# Lecture 11

- Alignment errors
  - ‘Gap attraction’
- Multiple sequence alignment
  - Evolutionary trees
  - Higher-dimensional edit graphs
  - Progressive alignment

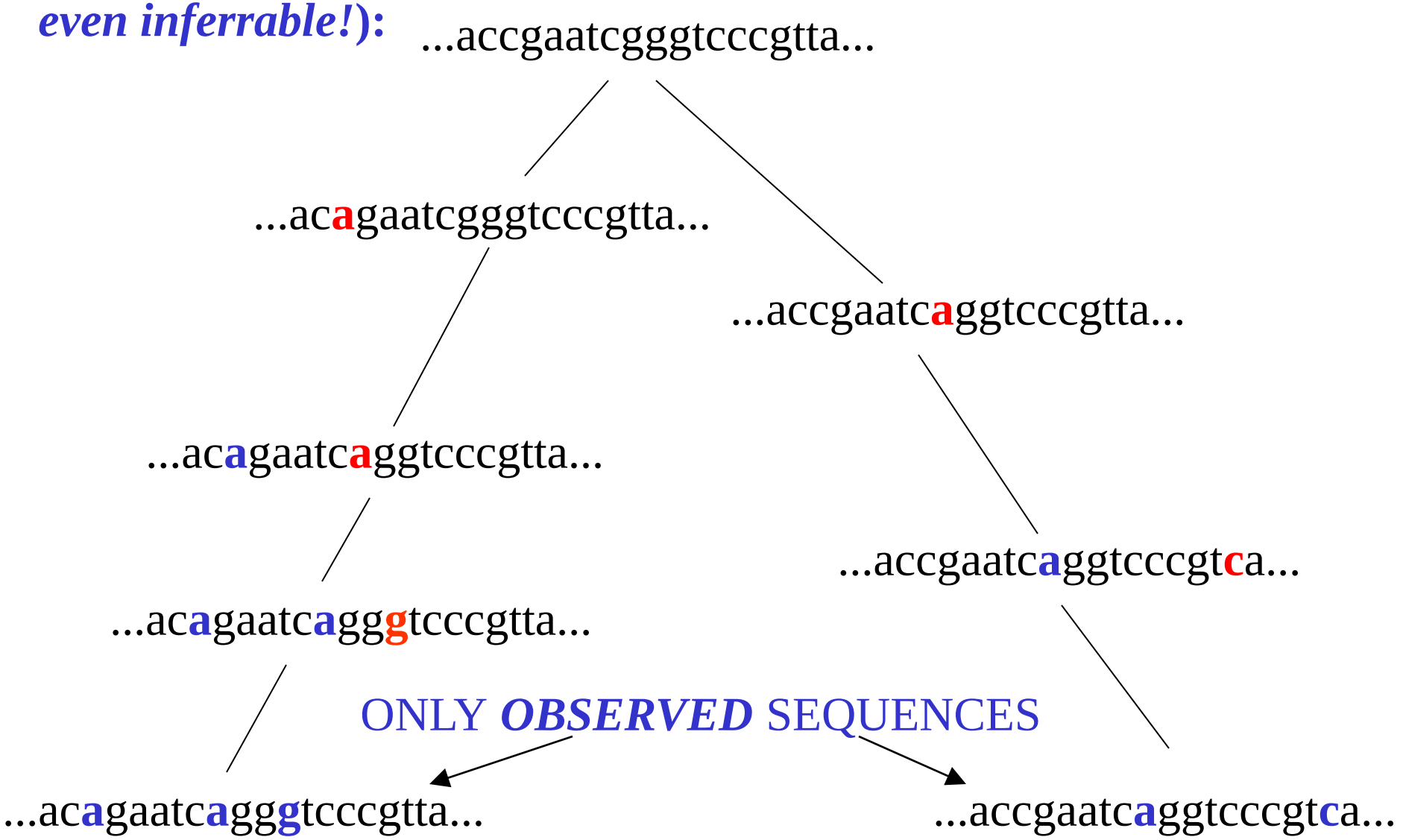
- Major motivation for sequence alignment:
  - illuminating ***mutation*** and ***selection*** in *evolutionarily related sequences*

- *Accuracy* of alignment matters!

