**Problem formulation:**

Input: A query function \( Q(q, g) \) that given a query graph \( q \) and a data graph \( g \), return a \texttt{Bool} indicating if \( q \) and \( g \) is relevant, say the set of all relevant graphs \( L(q) \).

- i.e. \( Q \): if GED of \( q \) and \( g \) is less than 5 then \( q \) and \( g \) is relevant.

A graph database.

A constant \( k \): \( k \) graphs in the answer set

**Neighborhood of a graph \( g \) (N(g))**: The number of relevant graphs to \( q \), say \( g' \), such that distance(\( g, g' \)) is less or equal to theta.

- i.e. GED of \( g \) and \( g' \) is less than theta.

**Representativeness of a set of graphs \( S \)**: |The union of all the neighborhoods of all graph in \( S \cup L(q) \)|, say \( \pi(S) \). \( \pi(g) \) means the special case that \( S \) only has 1 graph \( g \).

Output: \( k \) graphs in all the relevant graphs such that the \( k \) graph’s representativeness is maximized.

- i.e. top-3 answer could be \( g_1,g_3,g_4 \) if graphs are clustered based on GED and red ones are relevant. Suppose in C1 (left cluster) GED(\( g_1,g_2 \))>theta, then the representativeness of set \( \{g_1,g_3,g_4\} \) would be \((3 \cdot c_1 + 1 \cdot c_2 + 3 \cdot c_3) / \text{(number of all red ones)} = 7/8\)

**General framework:**

We have a pool of all relevant graphs \( L(q) \). For each graph \( g \) in \( L(q) \), it will cover a set of other graphs in \( L(q) \). However, the later picked graph may cover less graph than if it is picked earlier due to some of the graphs were already covered by previous picked graphs. So it is a set cover problem. Essentially we will use the greedy algorithm to find an approximate answer set.

However, each time we pick one graph to the answer set the covering power of the rest graphs are reduced. This reduction (update) of covering power (number of new neighbor graphs) is very expensive to compute in the scenario of graph edit distance. As a result, this paper build its index to help accelerate the updates of each graph’s covering power and prune some unnecessary GED computation.

**Building Index:**

1. Vantage Points:
A set of graphs in database are chosen. (Each graph is called a vantage point). The Index will record the GED between each vantage point to EVERY graph in database.

2. NB-Tree:

Before building the NB-tree index, we will set the Fan-out of the index tree: b (This is set by the user)

Step 1: There is one cluster C1 containing all graph in the database. C1 is the root of INDEX tree.

Step 2: Randomly choose 1 graph as cluster centroid.

Choose 1 graph that is farthest to the previous graphs in current depth until b fan-out are met.

i.e. depth 1: b=2, g3 is chosen randomly then we choose g5 as the second centroid since GED(g3,g5) is largest.

The rest of the graphs (Not selected as centroid) will be assigned to its closest centroids.

i.e. g2,g4 are closer to g3; g1 is closer to g5

Step 3: Repeat step 2 until all the clusters have less or equal to b graphs.

Property: 1. Leaves are all graphs in database

2. Non-leaf is a cluster of graphs. Two nod-leaf overlap iff one is the parent of the other. (clusters are disjoint) (i.e. c2 and c3 share no common graph in database)
Notice: We can expedite the NB-Tree building by avoiding unnecessary GED computation with the help of lower bound of estimation using vantage point.

Lower bound = max of $|d(v, g) - d(v, g')|$ for all v in Vantage points (Metric triangular inequality)

\[
\begin{align*}
\text{radius}(c) &= \max \{d(\text{centroid}(c), g) \forall g \in c\} \\
\text{diameter}(c) &= \max \{d(g, g') \forall g, g' \in c\} \leq 2 \times \text{radius}(c)
\end{align*}
\]

Radius of a cluster and diameter of a cluster is also stored. Which is used to compute the upper bound and lower bound of the distance that any graph g to any graph g' in c.

