

CSE 589 Applied Algorithms Spring 1999

Approximate String Matching
Contiguous Ordering - PQ Trees

DNA

- DNA is a large molecule that can be abstractly defined as a sequence of symbols from the set, A, C, G, T, called nucleotides.
- The human genome has about 3 billion nucleotides.
 - A huge percentage of the genome is shared by all humans.
 - Some of the variation makes us different.
 - Some of the variation is inconsequential.
 - The human genome is still being discovered.

CSE 589 - Lecture 18 - Spring 1999

2

Approximate Matching

- Two DNA sequences approximately match if one can be transformed into the other by a short sequence of replacements and insertions of gaps.
- Example:
 - S = AGCATG
 - T = AGATCGT
- Approximate matching - is a gap
 - S' = A G - - C A T G
 - T' = A G A T C G T -

CSE 589 - Lecture 18 - Spring 1999

3

Applications of Approximate Matching

- DNA string alignment.
 - Given two similar DNA sequences find the best way to align them to the same length.
- DNA database searching.
 - Find DNA sequences that are similar to the query.
- Approximate text matching for searching.
 - agrep in unix
- Spell checking
 - Find the words that most closely match the misspelled word.

CSE 589 - Lecture 18 - Spring 1999

4

Scoring an Approximate Matching

- We need a way of scoring the quality of an approximate matching.
- A scoring function is a mapping σ from $\{A, C, G, T, -\}^2$ to integers.
 - The quantity $\sigma(x,y)$ is the score of a pair of symbols, x and y.
- Example:
 - $\sigma(x,y) = +2$ if $x=y$ and x in $\{A,C,G,T\}$
 - $\sigma(x,y) = -1$ otherwise

CSE 589 - Lecture 18 - Spring 1999

5

Scoring Example

- Example:
 - S' = A G - - C A T G
 - T' = A G A T C G T -
- Score = $4 \times 2 + 4 \times (-1) = 4$
- Is this the best match between the two strings with this scoring function?
 - S = AGCATG
 - T = AGATCGT

CSE 589 - Lecture 18 - Spring 1999

6

Approximate String Matching Problem

- Input: Two strings S and T in an alphabet Σ and a scoring function σ .
- Output: Two strings S' and T' in the alphabet $\Sigma' = \Sigma$ union $\{-\}$ with the properties:
 - S = S' with the '-'s removed.
 - T = T' with the '-'s removed.
 - |S'| = |T'|
- The score $\sum_{i=1}^{|S'|} \sigma(S'[i], T'[i])$ is maximized.

CSE 589 - Lecture 18 - Spring 1999

7

Algorithms for Approximate String Matching

- $O(mn)$ time and storage algorithm (using dynamic programming) invented by Needleman and Wunsch, 1970.
- Fischer and Paterson, 1974, invented a very similar algorithm for computing the minimum edit distance between two strings.

CSE 589 - Lecture 18 - Spring 1999

8

Dynamic Programming for Approximate String Matching

- Assume S has length m and T has length n.
- For all i and j, $0 \leq i \leq m$ and $0 \leq j \leq n$, we find the maximum score for the sequences S[1..i] and T[1..j].
- The "dynamic program" fills in a $(m+1) \times (n+1)$ matrix M in increasing order of i and j with these maximum values.
- Once the dynamic program has completed we can recover the optimal string S' and T' from the matrix M.