**(Observed) ALIGNMENT:**
**(*may not be unique*!)**

```
...acagaatcagggtcccgtta...
...accgaatcagg-tcccgtca...
```

**(Unobserved) MUTATION HISTORY (*in general, this is not even inferrable!*):**



# Complications

- Parallel & back mutations

  $\Rightarrow$ estimating total # of mutations requires statistical modelling

- Insertion/deletion, & segmental mutations

  $\Rightarrow$ finding the correct alignment can be problematic (‘gap attraction’)

  -- even in closely related sequences!

# Gap attraction

- If *true* alignment is

```
...acagaatcagggtcc-gtta...
...acagaatcagg-tcccgtta...
```

*reported* (maximum-scoring) alignment will be

```
...acagaatcagggtccgtta...
...acagaatcaggtccgtta...
```

(2 mismatches cost less than 2 indels)

- Similarly, if *true* alignment is

```
...acagaatcagggtcccgtta...
...acagaatcagg-tcc-gtta...
```

*reported* alignment will be

```
...acagaatcagggtccgtta...
...acagaatcagg--tccgtta...
```

(size-2 indel + mismatch cost less than 2 size-1 indels)

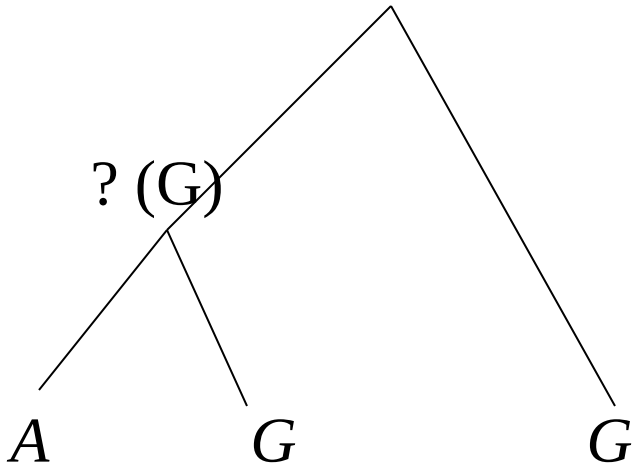
- This is an issue even for highly similar genomes!
  - But worse with increasing divergence
- Ideally, report alignments with local indications of uncertainty
  - or at least, several alignments with varying alignment penalties

