LOCALITY SENSITIVE HASHING

DSA TEAM PROJECT 4

BY
ANANYA AND YADUNANDANA

WHY LSH?

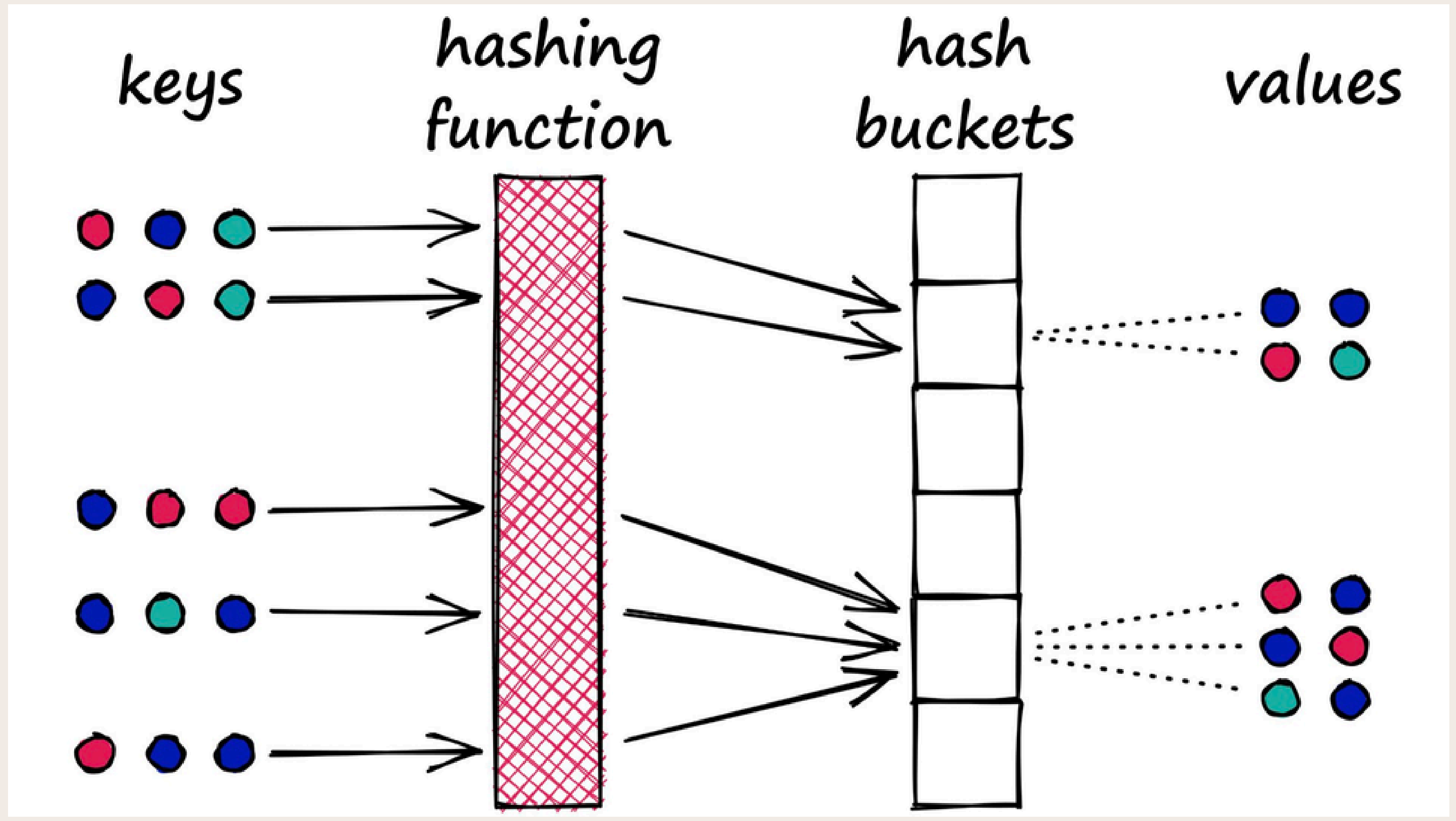
A dataset containing millions samples can’t be efficiently compared.
This produces an at best complexity of $O(n^2)$.
Even if comparing a single sample against the millions of samples, we still return an at best complexity of $O(n)$.

