Computing the Length of Constraint Longest Common Subsequence

Bai Yu   Lie Yan   Xiaolei Wang

Mathematical and Computer Sciences and Engineering Division
Zhejiang University, China

January 6, 2012

Outline

Brief Introduction to the Problem

Definition (CLCS)

The `Constraint Longest Common Subsequence (CLCS)` problem is to find the longest subsequence common to all the given strings with respect to the constraint sequence. Thus, given two strings X, Y, and a contraint string P, find the longest common subsequence Z of X and Y such that P is a subsequence of Z.

Brief Introduction to the Problem

Definition (**CLCS**)

The `Constraint Longest Common Subsequence (CLCS)` problem is to find the longest subsequence common to all the given strings with respect to the constraint sequence. Thus, given two strings X, Y, and a contraint string P, find the longest common subsequence Z of X and Y such that P is a subsequence of Z.

Example

For example, given input $X = "abc123"$, $Y = "123abc"$, and $P = "ab"$, the LCS of X and Y is $"abc"$ and $"123"$, while the CLCS with respect to P is $"abc"$.

Algorithms Description

- Algorithms
  We use the algorithms by *Yin-Te Tsai* and by *Francis Chin, et al.* to solve this problem.

Algorithms Description

- Algorithms
  We use the algorithms by *Yin-Te Tsai* and by *Francis Chin, et al.* to solve this problem.

- Time complexity
  Assume the lengths of X, Y and P be n, m and r respectively.
  - Tsai's algorithm solves the problem in $O(n^2m^2r)$ time.
  - Chin's algorithm in $O(nmr)$ time.

Algorithms Description

- Algorithms
  We use the algorithms by *Yin-Te Tsai* and by *Francis Chin, et al.* to solve this problem.

- Time complexity
  Assume the lengths of X, Y and P be n, m and r respectively.
  - Tsai's algorithm solves the problem in $O(n^2m^2r)$ time.
  - Chin's algorithm in $O(nmr)$ time.

- Technique
  Dynamic Programming

Algorithm by Yin-Te Tsai

Let $L(x, y, x', y')$ denotes the length of LCS of strings $X[x..x']$ and $Y[y..y']$.
Let $L_k(i, j)$ denotes the length of CLCS of strings $X[1..i]$ and $Y[1..j]$ w.r.t $P[1..k]$.

Algorithm by Yin-Te Tsai

Let $L(x, y, x', y')$ denotes the length of LCS of strings $X[x..x']$ and $Y[y..y']$.
Let $L_k(i, j)$ denotes the length of CLCS of strings $X[1..i]$ and $Y[1..j]$ w.r.t $P[1..k]$.

Recursive Relation.

For $1 \le i \le n$, $1 \le j \le m$ and $1 \le k \le r$, $L_1(i, j) = L(1, 1, i-1, j-1) + 1$ if $X[i] = Y[j] = P[1]$, and $-\infty$ otherwise.
For $2 \le k \le r$, $1 \le i \le n$, and $1 \le j \le m$,

$$L_k(i,j) = \begin{cases} \max\{L_{k-1}(x,y), L(x+1, y+1, i-1, j-1)+1\} & \text{if} X[i] = Y[j] = P[k]; \\ -\infty & \text{otherwise} \end{cases} \tag{1}$$

□

Algorithm by Yin-Te Tsai

Let $L(x, y, x', y')$ denotes the length of LCS of strings $X[x..x']$ and $Y[y..y']$.
Let $L_k(i, j)$ denotes the length of CLCS of strings $X[1..i]$ and $Y[1..j]$ w.r.t $P[1..k]$.

Recursive Relation.

For $1 \leq i \leq n$, $1 \leq j \leq m$ and $1 \leq k \leq r$, $L_1(i, j) = L(1, 1, i-1, j-1) + 1$ if $X[i] = Y[j] = P[1]$, and $-\infty$ otherwise.
For $2 \leq k \leq r$, $1 \leq i \leq n$, and $1 \leq j \leq m$,

$$L_k(i,j) = \begin{cases} \max\{L_{k-1}(x,y), L(x+1, y+1, i-1, j-1)+1\} & \text{if} X[i] = Y[j] = P[k]; \\ -\infty & \text{otherwise} \end{cases} \quad (1)$$

$\square$

After obtaining $L_k(\cdot, \cdot)$, we compute the length of CLCS with the following formula:

$$|Z| = \max_{1 \leq i \leq n, 1 \leq j \leq m} \{L_r(i,j) + L(i+1, j+1, n, m)\} \quad (2)$$

