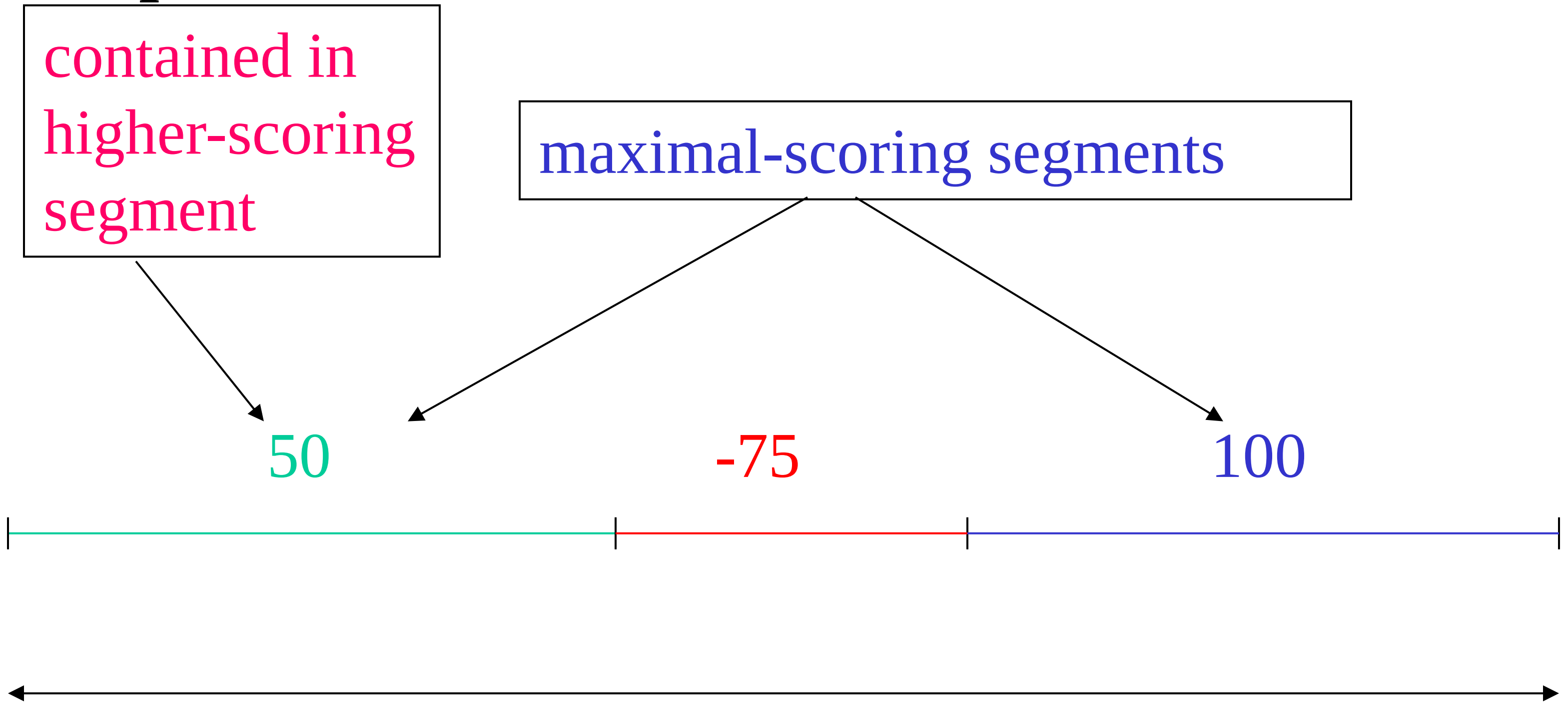
Today’s Lecture

- Finding multiple high-scoring segments
- “D-segments”
  - relationship to 2-state HMMs
- Sequence alignment & evolution

- A *maximal(-scoring) segment* I is one such that
  - *P1:* no subsegment of I has a higher score than I
  - *P2:* no segment properly containing I satisfies *P1*
- Example:



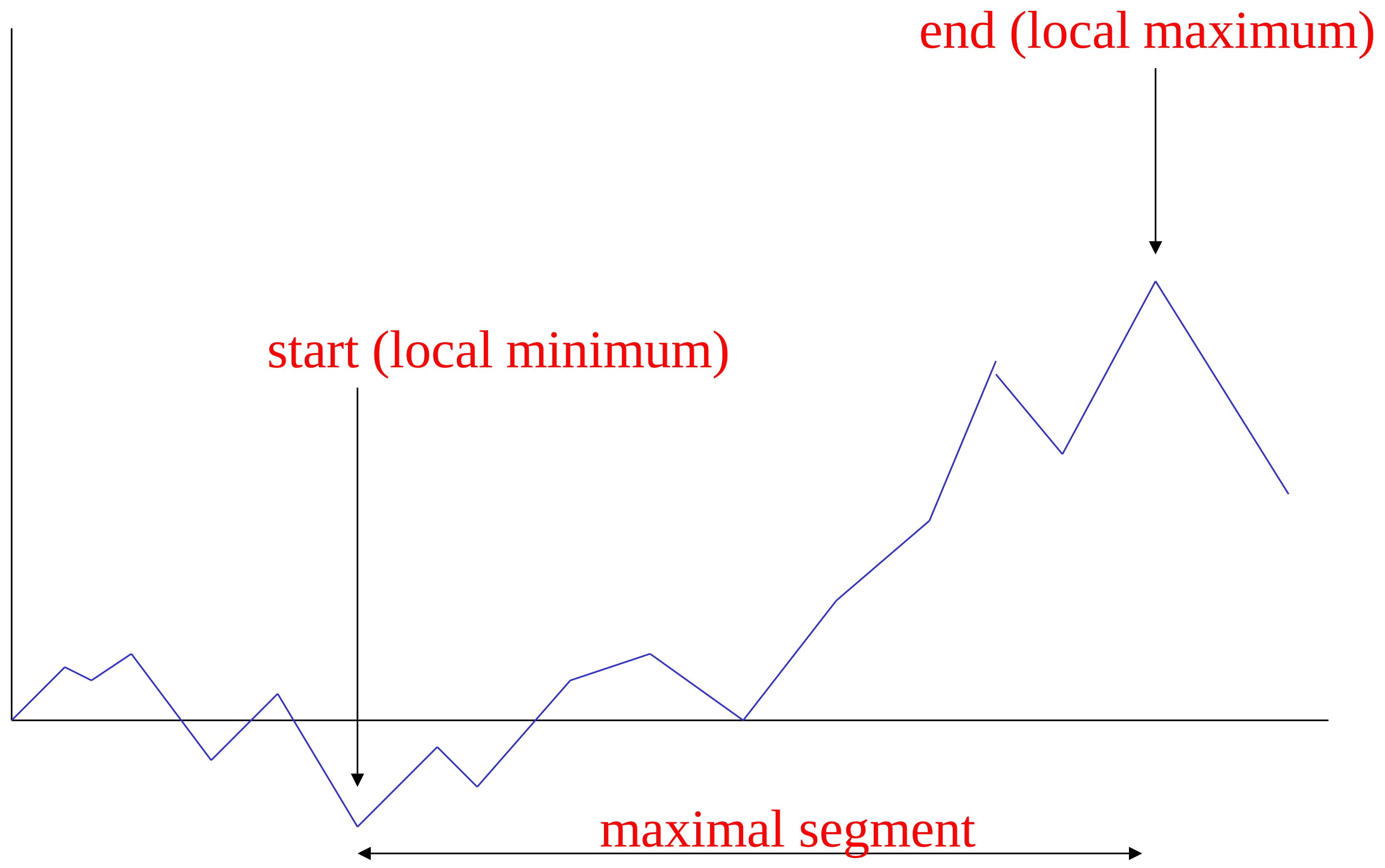
score = 75, but does not satisfy *P1*

- *Problem*: given $S > 0$, find all maximal segs of score $\geq S$
- Segments are *paths* in a linked-list WDAG with $N+1$ vertices and N edges
- *Highest weight path* is found by dynamic programming;

  in (pseudo-)pseudocode:

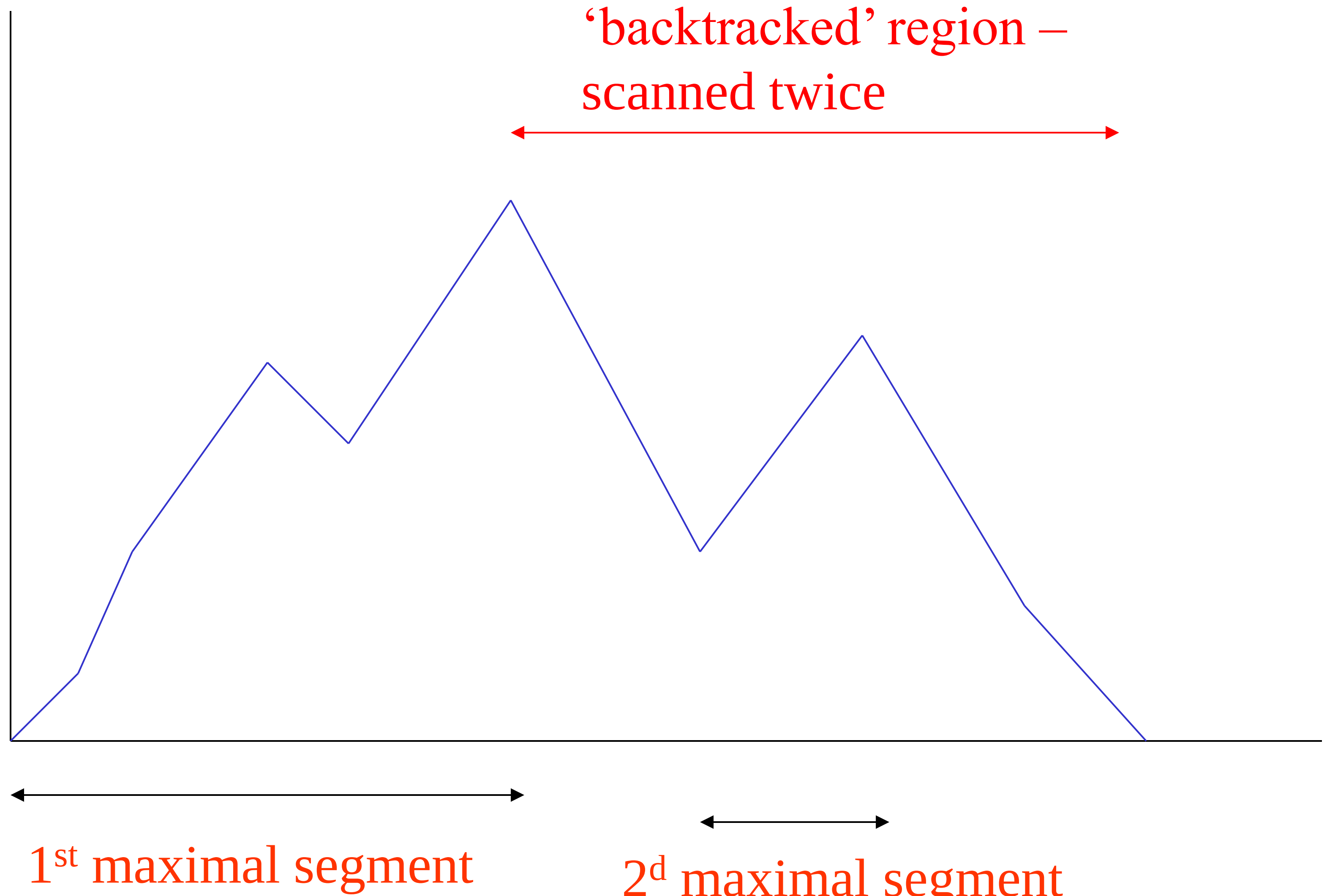
```
cumul = max = 0;  start = 1;
for (i = 1; i ≤ N; i++)  {
    cumul += s[i];
    if (cumul ≤ 0)
        {cumul = 0;  start = i + 1;}  /* NOTE RESET TO ZERO */
    else if (cumul ≥ max)
        {max = cumul;  best_end = i;  best_start = start;}
}
if (max ≥ S) print best_start, best_end, max
```

