Example 2

- Align the following strings
  - ERNSTY
  - QRSTXY
  - PRISTTY
- What should be the centre star?
  A. ERNSTY
  B. QRSTXY
  C. PRISTTY
  D. Any one

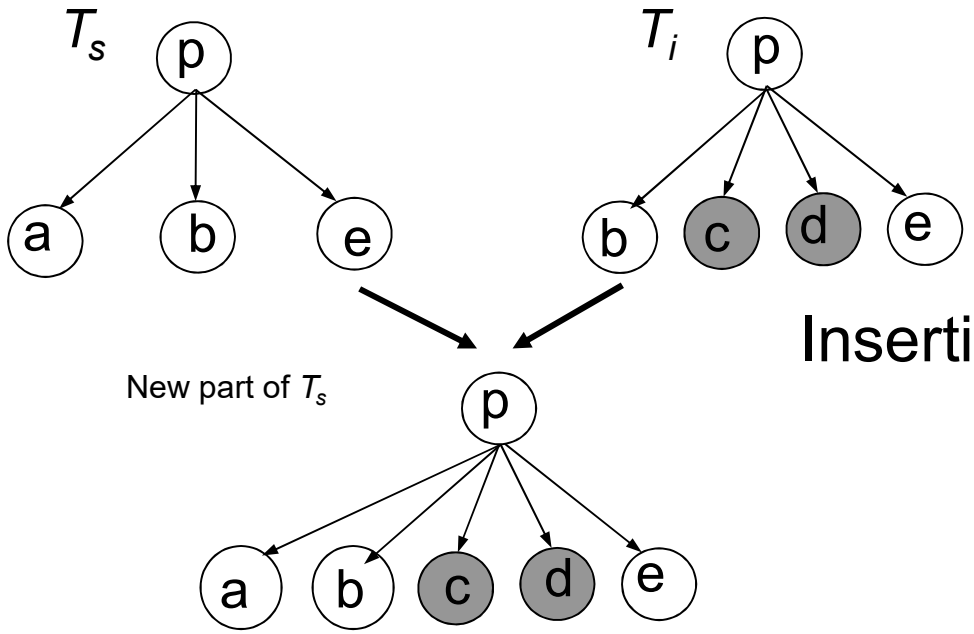
The partial tree alignment method

- Choose a seed tree: A seed tree, denoted by T_s, is picked with the maximum number of data items.
- The seed tree is similar to center string, but without the $O(k^2n^2)$ pair-wise tree matching to choose it.
- Tree matching:

  For each unmatched tree T_i ($i \neq s$),
  - match T_s and T_i.
  - Each pair of matched nodes are linked (aligned).
  - For each unmatched node n_j in T_i do
    - expand T_s by inserting n_j into T_s if a position for insertion can be uniquely determined in T_s.

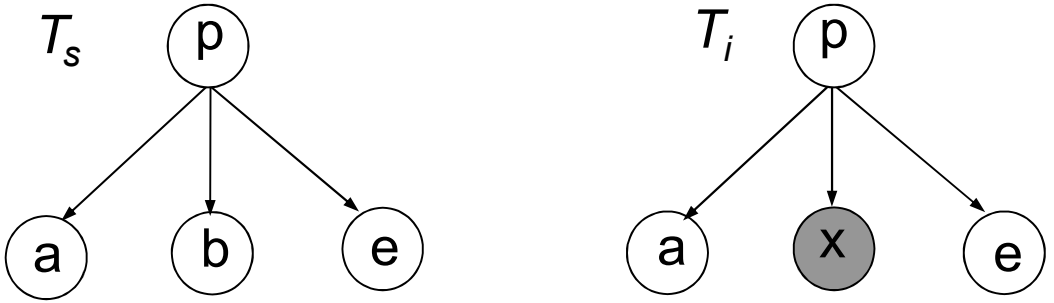
  The expanded seed tree T_s is then used in subsequent matching.

