String Searching

Suppose we are given a sequence of nucleotides, such as ACGCGCAGGCA, and we wish to find all occurrences of that string in the human genome, about 3,000,000,000 bases long.

How long would that take?

- Without pre-processing, it would take at least 3,000,000,000 steps, because we need at least to look at every entry in the human genome.

- With pre-processing, perhaps we can set up some sort of indexing system to more rapidly search for these strings.

Indeed we can! The resulting search, after pre-processing, would take just 11 steps.

Pre-Processing???

The idea is that a popular string such as the human genome will be searched many, many times. Some pre-processing up front may be worth it, as every query will enjoy improved performance.

In fact, given:

- Search string S, of size |S|, and
- Query strings Q_1, Q_2, ..., of sizes $|Q_i|$,

We can create a data structure in time proportional to |S| so that each subsequent query takes time $|Q_i|$.

This data structure is called a *suffix tree*.

Sales Pitch

A search of MathSciNet for articles in mathematical journals revealed 9 matches since 2000, plus these in a biology database.

1: Lu B, Chen T.
A suffix tree approach to the interpretation of tandem mass spectra: applications to peptides of non-specific digestion and post-translational modifications.
Bioinformatics. 2003 Oct;19 Suppl 2:II113-II121.

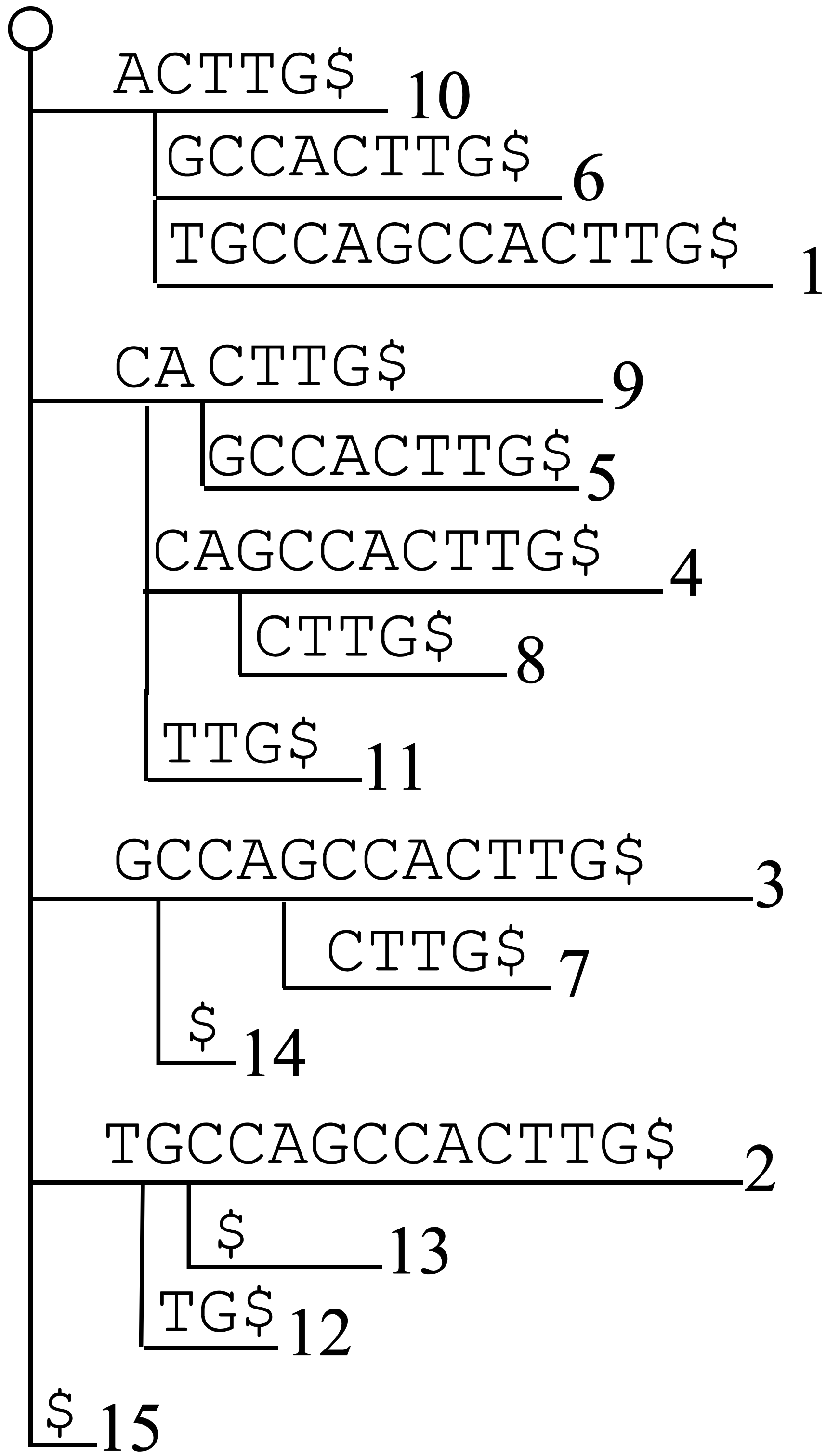
2: Marsan L, Sagot MF.
Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification.
J Comput Biol. 2000;7(3-4):345-62.

