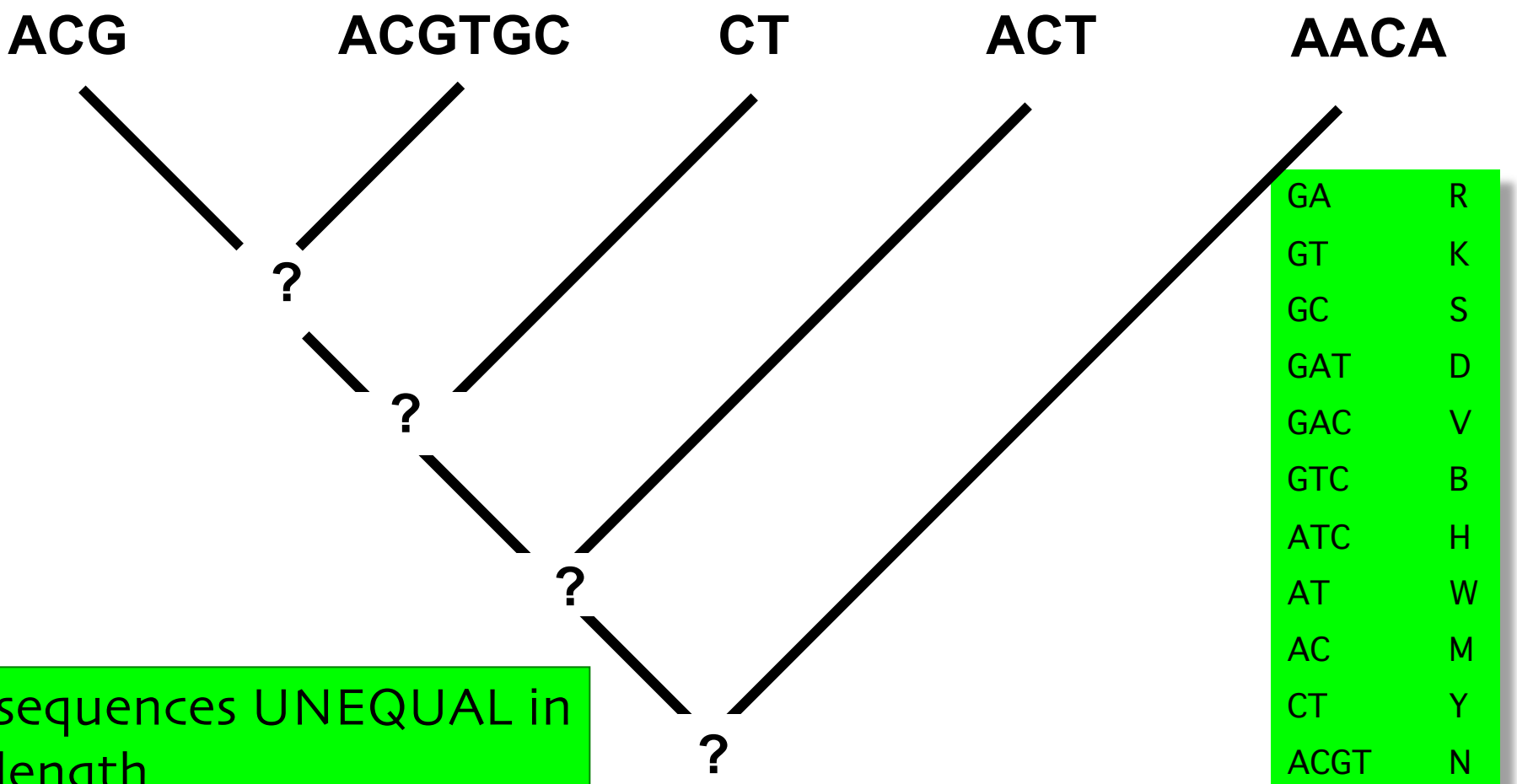
4.xi.

1. direct optimization

2. summary

Direct optimization (DO)

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT



sequences UNEQUAL in length

GA	R
GT	K
GC	S
GAT	D
GAC	V
GTC	B
ATC	H
AT	W
AC	M
CT	Y
ACGT	N
ACGT-	X

Sequence alignment

ONLY LOCAL comparison

BLAST, comparison of 2 sequences

MULTIPLE sequence alignment (MSA)

GLOBAL comparison

mostly done PROGRESSIVELY

computationally demanding

e.g. **CLUSTAL**

1) pairwise sequence comparison

2) pairwise distance matrix

3) **guide** tree formed based on this matrix

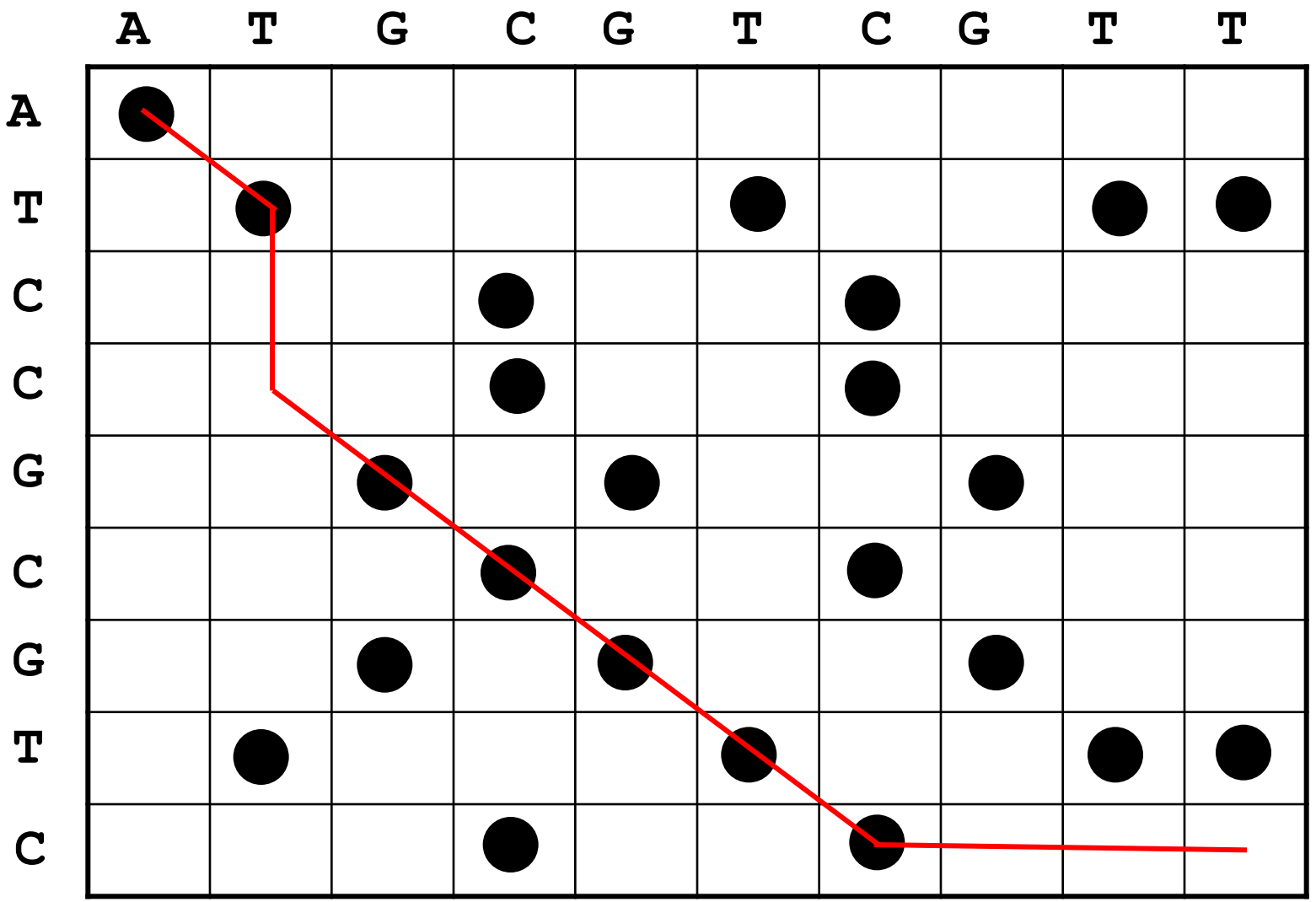
4) pair-wise alignment performed based on this tree
connecting seqs. of all the terminals included

in later developed programs (e.g. muscle, MAFFT) repeated cycles of these operations, with final round of edge refinement using sum of pairs