Partial tree alignment of two trees



Insertion is possible

Insertion is not possible



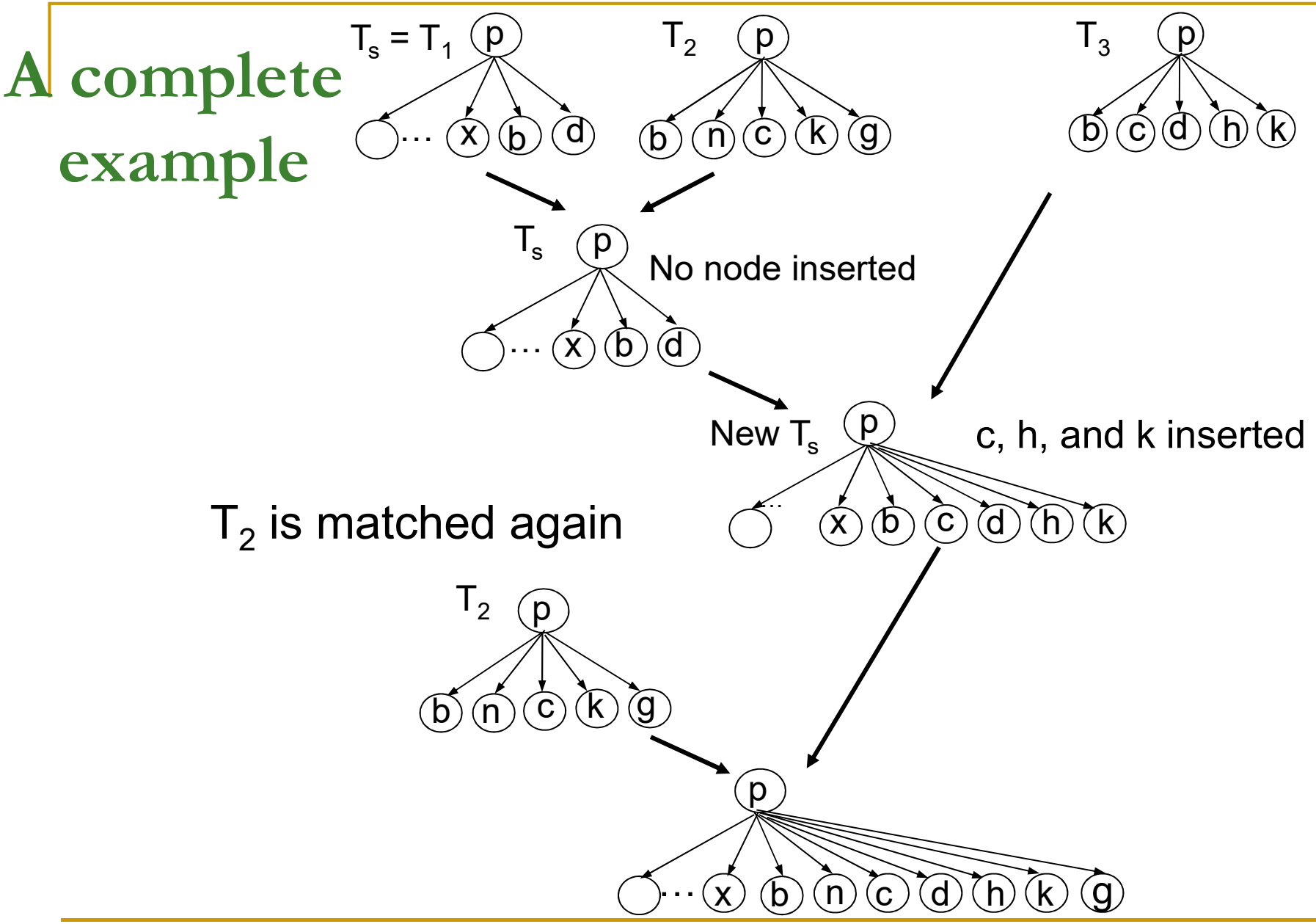
Partial alignment of two trees

Algorithm PartialTreeAlignment(S)

1. Sort trees in S in descending order of the number of unaligned data items;
2. $T_s \leftarrow$ the first tree (which is the largest) and delete it from S;
3. $R \leftarrow \varnothing$;
4. **while** ($S \neq \varnothing$) **do**
5. $T_i \leftarrow$ select and delete next tree from S; // follow the sorted order
6. STM(T_s, T_i); // tree matching
7. AlignTrees(T_s, T_i); // based on the result from line 6
8. **if** T_i is not completely aligned with T_s **then**
9. **if** InsertIntoSeed(T_s, T_i) **then** // True: some insertions are done
10. $S = S \cup R$;
11. $R \leftarrow \varnothing$
12 **endif**;
13. **if** there are still unaligned items in T_i that are not inserted into T_s **then**
14. $R \leftarrow R \cup \{T_i\}$
15. **endif**;
16. **endif**;
17. **endwhile**;
18. Output data fields from each T_i to a data table based on the alignment results.

