursively.
- Each node corresponds to a simple spatial cell.

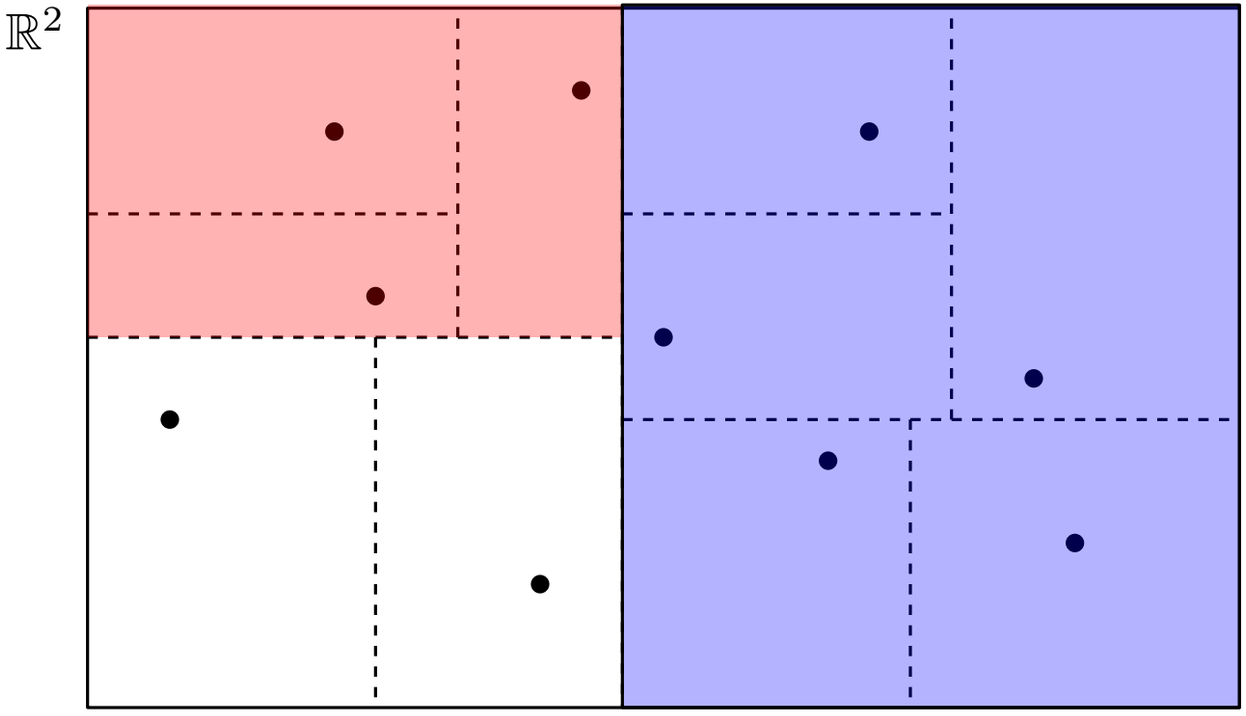


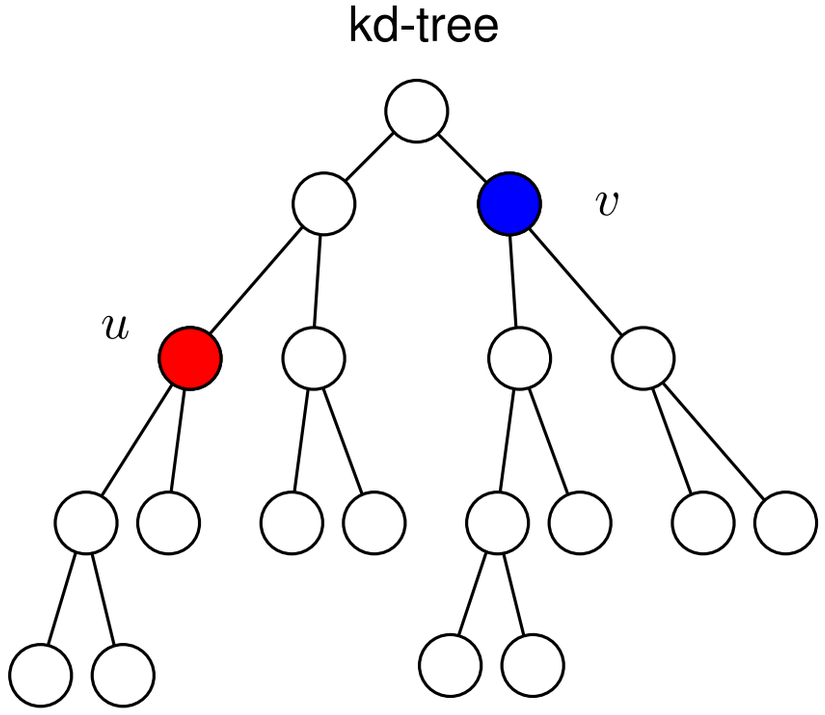


kd-tree and Ball Tree

Tree-based indexing: kd-tree

- Choose one coordinate axis.
- Cut the points into two almost equal-sized groups.
- Repeat this recursively.
- Each node corresponds to a simple spatial cell.

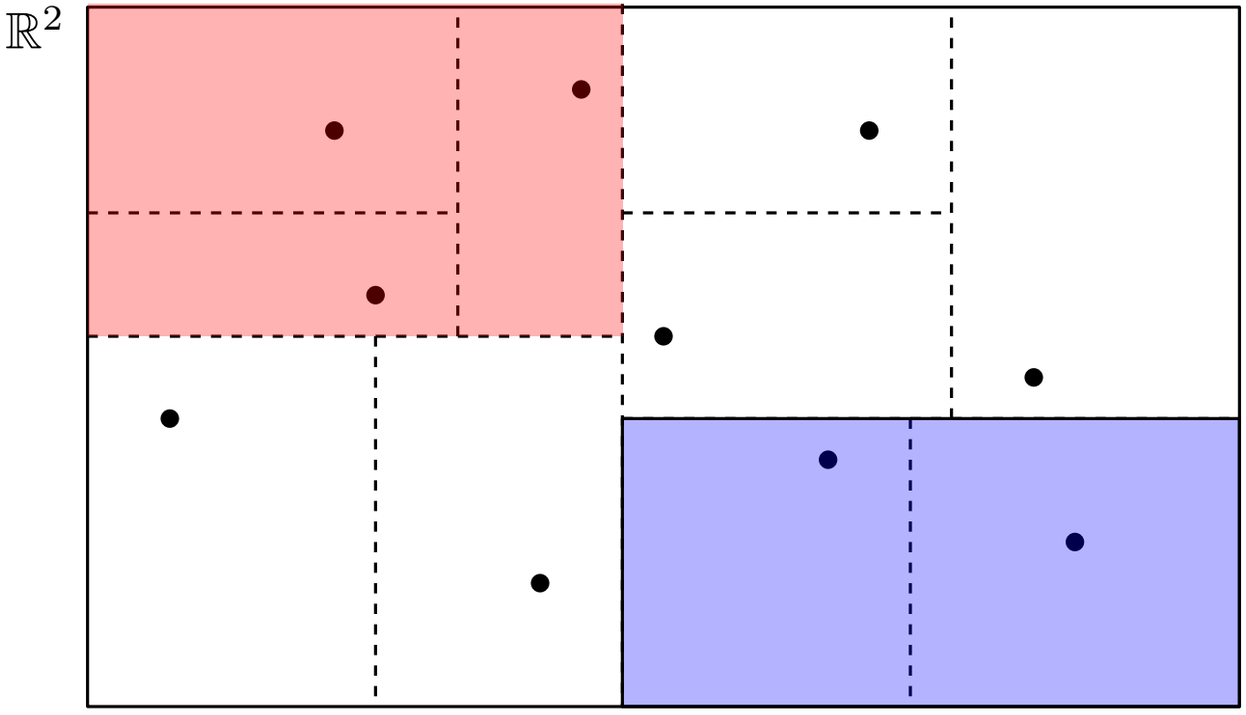


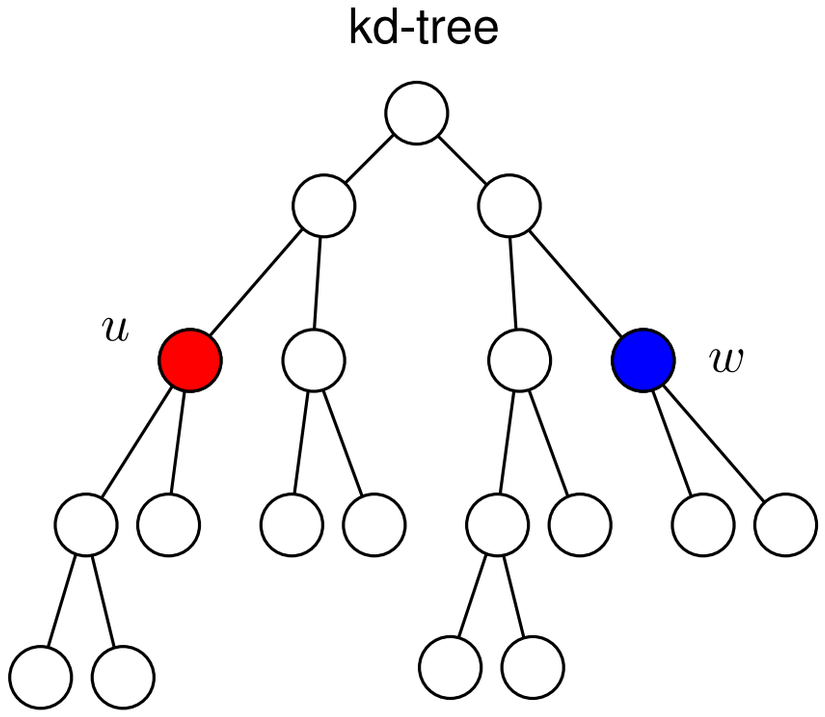


kd-tree and Ball Tree

Tree-based indexing: kd-tree

- Choose one coordinate axis.
- Cut the points into two almost equal-sized groups.
- Repeat this recursively.
- Each node corresponds to a simple spatial cell.

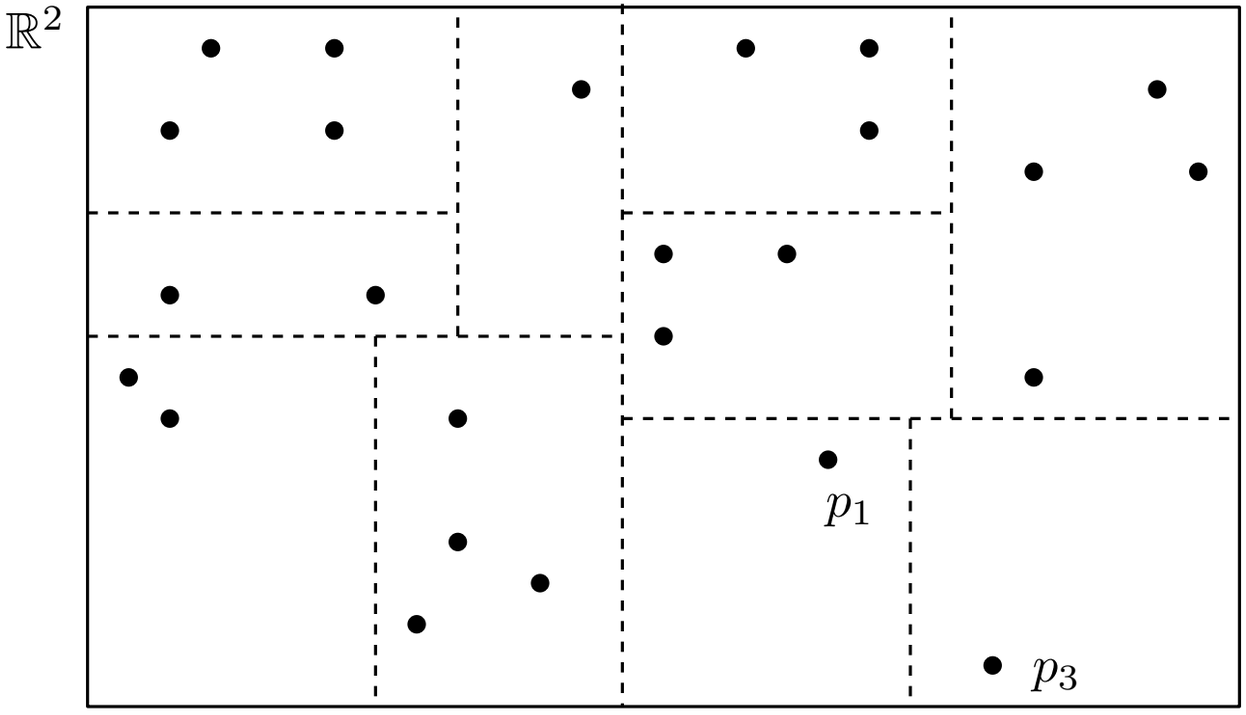


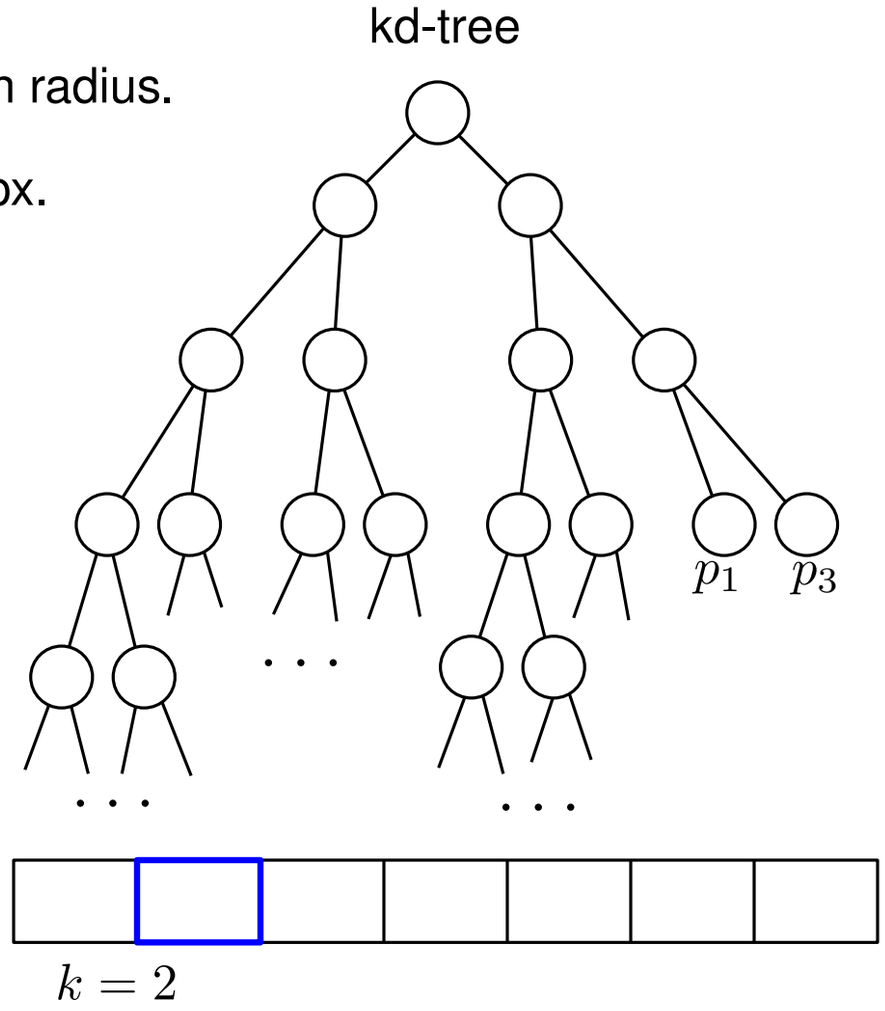


kd-tree and Ball Tree

kNN query in kd-tree

- Follow the split decisions to the leaf reached by the query.
- Find some nearby candidates.
- Use the current k-th nearest distance as a search radius.
- If a box is outside this radius, ignore the whole box.

