CS342: Bioinformatics
Lecture 4

Multiple Pattern Matching with Keyword Trees

Runtime? Assume N is sum of lengths of patterns, m is the length of the text, and n is length of longest pattern

$$O(N + nm)$$

Question: Is this better than brute force? [Think, pair, share]

Suffix Trees

Stores all suffixes of a text $t_1,\ldots,t_m$

- Similar to keyword tree, except edges that form paths are collapsed.
- All internal vertices have at least two outgoing edges
- Leaves labeled by index of pattern in text.

Ukkonen's Algorithm

- Builds a suffix tree for $s = s_1 s_2 \ldots s_m$ in $O(m)$ time.
- https://brenden.github.io/ukkonen-animation/

Keyword Trees vs. Suffix Trees

- Keyword and suffix trees are useful data structures supporting various pattern finding problems
- ***Keyword trees***:
  - Build keyword tree of patterns, and ***thread text*** through it
- ***Suffix trees***:
  - Build suffix tree of text, and ***thread patterns*** through it

Knuth-Morris-Pratt Algorithm

- Solves a version of the basic pattern matching problem.
- Rather than shifting p by one at each iteration (brute-force), use info about p to never go “backwards".

Input: Text $t = \ t_0 \dots t_m$ and pattern $p = \ p_0 \dots p_m$ (0 index)

Output: Index of the first occurrence of p in t.

- Step 1: Compute a table T based only on pattern p that tells us where the pattern contains potential repeats.
- Step 2: Use T to search for the first occurrence of p in t.

Computing T

- T is table of size length of p.