CS 466
Introduction to Bioinformatics

Instructor: Jian Peng
Teaching Assistant: Wesley Qian
Goal: Find all occurrences of a pattern in a text

Input: Pattern $p = p_1 \ldots p_n$ and text $t = t_1 \ldots t_m$

Output: All positions $1 \leq i \leq (m - n + 1)$ such that the $n$-letter substring of $t$ starting at $i$ matches $p$

Motivation: Searching database for a known pattern
Pattern Matching: Running Time

• Naïve runtime: $O(nm)$
  • How?

• On average, it should be close to $O(m)$
  • Why?

• Can solve problem in $O(m)$ time?
  • Yes, we’ll see how (in a later lecture)
Naive algorithm is inefficient

As we saw, our alignment algorithms scale as $O(nm)$. When $n \approx 10^9$ and $m \approx 10^2$ this becomes intractable (especially when we 10 of millions of strings of length $\sim m$)

Even ignoring, e.g memory access, say filling in each matrix cell takes $C = 10$ CPU cycles.

$N = 10^9$  
$M = 10^2$  
$R = 10^7$

order of genome order of read length order of # of reads

$\# \ of \ ops \approx N \times M \times R \times C = 10^{19}$

$\text{ops/sec} \approx 3 \times 10^9 \ (3\text{GHz CPU})$

$\# \ ops \ / \ (\text{ops/sec}) = \text{secs} \approx 10^{19} / (3 \times 10^9) = (1/3) \times 10^{10}$

$\sim 106 \text{ Years! (for a relatively small 10M read dataset)}$
Goal: Given a set of patterns and a text, find all occurrences of any of patterns in text

Input: k patterns $p^1, \ldots, p^k$, and text $t = t_1 \ldots t_m$

Output: Positions $1 \leq i \leq m$ where substring of $t$ starting at $i$ matches $p_j$ for $1 \leq j \leq k$

Motivation: Searching database for known multiple patterns
Multiple Pattern Matching

- **Solution:** k “pattern matching problems”: O(kmn)

- **Another Solution:**
  - Using “Keyword trees” => O(kn+nm) where n is maximum length of $p^i$
  - Preprocess all k patterns to construct a “keyword tree”
  - Now, any given text, all occurrences of all patterns can be found in time O(m)
Keyword tree approach

- **Keyword tree:**
  - Apple
  - Apropos
  - Banana
  - Bandana
  - Orange
Keyword tree approach: Properties

- Stores a set of keywords in a rooted labeled tree
- Each edge labeled with a letter from an alphabet
- Any two edges coming out of the same vertex have distinct labels
- Every keyword stored can be spelled on a path from root to some leaf
**Construction for** \( \mathcal{P} = \{P_1, \ldots, P_k\} \):

Begin with a root node only;
Insert each pattern \( P_i \), one after the other, as follows:
Starting at the root, follow the path labeled by chars of \( P_i \);

- If the path ends before \( P_i \), continue it by adding new edges and nodes for the remaining characters of \( P_i \);
- Store identifier \( i \) of \( P_i \) at the terminal node of the path.

This takes clearly \( O(|P_1| + \cdots + |P_k|) \)
A keyword tree for $\mathcal{P} = \{\text{he, she, his, hers}\}$:
Keyword tree: Lookup of a string

Lookup of a string $P$: Starting at root, follow the path labeled by characters of $P$ as long as possible;

1. If the path leads to a node with an identifier, $P$ is a keyword in the dictionary
2. If the path terminates before $P$, the string is not in the dictionary

How to check all occurrences in a text $t$?
Keyword tree approach: Complexity

• Build keyword tree in $O(kn)$ time; $kn$ is total length of all patterns

• Start “threading” at each position in text; at most $n$ steps tell us if there is a match here to any $p_i$

• $O(kn + nm)$
  • We’re down from $O(kmn)$ to this

• The next big idea, Aho-Corasick algorithm: $O(kn + m)$
Aho-Corasick algorithm: Key idea

Exploit the redundancy in the patterns

HERSHE

HERS

SHE

HE
Aho-Corasick algorithm: Key idea

Exploit the redundancy in the patterns
Aho-Corasick algorithm

With failing edges and node labels
Rules

- Transition among the different nodes by following edges depending on next character seen (say “h”)
- If outgoing edge with label “h”, follow it (e.g. 0->1, 3->4)
- If no such edge, and are at root, stay
- If no such edge, and at non-root, follow dashes edge (“fail” transition). However, DO NOT CONSUME THE CHARACTER (e.g. 9->3)

Consider text “hershe”
Aho-Corasick algorithm

A keyword tree for $\mathcal{P} = \{\text{he, she, his, hers}\}$:
Aho-Corasick algorithm

Add pattern labels

\[ \neq \{h, s\} \]

\[ \text{h, e, r, s} \]

\[ \{he\} \quad \{his\} \quad \{hers\} \quad \{she\} \]
Adding failing edges

- If currently at node q representing word L(q), find the longest proper suffix of L(q) that is a prefix of some pattern, and go to the node representing that prefix. Insert the labels of the pointed node (if there is any) to node q’s set of labels.

- Example: node q = 5, L(q) = she; longest proper suffix that is a prefix of some pattern: “he”. Dashed edge to node q’=2
Aho-Corasick Algorithm

Add Failing Edges and Labels