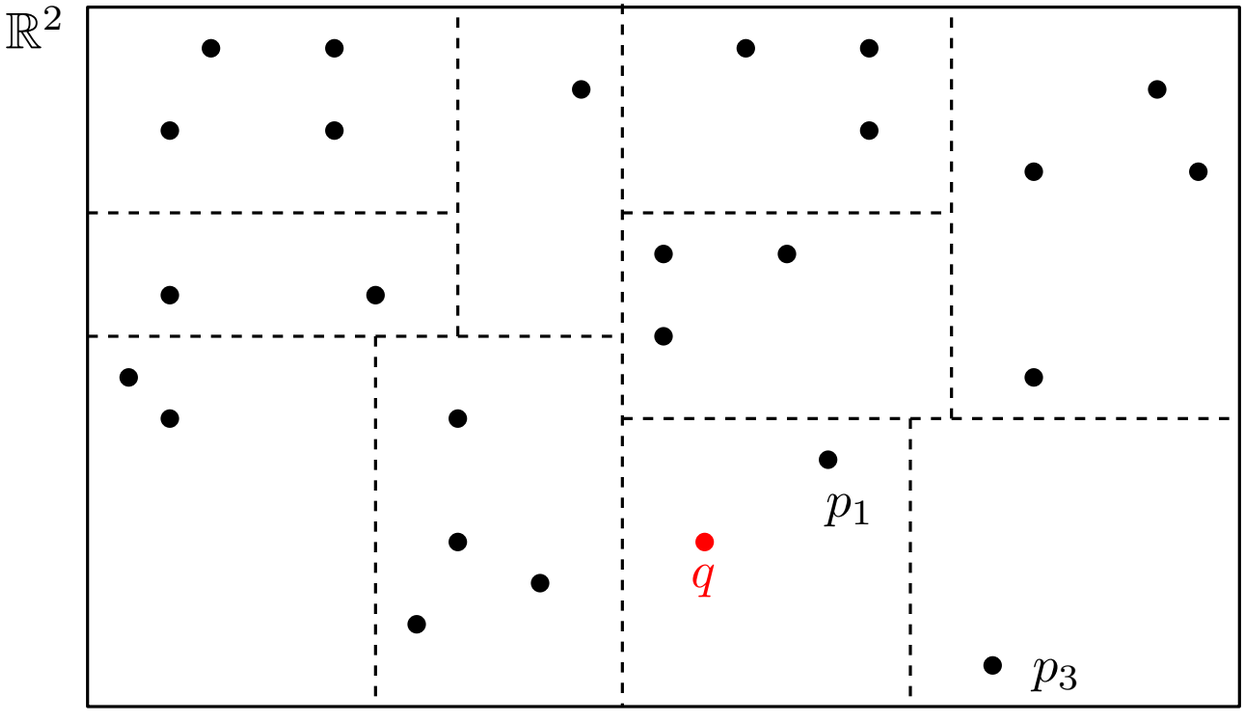


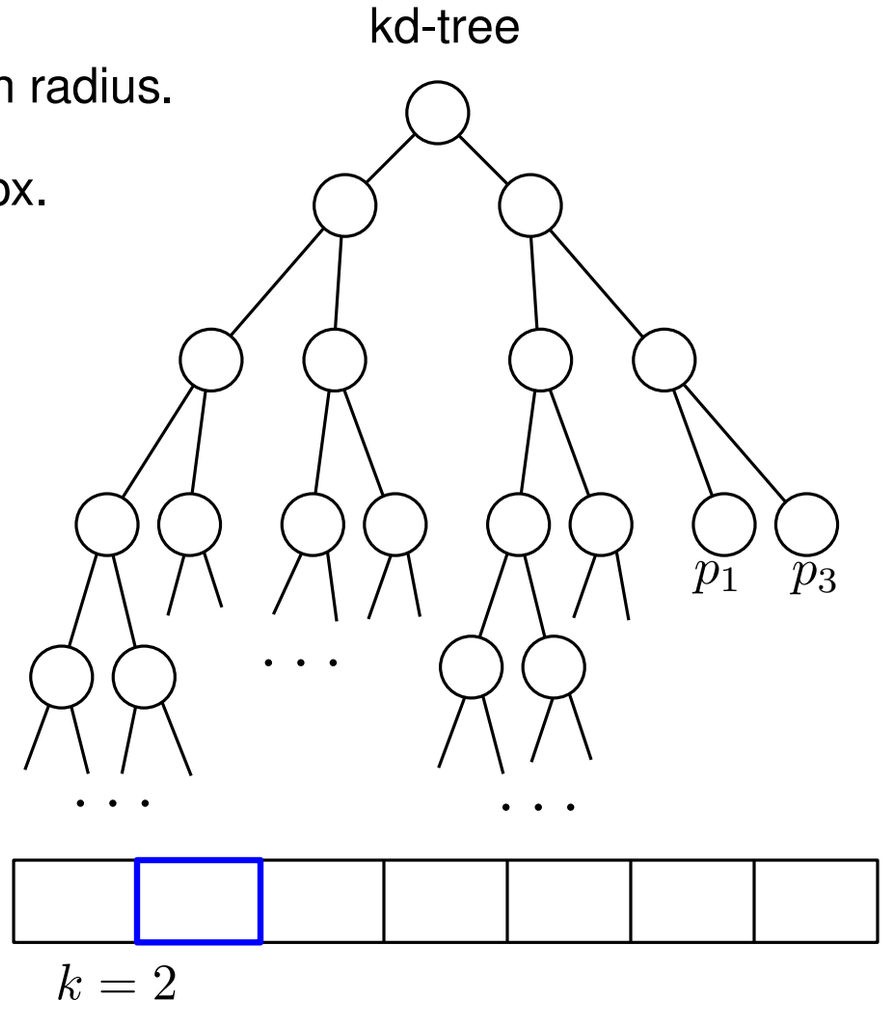


kd-tree and Ball Tree

kNN query in kd-tree

- Follow the split decisions to the leaf reached by the query.
- Find some nearby candidates.
- Use the current k-th nearest distance as a search radius.
- If a box is outside this radius, ignore the whole box.

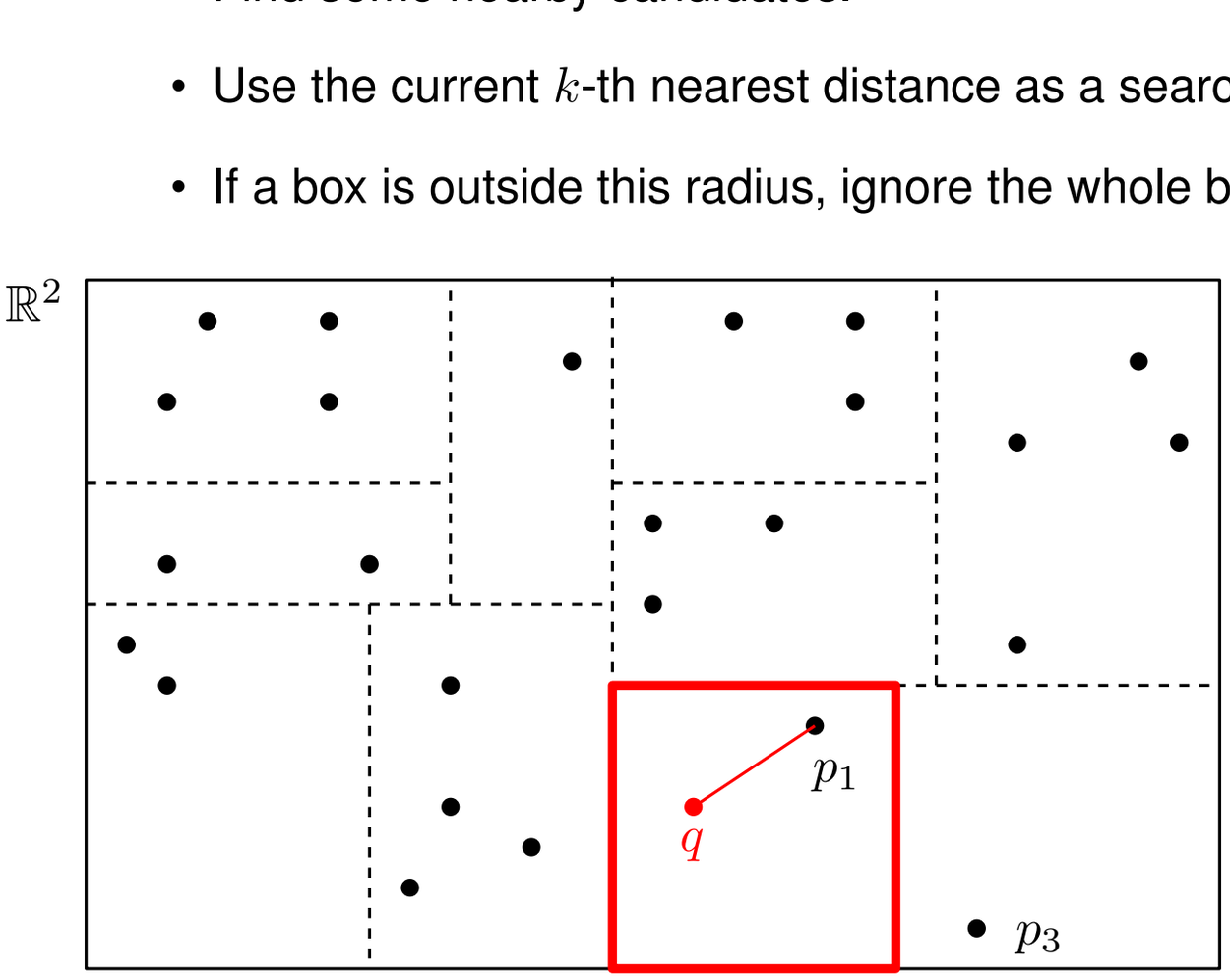


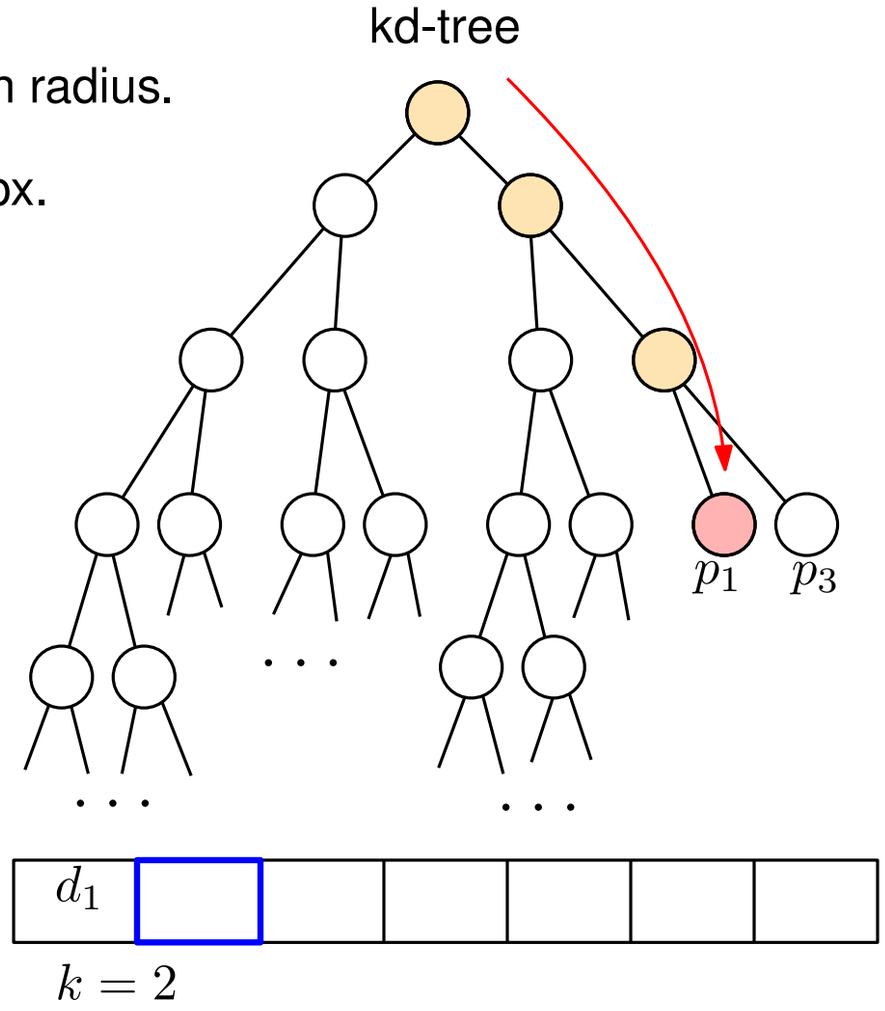


kd-tree and Ball Tree

kNN query in kd-tree

- Follow the split decisions to the leaf reached by the query.
- Find some nearby candidates.
- Use the current k-th nearest distance as a search radius.
- If a box is outside this radius, ignore the whole box.

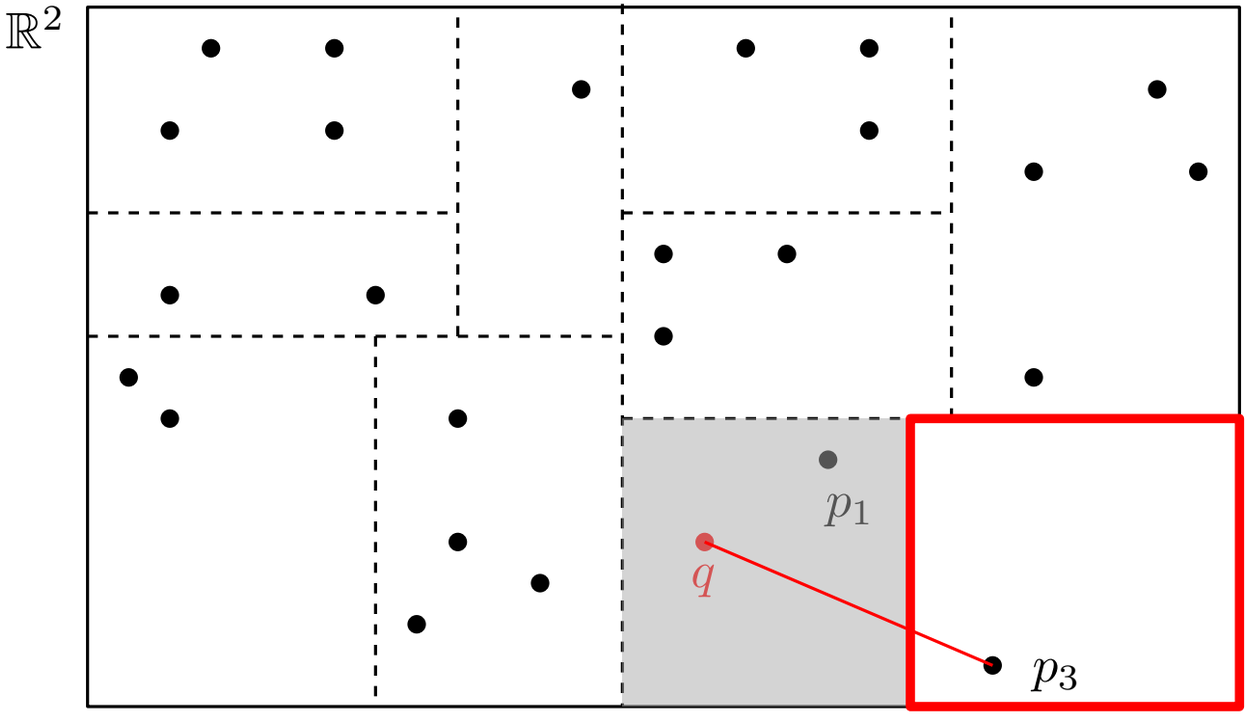


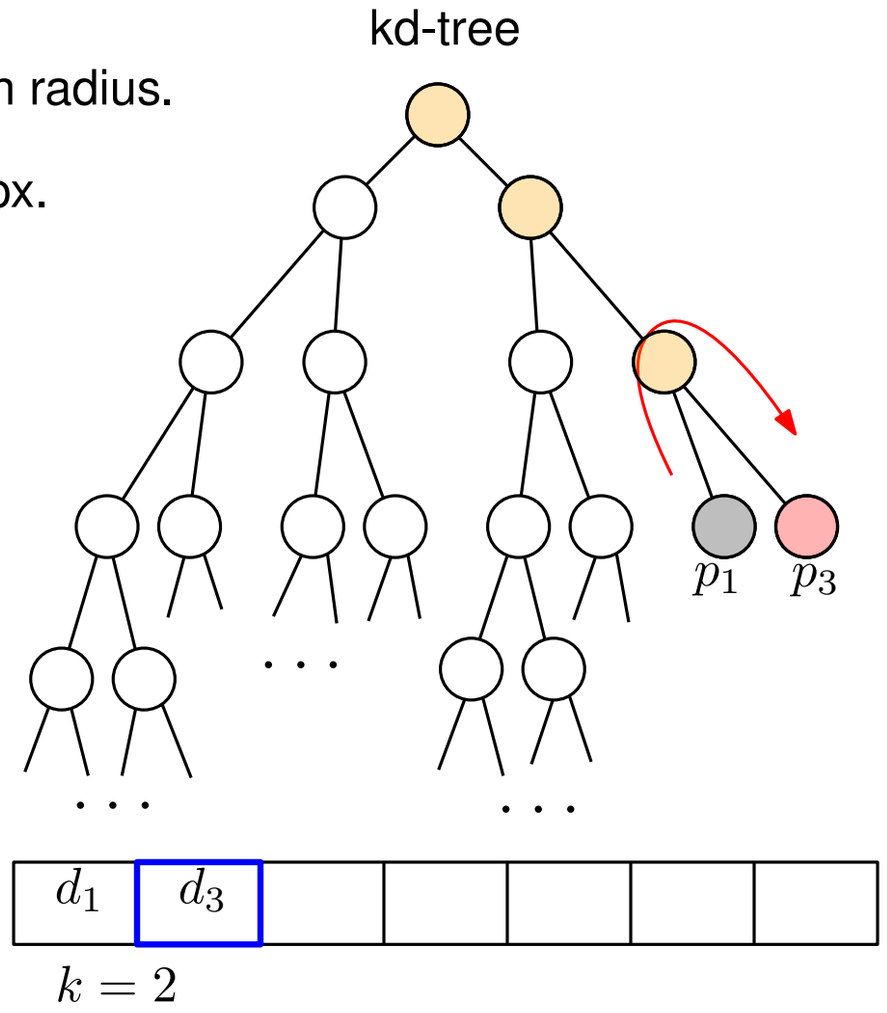


kd-tree and Ball Tree

kNN query in kd-tree

- Follow the split decisions to the leaf reached by the query.
- Find some nearby candidates.
- Use the current k-th nearest distance as a search radius.
- If a box is outside this radius, ignore the whole box.

