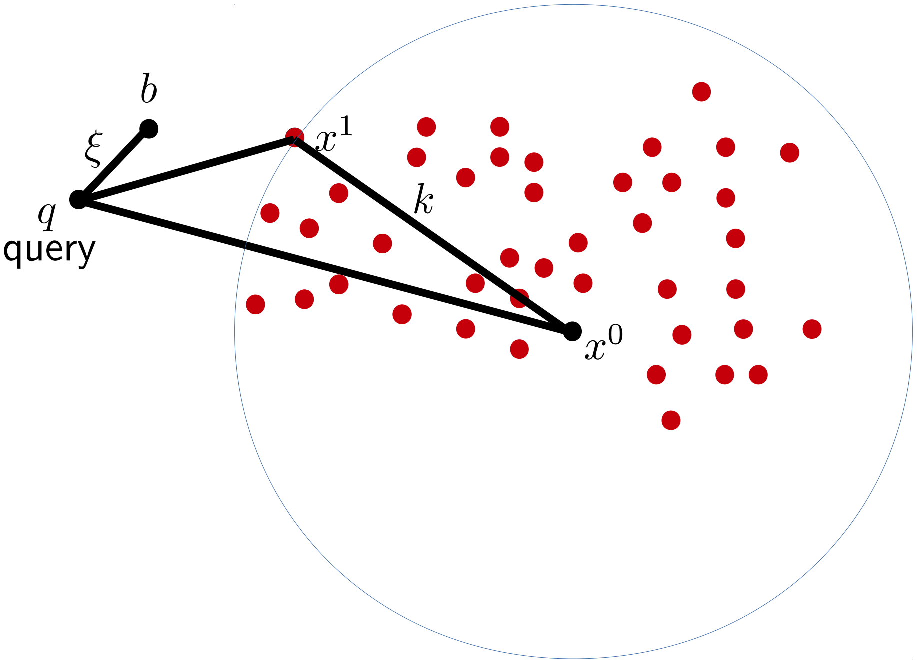
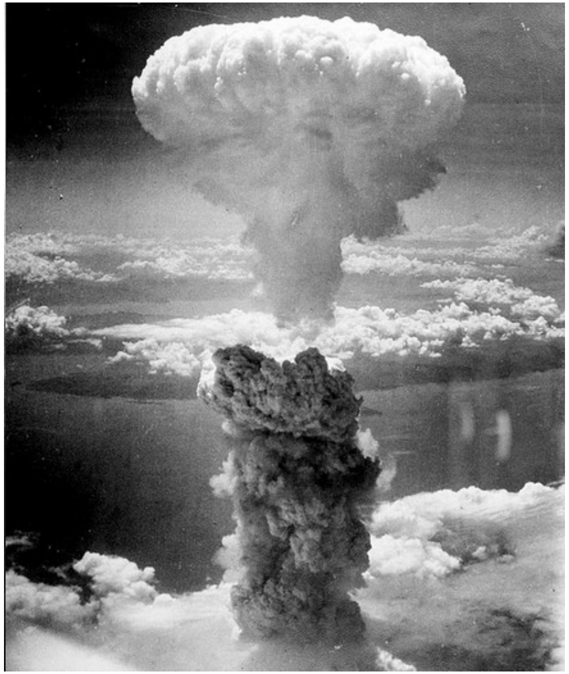
BK Trees

