Comp 561 Assignment 2

Jonathan Pearce, 260672004

October 19, 2017

Problem 1a. Using Blastn, word length 7 and expected threshold 10. It appears the most likely cause of the infection is Schistosoma mansoni. My query returned 111 hits, the DNA of the schistosoma mansoni parasite had the lowest E-value by a factor of 3 (0.16) and the next 20 best hits were all related to zebrafish DNA which seemed an unlikely source. Schistosoma mansoni is an agent of the disease Schistosomiasis in particular this parasite tends to cause intestinal schistosomiasis.

Problem 1b. According to the Phylogenetic tree output, Strain 5 is the most similar to my patients DNA sample. I would therefore recommend using strain 5 as treatment for my patient. I want to use Strain 5 because of its similarities in DNA structure, assuming this strain will cure my patient of schistosomiasis it will also alter their DNA as little as possible. My fear with strain 1-4 is that although they could possibly work, they may alter my patients DNA so much that they experience negative side effects (i.e. cure schistosomiasis, but introduce another issue). Therefore I would suggest using strain 5.
Note Phylogenetic Tree attached below

Problem 2. We will prove by induction that Fitch's Algorithm will produce the same answer as Sankoff's Algorithm, in fact we will show that the score at every node in tree T in Fitch's Algorithm is equal to the minimum score of the $\{A, C, G, T\}$ array at each corresponding node in Sankoff's Algorithm.

Notation: In the problem statement X_u was defined for Fitch's Algorithm. We will define $Xscore(u)$ as the Parsimony score for the subtree rooted at node u in Fitch's Algorithm. Now in Sankoff's Algorithm every node u has a 4x1 array Y_u that stores the possible Parsimony scores of the subtree rooted at u, where each element of the array considers if the node u was either A, C, G or T. We will define the following,

$$S_u = \underset{y \epsilon \{A,C,G,T\}}{\text{argmin}} \; Y_u(y)$$

$$Sscore(u) = \min_{y \epsilon \{A,C,G,T\}} Y_u(y)$$

For example if $Y_u = \{A : 5, C : 7, G : 6, T : 5\}$, then $S_u = \{A, T\}$ and $Sscore(u) = 5$

Base Case: At every leaf u in Fitch's Algorithm $Xscore(u) = 0$ and $X_u = \{x\}$ where x is the nucleotide at leaf u. Similarly at every leaf v in Sankoff's Algorithm $Sscore(v) = 0$

and $S_v = \{y\}$ where y is the nucleotide at leaf v. We conclude that if $u = v \Rightarrow X_u = S_v$ and $Xscore(u) = Sscore(v)$.

Induction Step: Given parent node u with children v and w. Assume that $Xscore(v) = Sscore(v)$ and $Xscore(w) = Sscore(w)$, as well Assume that $X_v = S_v$ and $X_w = S_w$.

Case 1: $X_v \cap X_w \neq \emptyset$. From Fitch's algorithm we conclude $X_u = X_v \cap X_w$ and $Xscore(u) = Xscore(v) + Xscore(w)$. Since $X_v \cap X_w \neq \emptyset \Rightarrow S_v \cap S_w \neq \emptyset$, therefore $\exists\, \alpha \,\epsilon\, S_v \cap S_w$. From the scoring equation in Sankoff's Algorithm it is clear that $Sscore(u) = Y_v(\alpha) + Y_w(\alpha) \Leftrightarrow Sscore(u) = Sscore(v) + Sscore(w)$ since $\alpha \,\epsilon\, S_v$ and $\alpha \,\epsilon\, S_w$. Using this we conclude that $Sscore(u) = Xscore(u)$. Furthermore since $Sscore(u) = Y_v(\alpha) + Y_w(\alpha) \Rightarrow \alpha \,\epsilon\, S_u$, therefore $S_u = S_v \cap S_w \Rightarrow S_u = X_u$.

Case 2: $X_v \cap X_w = \emptyset$. From Fitch's algorithm we conclude $X_u = X_v \cup X_w$ and $Xscore(u) = Xscore(v) + Xscore(w) + 1$. Since $X_v \cap X_w = \emptyset \Rightarrow S_v \cap S_w = \emptyset$ therefore $\nexists\, \beta \,\epsilon\, S_v \cap S_w$. We can see that $Sscore(u) = Y_v(\gamma) + Y_w(\delta) + 1$ where $\gamma \,\epsilon\, S_v$ and $\delta \,\epsilon\, S_w$ and the +1 because $\gamma \neq \delta$, it follows $Sscore(u) = sScore(v) + sScore(w) + 1 \Rightarrow Sscore(u) = Xscore(u)$. Furthermore since $Sscore(u) = Y_v(\gamma) + Y_w(\delta) + 1 \Rightarrow \gamma, \delta \epsilon S_u$, therefore $S_u = S_v \cup S_w \Rightarrow S_u = X_u$.