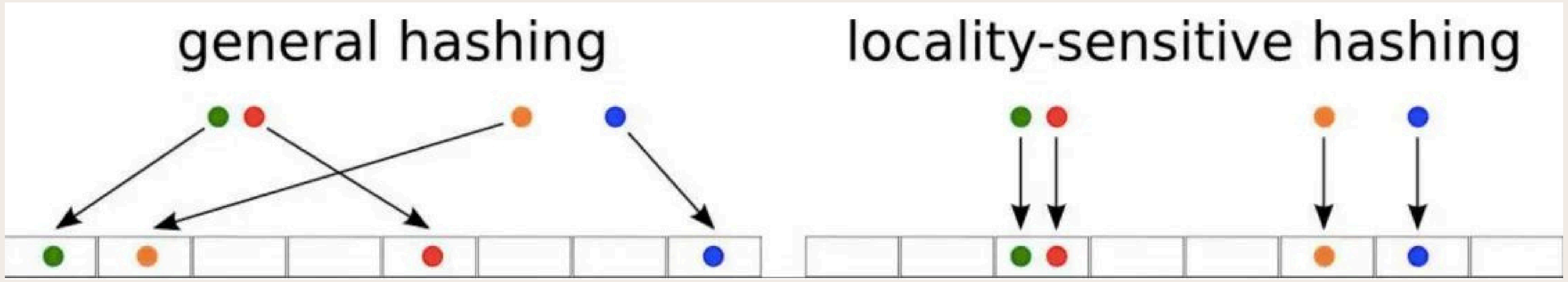
How can we avoid this?

LSH is one algorithm that helps us to reduce that complexity using approximate searches to find nearest neighbour.

WHAT IS LSH?

Locality Sensitive Hashing (LSH) is a technique widely applicable to the approximate similarity search. LSH is an algorithmic approach to clustering similar input points into the same buckets using their hash values.