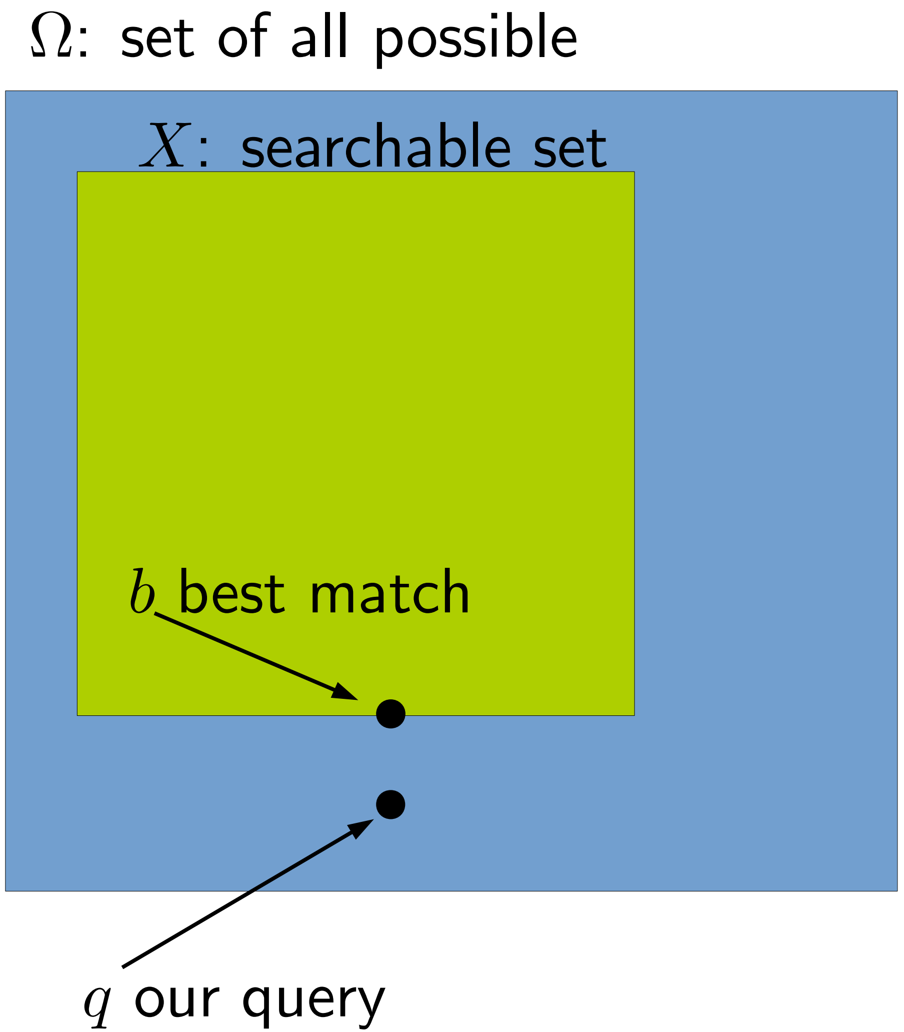
-or-

how I learned to stop worrying and love the triangle inequality





The problem: finding closest matching in a set



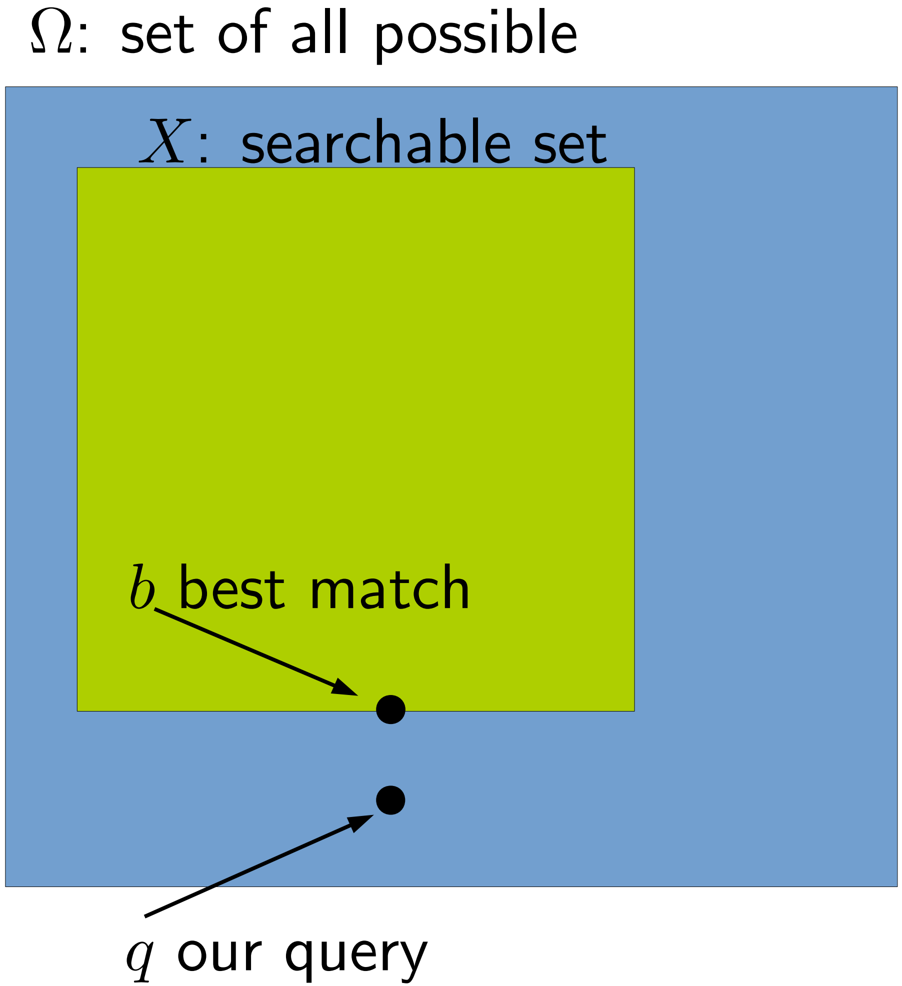
Example applications:

- Spell checkr
- Genetics
- Classification

Requirements:

- Distance measurement $d(x_i, x_j)$

Example approaches: brute search, binary search, hash



Brute:

$O(|X|)$ for every query.

Binary:

$O(|X|\log(|X|))$ for set-up

$O(\log(|X|))$ queries

Requires ordering property

