NPTEL MOOC, JAN-FEB 2015 Week 7, Module 5

DESIGNAND ANALYSIS OF ALGORITHMS

Edit Distance

MADHAVAN MUKUND, CHENNAI MATHEMATICAL INSTITUTE http://www.cmi.ac.in/~madhavan

Document similarity

- * "The students were able to appreciate the concept optimal substructure property and its use in designing algorithms"
- * "The lecture taught the students to appreciate how the concept of optimal substructures can be used in designing algorithms"
- * "The <u>lecture taught the students were able</u> to appreciate how_the concept_of_optimal_substructures_property_candit_bse_used_in_designing_algorithms"
- * 28 characters inserted, 18 deleted, 2 substituted

Edit distance

- * Minimum number of editing operations needed to transform one document to the other
 - * Insert a character
 - * Delete a character
 - * Substitute a character by another one
- * In our example, 28 characters inserted, 18 deleted, 2 substituted
- * Edit distance is at most 48

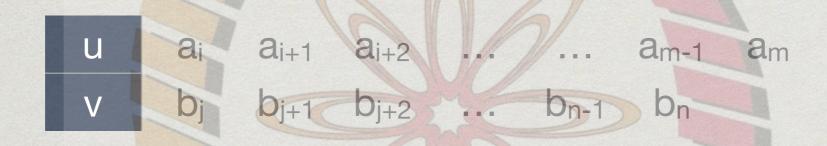
Edit distance

- * Also called Levenshtein distance
 - * First proposed by Vladimir Levenshtein
- * Applications
 - * Suggest spelling corrections in word processor, search engine queries
 - * Another way of comparing genetic similarity across species

Edit distance and LCS

- * Longest common subsequence of u and v
 - * What remains after minimum number of deletes to make them equal
- * Deleting a letter in u equivalent to inserting it in v
 - * "secret", "bisect" LCS is "sect"
 - * delete "r", "e" in "secret", "b", "i" in "bisect"
 - * delete "r", "e" then insert "b", "i" in "secret"
- * LCS is equivalent to edit distance without substitution

Inductive structure for edit distance



- * Recall LCS
 - * If $a_i = b_j$, LCS(i,j) = 1 + LCS(i+1,j+1)
 - * If $a_i \neq b_j$, LCS(i,j) = max(LCS(i+1,j), LCS(i,j+1))
 - * Boundary condition when one of the words is empty

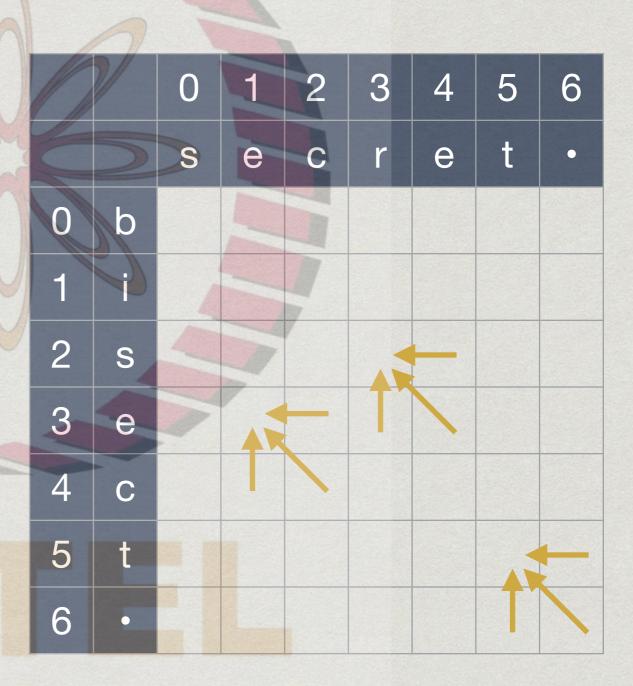
Edit distance...

- * Aim is to transform u into v
 - * If $a_i = b_j$, ED(i,j) = ED(i+1,j+1) nothing to be done at (a_i,b_j)
 - * If a_i ≠ b_j, can do one of three things
 - * Substitute a_i by b_j : 1 + ED(i+1,j+1)
 - * Delete $a_i: 1 + ED(i+1,j)$
 - * Insert b_j before a_i: 1 + ED(i,j+1)
 - * Take the minimum of these

Inductive structure

- * ED(i,j) stands for ED(aiai+1...am, bjbj+1...bn)
- * If $a_i = b_j$, ED(i,j) = ED(i+1,j+1)
- * If $a_i \neq b_j$, LCS(i,j) = 1 + min(ED(i+1,j+1), ED(i+1,j), ED(i,j+1))
- * As with LCS/LCW, extend positions to m+1, n+1
 - * ED(m+1,j) = n-j+1 for all j # Insert $b_j b_{j+1} \dots b_n$ in u
 - * ED(i,n+1) = m-i+1 for all i, # Insert $a_i a_{i+1} \dots a_m in v$

- * Like LCS, ED(i,j)
 depends on
 ED(i+1,j+1), ED(i+1,j)
 and ED(i,j+1)
- * Dependencies for ED(m,n) are known
- * Start at ED(m,n) and fill by row, column or diagonal



- * Like LCS, ED(i,j)
 depends on
 ED(i+1,j+1), ED(i+1,j)
 and ED(i,j+1)
- * Dependencies for ED(m,n) are known
- * Start at ED(m,n) and fill by row, column or diagonal

		0	1	2	3	4	5	6
	5	S	е	С	r	е	t	•
0	b							
1								
2	S							
3	е							
4	С							
5	t							
6	•							