  but this is almost never done
- Problem is ameliorated with multiple alignments

# Multiple alignments

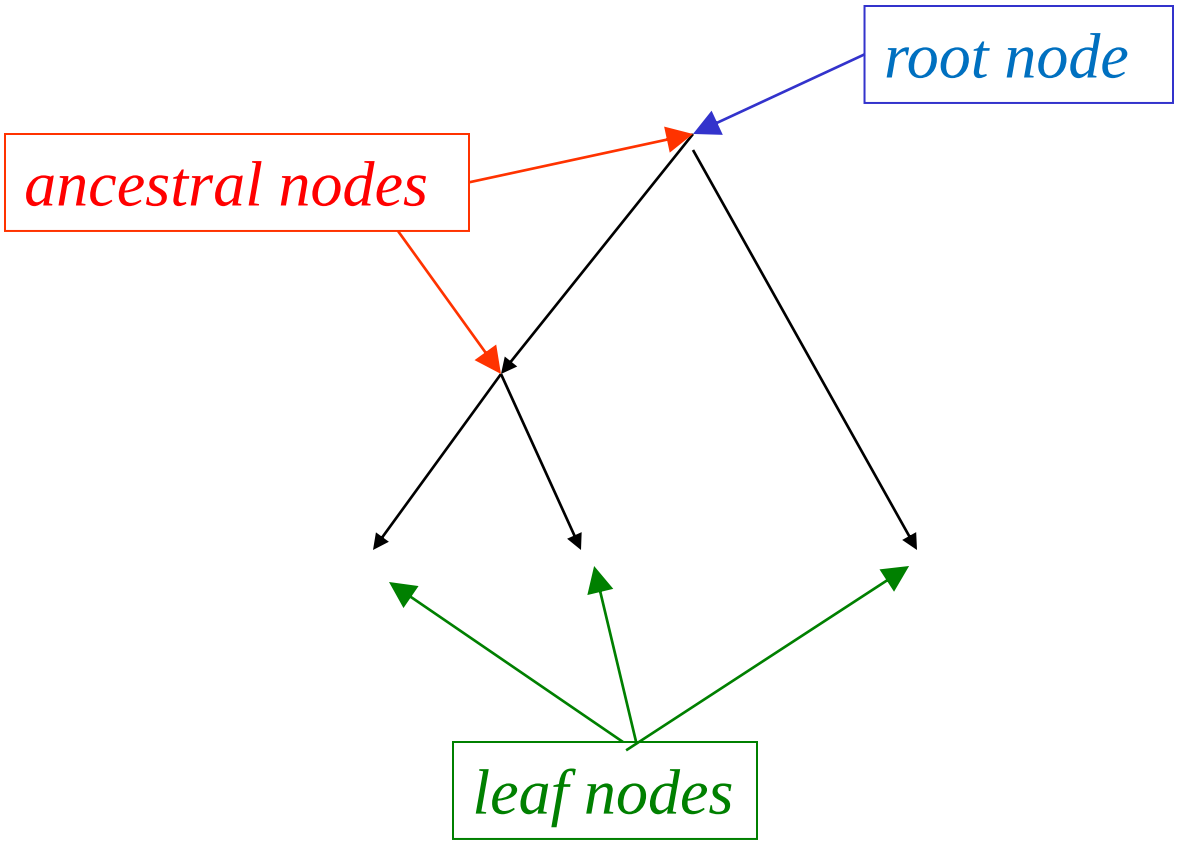
- More sequences =>
  - (potentially) more accurate alignments
  - better *resolution* of mutations, selection
- Need > 2 sequences to ***polarize*** mutations

- An evolutionary *tree* relates the sequences!


? (G)
A
G
G

# Evolutionary trees

- Binary tree with
  - $n_{leaf}$ ***leaf nodes*** (observed individuals)
  - $n_{anc}$ ***ancestral nodes*** (unobserved)
- Each ancestral node has two descendants ('left' and 'right'); leaves have none
- # edges:
  - # edge *starts* = $2\ n_{anc}$
  - # edge *ends* = $n_{leaf} + n_{anc} - 1$ (every node except root)
  - $\therefore\ 2\ n_{anc} = n_{leaf} + n_{anc} - 1$
  - $n_{anc} = n_{leaf} - 1$, # edges = $2\ n_{leaf} - 2$


root node
ancestral nodes
leaf nodes

- Want to compute *probabilities* of observed leaf sequences, given tree
- Requires considering possible evolutionary histories
  - i.e. sequences at ancestral nodes
  - # choices grows exponentially in both $n_{anc}$ and sequence length !!
- and a probability model for change along edges

# Mutational model for tree

- Will assume independent evolution at each sequence position
  - Doesn't allow for context effects (e.g. CpG hotspots!)
- Mutations along an edge $e$:

  $P_e(s \mid r)$ = prob a residue $r$ at beginning of $e$ is $s$ at end
- 'Background' residue freqs at the root:

  $P_{root}(r)$

- Usual assumptions:
  - (for DNA ) $P_e(s\text{^} \mid r\text{^}) = P_e(s \mid r)$
    - ( ^ = complementary nuc)
    - so each $P_e$ has 6 independent params
  - A *single*, *reversible*, *infinitesimal* (~per small time unit) mutation model $P_{inf}$ applies across entire tree
    - $P_e = (P_{inf})^t$ where $t$ = time along $e$
    - Reversibility implies root can't be uniquely placed

# Probability calculations on tree

- Given:
  1. a set of observed residues at the leaves

     ( a gap-free alignment column of the sequences)
  2. $\{P_e(s \mid r)\}$ and $\{P_{root}(r)\}$

