Assigment3 - Suffix Tree & Whole genome alignment

Weize Xu

June 8, 2018

1 Please give the generalized suffix tree for S_1=$ACGT\$$ and S_2=$TGCA\#$.

Answer:

See Fig.1. (My code for building and visualizing suffix tree, see: https://github.com/Nanguage/suffix-trees)

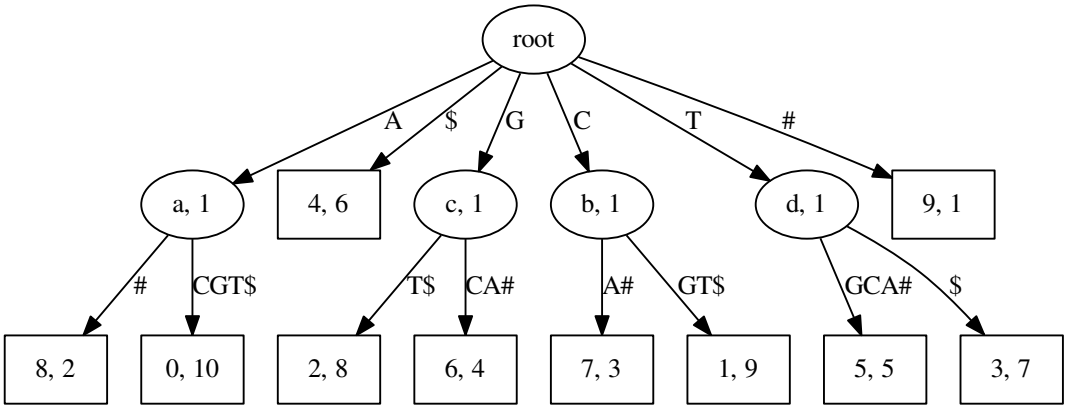

Figure 1: Generalized suffix tree.

2 Consider a reference genome $T[1..n]$ and a read $R[1..m]$ where $n > m$. We want to report all positions i such that $Hamming(R, T[i..i+m-1]) \leq k$. Please propose an $O(kn)$ - time algorithm. (Hint: We can build the suffix tree for $T\#R\$$ and the corresponding longest common prefix data structure using $O(n+m)$ time.)

Answer:

This is not easy, and I'm not have enough time, so I referenced some papers. In 1989 Landau and Vishkin proposed an approximate string matching algorithm[1]. This algorithm can solve approximate matching problem in $O(kn)$ time. In the original paper they denote k as edit distance, but here the k as Hamming distance, so it need to be modified for solve this problem.

Algorithm description:

We use $V(i, j)$ denote the Hamming distance between the $R[1..i]$ and $T[j-i..j]$. For example, when $T = GGGTCTA$ and $R = GTTC$, V will be look like this:

	G	G	G	T	C	T	A
G	0	0	0	1	0	1	1
T		1	1	0	2	1	2
T			2	1	1	2	2
C				3	1	2	3

Then we use $L_{d,e}$ denote the largest row in the V_d, that satisfy $V(row, row+d-1) \leq e$. Where V_d is the *d-th* diagonal in the table V. For example $V_1 = 0, 1, 2, 3$ $V_2 = 0, 1, 1, 1$ ... Clearly, if $L_{d,e} = m$ and $e \leq k$, we can say that $Hamming(R, T[d..d+m]) \leq k$. So our goal is equivalente to find all d, that $L_{d,e} = m$ where $e \leq k$.

Actually, according to Landau and Vishkin's proof[1], $L_{d,e}$ can be computed by a dynamic programming algorithm.

Algorithm 1: Dynamic programming for compute $L_{d,e}$

```
input : T[1..n], R[1..m], k
initialize(L_{d,e}, T, R)
for e ← 1 to k do
    for d ← 1 to n − m + 1 do
        row ← max{(L_{d,e−1} − 1), (L_{d−1,e−1}), (L_{d+1,e−1} + 1)}
        while (row < m) ∧ (row + d < n) ∧ (R[row + 1] = T[row + d + 1]) do
            row ← row + 1
        L_{d,e} ← row
        if L_{d,e} = m then
            print(d)
```

Then we discuss the base case of this dynamic programming algorithm:

Firstly we build the generalized suffix tree of T and R. The base case $L_{d,0}$ equal to find the LCP(Longest Common Prefix) of R and $T[d..d+m]$. This can be done in $O(1)$ time.

Algorithm 2: Base case initialization

```
procedure initialize(L_{d,e}, T, R)
    L_{0,e} ← 0
    build generalized suffix tree τ of T and R
    for d ← 1 to n − m + 1 do
        L_{d,0} ← LCP(R, T[d..d + m])
```

Time analyze:

In this algorithm, build suffix tree take $O(m+n)$ time. For table initialization, need fill $n-m+1$ entry, each entry take $O(1)$ time, so this step take $O(n)$ time. Fill the full table $L_{d,e}$ take $O(kn)$ time. So whole algorithm can be done in $O(kn)$ time.

3 Consider S_1=$AAAACGTCGGGATCG$ and S_2=$GGGCGTAAAGCTCT$. Suppose the minimum length of MUM is 3.
(a) What is the set of MUMs?
(b) If we further require that the MUMs are also unique in the sequence and its reverse complement, what is the set of MUMs?

Answer:

(a): The set of MUMs is $\{CGT, GGG\}$

(b): If consider unique in the reverse complement sequence, the set of MUMs is: $\{GGG, CCC\}$

4 Let A=123...9 and B=961472358. Suppose we run the $O(nlog(n))$ - time algorithm to find the longest common subsequence. Can you report the list of tuples stored in T_i for i=$0, 1, ..., 9$? Please discuss how do you construct T_i from T_{i-1}.

Answer:

i	T_i
1	{(2, 1)}
2	{(2, 1), (5, 2)}
3	{(2, 1), (5, 2), (6, 3)}
4	{(2, 1), (3, 2), (6, 3)}
5	{(2, 1), (3, 2), (6, 3), (7, 4)}
6	{(1, 1), (3, 2), (6, 3), (7, 4)}
7	{(1, 1), (3, 2), (4, 3), (7, 4)}
8	{(1, 1), (3, 2), (4, 3), (7, 4), (8, 5)}
9	{(0, 1), (3, 2), (4, 3), (7, 4), (8, 5)}

Construct T_i from T_{i-1}:

1. Delete all tuples (j, $C_{i-1}[j]$) where $j \geq \delta(i)$ and $C_{i-1} \leq Ci - 1[\delta(i) - 1] + 1$
2. Insert $(\delta(i), C_{i-1}[\delta(i) - 1] + 1)$

My implementation see:
https://github.com/Nanguage/Course-Algorithms-in-Bioinformatics/blob/master/L09/bs_lcs.py.

References

[1] Gad M Landau and Uzi Vishkin. Fast parallel and serial approximate string matching. *Journal of algorithms*, 10(2):157–169, 1989.