Bioinformatics

Richard Bonneau

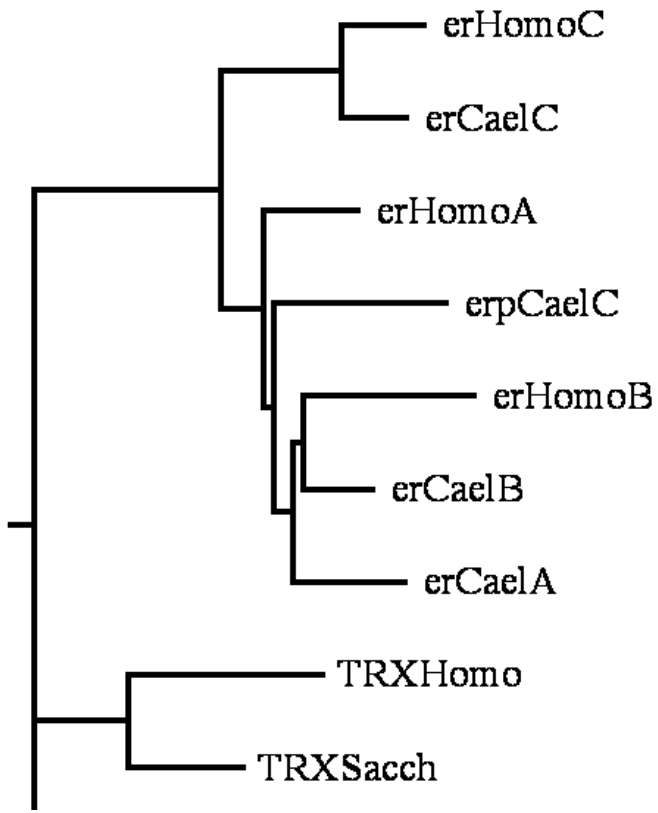
Lecture 5: building phylogenetic trees #1.

Associated reading.

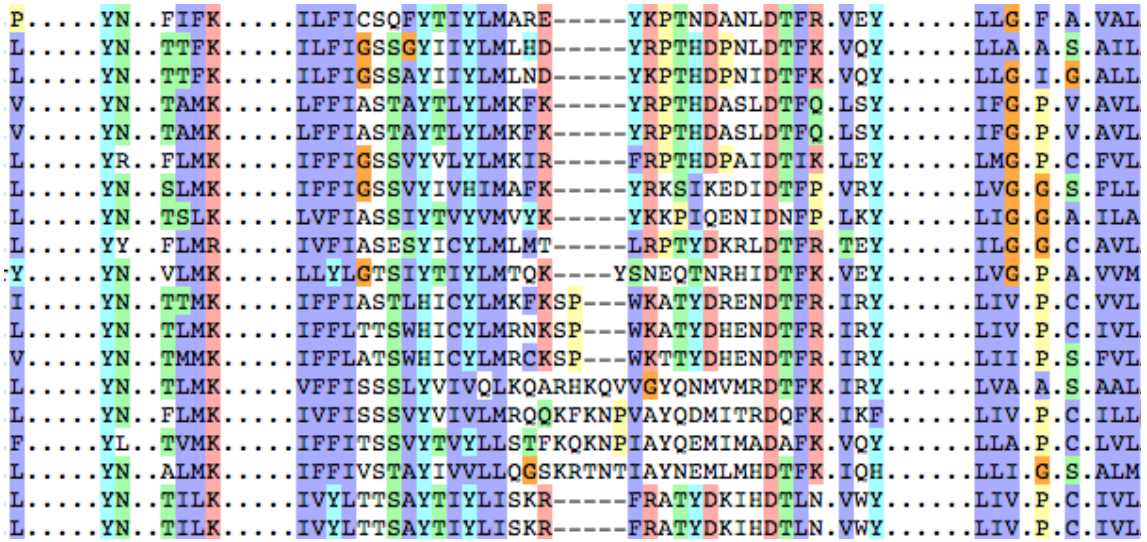
- BSA Chapter 7 (next next week will be 8)

- Drummond AJ,Ho SYW,Phillips MJ,Rambaut A (2006) Relaxed Phylogenetics and Dating with Confidence. PLoS Biol 4(5): e88

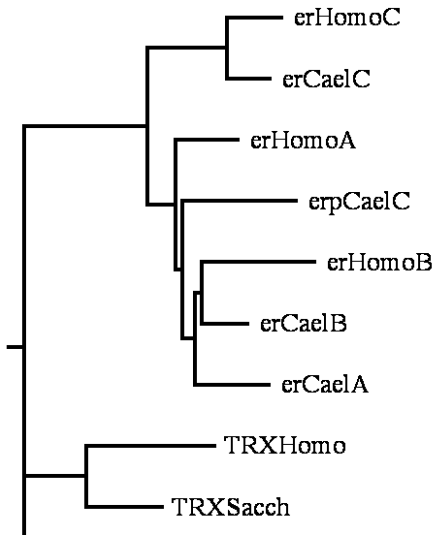
Phylogenetic trees (justification)



