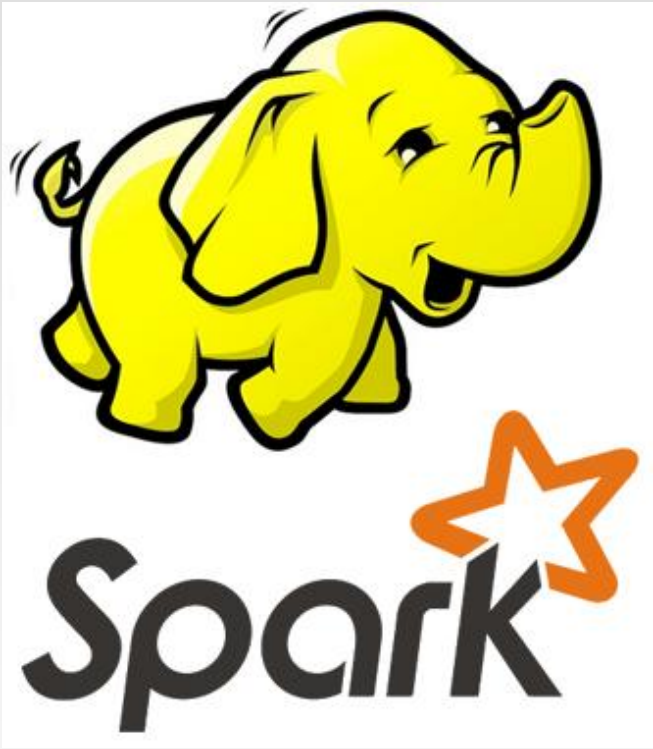
COMP9313: Big Data Management



Lecturer: Xin Cao

Course web site: http://www.cse.unsw.edu.au/~cs9313/

Chapter 7.1: Finding Similar Items

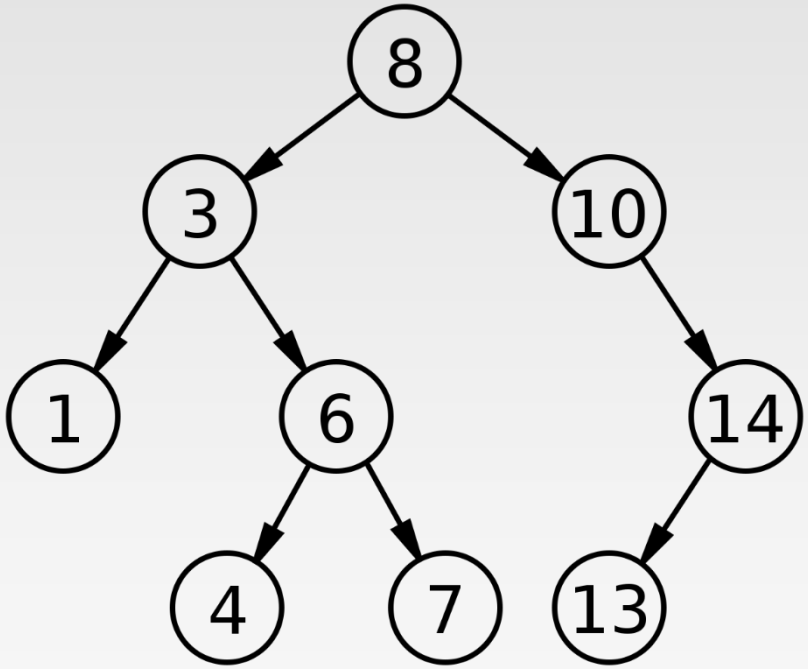
A Common Metaphor

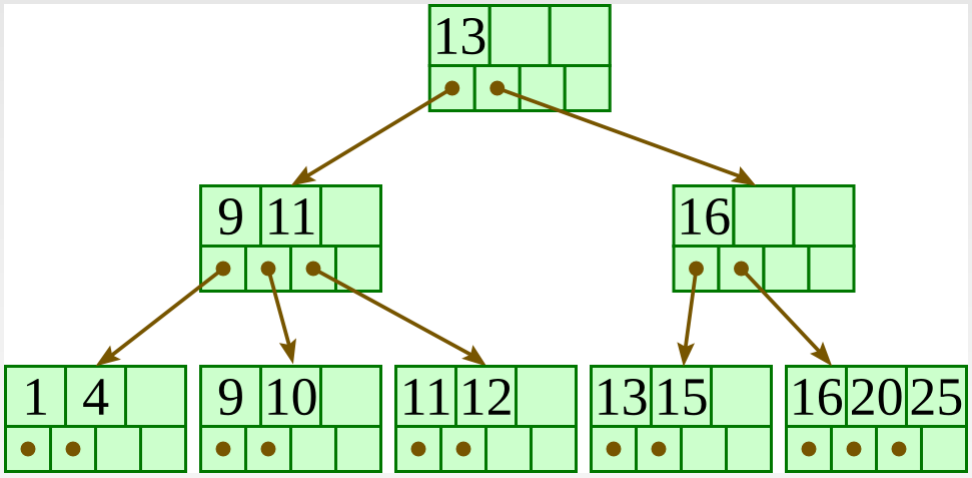
- **Many problems can be expressed as finding "similar" sets:**
  - **Find near-neighbors in high-dimensional space**

- **Examples:**
  - **Pages with similar words**
    - For duplicate detection, classification by topic
  - **Customers who purchased similar products**
    - Products with similar customer sets
  - **Images with similar features**
    - Google image search

Similarity Search in One Dimensional Space

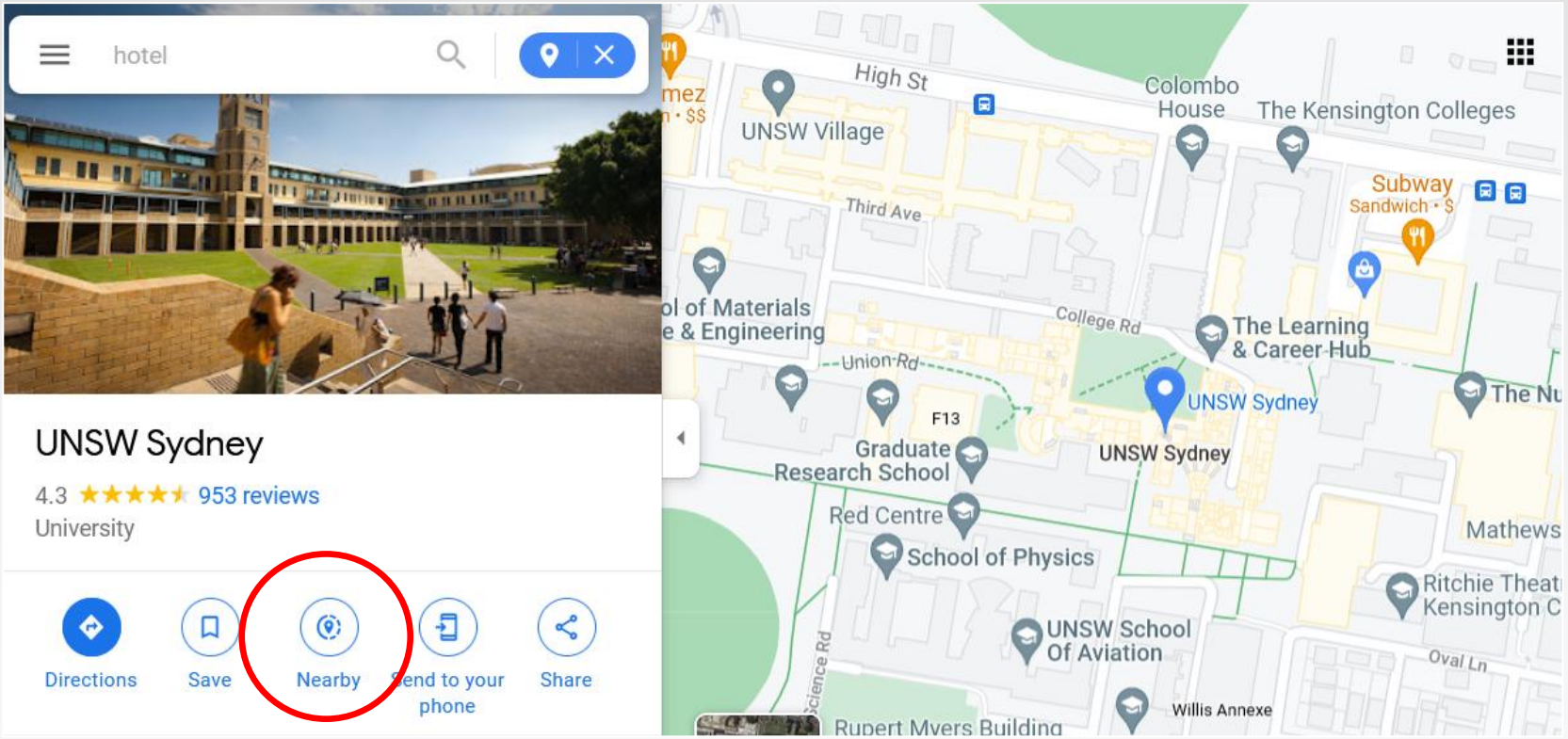
- Just numbers, use binary search, binary search tree, B+-Tree…
- The essential idea behind: objects can be sorted