CSE 589 - Lecture 18 - Spring 1999

9

Max Score Recurrence

- Define $M[i, j]$ = maximum score for a match between S[1..i] and T[1..j].

$$M[i, 0] = \sum_{k=1}^i \sigma(S[k], -) \quad \text{match of S[1..i] with empty string}$$

$$M[0, j] = \sum_{k=1}^j \sigma(-, T[k]) \quad \text{match of T[1..j] with empty string}$$

$$M[i, j] = \max \{ \begin{aligned} &M[i-1, j-1] + \sigma(S[i], T[j]), \\ &M[i-1, j] + \sigma(S[i], -), \\ &M[i, j-1] + \sigma(-, T[j]) \end{aligned} \}$$

CSE 589 - Lecture 18 - Spring 1999

10

Dynamic Program Initialization

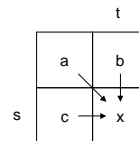
S = AGCATG scoring function
T = AGATCGT +2 for exact match
 -1 otherwise

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

CSE 589 - Lecture 18 - Spring 1999

11

The Dynamic Programming Pattern



$$d = a + 2 \text{ if } s = t \\ = a - 1 \text{ otherwise}$$

$$h = c - 1$$

$$v = b - 1$$

$$x = \max(d, h, v)$$

CSE 589 - Lecture 18 - Spring 1999

12

Dynamic Program Example (1)

S = AGCATG scoring function
T = AGATCGT +2 for exact match
 -1 otherwise

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

CSE 589 - Lecture 18 - Spring 1999

13

Dynamic Program Example (2)

S = AGCATG scoring function
T = AGATCGT +2 for exact match
 -1 otherwise

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

CSE 589 - Lecture 18 - Spring 1999

14

Dynamic Program Example (3)

S = AGCATG scoring function
T = AGATCGT +2 for exact match
 -1 otherwise

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

CSE 589 - Lecture 18 - Spring 1999

15

Dynamic Program Example (4)

S = AGCATG scoring function
T = AGATCGT +2 for exact match
 -1 otherwise

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

CSE 589 - Lecture 18 - Spring 1999

16

Dynamic Program Example (5)

S = AGCATG scoring function
T = AGATCGT +2 for exact match
 -1 otherwise

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

Max score for any matching

CSE 589 - Lecture 18 - Spring 1999

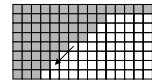
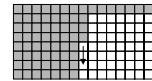
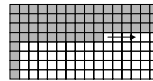
17

Dynamic Programming Order

By row
for i = 1 to m do
 for j = 1 to n do
 M[i,j] := ...

By column
for j = 1 to n do
 for i = 1 to m do
 M[i,j] := ...

By diagonal



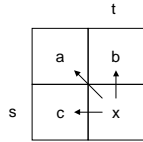
Which order is best?

CSE 589 - Lecture 18 - Spring 1999

18

How to Find the Matching

- To find S' and T' we build a matching graph.



$$x = a + 2 \text{ if } s = t$$

$$= a - 1 \text{ otherwise ?}$$

$$x = c - 1 ?$$

$$x = b - 1 ?$$

If the answer is yes, include the corresponding edge.

Computing the Matching Graph (1)

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

Computing the Matching Graph (2)

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

Computing the Matching Graph (3)

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

Computing the Matching Graph

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

Computing the Matching Path

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

Matching Path
 (0,0)
 (1,1)
 (2,2)
 (3,2)
 (4,3)
 (5,4)
 (5,5)
 (6,6)
 (6,7)

There can be multiple paths

Algorithm to find Matching

- Follow any path in the matching graph starting at (m,n).
- The path will end up at (0,0).
- Output each pair (i,j) visited to make a list of pairs forming a matching path.

Computing the Matching

```

p = length of the matching path P
i := 1;
j := 1;
for k = 1 to p do
  if P[k].first = P[k-1].first then
    S'[k] := -;
  else
    S'[k] := S[i];
    i := i + 1;
  if P[k].second = P[k-1].second then
    T'[k] := -;
  else
    T'[k] := T[j];
    j := j + 1;
  
```

P	0	(0,0)
	1	(1,1)
	2	(2,2)
	3	(3,2)
	4	(4,3)
	5	(5,4)
	6	(5,5)
	7	(6,6)
	8	(6,7)

↑ ↑
first second

Creating the Matching

P	0	(0,0)
	1	(1,1)
	2	(2,2)
	3	(3,2)
	4	(4,3)
	5	(5,4)
	6	(5,5)
	7	(6,6)
	8	(6,7)