Algorithm by Yin-Te Tsai

Algorithm by Yin-Te Tsai

Time:

Algorithm by Yin-Te Tsai

Time:

- Pre-computation of L: $O(n^2 m^2)$

Algorithm by Yin-Te Tsai

Time:

- Pre-computation of L: $O(n^2m^2)$
- Compute all the L_k: $O(n^2m^2r)$

Algorithm by Yin-Te Tsai

Time:

- Pre-computation of L: $O(n^2 m^2)$
- Compute all the L_k: $O(n^2 m^2 r)$
- Compute $|Z|$ from L_k to L: $O(nm)$

Algorithm by Yin-Te Tsai

Time:

- Pre-computation of L: $O(n^2m^2)$
- Compute all the L_k: $O(n^2m^2r)$
- Compute $|Z|$ from L_k to L: $O(nm)$
- Total: $O(n^2m^2) + O(n^2m^2r) + O(nm) = O(n^2m^2r)$

Algorithm by Yin-Te Tsai

Time:

- Pre-computation of L: $O(n^2m^2)$
- Compute all the L_k: $O(n^2m^2r)$
- Compute $|Z|$ from L_k to L: $O(nm)$
- Total: $O(n^2m^2) + O(n^2m^2r) + O(nm) = O(n^2m^2r)$

Space:

Algorithm by Yin-Te Tsai

Time:

- Pre-computation of L: $O(n^2 m^2)$
- Compute all the L_k: $O(n^2 m^2 r)$
- Compute $|Z|$ from L_k to L: $O(nm)$
- Total: $O(n^2 m^2) + O(n^2 m^2 r) + O(nm) = O(n^2 m^2 r)$

Space:

- Space for $L(.,.,.,.)$: $O(n^2 m^2)$

Algorithm by Yin-Te Tsai

Time:

- Pre-computation of L: $O(n^2 m^2)$
- Compute all the L_k: $O(n^2 m^2 r)$
- Compute $|Z|$ from L_k to L: $O(nm)$
- Total: $O(n^2 m^2) + O(n^2 m^2 r) + O(nm) = O(n^2 m^2 r)$

Space:

- Space for $L(.,.,.,.)$: $O(n^2 m^2)$
- Space for $L_k(.,.)$: $O(nmr)$

Algorithm by Yin-Te Tsai

Time:

- Pre-computation of L: $O(n^2m^2)$
- Compute all the L_k: $O(n^2m^2r)$
- Compute $|Z|$ from L_k to L: $O(nm)$
- Total: $O(n^2m^2) + O(n^2m^2r) + O(nm) = O(n^2m^2r)$

Space:

- Space for $L(.,.,.,.)$: $O(n^2m^2)$
- Space for $L_k(.,.)$: $O(nmr)$
- Total: $O(n^2m^2) + O(nmr) = O(n^2m^2)$

Algorithm by Yin-Te Tsai

Example: $X =$ "aba", $Y =$ "bab", $P =$ "a".

$M_{1,1}$		b	a	b
	0	0	0	0
a	0	0	1	1
b	0	1	1	2
a	0	1	2	2

$M_{1,2}$		a	b
	0	0	0
a	0	1	1
b	0	1	2
a	0	1	2

$M_{1,3}$		b
	0	0
a	0	0
b	0	1
a	0	1

$M_{2,1}$		b	a	b
	0	0	0	0
b	0	1	1	1
a	0	1	2	2

$M_{2,2}$		a	b
	0	0	0
b	0	0	1
a	0	1	1

$M_{2,3}$		b
	0	0
b	0	1
a	0	1

$M_{3,1}$		b	a	b
	0	0	0	0
a	0	0	1	1

$M_{3,2}$		a	b
	0	0	0
a	0	1	1

$M_{3,3}$		b
	0	0
a	0	0

Algorithm by Yin-Te Tsai

Example: $X =$ "aba", $Y =$ "bab", $P =$ "a".
Then we compute states $L_k(i, j)$.

Algorithm by Yin-Te Tsai

Example: $X =$ "aba", $Y =$ "bab", $P =$ "a".
Then we compute states $L_k(i,j)$.

1. $(X[1] \neq Y[1])$ $L_1(1,1) = -\infty$;

Algorithm by Yin-Te Tsai

Example: $X =$ "aba", $Y =$ "bab", $P =$ "a".
Then we compute states $L_k(i, j)$.