Fig. 21. The partial tree alignment algorithm

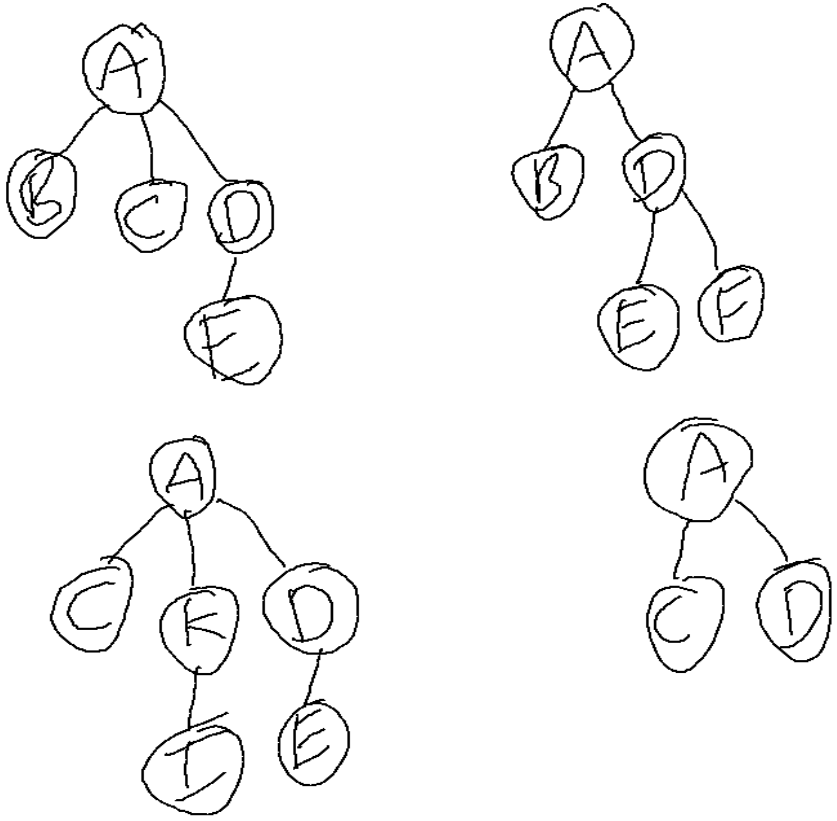
A complete example



Output Data Table

	…	**x**	**b**	**n**	**c**	**d**	**h**	**k**	**g**
T_1	…	1	1			1			
T_2			1	1	1			1	1
T_3			1		1	1	1	1	

Example: Align the following 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

Building DOM trees

- We now start to talk about actual data extraction.
- The usual first step is to build a DOM tree (tag tree) of a HTML page.
  - Most HTML tags work in pairs. Within each corresponding tag-pair, there can be other pairs of tags, resulting in a nested structure.
  - Building a DOM tree from a page using its HTML code is thus natural.
- In the tree, each pair of tags is a **node**, and the nested tags within it are the **children** of the node.

Two steps to build a tree

- **HTML code cleaning**:
  - Some tags do not require closing tags (e.g., <li>, <hr> and <p>) although they have closing tags.
  - Additional closing tags need to be inserted to ensure all tags are balanced.
  - Ill-formatted tags need to be fixed. One popular program is called **Tidy**, which can be downloaded from http://tidy.sourceforge.net/.
- **Tree building**: simply follow the nested blocks of the HTML tags in the page to build the DOM tree. It is straightforward.

Building tree using tags & visual cues

- Correcting errors in HTML can be hard.
- There are also dynamically generated pages with scripts.
- Visual information comes to the rescue.
- As long as a browser can render a page correct, a tree can be built correctly.
  - Each HTML element is rendered as a rectangle.
  - Containments of rectangles representing nesting.