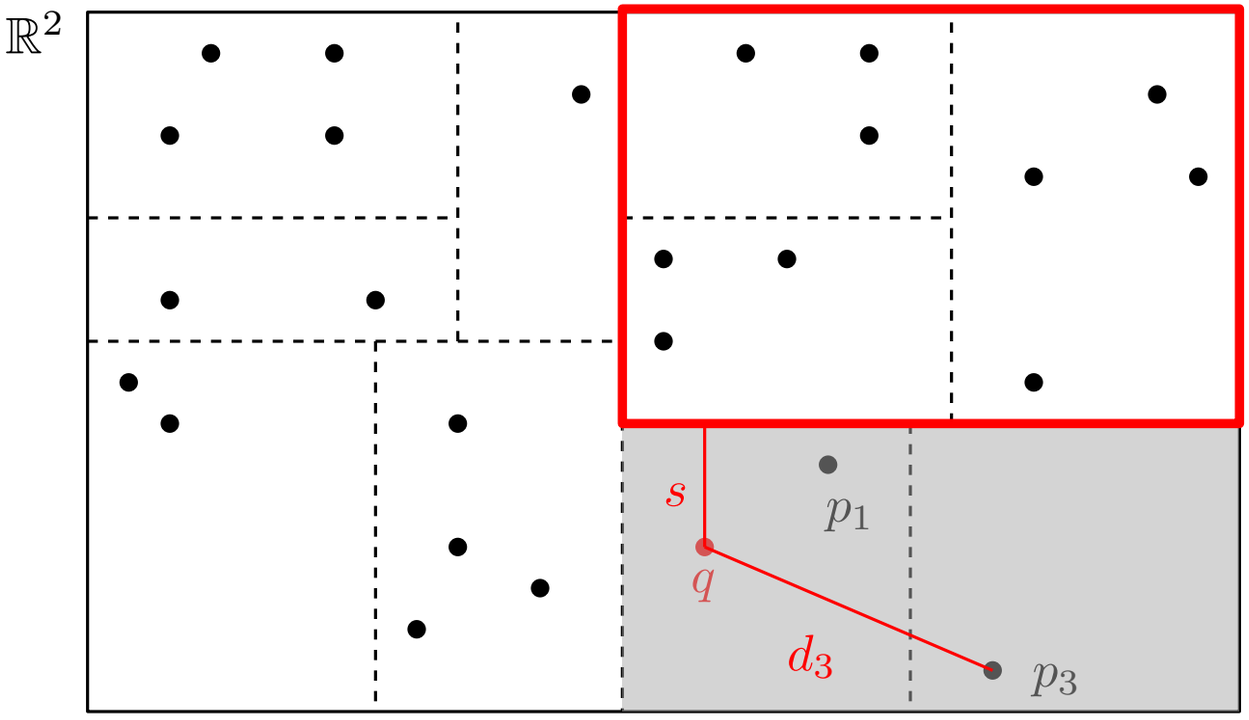


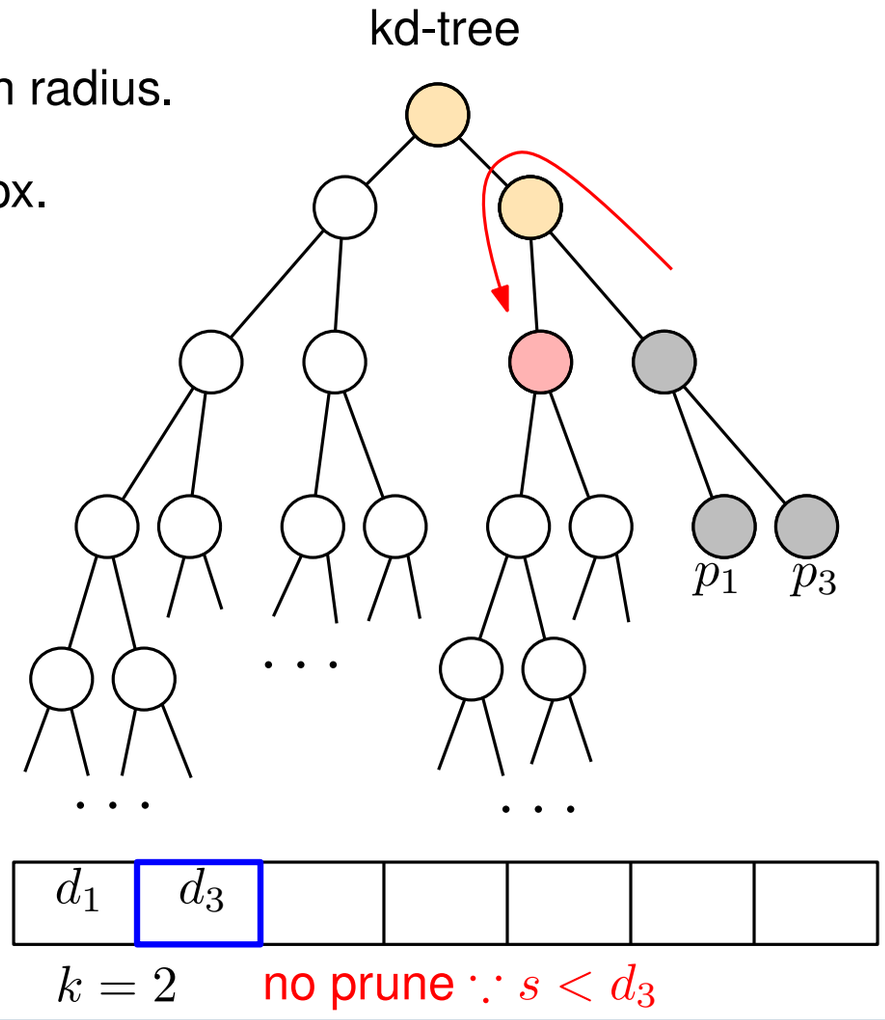


kd-tree and Ball Tree

kNN query in kd-tree

- Follow the split decisions to the leaf reached by the query.
- Find some nearby candidates.
- Use the current k-th nearest distance as a search radius.
- If a box is outside this radius, ignore the whole box.

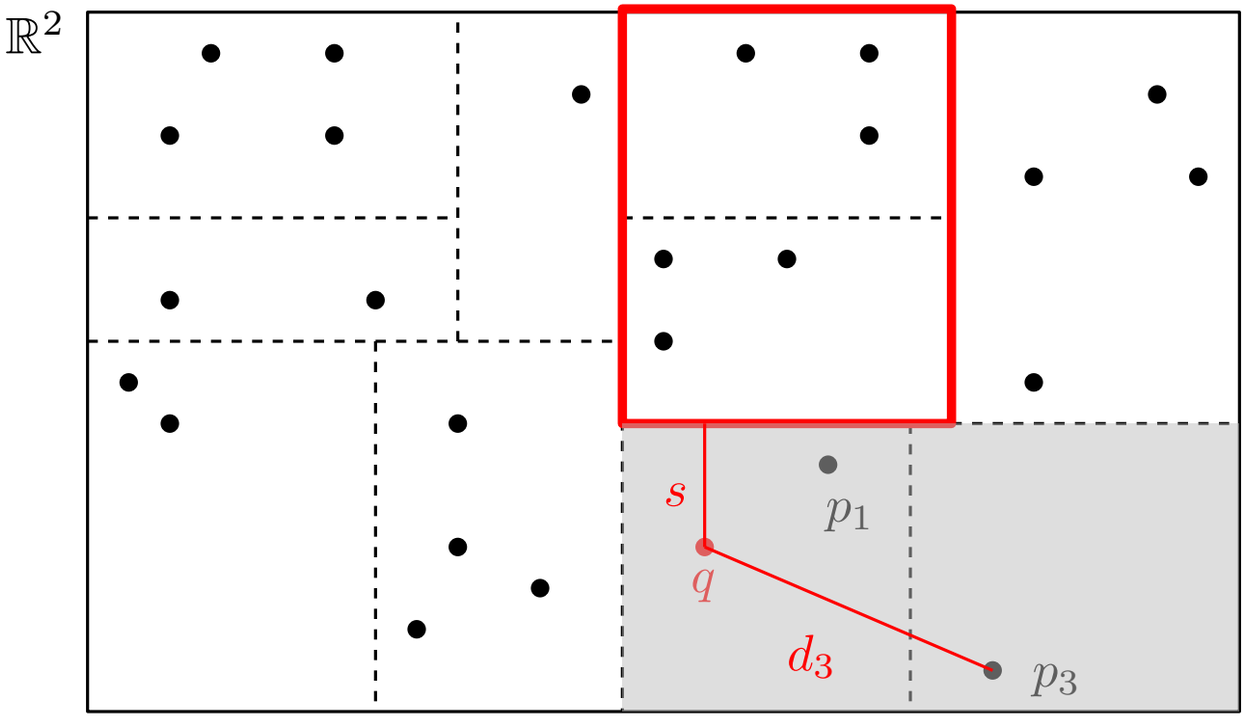


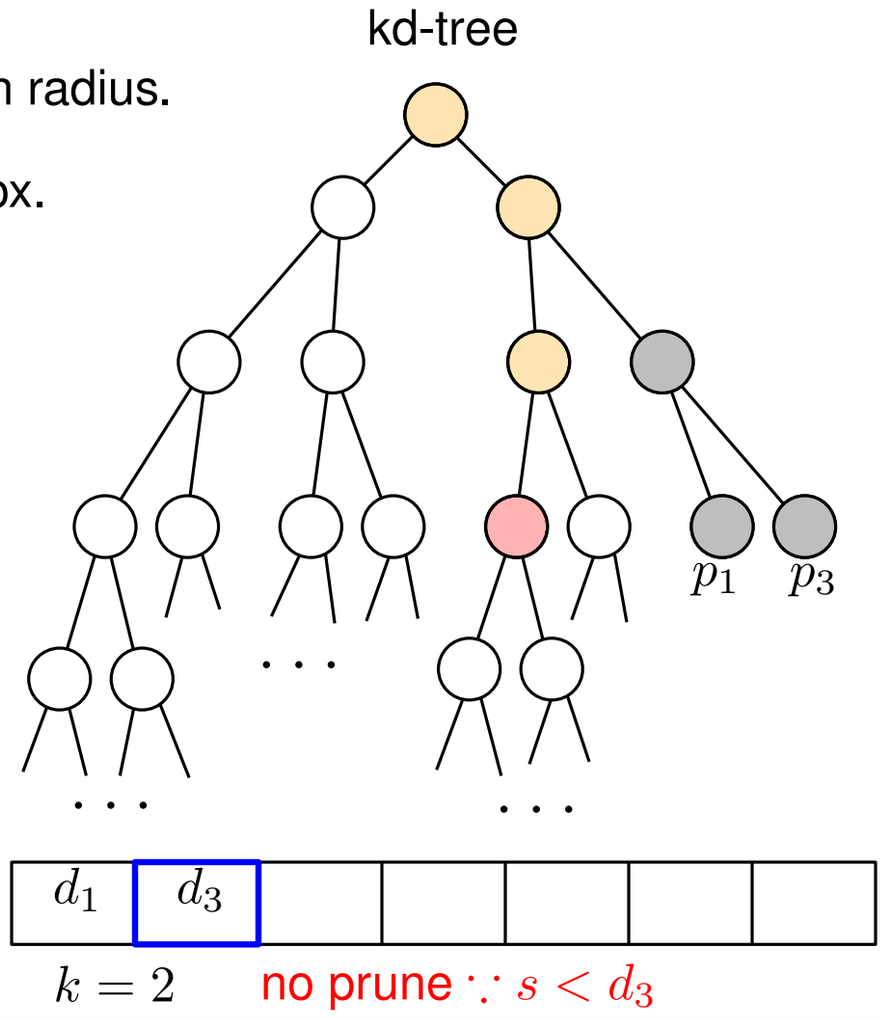


kd-tree and Ball Tree

kNN query in kd-tree

- Follow the split decisions to the leaf reached by the query.
- Find some nearby candidates.
- Use the current k-th nearest distance as a search radius.
- If a box is outside this radius, ignore the whole box.

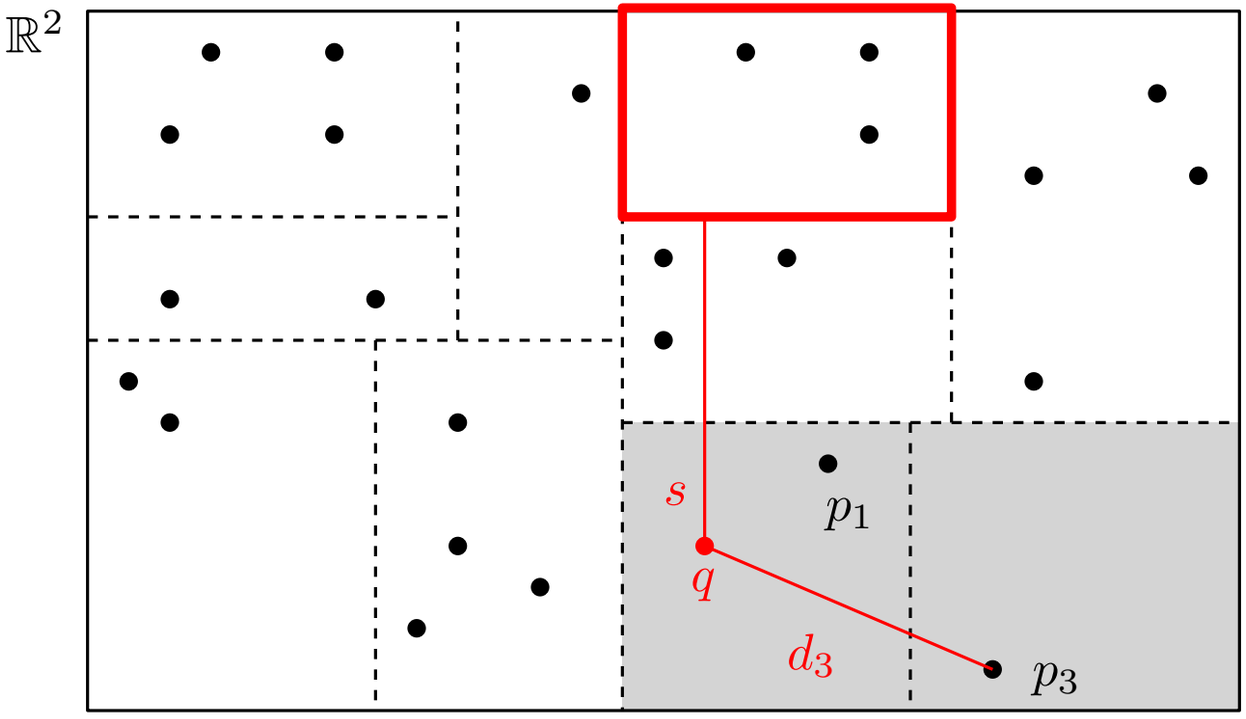


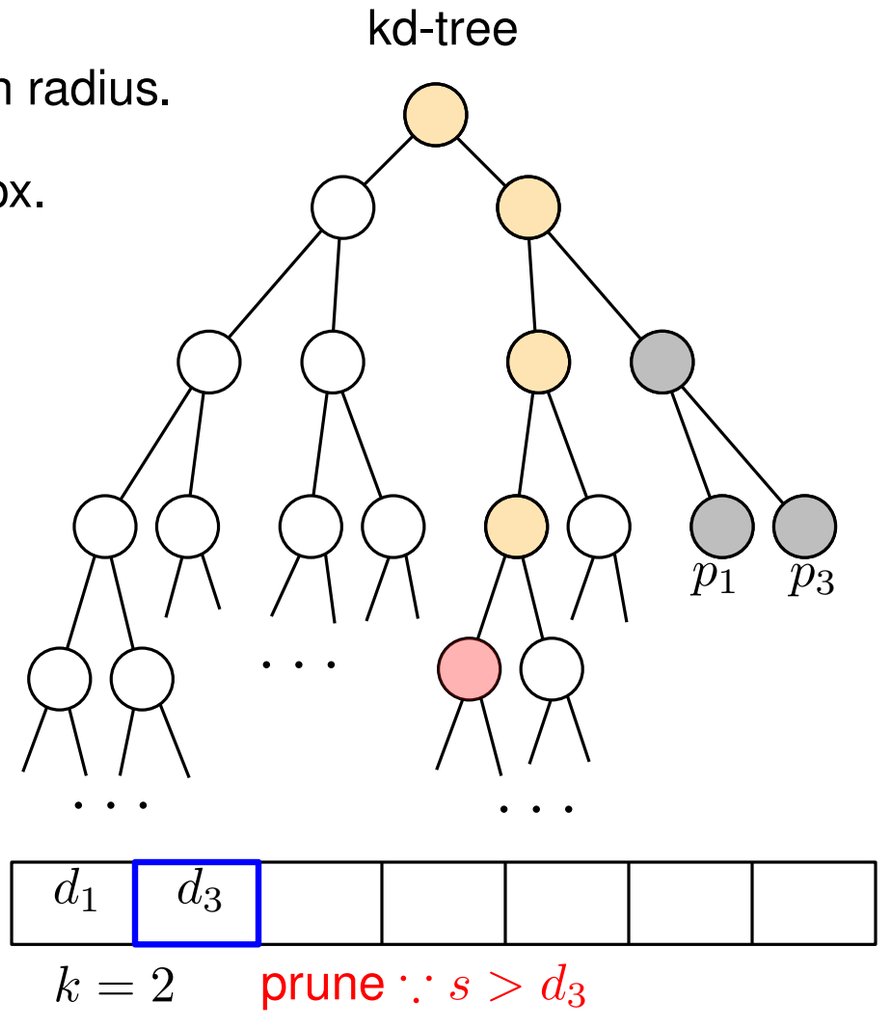


kd-tree and Ball Tree

kNN query in kd-tree

- Follow the split decisions to the leaf reached by the query.
- Find some nearby candidates.
- Use the current k-th nearest distance as a search radius.
- If a box is outside this radius, ignore the whole box.

