Exploiting Length

Suffix Length
Three-Dimensional Indexes

Adding Length to the Mix

- We can index on three attributes:
  1. Character at a prefix position.
  2. Number of that position.
  3. Length of the *suffix* = number of positions in the entire string to the right of the given position.

Edit Distance Revisted

- Suppose we are given probe string *s*, and we find string *t* because its *j* th position matches the *i* th position of *s*.
- A lower bound on edit distance E is:
  1. *i* + *j* – 2 plus
  2. The absolute difference of the lengths of the *suffixes* of *s* and *t* (what follows positions *i* and *j*, respectively).

LCS Revisited

- Suppose we are given probe string *s*, and we find string *t* first because its *j* th position matches the *i* th position of *s*.
- If the suffixes of *s* and *t* have lengths *k* and *m*, respectively, then an upper bound on the length C of the LCS is 1 + min(*k*, *m*).

Bound on Jaccard Distance

- If J is the limit on Jaccard distance, then the requirement that $E/(E+C) \leq J$ becomes:

- $i+j-2+|k-m| \leq J(i+j-2+|k-m|+1+\min(k,m))$

- Thus: $j+|k-m| \leq (J(i-1+\min(k,m))-i+2)/(1-J)$

Positions/Prefixes/Suffixes – Indexing

- Create a 3-attribute index on (symbol, position, suffix-length).
- If string *s* has symbol *a* as the *i* th position of its prefix, and the length of the suffix relative to that position is *k*, add *s* to the bucket (*a*, *i*, *k*).

Example: Indexing

- Consider string s = abcde with J = 0.2.
- Prefix length = 2.
- Index in: (*a*, 1, 4) and (*b*, 2, 3).

Lookup

- As for the previous case, to find candidate matches for a probe string *s* of length L, with required similarity J, visit the positions of *s*'s prefix in order.
- If position *i* has symbol *a* and suffix length *k*, look in index bucket (*a*, *j*, *m*) for all *j* and *m* such that

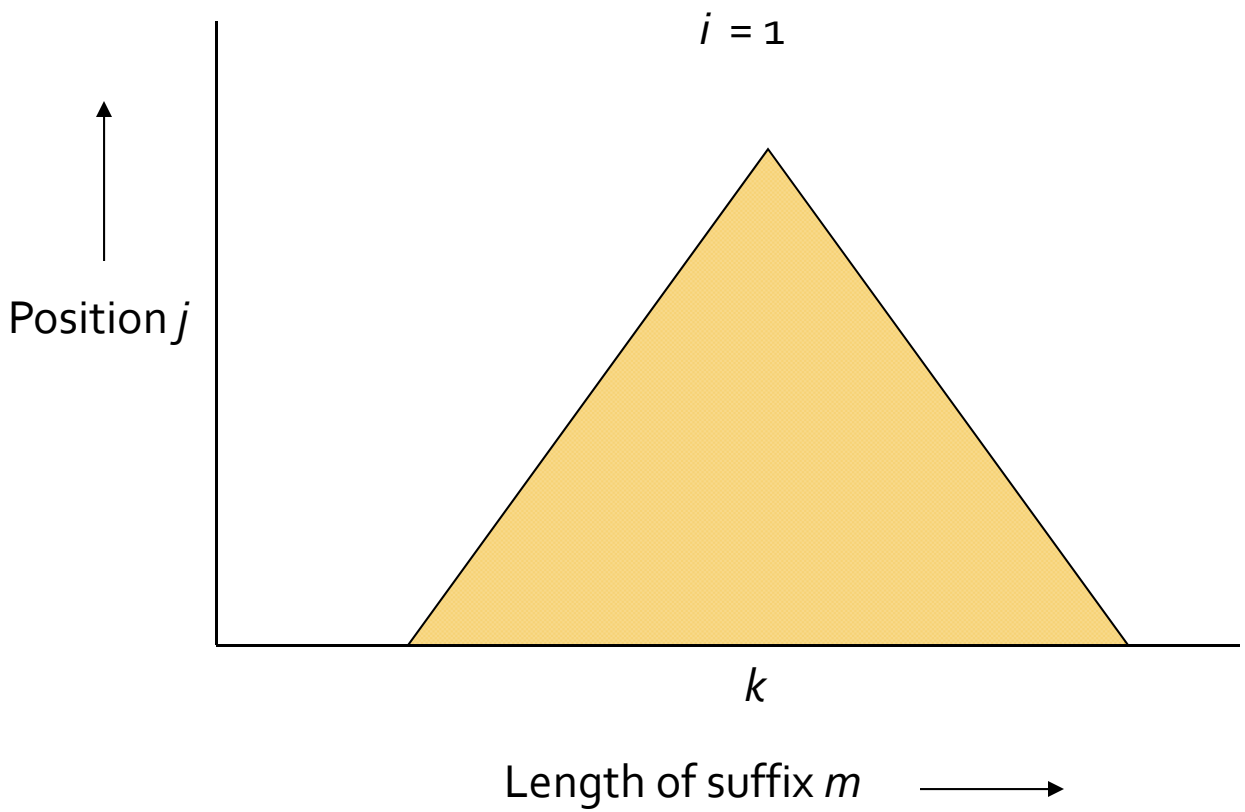
$$j + |k - m| \leq \left(\mathrm{J}(i - 1 + \min(k, m)) - i + 2\right)/(1 - \mathrm{J}).$$