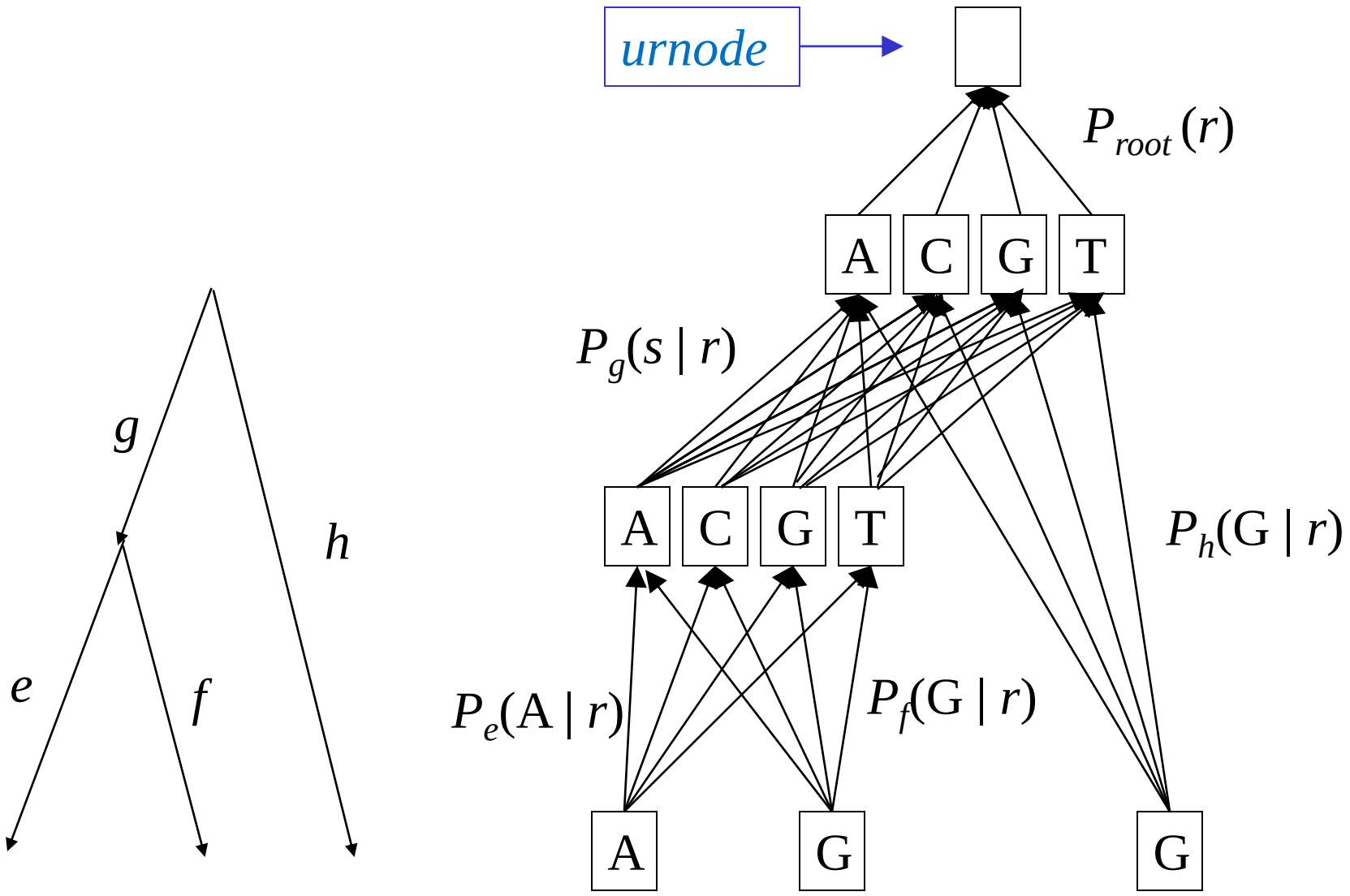
compute prob of observed residues

- Still exponentially many (in $n_{anc}$) possibilities for ancestral residues!
- But can use dynamic programming on a WDAG …