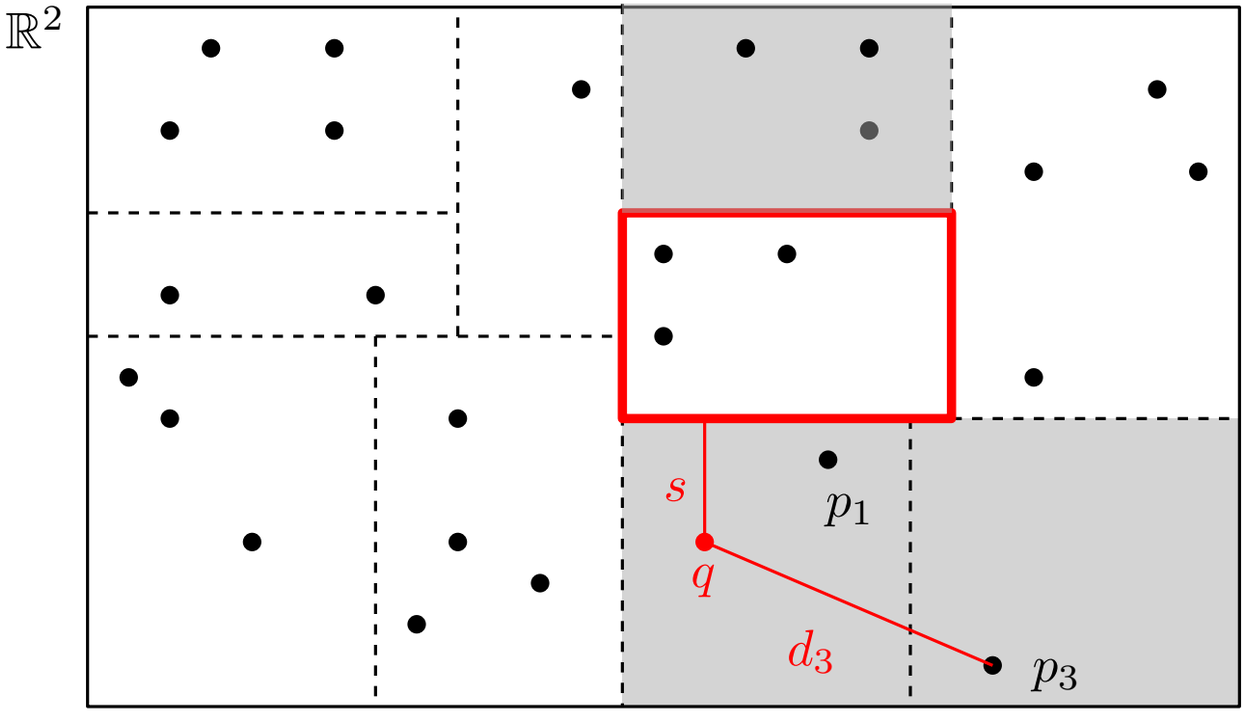


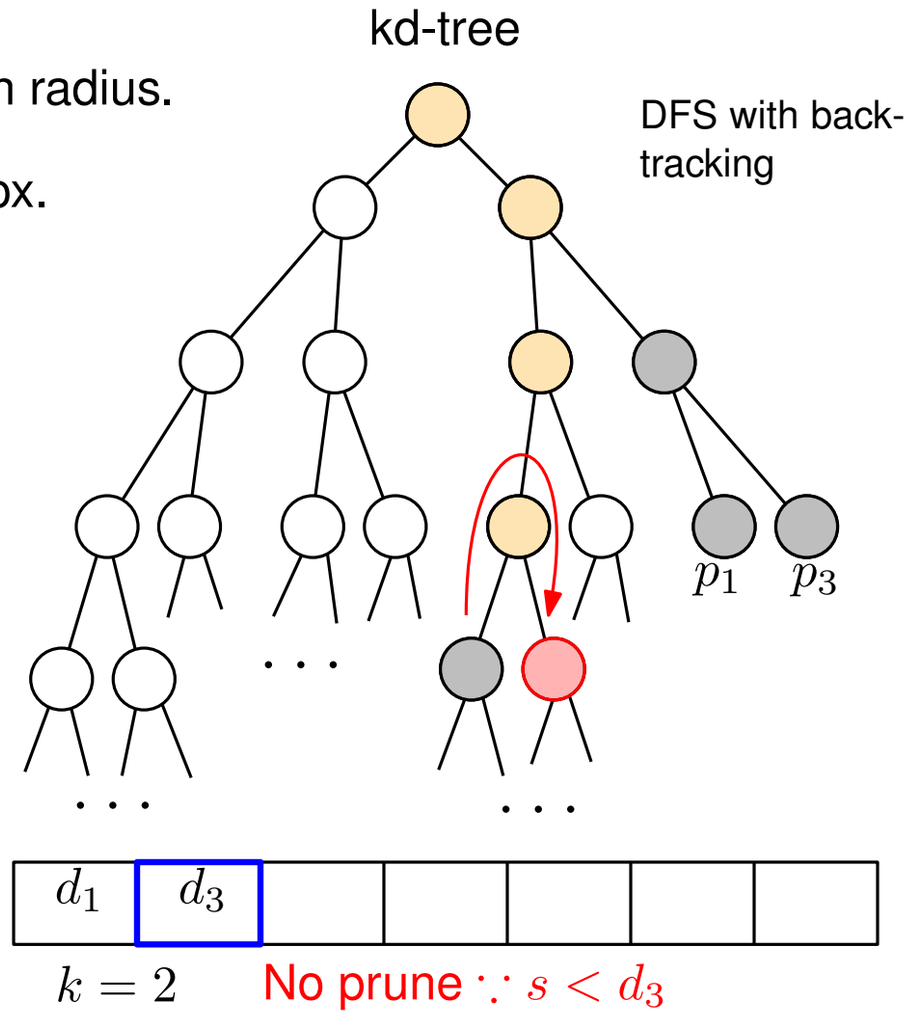


kd-tree and Ball Tree

kNN query in kd-tree

- Follow the split decisions to the leaf reached by the query.
- Find some nearby candidates.
- Use the current k-th nearest distance as a search radius.
- If a box is outside this radius, ignore the whole box.





kd-tree and Ball Tree

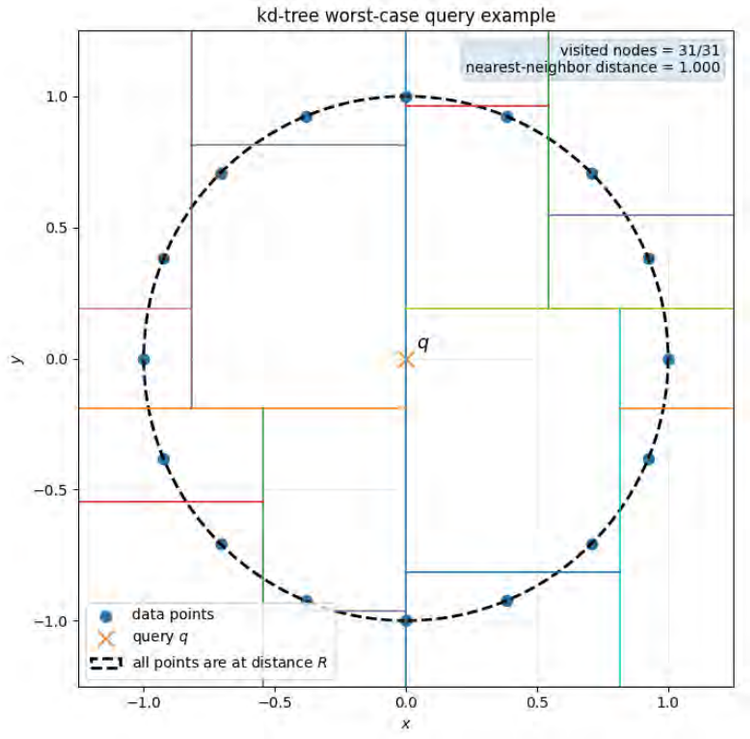
Brute-force vs. kd-tree

Problem	Brute force	kd-tree worst case
single kNN query	$O(nd + n \log k)$	$O(nd + n \log k)$
kNN graph	$O(n^2 d + n^2 \log k)$	$O(n^2 d + n^2 \log k)$
radius join	$O(nmd + z)$	$O(nmd + z)$

using a size-k max-heap

For radius join, $|A| = n$, $|B| = m$, and z is the output size.

- Spatial queries: kd-tree supports range, collision detection,...
- Worst case: Many points have similar distances to q: $d(q, p_i) \approx d(q, p_j)$.
- Then the search may visit all leaves and inspect all points.



kd-tree and Ball Tree

Appendix: Average-case analysis of kd-tree (Friedman et al. 1977; Skrodzki 2019)

- Analyze a single kNN query under a probabilistic model.
- Assumptions: $X_1, \ldots, X_n$ and query q are i.i.d. from a continuous density in $\mathbb{R}^d$; fixed d, k, b; n large; balanced median splits on the widest axis; leaf buckets roughly cubical; density locally smooth near q.
- Median splits give tree depth $O(\log n)$.
- Under these idealized assumptions,

$$\mathbb{E}[\#\text{records}] \leq C_{d,k,b} = \left(k^{1/d} + b^{1/d}\right)^d.$$

- Hence, for fixed d, k, b,

$$\mathbb{E}[T_{\text{query}}] = O(\log n).$$

- Caveat:

$$C_{d,1,1} = 2^d.$$

  Thus, this is a low-dimensional average-case guarantee.

kd-tree and Ball Tree

What if coordinates are not available?

In a general metric space, we may not have

coordinates, axes, boxes, or box distances.

We only have a distance function

$$d : X \times X \to \mathbb{R}_{\geq 0}$$

satisfying

$$d(x, y) = 0 \Leftrightarrow x = y, \qquad d(x, y) = d(y, x), \qquad d(x, z) \leq d(x, y) + d(y, z).$$

So pruning must rely only on distances.

coordinate cuts $\longrightarrow$ distance-based regions

This leads to structures such as ball trees.

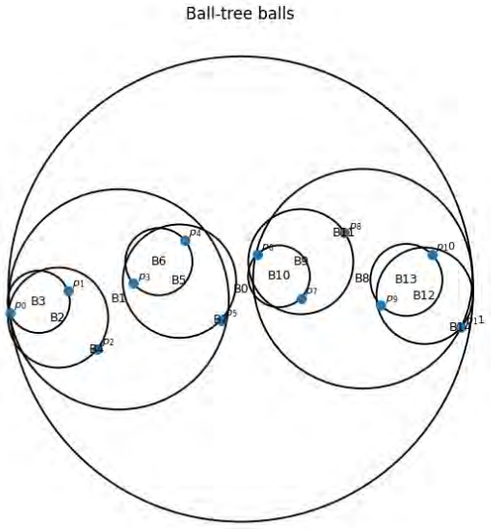
kd-tree and Ball Tree

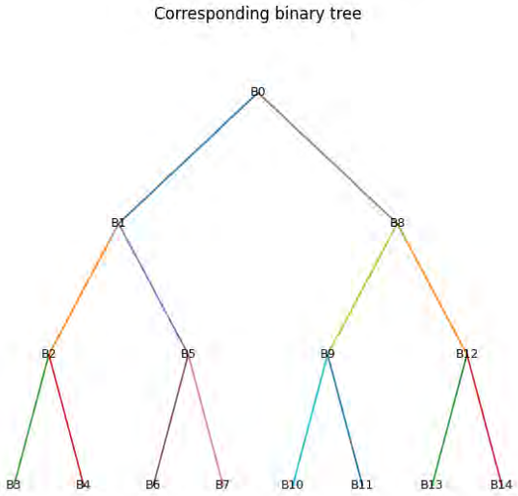
Ball Tree

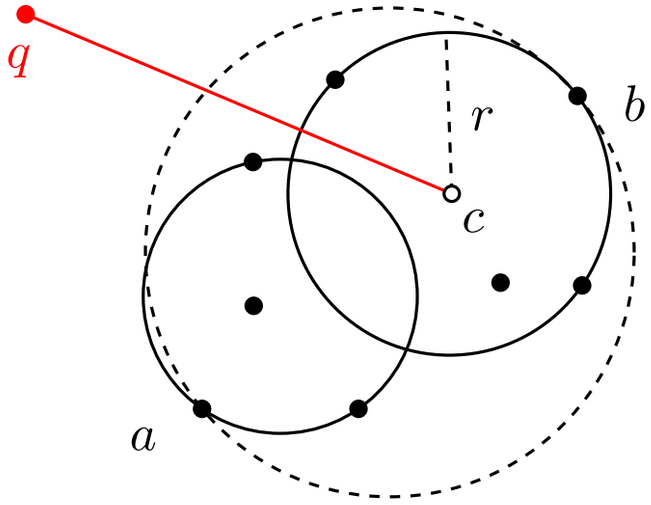
- Node = points + enclosing ball: $S_C \subseteq B(c_C, R_C)$.
- One common split uses two pivots a, b: $p \in S_L$ if $d(p, a) \leq d(p, b)$, otherwise $p \in S_R$.
- Wrap each child group by a ball.
- Lower bound:

$$L(q, C) = \max\{0,\ d(q, c_C) - R_C\}.$$

- Prune if $L(q, C) > r_k$.







Split by two pivots

kd-tree and Ball Tree

Worst-case search cost after build

Problem	Brute force	kd-tree	ball-tree
single kNN query	$O(nd + n \log k)$	same	same
kNN graph	$O(n^2 d + n^2 \log k)$	same	same
radius join	$O(nmd + z)$	same	same

Here the index build cost is ignored.

- **Index build cost**

Brute force	$O(1)$	no index
kd-tree	$O(n \log n)$	cyclic axes
	$O(dn \log n)$	widest-spread axes
ball-tree	$O(dn \log n)$	cheap pivots + balanced split
	$O(dn^2)$	exact farthest-pair pivots

- Once an index is built, it can be reused for repeated queries.

Curse of Dimensionality

Curse of dimensionality in NNS

- The phrase “curse of dimensionality” is used in many ways.
- For tree-based NNS, it means:

$$d \uparrow \quad \Rightarrow \quad \text{pruning collapses.}$$

- kd-tree / ball-tree may become no better than brute force, even around $d \approx 10$–15.
- A common reason is that distances become less separated, and geometric pruning bounds become weaker or more overlapping.

  Distances from a query become similar:

$$d(q, p_i) \approx d(q, p_j) \qquad \text{for many } p_i, p_j.$$

- Result: #visited points $\approx n$.

Recall Lec 1: nearest is not always meaningful. $R_Q(A, B) \not\gg R_Q(A, C)$
Recall Lec 2: kd-tree fails when many points look equally close.

Curse of Dimensionality

Distance concentration

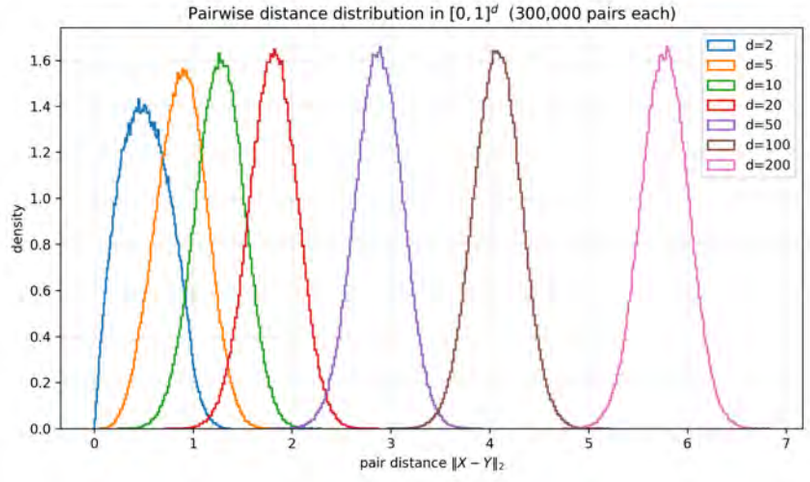
- Let $D^{(d)}_{\min}(q) = \min_{p \in P_d} d(q,p), \qquad D^{(d)}_{\max}(q) = \max_{p \in P_d} d(q,p).$
- In low dimension:

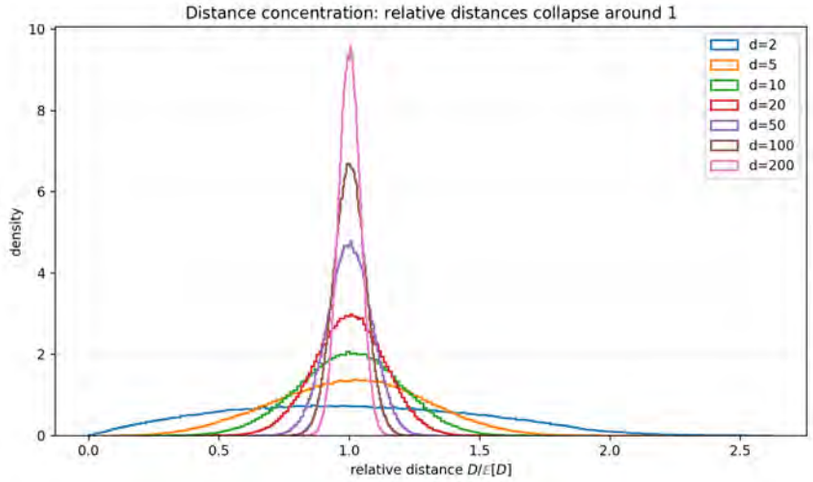
$$D^{(d)}_{\max}(q) \gg D^{(d)}_{\min}(q). \qquad \rightarrow \text{clearly separated}$$

- Under specified concentration assumptions on a sequence of data distributions, in high dimension:

$$\frac{D^{(d)}_{\max}(q) - D^{(d)}_{\min}(q)}{D^{(d)}_{\min}(q)} \xrightarrow{p} 0. \qquad \rightarrow \text{hard to distinguish}$$

- This is not a universal property of all high-dimensional datasets.

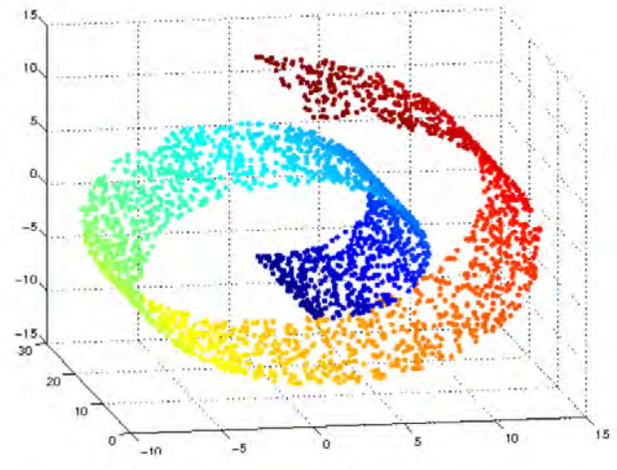




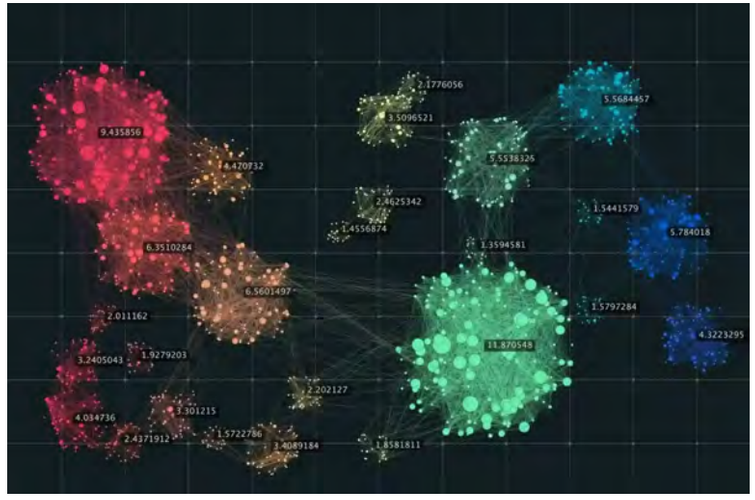
Curse of Dimensionality

Why Can Proximity Search Remain Useful on Structured Data?