3: Bejerano G, Yona G.
Variations on probabilistic suffix trees: statistical modeling and prediction of protein families.
Bioinformatics. 2001 Jan;17(1):23-43.

4: Dorohonceanu B, Nevill-Manning CG.
Accelerating protein classification using suffix trees.
Proc Int Conf Intell Syst Mol Biol. 2000;8:128-33.

A Suffix Tree

The following tree contains all suffixes of the string ATGCCAGCCACTTG



Some Uses for a Suffix Tree

1. Find all occurrences of the string “CCA”
    a. Start at the root
    b. Follow the string “CCA” as far as possible
        i. If we get stuck, the string does not occur
        ii. If we find the whole string, then the integers on all leaves below us tell where “CCA” may be found in the search string

2. Find all occurrences of “TGA”

3. How many times does “T” occur in the string?

4. Many more!

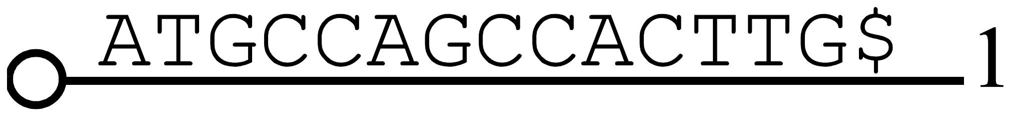
    a. See homework...

How does One Generate a Suffix Tree?

Naively:

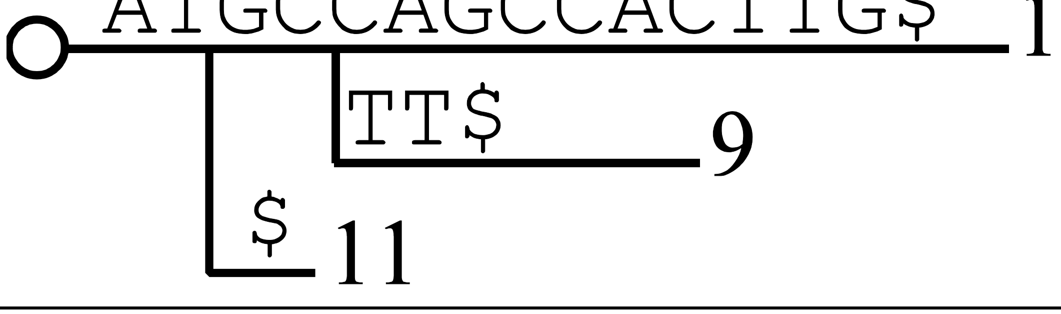
1. Make a list of all suffixes in the search string, with "$" as an end marker

2. Write down the longest one (i.e., the whole string) on a single edge connected to the root



3. Go through the list from longest to shortest, adding the suffixes one at a time. As each string is added, walk down the tree as far as possible, searching for as long a prefix of that suffix as can be found, before adding a new branch to the tree.

