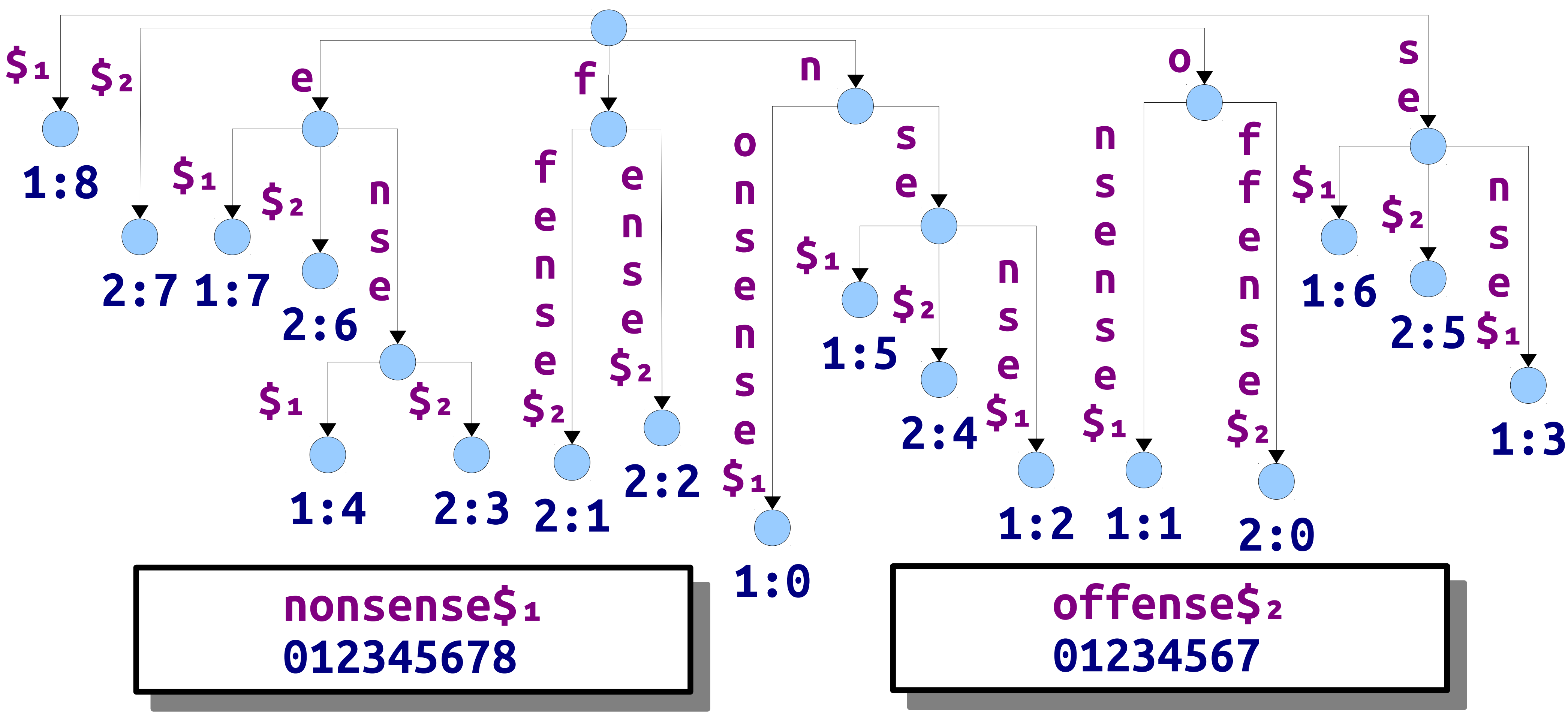
Generalized Suffix Trees

Suffix Trees for Multiple Strings

- Suffix trees store information about a single string and exports a huge amount of structural information about that string.
- However, many applications require information about the structure of multiple different strings.

Generalized Suffix Trees

- A **generalized suffix tree** for T_1, …, T_k is a Patricia trie of all suffixes of $T_1\$_1$, …, $T_k\$_k$. Each T_i has a unique end marker.
- Leaves are tagged with ***i*:*j***, meaning "jth suffix of string T_i"



Generalized Suffix Trees

- **Claim:** A generalized suffix tree for strings T_1, …, T_k of total length m can be constructed in time $\Theta(m)$.
- Use a two-phase algorithm:
  - Construct a suffix tree for the single string $T_1\$_1T_2\$_2 \ldots T_k\$_k$ in time $\Theta(m)$.
    - This will end up with some invalid suffixes.
  - Do a DFS over the suffix tree and prune the invalid suffixes.
    - Runs in time $O(m)$ if implemented intelligently.

