Nearest Neighbor Search (NNS):
- Given a data set S of n points in R^d and a query point q, report p in S which is nearest to q.
- Applications: Web search, Information retrieval etc.
- Exact NNS suffers from the "curse of dimensionality".
- Space/time required is exponential in d.

C-Approximate Nearest Neighbor Search (c-ANNS):
- Given a data set S of n points in R^d and a query point q, if p is nearest neighbor of q, report p' in S s.t. ||p'-q|| ≤ cl,
- The (c,l)-Near Neighbor problem or (c,l)-NN problem:
- Given a data set S and a query point q, if there exists a point p in S, ||p-q|| ≤ cl, report a point p' in S, ||p'-q|| ≤ cl.
- The (c,l)-NN problem is the decision version of the c-ANNS problem.

Solving the (1,c)-NN problem:
- Dimension Reduction:
  For e.g. Use of Johnson Lindenstrauss lemma to reduce no. of dimensions to O(log n).
- Locality Sensitive Hashing (LSH):
  Construct locality sensitive hash functions g: R^d → {0,1} such that
  if ||p-q|| ≤ l, P(g(p)=g(q)) ≥ p_1
  if ||p-q|| ≥ cl, P(g(p)=g(q)) ≤ p_2
  where p_1 denotes the TCAM match operation.

  Construction: of a ternary hash family G_δ: Let g be a ternary hash function in this family.
  a: random vector in R^d, drawn from N^d(0,1)
  b: chosen uniformly from (0,2^δ)
  c: chosen as a function of n and any other parameters.
  A point in S in the TCAM of width w.

  TLSh Theorem: G_δ is a (1,c,p_1(δ),p_2(δ/c))-TLSh family where
  p_1(δ) = 1 – exp(-c^2/2)(δ^2), and
  p_2(δ) = 1 – exp(-c^2/2)(5 δ^2).

  Comparision with LSH:
  - Lower Bound: Motwani et. al. showed that
    p ≥ 0.45/c^2.
  - On the other hand, for a TLSh family we have
    p = exp(-δ^2/2)((c^2)/c^2),
    which decreases to 0 as δ increases.

  Applications to Similarity Search and Nearest Neighbors:
  - The (1,c)-NN problem:
  - Algorithm:
    Parameters (w,δ) chosen as a function of n and error probability ε and separation c.
  1. Preprocessing: Choose w hash functions from a TLSh family and store the images of each point in S in the TCAM of width w.
  2. Runtime: Given a query q, find its TCAM representation T(q) using the same hash functions and perform a TCAM lookup of T(q). If the point returned p is at a distance of at most c from q, report “Yes” and that point as output, otherwise report “No”.

  In order to decrease the width of the TCAM required, it is possible to choose (w,δ) so that algorithm succeeds with a constant probability say(1/2) and repeating the algorithm O(log(1/ε)) times, it can be made to succeed with high probability.

  Analysis:
  - Analysis in terms of false negatives and false positives.
    - False negative probability is 1-p_1^w.
    - False positive probability w.r.t. a specific point is p_2^w.
    - Increasing δ reduces the probability of false negatives.
    - Increasing w increases the chance of false positives.
    - Choosing δ = O((log log n)/w) and width w = O(log (log n)/(log log n)) ensures that the false negative probability and expected no. of false positives can be kept small.

  Granularity:
  - If the width is restricted to O(log n), the TCAM can solve the (1,c)-NN problem when
    c = Θ((log log n)).
  - If LSH family is used along with a RAM of width O(log n), the (1,c)-NN problem can be solved only when c = O((log n)).
  - This implies an exponential improvement in the granularity by using TLSh method.

  Experiment:
  - TCAMs are commercially available in sizes of 72, 144, 288 bits.
  - Experimenting a TCAM of 288 bits, experiments were carried out on Wikipedia data (a snapshot of Wikipedia) obtained from Yahoo which contained a million data points (web pages).
  - A small percentage of false negatives (less than 1%) were observed and no false positives were observed.