

# Approximation: Theory and Algorithms

## Edit Distance Complexity, Upper and Lower Bounds

Nikolaus Augsten

Free University of Bozen-Bolzano  
Faculty of Computer Science  
DIS

Unit 8 – April 22, 2009

## Outline

- 1 Key Roots and Left-Most Leaf Descendants
- 2 Example: Tree Edit Distance Computation
- 1 Conclusion

Key Roots and Left-Most Leaf Descendants

## The Edit Distance Algorithm I/II

`tree-edit-dist( $T_1, T_2$ )`

```
td[1..|T1|, 1..|T2|] : empty array for tree distances;  
l1 = lmdl(root(T1)); kr1 = kr(l1, |leaves(T1)|);  
l2 = lmdl(root(T2)); kr2 = kr(l2, |leaves(T2)|);  
for x = 1 to |kr1| do  
  for y = 1 to |kr2| do  
    forest-dist(kr1[x], kr2[y], l1, l2, td);
```

- We need the following algorithms:
  - $lmdl(i)$ : computes an array with the left-most leaf descendants of all descendants of a node  $i$
  - $kr(l, lc)$ : given the array  $l = lmdl(i)$  of left-most leaf descendants, and the number  $lc$  of leaf descendants of  $i$ , compute all key roots of the subtree rooted in  $i$

Key Roots and Left-Most Leaf Descendants

## Computing the Left-Most Leaf Descendants

`lmdl( $v, l$ )`

```
foreach child c of v (left to right) do l ← lmdl(c, l);  
if v is a leaf then  
  l[id(v)] ← id(v)  
else  
  c1 ← first child of v;  
  l[id(v)] ← l[id(c1)];  
return l;
```

- Input: root node  $v$  of a tree  $T$ , empty array  $l[1..|T|]$
- Output: array  $l$ ,  $l[i]$  is the left-most leaf descendant of node  $T[i]$

## Computing the Key Roots

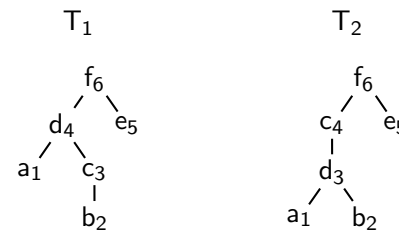
$kr(l, lc)$

```

kr[1..lc]: empty array;
visited[]: boolean array of size |l|, init with false;
k ← |kr|; i ← |l|;
while k ≥ 1 do
  if not visited[l[i]] then
    kr[k-] ← i;
    visited[l[i]] ← true;
    i--;
return sort(kr);
    
```

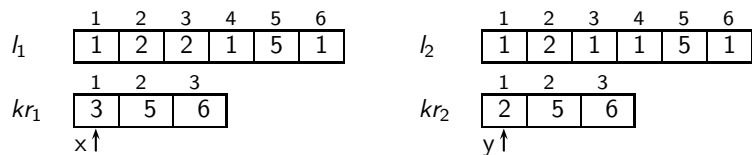
- Input:
  - $l[1..|T|]$ :  $l[i]$  is the left-most leaf descendent of node  $T[i]$
  - $lc = |leaves(T)|$  is the number of leaves in  $T$
- Output: array  $kr[1..|leaves(T)|]$  with key roots sorted by node ID
- Note: Loop condition is correct, as  $k \geq 1 \Rightarrow i \geq 1$  (the number of key roots is exactly the number of leaves, and it will always be filled when all nodes are traversed)

## Example Trees and Edit Costs



- Example: Edit distance between  $T_1$  and  $T_2$ .
  - $\omega_{ins} = \omega_{del} = 1$
  - $\omega_{ren} = 0$  for identical rename, otherwise  $\omega_{ren} = 1$
- Each of the following slide is the result of a call of forest-dist().