We have proven that in both cases $Sscore(u) = Xscore(u)$ and $S_u = X_u$. At the root node Sankoff's Algorithm would report the Parsimony score to be $xScore(root)$ and Fitch's Algorithm would report the Parsimony score to be $Sscore(root)$ therefore Fitch's Algorithm is correct and will always produce the same Parsimony score as Sankoff's Algorithm.

Problem 3a. Pseudo code attached below

Problem 3b. Runtime Calculation:

1. Run method 'calcTrait' for all k traits
2. Iterate over all n interior nodes in post order traversal
3. Populate M*M*M matrix for interior node
4. Find value of trait for root node
5. Recursively find value of traits for n-1 remaining interior nodes

Time = $O(k(nM^3 + M^3 + (n-1)M^2)) = O(knM^3 + kM^3 + k(n-1)M^2) = O(nkM^3)$.

Therefore the run time of my algorithm is $O(nkM^3)$

Problem 3c. The ranges of traits being different can effect how significant one trait is at influencing the relationship between species. For example, given species A's traits and asked to determine whether species B or species C is more similar we observe that B and C have nearly exactly the same traits except that B's lifespan differs by 10 percent and C's volume of brain differs by 5 percent from A. Simply looking at theses numbers you would guess that

C is more similar since the only difference is a 5 percent shift in one trait (rather than 10 with C). However the way we have defined this scoring system it would find C to be more similar than B because 10 percent of a range of 50 (lifespan) is 5, however 5 percent of a range of 1500 (volume of brain) is 75. Therefore we would conclude that B is more similar even though the one trait differs by twice as much (proportionally) than that of C's differing trait. A solution to this would be to normalize each trait's range and shift there values such that every trait has a range from 0 to 100 and thus they are all equally valuable to analysing the relationship between species.

```
3) internalNodes = $D_{n+1}, D_{n+2} \ldots D_{2n-1}$

for $i = 1$ to K :

    calcTrait ( $D_{1,i}, D_{2,i}, \ldots, D_{n,i}, T$ ) # extract $i^{th}$ trait
                                                          from n leaves
```

```
CalcTrait ($t_1, t_2, \ldots t_n, T$) # n traits and tree T

for a = (n+1) to (2n - 1)
    $X_a$ = [M][M][M]    # Every interior node gets MxMxM Matrix

                                                             # Traverse
                                                             # Tree in
for all interior nodes u with children v and w               # post-order
    for i = 0 to M :   # all possible values of u
        for j = 0 to M :   # values of v
            for k = 0 to M :   # values of w
                $X_u[i][j][k] = 0$

                if v is a leaf :
                    $X_u[i][j][k] \mathrel{+}= |i - t_v|$
                else :
                    $X_u[i][j][k] \mathrel{+}= \left[ \min_{\substack{a \in \{1:M\} \\ b \in \{1:M\}}} \{ X_v[j][a][b] \} + |i - j| \right]$

                if w is a leaf :
                    $X_u[i][j][k] \mathrel{+}= |i - t_w|$
                else :
                    $X_u[i][j][k] \mathrel{+}= \left[ \min_{\substack{c \in \{1:M\} \\ d \in \{1:M\}}} \{ X_w[k][c][d] \} + |i - k| \right]$
```