- Real data may have large ambient dimension: $P \subset \mathbb{R}^d$, d large.
- But they rarely fill $\mathbb{R}^d$ uniformly.
- **Manifold hypothesis:** data lie near a low-dimensional structure.
- **Clustered structure:** data form separated groups.
- These structures weaken distance concentration.



Manifold hypothesis



Clustered data

Intrinsic Dimension

Intrinsic dimension

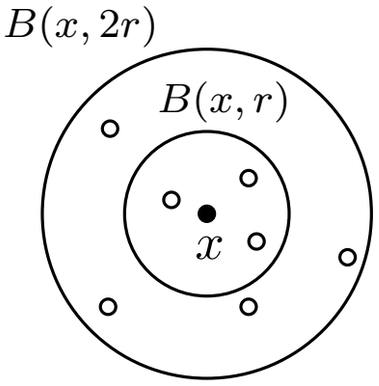
- Ambient dimension d can be misleading.
- Intrinsic dimension describes the effective geometry occupied by the data.
- Intrinsic-dimension notions:

manifold	data lie near a low-dimensional surface
low-rank	data vary along a few main directions
growth-based	nearby points do not explode too fast

- Other helpful structures: clustering, data density, anisotropy.
- For tree-based NNS, expansion dimension is especially relevant.

$$|B_P(x, 2r)| \leq c\,|B_P(x, r)|, \qquad d_{\exp} = \log_2 c.$$

- Small $d_{\exp}$: slow neighborhood growth $\Rightarrow$ pruning can still work.



uniform d-dimensional data
$\Rightarrow d_{\exp} \approx d$.

Intrinsic Dimension

Appendix: Cover tree

- Cover tree was introduced by Beygelzimer, Kakade, and Langford (2006).
- Cover tree is a metric-space index: uses distances, not coordinates.
- It organizes points by scales: coarse cover $\rightarrow$ fine cover.
- Expansion constant c gives the expansion dimension:

$$d_{\exp} = \log_2 c.$$

- The original paper claimed logarithmic-time exact 1-NN search in low-expansion metrics.
- However, Curtin's PhD thesis (2015) pointed out a gap in the proof.
- Elkin and Kurlin (2022) later constructed explicit counterexamples to the original complexity proof.

Parallel Search on Hardware

Geometry saves work; parallelism makes work cheap.

Two basic parallelization strategies

data parallelism	**task parallelism**
same operation	independent subproblems
on many data items	run concurrently

Techniques we will use

1. SIMD / SIMT execution	low-level data-parallel lanes and warps
2. Task decomposition	split independent work across execution units
3. GPU-oriented decomposition	batch work, regularize memory, reduce results
4. Multi-GPU decomposition	split queries or data across devices

Parallel Search on Hardware

- **SIMD:** one instruction, many data lanes.

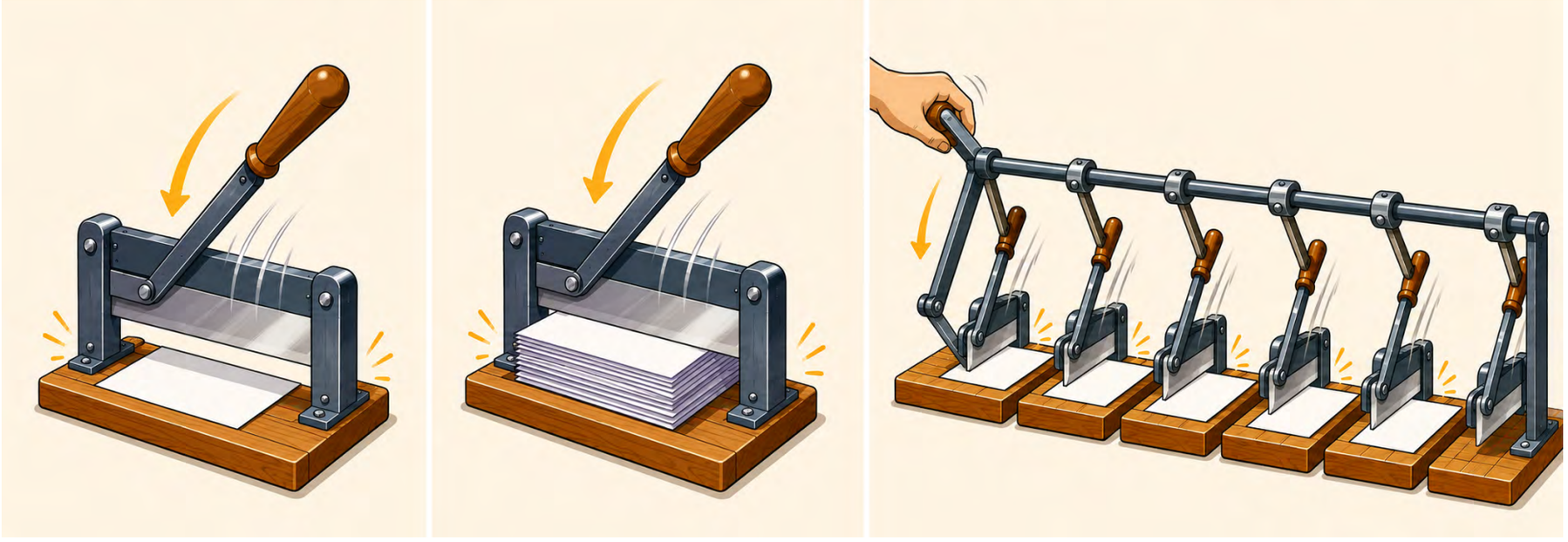
$$(q_1 - x_1)^2, \ldots, (q_w - x_w)^2$$

inside one distance computation

- **SIMT:** many GPU threads, one warp instruction stream.

$$d(q_i, x_j) \quad \text{for many pairs } (i, j)$$

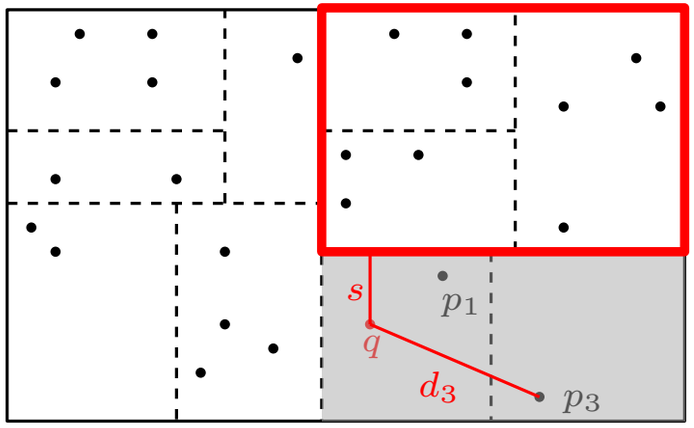
across many point pairs

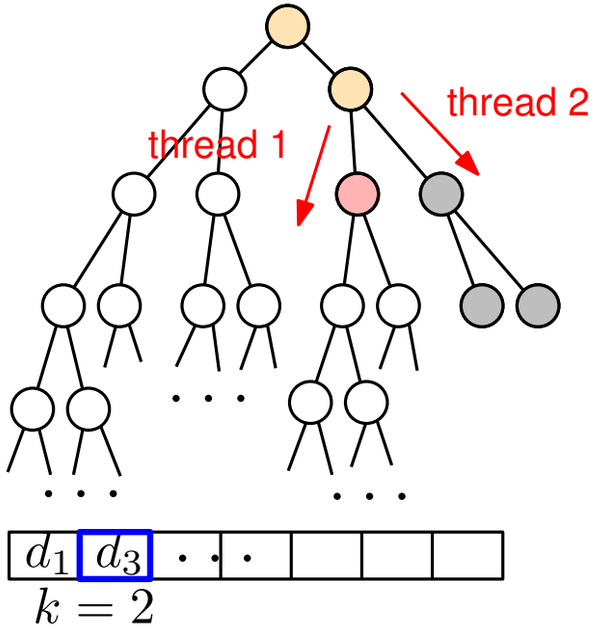


Parallel Search on Hardware

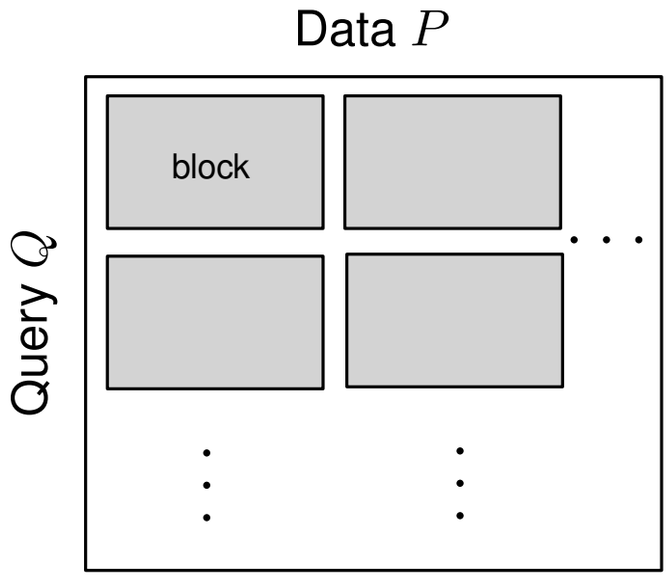
Task decomposition: split work into independent tasks.

- Query-level: different queries run independently.
- Domain / data level: split points or space into chunks, cells, leaves, or subtrees.





kd-tree



maxtrix $Q \times P$

Parallel Search on Hardware

GPU-oriented brute force

- Brute force has two stages:

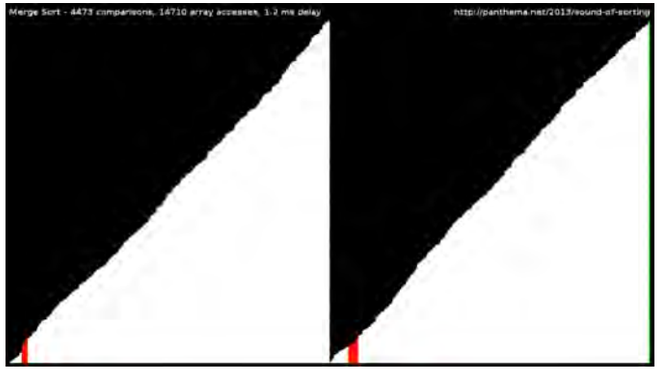
  compute all distances $\rightarrow$ keep the k smallest distances for each query.

- Distance stage:

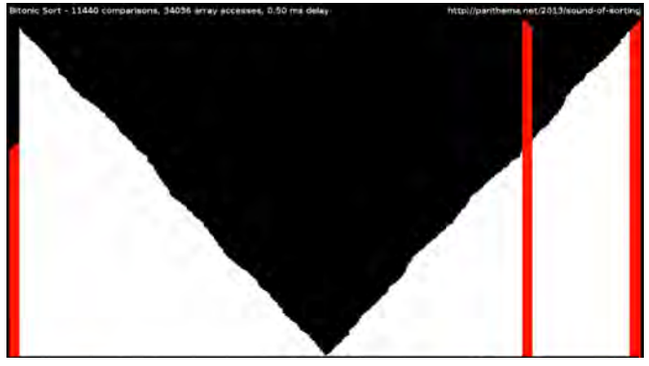
$$\|q - p\|^2 = \|q\|^2 + \|p\|^2 - 2q^\top p$$

  Compute many dot products using tiled, matrix-multiplication-like GPU kernels.

- Selection stage: for each query, keep only the nearest k references. Avoid sequential CPU heaps; use parallel sorting, merging, or reductions.



Merge sort



Bitonic sort

https://www.youtube.com/watch?v=kPRA0W1kECg

Parallel Search on Hardware

Memory and multi-GPU

- LEXICON V100: 32GB GPU memory.
- Materializing the full distance matrix often dominates memory use.

$$D \in \mathbb{R}^{|Q| \times |P|}.$$

The points may fit, but the full matrix D usually does not.

- Therefore, process queries in batches by partitioning Q:

$$Q = \bigsqcup_b Q_b, \qquad D_b \in \mathbb{R}^{|Q_b| \times |P|}.$$

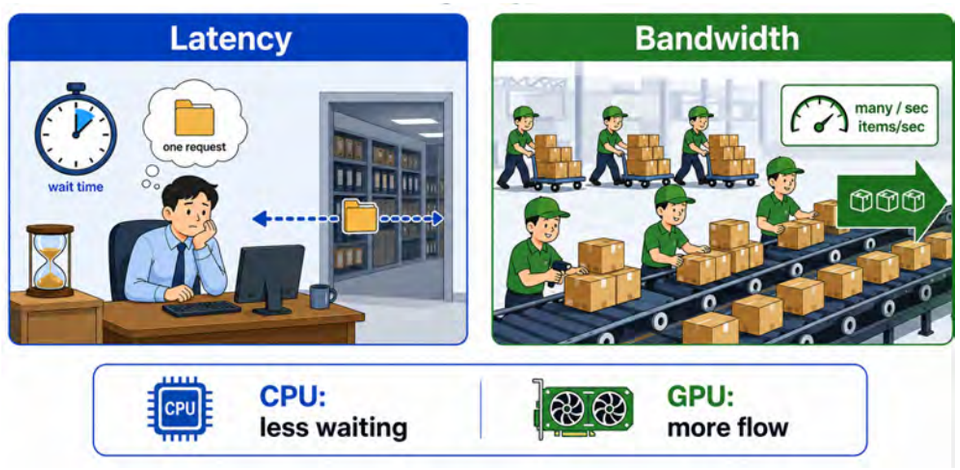
After computing each distance block D_b, immediately perform local top-k, so the full D need not be stored.

- In practice, on LEXICON V100, $|Q_b| \approx 5000$–6000 is feasible, depending on $|P|$, k, dtype, and workspace.
- Multi-GPU: assign different query batches to different GPUs. If P is split across GPUs, merge only local top-k candidates.

Hardware Bottlenecks and Profiling

Appendix: Memory hierarchy: CPU vs GPU

Level	CPU	GPU
Fastest	Registers	Registers
	L1 cache	Shared memory / L1 cache
	L2 cache	L2 cache
	L3 cache	—
Slowest	DRAM	HBM / GDDR

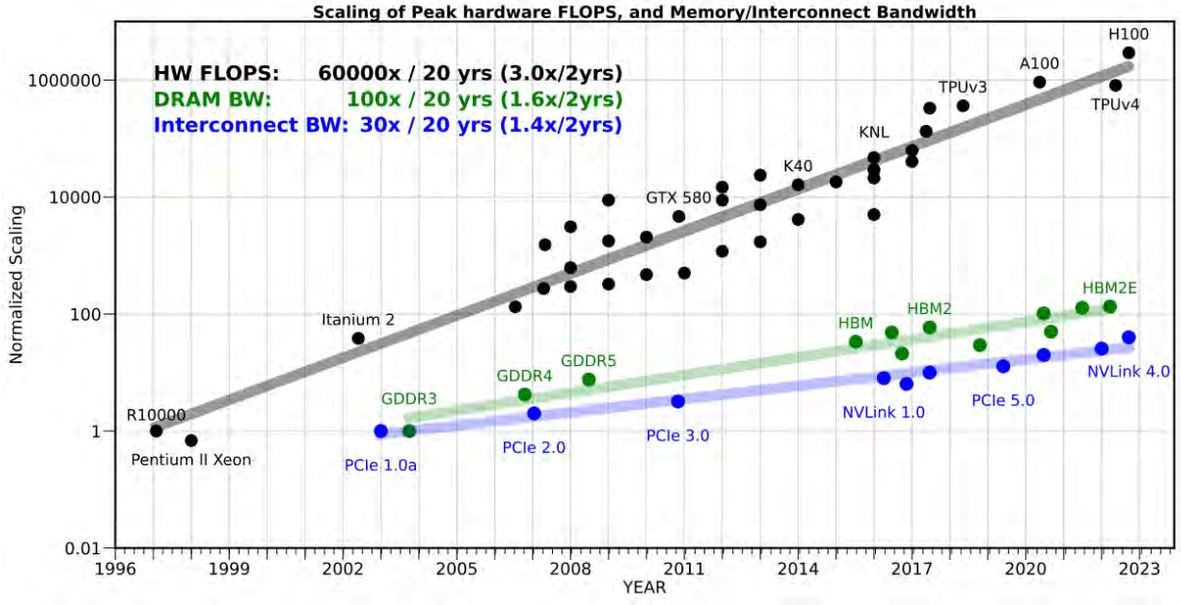


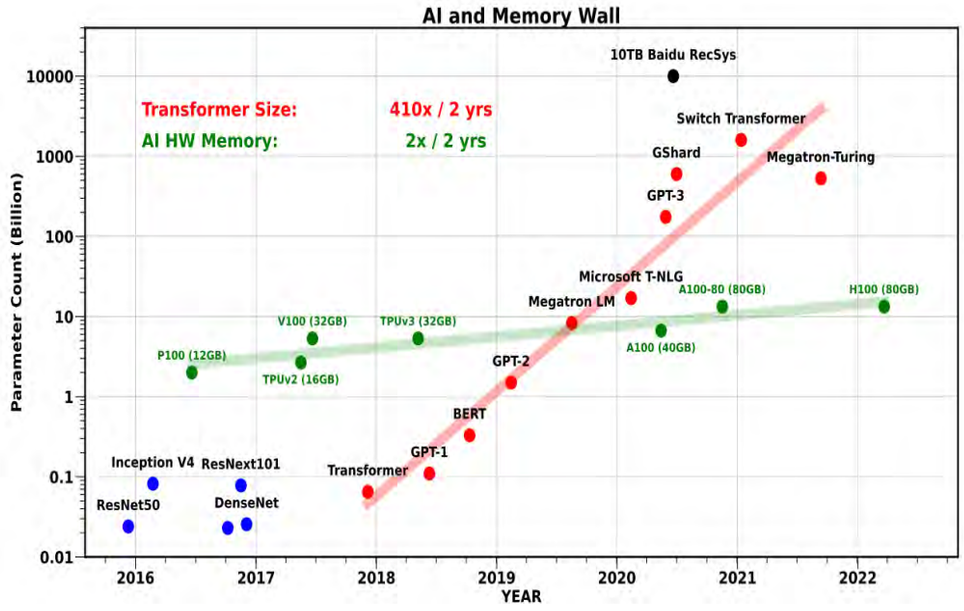
- Closer memory is faster, smaller, and more expensive.
- CPU: reduce **latency** for a few powerful cores.
- GPU: maximize **bandwidth** for many parallel threads.
- CPU performance suffers from cache misses, page faults.
- GPU performance suffers from scattered global memory access.
- Performance is often limited by data movement, not arithmetic.