E.g., if we add "ATGCTT" to our tree, then "AT":



Suffix Trees are Fast

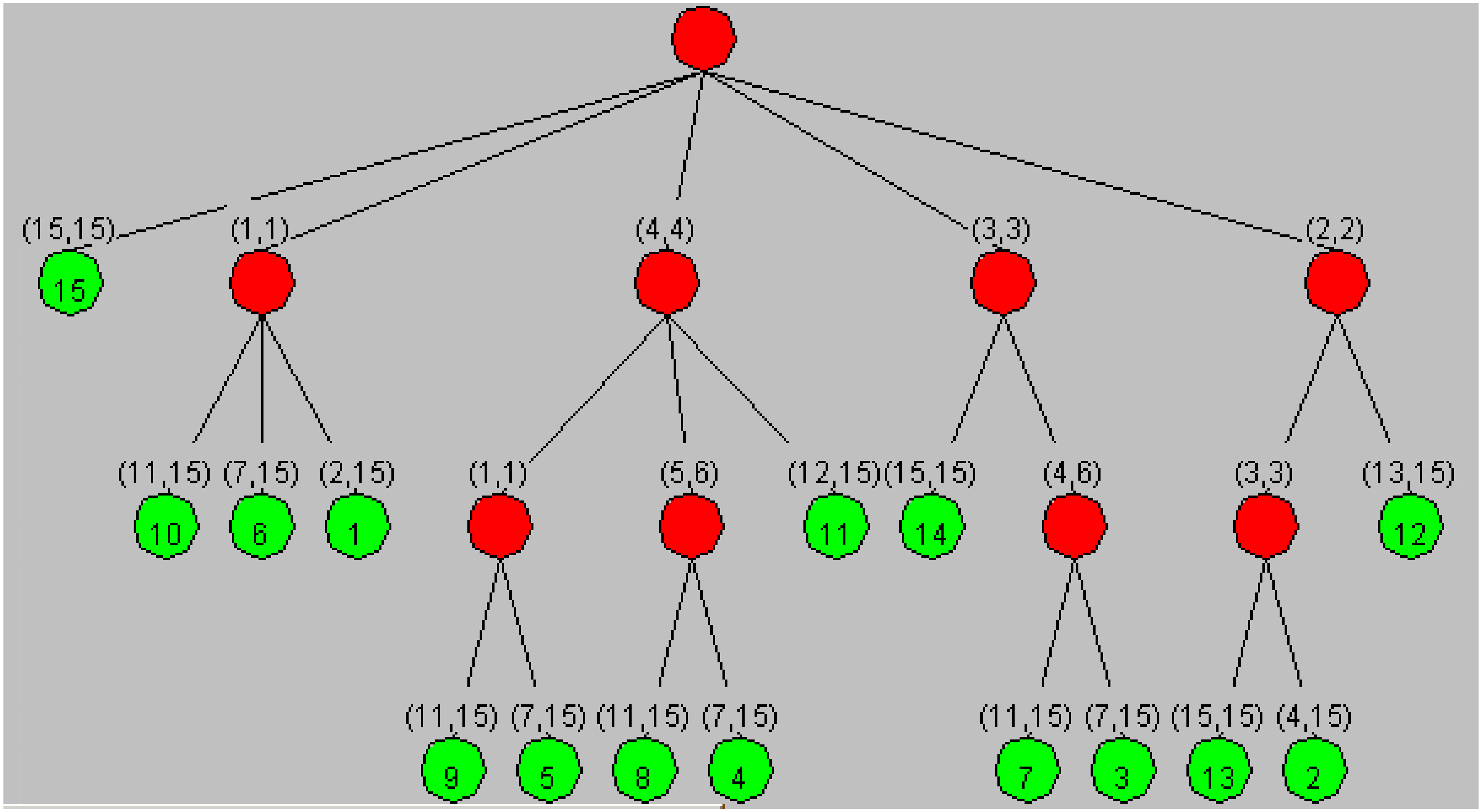
Our naive algorithm takes O(n^2) steps in the worst case, which can be quite long if our search string is the human genome.

Martin Farach-Colton, right here at Rutgers / DIMACS, developed a linear-time algorithm for generating suffix trees.

Storage space for his tree is about 25 bytes per character in the search string.

WHAT!??

Edge label encoding is used...