S = A G C A T G
T = A G A T C G T

S' = A G C A T - G -
T' = A G - A T C G T

$$\text{Score} = 5 \times 2 + 3 \times (-1) = 7$$

Example of Multiple Paths

		0	1	2	3	4	5
			C	A	T	G	T
0		0	-1	-2	-3	-4	-5
1	A	-1	-1	1	0	-1	-2
2	C	-2	1	0	0	-1	-2
3	G	-3	0	0	-1	2	1
4	C	-4	-1	-1	-1	1	1
5	T	-5	-2	-2	1	0	3
6	G	-6	-3	-3	0	3	-2

Multiple matching with same score

- A C G C T G
C A T G - T -
A C G C T G -
- C A - T G T
A C G C T G -
- - C A T G T

$$\text{score} = 3 \times 2 + 4 \times (-1) = 2$$

Approximate String Searching

- Input: Query string Q and target string T in an alphabet Σ and a scoring function σ , and a minimum score r.
- Output: The set of k such that for some $i \leq k$ $\text{score}(Q, T[i..k]) \geq r$. That is, an approximate match of some substring of T that ends at index k has a score of at least r.
 - score(X,Y) is the maximum score for all matchings between X and Y.

Search Algorithm

- We change the previous dynamic program slightly.

$$M[i,0] = \sum_{k=1}^i \sigma(Q[k], -)$$

$$M[0,j] = 0 \quad \text{We don't care where the match begins in T}$$

$$M[i,j] = \max\{$$

$$M[i-1, j-1] + \sigma(Q[i], T[j]),$$

$$M[i-1, j] + \sigma(Q[i], -),$$

$$M[i, j-1] + \sigma(-, T[j])\}$$

Choose all k such that $M[m,k] \geq r$ where m is the length of Q.

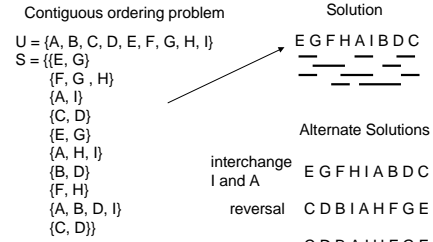
Contiguous Ordering Problem

- Input: A universe U and a set $S = \{S_1, S_2, \dots, S_m\}$ of subsets of U .
- Output: An ordering, b_1, b_2, \dots, b_n , of U such that for all i , $S_i = \{b_j, b_{j+1}, \dots, b_{j+k}\}$ for some j and k . That is each set in S is contiguous in the ordering of U .
- In this terminology the U is the set of tags and S is the set of tagged clones.

CSE 589 - Lecture 18 - Spring 1999

37

Contiguous Ordering Solutions



CSE 589 - Lecture 18 - Spring 1999

38

Linear Time Algorithm

- Booth and Lueker, 1976, designed an algorithm that runs in time $O(n+s)$.
 - n is the size of the universe and s is the sum of the sizes of the sets.
- It requires a novel data structure called the PQ tree that represents a set of orderings.
- PQ trees can also be used to test whether an undirected graph is planar.

CSE 589 - Lecture 18 - Spring 1999

39

DNA Downside

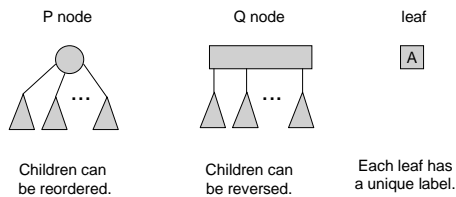
- Tagging can have errors.
 - A tag might be omitted from a clone
 - A clone might be identified as having a tag when it doesn't
 - A tag might be duplicated, not be unique.
- In all these cases finding the "best" ordering of the tags in an NP-hard problem.
 - In our perfect world without errors, ordering can be done in linear time.