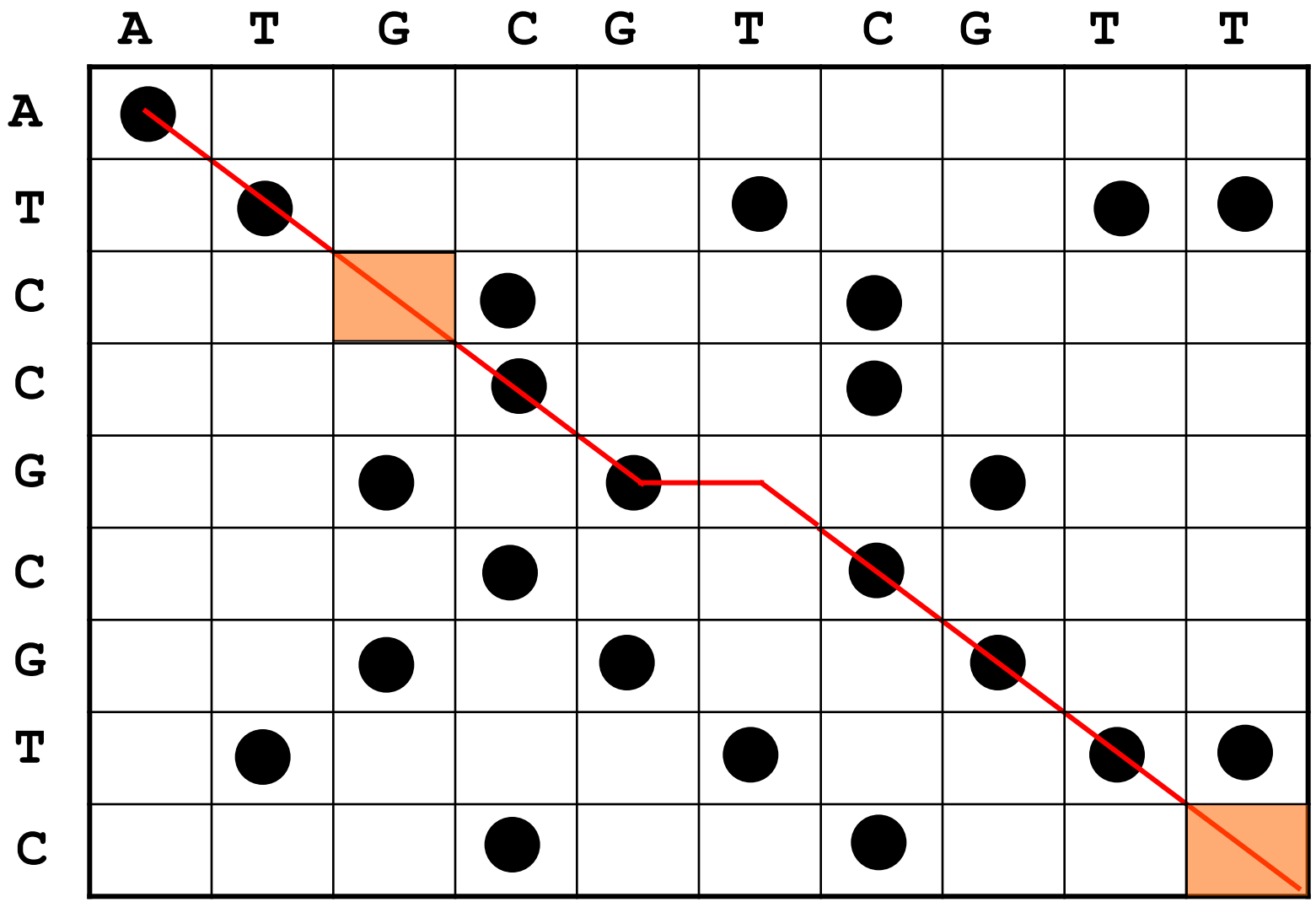
Pair-wise alignment

dot-plot method

simple visualization of matches




```
A T - - G C G T C G T T
| |       | | | | |
A T C C G C G T C - - -
```




```
A T G C G T C G T T
| |   *   |   |     |   |   |   *
A T C C G - C G T C
```

Needleman-Wunsch algorithm

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT

Needleman, S.B. & Wunsch, C.D. 1970. A general method applicable to the search of similarities in the amino acid sequence of two proteins. *J. Mol. Biol.* 48: 443-453.

4 basic steps:

1) laying out the alignment matrix

2 sequences define the axes

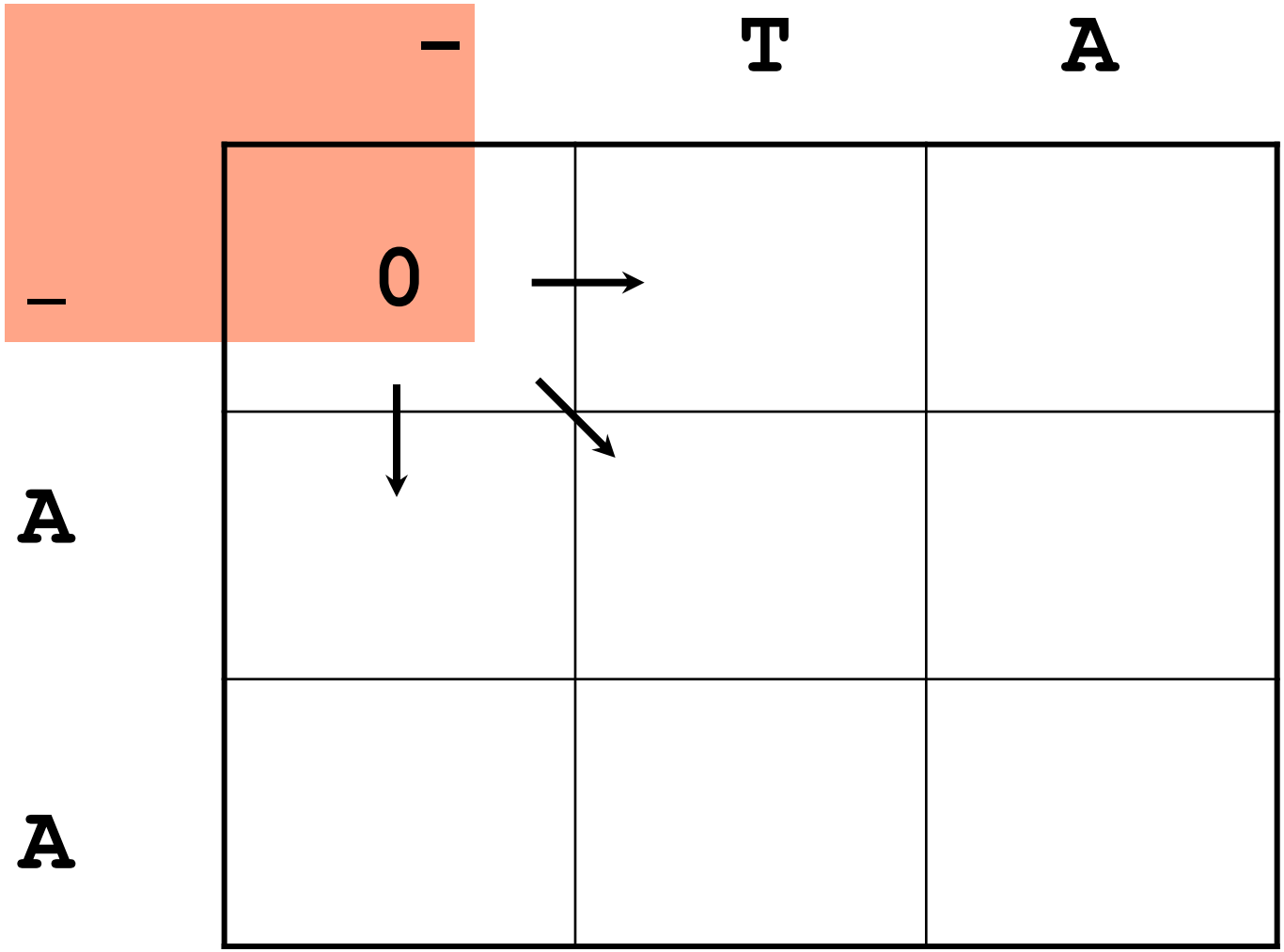
2) initiation of the matrix

3) wave-front update of the matrix elements

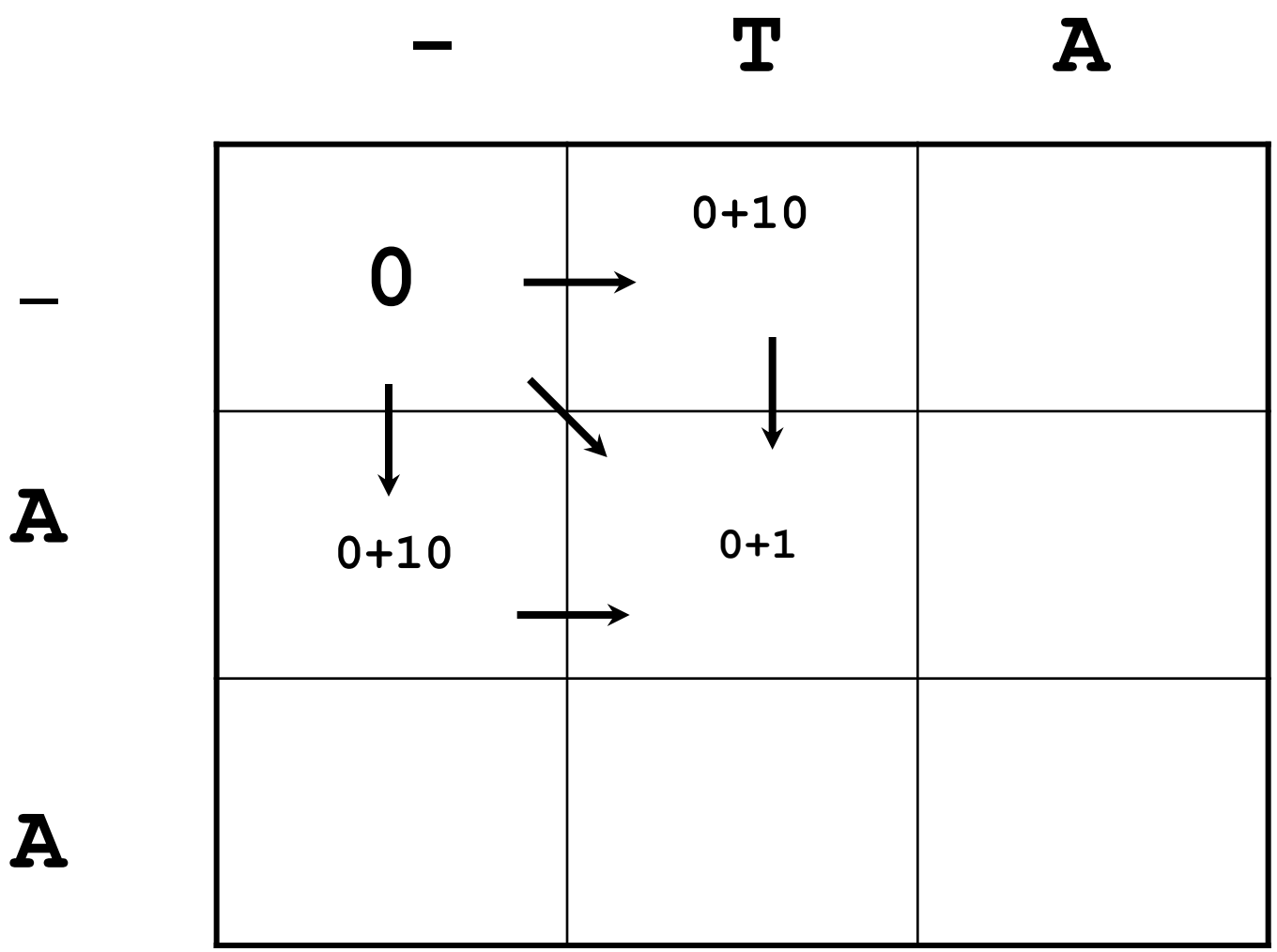
4) trace back

	-	T	A	A	A	T	T	G	C	A
-	0									
A		1	0	0	0	1	1	1	1	0
A		1	0	0	0	1	1	1	1	0
T		0	1	1	1	0	0	1	1	1
T		0	1	1	1	0	0	1	1	1
T		0	1	1	1	0	0	1	1	1
G		1	1	1	1	1	1	0	1	1
G		1	1	1	1	1	1	0	1	1
G		1	1	1	1	1	1	0	1	1
C		1	1	1	1	1	1	1	0	1
C		1	1	1	1	1	1	1	0	1
A		1	0	0	0	1	1	1	1	0

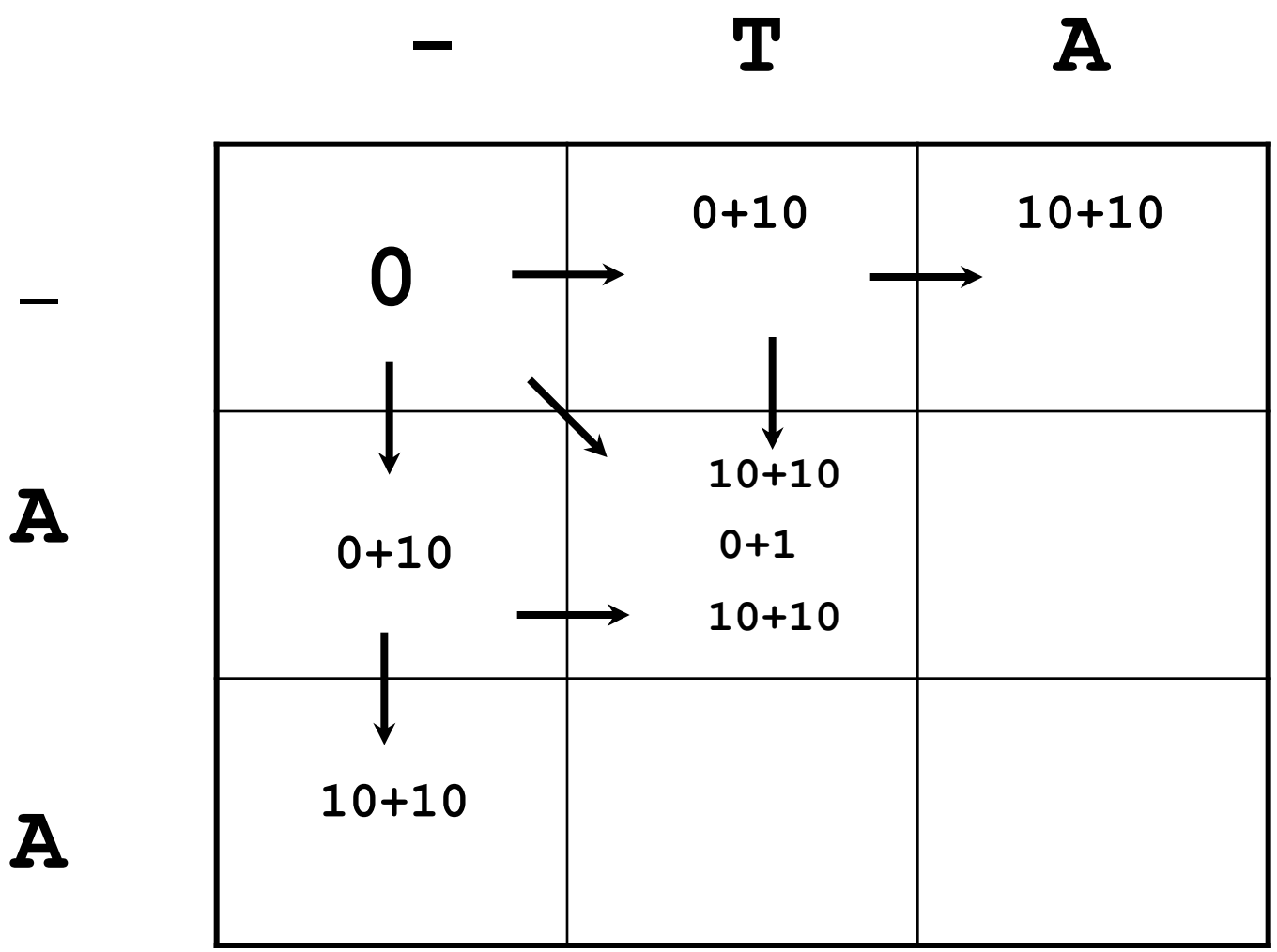
	-	T	A	A	A	T	T	G	C	A
-	0									
A		1	0	0	0	1	1	1	1	0
A		1	0	0	0	1	1	1	1	0
T		0	1	1	1	0	0	1	1	1
T		0	1	1	1	0	0	1	1	1
T		0	1	1	1	0	0	1	1	1
G		1	1	1	1	1	1	0	1	1
G		1	1	1	1	1	1	0	1	1
G		1	1	1	1	1	1	0	1	1
C		1	1	1	1	1	1	1	0	1
C		1	1	1	1	1	1	1	0	1
A		1	0	0	0	1	1	1	1	0



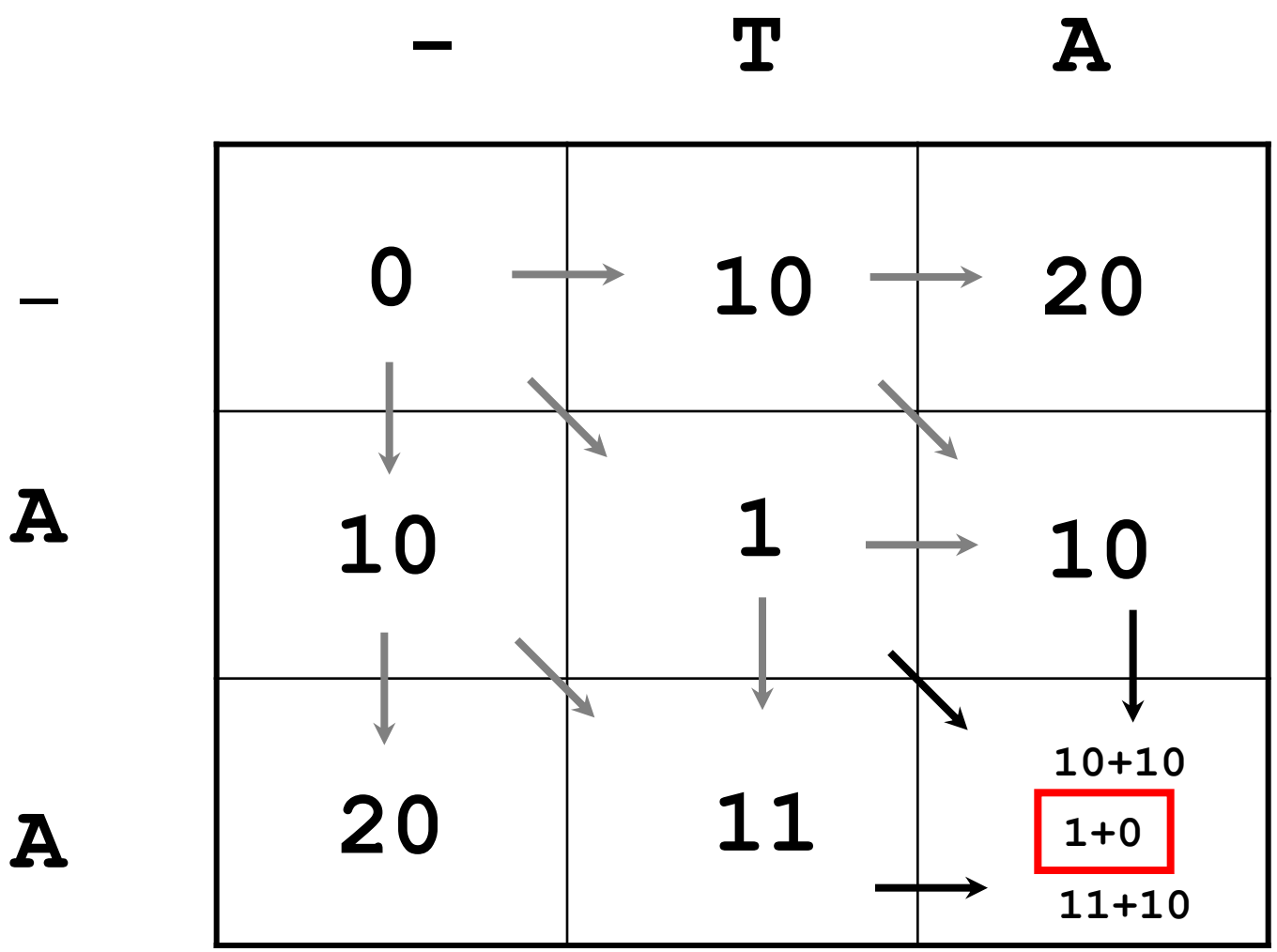
match = 0 substitution = 1 gap penalty = 10