Example: Lookup

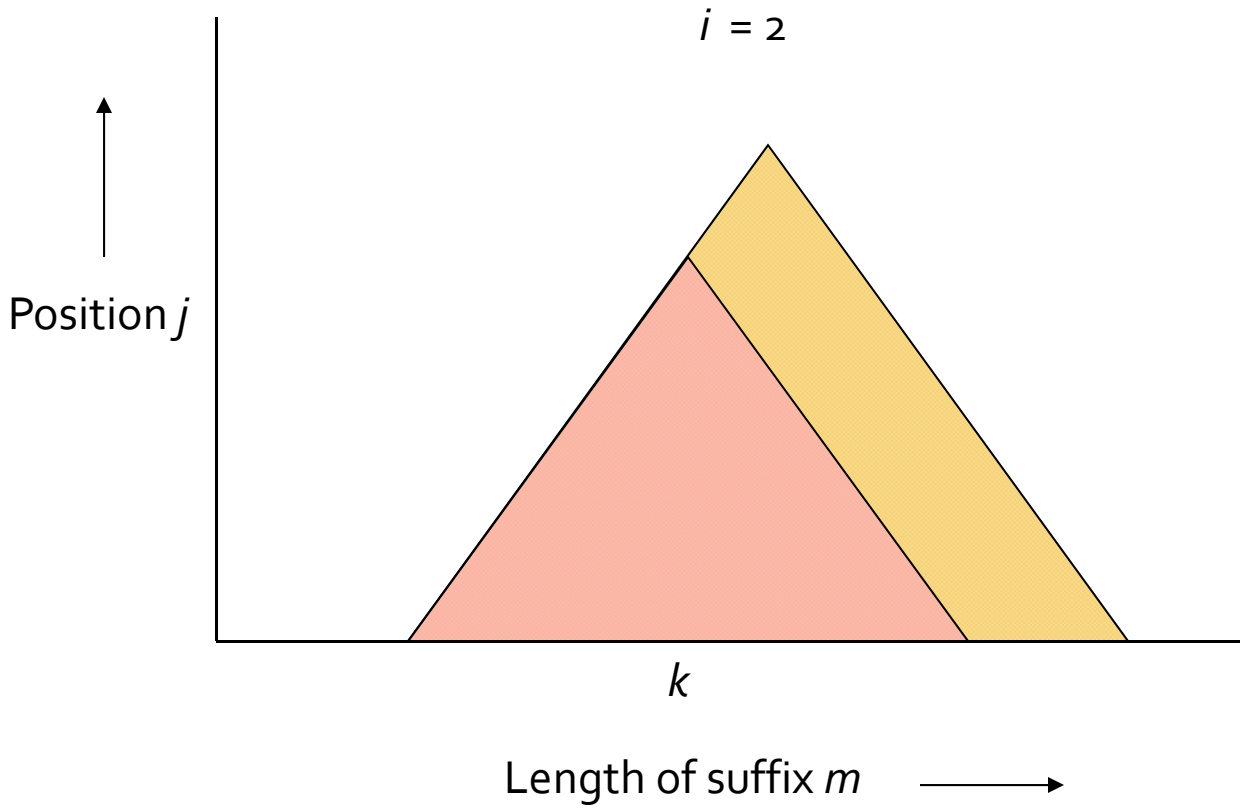
- Consider s = abcde with J = 0.2.
- Require: $j + |k-m| \leq (J(i-1+\min(k,m))-i+2)/(1-J)$.
- Look in (a, 1, 3), (a, 1, 4), (a, 1, 5), (a, 2, 4), (b, 1, 3).
- For i = 1, note k = 4. We want $j + |4-m| \leq (0.2\min(4,m)+1)/0.8$.