Applications of Generalized Suffix Trees

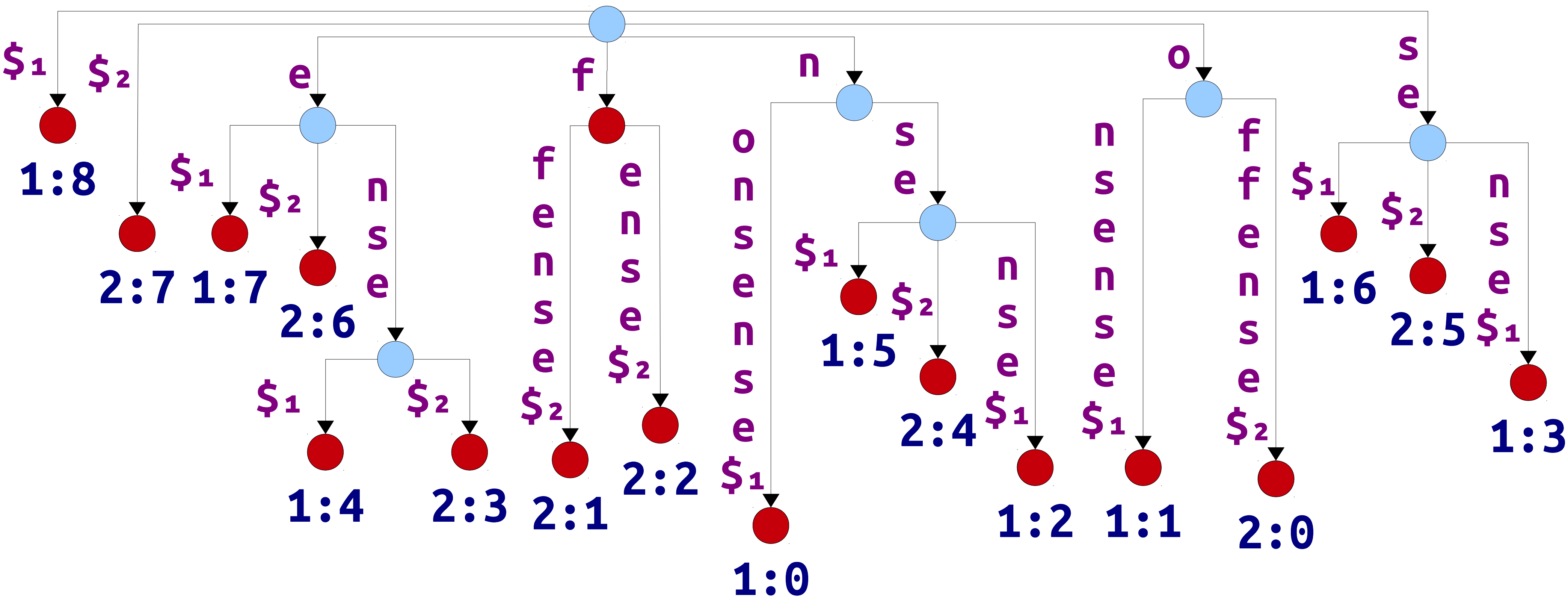
Longest Common Substring

- Consider the following problem:

  Given two strings T_1 and T_2, find the longest string w that is a substring of both T_1 and T_2.

- Can solve in time $O(|T_1| \cdot |T_2|)$ using dynamic programming.
- Can we do better?

Longest Common Substring



nonsense$\$_1$
012345678

offense$\$_2$
01234567

Longest Common Substring

- Build a generalized suffix tree for T_1 and T_2 in time $O(m)$.
- Annotate each internal node in the tree with whether that node has at least one leaf node from each of T_1 and T_2.
  - Takes time $O(m)$ using DFS.
- Run a DFS over the tree to find the marked node with the highest string depth.
  - Takes time $O(m)$ using DFS
- Overall time: $\mathbf{O(}\boldsymbol{m}\mathbf{)}$.