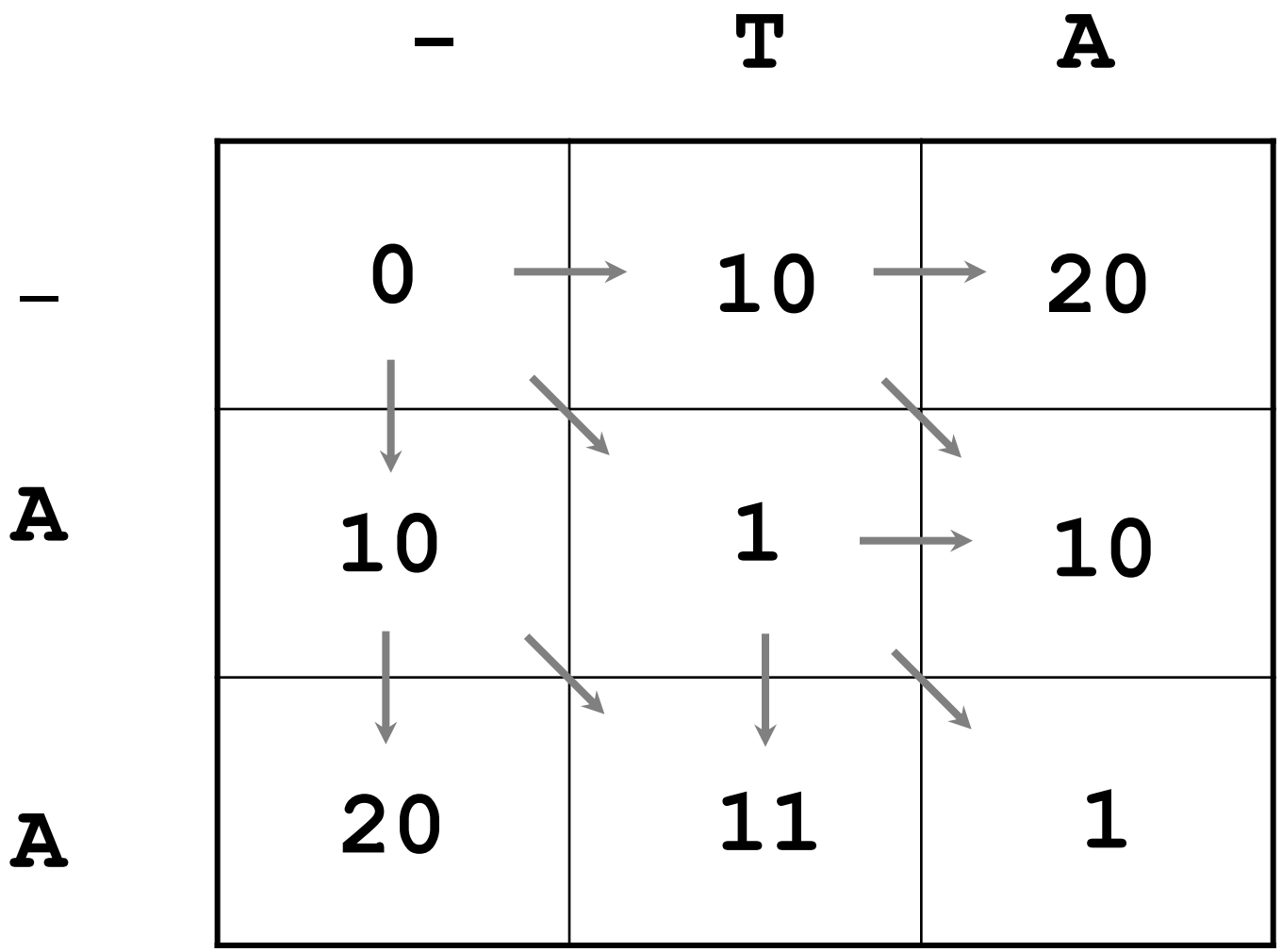


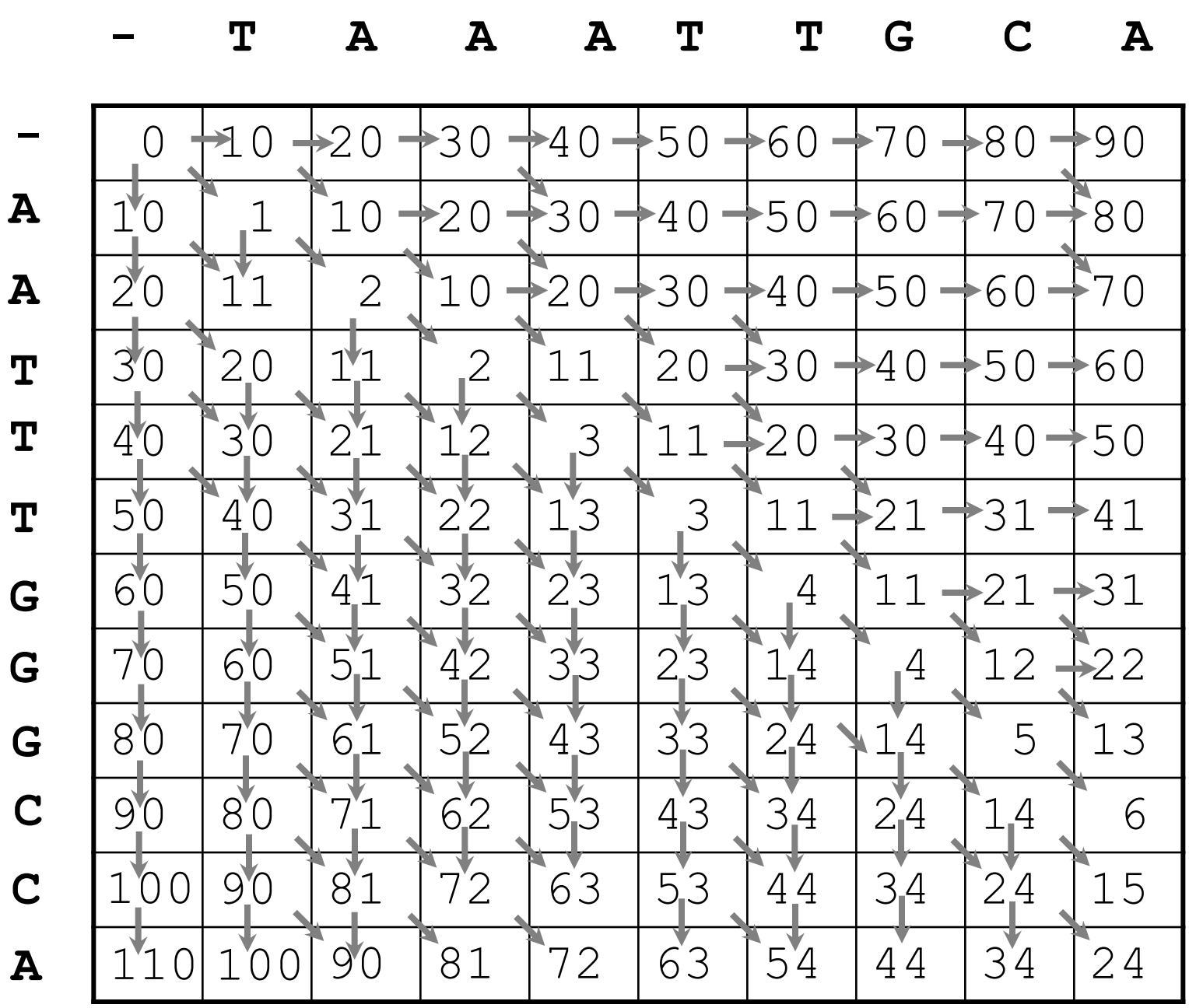
match = 0 substitution = 1 gap penalty = 10



match = 0 substitution = 1 gap penalty = 10


–
T
A
–
A
A
0
10
20
10
1
10
20
11
10+10
1+0
11+10


–
T
A
–
A
A
0
10
20
10
1
10
20
11
1


- T A A A T T G C A
- A A T T T G G G C C A
0 10 20 30 40 50 60 70 80 90
10 1 10 20 30 40 50 60 70 80
20 11 2 10 20 30 40 50 60 70
30 20 11 2 11 20 30 40 50 60
40 30 21 12 3 11 20 30 40 50
50 40 31 22 13 3 11 21 31 41
60 50 41 32 23 13 4 11 21 31
70 60 51 42 33 23 14 4 12 22
80 70 61 52 43 33 24 14 5 13
90 80 71 62 53 43 34 24 14 6
100 90 81 72 63 53 44 34 24 15
110 100 90 81 72 63 54 44 34 24