1. $(X[1] \neq Y[1])$ $L_1(1,1) = -\infty$;
2. $(X[1] = Y[2] = P[1])$ $L_1(1,2) = L(1,1,0,1) + 1 = 1$;

Algorithm by Yin-Te Tsai

Example: $X =$ "aba", $Y =$ "bab", $P =$ "a".
Then we compute states $L_k(i,j)$.

1. $(X[1] \neq Y[1])$ $L_1(1,1) = -\infty$;
2. $(X[1] = Y[2] = P[1])$ $L_1(1,2) = L(1,1,0,1) + 1 = 1$;
3. $(X[1] \neq Y[3])$ $L_1(1,3) = -\infty$;

Algorithm by Yin-Te Tsai

Example: $X =$ "aba", $Y =$ "bab", $P =$ "a".
Then we compute states $L_k(i, j)$.

1. $(X[1] \neq Y[1])$ $L_1(1, 1) = -\infty$;
2. $(X[1] = Y[2] = P[1])$ $L_1(1, 2) = L(1, 1, 0, 1) + 1 = 1$;
3. $(X[1] \neq Y[3])$ $L_1(1, 3) = -\infty$;
4. $(X[2] = Y[1] \neq P[1])$ $L_1(2, 1) = -\infty$;

Algorithm by Yin-Te Tsai

Example: $X =$ "aba", $Y =$ "bab", $P =$ "a".
Then we compute states $L_k(i,j)$.

1. $(X[1] \neq Y[1])$ $L_1(1,1) = -\infty$;
2. $(X[1] = Y[2] = P[1])$ $L_1(1,2) = L(1,1,0,1) + 1 = 1$;
3. $(X[1] \neq Y[3])$ $L_1(1,3) = -\infty$;
4. $(X[2] = Y[1] \neq P[1])$ $L_1(2,1) = -\infty$;
5. $(X[2] \neq Y[2])$ $L_1(2,2) = -\infty$;

Algorithm by Yin-Te Tsai

Example: $X =$ "aba", $Y =$ "bab", $P =$ "a".
Then we compute states $L_k(i, j)$.

1. $(X[1] \neq Y[1])$ $L_1(1, 1) = -\infty$;
2. $(X[1] = Y[2] = P[1])$ $L_1(1, 2) = L(1, 1, 0, 1) + 1 = 1$;
3. $(X[1] \neq Y[3])$ $L_1(1, 3) = -\infty$;
4. $(X[2] = Y[1] \neq P[1])$ $L_1(2, 1) = -\infty$;
5. $(X[2] \neq Y[2])$ $L_1(2, 2) = -\infty$;
6. $(X[2] = Y[3] \neq P[1])$ $L_1(2, 3) = -\infty$;

Algorithm by Yin-Te Tsai

Example: $X =$ "aba", $Y =$ "bab", $P =$ "a".
Then we compute states $L_k(i,j)$.

1. $(X[1] \neq Y[1])$ $L_1(1,1) = -\infty$;
2. $(X[1] = Y[2] = P[1])$ $L_1(1,2) = L(1,1,0,1) + 1 = 1$;
3. $(X[1] \neq Y[3])$ $L_1(1,3) = -\infty$;
4. $(X[2] = Y[1] \neq P[1])$ $L_1(2,1) = -\infty$;
5. $(X[2] \neq Y[2])$ $L_1(2,2) = -\infty$;
6. $(X[2] = Y[3] \neq P[1])$ $L_1(2,3) = -\infty$;
7. $(X[3] \neq Y[1])$ $L_1(3,1) = -\infty$;

Algorithm by Yin-Te Tsai

Example: X = "aba", Y = "bab", P = "a".
Then we compute states $L_k(i,j)$.

1. $(X[1] \neq Y[1])$ $L_1(1,1) = -\infty$;
2. $(X[1] = Y[2] = P[1])$ $L_1(1,2) = L(1,1,0,1) + 1 = 1$;
3. $(X[1] \neq Y[3])$ $L_1(1,3) = -\infty$;
4. $(X[2] = Y[1] \neq P[1])$ $L_1(2,1) = -\infty$;
5. $(X[2] \neq Y[2])$ $L_1(2,2) = -\infty$;
6. $(X[2] = Y[3] \neq P[1])$ $L_1(2,3) = -\infty$;
7. $(X[3] \neq Y[1])$ $L_1(3,1) = -\infty$;
8. $(X[3] = Y[2] = P[1])$ $L_1(3,2) = L(1,1,2,1) + 1 = 2$;

Algorithm by Yin-Te Tsai

Example: $X =$ "aba", $Y =$ "bab", $P =$ "a".
Then we compute states $L_k(i, j)$.