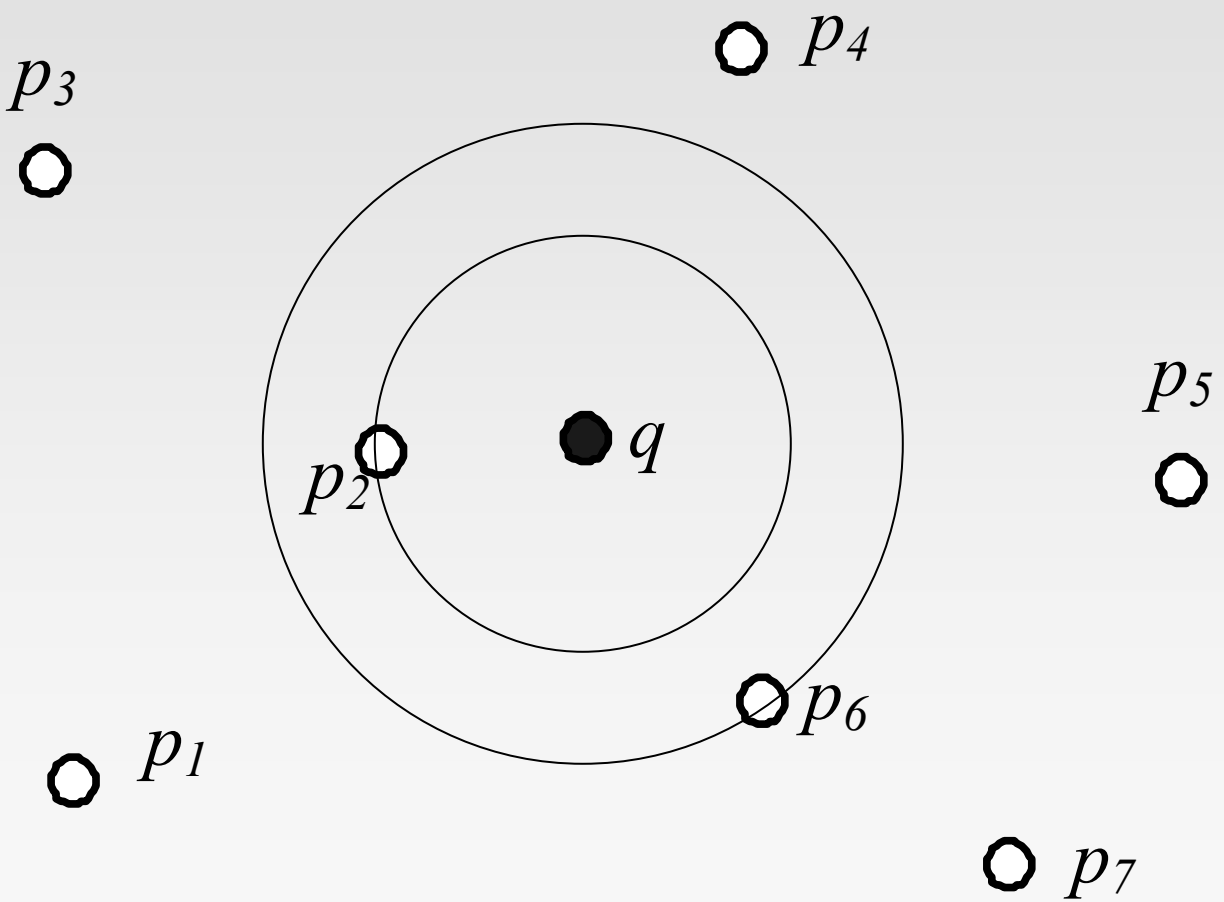
Similarity Search in 2D Space

- *k* nearest neighbour (*k*NN) query: find the top-k nearest spatial object to the query location
- E.g., find the top-5 closest restaurants to UNSW



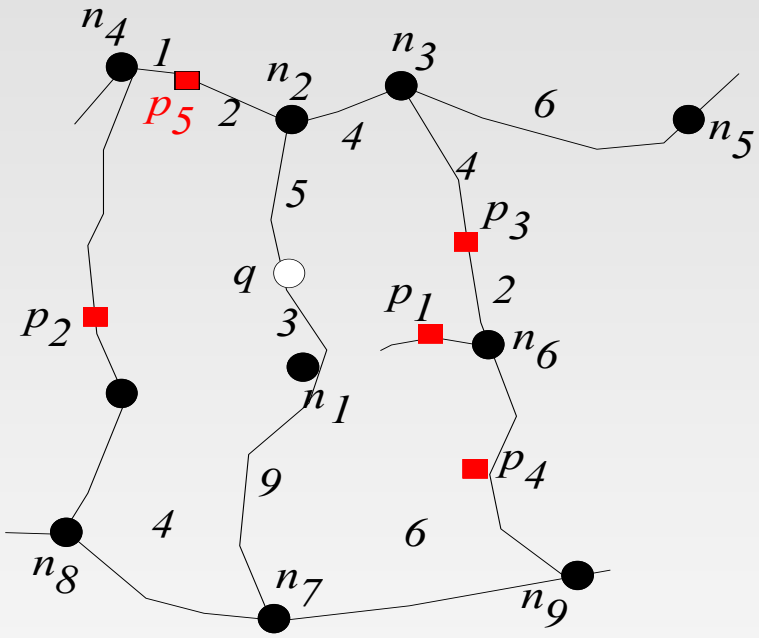
Similarity Search in 2D Space

- In Euclidean Space



Similarity Search in 2D Space

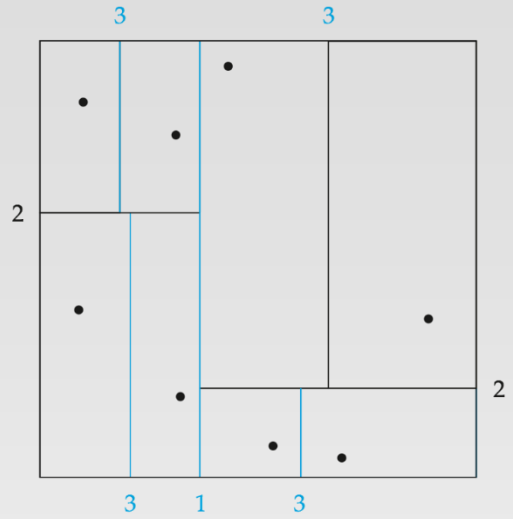
- In road networks: Distance is computed based on the network distance (such as the length of the shortest path)