From i = 2, k = 3,
j + |3–m| ≤ 0.2(1+min(4,m))/0.8.

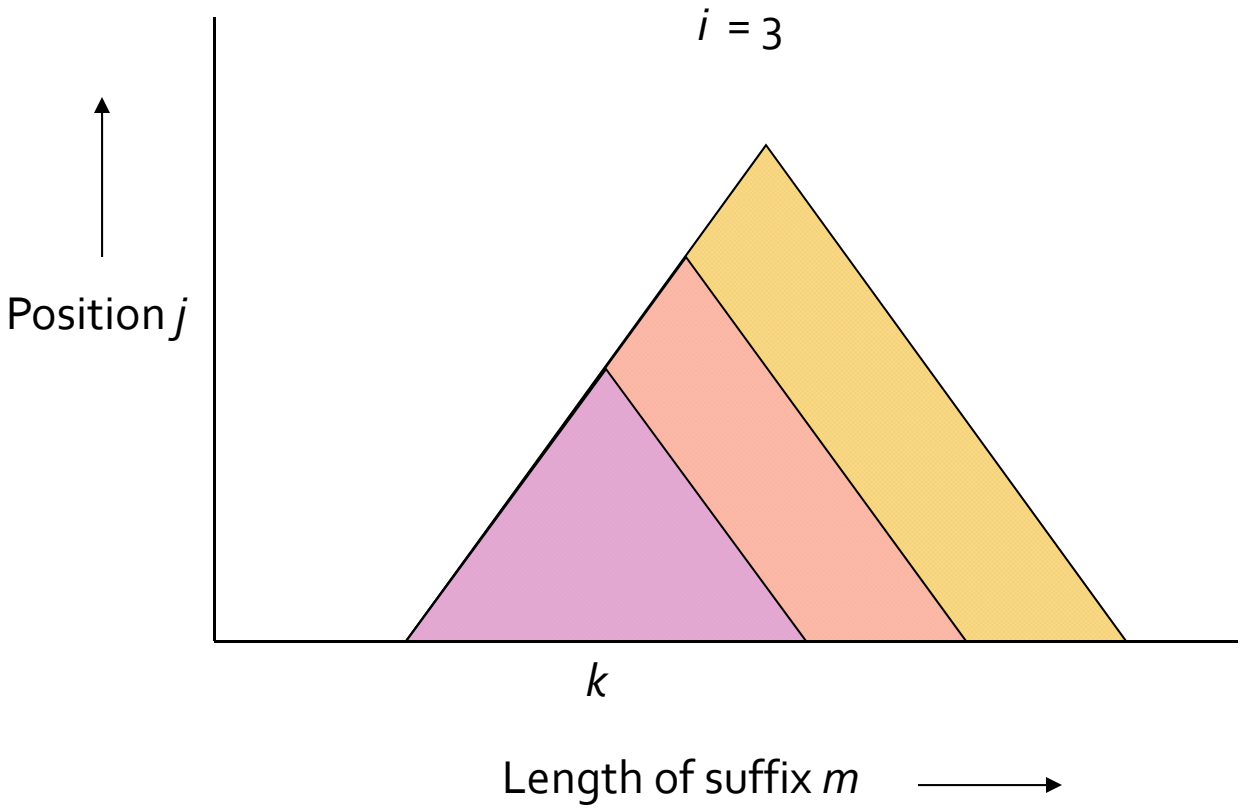
Pattern of Search



Pattern of Search



Pattern of Search



Summary

- We saw three index schemes:
  1. Symbol
  2. Symbol + position
  3. Symbol + position + suffix length
- The number of buckets grows as we add dimensions to the index, but the total size of the buckets remains the same.
  - Because each string is placed in $\lfloor JL+1 \rfloor$ buckets.
- Adding positions roughly halves the number of candidates.
- Adding suffix lengths is more powerful.