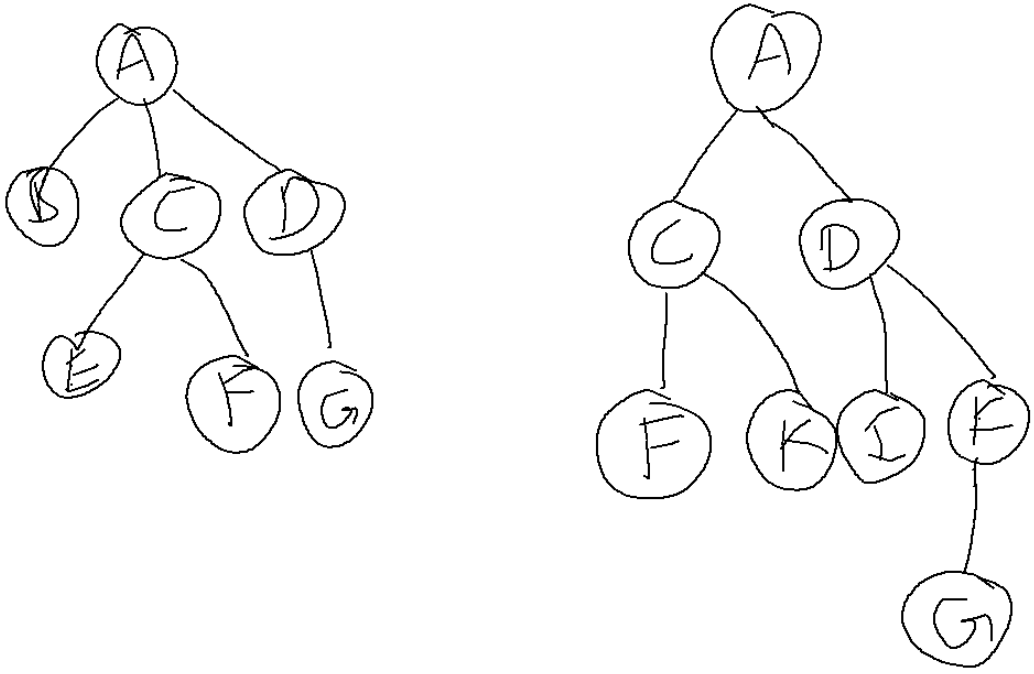
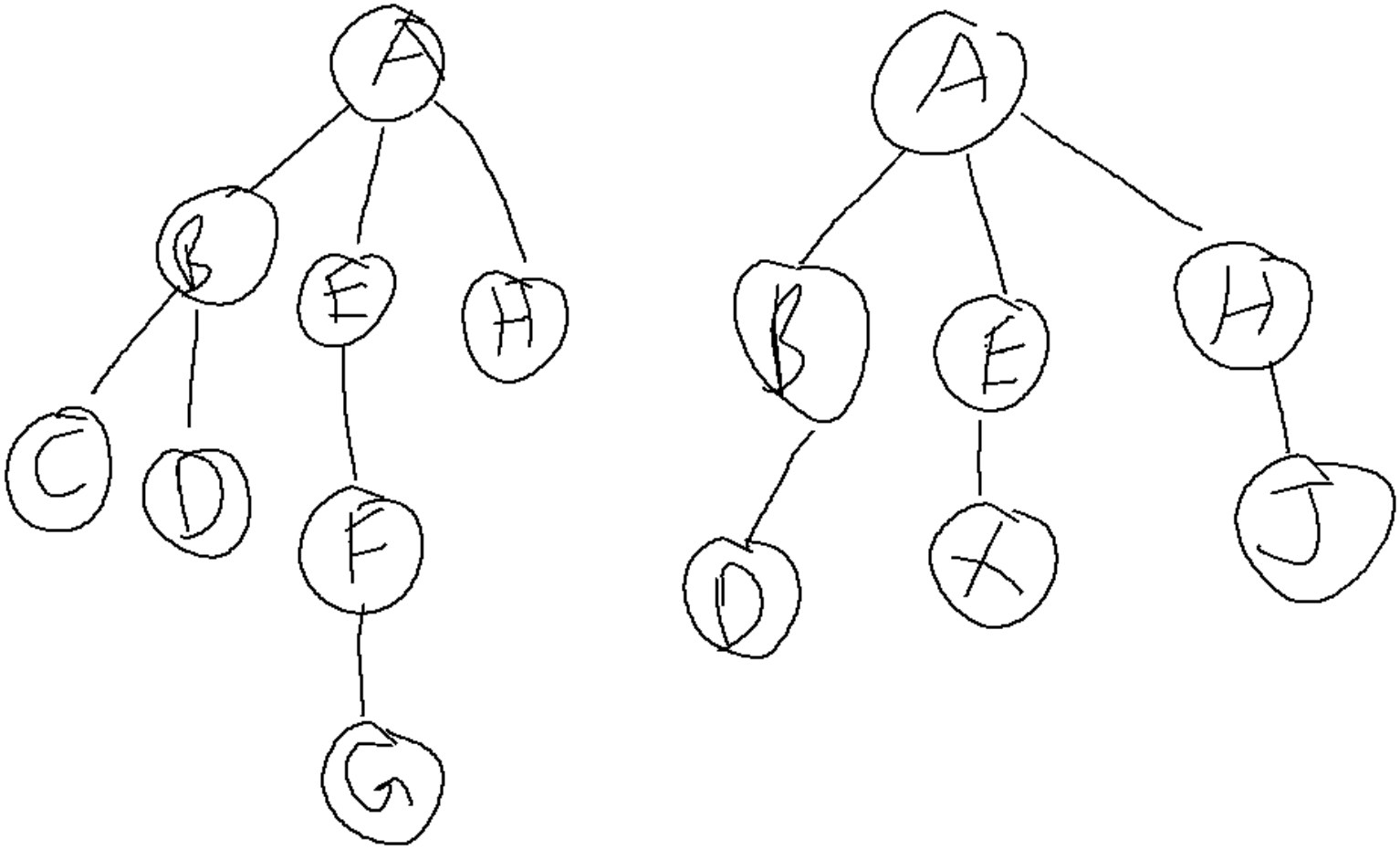
Example