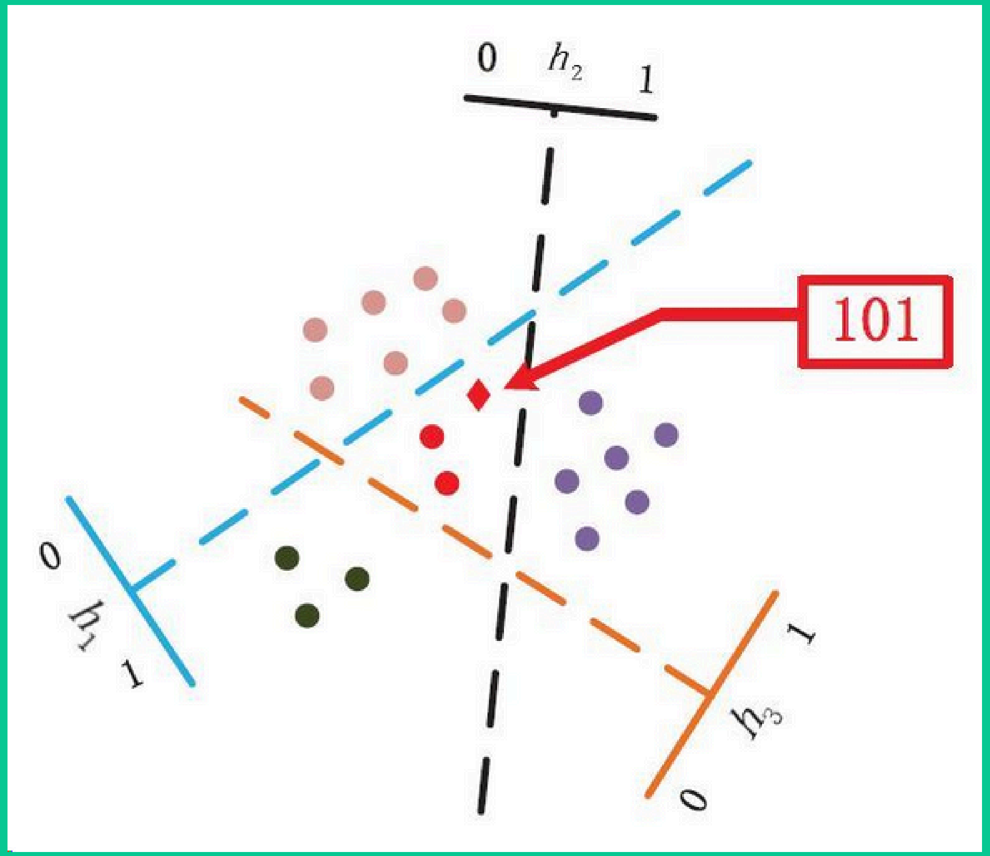


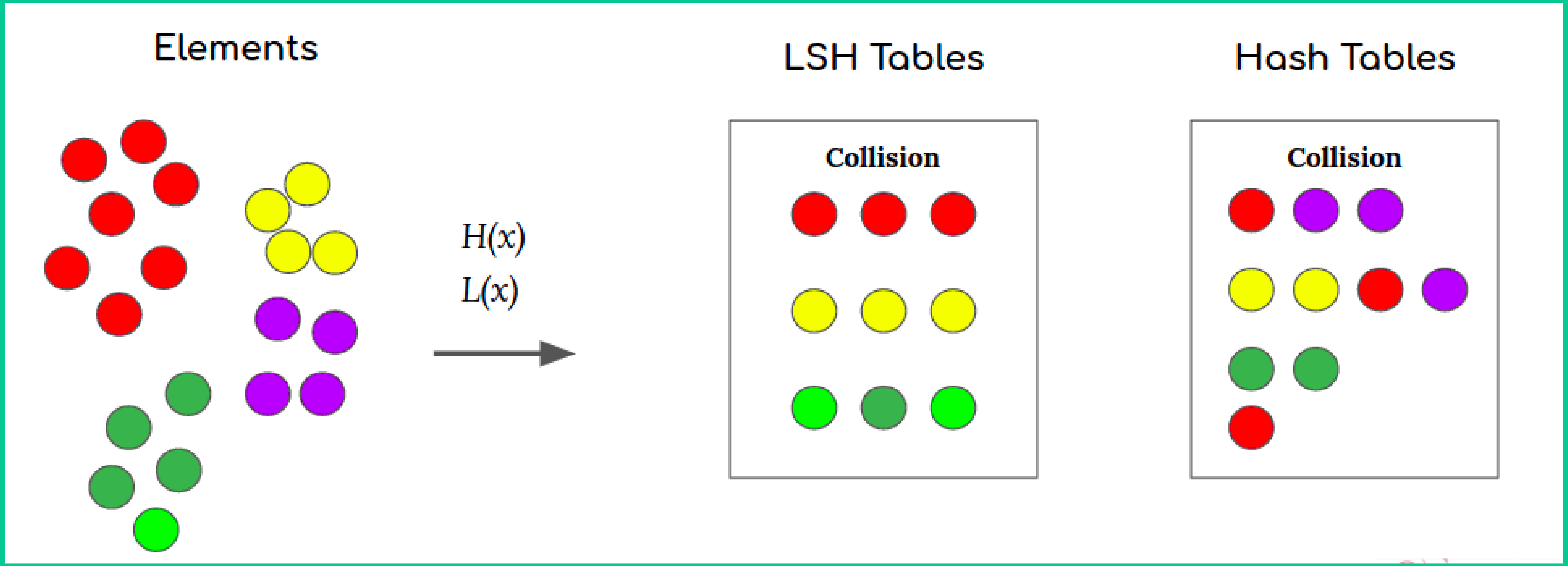


HOW WE IMPLEMENTED?

