Assignment-1

Dictionary Data Structure: Trie

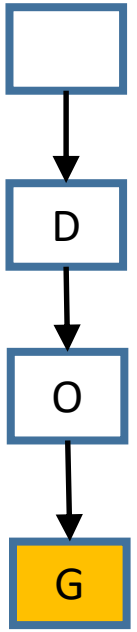
Trie

- A tree-like data structure for storing words efficiently
  - Insertion
  - Search
  - Sort

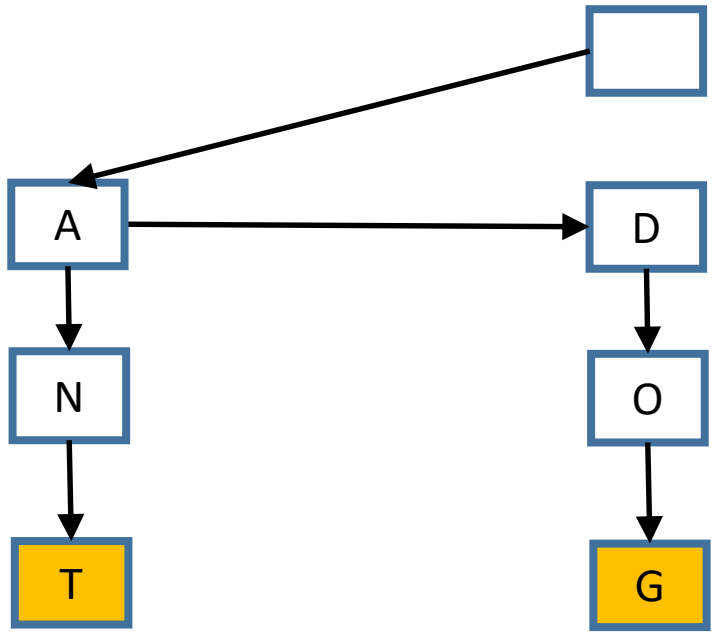
```
typedef struct trieNode {
        char ch;
        struct trienode *sibling;
        struct trienode *child;
        int isWord;
} trieNode;
```