# Evolut tree → WDAG

- Each ***ancestral node*** in tree becomes **4** nodes in WDAG
  - labelled with the 4 nucs
- ***leaf nodes*** remain unchanged
  - labelled with observed nuc
- Two nodes in WDAG are connected by an ***edge***

  if corresponding tree nodes are (but reverse direction)
  - weight = $P_e(s \mid r)$ where $e$ = tree edge, $r$, $s$ = node labels
- 'urnode'
  - unlabelled
  - 4 edges coming from the 4 root nodes
  - weights = $P_{root}(r)$


urnode
$P_{root}(r)$
A C G T
$P_g(s \mid r)$
g
h
e
f
A C G T
$P_h(\mathrm{G} \mid r)$
$P_e(\mathrm{A} \mid r)$
$P_f(\mathrm{G} \mid r)$
A
G
G

- Size of WDAG is linear in $n_{leaf}$
  - # nodes: $n_{leaf} + 4\, n_{anc} + 1$
  - # edges: $4\, n_{leaf} + 16\, (n_{anc} - 1) + 4$

- Edges in tree point *down*; in WDAG, *up*
  - so WDAG ‘parents’ are *below*

- Compute overall *probability* of leaf residues (nucleotides) by *dynamic programming* on WDAG:

- Let, for each node $v$, $p(v)$ = prob of leaf nucs *below* $v$ (i.e tree-descendants, or WDAG-ancestors, of $v$), given $v$'s nuc

  $p_{left}(v)$ = prob of leaf nucs *below* and to *left*

  $p_{right}(v)$ = prob of leaf nucs *below* and to *right*

  then $p(v) = p_{left}(v)\, p_{right}(v)$

- Compute these values node-by-node, visiting (WDAG-)parents before children:
  - *starting* at leaf nodes (setting $p(v) = 1$), *ending* at urnode

$p_{left}(v) = \sum_{left-u} w(u,v)p(u)$ where

  - $u$ ranges over parent nodes to the left
  - $w(u,v)$ = weight on edge from $u$ to $v$
    (= mutation prob from $v$ to $u$)

Similarly for $p_{right}(v)$

$p(v) = p_{left}(v)\, p_{right}(v)$

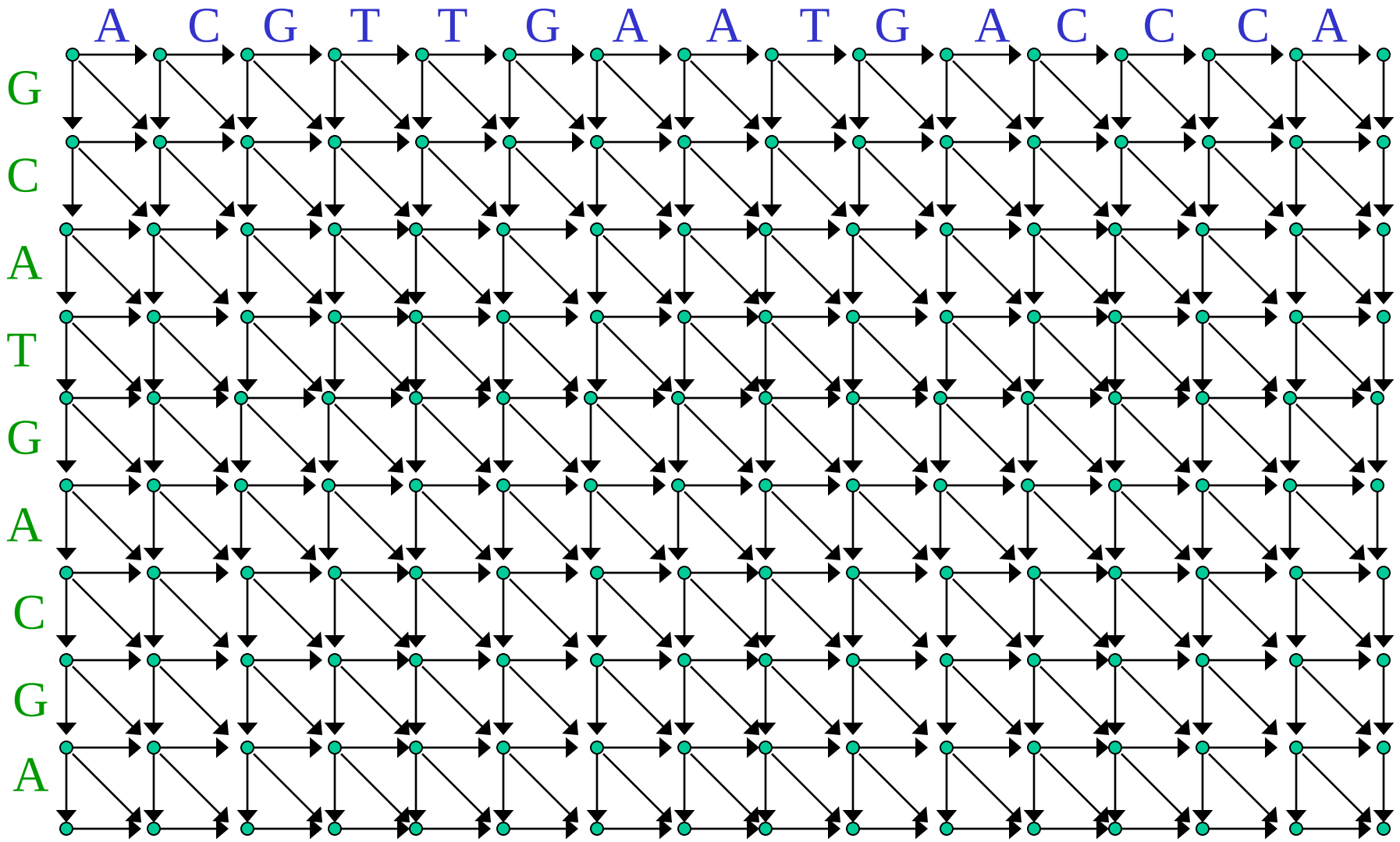
  - For $v$ = urnode, view *all* parents as being to 'left' and $p(v) = p_{left}(v)$