Maximal segments – from cumulative score plot (*without* 0 resets)



- Can find *all* maximal segs of score ≥ S using following practical (but non-optimal) algorithm:

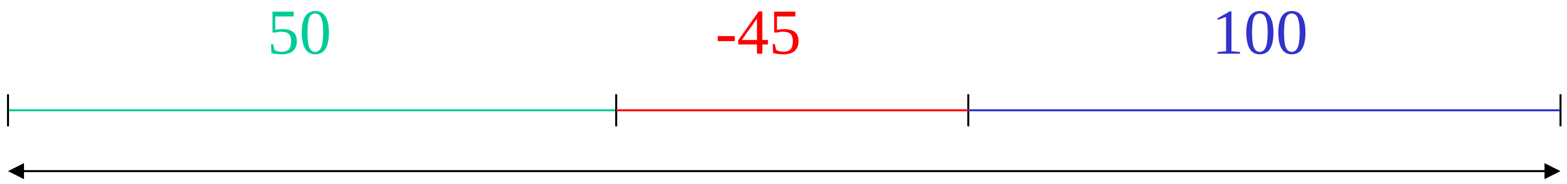
```
cumul = max = 0;  start = 1;
for (i = 1; i ≤ N; i++) {
    cumul += s[i];
    if (cumul ≥ max)
        {max = cumul; end = i;}
    if (cumul ≤ 0 or i == N) {
        if (max ≥ S)
            {print start, end, max;   i = end; }  /* N.B. MUST BACKTRACK! */
        max = cumul = 0;  start = end = i + 1;
    }
}
```


‘backtracked’ region –
scanned twice
1st maximal segment
2d maximal segment

- In worst case this is $O(N^2)$ (because of backtracking),
  - but in practice usually $O(N)$ because a given base is usually traversed only a few times
- Ruzzo-Tompa algorithm guarantees $O(N)$

- undesirable aspect of maximal segments as so defined:
  - single maximal seg may contain *two* (or more) high-scoring regions, separated by significant negative-scoring regions
  - i.e. two possibly biologically distinct target occurrences get merged into one maximal segment

- Example:

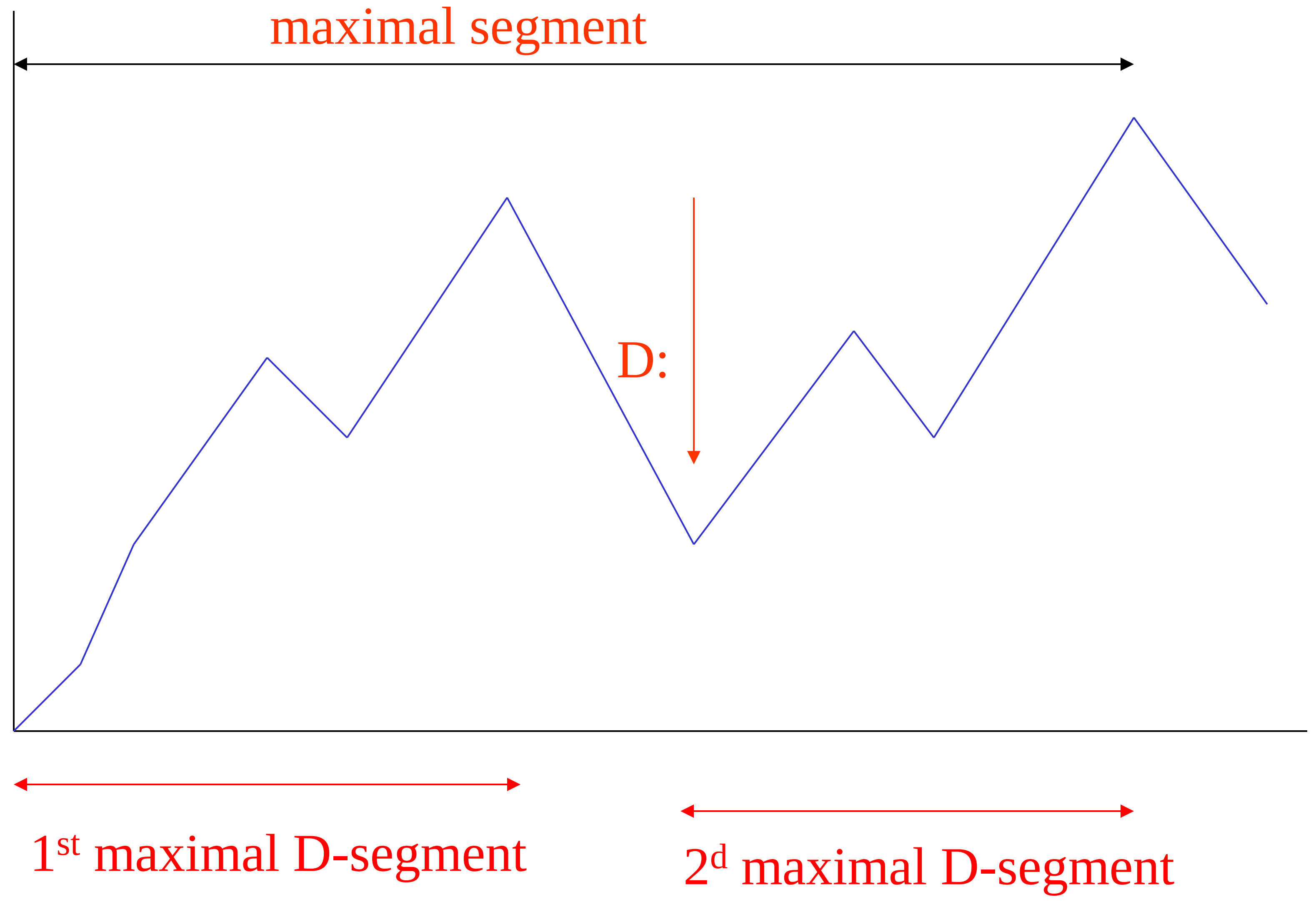


now entire segment has score = 105, & satisfies *P1* ***and*** *P2*

A better problem!

- to avoid this, have max allowed ‘dropoff’ D < 0
- *D-segment* is segment without any subsegments of score < D
- *maximal D-segment* is D-segment I such that
  - *P1:* no subsegment of I has higher score than I
  - *P2:* no D-segment properly containing I satisfies *P1*
- Problem: given *S* (≥ –D), find all maximal D-segs of score ≥ S
  – (algorithm fails if S < –D)

Maximal D-segments



- $O(N)$ algorithm to find all maximal D-segs:

```
cumul = max = 0; start = 1;
for (i = 1; i ≤ N; i++) {
    cumul += s[i];
    if (cumul ≥ max)
        {max = cumul; end = i;}
    if (cumul ≤ 0 or cumul ≤ max + D or i == N) {
        if (max ≥ S)
            {print start, end, max; }
        max = cumul = 0; start = end = i + 1; /* NO BACKTRACKING
            NEEDED! */
    }
}
```

- So *more biologically relevant* problem is also *computationally simpler*!
- what are appropriate S and D?
  - mainly an empirical question (based on known examples); altho
    - interpretation via 2-state HMM (next slide) can be useful
    - Karlin-Altschul theory tells when they are ‘statistically significant’

D-segments & 2-state HMMs

- Consider 2-state HMM
  - states 1 & 2, transition probs a_{11}, a_{12}, a_{21}, a_{22}
  - observed symbols $\{r\}$, emission probs $\{e_1(r)\}$, $\{e_2(r)\}$
- Define