Needleman-Wunsch algorithm

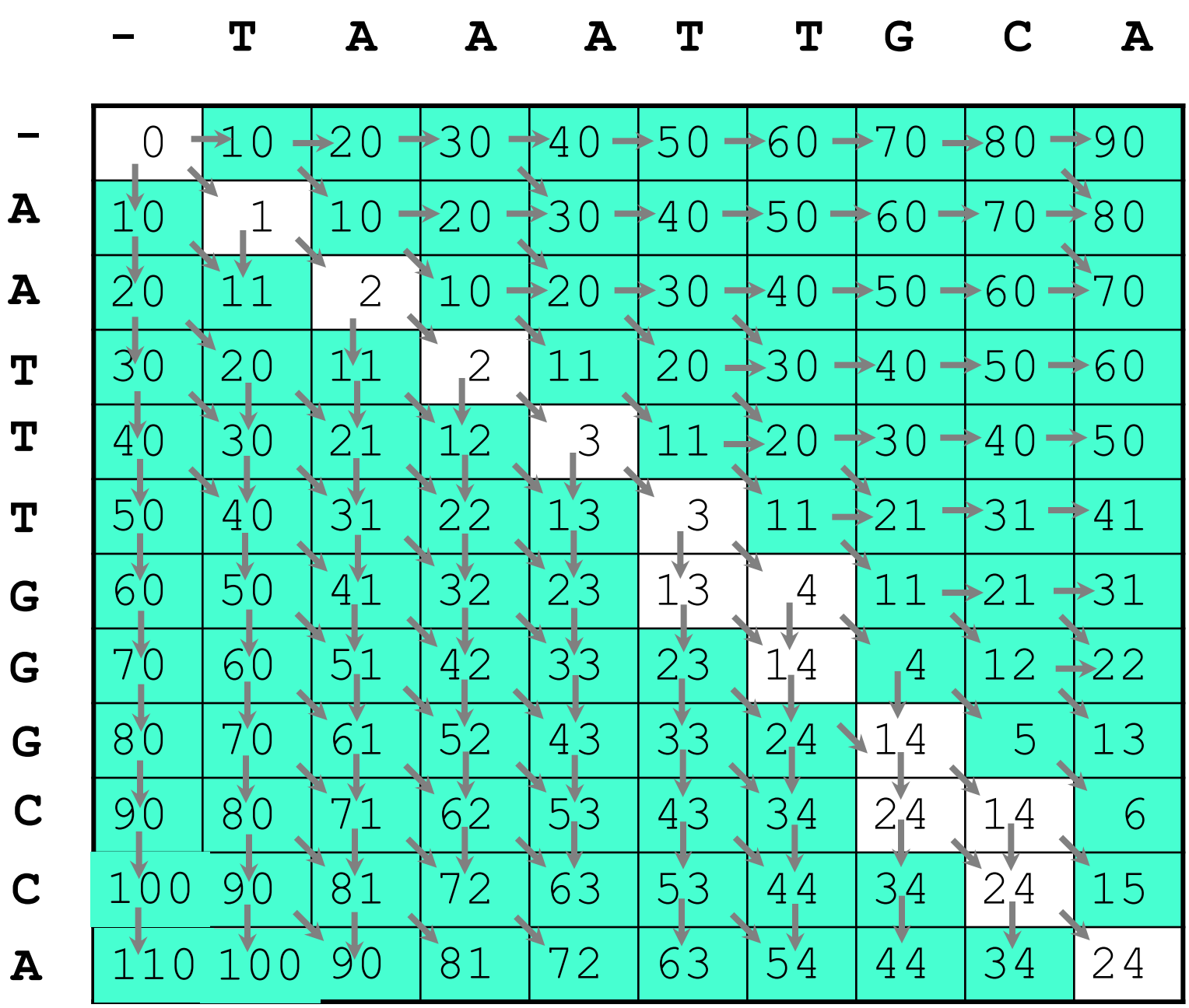
4 basic steps:

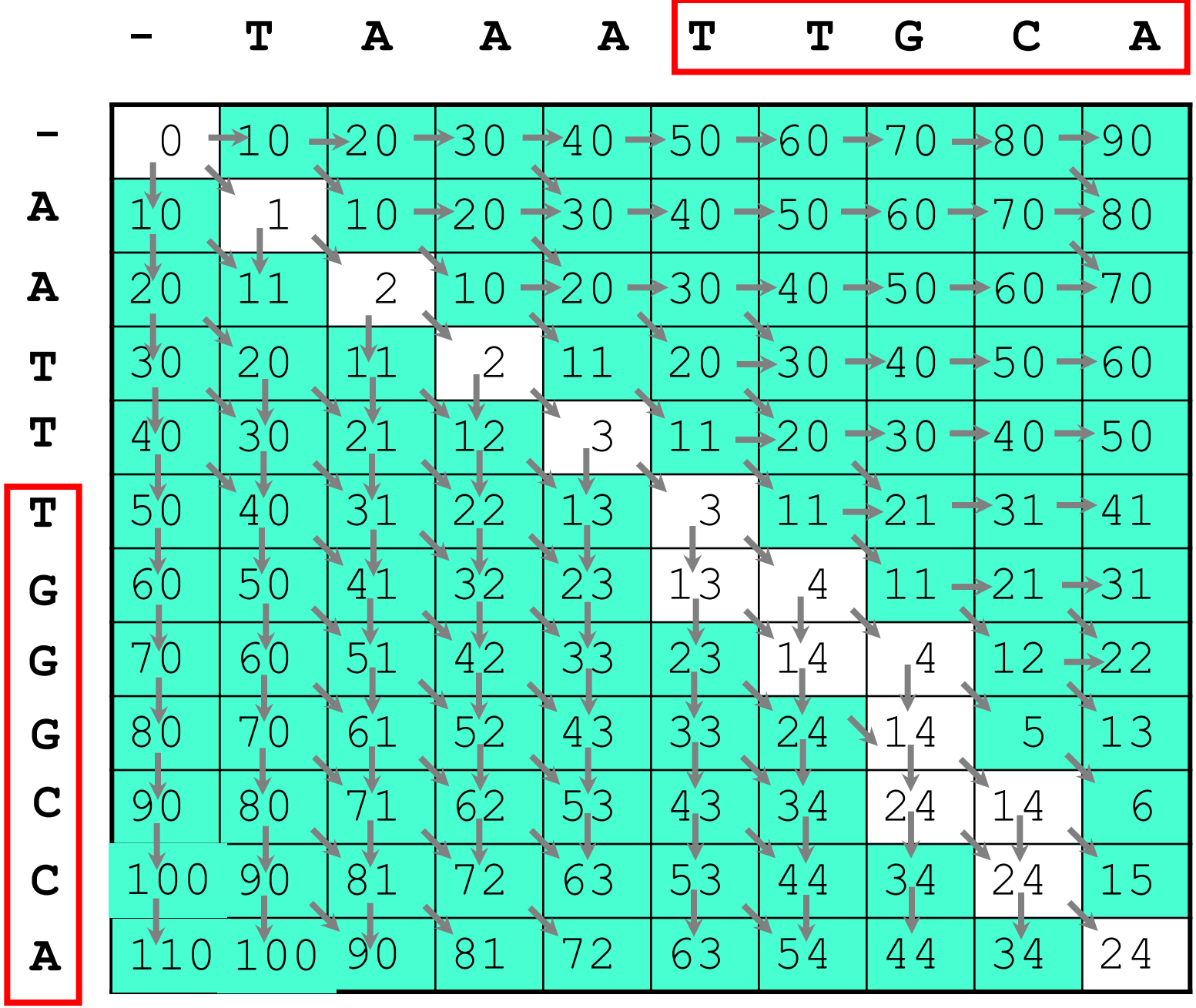
1) laying out the alignment matrix

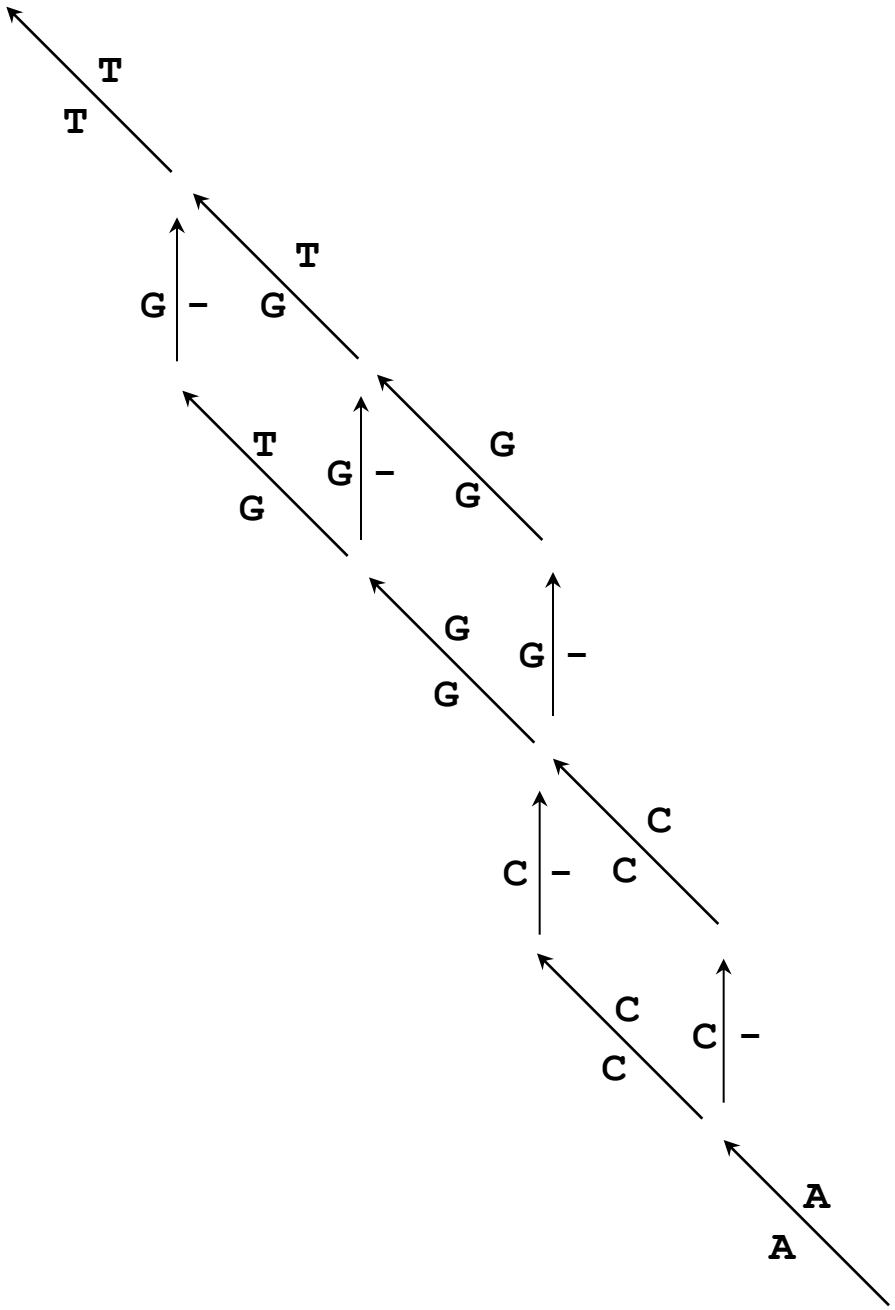
2) initiation of the matrix

3) wave-front update of the matrix elements

4) trace back


- T A A A T T G C A
- A A T T T G G G C C A
0 10 20 30 40 50 60 70 80 90
10 1 10 20 30 40 50 60 70 80
20 11 2 10 20 30 40 50 60 70
30 20 11 2 11 20 30 40 50 60
40 30 21 12 3 11 20 30 40 50
50 40 31 22 13 3 11 21 31 41
60 50 41 32 23 13 4 11 21 31
70 60 51 42 33 23 14 4 12 22
80 70 61 52 43 33 24 14 5 13
90 80 71 62 53 43 34 24 14 6
100 90 81 72 63 53 44 34 24 15
110 100 90 81 72 63 54 44 34 24


- T A A A T T G C A
- A A T T T G G G C C A
0 10 20 30 40 50 60 70 80 90
10 1 10 20 30 40 50 60 70 80
20 11 2 10 20 30 40 50 60 70
30 20 11 2 11 20 30 40 50 60
40 30 21 12 3 11 20 30 40 50
50 40 31 22 13 3 11 21 31 41
60 50 41 32 23 13 4 11 21 31
70 60 51 42 33 23 14 4 12 22
80 70 61 52 43 33 24 14 5 13
90 80 71 62 53 43 34 24 14 6
100 90 81 72 63 53 44 34 24 15
110 100 90 81 72 63 54 44 34 24