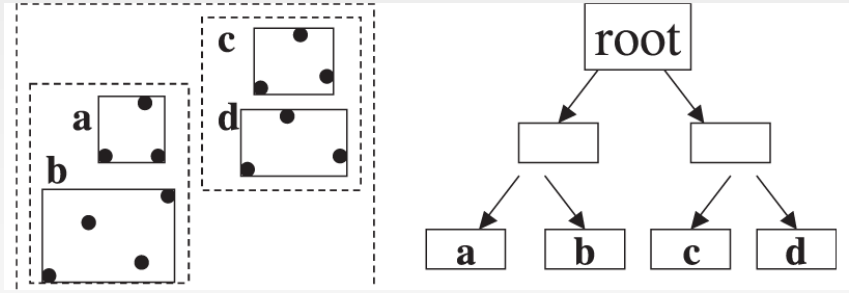


p_5 is the closest in the spatial network setting
p_1 is the closest in the Euclidean space

The Problem in 2D Space

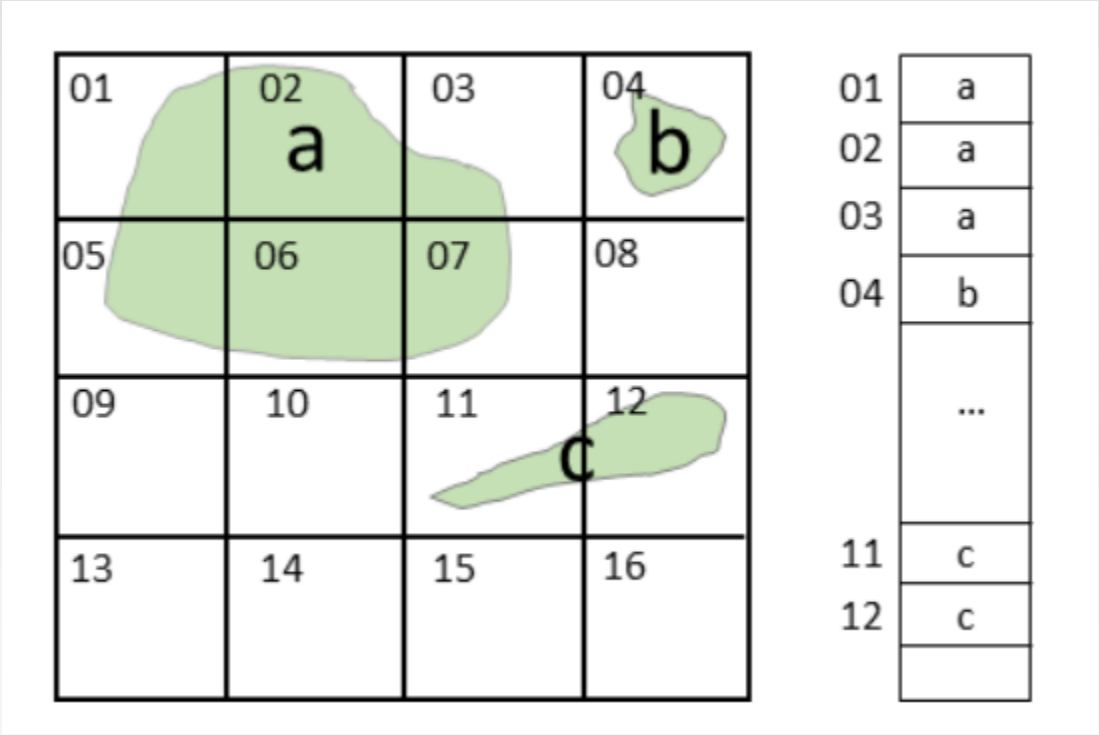
- Euclidean space
  - Grid index
  - Quad-tree
  - *k-d* tree
  - R-tree (R+-tree, R*-tree, etc.)
  - m-tree, x-tree, … …
  - Space filing curves: Z-order, Hilbert order, … …
- Road Networks
  - G-tree
  - Contraction Hierarchy
  - 2-hop labeling
  - … …





Grid Index

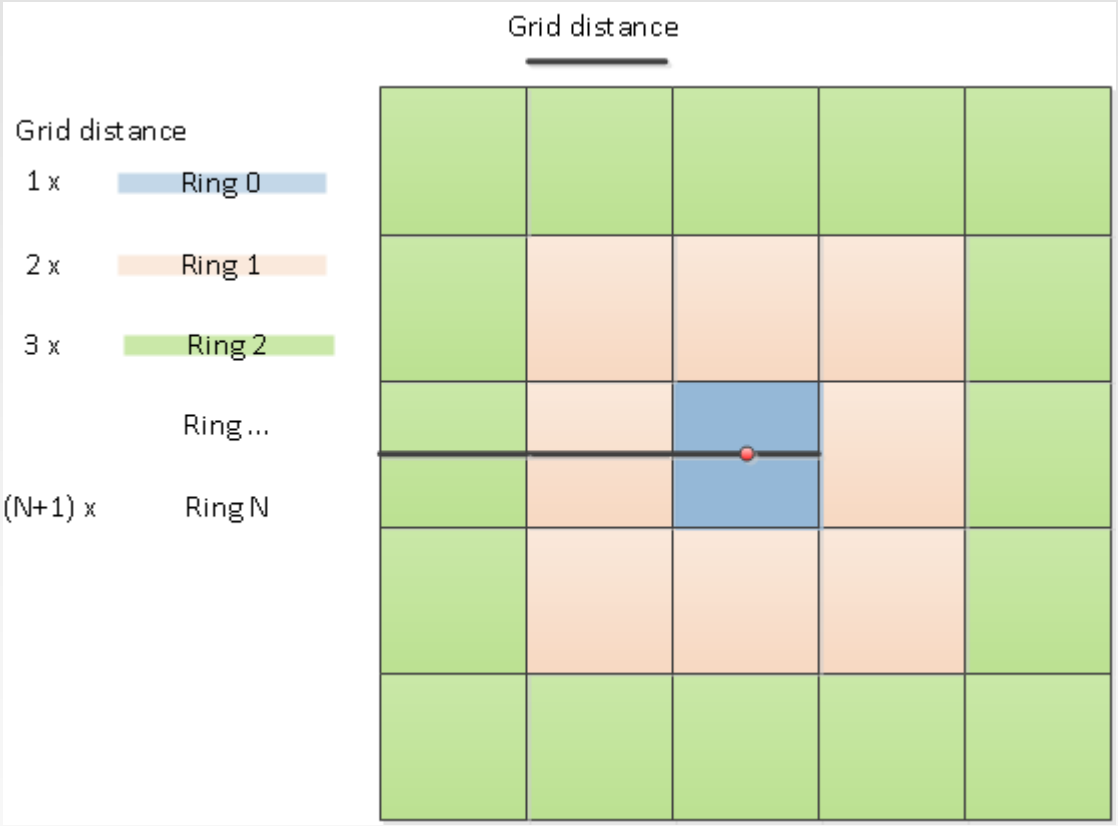
- Fixed grid index is an n × n array of equal-size cells. Each one is associated with a list of spatial objects which intersect or overlap with the cell.



- Multi-level grid index to decrease the redundancy

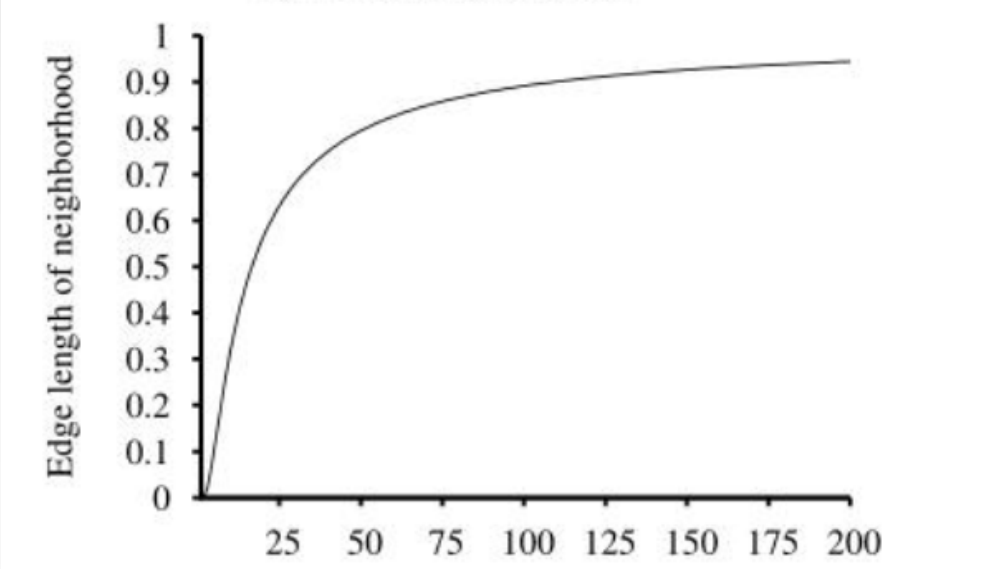
kNN Search using a Grid Index

- Locate the query point in a cell, expand the search area ring by ring.
- For cell C[i, j], the first ring includes 8 cells: C[i-1, j-1] ~ C[i+1,j+1]