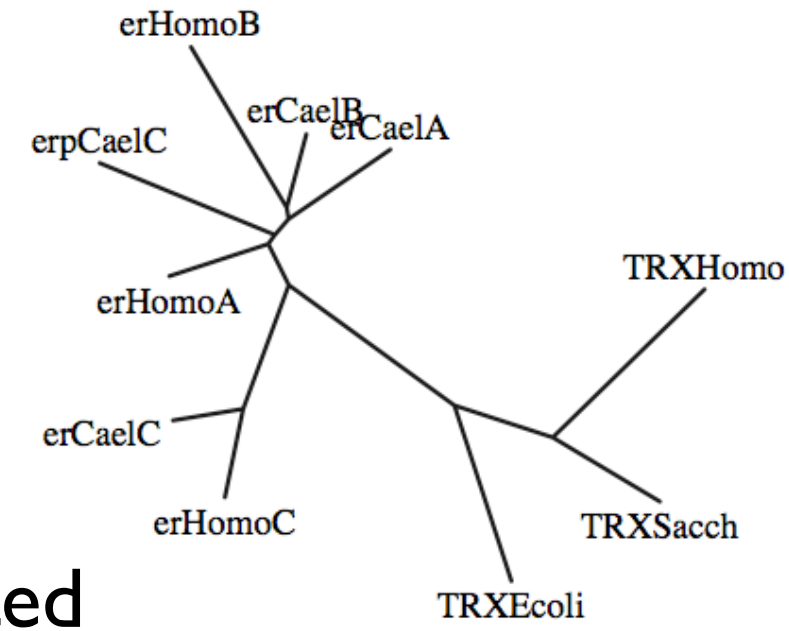
We want to, given an alignment of biological sequences (and possibly the species origin of each sequence) reconstruct the pattern of divisions/duplications and reproductive events (with mutation occurring at each step) that resulted in these sequences.



Phylogenetic trees (justification)



rooted



unrooted

Definitions:

Each internal node (joint) represents a ancestral sequence mutating into two different sequences.

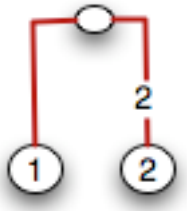
Length of branches proportional to change.

Root is ancestor of whole tree, if tree is rooted. Many of the alg we'll talk about today don't care where the root is, but need one anyway

Leaf nodes are labeled with sequence OR species names or both.

Making trees from pairwise distances UPGMA

d	1:	2:	3:	4:	5:
1:	0				
2:	2	0			
3:	2	3	0		
4:	5	4	4	0	
5:	4	4	2	2	0



Given alignment:

```
1: AGCTTC-TA
2: ACGTTCTTA
3: AGCTTATTA
4: TCCTATTTA
5: TCCTTATTA
```

(between leaf nodes):

D_{ij}= number of mismatches (we could also use s(x,y) as in lecture 2

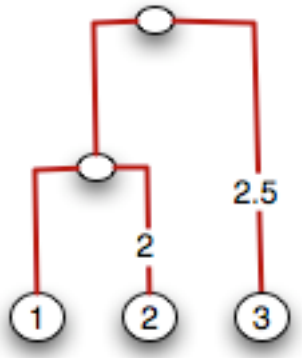
(between branch/cluster and leaf or branch/cluster):

D_{ij}= mean distance between cluster members:

$$d_{ij} = \frac{1}{|C_i||C_j|} \sum_{p \in C_i, q \in C_j} d_{pq}$$

Making trees from pairwise distances

d	1:	2:	3:	4:	5:
1:	0				
2:	2	0			
3:	2	3	0		
4:	5	4	4	0	
5:	4	4	2	2	0



Given alignment:

```
1: AGCTTC-TA
2: ACGTTCTTA
3: AGCTTATTA
4: TCCTATTTA
5: TCCTTATTA
```

(between leaf nodes):

D_{ij}= number of mismatches (we could also use s(x,y) as in lecture 2

(between branch/cluster and leaf or branch/cluster):

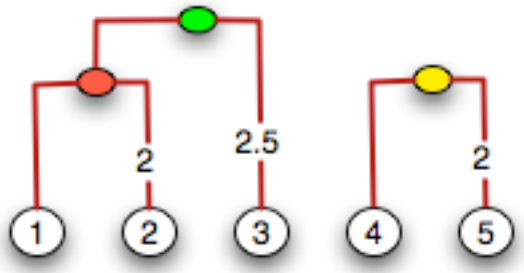
D_{ij}= mean distance between cluster members:

$$d_{ij} = \frac{1}{|C_i||C_j|} \sum_{p \in C_i, q \in C_j} d_{pq}$$

Making trees from pairwise distances

d	1:	2:	3:	4:	5:
1:	0				
2:	2	0			
3:	2	3	0		
4:	5	4	4	0	
5:	4	4	2	2	0

2 more steps



Given alignment:

```
1: AGCTTC-TA
2: ACGTTCTTA
3: AGCTTATTA
4: TCCTATTTA
5: TCCTTATTA
```

(between leaf nodes):

D_{ij}= number of mismatches (we could also use s(x,y) as in lecture 2

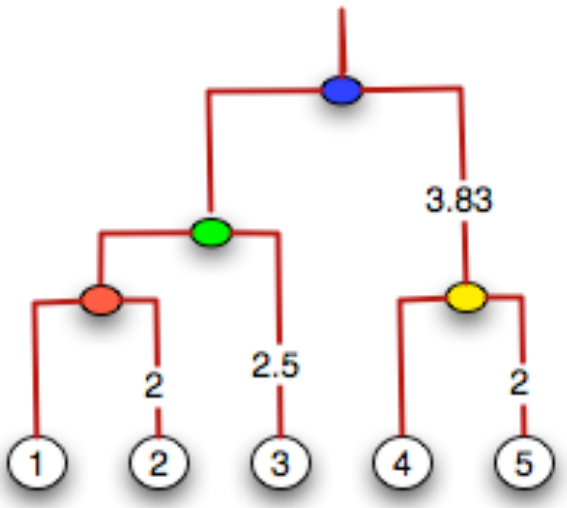
(between branch/cluster and leaf or branch/cluster):

D_{ij}= mean distance between cluster members:

$$d_{ij} = \frac{1}{|C_i||C_j|} \sum_{p \in C_i, q \in C_j} d_{pq}$$

Making trees from pairwise distances

d	1:	2:	3:	4:	5:
1:	0				
2:	2	0			
3:	2	3	0		
4:	5	4	4	0	
5:	4	4	2	2	0




```
1:  AGCTTC-TA
2:  ACGTTCTTA
3:  AGCTTATTA
4:  TCCTATTTA
5:  TCCTTATTA
```