1. $(X[1] \neq Y[1])$ $L_1(1,1) = -\infty$;
2. $(X[1] = Y[2] = P[1])$ $L_1(1,2) = L(1,1,0,1) + 1 = 1$;
3. $(X[1] \neq Y[3])$ $L_1(1,3) = -\infty$;
4. $(X[2] = Y[1] \neq P[1])$ $L_1(2,1) = -\infty$;
5. $(X[2] \neq Y[2])$ $L_1(2,2) = -\infty$;
6. $(X[2] = Y[3] \neq P[1])$ $L_1(2,3) = -\infty$;
7. $(X[3] \neq Y[1])$ $L_1(3,1) = -\infty$;
8. $(X[3] = Y[2] = P[1])$ $L_1(3,2) = L(1,1,2,1) + 1 = 2$;
9. $(X[3] \neq Y[3])$ $L_1(3,3) = -\infty$.

Algorithm by Yin-Te Tsai

Example: $X =$ "*aba*", $Y =$ "*bab*", $P =$ "*a*".
Then we compute states $L_k(i,j)$.

1. $(X[1] \neq Y[1])$ $L_1(1,1) = -\infty$;
2. $(X[1] = Y[2] = P[1])$ $L_1(1,2) = L(1,1,0,1) + 1 = 1$;
3. $(X[1] \neq Y[3])$ $L_1(1,3) = -\infty$;
4. $(X[2] = Y[1] \neq P[1])$ $L_1(2,1) = -\infty$;
5. $(X[2] \neq Y[2])$ $L_1(2,2) = -\infty$;
6. $(X[2] = Y[3] \neq P[1])$ $L_1(2,3) = -\infty$;
7. $(X[3] \neq Y[1])$ $L_1(3,1) = -\infty$;
8. $(X[3] = Y[2] = P[1])$ $L_1(3,2) = L(1,1,2,1) + 1 = 2$;
9. $(X[3] \neq Y[3])$ $L_1(3,3) = -\infty$.

Finally, we compute the length of CLCS:

$$|Z| = \max_{1 \leq i \leq 3, 1 \leq j \leq 3} \{L_1(i,j) + L(i+1, j+1, 3, 3)\} = 2$$

Algorithm by Yin-Te Tsai

Algorithm by Francis Chin et al.

Given two strings X, Y of length n, m respectively, and a contraint string P of length r, we define $L(i, j, k)$ as the CLCS length of $X[1..i]$, $Y[1..j]$ w.r.t $P[1..k]$.

Algorithm by Francis Chin et al.

Given two strings X, Y of length n, m respectively, and a contraint string P of length r, we define $L(i,j,k)$ as the CLCS length of $X[1..i]$, $Y[1..j]$ w.r.t $P[1..k]$.

Recursive Relation.

For any $0 \le i \le n$, $0 \le j \le m$ and $0 \le k \le r$:

$$L(i,j,k) = \begin{cases} 1 + L(i-1, j-1, k-1) & \text{if } i,j,k > 0 \text{ and } X[i] = Y[j] = P[k] \\ 1 + L(i-1, j-1, k) & \text{if } i,j > 0, X[i] = Y[j] \text{ and } (k = 0 \text{ or } X[i] \neq P[k]) \\ \max(L(i-1,j,k), L(i,j-1,k)) & \text{if } i,j > 0 \text{ and } X[i] \neq Y[j] \end{cases} \tag{3}$$

with boundary conditions, $L(i,0,0) = L(0,j,0) = 0$ and $L(0,j,k) = L(i,0,k) = -\infty$ for any $0 \le i \le n$, $0 \le j \le m$ and $0 < k \le r$. □

Algorithm by Francis Chin et al.

Given two strings X, Y of length n, m respectively, and a contraint string P of length r, we define $L(i,j,k)$ as the CLCS length of $X[1..i]$, $Y[1..j]$ w.r.t $P[1..k]$.

Recursive Relation.