Curse of Dimensionality

- Refers to various phenomena that arise in high dimensional spaces that do not occur in low dimensional settings.
- Specifically, refers to the decrease in performance of similarity search query processing when the dimensionality increases.
- In high dimensional space, almost all points are far away from each other.
  - To find the top-10 nearest neighbors, what is the length of the average neighborhood cube?



Problem

- **Given: High dimensional data points $x_1, x_2, \ldots$**
  - **For example:** Image is a long vector of pixel colors

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix} \rightarrow [1\ 2\ 1\ 0\ 2\ 1\ 0\ 1\ 0]$$

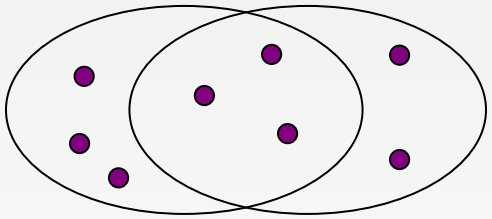
- **And some distance function $d(x_1, x_2)$**
  - Which quantifies the "distance" between x_1 and x_2
- **Goal:** Find **all pairs of data points** (x_i, x_j) that are within some distance threshold $d(x_i, x_j) \leq s$
- **Note:** Naïve solution would take $O(N^2)$ ☹

  where N is the number of data points

- **MAGIC: This can be done in $O(N)$!! How?**

Distance Measures

- **Goal**: Find near-neighbors in high-dim. space
  - We formally define “near neighbors” as points that are a “small distance” apart
- For each application, we first need to define what “**distance**” means
- **Today: Jaccard distance/similarity**
  - The **Jaccard similarity** of two **sets** is the size of their intersection divided by the size of their union:
    $\boldsymbol{sim(C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|}$
  - **Jaccard distance:** $\boldsymbol{d(C_1, C_2) = 1 - |C_1 \cap C_2| / |C_1 \cup C_2|}$



3 in intersection
8 in union
Jaccard similarity= 3/8
Jaccard distance = 5/8

Task: Finding Similar Documents

- **Goal: Given a large number (N in the millions or billions) of documents, find "near duplicate" pairs·**
- **Applications:**
  - Mirror websites, or approximate mirrors
    - Don't want to show both in search results
  - Similar news articles at many news sites
    - Cluster articles by "same story"
- **Problems:**
  - Many small pieces of one document can appear out of order in another
  - Too many documents to compare all pairs
  - Documents are so large or so many that they cannot fit in main memory

Project 3: Spatial and Textual Similarity Join

Problem

Task: Given two sets of records, **Dataset A** and **Dataset B**, where each record R = (L, S) has two attributes: a spatial location in 2D space (i.e., R.L is a coordinate (x, y)) and a textual description (i.e., R.S is a set of terms/words).

The task is to find all pairs of records, (R_A, R_B), where R_A in Dataset A and R_B in Dataset B, which satisfy **both** of the following conditions simultaneously:

- Their **Euclidean distance** is less than or equal to a given threshold ***d***.
- Their **Jaccard Similarity** on the set of terms is greater than or equal to a given threshold ***s***.

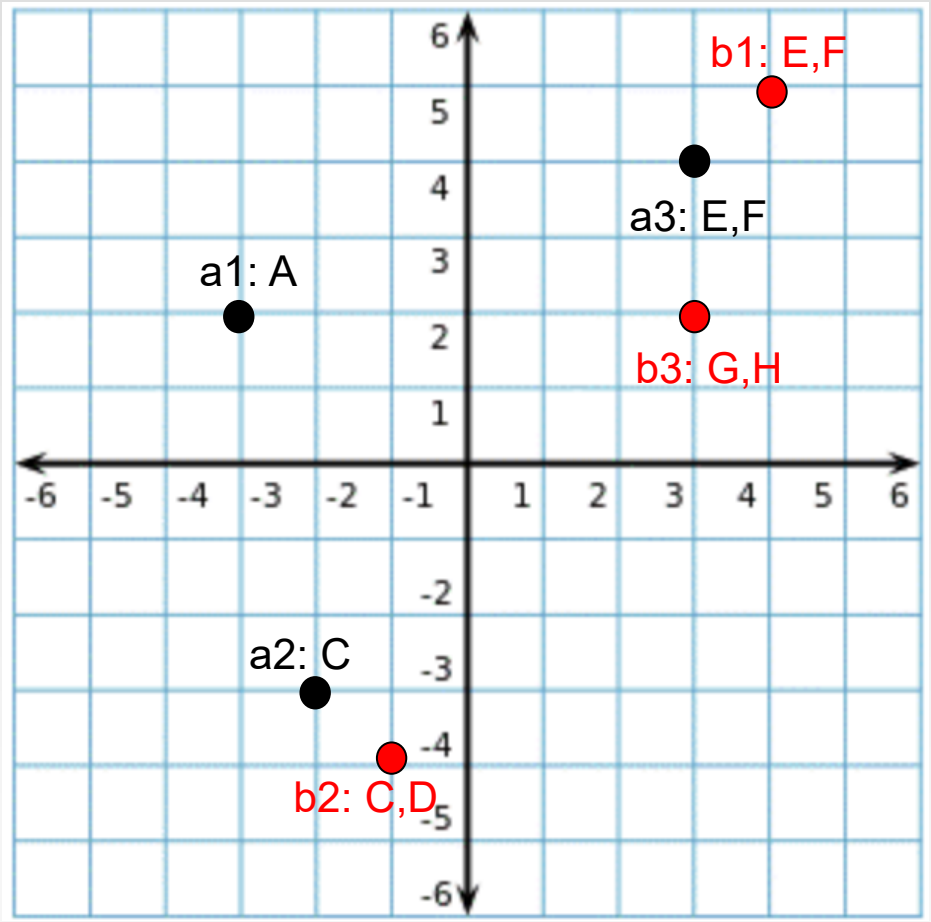
Their **Euclidean distance** is less than or equal to a given threshold ***d***:

$$\sqrt{(R_A.L.x - R_B.L.x)^2 + (R_A.L.y - R_B.L.y)^2} \leq d$$

Their **Jaccard Similarity** is larger than or equal to a given threshold ***s***, that is:

$$\frac{|R_A.S \cap R_B.S|}{|R_A.S \cup R_B.S|} \geq s$$

Example



Given d=2 and s=0.5

The results include:
(a2, b2): sqrt(2), 0.5
(a3, b1): sqrt(2), 1

Set-Similarity Join

- Given two collections of records R and S, a similarity function sim(., .), and a threshold τ, the set similarity join between R and S, is to find all record pairs r (from R) and s (from S), such that sim(r, s) >= τ.

id	set
r_1	$\{e_1, e_4, e_5, e_6\}$
r_2	$\{e_2, e_3, e_6\}$
r_3	$\{e_4, e_5, e_6\}$