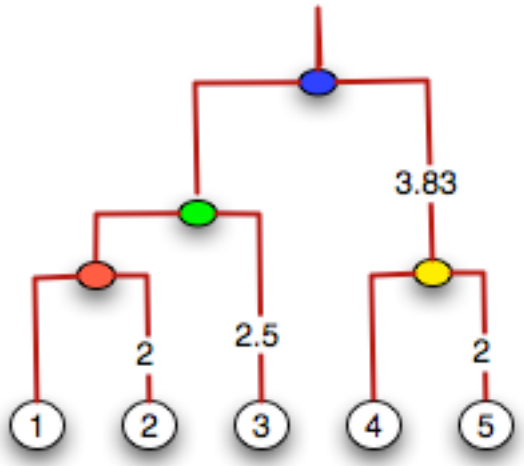
Note: We could also use the Jukes-Cantor dist, which has better limiting behavior at large distances, where f is # of mismatches:

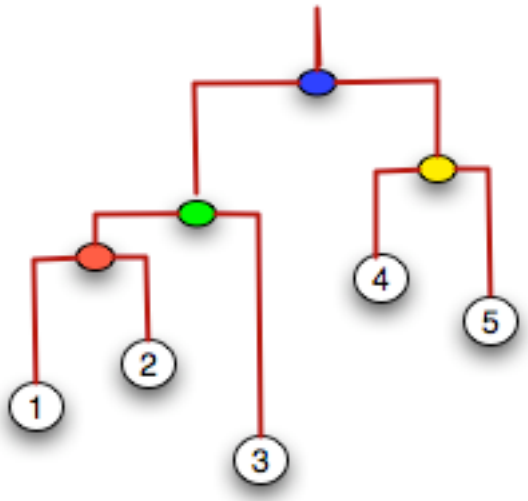
$$d_{ij} = -3/4 \log\left(1 - 4f/3\right)$$

Extra credit:

[What is value of this distance for two infinitely long random sequences where P(A)=P(C) = P(G) = P(T) ?]

Assumptions:

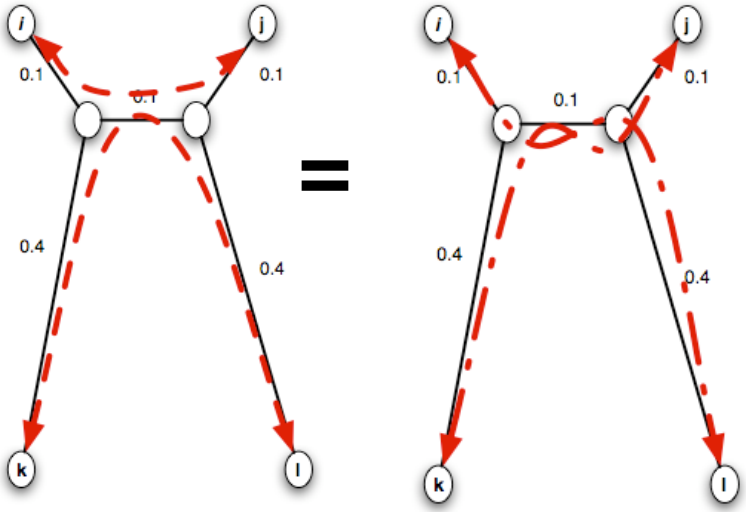




1. <u>Molecular clock:</u>
   that all sequences evolve at similar rate and are thus same distance from tree.

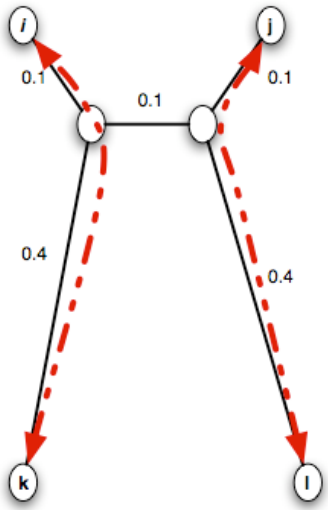
[discussed in length in Kimura, 1983]

Assumptions:



=

≥



1. Molecular clock:
that all sequences evolve at similar rate and are thus same distance from tree.
[discussed in length in Kimura, 1983]

2. Additivity:

That any distance can be reconstructed based on traversing the tree.

Test for additivity on reconstructed tree:
given any three distances on tree:
d_{ij}, d_{ik} , d_{jk} , d_{jk}
d(i,j) + d(i,l); d(i,k) + d(j,l);
d(i,l) + d(j,k)
Two should be equal and greater than the third.

Neighbor joining:

We still assume additivity, we still use a deterministic joining algorithm, but we redefine distance and the algorithms slightly to better deal with variable branch lengths.

$$d_{ij} = -3/4 \log\left(1 - 4f/3\right)$$

$$D_{ij} = d_{ij} - (r_i + r_j)$$

We calc a distance D, where d is corrected by mean path to other nodes, r. (where L is number of leaves)

$$r_i = \frac{1}{|L| - 2} \sum_{k \in L} d_{ik}$$

Algorithm:

0. leaf nodes -> L, we'll grow tree from this set, L
1. Pick $\text{argmin}_{ij}(D_{ij})$ and make node k joining i and j
2. Calc distance from k to all other nodes
3. Add k to growing tree
4. Remove i and j from node list (now they are represented by k
5. Rinse, lather, repeat.