- $p$(urnode) = probability of the observed leaf nucs

# Scoring multiple alignments

- Can now define LLR scores for alignment columns:
  - log((prob of col | $P_e$ model) / (prob of col | background))
- How do we get $P_e$ ?
  - Given alignment, can estimate using 'forwards-backwards' approach (cf linear-space algorithm, & HMMs)
- But need scores to get alignment!
  - Possible iterative procedure:
  - crude alignment → $P_e$ → scores → better alignment etc
- In current practice, use ad hoc (easy to compute) scores, e.g. sum of pairwise scores
  - But still want $P_e$ for its own sake!

# The *Edit Graph* for a Pair of Sequences

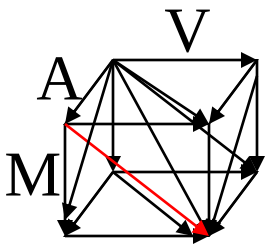


# Multiple Alignment via Dynamic Programming

- Higher dimension edit graph
  - each dimension corresponds to a sequence; co-ordinates labelled by residues
  - Each edge corresponds to aligned column of residues (with gaps).
  - Can put arbitrary weights on edges; in particular,
    - can make these correspond to probabilities under an evolutionary model (Sankoff 1975).
  - implicitly assumes independence of columns
- Highest weight path through graph again gives optimal alignment

# Generalization to Higher Dimension

Each “cell” in 3-dimensional case looks like this:




Each edge projects onto a gap or residue in each dimension, defining an alignment column; e.g. red edge defines

V

–

M

- # edges & # vertices are proportional to product of sequence lengths.
  - For $k$ sequences of size $N$, is of order $O(N^k)$
    - impractical even for proteins ($N \sim 300$ to 500 residues) if $k > 5$:

$$300^5 = 2.4 \times 10^{12}$$

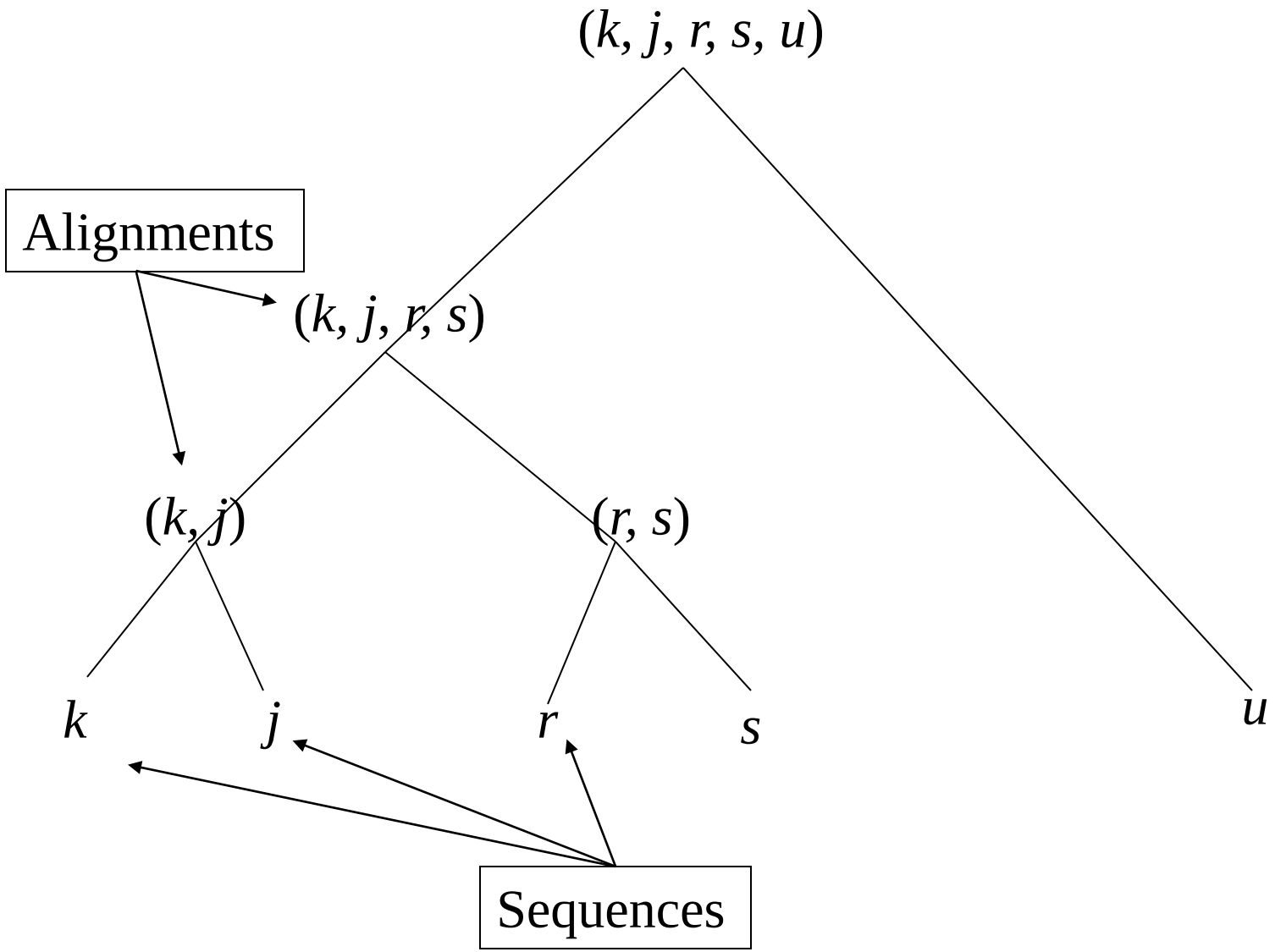
# Multiple alignments: paths in huge WDAGs

- To find high-scoring paths, need to
  - reduce size of graph
  - restrict allowed weighting schemes, and/or
  - sacrifice optimality guarantees
- Durbin *et al.* discuss methods implementing these ideas:
  - Hein
  - Carillo-Lipman
  - progressive alignment (e.g. Clustal)
- HMMs provide nice (but not guaranteed optimal) approach for constructing multiple alignments

# Progressive alignment

- Simplest version: align one sequence (the reference) to each of the others, pairwise; construct multiple alignment from that.
- More generally, progressively align *pairs* of (*sequences or*) *alignments*, using a *guide tree*
  - Tree may reflect evolution, or sequence quality
  - Will tend to be more accurate
- Revise gaps
  - correct errors due to gap placement & gap attraction

# Guide Tree



- Complexity: $N^2 \times (n - 1)$ where
  - N = seq length, $n$ = # seqs

  instead of $N^n$
- (does not count gap correction)