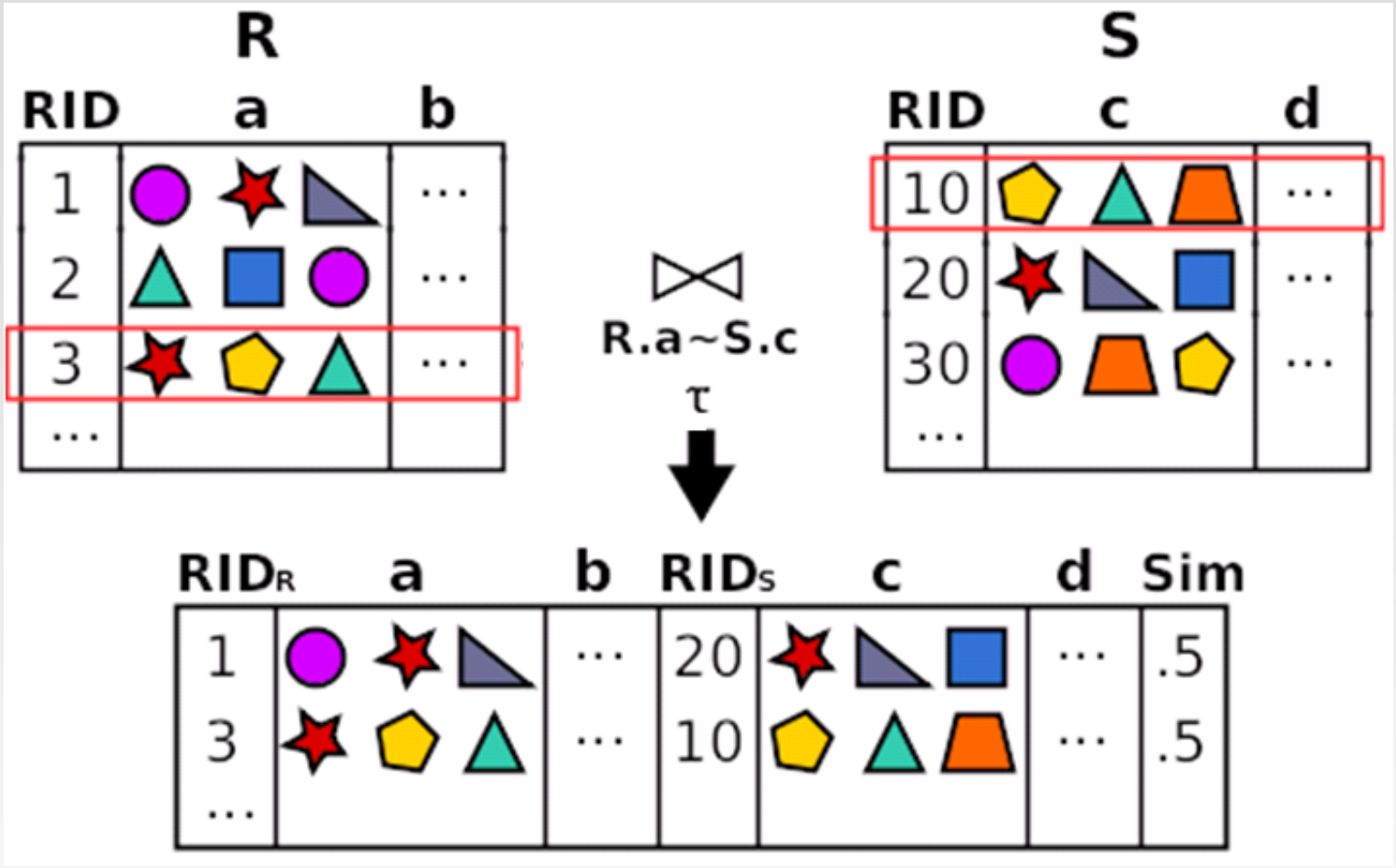
(a) $\mathcal{R}$ sets

id	set
s_1	$\{e_1, e_4, e_6\}$
s_2	$\{e_2, e_5, e_6\}$
s_3	$\{e_3, e_5\}$

(b) $\mathcal{S}$ sets

- Given the above example, and set τ=0.5, the results are: (r1, s1) (similarity 0.75), (r2, s2) (similarity 0.5), (r3, s1) (similarity 0.5), (r3, s2) (similarity 0.5).

Set-Similarity Join



Finding pairs of records with a **similarity** on their join attributes > t

Application: Record linkage

Table R

Star
Keanu Reeves
Samuel Jackson
Schwarzenegger
…

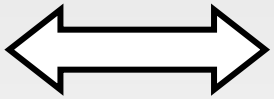
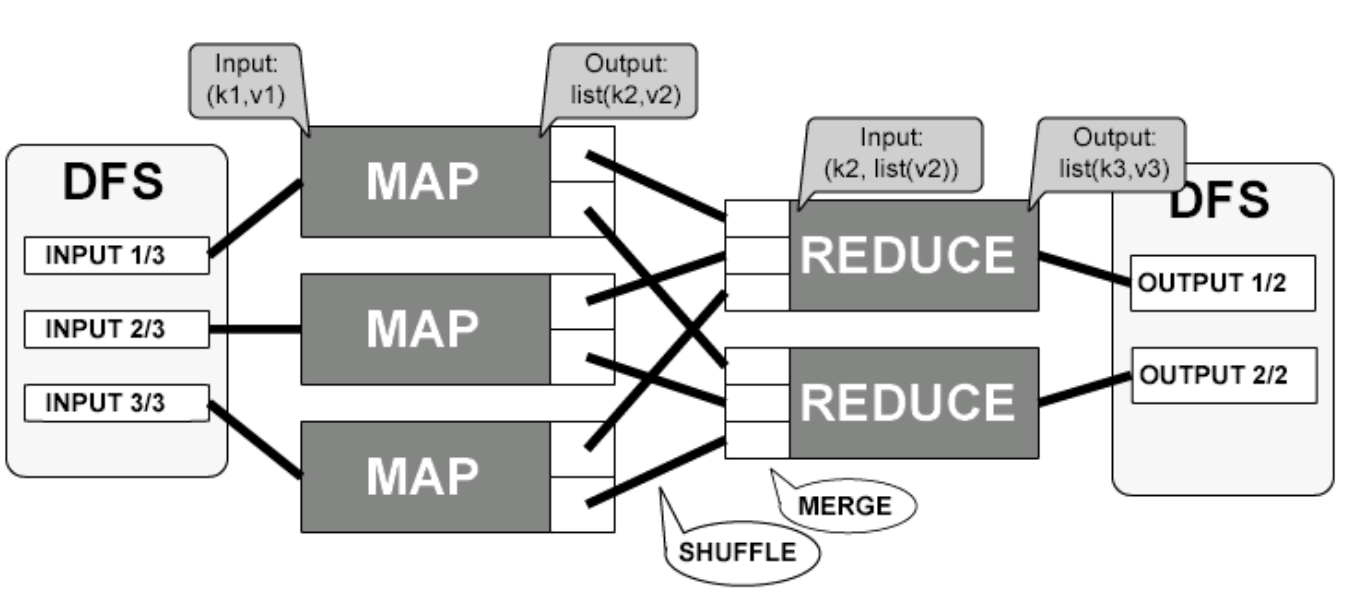


Table S

Star
Keanu Reeves
Samuel L. Jackson
Schwarzenegger
…

A Naïve Solution

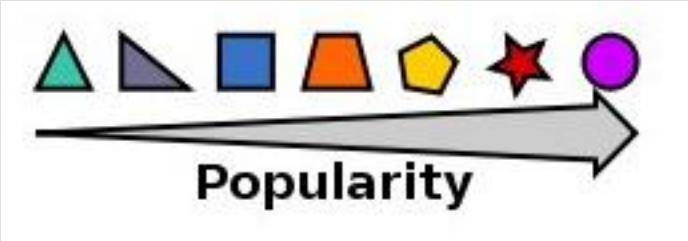
- Map: <23, (a,b,c)> → (a, 23), (b, 23), (c, 23)
- Reduce: (a,23),(a,29),(a,50), … → Verify each pair (23, 29), (23, 50), (29, 50) … …