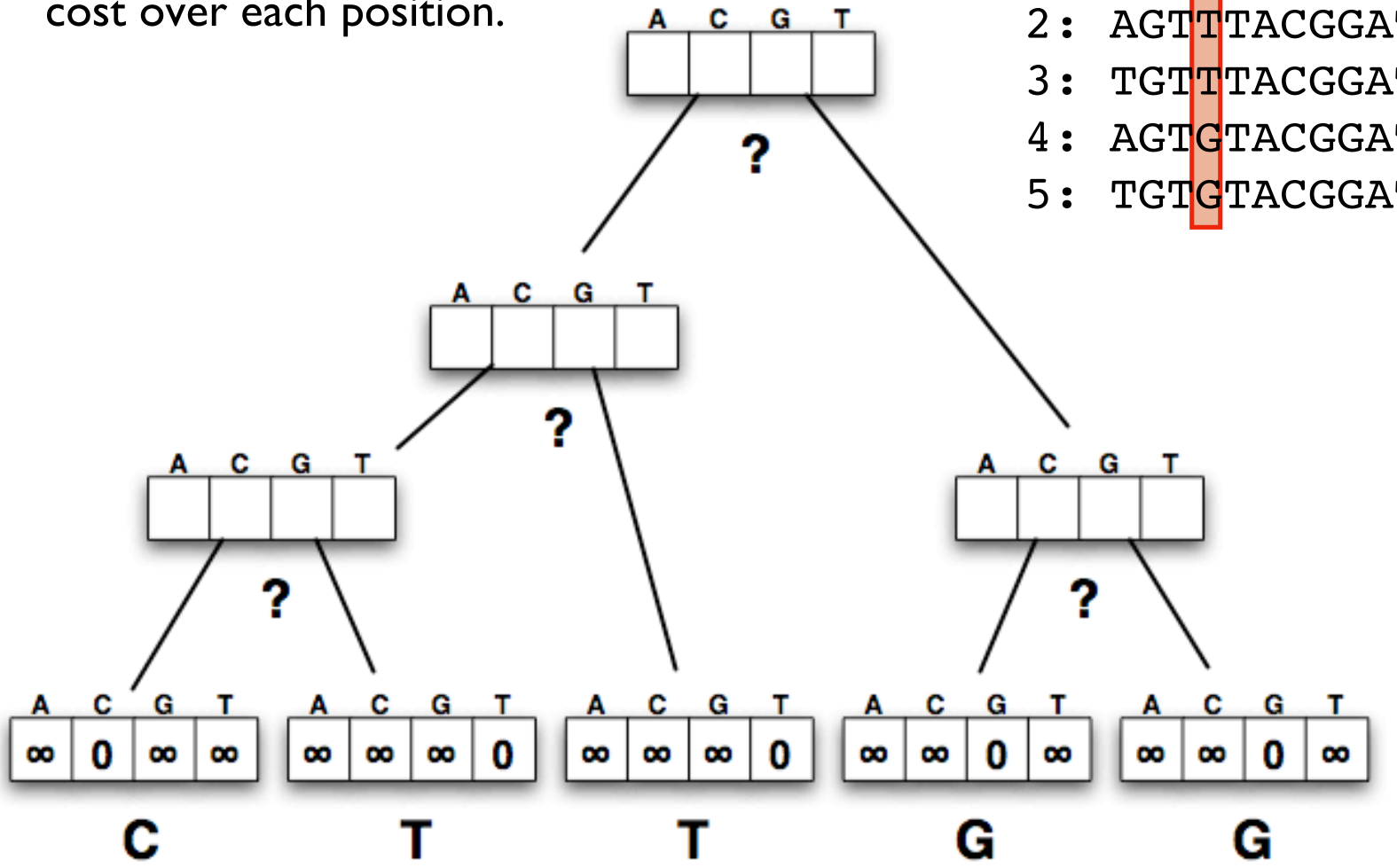
$$d_{k\bullet} = \frac{1}{2}(d_{i\bullet} + d_{j\bullet} - d_{ij})$$

Sankoff + Cedergren (weighted parsimony)

We evaluate the tree using the parsimony algorithm summing the cost over each position.

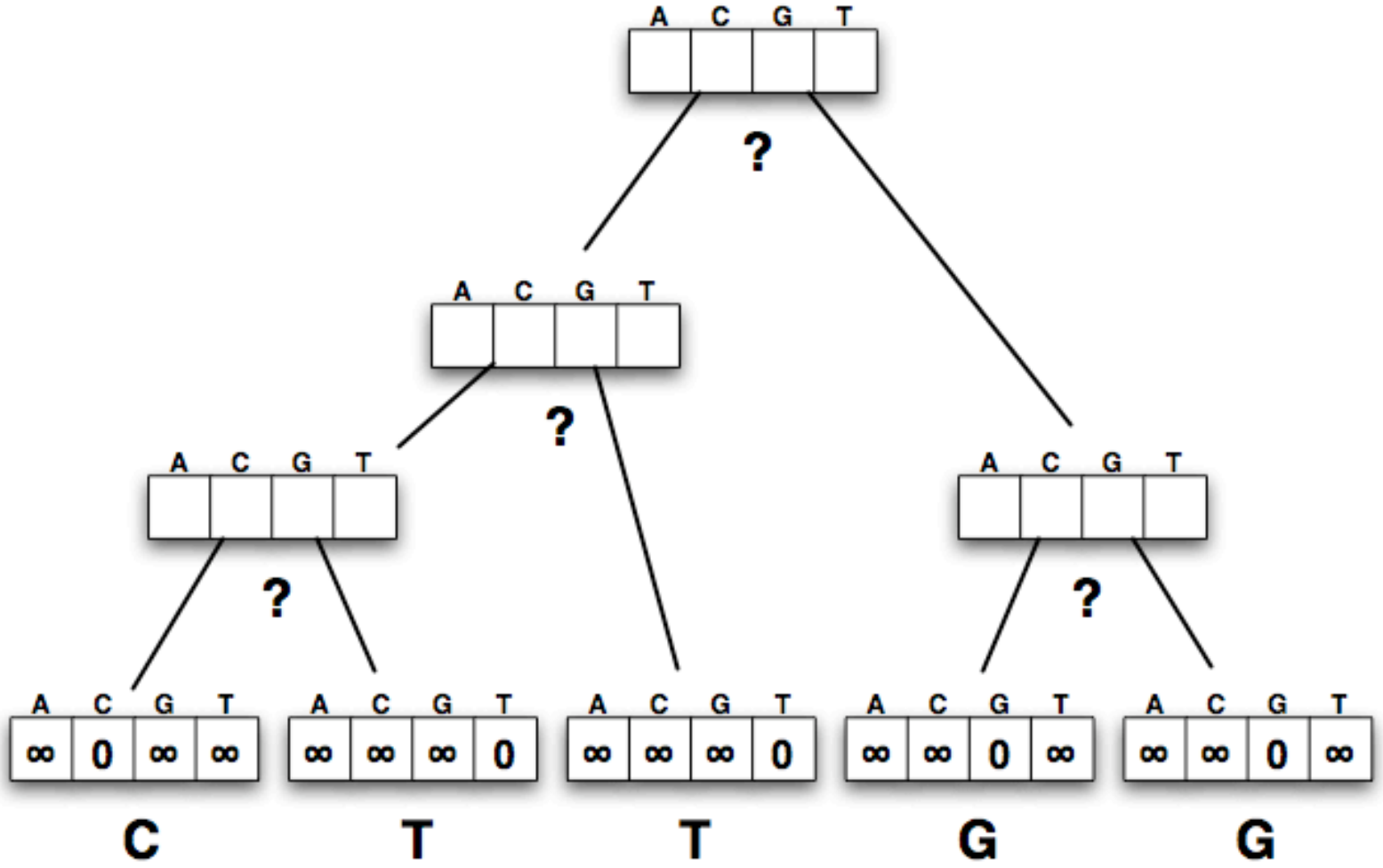
Evaluate cost of tree: Given tree + Given alignment.

```
1: AGTCTACGGATTAT
2: AGTTTACGGATTAT
3: TGTTTACGGATTAT
4: AGTGTACGGATTAA
5: TGTGTACGGATTAA
```