For any $0 \le i \le n$, $0 \le j \le m$ and $0 \le k \le r$:

$$L(i,j,k) = \begin{cases} 1 + L(i-1, j-1, k-1) & \text{if } i,j,k > 0 \text{ and } X[i] = Y[j] = P[k] \\ 1 + L(i-1, j-1, k) & \text{if } i,j > 0, X[i] = Y[j] \text{ and } (k = 0 \text{ or } X[i] \neq P[k]) \\ \max(L(i-1,j,k), L(i,j-1,k)) & \text{if } i,j > 0 \text{ and } X[i] \neq Y[j] \end{cases} \tag{3}$$

with boundary conditions, $L(i,0,0) = L(0,j,0) = 0$ and $L(0,j,k) = L(i,0,k) = -\infty$ for any $0 \le i \le n$, $0 \le j \le m$ and $0 < k \le r$. □

The length of CLCS of X, Y with respect to P is given by $L(n,m,r)$.

Algorithm by Francis Chin et al.

Let $m' = n - a + 1$ and $n' = m - b + 1$, the pseudocode for the algorithm is as follows (Algorithm 1). It can be easily seen that the time complexity is $O(m'n')$.

Algorithm 1: LCS-LENGTH(x, y, m', n')

initialize $L_{a,b}[i, 0] \leftarrow 0$ for $1 \leq i \leq m'$ initialize $L_{a,b}[0, j] \leftarrow 0$ for $1 \leq j \leq n'$

```
for i = 1 → m' do
    for j = 1 → n' do
        if x[i] = y[j] then
            L_{a,b}[i, j] ← L_{a,b}[i − 1, j − 1] + 1
        else
            L_{a,b}[i, j] ← max{L_{a,b}[i − 1, j], L_{a,b}[i, j − 1]}
        end
    end
end
return L_{a,b}
```

Algorithm by Francis Chin

From the equation (3), the recursive relation of Chin's algorithm, we can use recursive backtracking from the relations to analysis the complexity of this algorithm.

Algorithm by Francis Chin

From the equation (3), the recursive relation of Chin's algorithm, we can use recursive backtracking from the relations to analysis the complexity of this algorithm.

From analysis of the computation path, we know that it takes at most $O(n + m + r)$ steps.And we can also say that it takes $O(n * m * r)$ time and space to build and calculate the constraint longest common sequence.

Algorithm by Francis Chin

Example: $X =$ "aba", $Y =$ "bab", $P =$ "a".

Algorithm by Francis Chin

Example: $X =$ "aba", $Y =$ "bab", $P =$ "a".

1. (Boundary states) $L(i,0,0) = L(0,j,0) = 0$ for $0 \leq i \leq 3$, $0 \leq j \leq 3$; $L(0,j,k) = L(i,0,k) = -\infty$ for $0 \leq i \leq 3$, $0 \leq j \leq 3$, $0 < k \leq 1$;

Algorithm by Francis Chin

Example: $X =$ "aba", $Y =$ "bab", $P =$ "a".

1. (Boundary states) $L(i, 0, 0) = L(0, j, 0) = 0$ for $0 \leq i \leq 3$, $0 \leq j \leq 3$; $L(0, j, k) = L(i, 0, k) = -\infty$ for $0 \leq i \leq 3$, $0 \leq j \leq 3$, $0 < k \leq 1$;
2. $(X[1] \neq Y[1])$ $L(1, 1, 0) = \max\{L(0, 1, 0), L(1, 0, 0)\} = 0$; $L(1, 1, 1) = \max\{L(0, 1, 1), L(1, 0, 1)\} = 0$;

Algorithm by Francis Chin

Example: $X =$ "*aba*", $Y =$ "*bab*", $P =$ "*a*".

1. (Boundary states) $L(i, 0, 0) = L(0, j, 0) = 0$ for $0 \leq i \leq 3$, $0 \leq j \leq 3$;
   $L(0, j, k) = L(i, 0, k) = -\infty$ for $0 \leq i \leq 3$, $0 \leq j \leq 3$, $0 < k \leq 1$;
2. $(X[1] \neq Y[1])$ $L(1, 1, 0) = \max\{L(0, 1, 0), L(1, 0, 0)\} = 0$;
   $L(1, 1, 1) = \max\{L(0, 1, 1), L(1, 0, 1)\} = 0$;
3. $(X[1] = Y[2], k = 0)$ $L(1, 2, 0) = 1 + L(0, 1, 0) = 1$;
   $(X[1] = Y[2] = P[1])$ $L(1, 2, 1) = 1 + L(0, 1, 0) = 1$;

Algorithm by Francis Chin

Example: X = "aba", Y = "bab", P = "a".