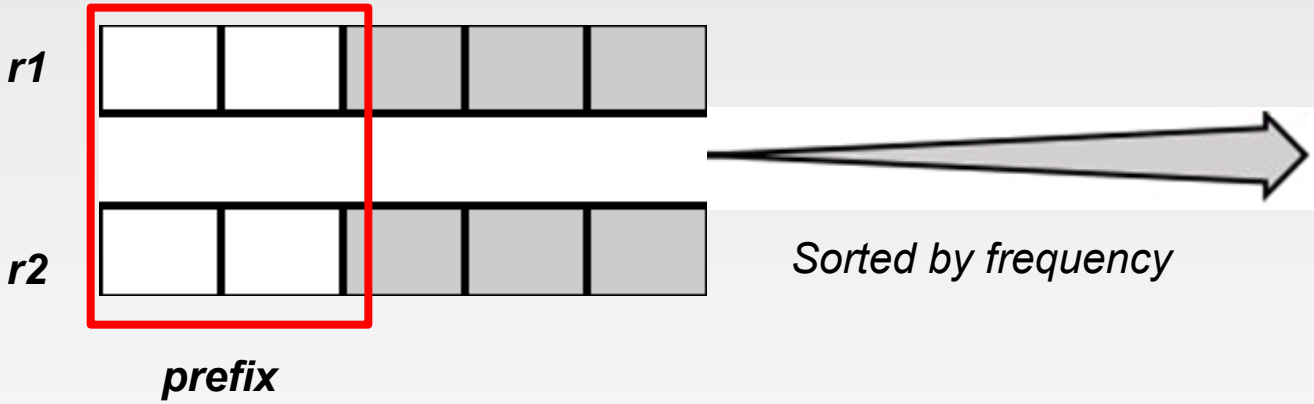
- Too much data to transfer ☹
- Too many pairs to verify ☹

Solving frequency skew: prefix filtering

- Sort tokens by frequency (ascending)

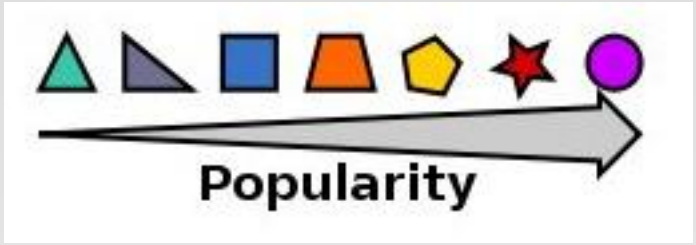


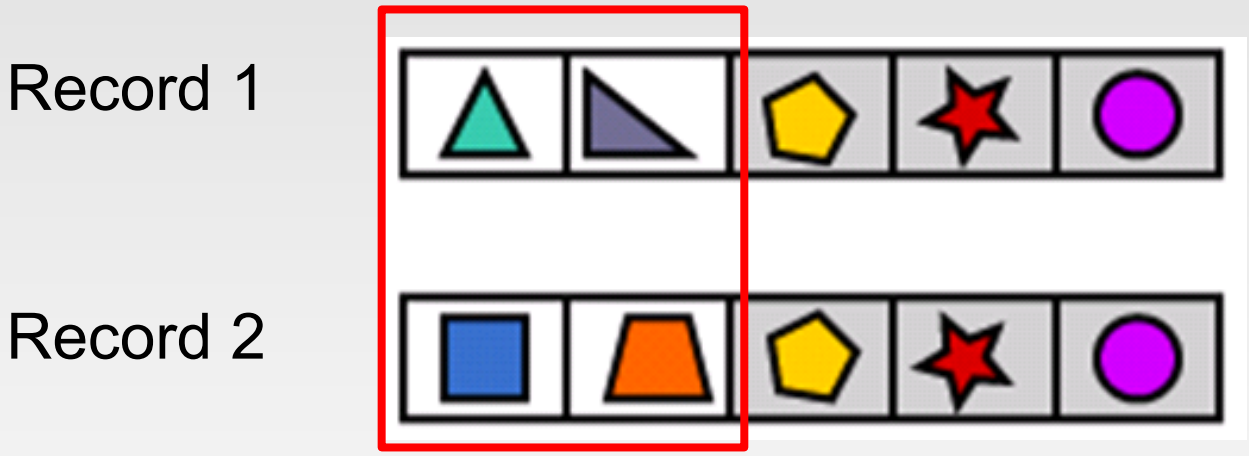
- Prefix of a set: least frequent tokens



- Prefixes of similar sets should share tokens

Prefix filtering: example

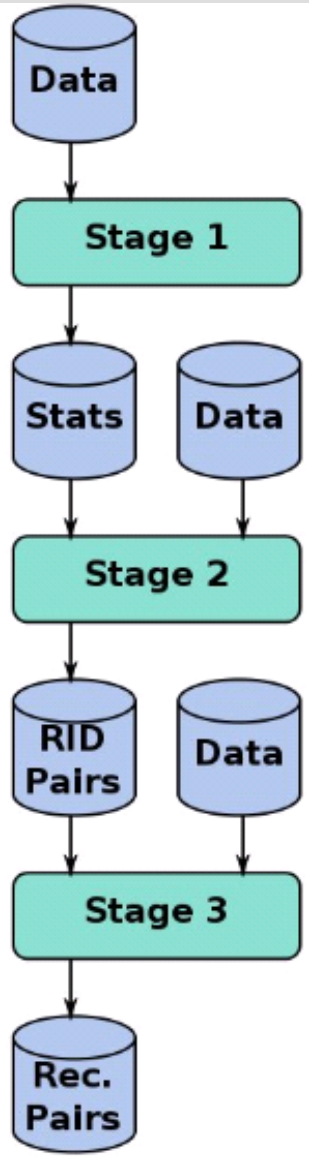




- Each set has 5 tokens
- “Similar”: they share at least 4 tokens
- Prefix length: 2

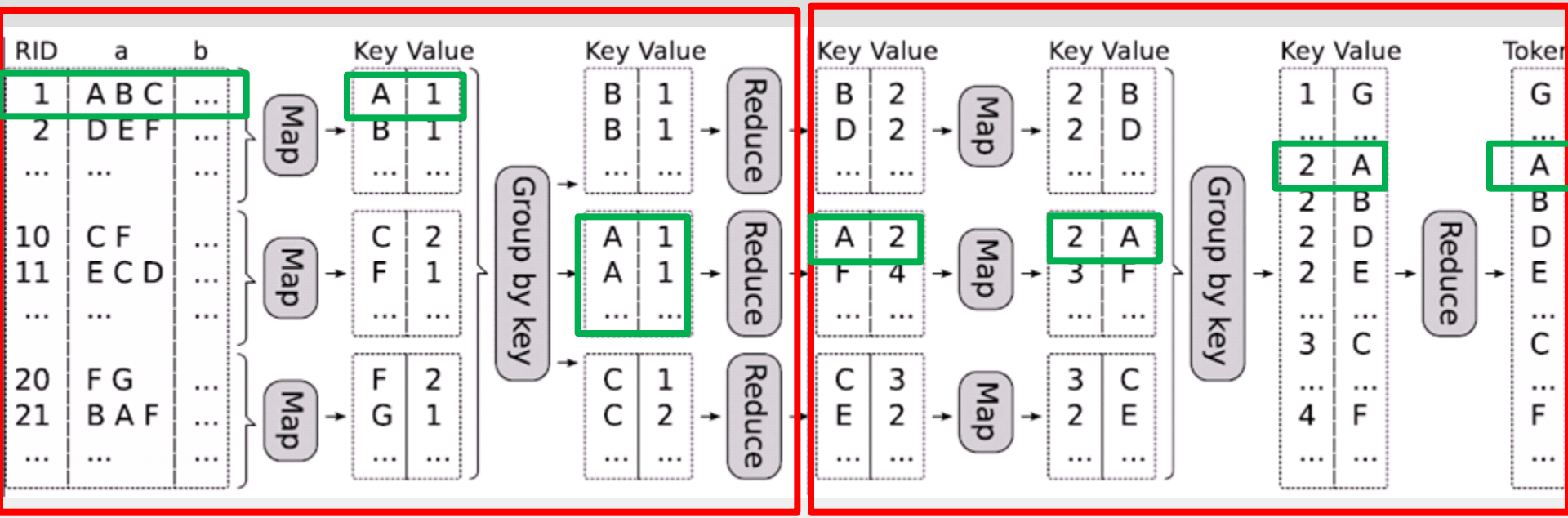
Hadoop Solution: Overview

- Stage 1: Order tokens by frequency
- Stage 2: Finding “similar” id pairs (verification)
- Stage 3: remove duplicates