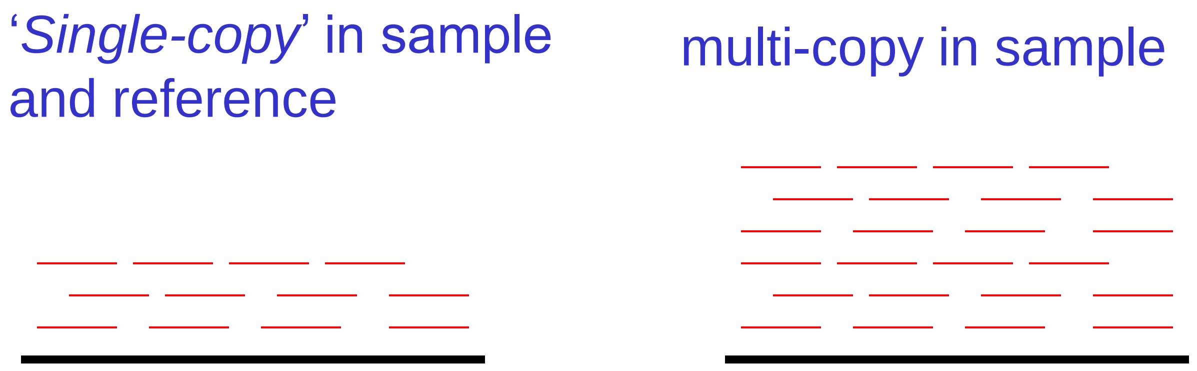
  scores $s(r) = \log(e_2(r)\, a_{22}/(e_1(r)\, a_{11}))$

  $S = -D = \log(a_{11}a_{22}/(a_{21}a_{12}))$
- Then if $S > 0$, the maximal D-segments in a sequence $(r_i)_{i=1,\, n}$ are the state-2 segments in the Viterbi parse.
- So via D-segment algorithm can get Viterbi parse in just one pass through the sequence!
- can allow for non-.5 initiation probs by starting cumul at non-zero value

- For HW 3, implement D-segment algorithm to find CNVs
  - data: next-gen read alignments to genome
  - observed symbols are counts of # read starts at each position (0, 1, 2, ≥ 3)
  - 2 states: non-dup, dup (dup has twice as many read starts per base as non-dup state)
  - emission probs given by Poisson dist’n with approp mean
  - transition probs set empirically

CNVs & Read Depth

- CNV = 'copy number variant'– e.g. region that is single copy in reference sequence but duplicated in sample
- One way to detect: map reads from sample onto reference, look for regions of atypical coverage depth



D-Segments – concluding remarks

- Powerful tool for analyzing ‘linear’ data
  - Single sequences (incl. motifs, numerical data)
  - Fixed alignment
- Strengths:
  - Very simple to program
  - Very fast, even for mammalian genomes
- Main limitation:
  - Only allows two types of segments (‘target’ and ‘background’)
    - Essentially a generalization of 2-state HMMs
    - multi-state HMMs are more flexible

Aligning sequences

- Major uses in genome analysis:
  - To find relationship between sequences from “same” genome
    - (still need to allow for discrepancies – due to errors/polymorphisms)

    E.g.

    - finding gene structure by aligning cDNA to genome
    - assembling sequence reads in genome sequencing project
    - NextGen applications: “Resequencing”, ChIPSeq, etc
  - To detect evolutionary relationships:
    - illuminates function of distantly related sequences under selection
    - finds corresponding positions in neutrally evolving sequence
      - to illuminate mutation process
      - helps find non-neutrally evolving (functional) regions

- Often we’re interested in details of alignment
  - (i.e. precisely which residues are aligned),

  but
- sometimes only interested in whether alignment score is large enough to imply that sequences are likely to be related