Insertion into Trie

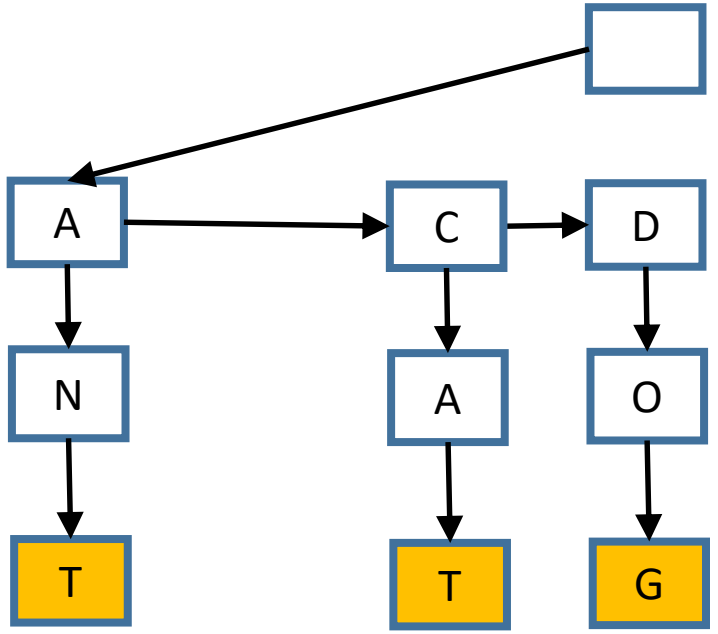
Insertion: DOG



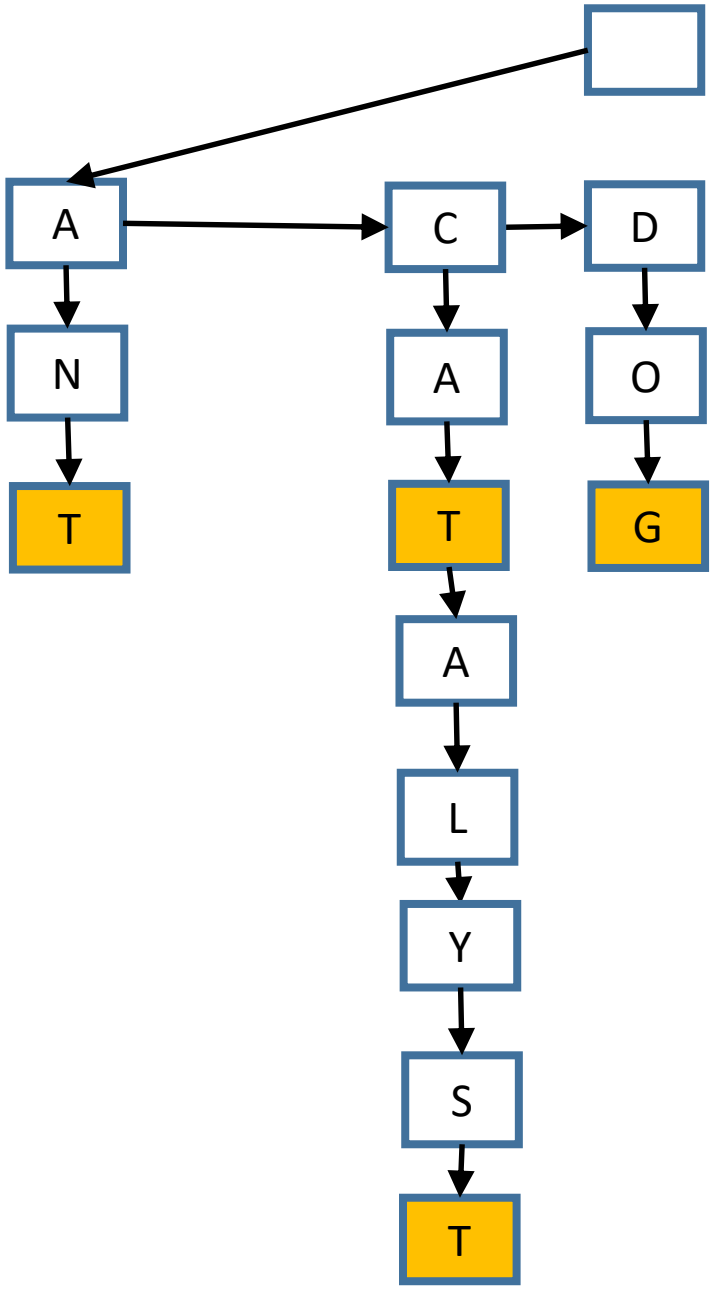
Insertion: ANT



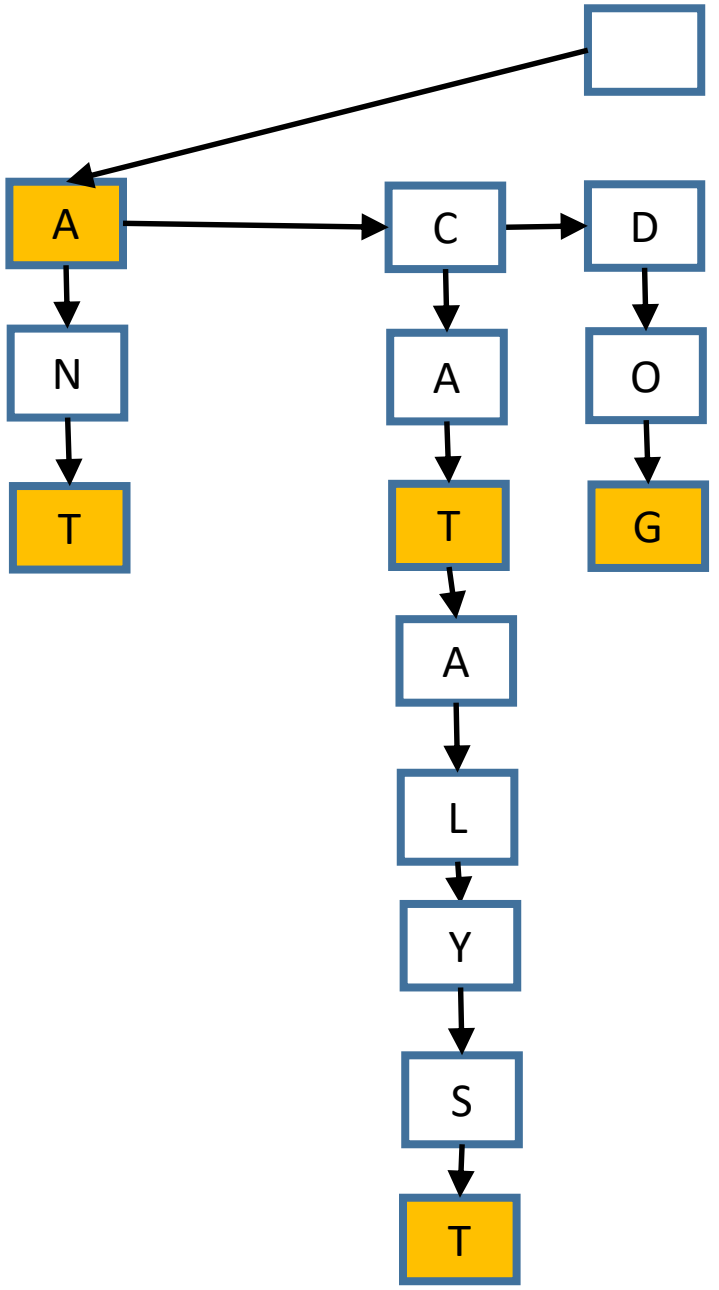
Insertion: CAT



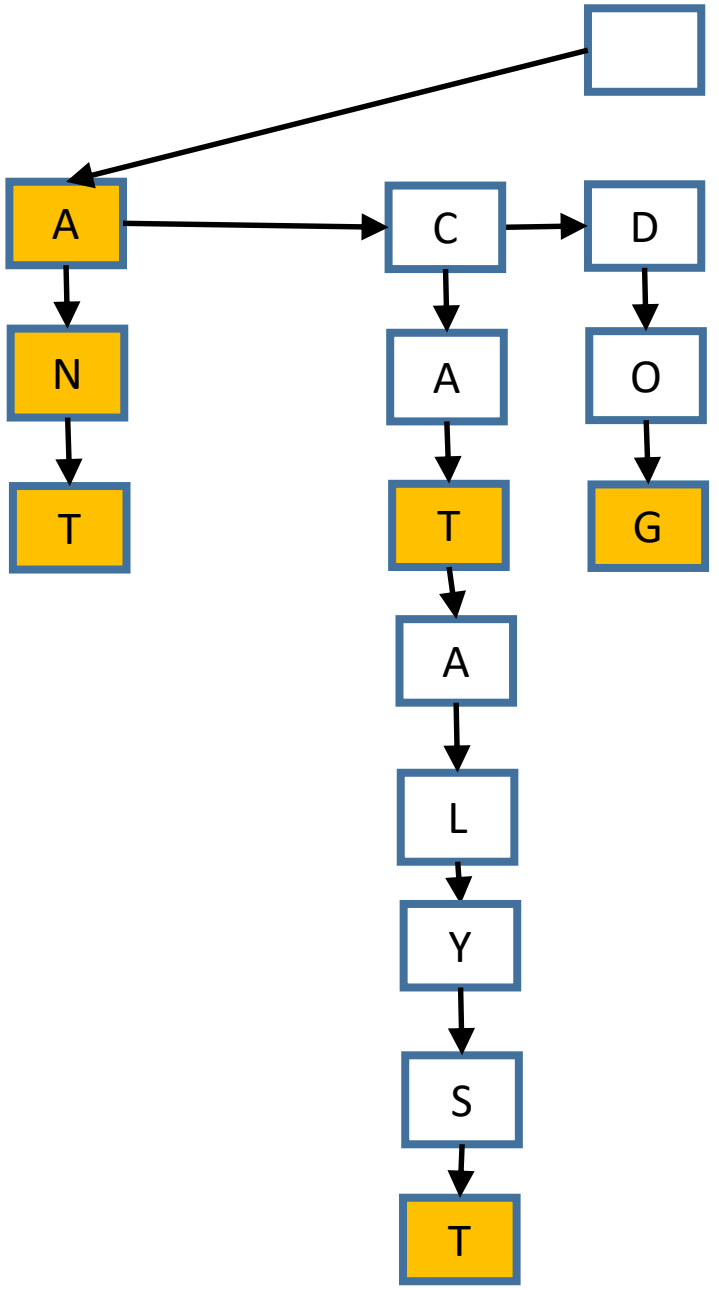
Insertion: CATALYST



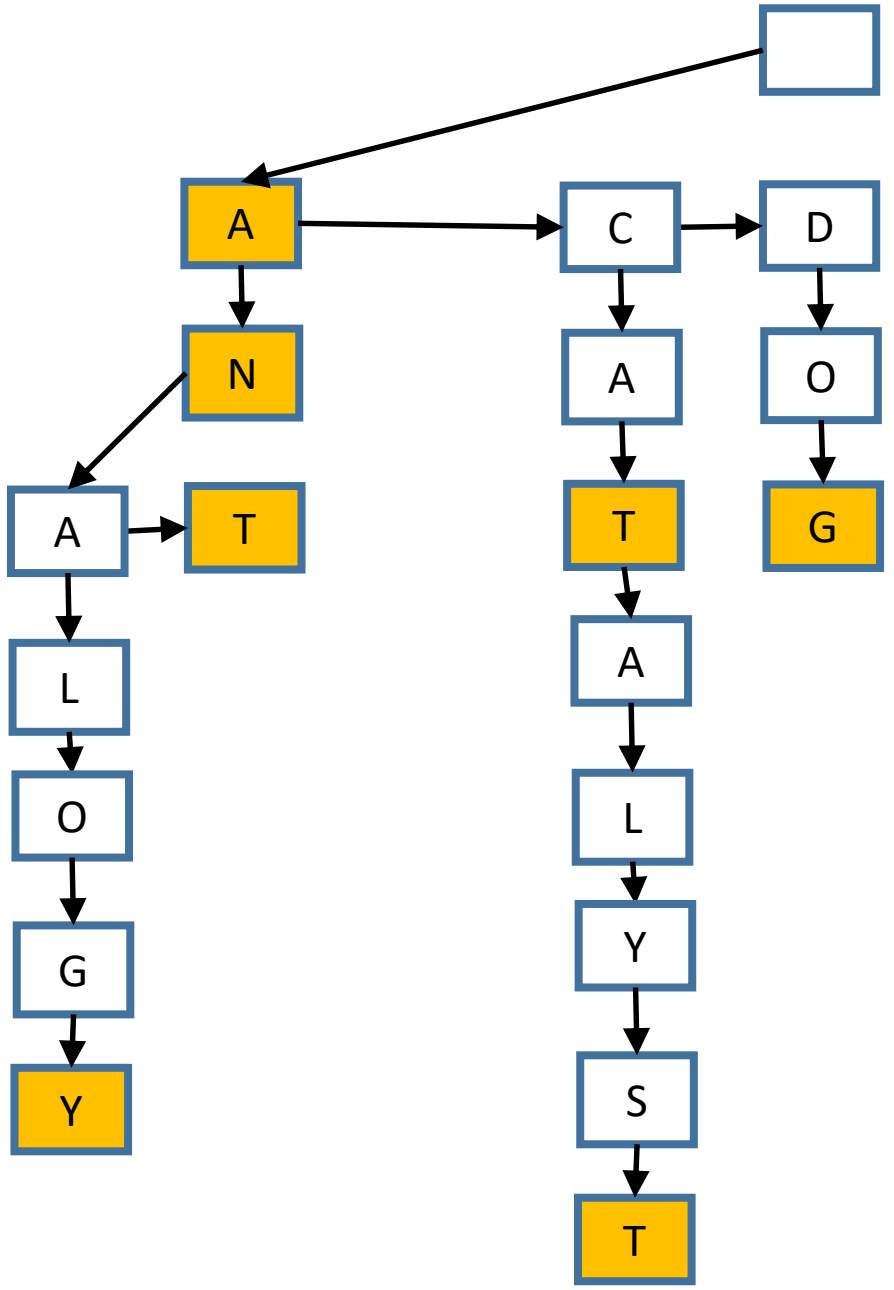