Efficient Parallel Set-Similarity Joins Using MapReduce. SIGMOD’10

Stage 1: Sort tokens by frequency



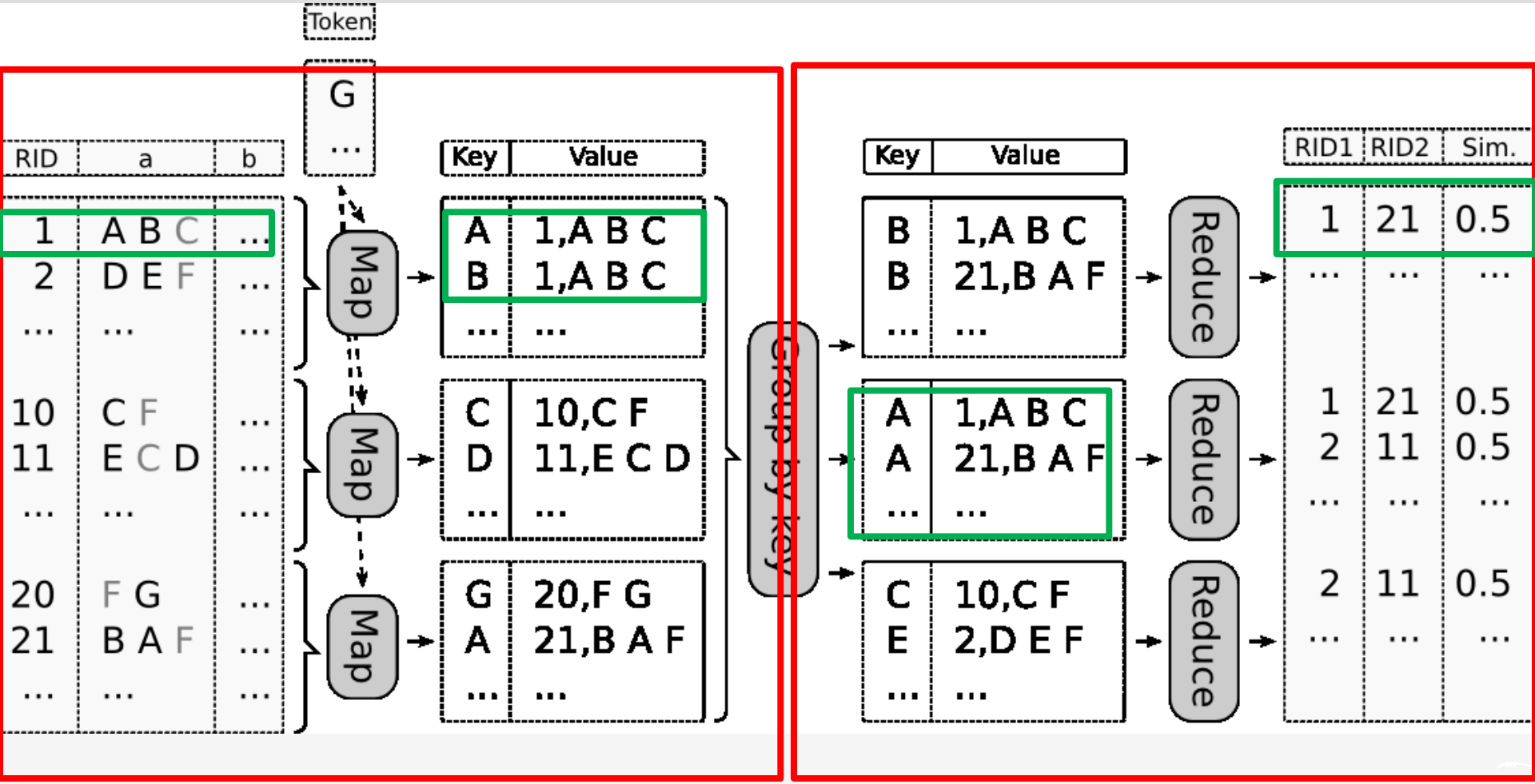
Compute token frequencies

Sort them

MapReduce phase 1

MapReduce phase 2

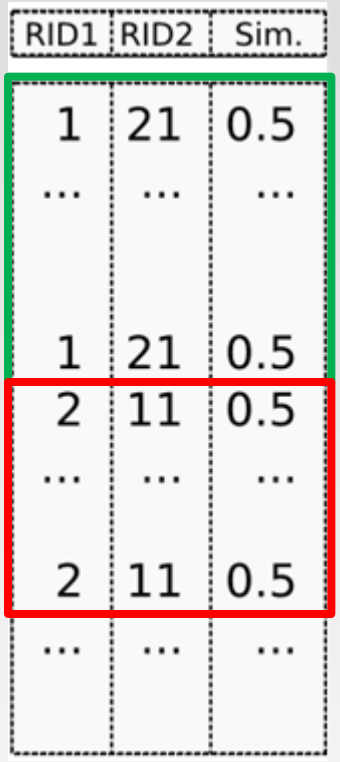
Stage 2: Find "similar" id pairs



Partition using prefixes

Verify similarity

Stage 3: Remove Duplicates

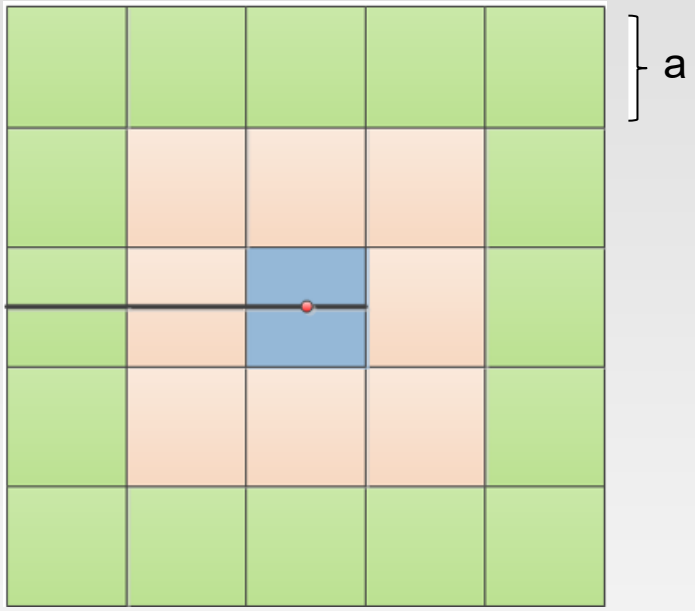


Compute the Length of Shared Tokens

- Jaccard Similarity: **sim(r, s) = |r∩s|/|r∪s|**
- If **sim(r, s) >= τ, |r∩s| >= |r∪s| * τ >= |r| * τ = l**
- Given a record r, you can compute the prefix length as p = |r| - l + 1
- r and s is a candidate pair, they must share at least one token in the first (|r| - l + 1) tokens
- Given a record r = (A, B, C, D) and p = 2, the mapper emits (A, r) and (B, r)

Spatial Distance Join using Grid

- Partition the data space using a grid
- For each grid cell, calculate the neighbor cells
  - For example, if d<a, we need to check 8 cells instead of the whole space
- After the filtering steps, we can proceed to the verification step, i.e, compute the distances for all candidate pairs



More Tips on Project 3

- 1. Using either RDD or DataFrame, no third-party libraries
- 2. This project requires exact set similarity join results. Please follow the slides as introduced.
- 3. More test cases will be released to you soon.
- 4. It is your job to design more optimization strategies. The faster the better! Some hints to accelerate your program:
  - How to represent the terms in your solution?
  - Broadcast the frequency lookup table.
  - Try to avoid computing similarities for duplicated pairs.
- 5. You do not need to merge the results.
- 6. To guarantee the fairness of comparing the efficiency, we will run your code in the VM using two local threads
  - In your submission, remove all partition numbers

More Tips on Project 3 (Dataproc)

- 1. Check your bill!!! Be careful of how much you have already spent. Remember to terminate the cluster and delete the data in your bucket after you finish your jobs!!! If you have used the $300 credits, you can register with Dataproc using a new email.
- 2. If the CPU limit of your account is only 8, you just need to create two clusters: one with 2 worker nodes and the other one with 3 worker nodes. I do not know why the CPU limit is different for us, and I haven't found a solution for this.
- 3. Someone may see some error messages like: "Broadcasting large task binary with size XXXMB" or "java.lang.InterruptedException". If you job can complete successfully, you can ignore these messages. I also saw such messages when running my job.
- 4. Do not use SetMaster("local") when running on Dataproc.
- 5. More partitions of your RDD/DataFrame could be helpful on clusters with more worker nodes.