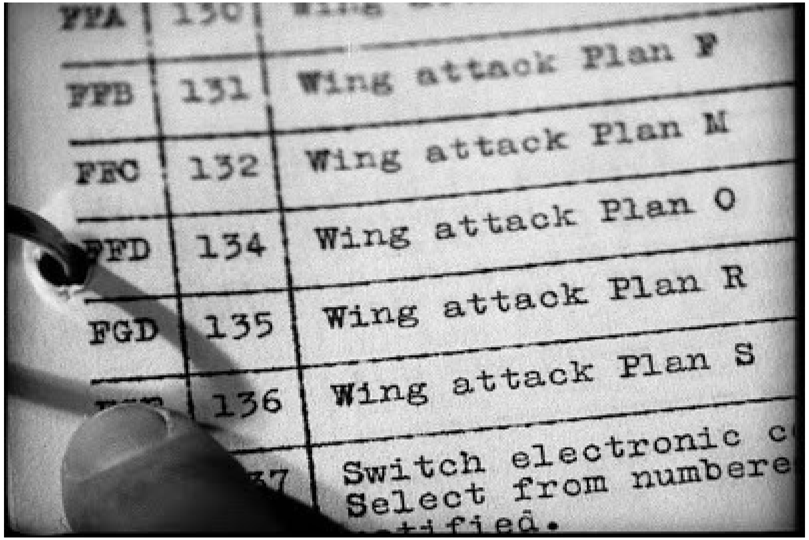
Hash:

$O(1)$ lookup

$O(\Omega^2)$ space to store $d(x_i, x_j)$.

Note: Ω is combinatorical

A concrete example: closest word in a set of words



FFA	130	Wing
FFB	131	Wing attack Plan F
FFC	132	Wing attack Plan M
FFD	134	Wing attack Plan O
FGD	135	Wing attack Plan R
[illegible]	136	Wing attack Plan S
	37	Switch electronic c Select from numbere tified.

Code is FGD 134?
Just do attack plan R

Rudimentary spell checkr

checkr is wrong, but what is right?