## Executing the Algorithm (1/9)



- $i = kr_1[x] = 3 \Rightarrow l_1[i] = 2$
- $j = kr_2[y] = 2 \Rightarrow l_2[j] = 2$

• temporary array  $fd$ :

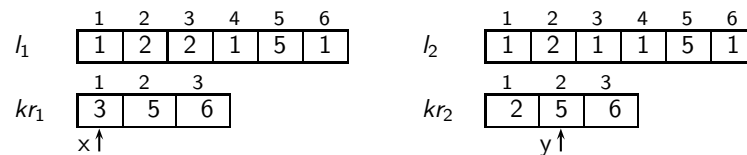
		$d_j \rightarrow 2$	
$d_i \downarrow$	0	1	
2	1	0	
3	2	1	

■  $h_1[i] = h_1[d_i]$  and  $h_2[j] = h_2[d_j]$

• permanent array  $td$ :

		1	2	3	4	5	6
1							
2		0					
3		1					
4							
5							
6							

## Executing the Algorithm (2/9)



- $i = kr_1[x] = 3 \Rightarrow l_1[i] = 2$
- $j = kr_2[y] = 5 \Rightarrow l_2[j] = 5$

• temporary array  $fd$ :

		$d_j \rightarrow 5$	
$d_i \downarrow$	0	1	
2	1	1	
3	2	2	

■  $h_1[i] = h_1[d_i]$  and  $h_2[j] = h_2[d_j]$

• permanent array  $td$ :

		1	2	3	4	5	6
1							
2		0			1		
3		1			2		
4							
5							
6							

## Executing the Algorithm (3/9)

$h_1$

1	2	3	4	5	6
1	2	2	1	5	1

$h_2$

1	2	3	4	5	6
1	2	1	1	5	1

$kr_1$

1	2	3
3	5	6

$x \uparrow$

$kr_2$

1	2	3
2	5	6

$y \uparrow$

- $i = kr_1[x] = 3 \Rightarrow h_1[i] = 2$
- $j = kr_2[y] = 6 \Rightarrow h_2[j] = 1$

- temporary array  $fd$ :

	$d_j \rightarrow$	1	2	3	4	5	6
$d_i \downarrow$	0	1	2	3	4	5	6
2	1	1	1	2	3	4	5
3	2	2	2	2	2	3	4

$h_1[i] = h_1[d_i]$  and  $h_2[j] = h_2[d_j]$

- permanent array  $td$ :

	1	2	3	4	5	6
1						
2	1	0	2	3	1	5
3	2	1	2	2	2	4
4						
5						
6						

## Executing the Algorithm (4/9)

$h_1$

1	2	3	4	5	6
1	2	2	1	5	1

$h_2$

1	2	3	4	5	6
1	2	1	1	5	1

$kr_1$

1	2	3
3	5	6

$x \uparrow$

$kr_2$

1	2	3
2	5	6

$y \uparrow$

- $i = kr_1[x] = 5 \Rightarrow h_1[i] = 5$
- $j = kr_2[y] = 2 \Rightarrow h_2[j] = 2$

- temporary array  $fd$ :

	$d_j \rightarrow$	2
$d_i \downarrow$	0	1
5	1	1

$h_1[i] = h_1[d_i]$  and  $h_2[j] = h_2[d_j]$

- permanent array  $td$ :

	1	2	3	4	5	6
1						
2	1	0	2	3	1	5
3	2	1	2	2	2	4
4						
5		1				
6						

## Executing the Algorithm (5/9)

$h_1$

1	2	3	4	5	6
1	2	2	1	5	1

$h_2$

1	2	3	4	5	6
1	2	1	1	5	1

$kr_1$

1	2	3
3	5	6

$x \uparrow$

$kr_2$

1	2	3
2	5	6

$y \uparrow$

- $i = kr_1[x] = 5 \Rightarrow h_1[i] = 5$
- $j = kr_2[y] = 5 \Rightarrow h_2[j] = 5$

- temporary array  $fd$ :

	$d_j \rightarrow$	5
$d_i \downarrow$	0	1
5	1	0

$h_1[i] = h_1[d_i]$  and  $h_2[j] = h_2[d_j]$

- permanent array  $td$ :

	1	2	3	4	5	6
1						
2	1	0	2	3	1	5
3	2	1	2	2	2	4
4						
5		1			0	
6						

## Executing the Algorithm (6/9)

$h_1$

1	2	3	4	5	6
1	2	2	1	5	1

$h_2$

1	2	3	4	5	6
1	2	1	1	5	1

$kr_1$

1	2	3
3	5	6

$x \uparrow$

$kr_2$

1	2	3
2	5	6

$y \uparrow$

- $i = kr_1[x] = 5 \Rightarrow h_1[i] = 5$
- $j = kr_2[y] = 6 \Rightarrow h_2[j] = 1$

- temporary array  $fd$ :