Hardware Bottlenecks and Profiling

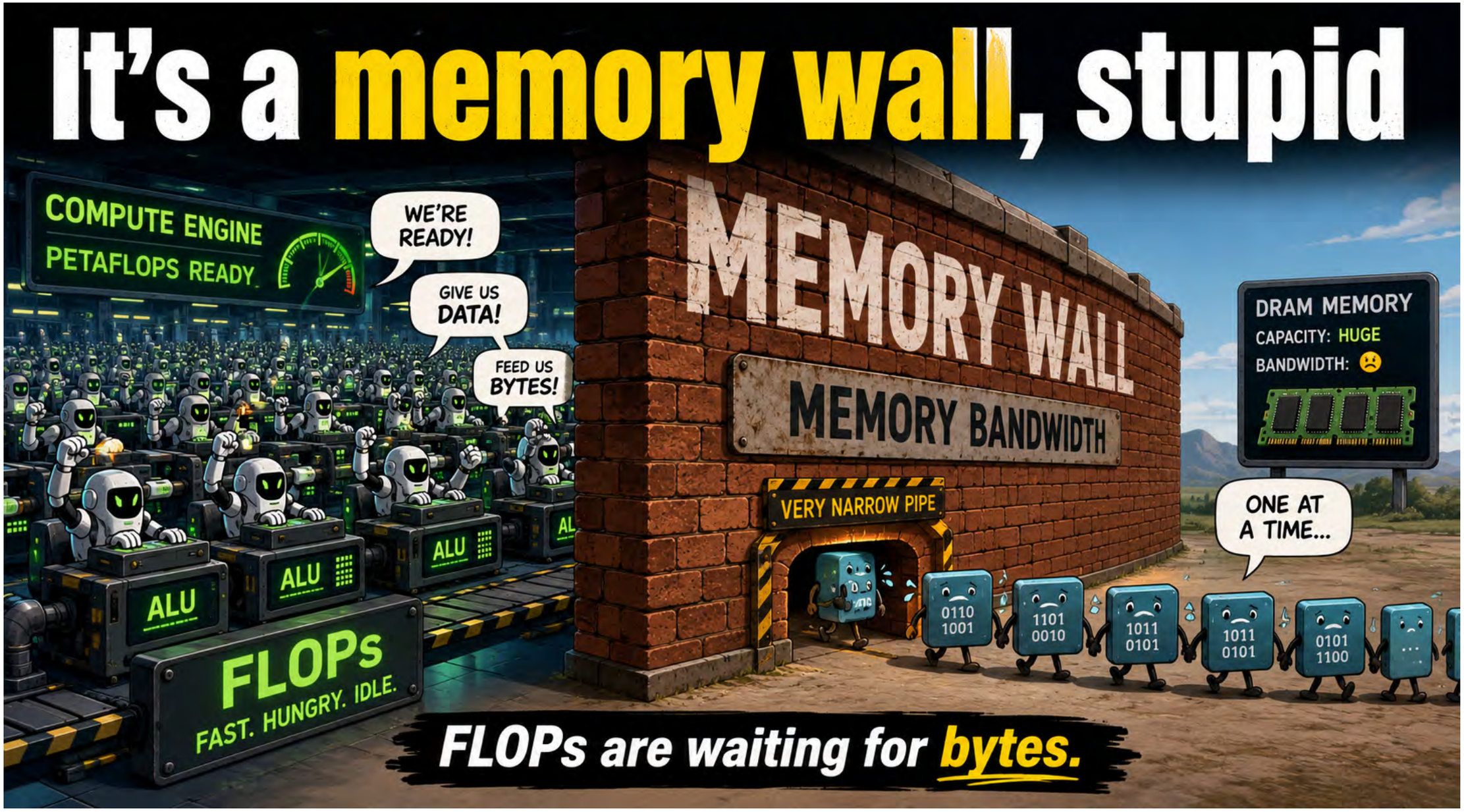
Appendix: Memory Wall

- Peak compute throughput has grown rapidly.
- Memory and interconnect capabilities have also improved, but often not at the same rate.
- In many modern accelerator workloads, data movement becomes an increasingly important bottleneck, not just arithmetic.





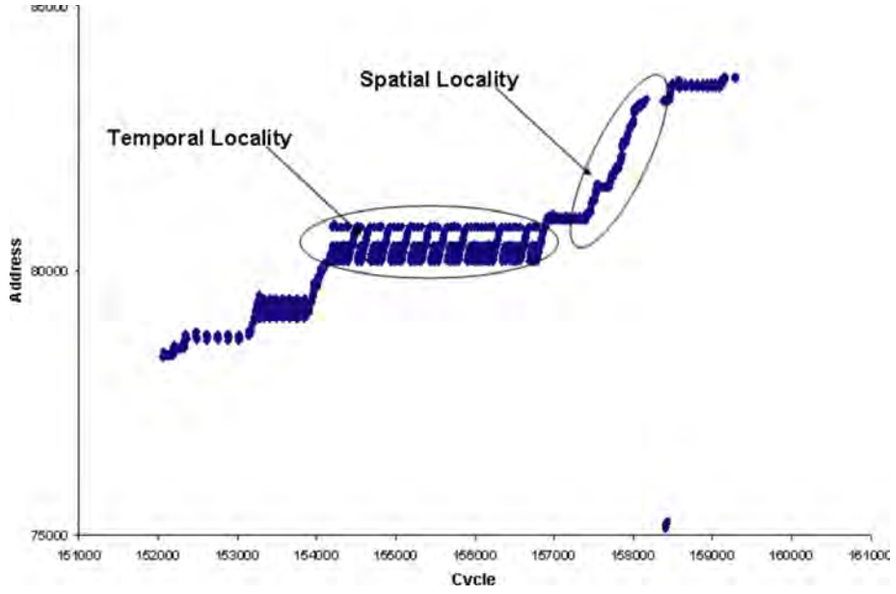
Appendix: Memory Wall



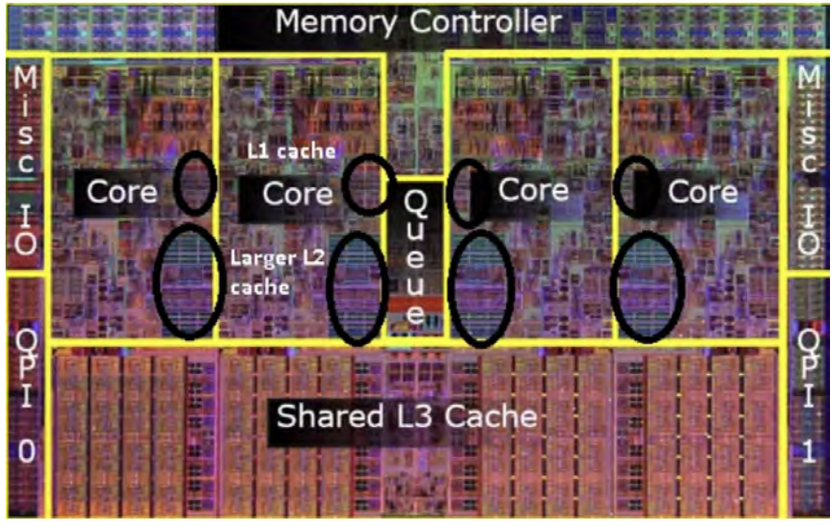
Hardware Bottlenecks and Profiling

Appendix: Memory locality

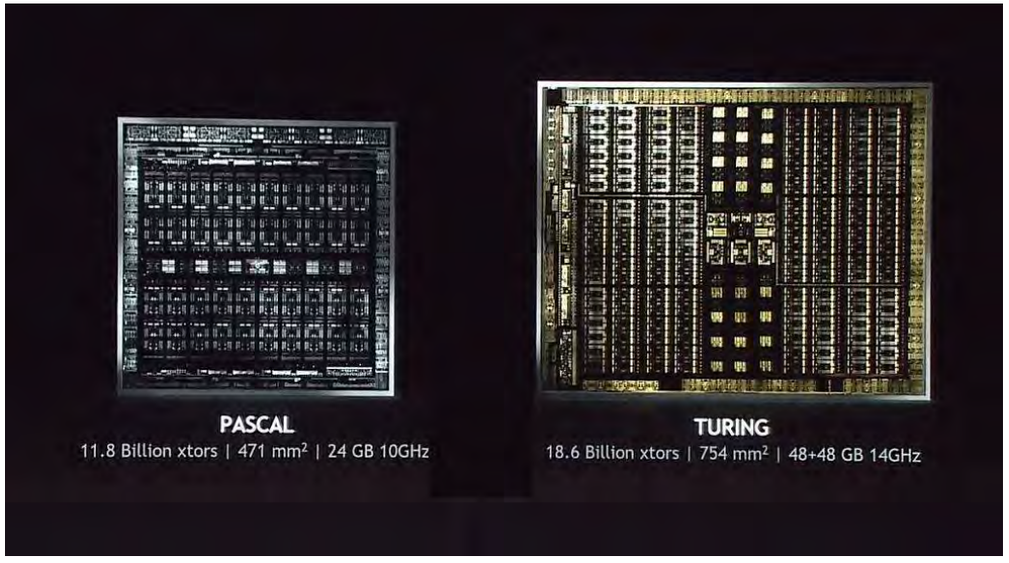
- Hardware has a physical geometry.
- Nearby data are cheaper to access $\rightarrow$ fast.
- Temporal locality: reuse soon.
- Spatial locality: access neighbors together.
- Algorithm design should follow this geometry.



Memory access locality



CPU architecture



GPU architecture

Hardware Bottlenecks and Profiling

Appendix: Two GPU performance caveats

- **Warp divergence.** A warp executes many threads together. If some take different branches, (`if / else`), then paths are serialized, and some lanes become inactive.

  regular work $\Rightarrow$ high throughput, irregular branches $\Rightarrow$ lost parallelism.

- **Cold vs. warm timing.** The first GPU run may include extra overhead:

  CUDA context setup, kernel loading/JIT, memory allocation.

- For fair profiling, run once to warm up, then measure repeated runs:

  cold run $\neq$ steady-state performance.

Warm-up Run

Agentic Coding for Performance Engineering

Appendix: LLMs still struggle with systems optimization
(Jing et al. 2026, PerfCodeBench.)

- Correct code is not necessarily fast code.
- PerfCodeBench evaluates executable system-level optimization, not only algorithmic problem solving.
- Each task compares:

$$\text{baseline code} \quad \rightarrow \quad \text{LLM-optimized code} \quad \text{vs.} \quad \text{expert reference.}$$

- System-level efficiency depends on hardware-aware choices:

$$\text{cache locality}, \quad \text{data movement}, \quad \text{parallel execution}, \quad \text{synchronization}.$$

- Even the relatively latent model reaches or beats the expert reference on only about 61.6% of comparable tasks.
- CUDA was the most difficult language subset in this PerfCodeBench evaluation.

Agentic Coding for Performance Engineering

Appendix: PerfCodeBench: CPU code vs. CUDA kernels

Metrics

Metric	Meaning
CRR	correct and runnable
FBR	faster than baseline
RBR	reaches expert speed
CGRE	close to expert speed

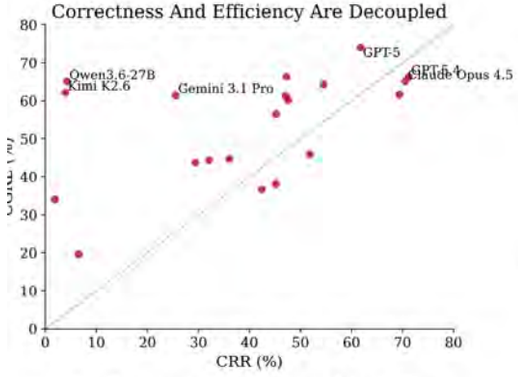
$\mathrm{CGRE} = 0$: no valid speedup
$\mathrm{CGRE} = 1$: expert-level or better

Language subsets

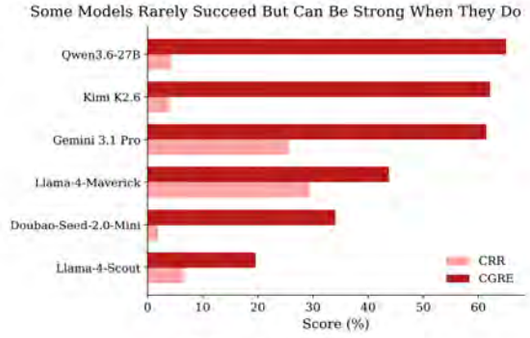
Subset	Task share	Avg. CRR	Avg. CGRE	Best CGRE model
C	8.74	45.16	31.33	GPT-5.4
C++	58.63	74.73	69.20	GPT-5.4
Go	8.63	76.67	71.14	Claude Opus 4.5
Java	9.49	77.27	71.36	Claude Opus 4.5
Python	7.77	85.53	71.87	DeepSeek-V4-Flash
CUDA	6.74	10.53	6.97	DeepSeek-V4-Pro

CPU–GPU gap

Model	CPU CRR	GPU CRR	CPU CGRE	GPU CGRE
GPT-5.4	82.06	2.40	68.67	0.00
GPT-5	71.02	9.60	64.74	6.70
Claude Opus 4.5	79.68	12.00	69.47	10.26
Claude Sonnet 4.5	75.44	9.60	60.65	5.76
DeepSeek-V4-Pro	62.04	18.40	57.18	11.99
DeepSeek-V4-Flash	61.00	11.20	57.18	7.12



(a) Correctness–efficiency decoupling.



(b) Rare successes.

high correctness rate

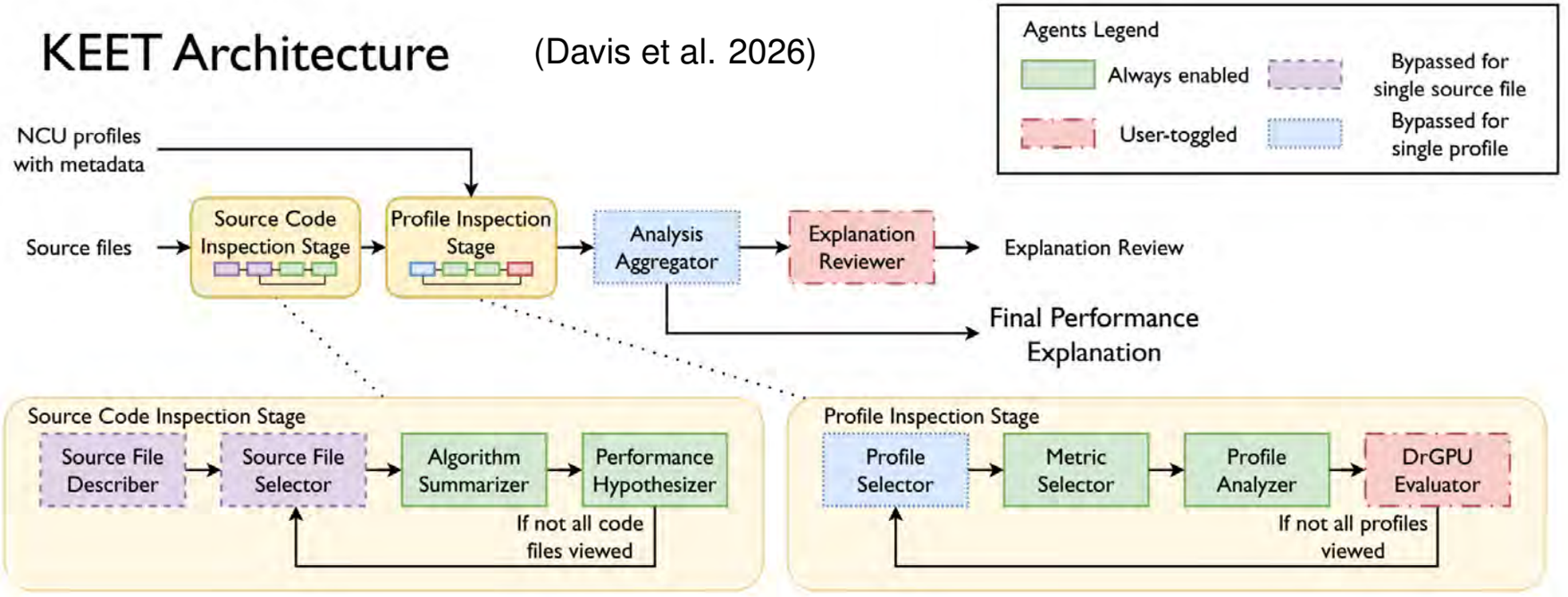
⇓

high runtime efficiency

Agentic Coding for Performance Engineering

Appendix: Agentic coding for performance engineering

- Input: CUDA source code + Nsight Compute profiles.
- Output: data-grounded explanations of bottlenecks and optimization suggestions.
- Key idea: use LLM agents to turn profiling data into actionable performance advice.
- Recent systems: KEET, PRAGMA, CUDA Agent, CudaForge, Opal, and NCU Copilot



Nearest Neighbor Search for Scientific Computing

Lecture 3

Nearest Neighbors in Scientific Computing

Jaehoon Chung
Research Fellow, Center for AI and Natural Sciences
Korea Institute for Advanced Study
sk7755@kias.re.kr

Contents

Lecture 3: Nearest Neighbors in Scientific Computing

- **Nearest neighbors across scientific domains**
  Different domains lead to different data spaces, distance notions, and search tasks.