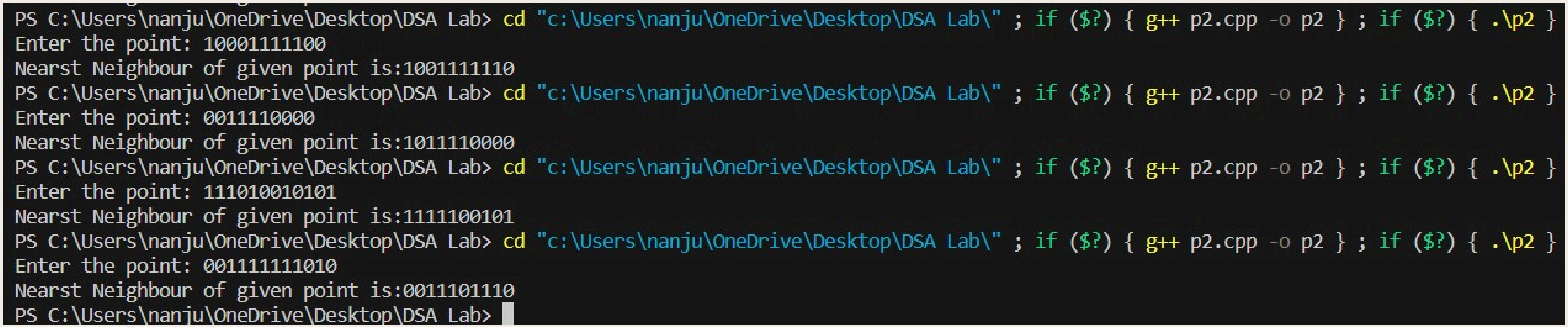
- In this code , we generate n d-D binary vectors.
- We hash all these vectors into each of L hash tables by Locality-Sensitive Hashing (LSH) using k random hyperplanes for each hash function.
- This efficiently searches for the nearest neighbour of a given input query point in a high-dimensional binary space.
- For each hash function we use, one corresponding table is generated, so we ask for NN of a point, the code runs through each table to give us the nearest neighbour.

- The nearest neighbour search(NNS) is performed across multiple hash tables, and the final nearest neighbour is determined by comparing distances across these tables.





OUTPUT



```
PS C:\Users\nanju\OneDrive\Desktop\DSA Lab> cd "c:\Users\nanju\OneDrive\Desktop\DSA Lab\" ; if ($?) { g++ p2.cpp -o p2 } ; if ($?) { .\p2 }
Enter the point: 10001111100
Nearst Neighbour of given point is:1001111110
PS C:\Users\nanju\OneDrive\Desktop\DSA Lab> cd "c:\Users\nanju\OneDrive\Desktop\DSA Lab\" ; if ($?) { g++ p2.cpp -o p2 } ; if ($?) { .\p2 }
Enter the point: 0011110000
Nearst Neighbour of given point is:1011110000
PS C:\Users\nanju\OneDrive\Desktop\DSA Lab> cd "c:\Users\nanju\OneDrive\Desktop\DSA Lab\" ; if ($?) { g++ p2.cpp -o p2 } ; if ($?) { .\p2 }
Enter the point: 111010010101
Nearst Neighbour of given point is:1111100101
PS C:\Users\nanju\OneDrive\Desktop\DSA Lab> cd "c:\Users\nanju\OneDrive\Desktop\DSA Lab\" ; if ($?) { g++ p2.cpp -o p2 } ; if ($?) { .\p2 }
Enter the point: 001111111010
Nearst Neighbour of given point is:0011101110
PS C:\Users\nanju\OneDrive\Desktop\DSA Lab>
```

WHERE DO WE FIND ITS USE?

Near-duplicate detection: LSH is commonly used to deduplicate large quantities of documents, webpages, and other files.

Genome-wide association study: Biologists often use LSH to identify similar gene expressions in genome databases.

MORE...

Spotify recommends good music because of their similarity search algorithms successfully matches you to other customers with a similarly good (or not so good) taste in music.

LSH helps in audio fingerprinting, identifying similar audio clips, or finding relevant audio samples in large databases, making it useful in speech recognition and audio retrieval systems.

CHALLENGES FACED :(

- While creating hyperplanes: Less number of hyperplanes create an issue because points are not divided efficiently.
- Creating hash function: We had convert a point into binary based on the side of plane the point lies in (by using dot product).

HOW TO OVERCOME?

- Multiple hyperplanes not only increase the chances of similar items being hashed to the same bucket but also reduce the probability of collision for dissimilar items.

- By employing multiple hash functions with different hyperplanes, it's possible to enhance both recall (finding all relevant items) and precision NNS.

HOPE YOU HAVE LEARNT SOMETHING NEW!

THANK YOU !