\[
\begin{align*}
d_{ab}(g, c) &= d(g, \text{centroid}(c)) + \text{radius}(c) \\
d_{lb}(g, c) &= \max \{0, d(g, \text{centroid}(c)) - \text{radius}(c)\}
\end{align*}
\]

Query Processing:

1. Initialization phase
   - For each leaf g we compute $\pi'(g)$ which is a vector of length $\theta$.
   - $\pi'(g) = \{\text{upper bound of } \pi(g) \text{ when GED threshold}=1, \ldots, \text{upper bound of } \pi(g) \text{ when GED threshold}=i, \ldots \text{upper bound of } \pi(g) \text{ when GED threshold} = \theta\}$
   - How to compute upper bound of $\pi(g)$ when GED threshold $=i$?
     - The union of all g' such that $|d(v, g) - d(v, g')| \leq i$ for all v in Vantage Points.
     - $\tilde{N}(g) = \{g'|d(v, g) - d(v, g')| \leq i \forall v \in \forall\}$.
   - The size of this union is an upper bound of the number of graphs that GED is less or equal to i.
   - For the rest nodes, $\pi'(c \text{ is a cluster and a non-leaf}) = \{\text{the ceiling of all its children when threshold}=i, \ldots\}$
     - i.e. $\pi'(g3) = \{0.6,1\}$, $\pi'(g4) = \{0.4,1\}$. Then $\pi'(c4) = \{\max(0.6,0.4),1\}$

2. Search and update
   a. Search
      - A priority queue initially store the root of the NB-Tree. The priority queue is sorted on representativeness.
      - Each time, we dequeue the top of the queue.
      - If it is a non-leaf and the marginal representativeness gain of adding it is less than the best gain we have seen so far by adding a graph to $S$, current iteration ends go to Update. (Since $\pi'(g)$ records the upper bound of gains for each )
If it is a non-leaf, we add all the children of the node.

Only when the marginal representativeness gain of adding current non-leaf cluster is larger than the best gain we have seen so far by adding a graph to $S$, we will add the non-leaf cluster to the queue.

i.e. if best gain so far is 0.5 then only $c_2$ will be added in above figure when theta=1, since $c_2=[0.6,1]$, $0.6>0.5$

If it is a leaf, which means it is a graph $g$.

We can get the union of all $g'$ such that $|d(v, g) - d(v, g')|\leq i$ for all $v$ in Vantage Points, given $g$ and Vantage Points, denoted as $N'(g)$.

For any $g'$ in $N'(g)$, if the GED$(g,g') \leq$ theta (Real GED computation here), which means $g'$ is a theta neighborhood of $g$ and $g'$ contributes to the representativeness of $g$. Then we can get the union of all $g'$ that is the neighborhood of $g$, denoted as $N(g)$.

With $N(g)$, we can compute the marginal representativeness when adding $g$ to the current answer set $S$. Only record the graph with the largest marginal gain.

When all the nodes are popped, we know which graph to be added to $S$. Repeat until size of $S$ is $k$.

b. Update

The core of this paper and the reason that we can’t use a set cover setting directly.

When adding a graph $g$ to $S$, some $\pi'(g)$ in NB-Tree will be impacted.

1. In case of $c_4$ which means the rest of the relevant graphs that the distance between it and $g$ is larger than 2 * Theta. No impact. (lower bound is computed with the help of Index)
2. In case of $c_1$ and $c_2$, which means the upper bound of GED $(g$, any $g'$ in $c)$ is less than Theta and the diameter of $c$ is less than Theta, all the neighborhood representativeness of any node in $c$ are reduced to zero. ($g_1$ covered all neighbors of graph in $c$)
3. In case of $c_3$, which means the upper bound of GED $(g$, any $g'$ in $c)$ is less than Theta and the diameter of $c$ is less than $\frac{1}{2}$ * Theta. We reduce the representativeness of graphs in $c$ by the value of the sum of the representativeness of $c_1$ and $c_2$. 

![Diagram of sets and neighborhoods]