Insertion: A



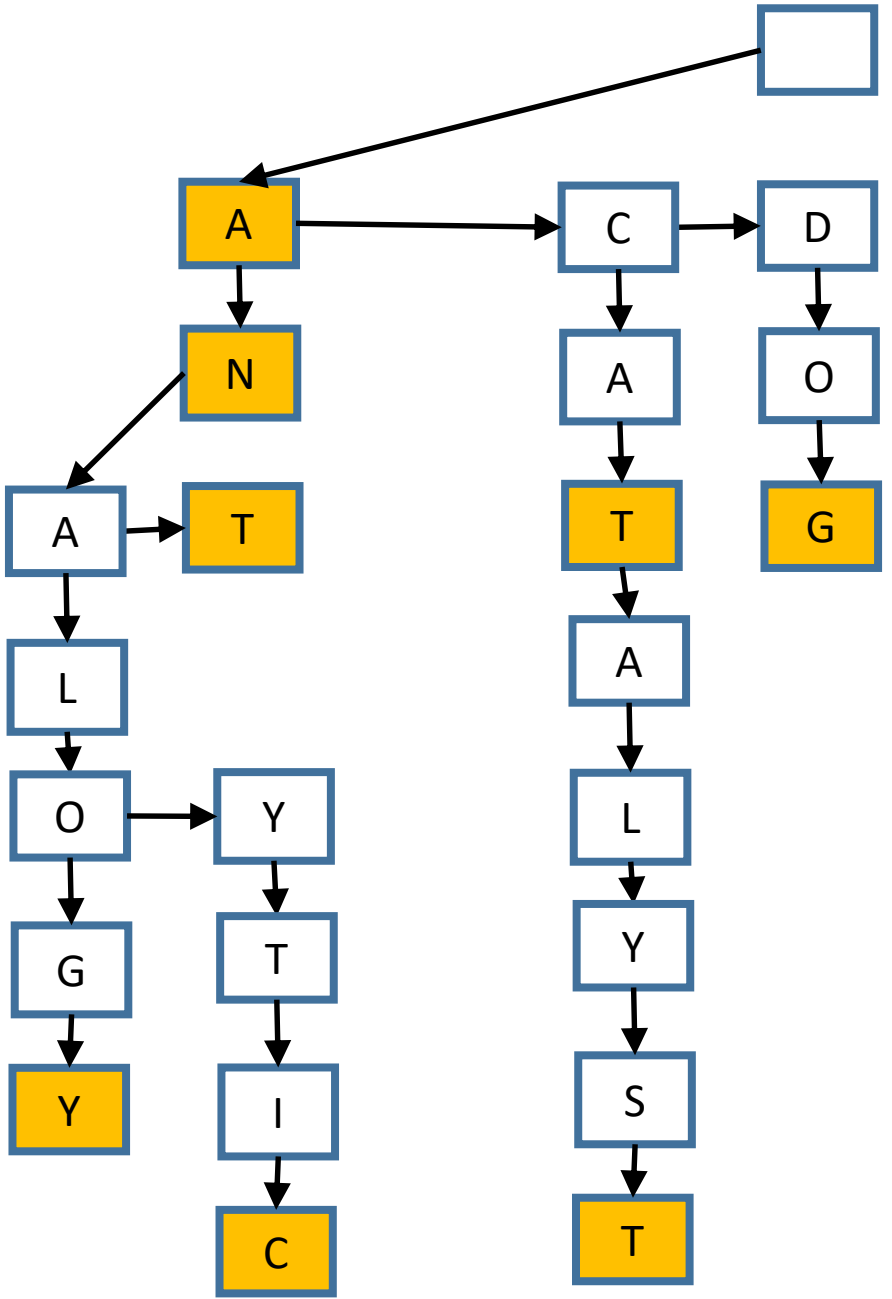
Insertion: AN



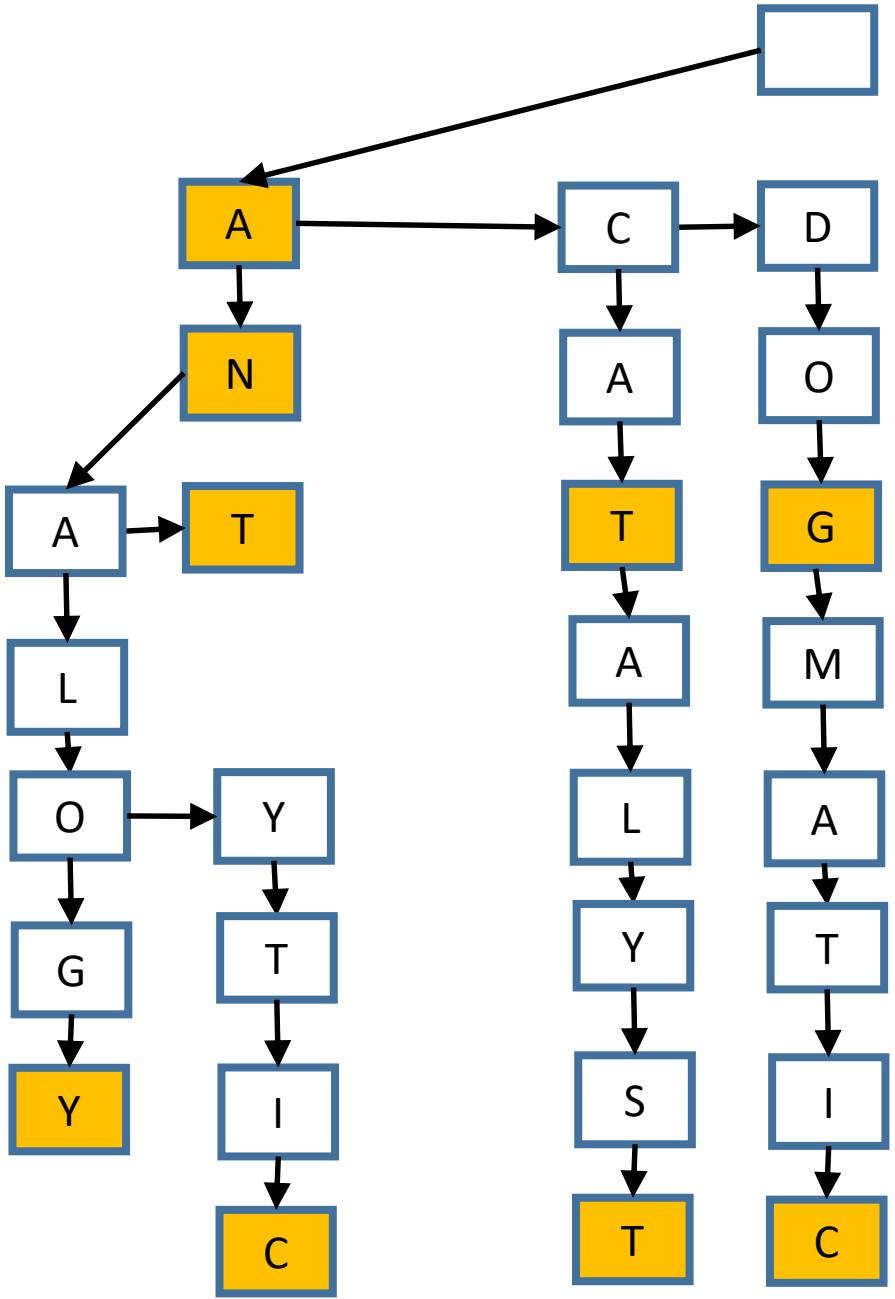
Insertion: ANALOGY



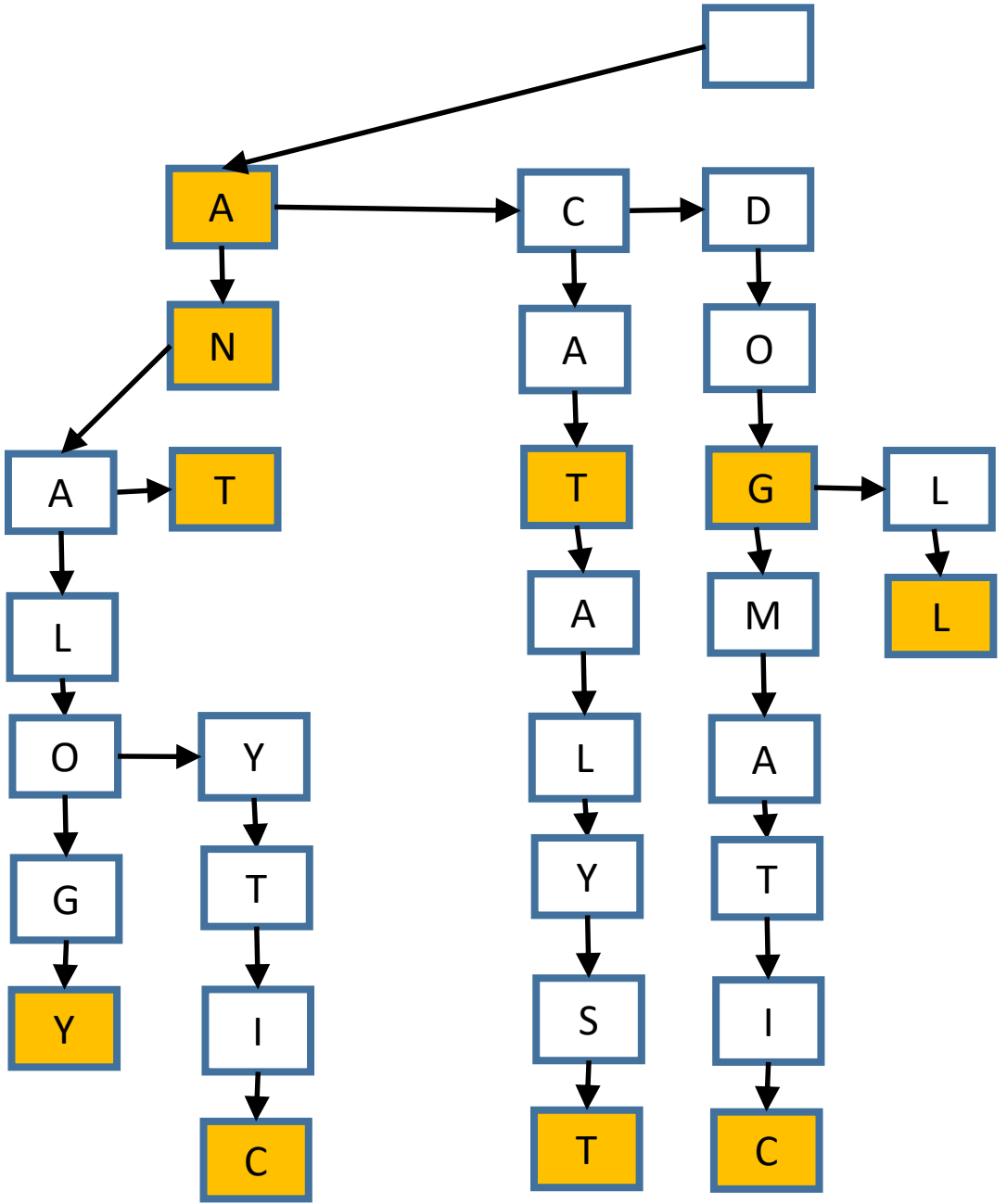
Insertion: ANALYTIC



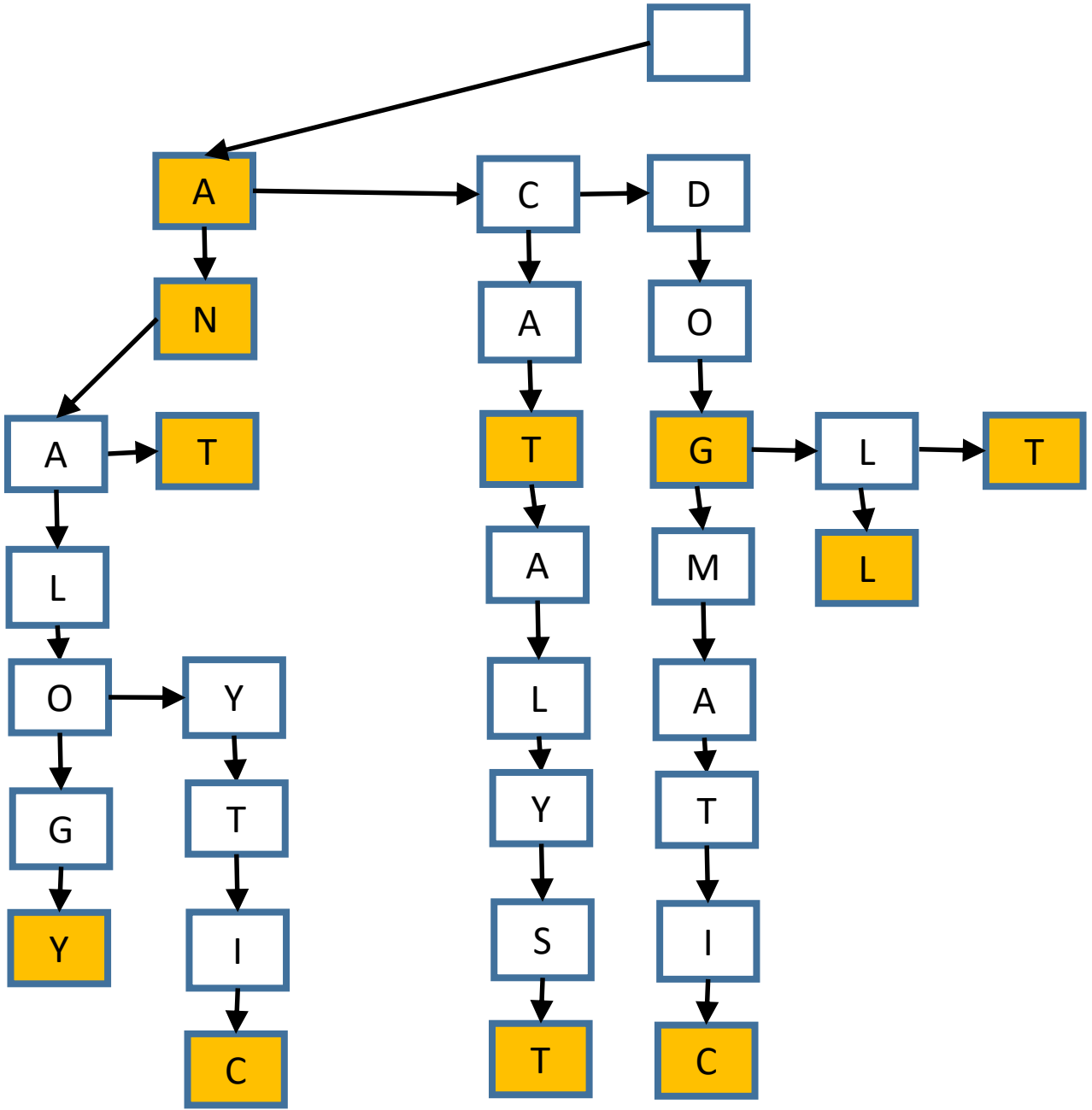
Insertion: DOGMATIC



Insertion: DOLL

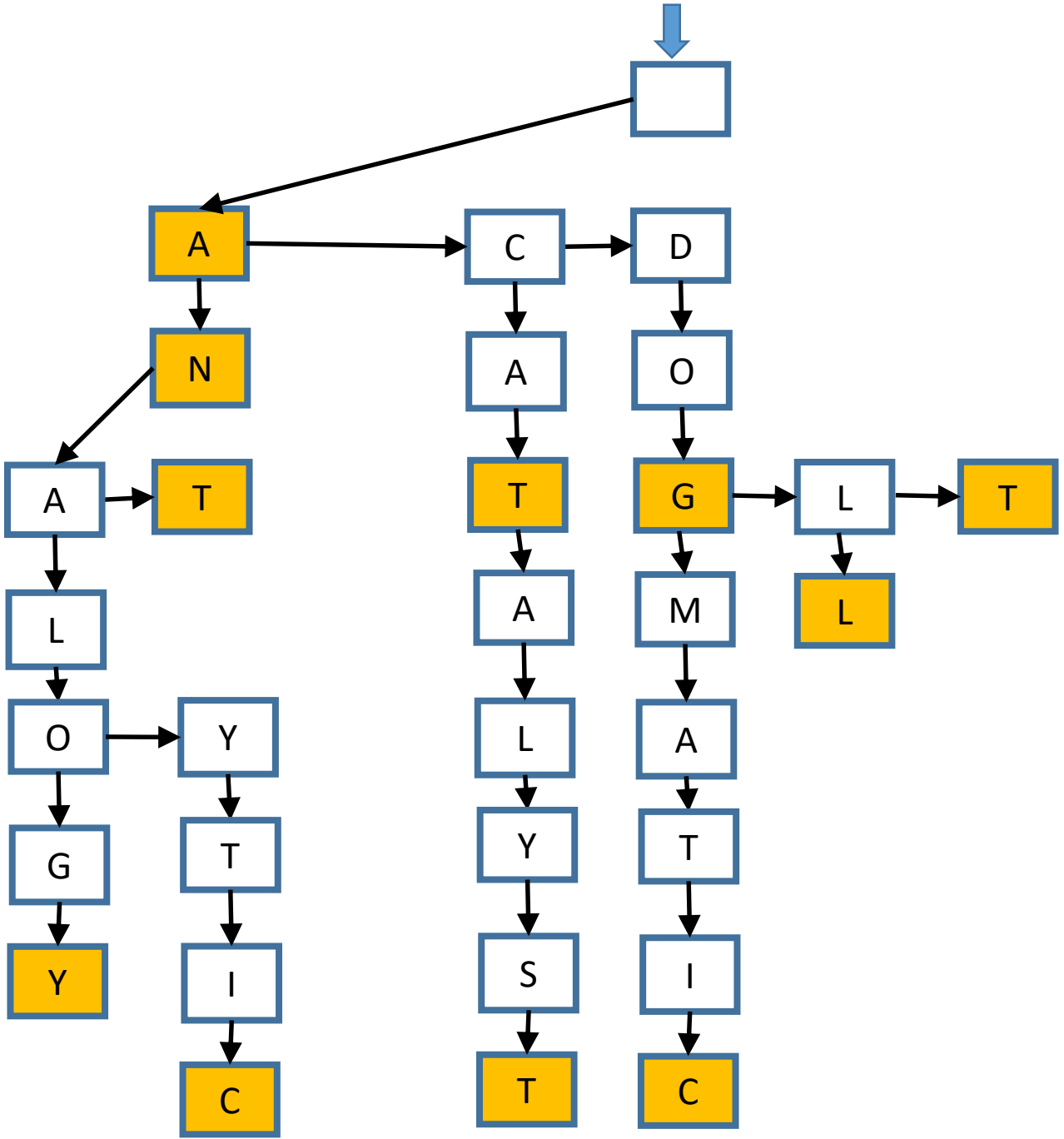