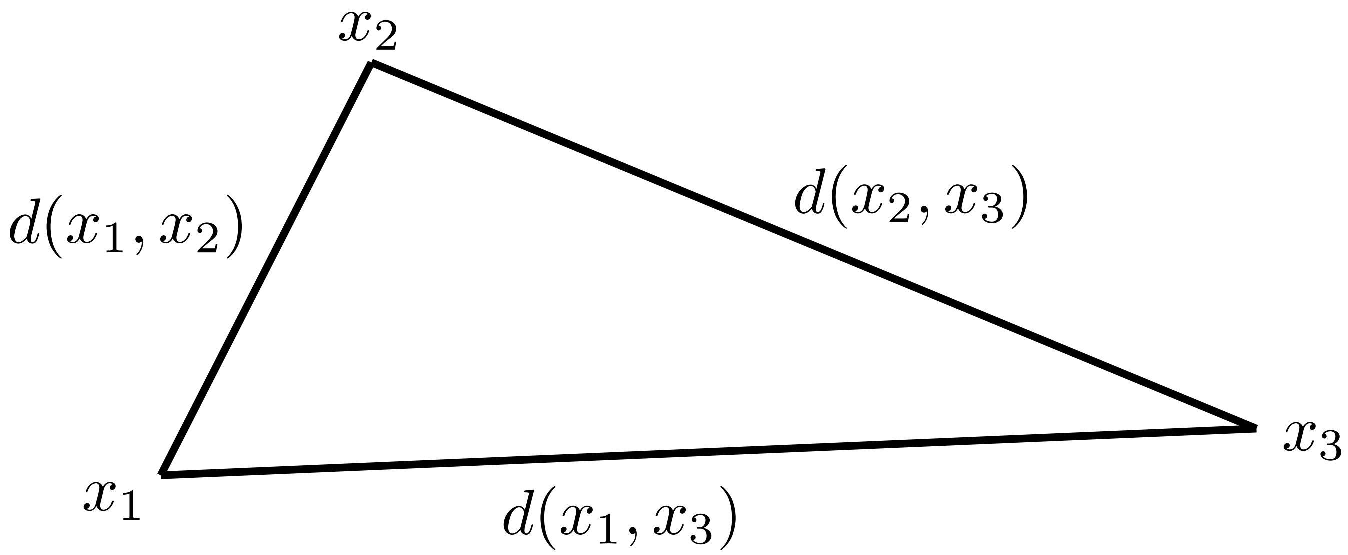
$q =$checkr

$b =$ checker

Note: you may also get check

Basic idea: many metics obey triangle inequality

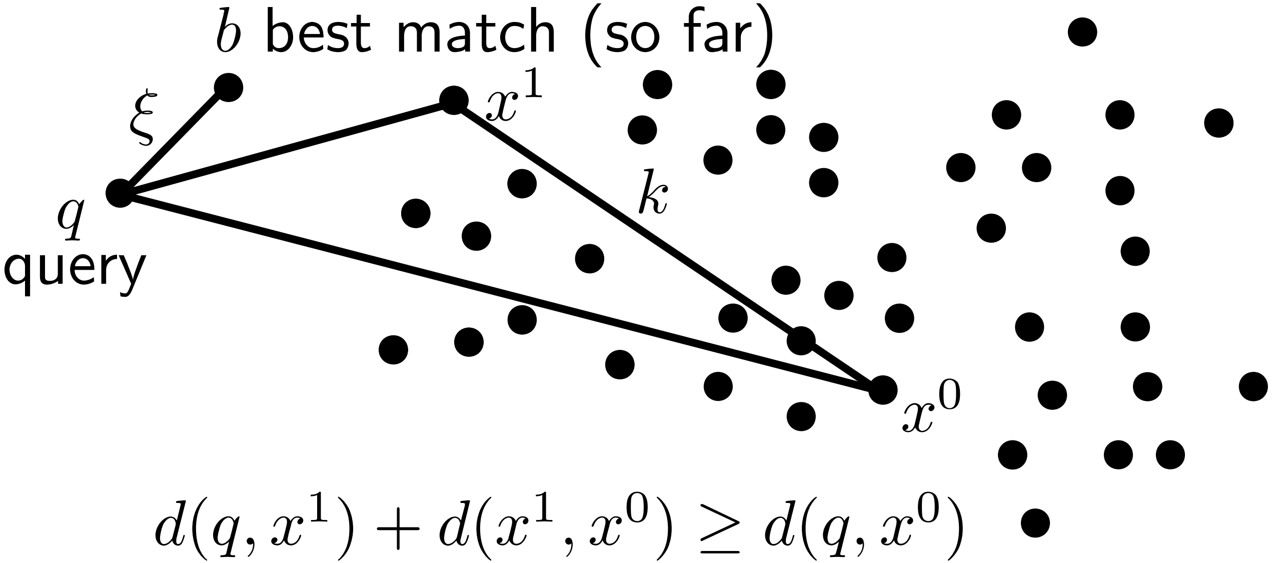
$$d(x_1, x_3) \leq d(x_1, x_2) + d(x_2, x_3)$$