TTG-C-A
TGGGCCA

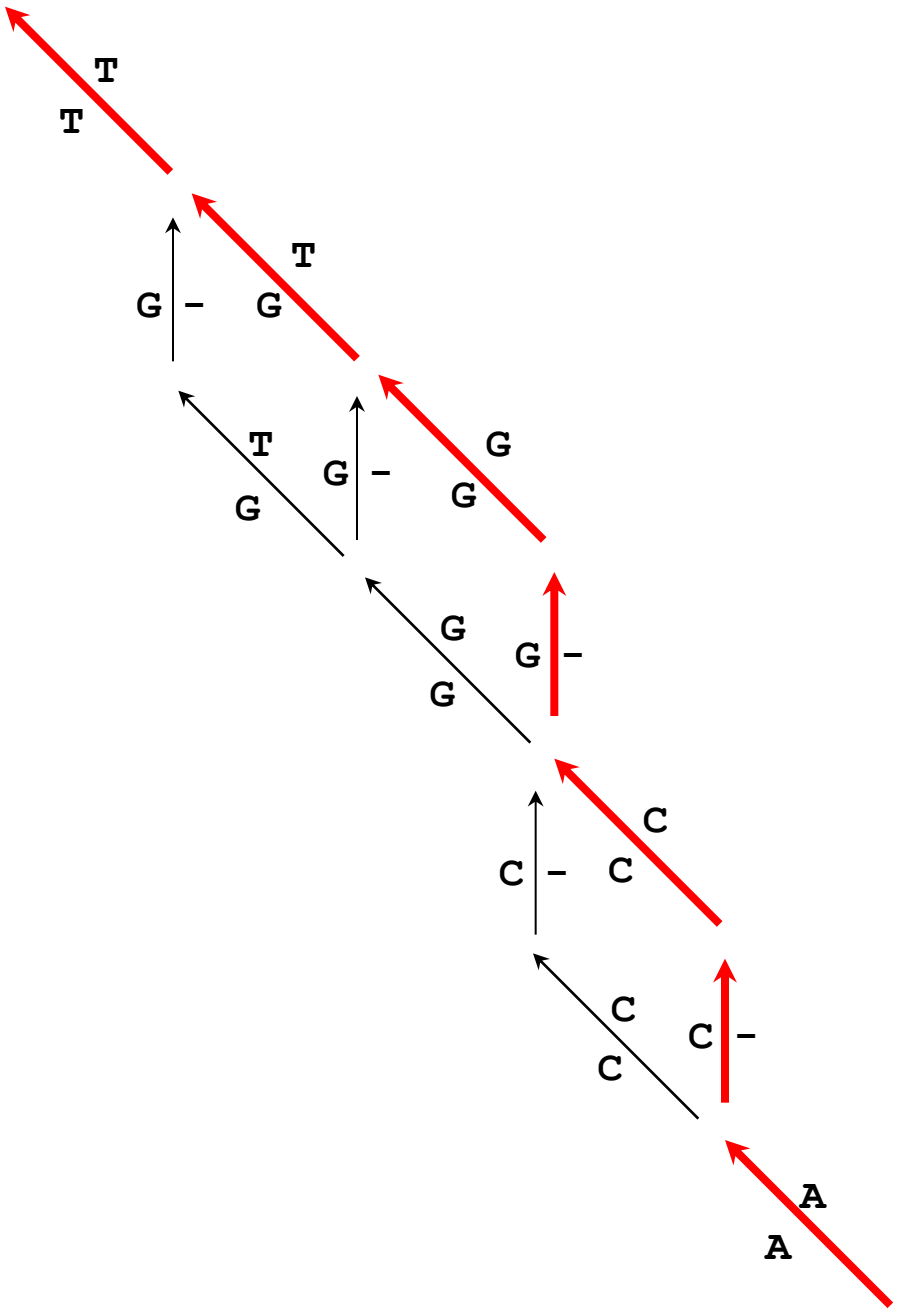
TTG--CA
TGGGCCA

TT-G-CA
TGGGCCA

TT-GC-A
TGGGCCA

T-TGC-A
TGGGCCA

T-TG-CA
TGGGCCA



TTG-C-A
TGGGCCA

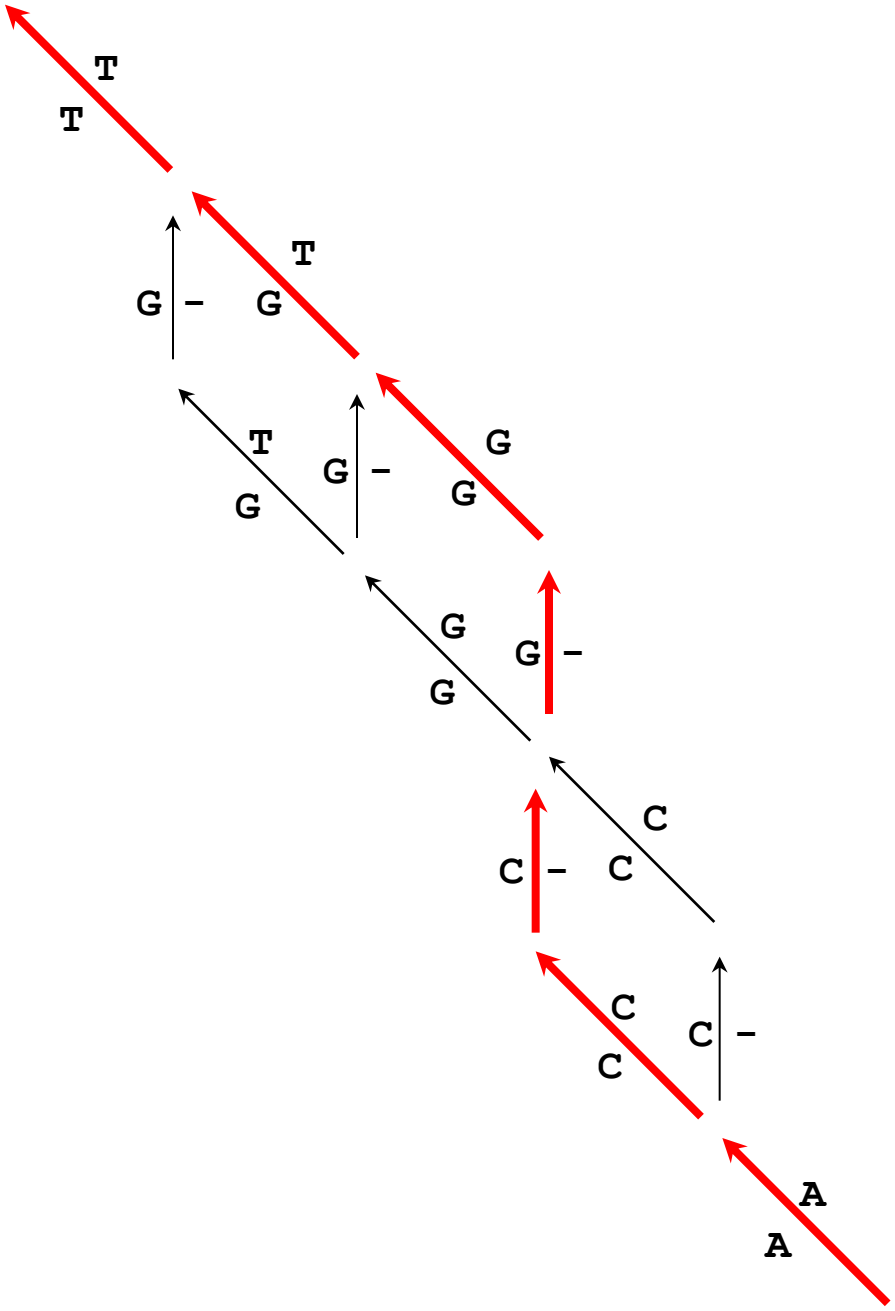
TTG--CA
TGGGCCA

TT-G-CA
TGGGCCA

TT-GC-A
TGGGCCA

T-TGC-A
TGGGCCA

T-TG-CA
TGGGCCA



TTG-C-A
TGGGCCA

TTG--CA
TGGGCCA

TT-G-CA
TGGGCCA

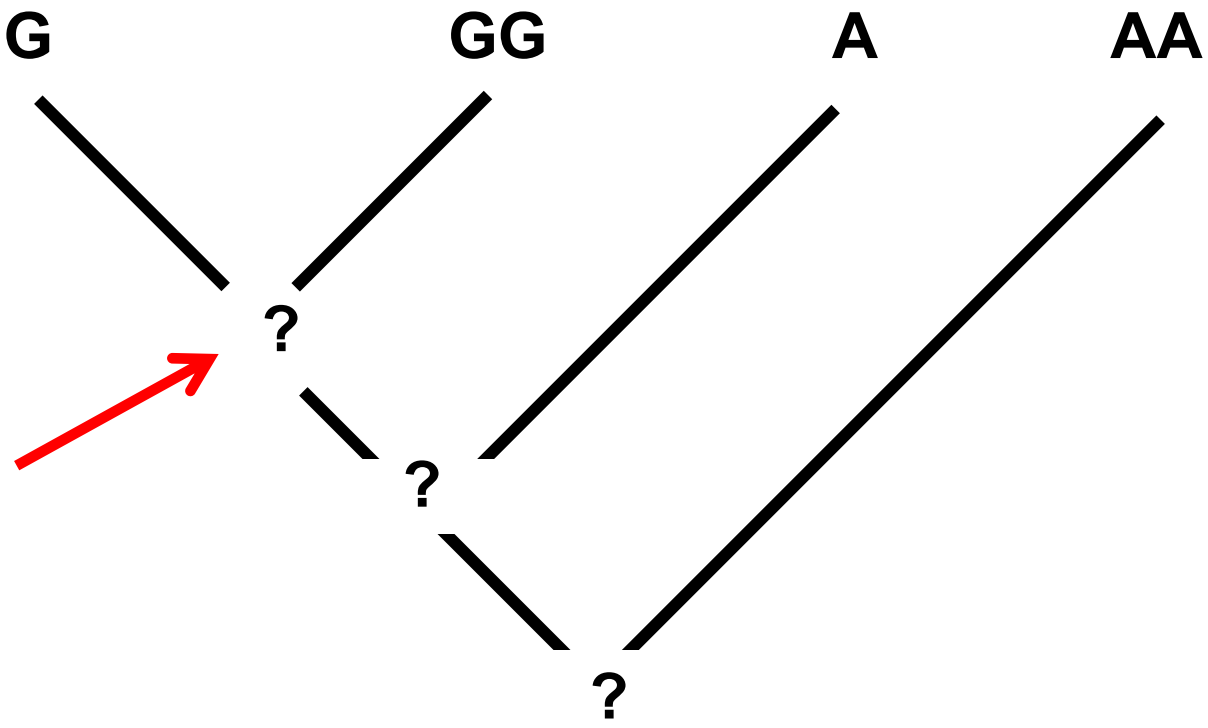
TT-GC-A
TGGGCCA

T-TGC-A
TGGGCCA

T-TG-CA
TGGGCCA

Direct optimization (DO)

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT



	A	C	G	T	-	M	R	W	(A)	S	Y	(C)	K	(G)	(T)	V	H	(M)	D	(R)	(W)	B	(S)	(Y)	(K)	N	(V)	(H)	(B)	(D)	X
A	0	2	1	2	3	0	0	0	0	1	2	2	1	1	2	0	0	0	0	0	0	1	1	2	1	0	0	0	1	0	0
C		0	2	1	3	0	2	1	2	0	0	0	1	2	1	0	0	0	1	2	1	0	0	0	1	0	0	0	0	1	0
G			0	2	3	1	0	1	1	0	2	2	0	0	2	0	1	1	0	0	1	0	0	2	0	0	0	1	0	0	0
T				0	3	1	2	0	2	1	0	1	0	2	0	1	0	1	0	2	0	0	1	0	0	0	1	0	0	0	0
-					0	3	3	3	0	3	3	0	3	0	0	3	3	0	3	0	0	3	0	0	0	3	0	0	0	0	3
M						0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
R							0	0	0	0	2	2	0	0	2	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0
W								0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
(A)									0	1	2	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
S										0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Y											0	0	0	2	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
(C)												0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
K													0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
(G)														0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(T)															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(M)																		0	0	0	0	0	0	0	0	0	0	0	0	0	0
D																			0	0	0	0	0	0	0	0	0	0	0	0	0
(R)																				0	0	0	0	0	0	0	0	0	0	0	0
(W)																					0	0	0	0	0	0	0	0	0	0	0
B																						0	0	0	0	0	0	0	0	0	0
(S)																							0	0	0	0	0	0	0	0	0
(Y)																								0	0	0	0	0	0	0	0
(K)																									0	0	0	0	0	0	0
N																										0	0	0	0	0	0
(V)																											0	0	0	0	0
(H)																												0	0	0	0
(B)																													0	0	0
(D)																														0	0
X																															0

GA	R
GT	K
GC	S
GAT	D
GAC	V
GTC	B
ATC	H
AT	W
AC	M
CT	Y
ACGT	N
A-	(A)
AC-	(M)
.....	
N-	X

transitions	1
transversions	2
indels	3

31 x 31 = 961 - 31 = 930/2 = 465

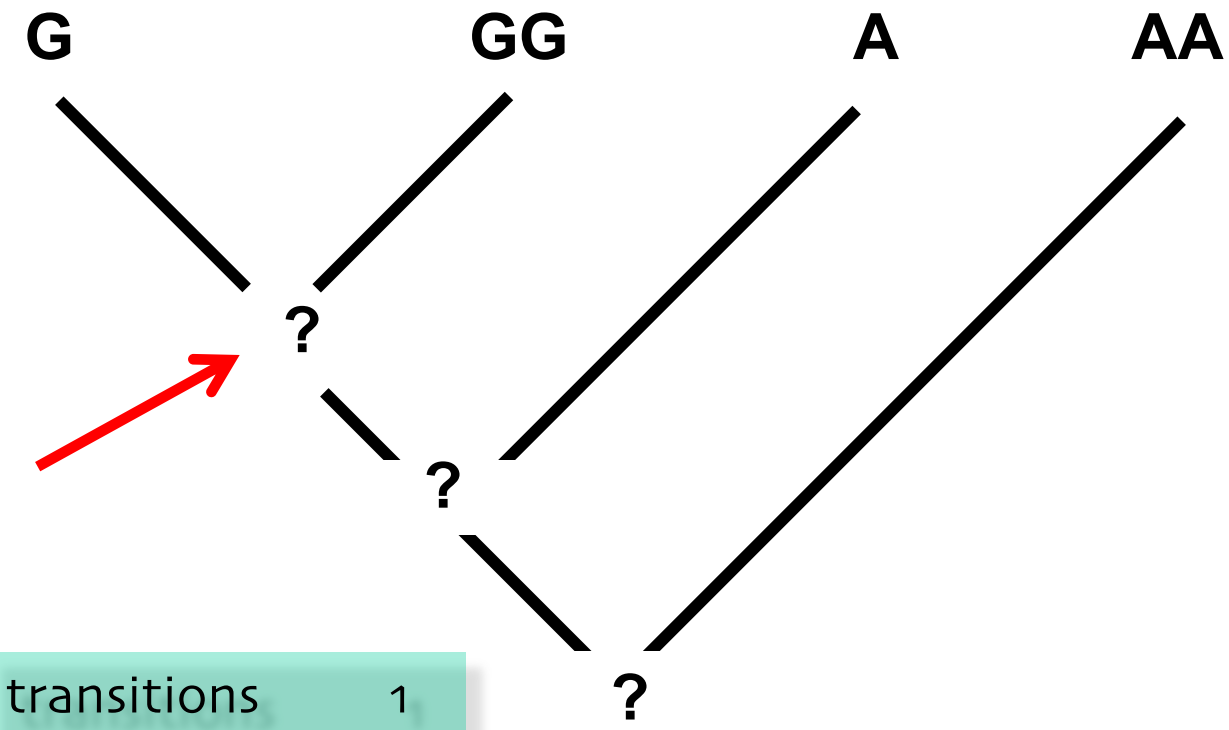
	A	C	G	T	-	M	R	W	(A)	S	Y	(C)
A	0	2	1	2	3	0	0	0	0	1	2	2
C		0	2	1	3	0	2	1	2	0	0	0
G			0	2	3	1	0	1	1	0	2	2
T				0	3	1	2	0	2	1	0	1
-					0	3	3	3	0	3	3	0
M						0	0	0	0	0	0	0
R							0	0	0	0	2	2
W								0	0	0	1	0
(A)									0	1	2	0
S										0	0	0
Y											0	0
(C)												0