CSE 589 - Lecture 18 - Spring 1999

40

PQ Trees

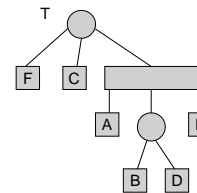
- PQ trees are built from three types of nodes



CSE 589 - Lecture 18 - Spring 1999

41

Example PQ-Tree



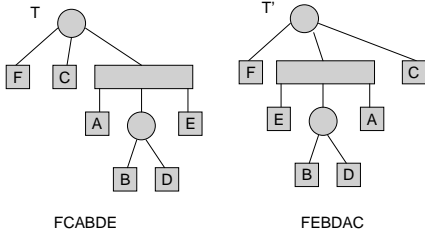
The frontier of T defines the ordering $F(T) = FCABDE$, just read the leaves left to right.

T' is equivalent to T if T' can be transformed into T by reordering the children of P nodes and reversing the children of Q nodes.

CSE 589 - Lecture 18 - Spring 1999

42

Equivalent PQ Trees

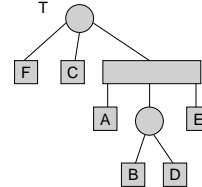


CSE 589 - Lecture 18 - Spring 1999

43

Orderings Defined by a PQ Tree

- Given a PQ tree T the orderings defined by T is
 - $PQ(T) = \{F(T') : T' \text{ is equivalent to } T\}$



There are $6 \times 2 \times 2 = 24$ distinct orderings in $PQ(T)$.

CSE 589 - Lecture 18 - Spring 1999

44

PQ Tree Solution for the Contiguous Ordering Problem

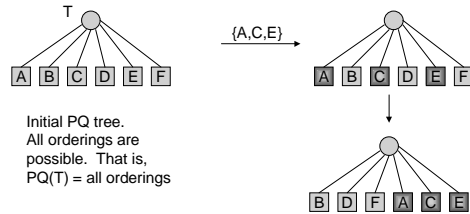
- Input: A universe U and a set $S = \{S_1, S_2, \dots, S_m\}$ of subsets of U .
- Output: A PQ tree T with leaves U with the property that $PQ(T)$ is the set of all orderings of U where each set in S is contiguous in the ordering.

CSE 589 - Lecture 18 - Spring 1999

45

Example (1)

$U = \{A, B, C, D, E, F\}$
 $S = \{\{A, C, E\}, \{A, C, F\}, \{B, D, E\}\}$



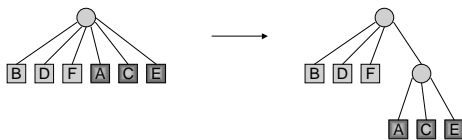
Initial PQ tree.
 All orderings are possible. That is,
 $PQ(T) = \text{all orderings}$

CSE 589 - Lecture 18 - Spring 1999

46

Example (2)

$U = \{A, B, C, D, E, F\}$
 $S = \{\{A, C, E\}, \{A, C, F\}, \{B, D, E\}\}$

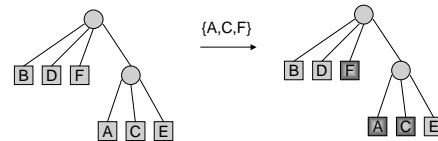


CSE 589 - Lecture 18 - Spring 1999

47

Example (3)

$U = \{A, B, C, D, E, F\}$
 $S = \{\{A, C, E\}, \{A, C, F\}, \{B, D, E\}\}$

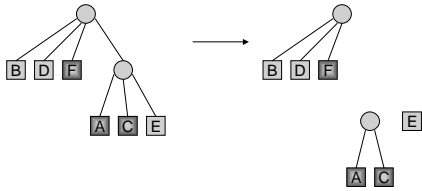


CSE 589 - Lecture 18 - Spring 1999

48

Example (4)