Sankoff + Cedergren (weighted parsimony)

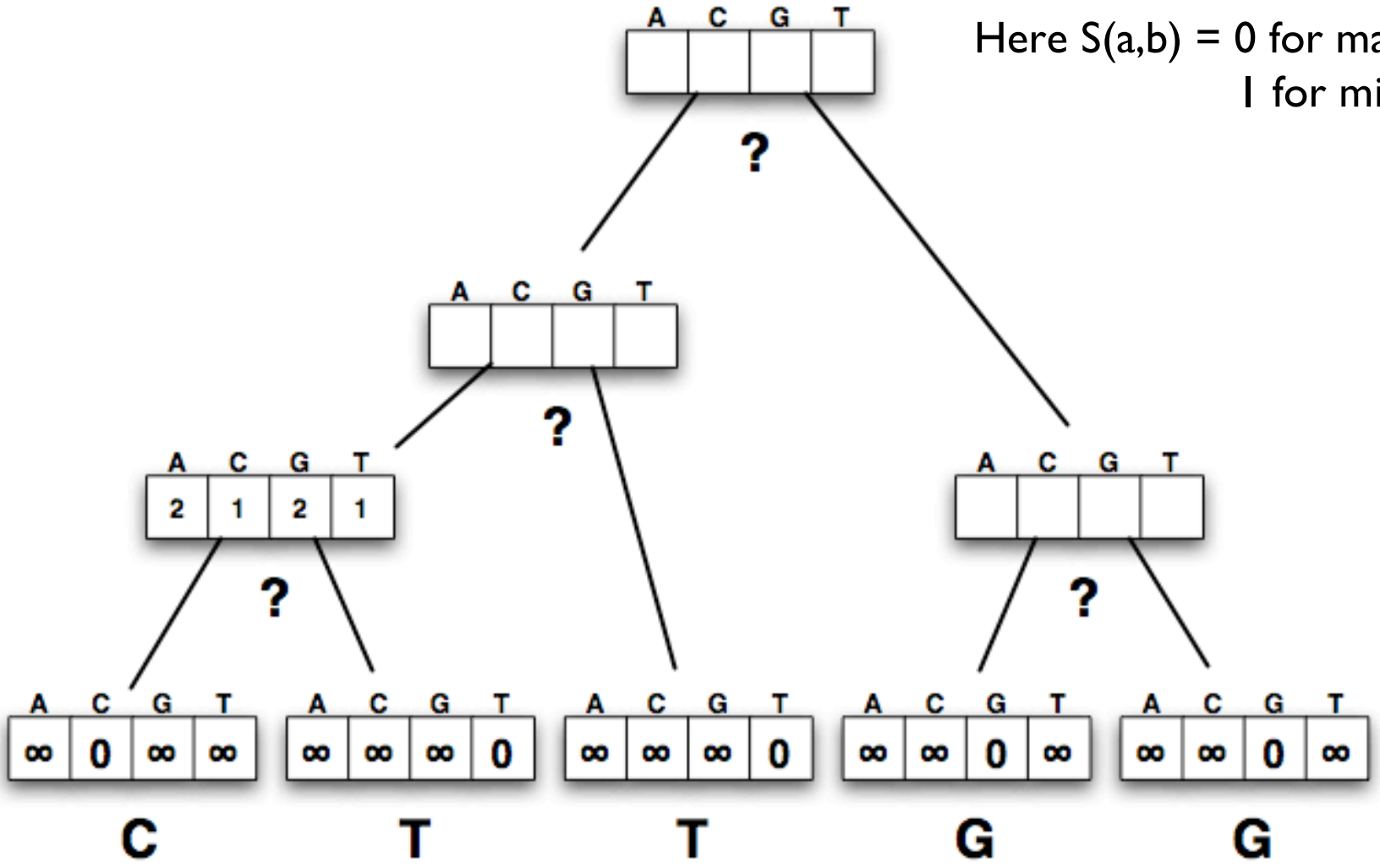
1. Set $S_k(a) = 0$ for a = x, infinity otherwise.