BK-trees will **require** this property

BK cutoff criterion 1

$X_1 : \{x \in X | d(x, x^0) \leq k\}$



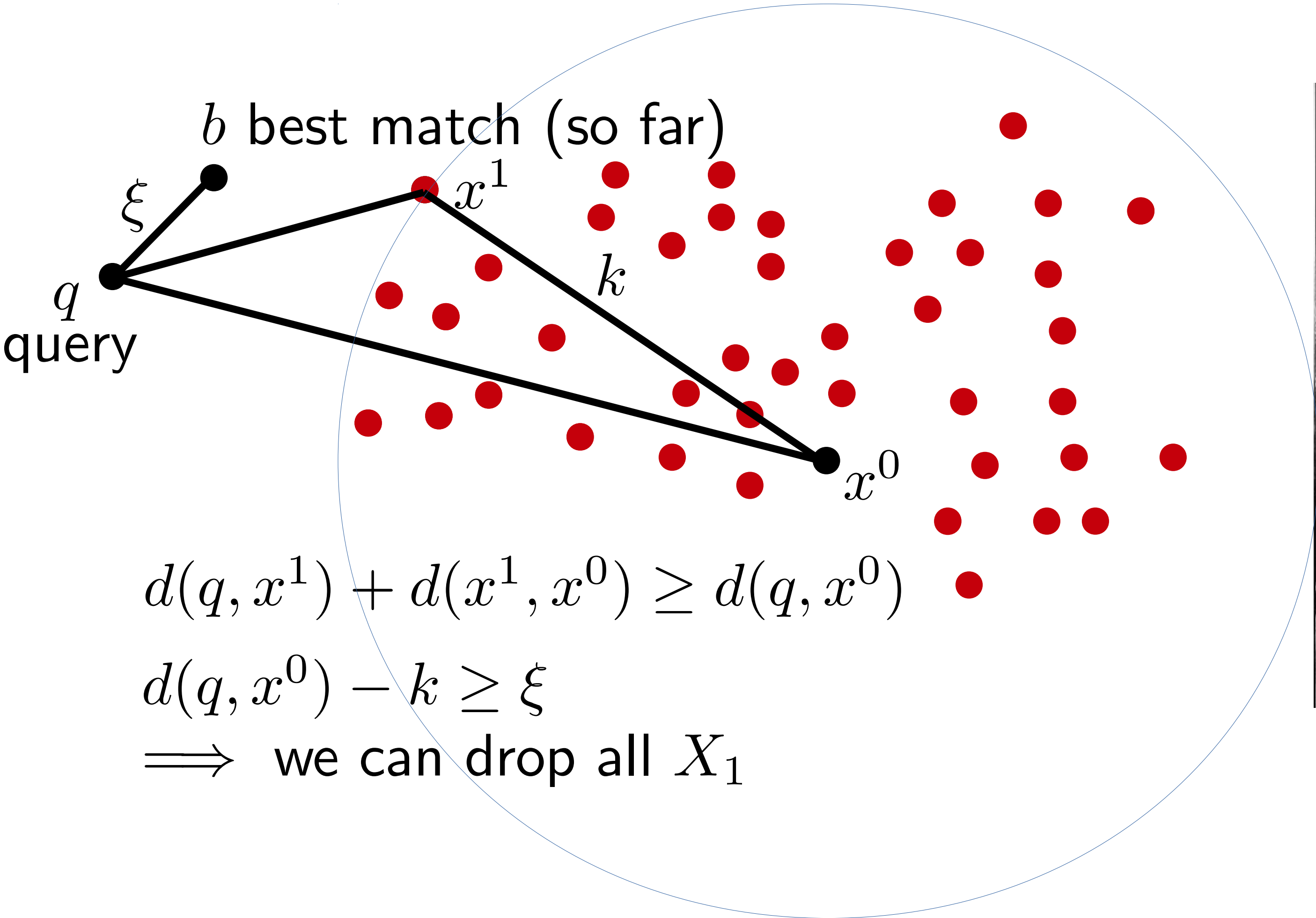
$$d(q, x^1) + d(x^1, x^0) \geq d(q, x^0)$$

