CSE 417: Algorithms and Computational Complexity

Dynamic Programming, III

Autumn 2002
Paul Beame

1

Three Steps to Dynamic Programming

- Formulate the answer as a recurrence relation or recursive algorithm
- Show that the number of different parameters in the recursive algorithm is "small"
  - e.g., bounded by a low-degree polynomial
- Specify an order of evaluation for the recurrence so that you already have the partial results ready when you need them.

2

Sequence Comparison: Edit Distance

- Given:
  - Two strings of characters **$A=a_1\ a_2\ ...\ a_n$** and **$B=b_1\ b_2\ ...\ b_m$**
- Find:
  - The minimum number of edit steps needed to transform **A** into **B** where an edit can be:
  - insert a single character
  - delete a single character
  - substitute one character by another

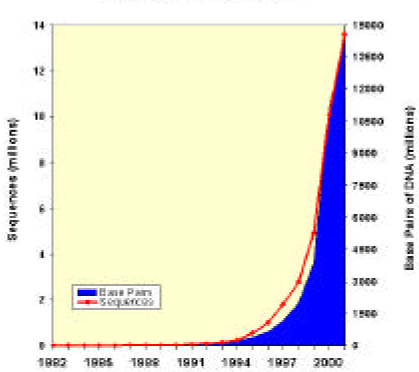
3

Applications

- "diff" utility – where do two files differ
- Version control & patch distribution – save/send only changes
- Molecular biology
  - Similar sequences often have similar origin and function
  - Similarity often recognizable despite millions or billions of years of evolutionary divergence

4

Growth of GenBank



Recursive Solution

- Sub-problems: Edit distance problems for **all prefixes** of **A** and **B** that don't include all of both **A** and **B**
- Let **D(i,j)** be the number of edits required to transform **$a_1\ a_2\ ...\ a_i$** into **$b_1\ b_2\ ...\ b_j$**
- Clearly **D(0,0)=0**

6

Computing D(n,m)

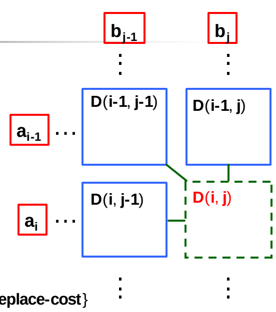
- Imagine how best sequence handles the last characters a_n and b_m
- If best sequence of operations
  - deletes a_n then $D(n,m)=D(n-1,m)+1$
  - inserts b_m then $D(n,m)=D(n,m-1)+1$
  - replaces a_n by b_m then $D(n,m)=D(n-1,m-1)+1$
  - matches a_n and b_m then $D(n,m)=D(n-1,m-1)$

Recursive algorithm D(n,m)

```
if n=0 then
    return (m)
elseif m=0 then
    return(n)
else
    if a_n=b_m then
        replace-cost ← 0
    else                          } cost of substitution of a_n by b_m (if used)
        replace-cost ← 1
    endif
    return(min{ D(n-1, m) + 1,
                D(n, m-1) +1,
                D(n-1, m-1) + replace-cost})
```


Dynamic Programming

```
for j = 0 to m;  D(0,j) ← j; endfor
for i = 1 to n;  D(i,0) ← i; endfor
for i = 1 to n
    for j = 1 to m
        if a_i=b_j then
            replace-cost ← 0
        else
            replace-cost ← 1
        endif
        D(i,j) ← min { D(i-1, j) + 1,
                       D(i, j-1) + 1,
                       D(i-1, j-1) + replace-cost}
    endfor
endfor
```



Example run with AGACATTG and GAGTTA

	A	G	A	C	A	T	T	G
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0								
G 1								
A 2								
G 3								
T 4								
T 5								
A 6								

Example run with AGACATTG and GAGTTA

		A	G	A	C	A	T	T	G
	0	1	2	3	4	5	6	7	8
G	1	1	1	2	3	4	5	6	7
A	2								
G	3								
T	4								
T	5								
A	6								

Example run with AGACATTG and GAGTTA

		A	G	A	C	A	T	T	G
	0	1	2	3	4	5	6	7	8
G	1	1	1	2	3	4	5	6	7
A	2	1	2	1					
G	3								
T	4								
T	5								
A	6								

Example run with AGACATTG and GAGTTA

		A	G	A	C	A	T	T	G
	0	1	2	3	4	5	6	7	8
G	1	1	1	2	3	4	5	6	7
A	2	1	2	1	2	3	4	5	6
G	3	2	1	2	2	3	4	5	5
T	4								
T	5								
A	6								

Example run with AGACATTG and GAGTTA

		A	G	A	C	A	T	T	G
	0	1	2	3	4	5	6	7	8
G	1	1	1	2	3	4	5	6	7
A	2	1	2	1	2	3	4	5	6
G	3	2	1	2	2	3	4	5	5
T	4	3	2	2	3	3	3	4	5
T	5	4	3	3	3	4	3	3	4
A	6	5	4	3	4	3	4	4	4

Example run with AGACATTG and GAGTTA

		A	G	A	C	A	T	T	G
	0	1	2	3	4	5	6	7	8
G	1	1	1	2	3	4	5	6	7
A	2	1	2	1	2	3	4	5	6
G	3	2	1	2	2	3	4	5	5
T	4	3	2	2	3	3	3	4	5
T	5	4	3	3	3	4	3	3	4
A	6	5	4	3	4	3	4	4	4

Example run with AGACATTG and GAGTTA

		A	G	A	C	A	T	T	G
	0	**1**	2	3	4	5	6	7	8
G	1	1	**1**	2	3	4	5	6	7
A	2	1	2	**1**	2	3	4	5	6
G	3	2	1	2	**2**	**3**	4	5	5
T	4	3	2	2	3	3	**3**	4	5
T	5	4	3	3	3	4	3	**3**	4
A	6	5	4	3	4	3	4	4	**4**

Reading off the operations

- Follow the sequence and use each color of arrow to tell you what operation was performed.