An example

```
1  <table>
2     <tr>
3        <td>…</td>
4        <td>…</td>
5     </tr>
6     <tr>
7        <td>…</td>
8        <td>…</td>
9     </tr>
10 </table>
```

left	right	top	bottom
100	300	200	400
100	300	200	300
100	200	200	300
200	300	200	300
100	300	300	400
100	200	300	400
200	300	300	400

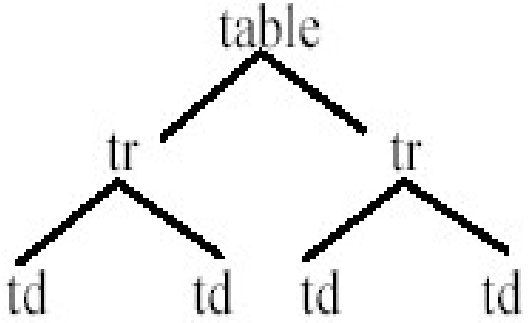


Fig. 23. A HTML code segment, boundary coordinates and the resulting tree

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

Extraction Given a List Page: Flat Data Records

- Given a single list page with multiple data records,
  - Automatically segment data records
  - Extract data from data records.
- Since the data records are flat (no nested lists), string similarity or tree matching can be used to find similar structures.
  - Computation is a problem
  - A data record can start anywhere and end anywhere

Two important observations

- **Observation 1**: A group of data records that contains descriptions of a set of similar objects are typically presented in a contiguous region of a page and are formatted using similar HTML tags. Such a region is called a **data region**.
- **Observation 2**: A set of data records are formed by some child sub-trees of the same parent node.

An example

1. 

Customer Rating:

Apple iBook Notebook M8600LL/A (600-MHz PowerPC G3, 128 MB RAM, 20 GB hard drive)

Buy new: $1,194.00

Usually ships in 1 to 2 days

Best use: (what's this?)	Business:	Portability:	Desktop Replacement:	Entertainment:
	●●●○○	●●●●●	●●●○○	●●●○○

600 MHz PowerPC G3, 128 MB SRAM, 20 GB Hard Disk, 24x CD-ROM, AirPort ready, and Mac OS X, Mac OS X,Mac OS 9.2,Quick Time,iPhoto,iTunes 2,iMovie 2,AppleWorks,Microsoft IE

2. 

Customer Rating:

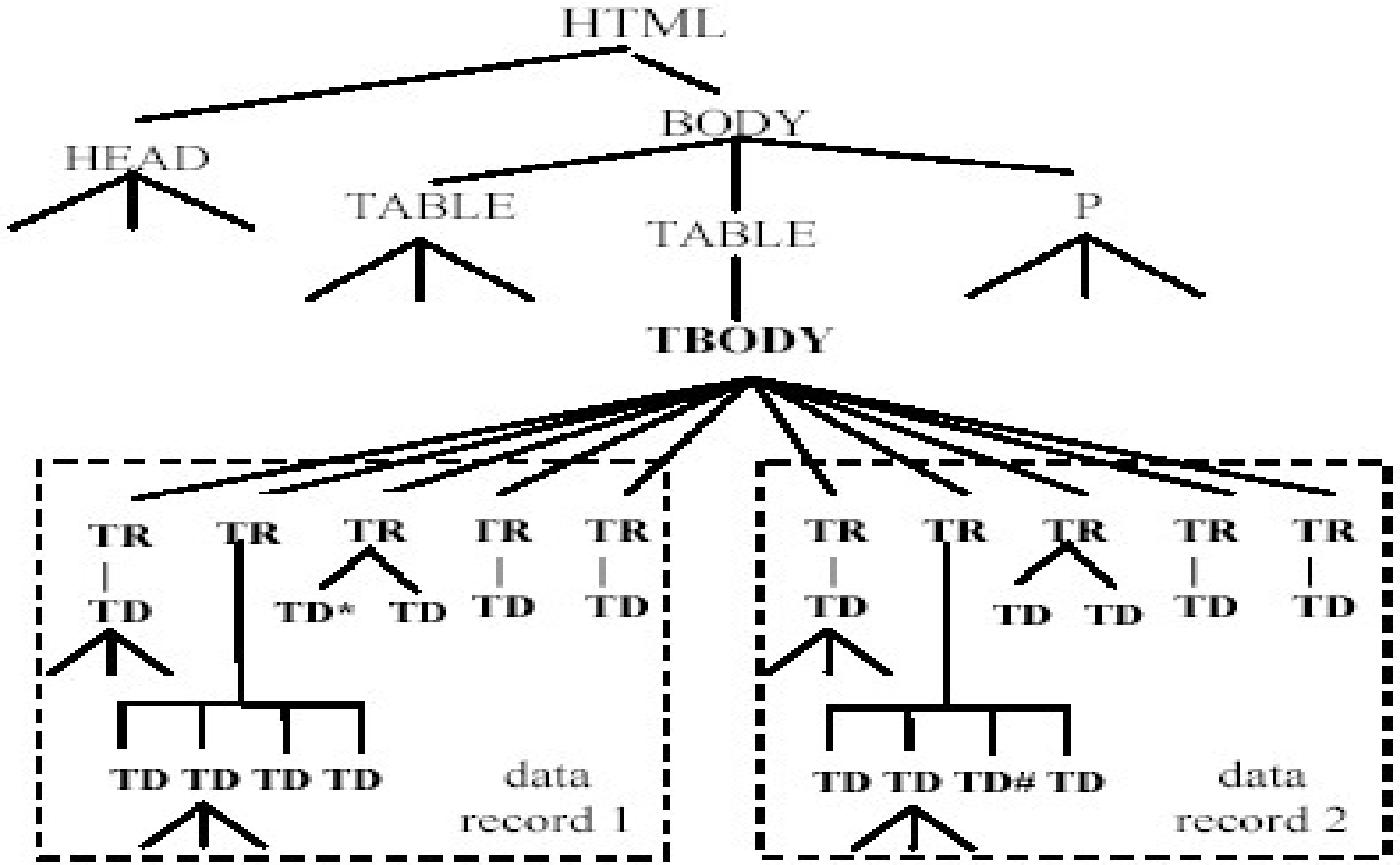
Apple Powerbook Notebook M8591LL/A (667-MHz PowerPC G4, 256 MB RAM, 30 GB hard drive)