$$d(q, x^0) - k \geq \xi$$

$\implies$ we can drop all X_1

BK cutoff criterion 1

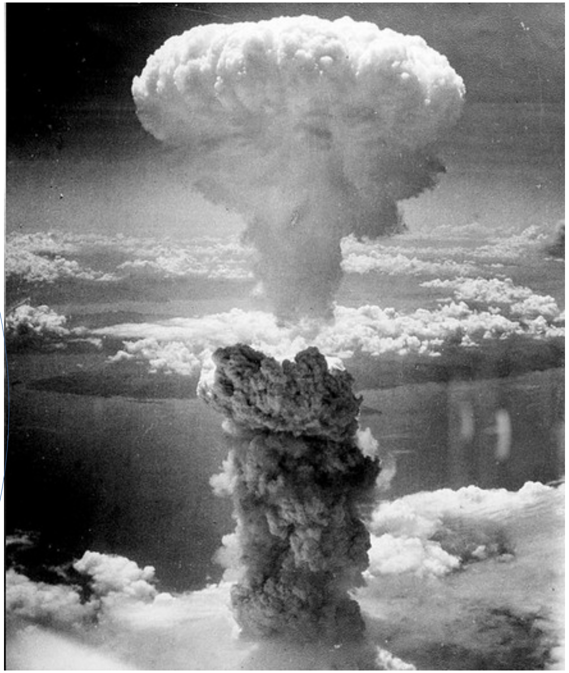
$$X_1 : \{x \in X | d(x, x^0) \leq k\}$$