Insertion: DOT

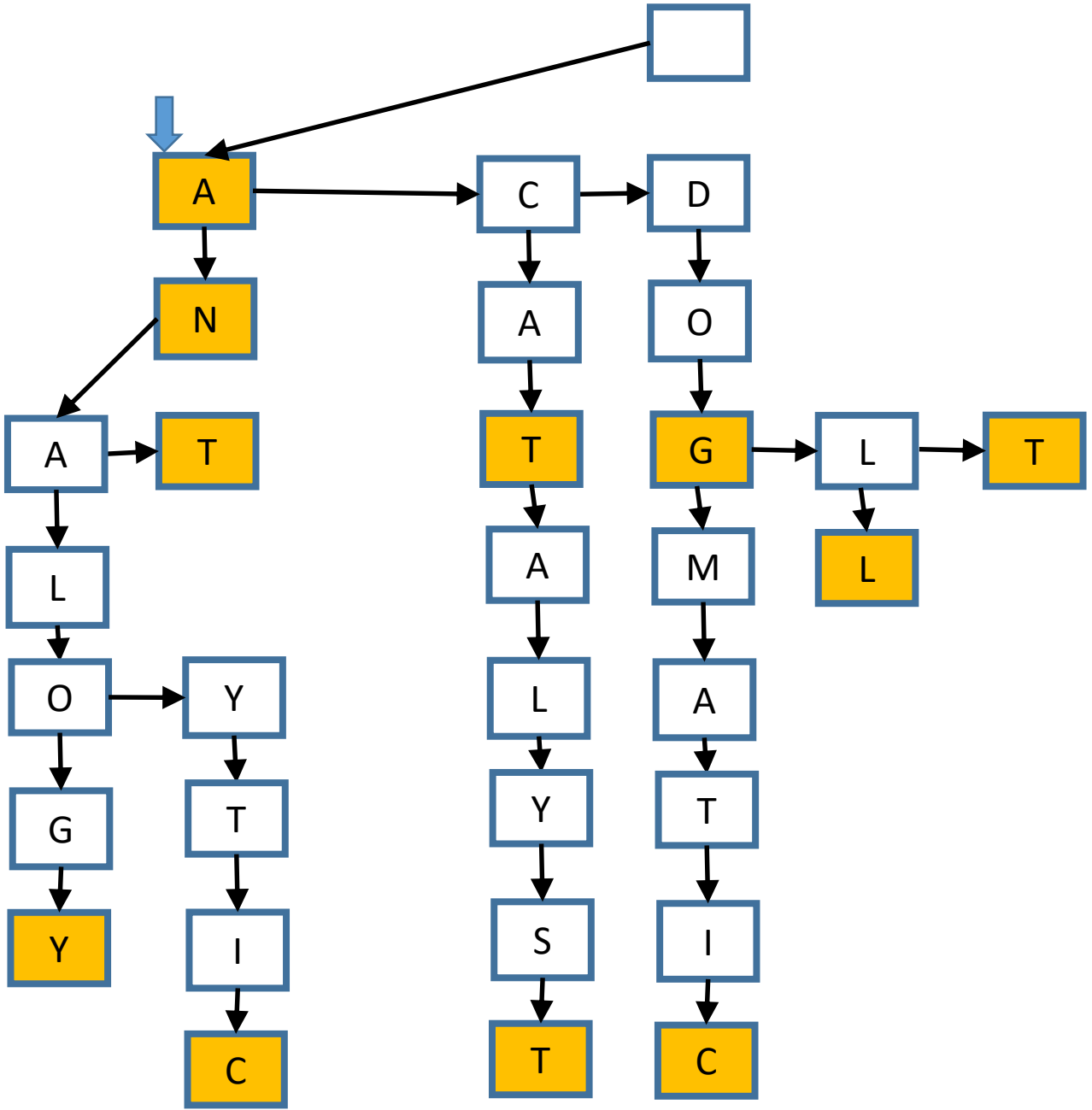


Search in a Trie

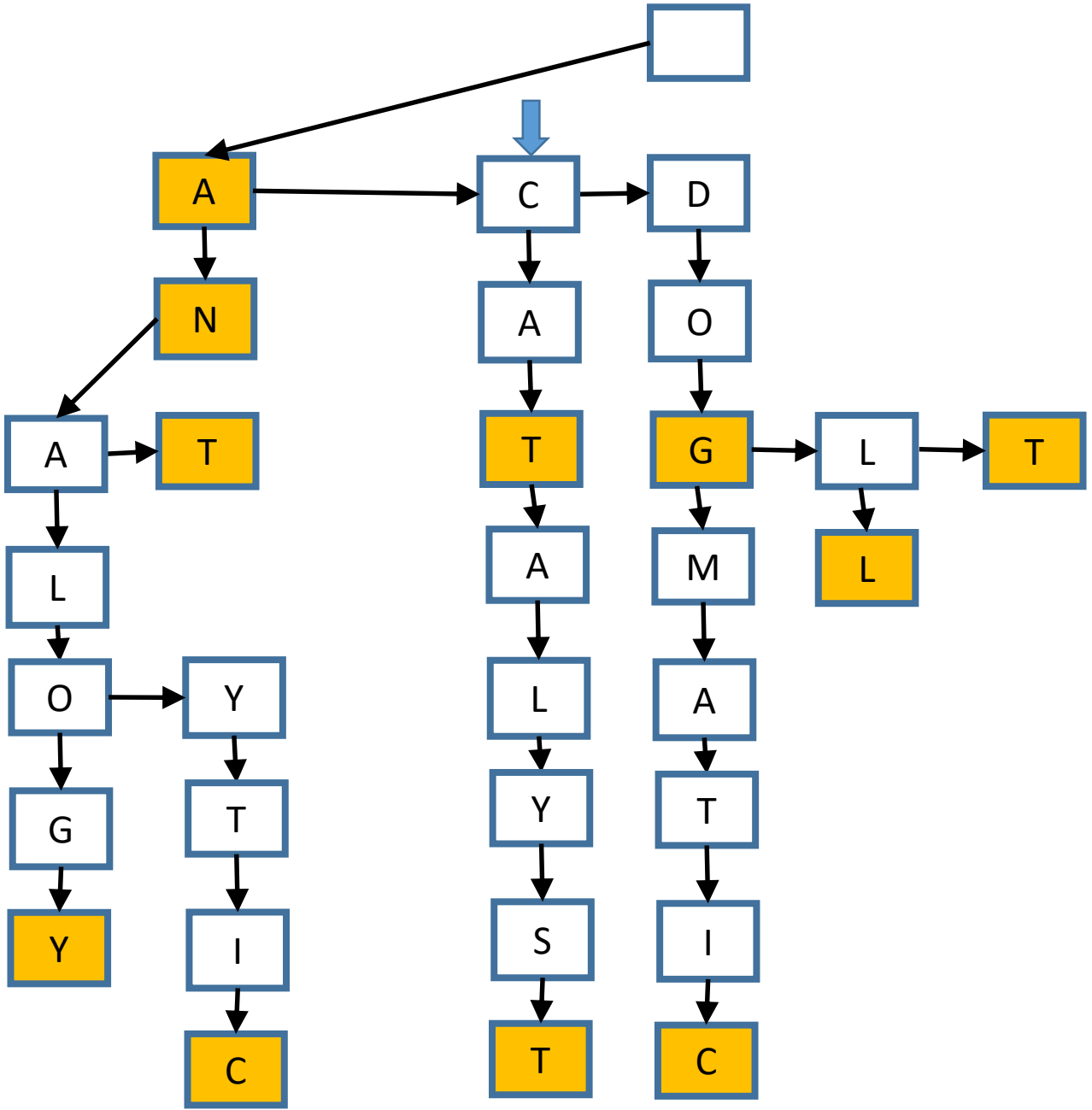
Search: DOT



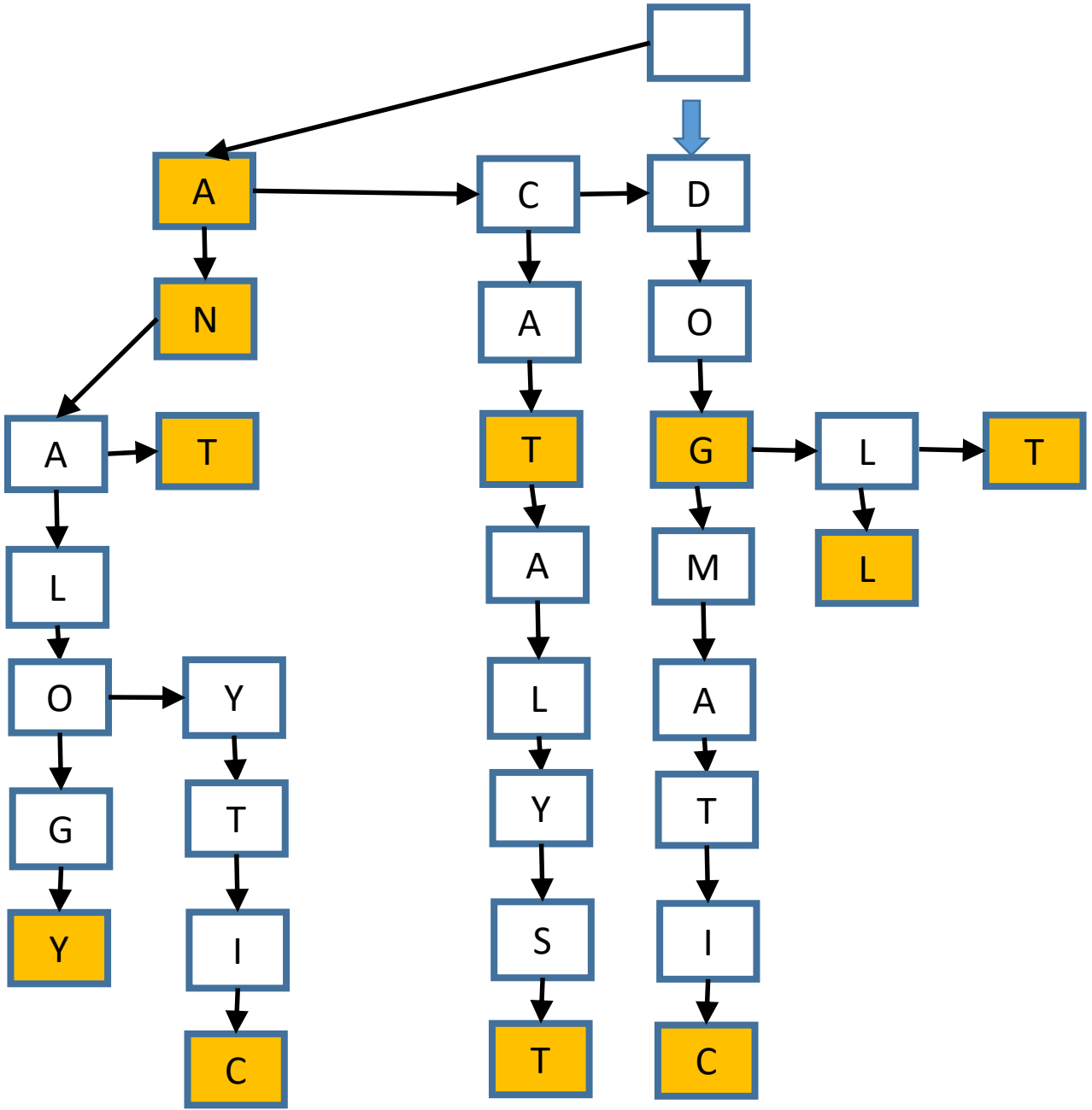
Search: DOT



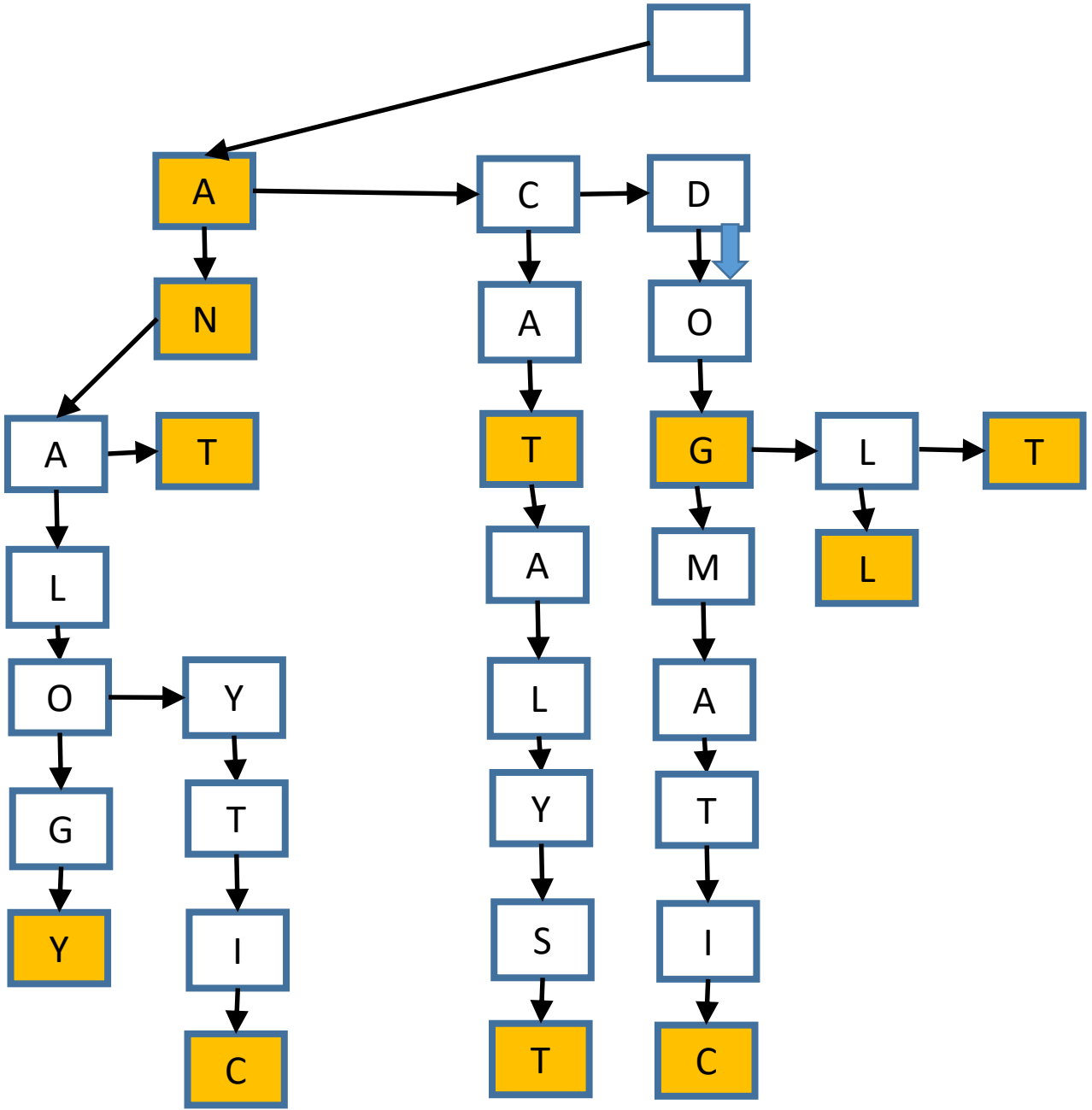
Search: DOT



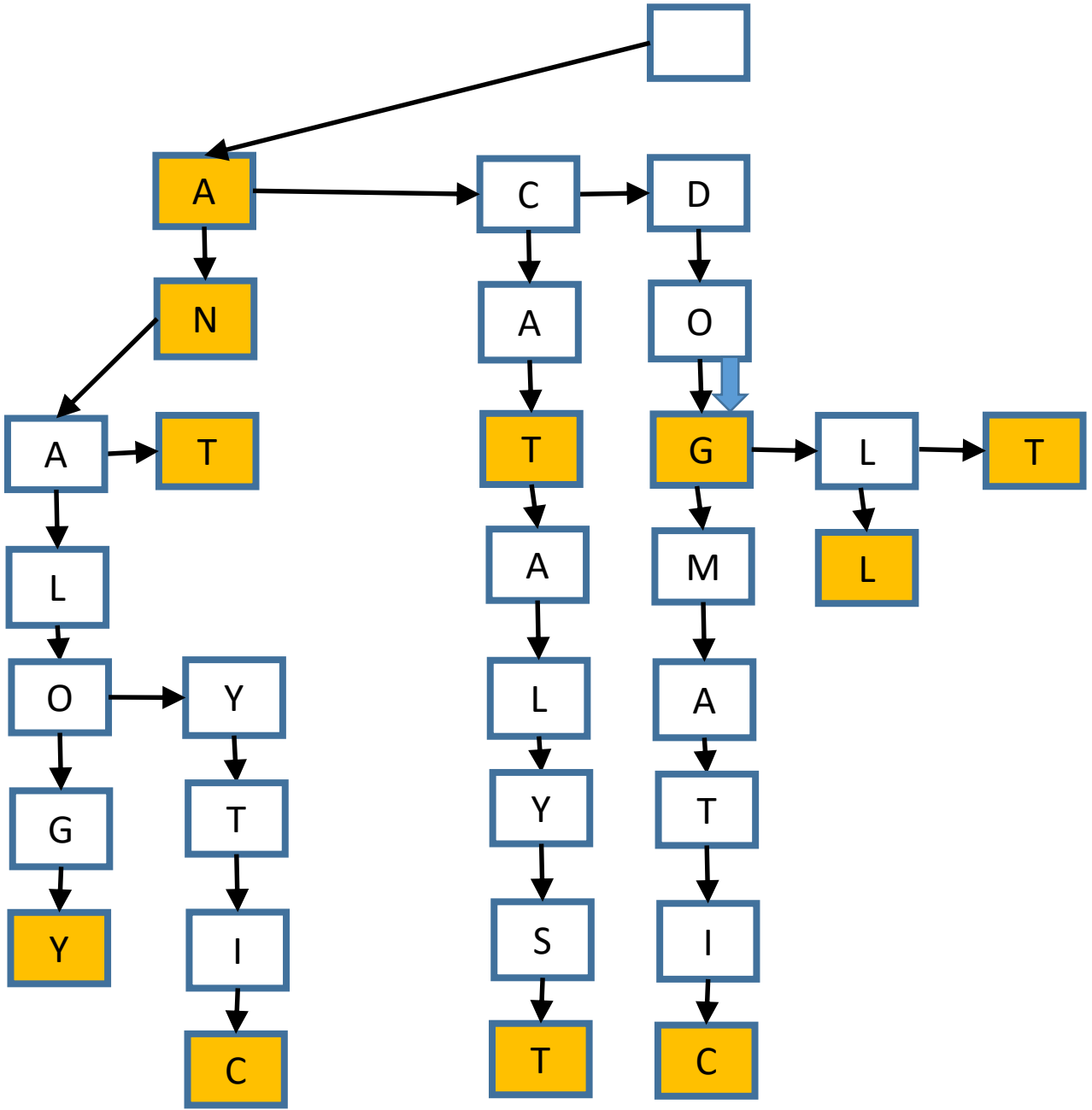
Search: DOT