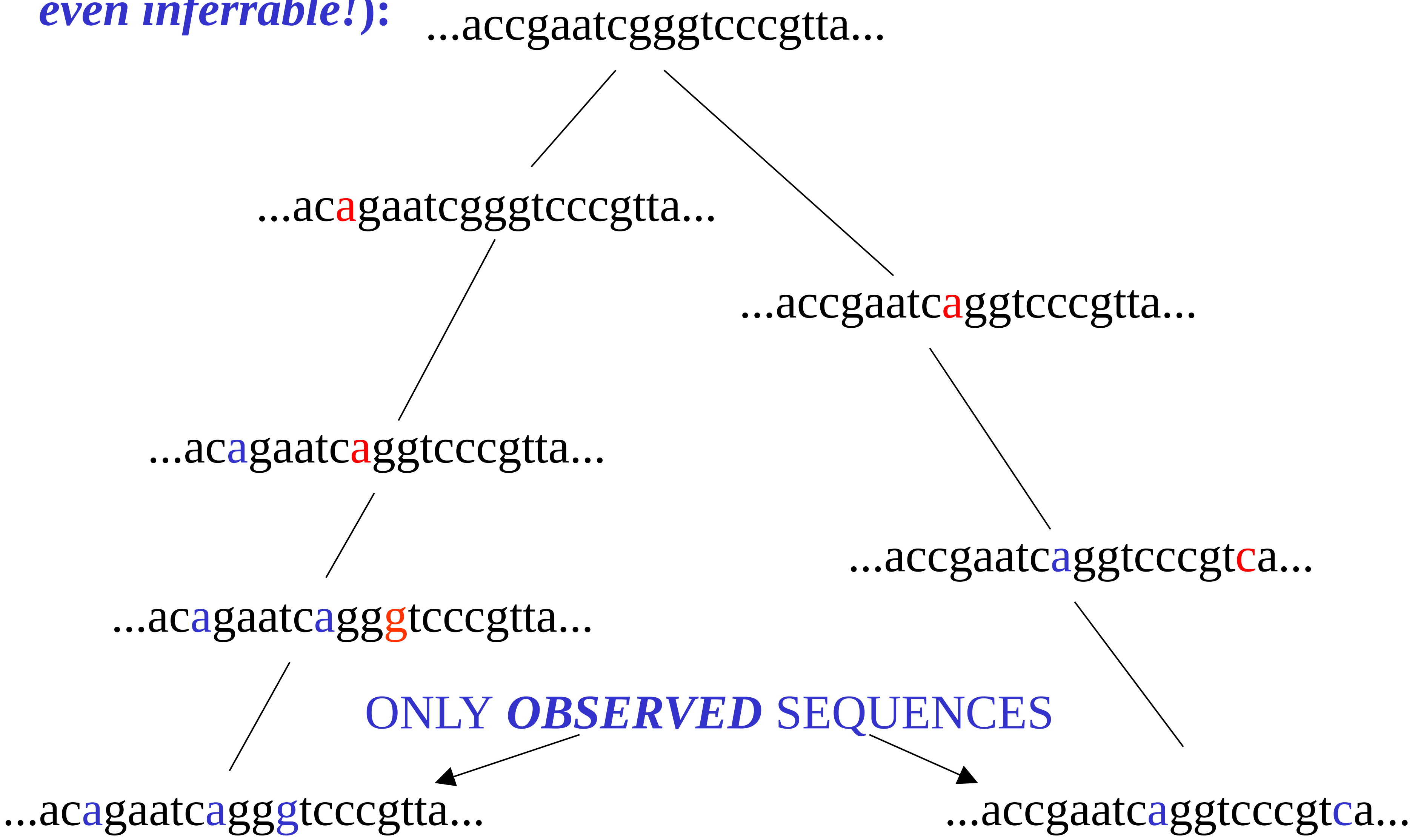
Sequences & evolution

- Similar sequences of sufficient length usually have a common evolutionary origin
  - i.e. are homologous
- For a pair of sequences
  - "% similarity" makes sense
  - "% homology" doesn't
- In alignment of two homologous sequences
  - differences mostly represent *mutations* that occurred in one or both lineages, but
  - Not all mutations are inferrable from the alignment

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

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

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


...accgaatcgggtcccgtta...
...acagaatcgggtcccgtta...
...accgaatcaggtcccgtta...
...acagaatcaggtcccgtta...
...accgaatcaggtcccgtca...
...acagaatcagggtcccgtta...
ONLY *OBSERVED* SEQUENCES
...acagaatcagggtcccgtta...
...accgaatcaggtcccgtca...

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!