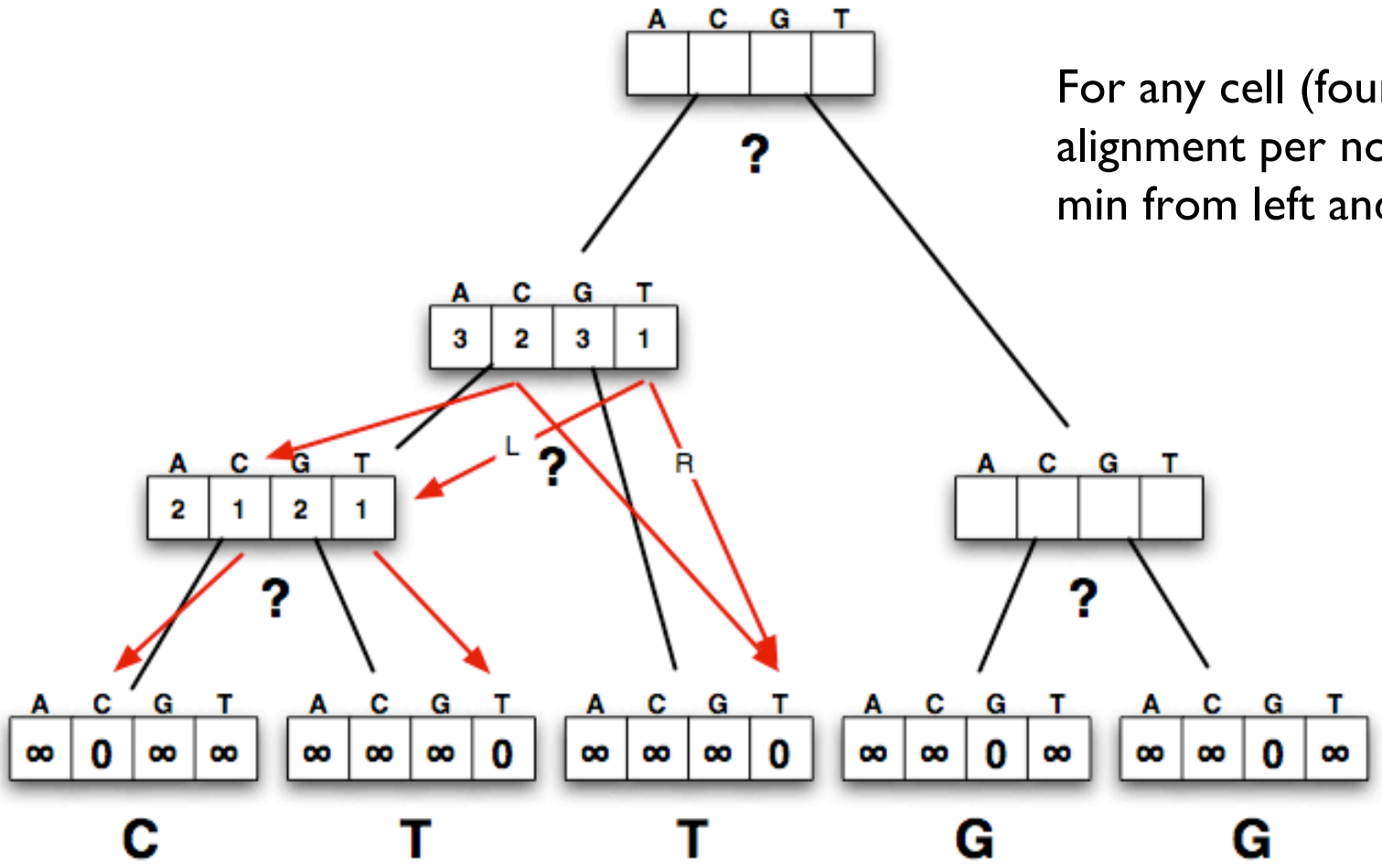
Sankoff + Cedergren (weighted parsimony)

2. If not at leaf node (internal vertex):
$S_k(a) = \min_b(S_i(b) + S(a,b)) + \min_b(S_j(b) + S(a,b))$

Here S(a,b) = 0 for match,
1 for missmatch



Sankoff + Cedergren (weighted parsimony)



Keep filling out tree, traversing from leaves to top.