- **Molecular dynamics: locality is interaction**
  Cutoff neighbor search as a radius join; cell lists, Verlet lists, and LBVH.

- **Astronomy: proximity suggests identity**
  Catalog cross-matching with angular distance; HEALPix and HTM.

- **Single-cell genomics: neighbors define cell states**
  kNN graph construction; NN-descent.

- **Information retrieval: meaning becomes geometry**
  High-dimensional embedding search with approximation; LSH and HNSW, IVF-PQ

Nearest neighbors Across Scientific Domains

https://cac.kias.re.kr/2026

이 홈페이지를 참고해서, 참가자의 전공을 추론해줘.

참가자 분야 구성

분야군	학생 참가자 $(n = 49)$	전체 참가자 $(n = 118)$
천문 / 우주과학 / 우주탐사	12명 (24.5%)	19명 (16.1%)
물리 / 양자 / 광과학	13명 (26.5%)	22명 (18.6%)
소재 / 화학·바이오 / 의학 / 신	9명 (18.4%)	15명 (12.7%)
수학 / 통계 / 수리과학	8명 (16.3%)	15명 (12.7%)
AI / CS / 계산과학 / 산업공학	6명 (12.2%)	17명 (14.4%)
심리학 / 과학교육	–	3명 (2.5%)
소속만으로 판단 어려움	1명 (2.0%)	27명 (22.9%)
합계	49명 (100%)	118명 (100%)

Nearest Neighbors Across Scientific Domains

Nearest-neighbor search offers a rich toolbox.

- Different tasks: kNN query, radius join, all-kNN graph.
- Different guarantees: Exact vs. approximate; deterministic vs. randomized or probabilistic guarantees
- Different data geometry: intrinsic dimension, metric, high-dimensional.
- Different scale and hardware: CPU, GPU, distributed systems.

Nearest Neighbors Across Scientific Domains

Representative Algorithmic Families

Family	**Idea**	**Examples**
Brute force	Compute all distances	parallel Bruteforce
Tree-based	Prune regions	kd-tree, ball tree, cover tree
Grid-based	Search nearby cells	Cell list, HEALPix
Hash-based	Similar points collide	MinHash, cross-polytope LSH, LSF
Graph-based	Navigate neighbors	HNSW, NSG, DiskANN
Quantization-based	Compress vectors	PQ, OPQ, AQ, LSQ
Partition-based	Search promising clusters	IVF, k-means tree, inverted multi-index
Hybrid systems	Combine multiple ideas	IVF-PQ, HNSW-PQ, ScaNN

These are only representative families. Even within each scientific domain, many more specialized methods are used.

Nearest Neighbors Across Scientific Domains

Which NNS tool should we use?

Scientific domains use different versions of nearest neighbor search.

Geometry $\times$ scale $\times$ output type $\times$ exactness determines the toolbox.

Domain	Problem	Data regime	Metric or score	Exact?
Molecular dynamics	cutoff neighbor search	$d = 3$, periodic box, $n \approx 10^5$–10^9	Euclidean / min-image	exact
Astronomy	catalog cross-match	sphere S^2, $n \approx 10^6$–10^{10}	angular / great-circle	usually exact
Single-cell genomics	kNN / SNN graph	PCA space, $d \approx 20$–50, $n \approx 10^4$–10^7	Euclidean / cosine	usually approx.
Information retrieval	top-k vector search	dense vectors, $d \approx 10^2$–10^3, $n \approx 10^6$–10^{11}	cosine / inner product	usually approx.

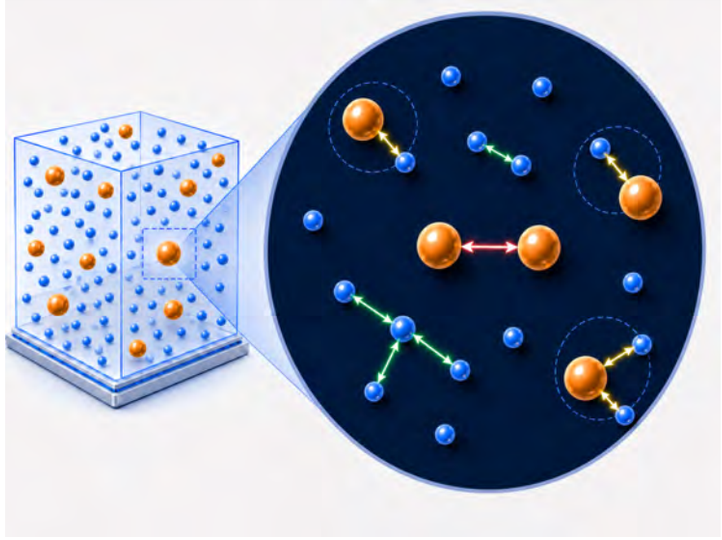
Molecular Dynamics: locality is interaction

- Atoms move in a 3D periodic box.
- Short-range forces are truncated beyond cutoff radius/radii.
- The cutoff is chosen by the physical model and accuracy–cost tradeoff.
- MD builds a Verlet list: candidate pairs within

$$r_{\text{list}} = r_{\text{cut}} + r_{\text{skin}}.$$

$\Rightarrow$ Radius join!

- The list is reused for several timesteps, then rebuilt.



Per-particle Verlet lists

Particle	Verlet list
A	D, B, E, F
B	A, C
C	B, D, E, G
D	A, C, F
E	A, C, G, H, D
F	A, D, G
G	C, E, H, F
H	E, G

Molecular Dynamics: locality is interaction

Cell list: divide space into small boxes

- Partition the simulation box into grid cells.
- Put each atom into its cell.
- To find neighbors of an atom, check only atoms in nearby cells.
- This is likely to avoids testing all $O(n^2)$ atom pairs.

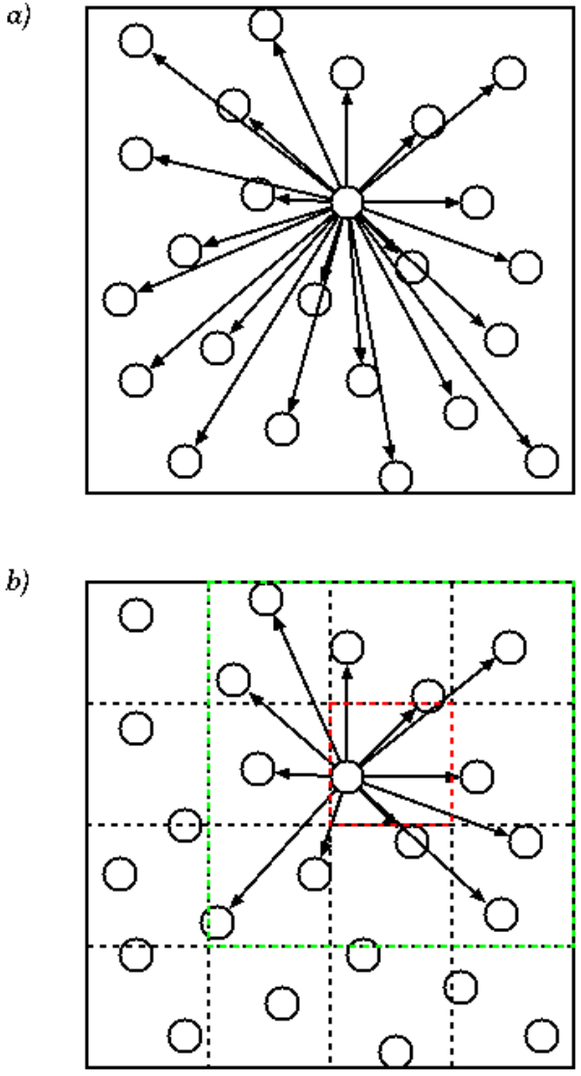
Amortized view

If the list is rebuilt every τ timesteps, then

$$\text{amortized rebuild cost per step} \approx \frac{C_{\text{build}}}{\tau}.$$

At fixed density and fixed cutoff,

$$C_{\text{build}} = O(n), \qquad \text{so amortized cost} = O(n/\tau).$$



Cell list

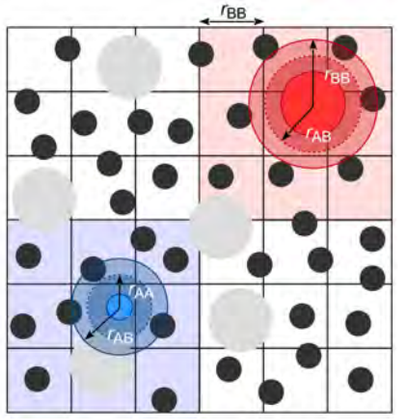
Molecular Dynamics: locality is interaction

Appendix: More advanced neighbor-list builders

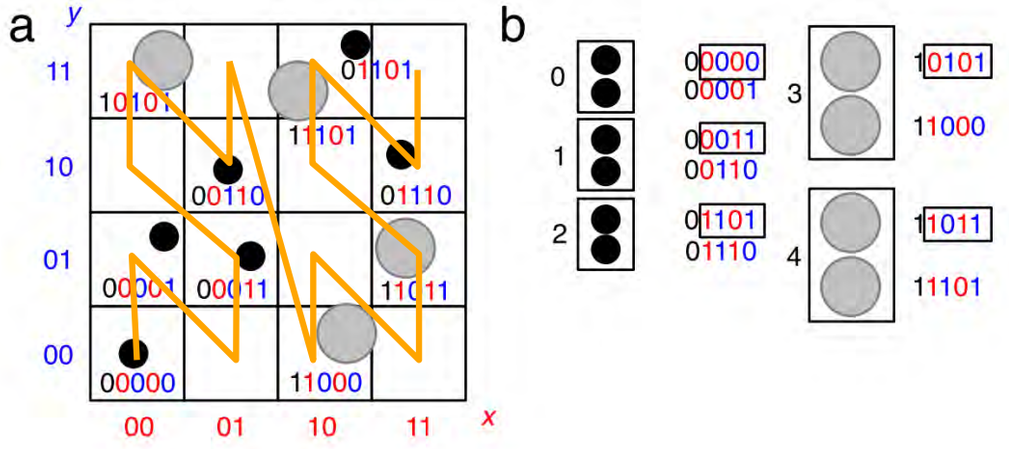
- **Stencil cell list:** precompute type-dependent cell-offset stencils and minimum cell distances to skip unnecessary distance checks.
- **LBVH:** sort particles by a type-Morton key and build one GPU-friendly bounding-volume hierarchy per particle type. (Howard et al. 2016)

The authors claimed.
But..

Method	Good when
Cell list	Uniform density; nearly equal cutoffs
Stencil cell list	Cutoff disparity, but not too sparse in the large-cutoff type
LBVH	Moderate/large disparity with dilute large particles; nonuniform or clustered systems



Stencil cell list

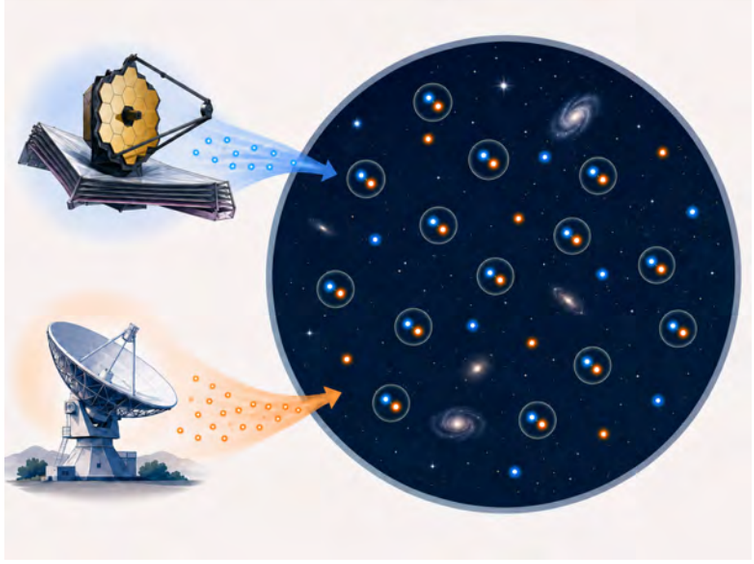


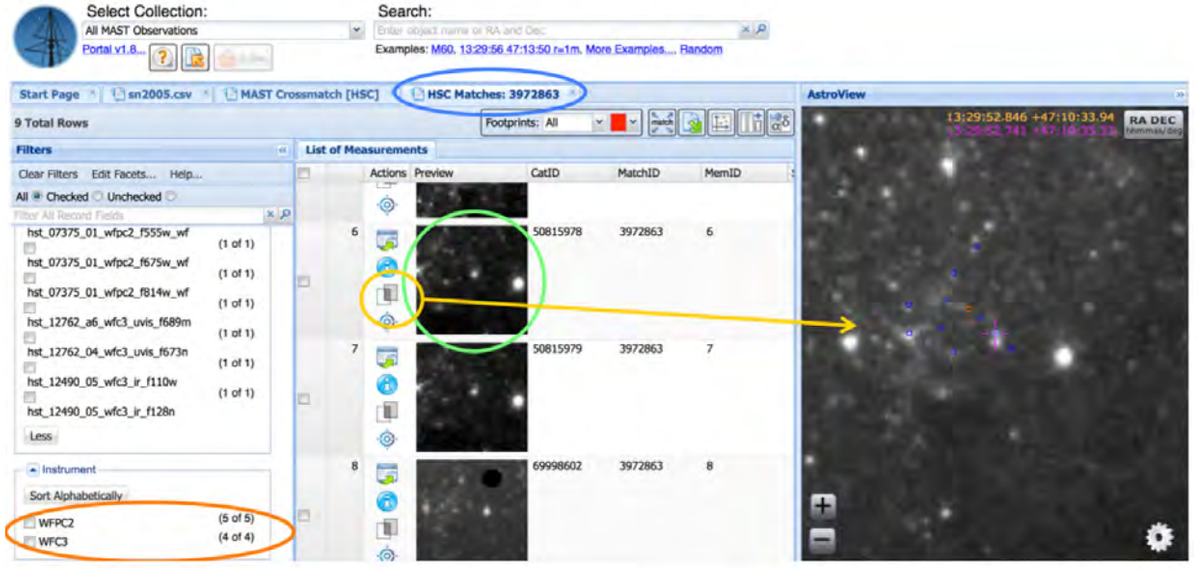
Morton code with Z-order curve

Astronomy: proximity suggests identity

- Astronomical surveys produce catalogs of sources on the sky.
- Source positions are points on the celestial sphere: $A_i, B_j \in S^2$.
- Cross-matching asks whether detections in two catalogs refer to the same object.
- A small angular tolerance r defines candidate matches.
- Thus the computational task is to find nearby pairs on the sphere:

$$\mathrm{angdist}(A_i, B_j) \leq r. \qquad \Rightarrow \text{Radius join!}$$



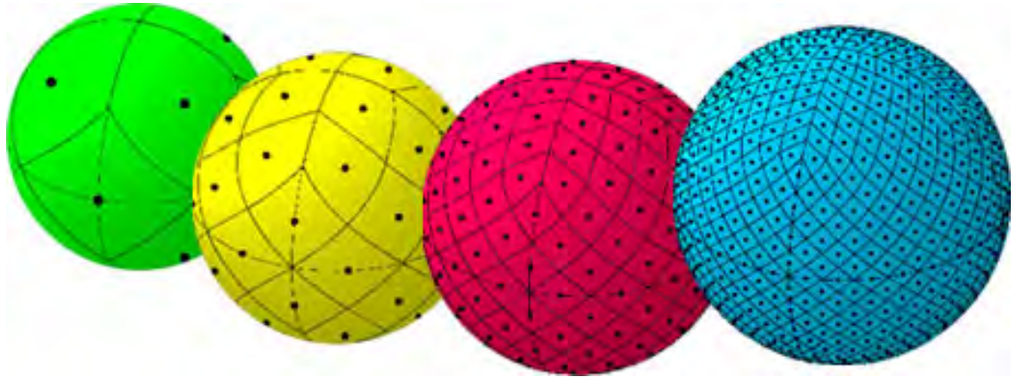


https://archive.stsci.edu/hst/hscv2/help/hsc_use_case_4.html

Astronomy: proximity suggests identity

HEALPix for spherical grid search

- HEALPix divides the sphere into hierarchical equal-area pixels.
- Sources are assigned to HEALPix pixel IDs.
- Cross-matching checks the same or neighboring pixels first.
- This generates candidate pairs, followed by exact angular filtering.
- Other spherical indexes also exist, such as HTM and Q3C.



HEALPix

HTM

Q3C

Astronomy: proximity suggests identity

Appendix: Spherical indexing methods (Nádvorník et al., 2015)

- HEALPix, HTM, and Q3C all partition the celestial sphere into index cells.
- They are used to turn angular search into cell lookup plus exact filtering.