- Find similarity between the following trees using simple tree matching (STM)



Find similarities between the following trees using STM



Time and Space Complexities

- Comparison of two strings
  - Time complexity: $O(|s_1||s_2|)$
  - Space complexity: $O(|s_1||s_2|)$
- Comparison of two trees
  - Time complexity: $O(n_1 n_2 h_1 h_2)$, where n_1 and n_2 are number of nodes of the trees and h_1 and h_2 are height of the trees.

Road map

- Introduction
- Data Model and HTML encoding
- Wrapper induction
- Automatic Wrapper Generation: Two Problems
- String Matching and Tree Matching
- **Multiple Alignments**
- Building DOM Trees
- Extraction Given a List Page: Flat Data Records
- Extraction Given a List Page: Nested Data Records
- Extraction Given Multiple Pages
- Summary

Multiple alignment

- Previous techniques were used to compare two strings or two trees.
- Pairwise alignment is not sufficient because a web page usually contain more than one data records.
- We need multiple alignment to compare more than two strings or two trees
- We discuss two techniques
  - Center Star method
  - Partial tree alignment.

Center star method

- This is a classic technique, and quite simple. It commonly used for multiple string alignments, but can be adopted for trees.
- Let the set of strings to be aligned be *S*. In the method, a string $s_c \in S$ that minimizes,

$$\sum_{s_i \in S} d(s_c, s_i) \qquad (3)$$

- is first selected as the **center string**. $d(s_c, s_i)$ is the distance of two strings.
- The algorithm then iteratively computes the alignment of rest of the strings with s_c.

The algorithm

CenterStar(S)

1. choose the center star s_c using Equation (3);
2. initialize the multiple sequence alignment M that contains only s_c;
4. **for** each s in S-$\{s_c\}$ **do**
5. let c^* be the aligned version of s_c in M;
6. let s' and $c^{*\prime}$ be the optimally aligned strings of s and c^*;
7. add aligned strings s' and $c^{*\prime}$ into the multiple alignment M;
8. add spaces to each string in M, except, s' and $c^{*\prime}$, at locations where new spaces are added to c^*
9. **endfor**
10. **return** multiple string alignment M

An example

Example 2: We have three strings, i.e., S = {ABC, XBC, XAB}. ABC is selected as the center string s_c. Let us align the other strings with ABC.

Iteration 1: Align c^* (= s_c) with s =XBC:

$$
\begin{array}{llccc}
c^{*\prime}: & & A & B & C \\
 & & & | & | \\
s': & & X & B & C
\end{array}
$$

Update M: A B C →

$$
\begin{array}{ccc}
A & B & C \\
X & B & C
\end{array}
$$

Iteration 2: Align c^* with s = XAB:

$$
\begin{array}{llcccc}
c^{*\prime}: & & - & A & B & C \\
 & & & | & | & \\
s': & & X & A & B & -
\end{array}
$$

Update M:

$$
\begin{array}{ccc}
A & B & C \\
X & B & C
\end{array}
\rightarrow
\begin{array}{cccc}
\mathbf{-} & \mathbf{A} & \mathbf{B} & \mathbf{C} \\
\mathbf{-} & \mathbf{X} & \mathbf{B} & \mathbf{C} \\
\mathbf{X} & \mathbf{A} & \mathbf{B} & \mathbf{-}
\end{array}
$$

The shortcomings

- Assume there are k strings in S and all strings have length n, finding the center takes $O(k^2n^2)$ time and the iterative pair-wise alignment takes $O(kn^2)$ time. Thus, the overall time complexity is $O(k^2n^2)$.

For our data extraction task, this method has two shortcomings:

1. the algorithm runs slowly for pages containing many data records and/or data records containing many tags (i.e., long strings) because finding the center string needs $O(k^2n^2)$ time.
2. if the center string (or tree) does not have a particular data item, other data records that contain the same data item may not be aligned properly. For example, the letter X's in the last two strings (in bold) are not aligned in the final result, but they should.

Shortcomings (cont …)

- Giving the cost of 1 for “changing a letter” in edit distance is problematic (e.g., A and X in the first and second strings in the final result) because of optional data items in data records.
- The problem can be partially dealt with by disallowing “changing a letter” (e.g., giving it a larger cost). However, this introduces another problem.
- For example, if we align only ABC and XBC, it is not clear which of the following alignment is better.

(1)
A – B C
– X B C

(2)
– A B C
X – B C

Example 1

- Align the following strings
  - ABRGQ
  - BRTGZ
  - AQRTQ
  - AAGGQ

Example 2

- Align the following strings
- ERNSTY
- QRSTXY
- PRISTTY