1. (Boundary states) $L(i,0,0) = L(0,j,0) = 0$ for $0 \le i \le 3$, $0 \le j \le 3$; $L(0,j,k) = L(i,0,k) = -\infty$ for $0 \le i \le 3$, $0 \le j \le 3$, $0 < k \le 1$;
2. $(X[1] \ne Y[1])$ $L(1,1,0) = \max\{L(0,1,0), L(1,0,0)\} = 0$; $L(1,1,1) = \max\{L(0,1,1), L(1,0,1)\} = 0$;
3. $(X[1] = Y[2], k = 0)$ $L(1,2,0) = 1 + L(0,1,0) = 1$; $(X[1] = Y[2] = P[1])$ $L(1,2,1) = 1 + L(0,1,0) = 1$;
4. $(X[1] \ne Y[3])$ $L(1,3,0) = \max\{L(0,3,0), L(1,2,0)\} = 1$; $L(1,3,1) = \max\{L(0,3,1), L(1,2,1)\} = 1$;

Algorithm by Francis Chin

Example: X = "aba", Y = "bab", P = "a".

1. (Boundary states) $L(i,0,0) = L(0,j,0) = 0$ for $0 \le i \le 3$, $0 \le j \le 3$; $L(0,j,k) = L(i,0,k) = -\infty$ for $0 \le i \le 3$, $0 \le j \le 3$, $0 < k \le 1$;
2. $(X[1] \neq Y[1])$ $L(1,1,0) = \max\{L(0,1,0), L(1,0,0)\} = 0$; $L(1,1,1) = \max\{L(0,1,1), L(1,0,1)\} = 0$;
3. $(X[1] = Y[2], k = 0)$ $L(1,2,0) = 1 + L(0,1,0) = 1$; $(X[1] = Y[2] = P[1])$ $L(1,2,1) = 1 + L(0,1,0) = 1$;
4. $(X[1] \neq Y[3])$ $L(1,3,0) = \max\{L(0,3,0), L(1,2,0)\} = 1$; $L(1,3,1) = \max\{L(0,3,1), L(1,2,1)\} = 1$;
5. $(X[2] = Y[1], k = 0)$ $L(2,1,0) = 1 + L(1,0,0) = 1$; $(X[2] = Y[1] \neq P[1])$ $L(2,1,1) = 1 + L(1,0,1) = -\infty$;

Algorithm by Francis Chin

Example: $X =$ "aba", $Y =$ "bab", $P =$ "a".

Algorithm by Francis Chin

Example: $X =$ "aba", $Y =$ "bab", $P =$ "a".

6. $(X[2] \neq Y[2])$ $L(2,2,0) = \max\{L(1,2,0), L(2,1,0)\} = 1$;
   $L(2,2,1) = \max\{L(1,2,1), L(2,1,1)\} = 1$;

Algorithm by Francis Chin

Example: $X =$ "aba", $Y =$ "bab", $P =$ "a".

6. $(X[2] \neq Y[2])$ $L(2,2,0) = \max\{L(1,2,0), L(2,1,0)\} = 1$;
   $L(2,2,1) = \max\{L(1,2,1), L(2,1,1)\} = 1$;
7. $(X[2] = Y[3], k = 0)$ $L(2,3,0) = 1 + L(1,2,0) = 2$;
   $(X[2] = Y[3] \neq P[1])$ $L(2,3,1) = 1 + \max\{L(1,3,1), L(2,2,1)\} = 2$

Algorithm by Francis Chin

Example: $X =$ "*aba*", $Y =$ "*bab*", $P =$ "*a*".

6. $(X[2] \neq Y[2])$ $L(2,2,0) = \max\{L(1,2,0), L(2,1,0)\} = 1$; $L(2,2,1) = \max\{L(1,2,1), L(2,1,1)\} = 1$;
7. $(X[2] = Y[3], k = 0)$ $L(2,3,0) = 1 + L(1,2,0) = 2$; $(X[2] = Y[3] \neq P[1])$ $L(2,3,1) = 1 + \max\{L(1,3,1), L(2,2,1)\} = 2$
8. $(X[3] \neq Y[1])$ $L(3,1,0) = \max\{L(2,1,0), L(3,0,0)\} = 1$; $L(3,1,1) = \max\{L(2,1,1), L(3,0,1)\} = -\infty$;

Algorithm by Francis Chin

Example: X = "*aba*", Y = "*bab*", P = "*a*".