- * Like LCS, ED(i,j)
 depends on
 ED(i+1,j+1), ED(i+1,j)
 and ED(i,j+1)
- * Dependencies for ED(m,n) are known
- * Start at ED(m,n) and fill by row, column or diagonal

	/	0	1	2	3	4	5	6
	5	S	е	С	r	е	t	•
0	b							6
1		4						5
2	S							4
3	е							3
4	С							2
5	t							1
6	•	6	5	4	3	2	1	0

- * Like LCS, ED(i,j)
 depends on
 ED(i+1,j+1), ED(i+1,j)
 and ED(i,j+1)
- * Dependencies for ED(m,n) are known
- * Start at ED(m,n) and fill by row, column or diagonal

		0	1	2	3	4	5	6
		S	е	C	r	е	t	
		3						
0	b						5	6
1		4					4	5
2	S						3	4
3	е						2	3
4	С						1	2
5	t						0	1
6	•	6	5	4	3	2	1	0

- * Like LCS, ED(i,j)
 depends on
 ED(i+1,j+1), ED(i+1,j)
 and ED(i,j+1)
- * Dependencies for ED(m,n) are known
- * Start at ED(m,n) and fill by row, column or diagonal

		0	1	2	3	4	5	6
	2	S	е	С	r	е	t	•
0	b					5	5	6
1	1)	4				4	4	5
2	S					3	3	4
3	е	1				2	2	3
4	С					1	1	2
5	t					1	0	1
6	•	6	5	4	3	2	1	0

- * Like LCS, ED(i,j)
 depends on
 ED(i+1,j+1), ED(i+1,j)
 and ED(i,j+1)
- * Dependencies for ED(m,n) are known
- * Start at ED(m,n) and fill by row, column or diagonal

	1	0	1	2	3	4	5	6
6	5	S	е	С	r	е	t	•
0	b				5	5	5	6
1		/			4	4	4	5
2	S				3	3	3	4
3	е				2	2	2	3
4	С				2	1	1	2
5	t				2	1	0	1
6	•	6	5	4	3	2	1	0

- * Like LCS, ED(i,j)
 depends on
 ED(i+1,j+1), ED(i+1,j)
 and ED(i,j+1)
- * Dependencies for ED(m,n) are known
- * Start at ED(m,n) and fill by row, column or diagonal

		0	1	2	3	4	5	6
	2	S	е	C	r	е	t	•
0	b		1	5	5	5	5	6
1				4	4	4	4	5
2	S			3	3	3	3	4
3	е	1		3	2	2	2	3
4	С			2	2	1	1	2
5	t			3	2	1	0	1
6	•	6	5	4	3	2	1	0

- * Like LCS, ED(i,j)
 depends on
 ED(i+1,j+1), ED(i+1,j)
 and ED(i,j+1)
- * Dependencies for ED(m,n) are known
- * Start at ED(m,n) and fill by row, column or diagonal

		0	1	2	3	4	5	6
	5	S	е	С	r	е	t	•
0	b		5	5	5	5	5	6
1		4	4	4	4	4	4	5
2	S		3	3	3	3	3	4
3	е		2	3	2	2	2	3
4	С		3	2	2	1	1	2
5	t		4	3	2	1	0	1
6	•	6	5	4	3	2	1	0

- * Like LCS, ED(i,j)
 depends on
 ED(i+1,j+1), ED(i+1,j)
 and ED(i,j+1)
- * Dependencies for ED(m,n) are known
- * Start at ED(m,n) and fill by row, column or diagonal

				1				
		0	1	2	3	4	5	6
		S	е	C	r	е	t	•
0	b	4	5	5	5	5	5	6
1		3	4	4	4	4	4	5
2	S	2		3	3	3	3	4
3	е	3	2	3	2	2	2	3
4	С	4	3	2	2	1	1	2
5	t	5	4	3	2	1	0	1
6	•	6	5	4	3	2	1	0

Recovering the solution

- * Trace back the path
- * Transforming "secret" to "bisect"

* Ins "r":
$$(5,3) \rightarrow (5,4)$$

* Ins "e":
$$(5,4) \rightarrow (5,5)$$

		0	1	2	3	4	5	6
	2	S	е	C	r	е	t	·
0	b	4	5	5	5	5	5	6
1		3	4	4	4	4	4	5
2	S	2	3	3	3	3	3	4
3	е	3	2	3	2	2	2	3
4	С	4	3	2	2	1	1	2
5	t	5	4	3	2	1	-0	1
6	•	6	5	4	3	2	1	0

ED(u,v), DP

```
function ED(u,v) # u[0..m], v[0..n]
for r = 0, 1, ..., m+1  { ED[r][n+1] = m-r+1 }
for c = 0, 1, ..., m+1  { ED[m+1][c] = n-c+1 }
for c = n, n-1, ..., 0
  for r = m, m-1, ...0
    if (u[r] == v[c])
      ED[r][c] = ED[r+1][c+1]
    else
      ED[r][c] = 1 + min(ED[r+1][c+1],
                            ED[r+1][c],
                            ED[r][c+1])
```

return(ED[0][0])

Complexity

- * Again O(mn) using dynamic programming (or memoization)
 - * Need to fill an O(mn) size table
 - * Each table entry takes constant time to compute

Space complexity

- * For LCW, LCS, ED
 - * Need to fill an O(mn) size table
 - * Do we need to store the entire table?
- * Filling column by column, only need next column and current column
 - * Or next row and current row
- * Reduce space to O(n), assuming m ≥ n