Finding Similar Docs Approximately

3 Essential Steps for Similar Docs

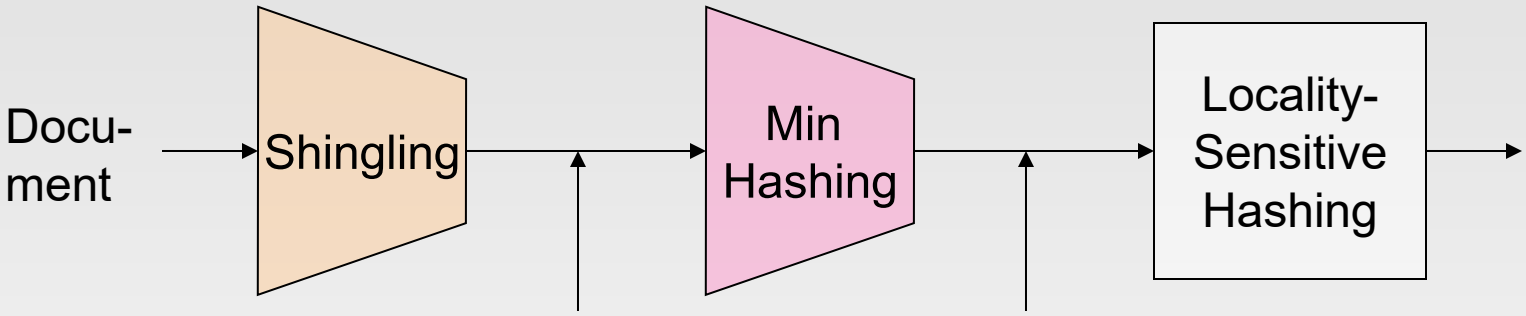
1. ***Shingling:*** Convert documents to sets

2. ***Min-Hashing:*** Convert large sets to short signatures, while preserving similarity

3. ***Locality-Sensitive Hashing:*** Focus on pairs of signatures likely to be from similar documents

   - **Candidate pairs!**

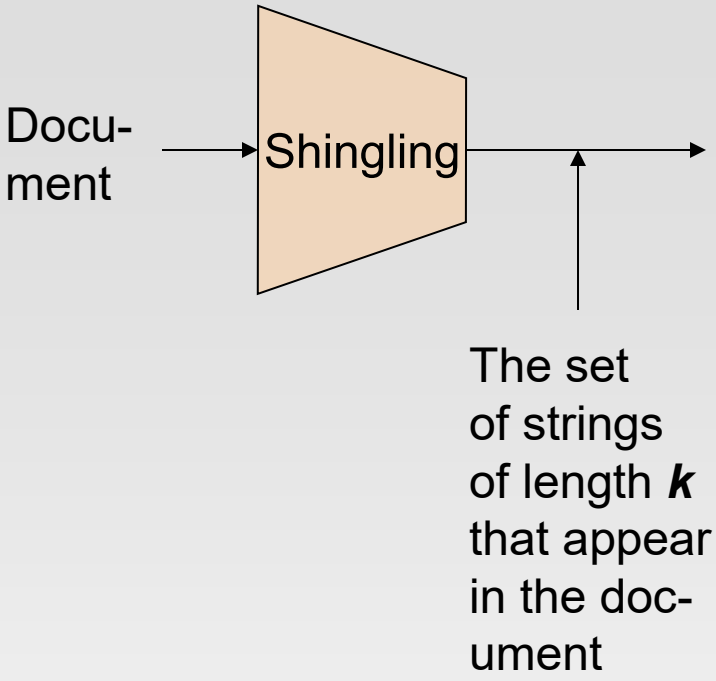
The Big Picture



The set of strings of length ***k*** that appear in the document

Signatures: short integer vectors that represent the sets, and reflect their similarity

Candidate pairs: those pairs of signatures that we need to test for similarity



Step 1: ***Shingling:*** Convert documents to sets

Documents as High-Dim Data

- **Step 1: *Shingling:* Convert documents to sets**

- **Simple approaches:**
  - Document = set of words appearing in document
  - Document = set of "important" words
  - Don't work well for this application. Why?

- **Need to account for ordering of words!**
- A different way: **Shingles!**

Define: Shingles

- A *k*-shingle (or *k*-gram) for a document is a sequence of *k* tokens that appears in the doc
  - Tokens can be characters, words or something else, depending on the application
  - Assume tokens = characters for examples

- **Example: k=2**; document $\mathbf{D_1}$ = abcab
  Set of 2-shingles: $\mathbf{S(D_1)}$ = {ab, bc, ca}
  - **Option:** Shingles as a bag (multiset), count ab twice: $\mathbf{S'(D_1)}$ = {ab, bc, ca, ab}

Shingles and Similarity

- Documents that are intuitively similar will have many shingles in common.
- Changing a word only affects k-shingles within distance k-1 from the word.
- Reordering paragraphs only affects the 2k shingles that cross paragraph boundaries.
- Example: k=3, “The dog which chased the cat” versus “The dog that chased the cat”.
  - Only 3-shingles replaced are g_w, _wh, whi, hic, ich, ch_, and h_c.

Compressing Shingles

- To **compress long shingles**, we can **hash** them to (say) 4 bytes

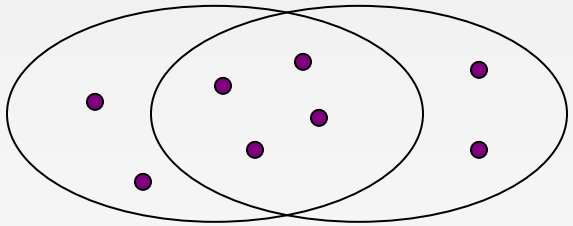
- **Represent a document by the set of hash values of its *k*-shingles**
  - **Idea:** Two documents could (rarely) appear to have shingles in common, when in fact only the hash-values were shared

- **Example: k=2**; document $\mathbf{D_1}$= abcab
  Set of 2-shingles: $\mathbf{S(D_1)}$ = {ab, bc, ca}
  Hash the singles: $\mathbf{h(D_1)}$ = {1, 5, 7}

Similarity Metric for Shingles

- **Document D_1 is a set of its k-shingles $C_1=S(D_1)$**
- Equivalently, each document is a 0/1 vector in the space of *k*-shingles
  - Each unique shingle is a dimension
  - Vectors are very sparse
- **A natural similarity measure is the Jaccard similarity:**

$$sim(D_1, D_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$$



Working Assumption

- **Documents that have lots of shingles in common have similar text, even if the text appears in different order**

- If we pick ***k*** too small, then we would expect most sequences of ***k*** characters to appear in most documents
  - We could have documents whose shingle-sets had high Jaccard similarity, yet the documents had none of the same sentences or even phrases
  - Extreme case: when we use ***k*** = 1, almost all Web pages will have high similarity.

- **Caveat:** You must pick ***k*** large enough, or most documents will have most shingles
  - ***k*** = 5 is OK for short documents
  - ***k*** = 10 is better for long documents

References

- Chapter 3 of Mining of Massive Datasets.

End of Chapter 7.1