6. $(X[2] \neq Y[2])$ $L(2,2,0) = \max\{L(1,2,0), L(2,1,0)\} = 1$;
   $L(2,2,1) = \max\{L(1,2,1), L(2,1,1)\} = 1$;
7. $(X[2] = Y[3], k = 0)$ $L(2,3,0) = 1 + L(1,2,0) = 2$;
   $(X[2] = Y[3] \neq P[1])$ $L(2,3,1) = 1 + \max\{L(1,3,1), L(2,2,1)\} = 2$
8. $(X[3] \neq Y[1])$ $L(3,1,0) = \max\{L(2,1,0), L(3,0,0)\} = 1$;
   $L(3,1,1) = \max\{L(2,1,1), L(3,0,1)\} = -\infty$;
9. $(X[3] = Y[2], k = 0)$ $L(3,2,0) = 1 + L(2,1,0) = 2$;
   $(X[3] = Y[2] = P[1])$ $L(3,2,1) = 1 + L(2,1,0) = 2$;

Algorithm by Francis Chin

Example: $X =$ "*aba*", $Y =$ "*bab*", $P =$ "*a*".

6. $(X[2] \neq Y[2])$ $L(2,2,0) = \max\{L(1,2,0), L(2,1,0)\} = 1$;
   $L(2,2,1) = \max\{L(1,2,1), L(2,1,1)\} = 1$;
7. $(X[2] = Y[3], k = 0)$ $L(2,3,0) = 1 + L(1,2,0) = 2$;
   $(X[2] = Y[3] \neq P[1])$ $L(2,3,1) = 1 + \max\{L(1,3,1), L(2,2,1)\} = 2$
8. $(X[3] \neq Y[1])$ $L(3,1,0) = \max\{L(2,1,0), L(3,0,0)\} = 1$;
   $L(3,1,1) = \max\{L(2,1,1), L(3,0,1)\} = -\infty$;
9. $(X[3] = Y[2], k = 0)$ $L(3,2,0) = 1 + L(2,1,0) = 2$;
   $(X[3] = Y[2] = P[1])$ $L(3,2,1) = 1 + L(2,1,0) = 2$;
10. $(X[3] \neq Y[3])$ $L(3,3,0) = \max\{L(2,3,0), L(3,2,0)\} = 2$;
    $L(3,3,1) = \max\{L(3,2,1), L(2,3,1)\} = 2$.

Algorithm by Francis Chin

Example: X = "*aba*", Y = "*bab*", P = "*a*".

6. $(X[2] \neq Y[2])$ $L(2,2,0) = \max\{L(1,2,0), L(2,1,0)\} = 1$;
   $L(2,2,1) = \max\{L(1,2,1), L(2,1,1)\} = 1$;
7. $(X[2] = Y[3], k = 0)$ $L(2,3,0) = 1 + L(1,2,0) = 2$;
   $(X[2] = Y[3] \neq P[1])$ $L(2,3,1) = 1 + \max\{L(1,3,1), L(2,2,1)\} = 2$
8. $(X[3] \neq Y[1])$ $L(3,1,0) = \max\{L(2,1,0), L(3,0,0)\} = 1$;
   $L(3,1,1) = \max\{L(2,1,1), L(3,0,1)\} = -\infty$;
9. $(X[3] = Y[2], k = 0)$ $L(3,2,0) = 1 + L(2,1,0) = 2$;
   $(X[3] = Y[2] = P[1])$ $L(3,2,1) = 1 + L(2,1,0) = 2$;
10. $(X[3] \neq Y[3])$ $L(3,3,0) = \max\{L(2,3,0), L(3,2,0)\} = 2$;
   $L(3,3,1) = \max\{L(3,2,1), L(2,3,1)\} = 2$.

Finally, $|Z| = L(3,3,1) = 2$.

Algorithm by Francis Chin

```
class CLCSSolver
{
public:
  CLCSSolver(string x, string y, string p);
  ~CLCSSolver();
  void solv();
  int getMaxLength();
  string getCLCS();
protected:
  void solv(string x, string y, string p);
private:
  string _x, _y, _p;
  CLCSState *_states;
  CLCSState *_direct;
  enum {D_NNN, D_NNZ, D_NZZ, D_ZNZ};
};
```

Listing 1: class CLCSSolver

Experimental Study

We took two steps to perform this study.