AT this point all the M×M×M arrays have been filled

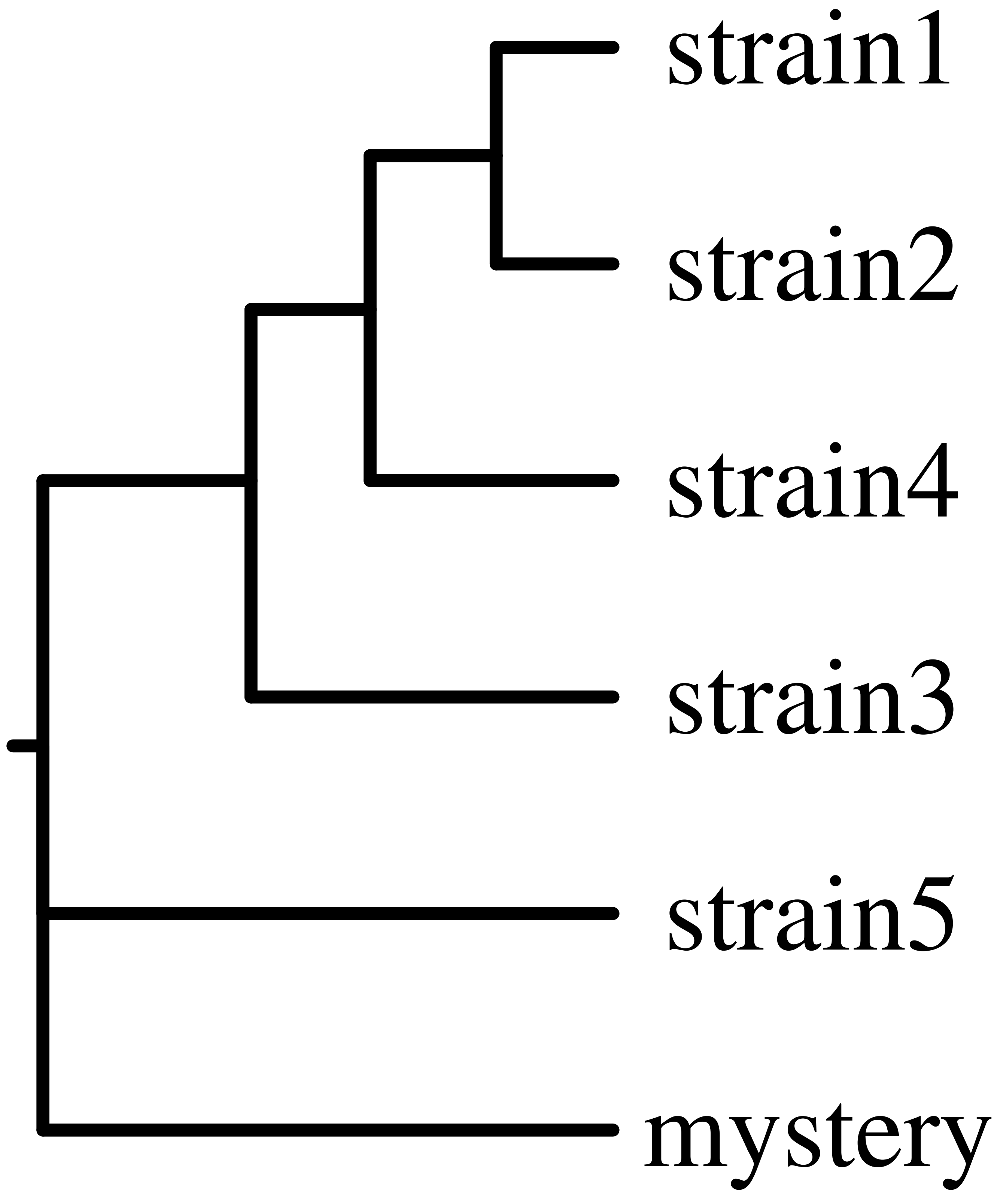
```
Min = $X_{2n-1}[0][0][0]$ , leftChild = 0, rightChild. = 0
                                    , trait = 0
For i = 0 to M:
    For j = 0 to M:
        For K = 0 to M:
            if $X_{2n-1}[i][j][k]$ < min :
                min = $X_{2n-1}[i][j][k]$
                leftChild = j
                rightchild = K
                trait = i

$D_{2n-1,i}$ = trait   # get the ith trait of root node
backtrack ( leftChild , v ) #where v is the left child node
backtrack ( rightchild, w ) #where w is the right child node
```

these 2 lines will recursively traverse the interior nodes

```
return
```

backtrack (n, u) | This method recursivly finds the ith trait of the interior nodes

```
if u is a leaf :
    return
else :
    min = $X_u[n][0][0]$ , leftChild = 0, rightchild = 0
    for i = 0 to M :
        for j = 0 to M :
            If $X_u[n][i][j] < min$ :
                min = $X_u[n][i][j]$
                leftChild = i
                rightchild = j
    $D_{u,i} = n$   # ith trait of this interior node
    backtrack (leftchild, v)
    backtrack (rightchild, w)
    return
```


strain1
strain2
strain4
strain3
strain5
mystery