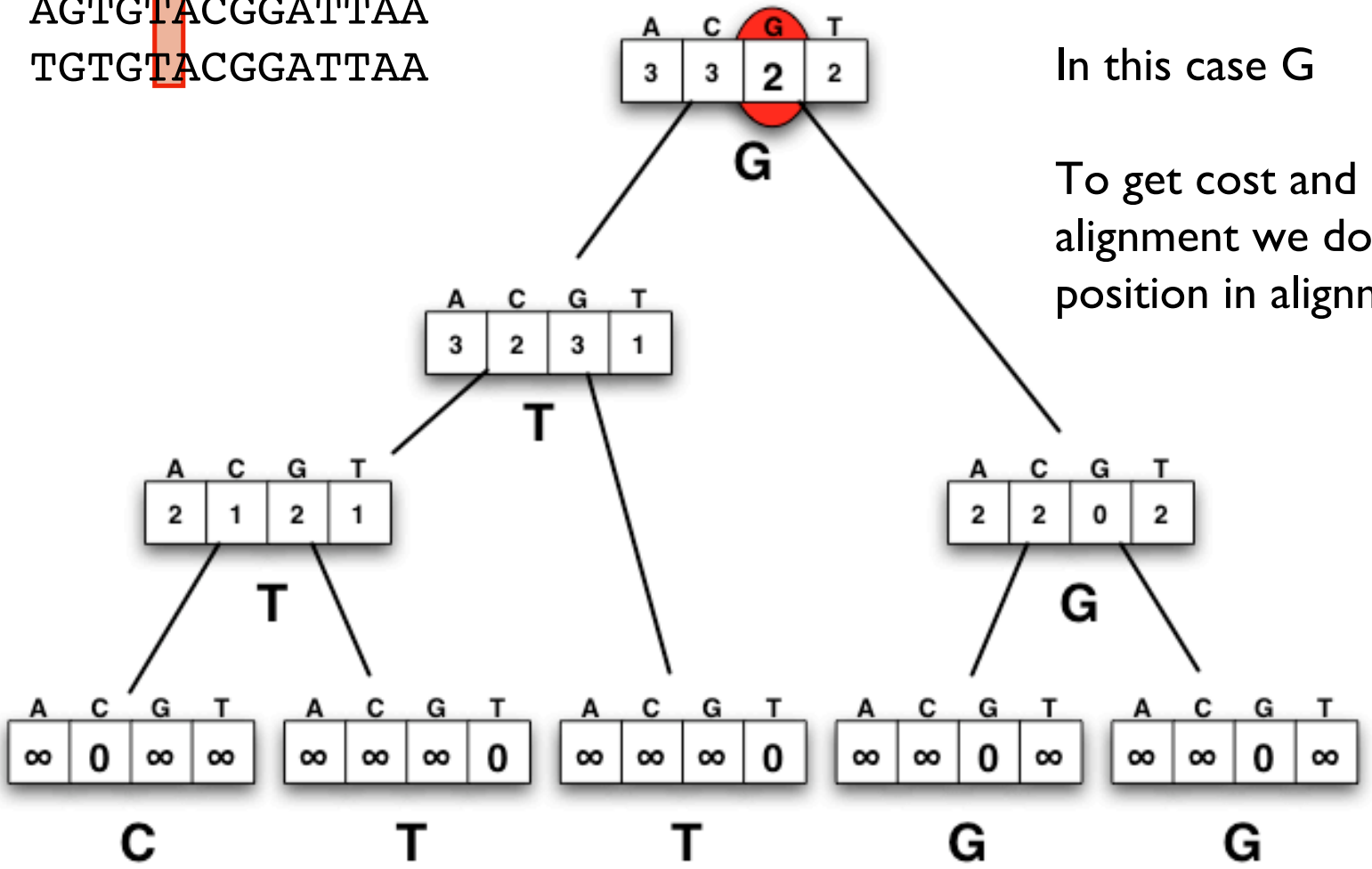
Does trees have to be rooted to use this algorithm?

For any cell (four per position in alignment per node) keep pointers to min from left and min from right.

Sankoff + Cedergren (weighted parsimony)

Cost and ancestral seq at position 4

```
1: AGTCTACGGATTAT
2: AGTTTACGGATTAT
3: TGTTTACGGATTAT
4: AGTGTACGGATTAA
5: TGTGTACGGATTAA
```



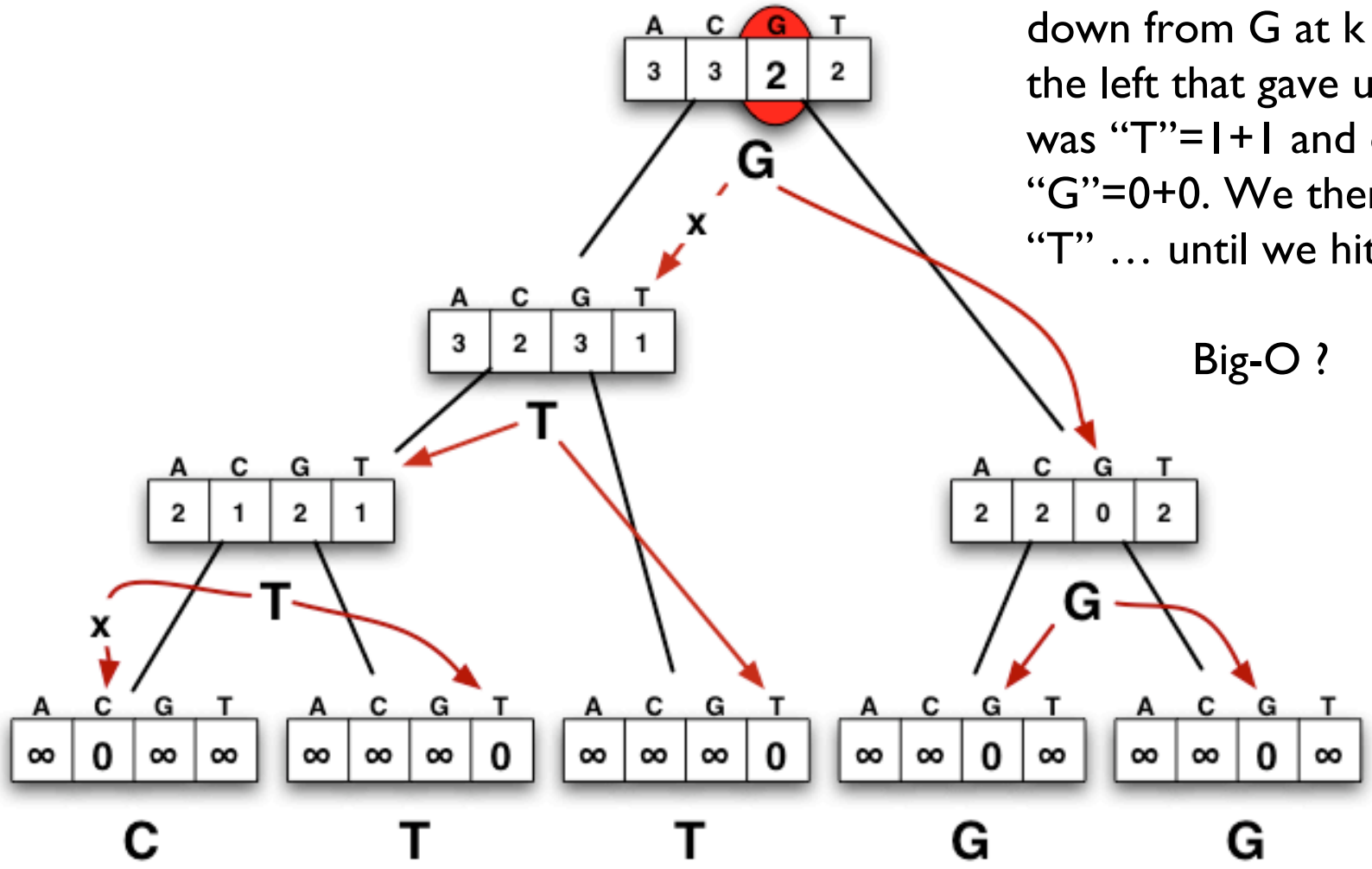
Keep filling until k = 2n-1 (root)