1. **Correctness**
   First we implemented the algorithms and tested the correctness of implementation with manually selected test cases and program-generated test cases.

Experimental Study

We took two steps to perform this study.

1. **Correctness**
   First we implemented the algorithms and tested the correctness of implementation with manually selected test cases and program-generated test cases.
2. **Performance**
   Then we ran the programs with large program-generated test cases to assess the performance.

Correctness

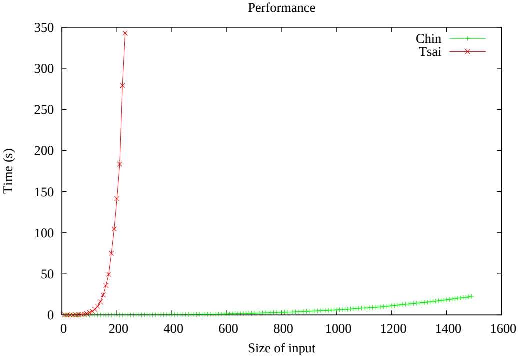
To test whether the algorithms were correctly implemented, we chose a few boundary cases to do white-box test, and generated a series of test cases for black-box test.

The program-generated test cases consist of all the possible combinations of the strings of length less than four. We have $39^3 = 59319$ different test cases.

As the test cases are regularly generated, it is easy to manually find out the answers to the cases.

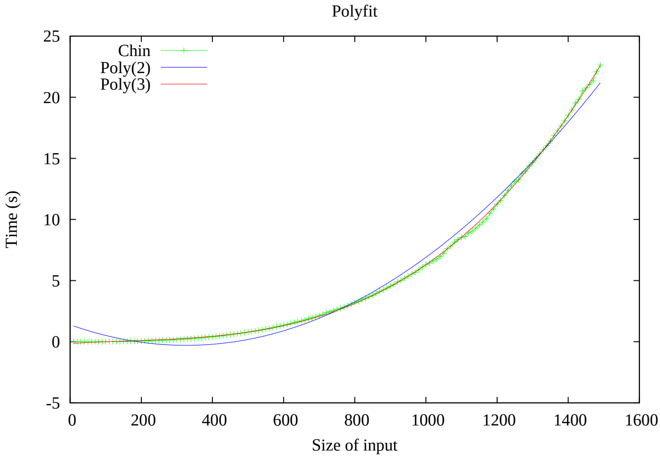
Performance

Randomly generated a series of 3-tuples as test cases. Each tuple consists of two strings and a constraint string, which are defined on the alphabet $\Sigma = \{A, T, G, C\}$. For the k-th tuple (X, Y, P), the lengths of the elements satisfy the condition $|X| = |Y| = 2|P| = 10k$.



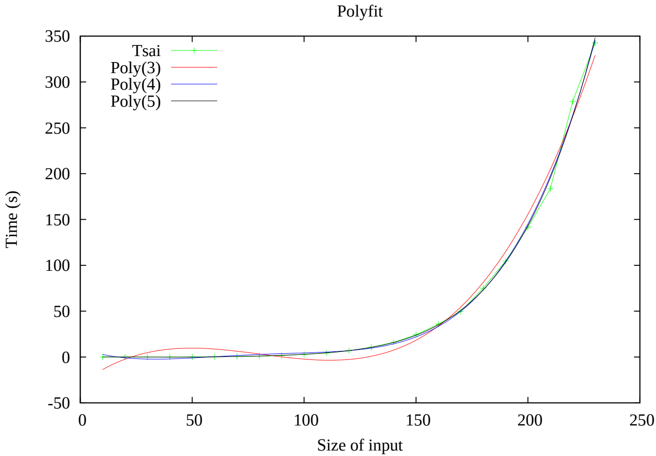
Fitting Data with Polynomials

Chin's algorithm



Fitting Data with Polynomials

Tsai's algorithm



References

Robert A. Wagner and Michael J. Fischer.
The string-to-string correction problem.
J. ACM, 21:168–173, January 1974.

Yin-Te Tsai.
The constrained longest common subsequence problem.
Inf. Process. Lett., 88:173–176, November 2003.

Francis Y. L. Chin, Alfredo De Santis, Anna Lisa Ferrara, N. L. Ho, and S. K. Kim.
A simple algorithm for the constrained sequence problems.
Information Processing Letters, 90:175–179, 2004.