Buy new: $2,399.99

Best use: (what's this?)	Portability:	Desktop Replacement:	Entertainment:
	●●●●○	●●●●○	●●●●○

667 MHz PowerPC G4, 256 MB SDRAM, 30 GB Ultra ATA Hard Disk, 24x (read), 8x (write) CD-RW, 8x; included via combo drive DVD-ROM, and Mac OS X, QuickTime, iMovie 2, iTunes(6), Microsoft Internet Explorer, Microsoft Outlook Express, ...

The DOM tree


HTML
HEAD
BODY
TABLE
TABLE
P
TBODY
TR
TD
TD*
data record 1
data record 2

The Approach

Given a page, three steps:

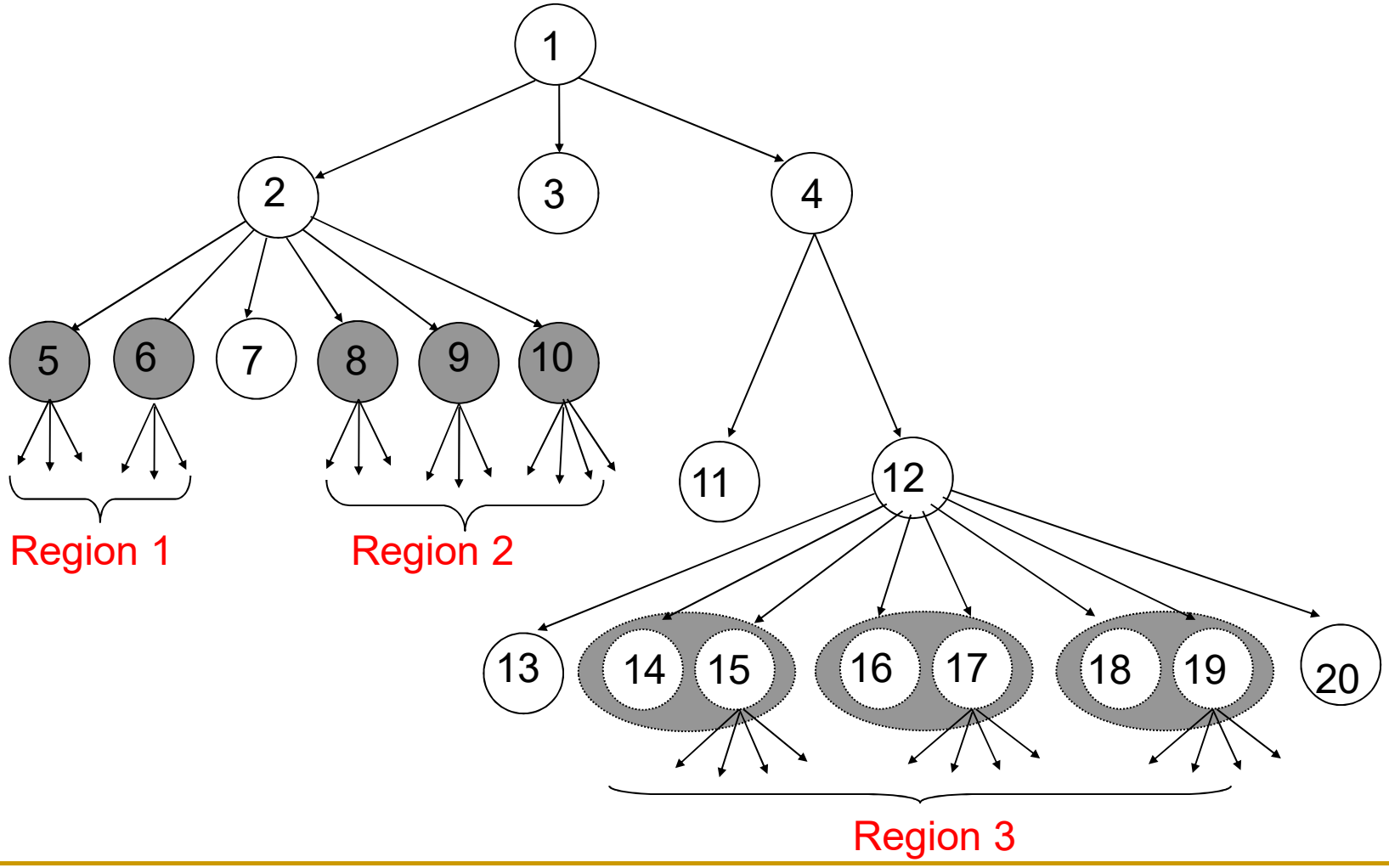
- Building the HTML Tag Tree
  - Erroneous tags, unbalanced tags, etc
- Mining Data Regions
  - Spring matching or tree matching
- Identifying Data Records