$U = \{A, B, C, D, E, F\}$
 $S = \{\{A, C, E\}, \{A, C, F\}, \{B, D, E\}\}$

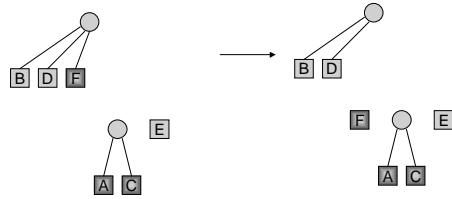


CSE 589 - Lecture 18 - Spring 1999

49

Example (5)

$U = \{A, B, C, D, E, F\}$
 $S = \{\{A, C, E\}, \{A, C, F\}, \{B, D, E\}\}$

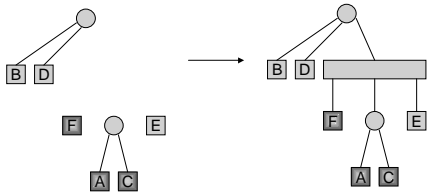


CSE 589 - Lecture 18 - Spring 1999

50

Example (6)

$U = \{A, B, C, D, E, F\}$
 $S = \{\{A, C, E\}, \{A, C, F\}, \{B, D, E\}\}$

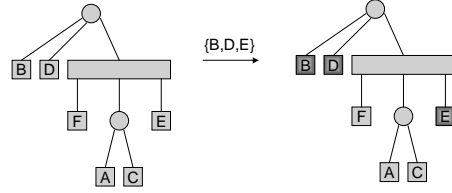


CSE 589 - Lecture 18 - Spring 1999

51

Example (7)

$U = \{A, B, C, D, E, F\}$
 $S = \{\{A, C, E\}, \{A, C, F\}, \{B, D, E\}\}$

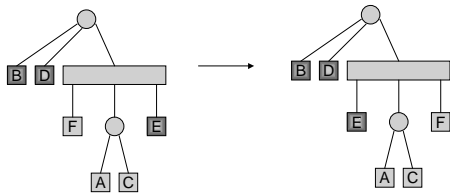


CSE 589 - Lecture 18 - Spring 1999

52

Example (8)

$U = \{A, B, C, D, E, F\}$
 $S = \{\{A, C, E\}, \{A, C, F\}, \{B, D, E\}\}$

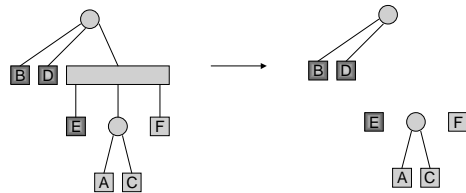


CSE 589 - Lecture 18 - Spring 1999

53

Example (9)

$U = \{A, B, C, D, E, F\}$
 $S = \{\{A, C, E\}, \{A, C, F\}, \{B, D, E\}\}$

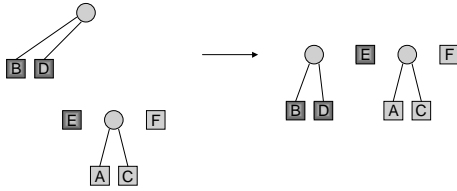


CSE 589 - Lecture 18 - Spring 1999

54

Example (10)

U = {A,B,C,D,E,F}
 S = {{A,C,E}, {A,C,F}, {B,D,E}}

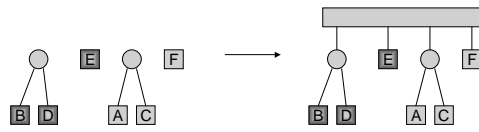


CSE 589 - Lecture 18 - Spring 1999

55

Example (11)

U = {A,B,C,D,E,F}
 S = {{A,C,E}, {A,C,F}, {B,D,E}}

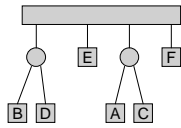


CSE 589 - Lecture 18 - Spring 1999

56

Example (12)

U = {A,B,C,D,E,F}
 S = {{A,C,E}, {A,C,F}, {B,D,E}}



There are 8 orderings that are possible in keeping each of these sets contiguous.

CSE 589 - Lecture 18 - Spring 1999

57