$$d(q, x^1) + d(x^1, x^0) \geq d(q, x^0)$$

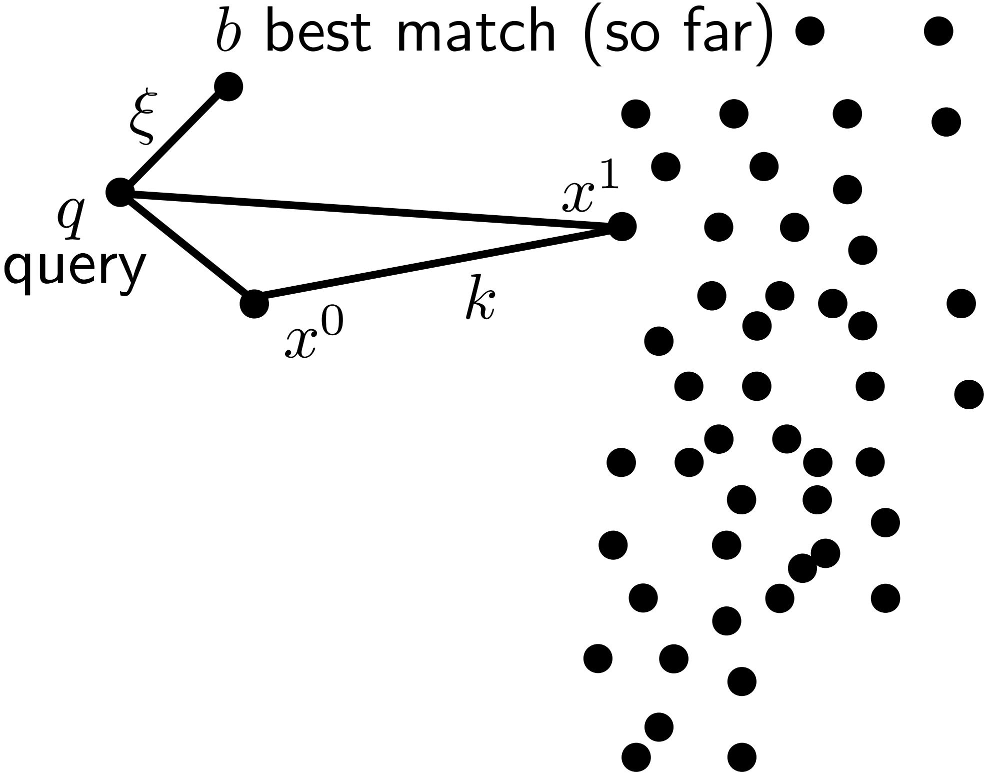
$$d(q, x^0) - k \geq \xi$$

$\implies$ we can drop all X_1



BK cutoff criterion 2

$X_2 : \{x \in X | d(x, x^0) \geq k\}$



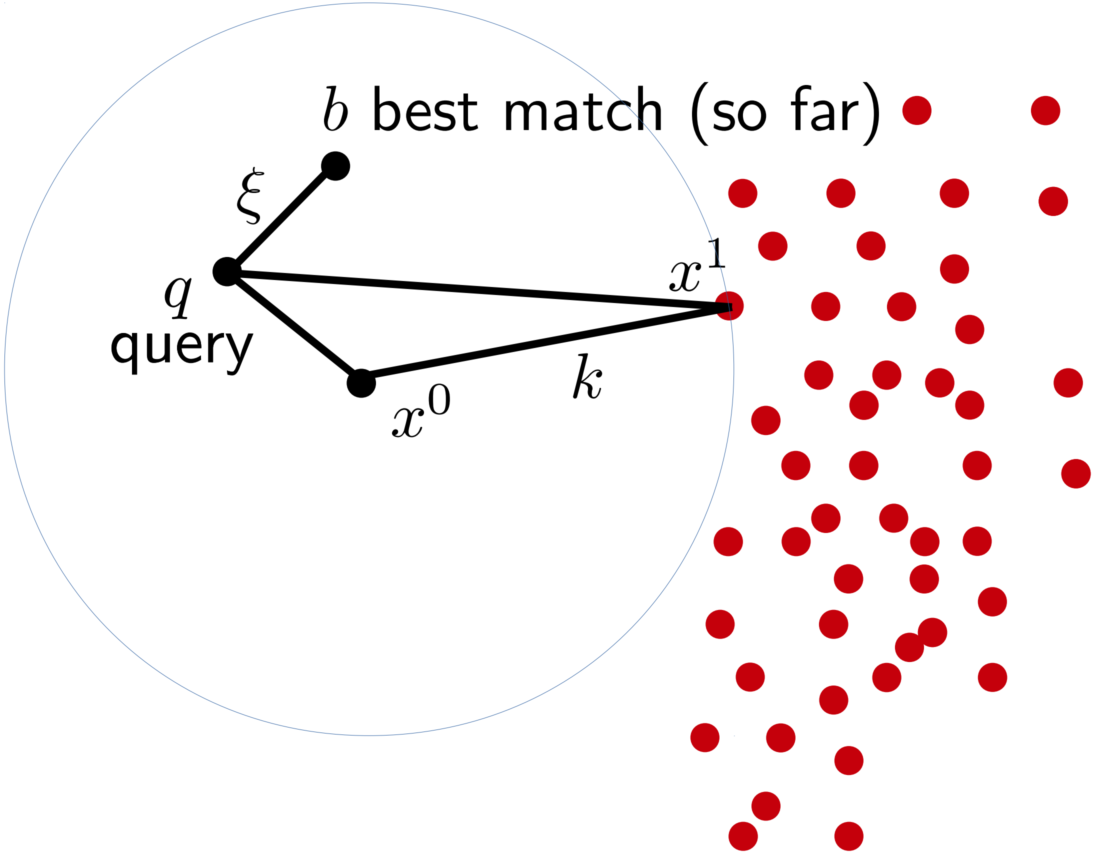
$d(q, x^1) + d(q, x^0) \geq d(x, x^0)$

$k - d(q, x^0) \geq \xi$

$\Longrightarrow$ we can drop all X_2

BK cutoff criterion 2

$$X_2 : \{x \in X | d(x, x^0) \geq k\}$$



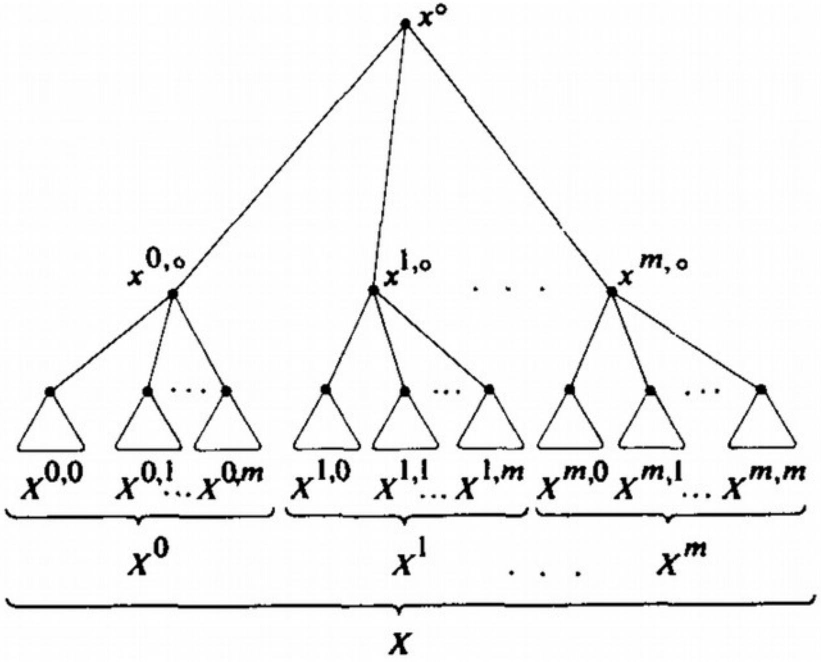
$$d(q, x^1) + d(q, x^0) \geq d(x, x^0)$$

$$k - d(q, x^0) \geq \xi$$

$\implies$ we can drop all X_2

Algorithm

Fig. 1. File Structure 1 viewed as a tree.



Query code

```
def query(self,key):
  jdist=self.compare(self.x0,key)
  best={'word':self.x0,'dist':jdist}

  # Search child trees.
  for child_dist,child in self.children.items():

    # Apply joint cutoff criterion.
    if abs(child_dist - jdist) < best['dist']:
      child.ncompares=0
      new=child.query(key)
      self.ncompares+=child.ncompares

      # Possibly update keys and best distance.
      if best['dist'] > new['dist']:
        best=new

  return best
```

Performance

/usr/share/dict/words (99171 words)

Mean number of comparisons needed for 500 random queries queries of the form 'checkr'

Percent of possible queries needed

Brute force: 98.1 %

BK-tree: 34.2 %

(non-optimized x_0)