Rendering (or visual) information is very useful in the whole process

Mining a set of similar structures

- **Definition:** A *generalized node* (a *node combination*) of length *r* consists of *r* ($r \geq 1$) nodes in the tag tree with the following two properties:
  - the nodes all have the same parent.
  - the nodes are adjacent.
- **Definition:** A *data region* is a collection of two or more generalized nodes with the following properties:
  - the generalized nodes all have the same parent.
  - the generalized nodes all have the same length.
  - the generalized nodes are all adjacent.
  - the similarity between adjacent generalized nodes is greater than a fixed threshold.

Mining Data Regions



Mining data regions

- We need to find where each generalized node starts and where it ends.
  - perform string or tree matching
- Computation is not a problem anymore
  - Due to the two observations, we only need to perform comparisons among the children nodes of a parent node.
  - Some comparisons done for earlier nodes are the same as for later nodes (see the example below).

Comparison

We use Fig. 27 to illustrate the comparison process. Fig. 27 has 10 nodes below a parent node *p*. We start from each node and perform string (or tree) comparison of all possible combinations of component nodes. Let the maximum number of components that a generalized node can have be 3 in this example.

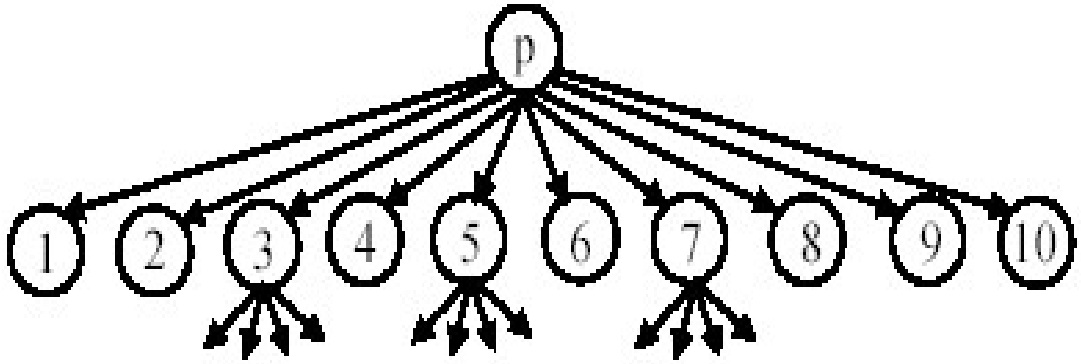


Fig. 27. Combination and comparison

Comparison (cont …)

Start from node 1: We compute the following string comparisons.