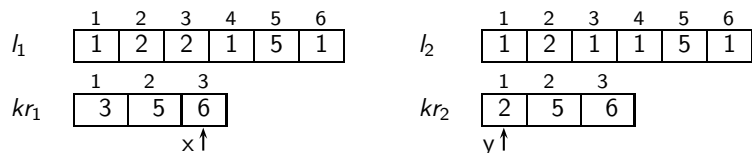
	$d_j \rightarrow$	1	2	3	4	5	6
$d_i \downarrow$	0	1	2	3	4	5	6
5	1	1	2	3	4	4	5

$h_1[i] = h_1[d_i]$  and  $h_2[j] = h_2[d_j]$

- permanent array  $td$ :

	1	2	3	4	5	6
1						
2	1	0	2	3	1	5
3	2	1	2	2	2	4
4						
5	1	1	3	4	0	5
6						

## Executing the Algorithm (7/9)



- $i = kr_1[x] = 6 \Rightarrow l_1[i] = 1$
- $j = kr_2[y] = 2 \Rightarrow l_2[j] = 2$
- temporary array  $fd$ :

$d_j \rightarrow 2$

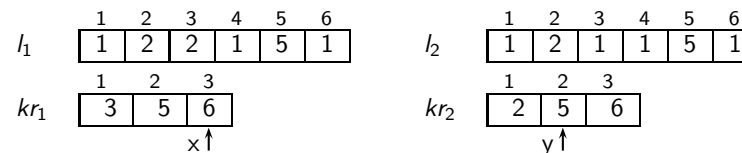
$d_i \downarrow$	0	1
1	1	1
2	2	1
3	3	2
4	4	3
5	5	4
6	6	5

- permanent array  $td$ :

	1	2	3	4	5	6
1		1				
2	1	0	2	3	1	5
3	2	1	2	2	2	4
4		3				
5	1	1	3	4	0	5
6		5				

   $l_1[i] = l_1[d_i]$  and  $l_2[j] = l_2[d_j]$

## Executing the Algorithm (8/9)



- $i = kr_1[x] = 6 \Rightarrow l_1[i] = 1$
- $j = kr_2[y] = 5 \Rightarrow l_2[j] = 5$
- temporary array  $fd$ :

$d_j \rightarrow 5$

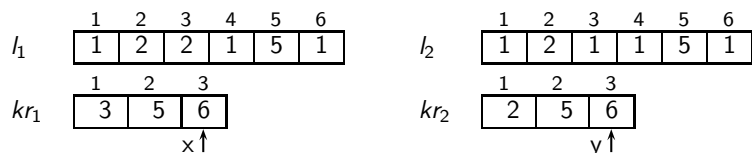
$d_i \downarrow$	0	1
1	1	1
2	2	2
3	3	3
4	4	4
5	5	4
6	6	5

- permanent array  $td$ :

	1	2	3	4	5	6
1		1			1	
2	1	0	2	3	1	5
3	2	1	2	2	2	4
4		3			4	
5	1	1	3	4	0	5
6		5			5	

   $l_1[i] = l_1[d_i]$  and  $l_2[j] = l_2[d_j]$

## Executing the Algorithm (9/9)



- $i = kr_1[x] = 6 \Rightarrow l_1[i] = 1$
- $j = kr_2[y] = 6 \Rightarrow l_2[j] = 1$
- temporary array  $fd$ :

$d_j \rightarrow 1 \ 2 \ 3 \ 4 \ 5 \ 6$

$d_i \downarrow$	0	1	2	3	4	5	6
1	1	0	1	2	3	4	5
2	2	1	0	1	2	3	4
3	3	2	1	2	3	4	5
4	4	3	2	1	2	3	4
5	5	4	3	2	3	2	3
6	6	5	4	3	3	3	2

- permanent array  $td$ :

	1	2	3	4	5	6
1	0	1	2	3	1	5
2	1	0	2	3	1	5
3	2	1	2	2	2	4
4	3	3	1	2	4	4
5	1	1	3	4	0	5
6	5	5	3	3	5	2

   $l_1[i] = l_1[d_i]$  and  $l_2[j] = l_2[d_j]$

## Summary

- Tree Edit Distance
  - Computing Key Roots and Left-Most Leaf Descendants
  - Tree Edit Distance Example

## What's Next?

- Complexity of the Tree Edit Distance
- Lower/Upper Bounds for the Tree Edit Distance