Your Turn

Let’s generate a suffix tree for “CCGCATGCAAT”

Reading Edge-Label Encodings

Search the edge-label encoding for the following strings:

CACT

TTT

C

GCCAA

Constructing Edge-Label Encodings

Here is our string: ATGCCAGCCACTTG

Position table:

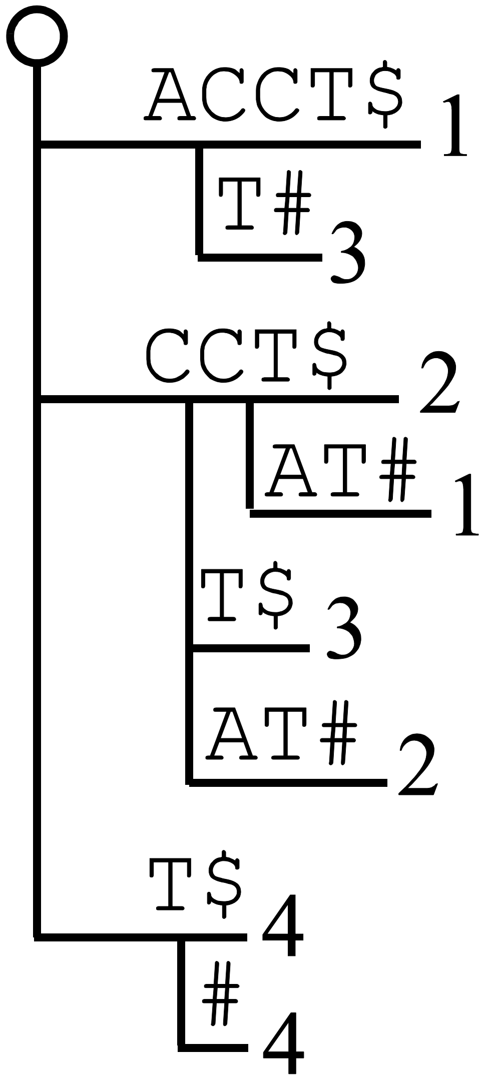
A	T	G	C	C	A	G	C	C	A	C	T	T	G	$
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Suffix Trees on Pairs of Strings

A small modification allows us to encode *two* search strings on a single suffix tree.

The trick is to use separate string termination symbols, such as “$” and “#.”

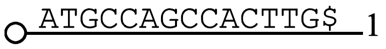
Here’s the tree for “ACCT”($) and “CCAT”(#)



Handout #1 — Generating a Suffix Tree

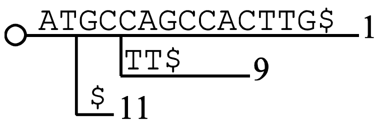
Steps to Generate a Suffix Tree:

1. Make a list of all suffixes in the search string, with "$" as an end marker
2. Write down the longest one (i.e., the whole string) on a single edge connected to the root



3. Go through the list from longest to shortest, adding the suffixes one at a time. As each string is added, walk down the tree as far as possible, searching for as long a prefix of that suffix as can be found, before adding a new branch to the tree. Mark each new leaf with the index of where the prefix started.

E.g., if we add "ATGCTT" to our tree, then "AT":



Why not try building a suffix tree for the string "CCGCATGCAAT"

Handout #2 — Edge Label Encoding

Construct a suffix tree for “GACACCCATAAC” using edge-label encoding. Here’s a helpful table:

G	A	C	A	C	C	C	A	T	A	A	C
1	2	3	4	5	6	7	8	9	10	11	12

Exercises — Suffix Trees

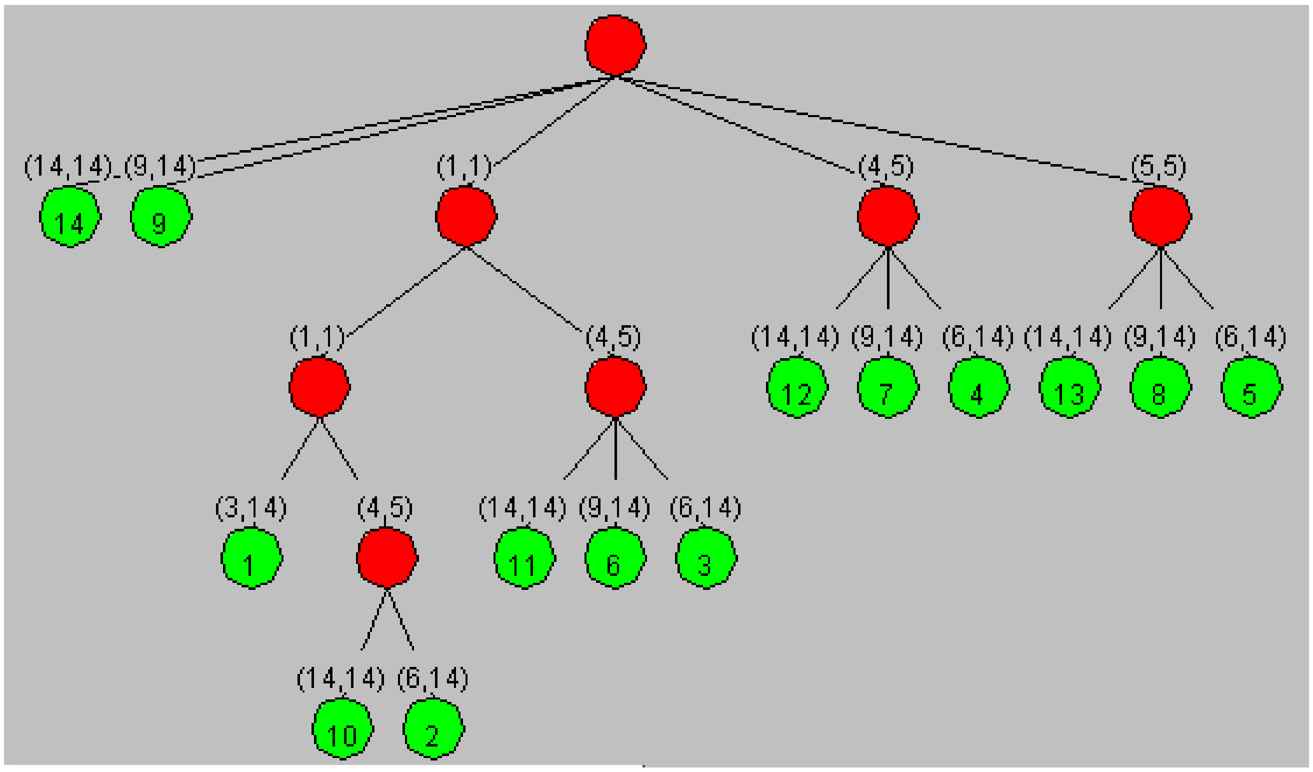
Presentation Problems:

1. Prove that every unrooted binary tree containing at least 3 vertices has a pair of leaves adjacent to a common internal node.