Minimal cost of tree in min $S_k(a)$ at this node (k=2n-1) = 2

In this case G

To get cost and seq for whole alignment we do this for each position in alignment.

Sankoff + Cedergren (weighted parsimony)

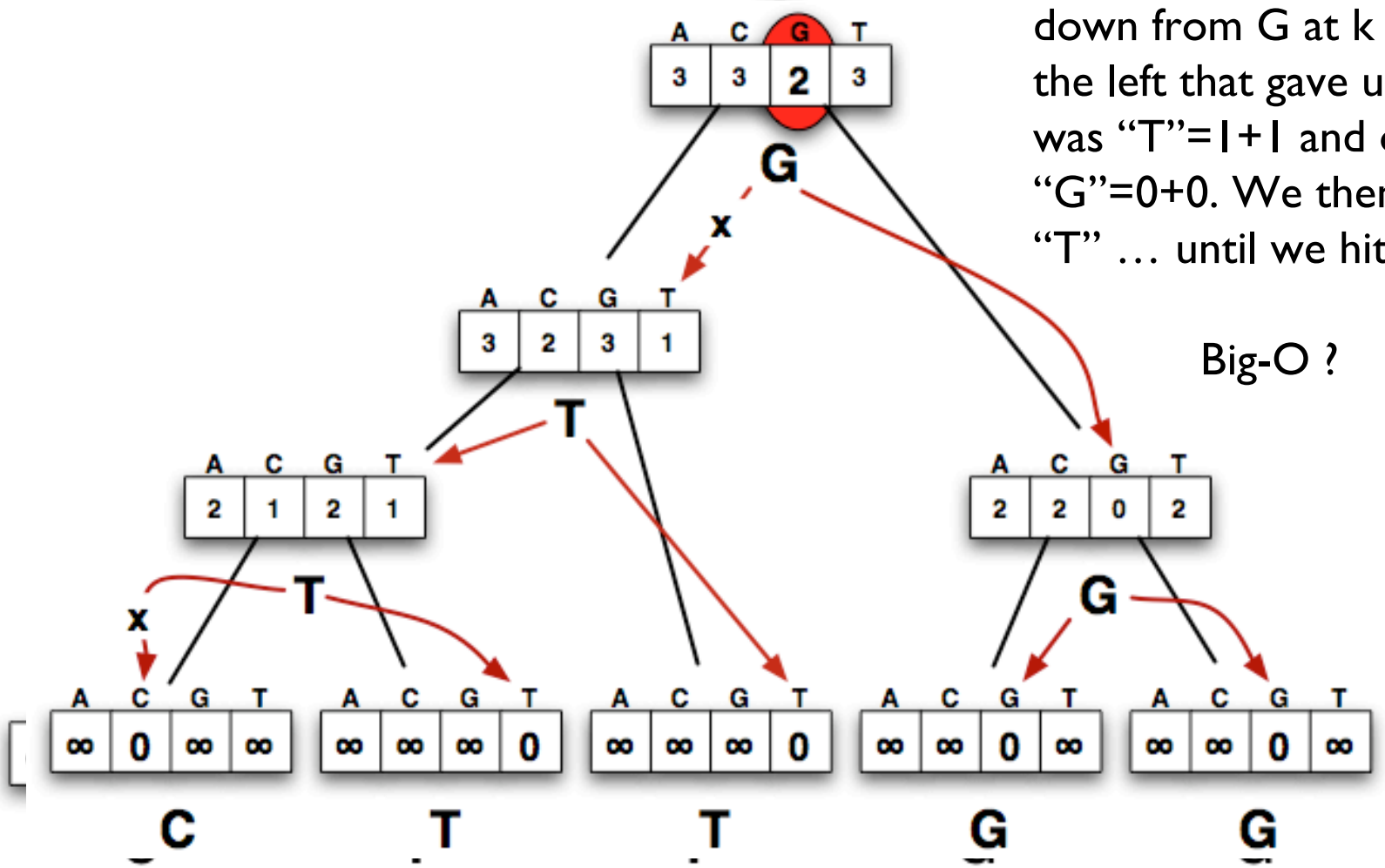


"x" indicates a mutation in our reconstructed tree.

We trace our left and right pointers down from G at k = 2n-1. The cell on the left that gave us our min value was "T"=1+1 and on the right "G"=0+0. We then trace from that "T" … until we hit leaf nodes.

Big-O ?

Sankoff + Cedergren (weighted parsimony)



"x" indicates a mutation in our reconstructed tree.

We trace our left and right pointers down from G at k = 2n-1. The cell on the left that gave us our min value was "T"=1+1 and on the right "G"=0+0. We then trace from that "T" … until we hit leaf nodes.

Big-O ?

Next week's reading

- Ch. 8 BSA
- MAIN:
  Large punctual contribution of speciation to evolutionary divergence. Science 314:2006, p. 119
- Optional:
  Branch and bound algorithms to determine minimal evolutionary trees. Hendy + Penny. Mathamatical Biosciences 59:277-290(1982)