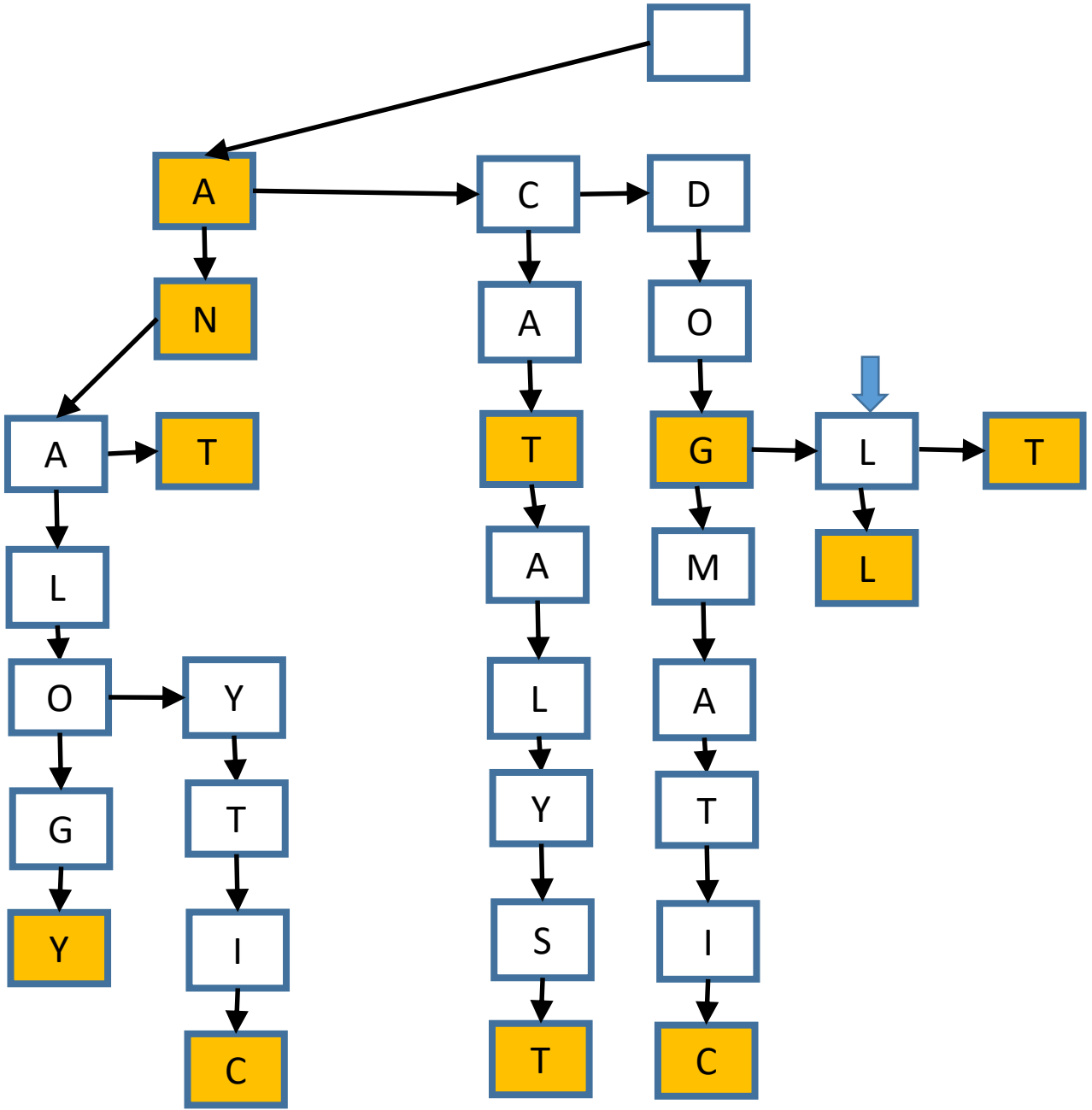
Search: DOT



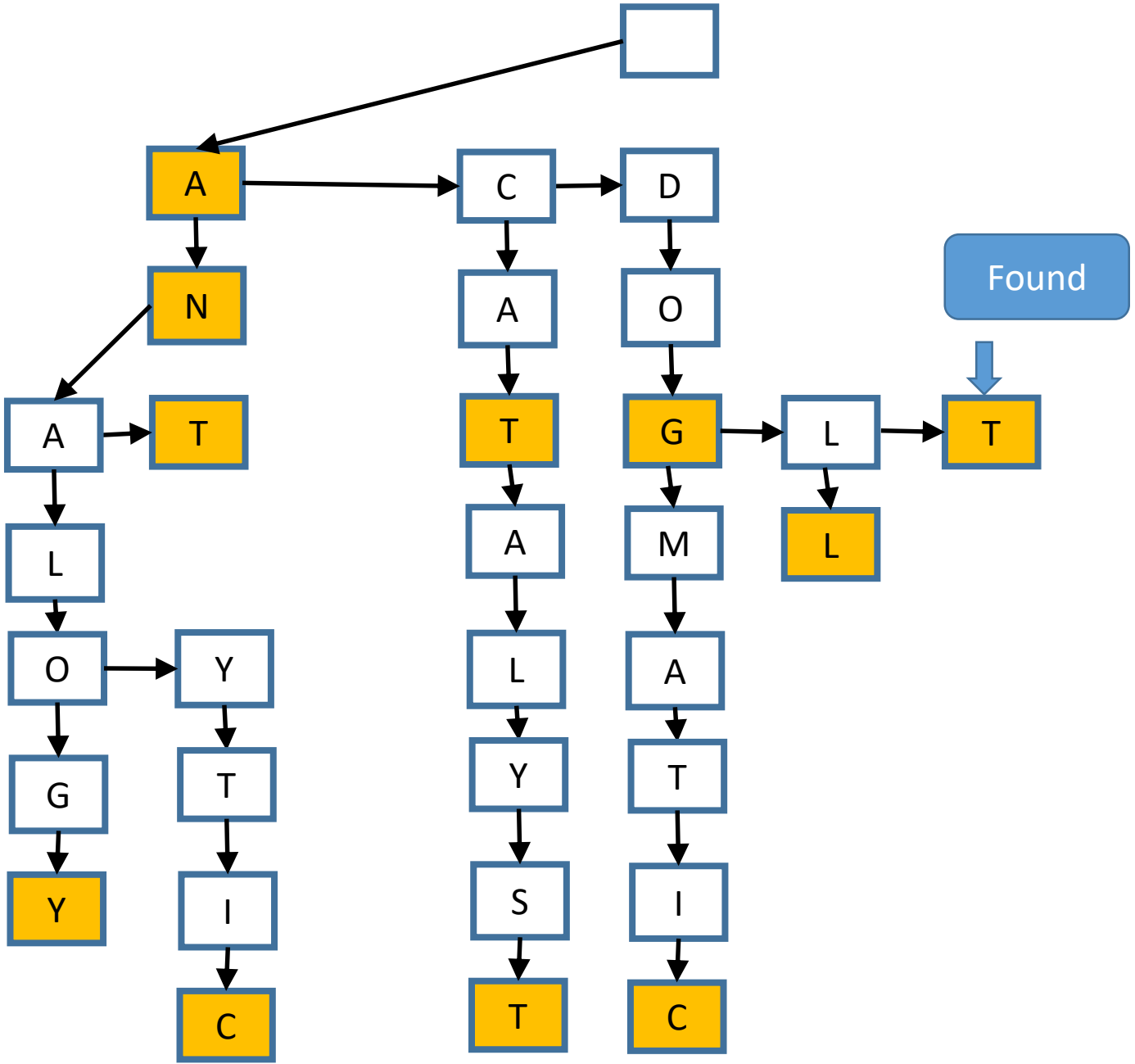
Search: DOT



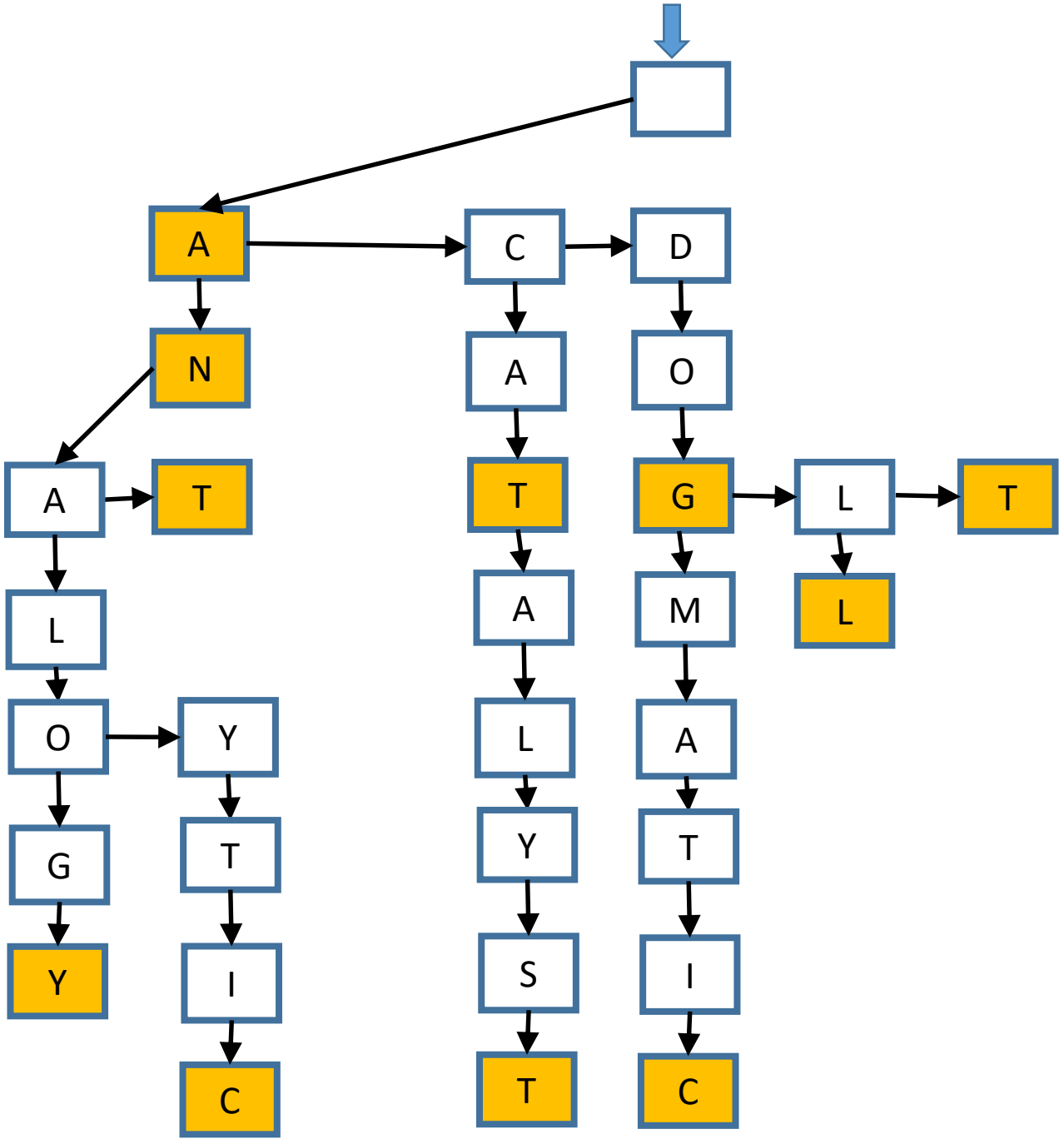
Search: DOT



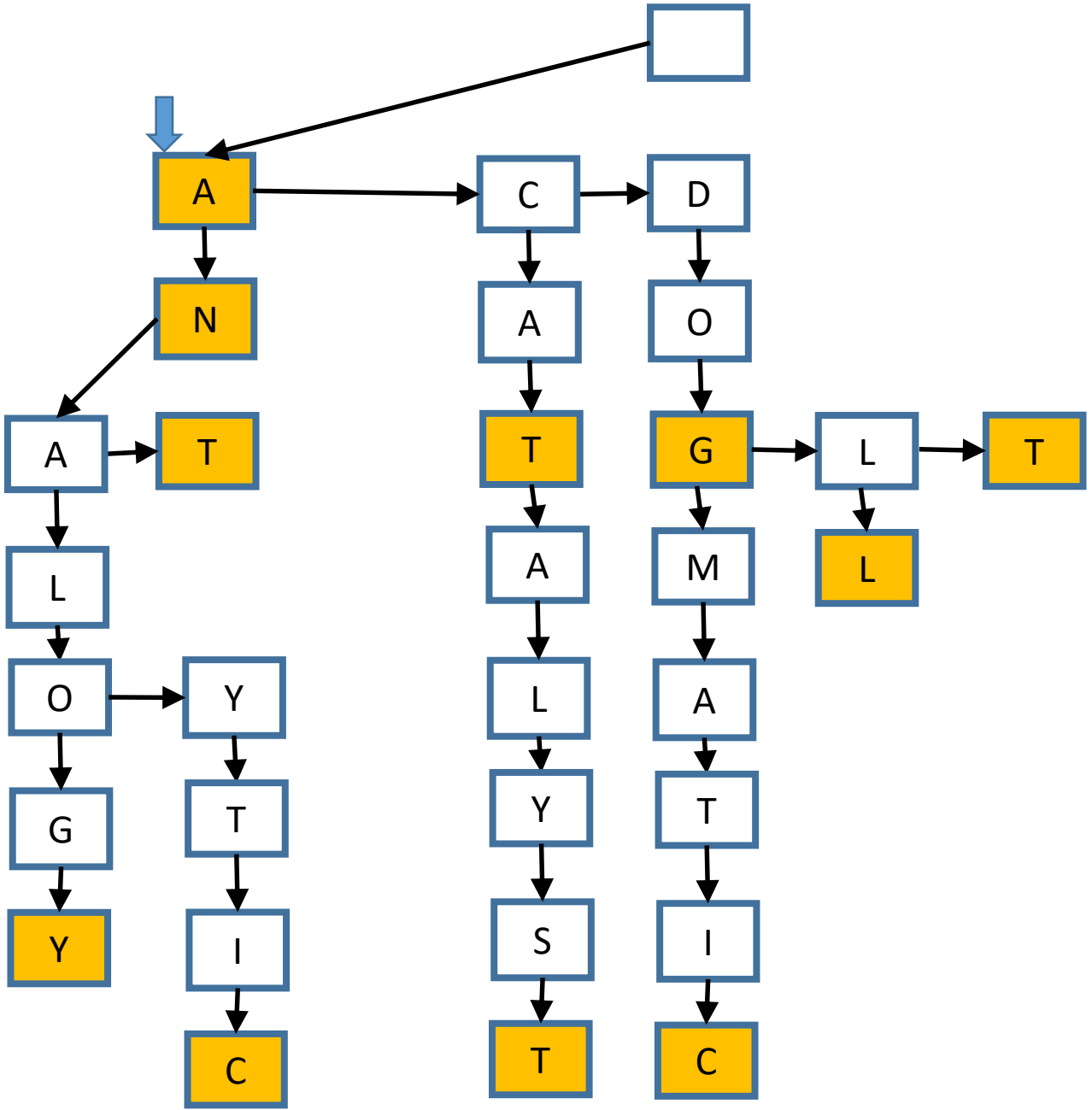
Search: DOT



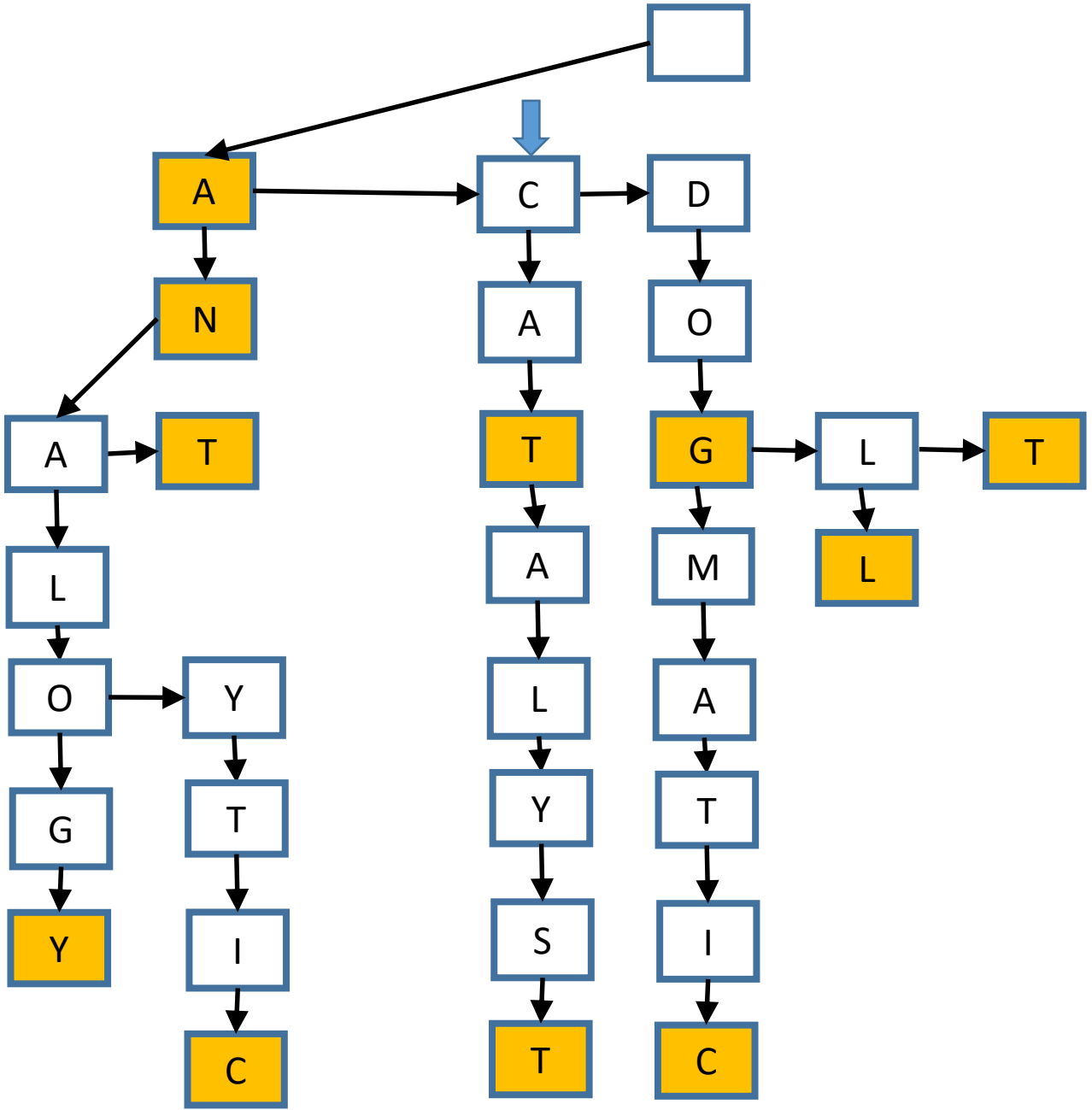
Search: DO



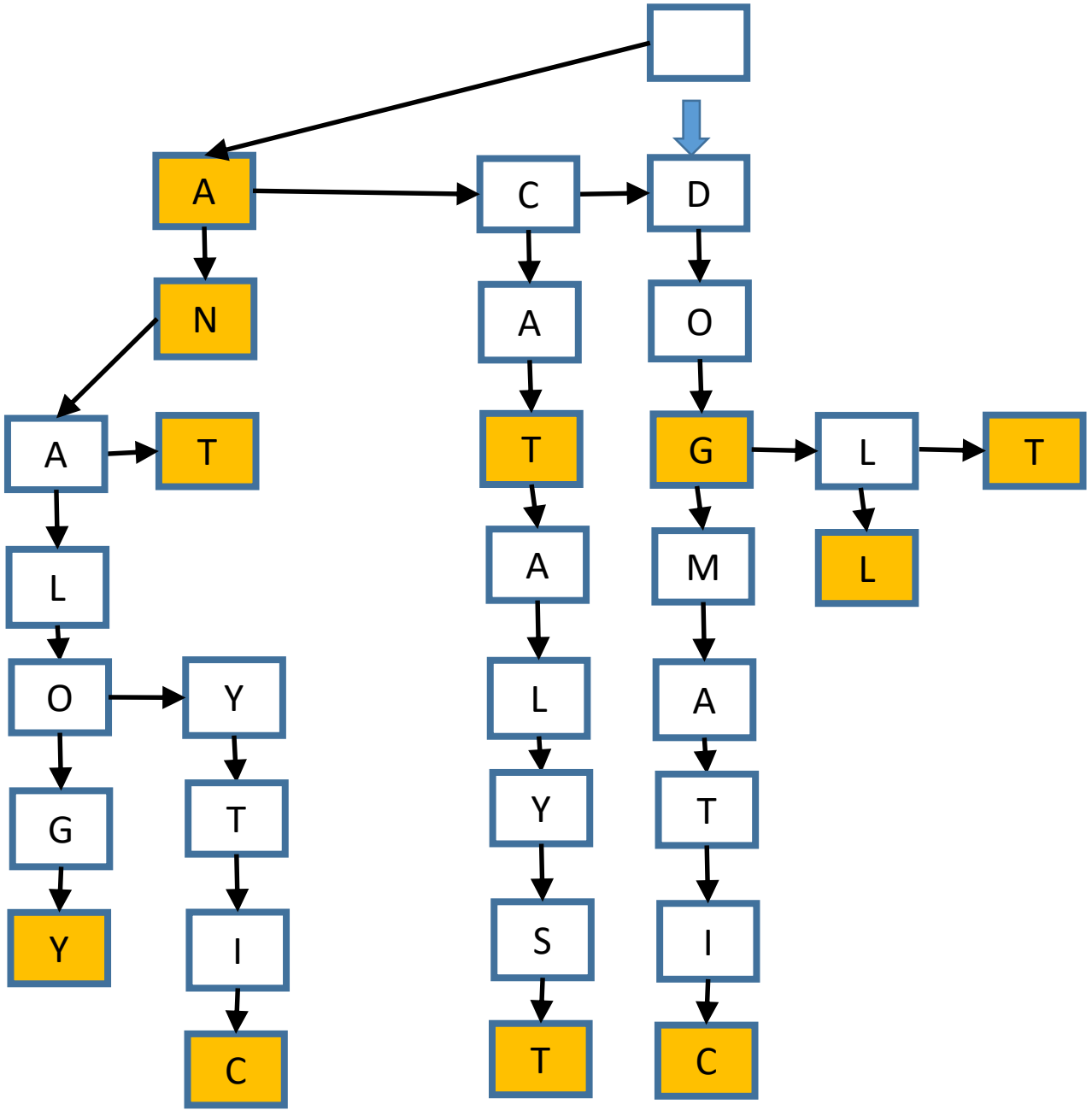
Search: DO



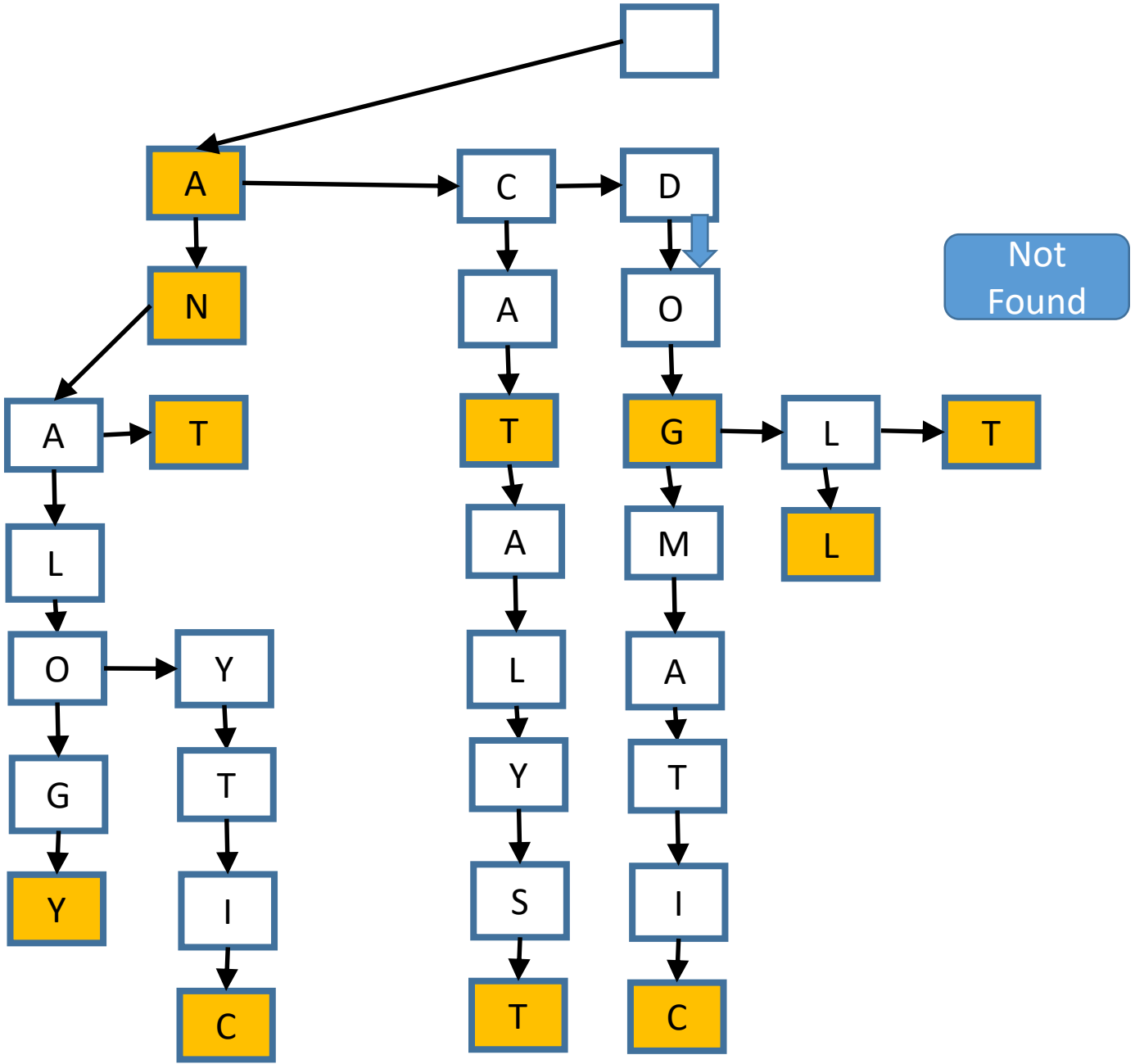
Search: DO



Search: DO



Search: DO



Print the words in the Lexicographic order

Sort

"words.txt"

A

AN

ANALOGY

ANALYTIC

ANT

CAT

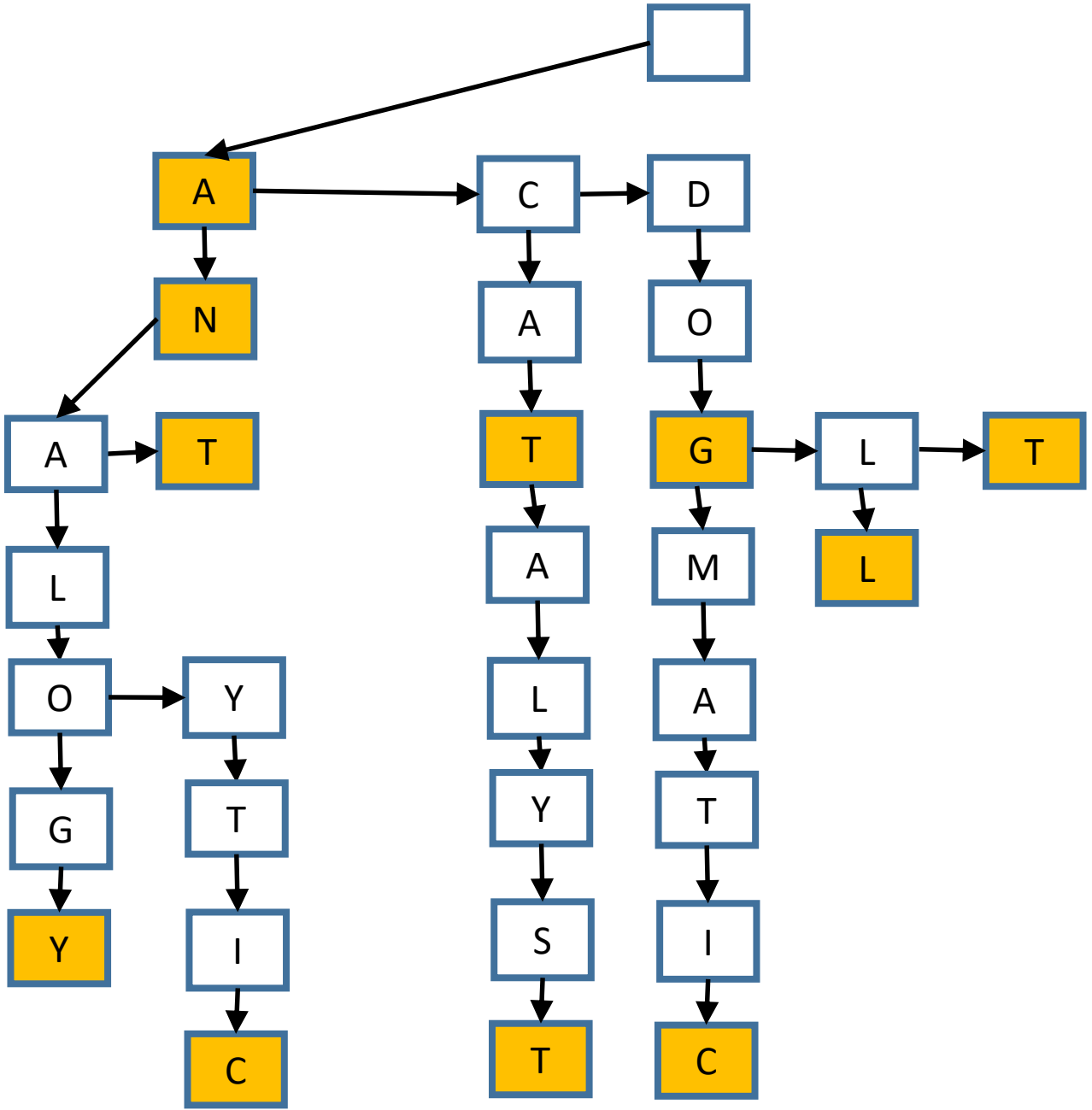
CATALYST

DOG

DOGMATIC

DOLL

DOT



Program Structure

1. Write a main() function that reads a file “words.txt” and populate the trie. After that there is no file read/write operations. Therefore, close the file.

2. There is a loop asking for four options:

- ‘i’ – InsertIntoTrie (root, “word”)
- ‘f’ – SearchInTrie(root, “word”)
- ‘s’ - Sort
- ‘q’ – Quit

Based on the options, and the input “word” (from console), the program acts accordingly.

Submission

- Last date: 11-AUG-2024 (till 11:59 P.M.) (Sunday)
- Programming language: C/C++
- Single File: 24CS06001_A1.c/.cpp or 24AI06001_A1.c/.cpp
- Subject Line: 24CS06001_A1 or 24AI06001_A1
- Email to: pds2016autumn@gmail.com