2. Build the suffix tree for the search string "CACCTACCTCACC."

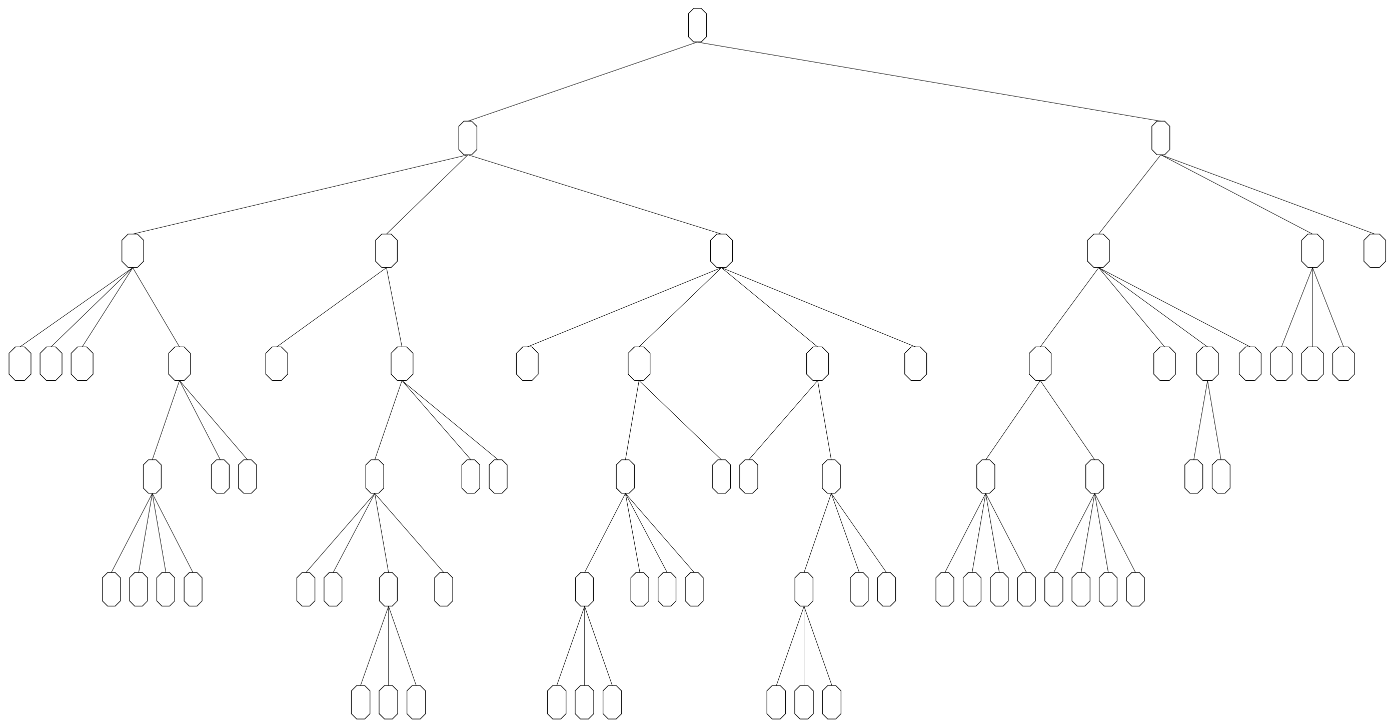
3. Show how to read, from the suffix tree in problem 2, where all occurrences of the string "ACC" can be found.

4. Here is a suffix tree for the string for one of the strings "CAAATGTGCATGC," "CCCGTCGTACCGT" or "TTCACAAACTGGT," represented with edge label compression. Which string?



5. How would you use the edge-label compressed suffix tree given above to find all occurrences of "CGT" in the search string? (Assume that you know which string gave rise to the tree.)

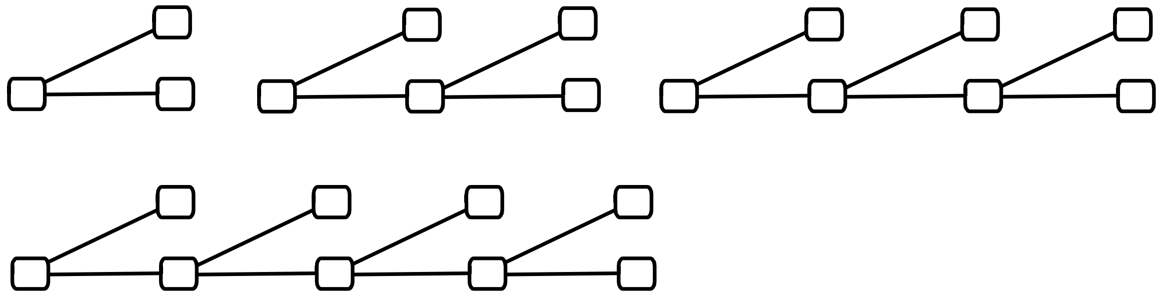
6. Here is a suffix tree for some search string with details graciously omitted. Suppose that each edge has a string of length exactly 2, except for the edges at the leaves, which are of varying lengths. (There are more copies of this tree on the last page.)
   a. Compute for each node the number of leaves that descend (not necessarily directly) from that node. (Consider leaves as having one leaf-descendent, including itself.)
   b. What is the length of the longest string that occurs at least 8 times in the search string? How many such strings are there? Do you see a neat dynamic programming way to do this?



7. Build the suffix tree that encodes the *two* strings "CAGAGT" and "GAGAG." Show how to use your string to find the longest common substring to those two strings.

8. Use the idea in the previous problem to show how to find palindromes in the string ACATTACAT

9. Here are four suffix trees, for four different strings, each of which begins with "A." What are the four strings?



Gratuitous copies of the suffix tree: