**Department of Computer Engineering**



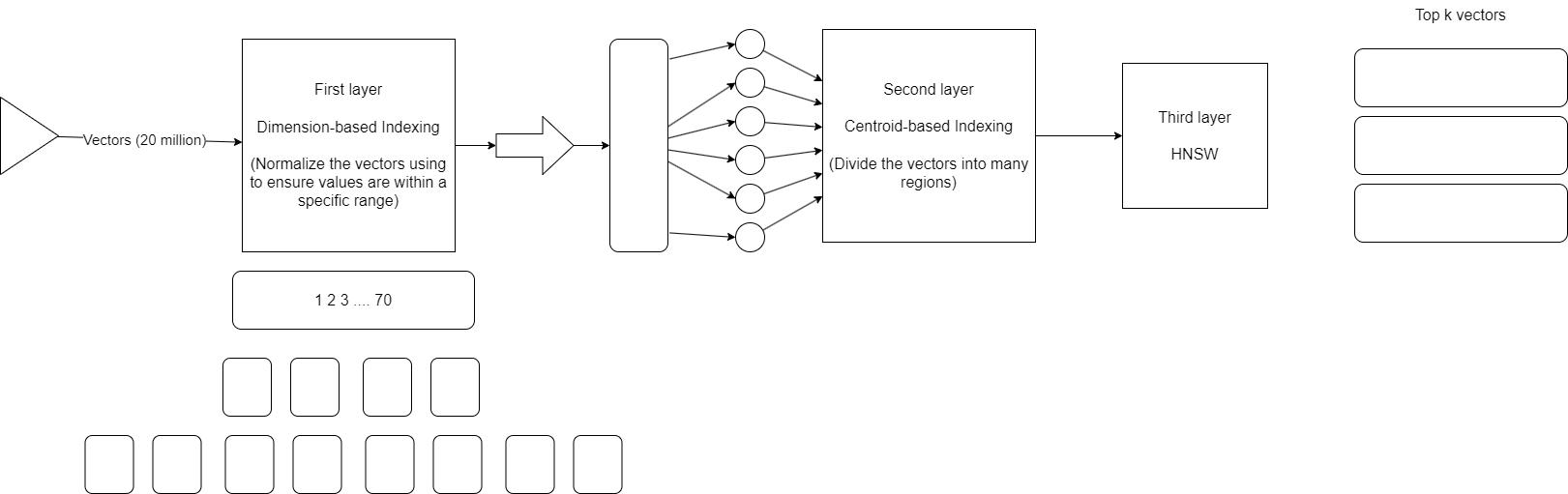
**Cairo University**

**Faculty of Engineering**

**Team 10**

|  |  |
| --- | --- |
| Eslam Ashraf | لا نعلم |
| Beshoy Morad Atya | 9202405 |
| Abdelrahman Hamdy | 9202833 |
| Abdelrahman Noaman | 9202851 |

**Design Image:**



Layer 1:

Input: vector<70>

Output: Name of the file that will store that vector

Description:

* Multi-dimensional Space Partitioning Algorithm: it partitions the vector based on its elements.

Example: vector1: [0.5, 0.1, 0.2, ….]  
First space: [(>0.5), (>0.5), (>0.5), ….]

Second space: [(>0.5), (>0.5), (<0.5), ….]

And so, on

* Data structures: Can simply be a variable size multi-dimensional array.

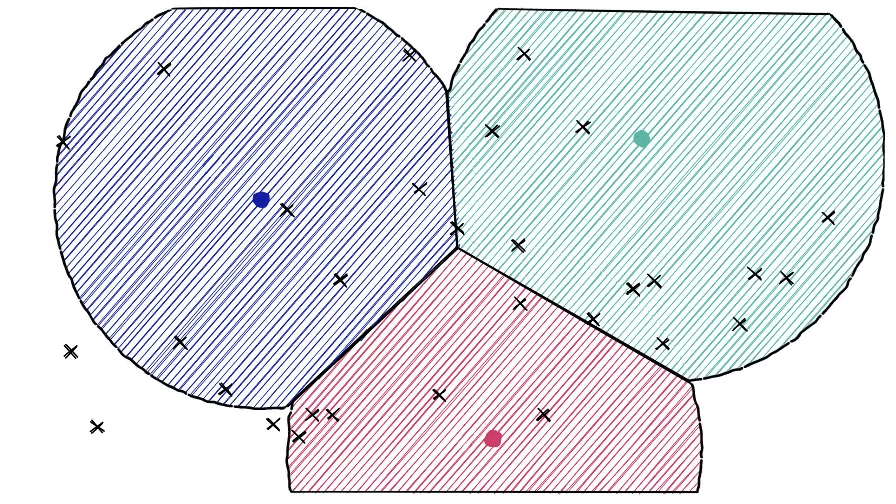
Layer 2:

Input: Single space file from the previous layer containing ~10k vectors (for example)

Output: Index file that have only the centroids of that space ~30 centroid (for example)

Description:

* Centroid-based partitioning: We can use any algorithm that divide the space into centers like (IVF or IVF-PQ).



Layer 3:

Input: Minimum number of vectors which will contain the nearest neighbors

Output: The top K similar vectors to our query

Description:

* Use a Hierarchical Navigable Small World (HNSW) index for the final layer. HNSW is known for its efficiency in finding approximate nearest neighbors in high-dimensional spaces and for its high speed in searching.
* Index the vectors using HNSW, with each node in the graph representing a vector.
* During querying, start at the root node and navigate the graph to find the k most similar vectors.