- (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (8, 9), (9, 10)
- (1-2, 3-4), (3-4, 5-6), (5-6, 7-8), (7-8, 9-10)
- (1-2-3, 4-5-6), (4-5-6, 7-8-9)

(1, 2) means that the tag string of node 1 is compared with the tag string of node 2. The tag string of a node includes all the tags of the sub-tree of the node. (1-2, 3-4) means that the combined tag string of nodes 1 and 2 is compared with the combined tag string of nodes 3 and 4.

Start from node 2: We only compute:

- (2-3, 4-5), (4-5, 6-7), (6-7, 8-9)
- (2-3-4, 5-6-7), (5-6-7, 8-9-10)

The MDR algorithm

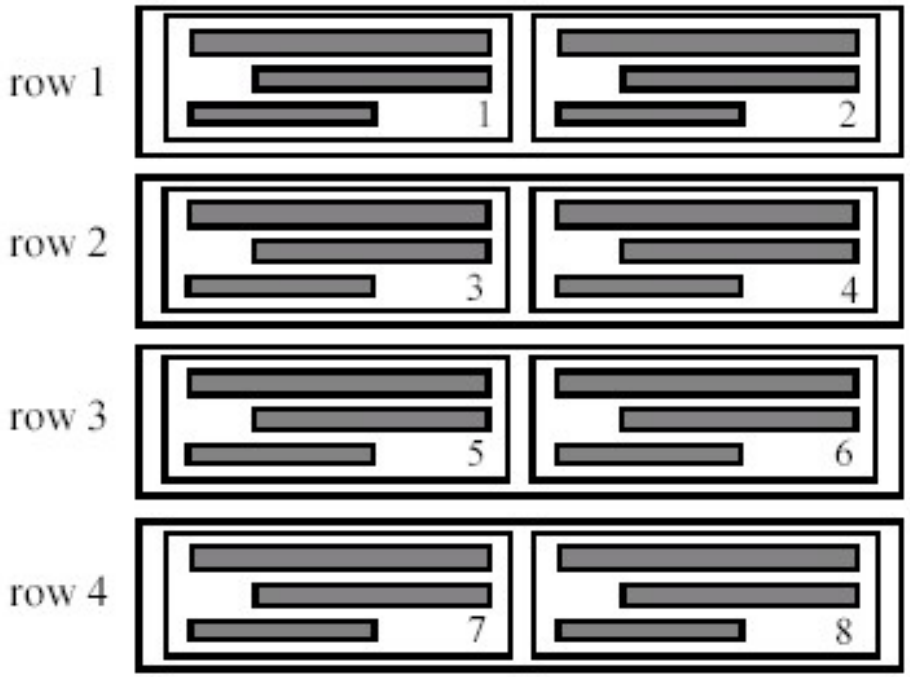
- K: Maxium number of tag nodes
- T: Similarity threshold

```
Algorithm MDR(Node, K, T)
1  if TreeDepth(Node) >= 3 then
2      CombComp(Node.Children, K);
3      DataRegions ← IdenDRs(Node, K, T);
4      if (UncoveredNodes ← Node.Children − ⋃_{DR∈DataRegions} DR ) ≠ ∅ then
5          for each ChildNode ∈ UncoveredNodes do
6              DataRegions ← DataRegions ∪ MDR(ChildNode, K, T);
7      return DataRegions
8  else return ∅
```

Fig. 28. The overall algorithm

Find data records from generalized nodes

- A generalized node may not represent a data record.
- In the example on the right, each row is found as a generalized node.
- This step needs to identify each of the 8 data record.
  - Not hard
  - We simply run the MDR algorithm given each generalized node as input
- There are some complications (read the notes)



2. Extract Data from Data Records

- Once a list of data records is identified, we can align and extract data items from them.
- Approaches (align multiple data records):
  - Multiple string alignment
    - Many ambiguities due to pervasive use of table related tags.
  - Multiple tree alignment (partial tree alignment)
    - Together with visual information is effective

Generating extraction patterns and data extraction

- Once data records in each data region are discovered, we align them to produce an extraction pattern that can be used to extract data from the current page and also other pages that use the same encoding template.
- P**artial tree alignment algorithm** is just for the purpose.
- Visual information can help in various ways (read the notes)