	HEALPix	**HTM**	**Q3C**
Partition	Hierarchical equal-area pixels	Hierarchical spherical triangles	Quad-tree cube pixels
Resolution	Controlled by `Nside`	Controlled by tree depth	Fixed hierarchical encoding
Cell area	Equal area at fixed resolution	Areas can vary across the sphere	Areas can vary across the sphere
Search style	Pixel lookup and neighboring pixels	Recursive traversal of trixels	Database index lookup on sky pixels
Implementation fit	Uniform all-sky chunking; parallel tasks	Cone search; recursive spatial indexing	PostgreSQL cone search and cross-match
Caveat	Resolution must be tuned	Less uniform for balanced chunks	Locality not always guaranteed for chunks

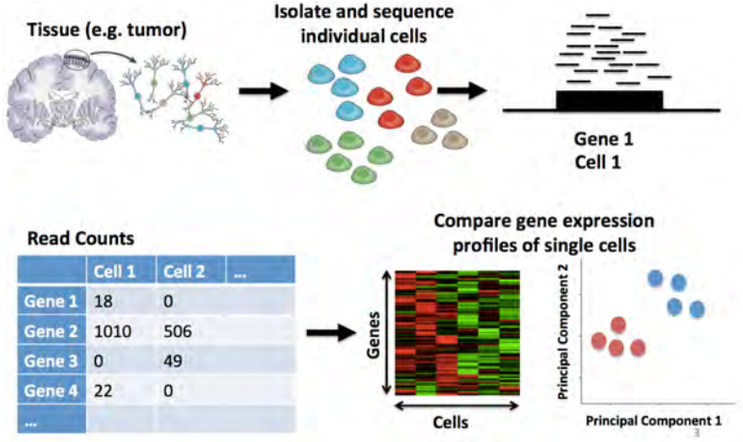
HEALPix is well suited for large batch cross-matching

Single-Cell Genomics: neighbors define cell states

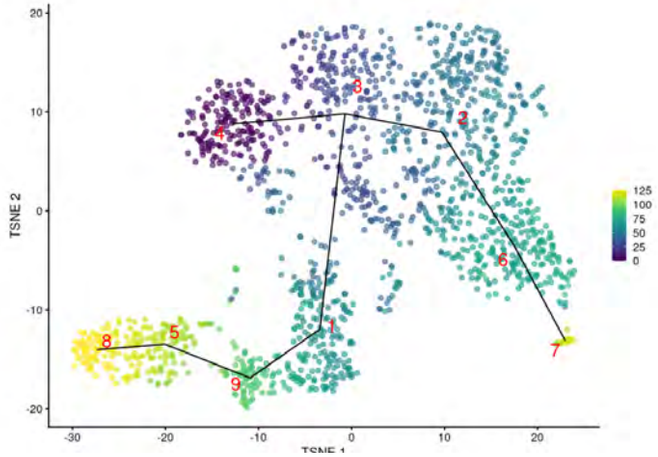
- One data point = one cell.
- Each cell is a high-dimensional gene-count vector.
- For downstream tasks, we only need a good local NN graph, not exact NNs.

Raw data	$X = \texttt{adata.X} \in \mathbb{R}^{68{,}000 \times 32{,}738}$
Entry	$X_{ij} =$ RNA count of gene j in cell i
Raw vector	$x_i \in \mathbb{R}^{32{,}738}$
kNN input	$\texttt{adata.obsm["X_pca"]} \in \mathbb{R}^{68{,}000 \times 50}$

$32{,}738$ genes $\longrightarrow$ $1{,}499$ HVGs $\longrightarrow$ 50 PCs $\longrightarrow$ $\boxed{k\text{NN graph}}$ $\longrightarrow$ clustering / trajectory.



Single-cell RNA sequencing



Clustering / trajectory

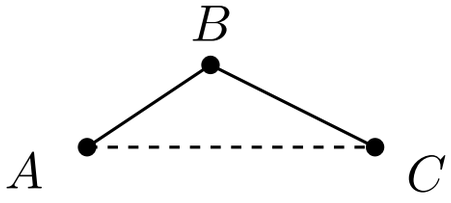
Single-Cell Genomics: neighbors define cell states

Triadic closure: friend of a friend

- A friend of a friend is a compound social relation:
  $A \sim B, \quad B \sim C.$
- Friendship may not be transitive:
  $A \sim B, \quad B \sim C \not\Rightarrow A \sim C.$
- Yet social triangles often tend toward closure.

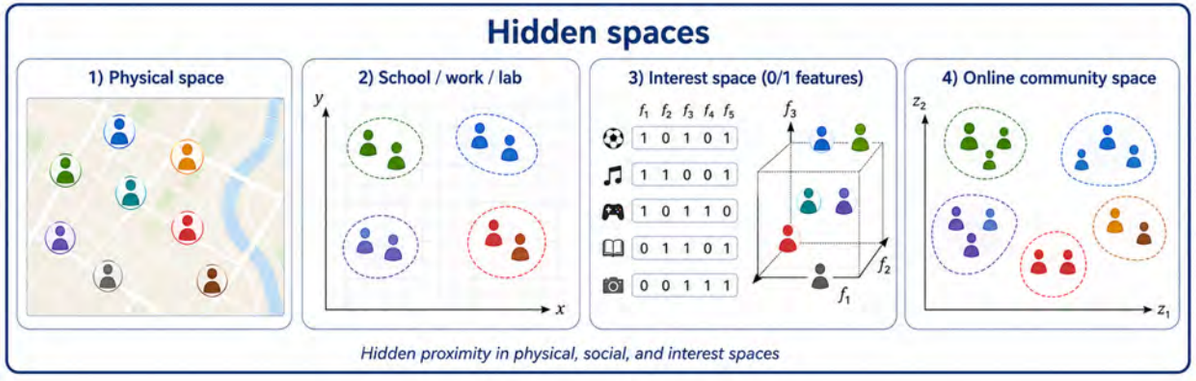
$$\Pr(A \sim C \mid A \sim B,\ B \sim C) > \Pr(A \sim C).$$

- Why, if friendship is not a metric? Because social links are often shaped by hidden spaces.



$$d(A, C) \leq d(A, B) + d(B, C)$$

In metric space



Single-Cell Genomics: neighbors define cell states

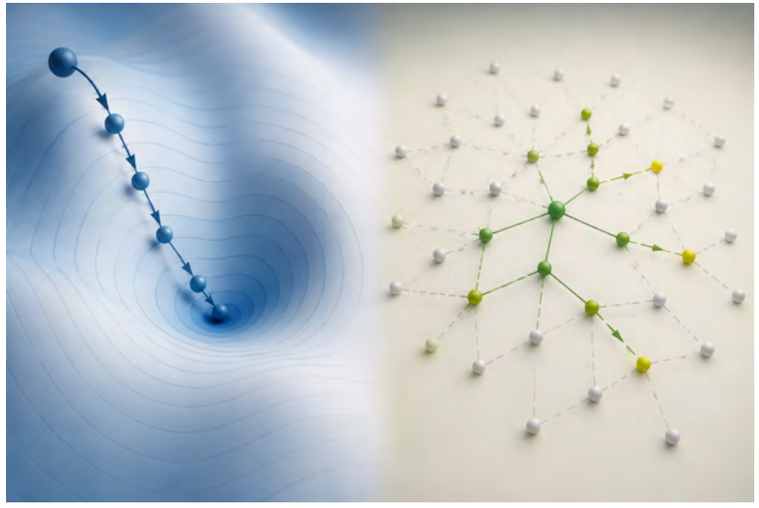
NN-Descent: neighbors of neighbors

- Start with a rough neighbor graph.
- For each point v, look at neighbors of its neighbors.

$$u \in B[v], \quad w \in B[u] \quad \Rightarrow \quad w \text{ is a candidate for } v.$$

- Keep the best k candidates and repeat.

$$G_0 \to G_1 \to \cdots \to \hat{G}^T.$$



Single-Cell Genomics: neighbors define cell states

Appendix: Why does NN-Descent converge fast? (Dong et al. 2011)

- Assume a metric space with bounded expansion constant c:

  balls do not grow too much when the radius doubles.

$$|B_{2r}(v)| \leq c\,|B_r(v)|.$$

- Suppose the current list $B[v]$ is already reasonably good: its K candidates are roughly uniformly distributed in $B_r(v)$.
- Then neighbors of neighbors can reveal closer candidates:

$$B'(v) = \bigcup_{u \in B[v]} B[u].$$

  Under the heuristic assumptions, $B'(v)$ is likely to contain K candidates inside $B_{r/2}(v)$.

- Hence the effective search radius shrinks geometrically:

$$r \;\to\; r/2 \;\to\; r/4 \;\to\; \cdots$$

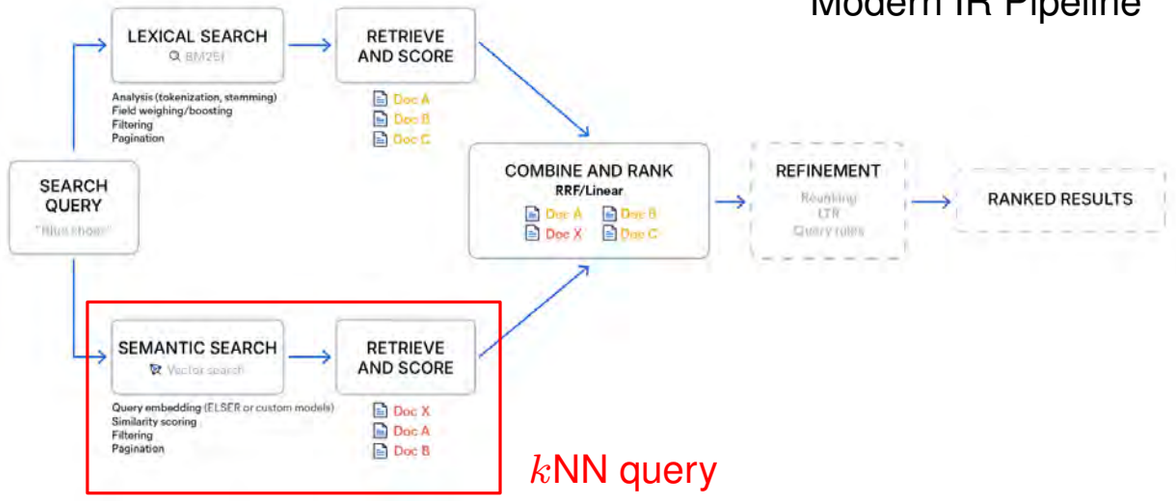
$\to$ under very strong assumptions...

Information Retrieval: meaning becomes geometry

Vector search at scale

- Data are embedded as high-dimensional vectors: documents, images, users, queries, etc.
- Massive scale: millions to billions of vectors.
- ANN setting: retrieval needs good candidates, not perfect distance ranks.
- Goal: push the recall–latency frontier: high recall with low latency, memory, and cost.
- Full-stack trade-off problem: algorithm, hardware, memory, network, and query workload all matter.

Modern IR Pipeline

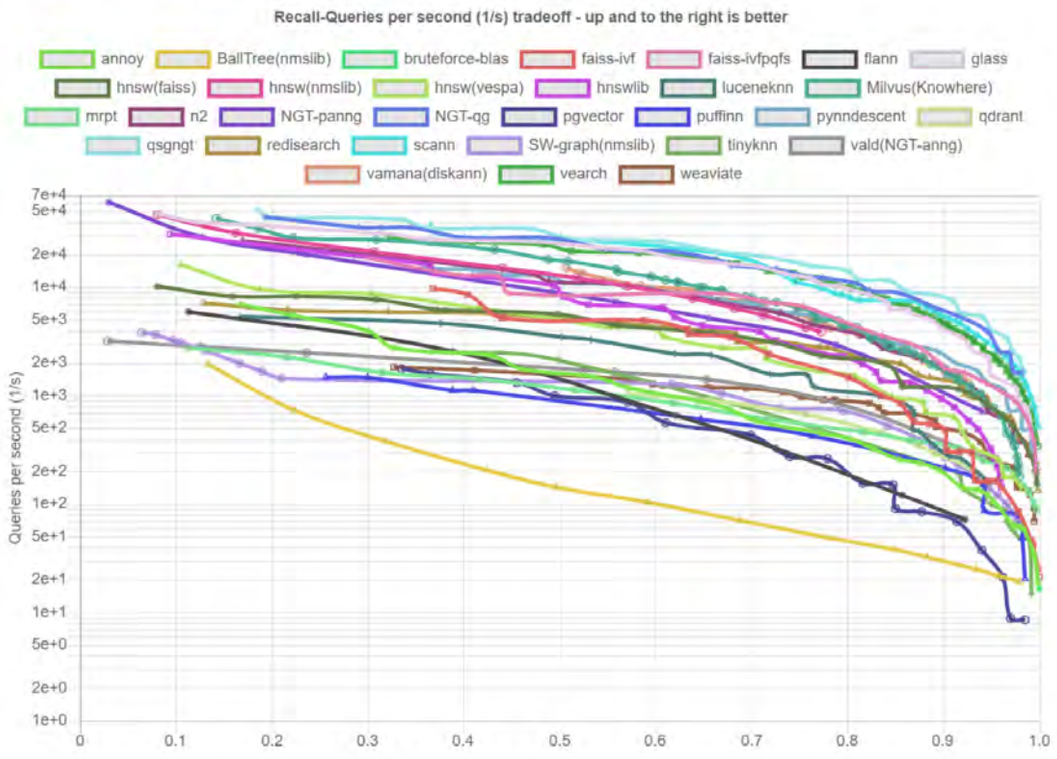


Information Retrieval: meaning becomes geometry

ANN Benchmarks (https://ann-benchmarks.com/index.html)

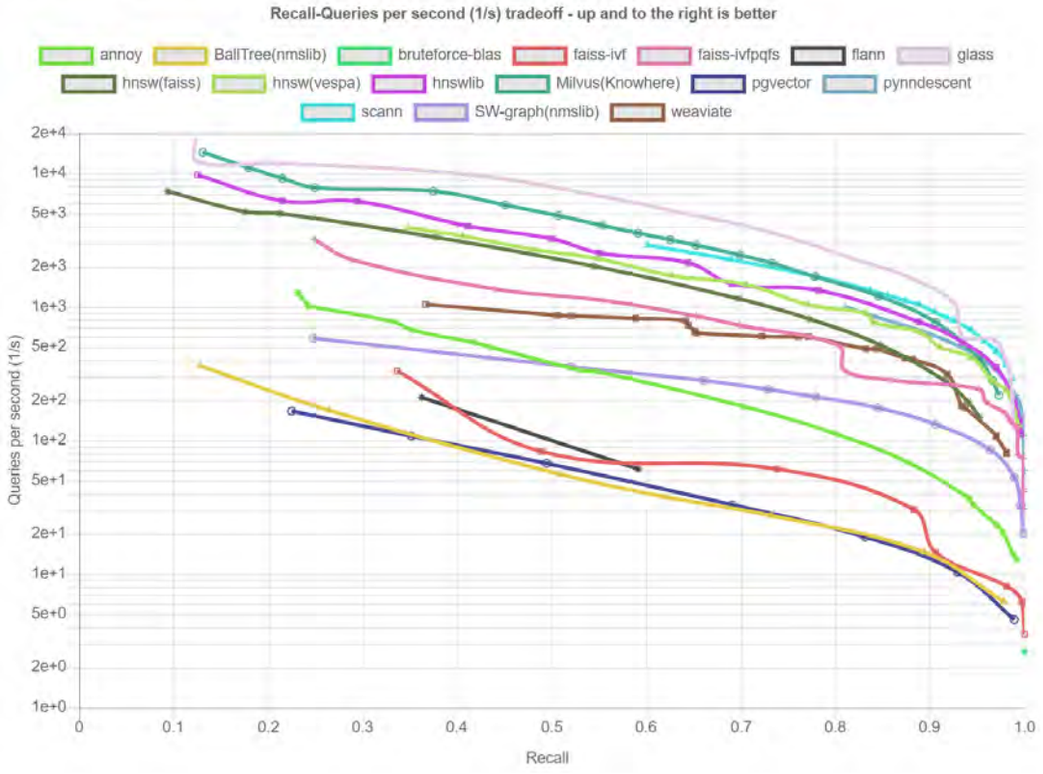
Plots for glove-100-angular (k = 10)

Recall/Queries per second (1/s)



Plots for gist-960-euclidean (k = 10)

Recall/Queries per second (1/s)



- **Recall:** fraction of true k-nearest neighbors recovered.
- **QPS:** number of queries answered per second.
- **Upper-right is better:** high recall and high QPS.

Information Retrieval: meaning becomes geometry

Appendix: ANN-Benchmarks Algorithms by Indexing Style

Family	Examples in ANN-Benchmarks
Brute force	`bf`, `bruteforce-blas`
Tree/forest-based	`kd`, `ckdtree`, `BallTree(nmslib)`, `annoy`, `mrpt`, `flann`*
Grid-based	–
Hash/LSH-based	`puffinn`, `elastiknn-l2lsh`, `datasketch`
Graph-based	`hnswlib`, `hnsw(faiss)`, `hnsw(nmslib)`, `hnsw(vespa)`, `vald(NGT-anng)`, `NGT-panng`, `NGT-onng`, `SW-graph(nmslib)`, `n2`, `pynndescent`, `vamana(diskann)`, `NGT-qg`, `qsgngt`, `glass`, `pgvector(HNSW)`, `Qdrant(HNSW)`, `Weaviate(HNSW)`, `OpenSearchKNN(HNSW)`, `Redisearch(HNSW)`, `luceneknn(HNSW)`
Quantization/compression-based	`NGT-qg`, `qsgngt`, `faiss-ivfpqfs`, `scann`, `glass`†, `Qdrant`†
Partition/cluster-based	`faiss-ivf`, `tinyknn`, `faiss-ivfpqfs`, `scann`, `flann`*

Classified with GPT; categories may overlap. *`FLANN` is run in autotuned mode; the selected index may be tree- or cluster-based. †Only configurations using vector quantization.

Empirical lesson from ANN-Benchmarks

- Graph-based methods are often strong on the high-recall frontier.
- But practical winners are not just abstract data structures: performance also depends on compression, partitioning, reranking, parameter tuning, and hardware-aware implementation.

Summary

- **Proximity is a model:** nearest does not automatically mean scientifically meaningful.
- **Choose the right search task:** k-NN query, radius join, or k-NN graph.
- **Choose the right toolbox:** geometry, dimension, metric, scale, exactness, and hardware matter.
- **Neighbors are usually candidates:** MD interactions, catalog matches, cell states, and retrieved documents still need domain validation.
- **There is no universal best algorithm:** measure accuracy, latency, memory, and scientific usefulness together.

"The purpose of computing is insight, not numbers."

— Richard W. Hamming, *Numerical Methods for Scientists and Engineers* (1962)