GA	R
GT	K
GC	S
GAT	D
GAC	V
GTC	B
ATC	H
AT	W
AC	M
CT	Y
ACGT	N
A-	(A)
AC-	(M)
.....	
N-	X

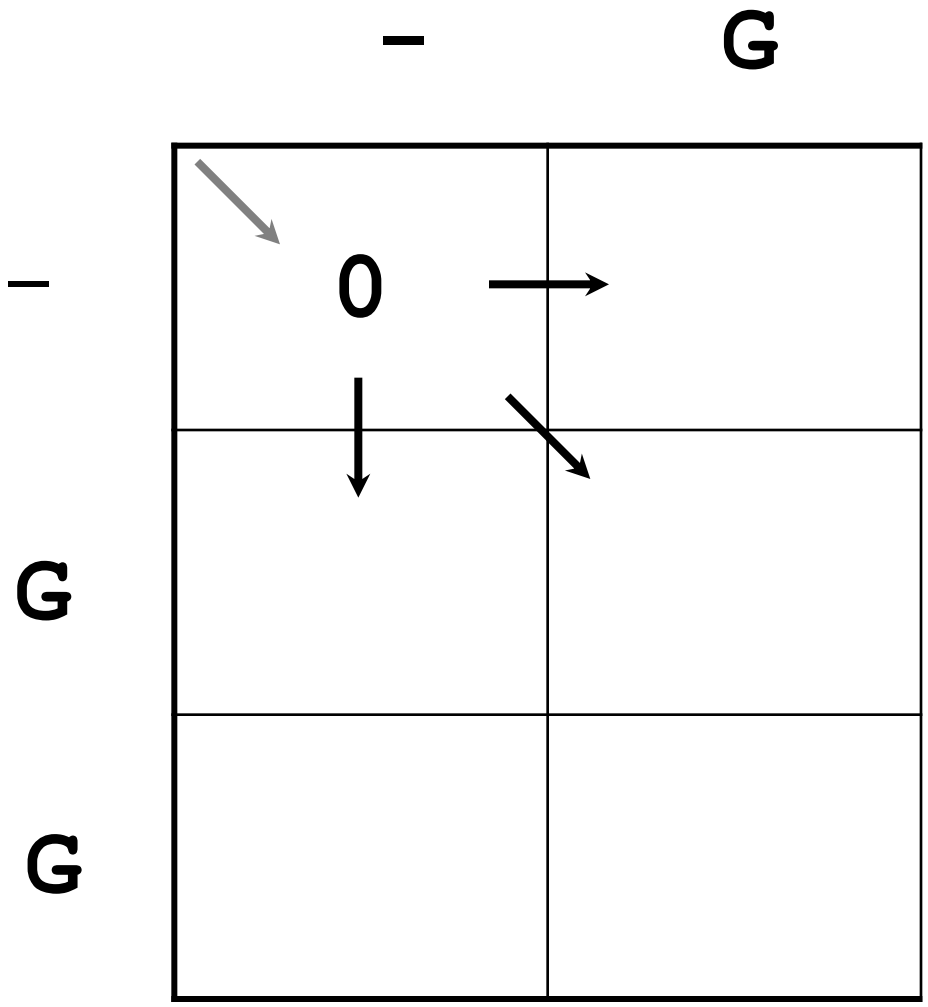
transitions	1
transversions	2
indels	3

Direct optimization (DO)

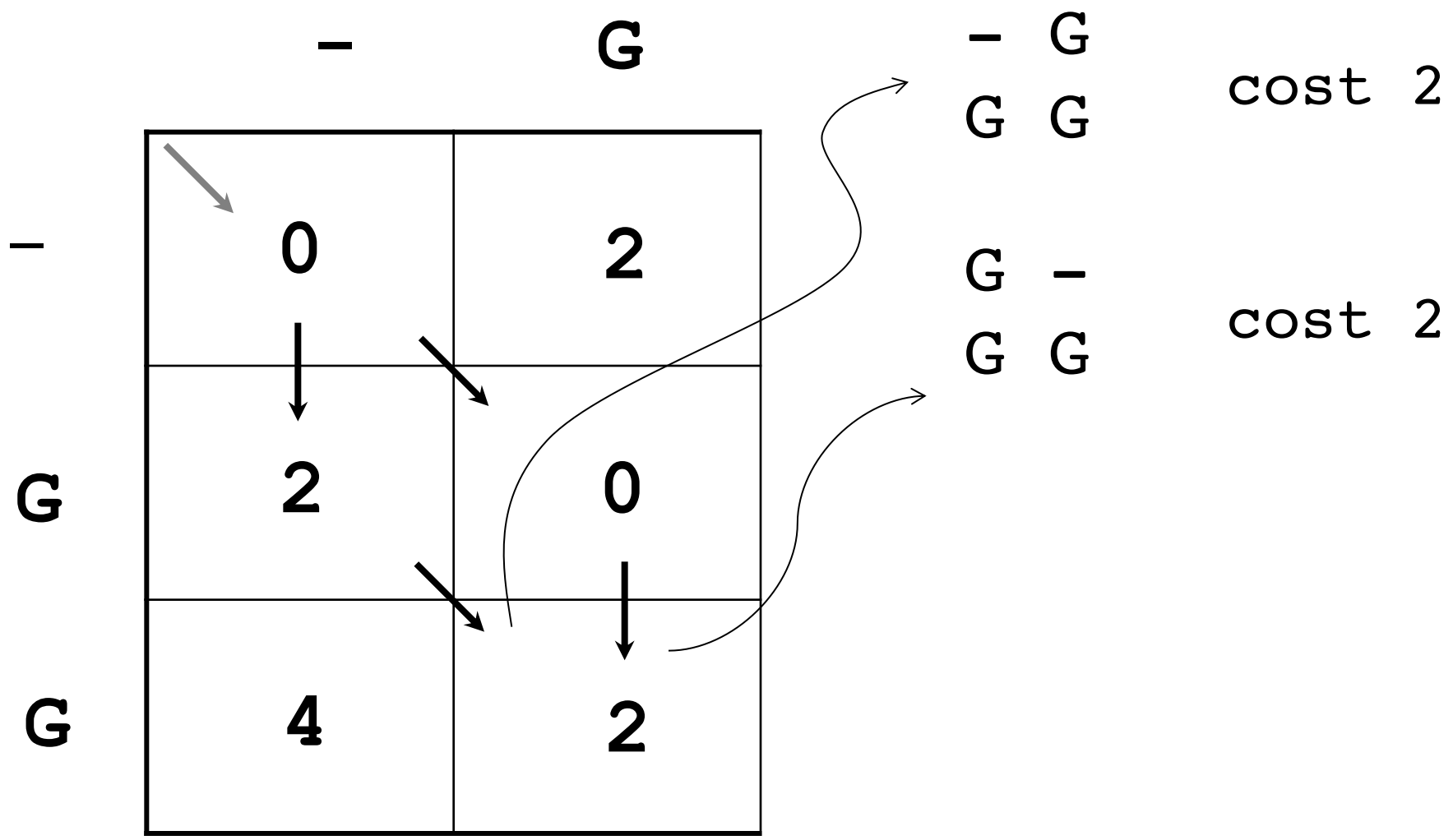
AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT



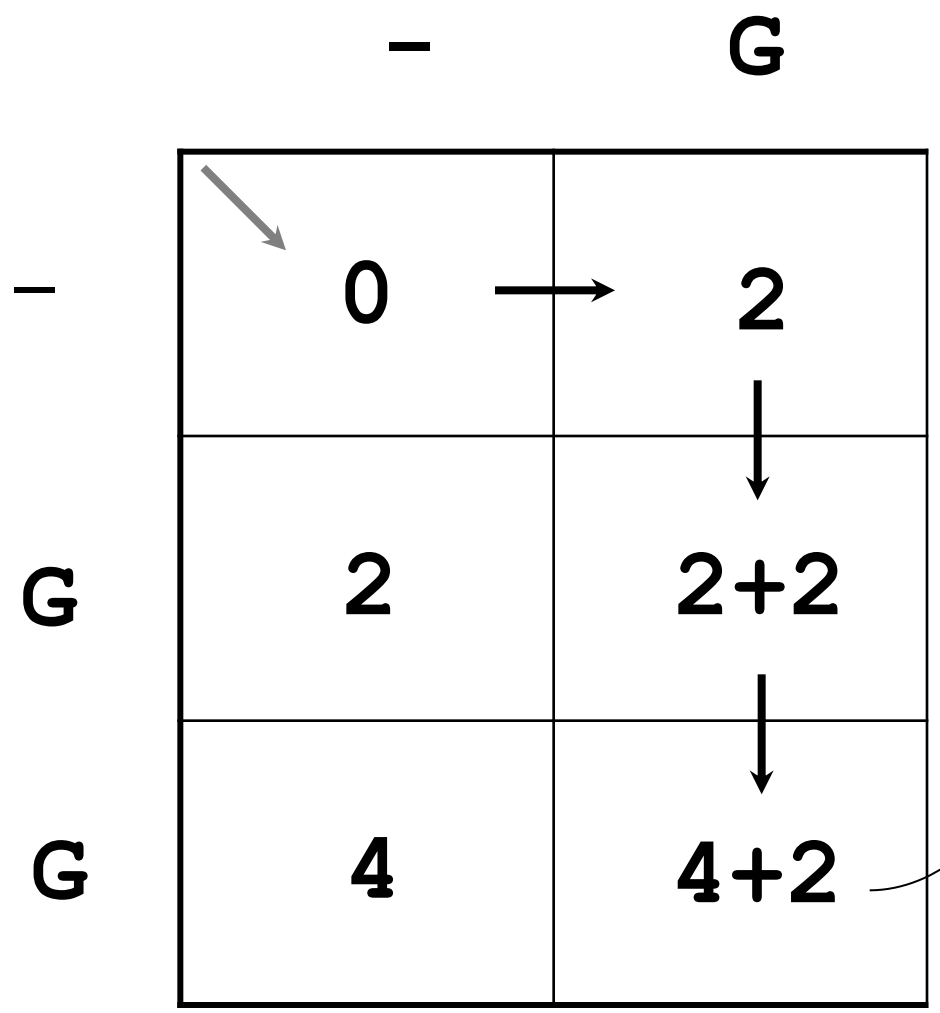
transitions	1
transversions	1
indels	2



transitions	1
transversions	1
indels	2



transitions	1
transversions	1
indels	2

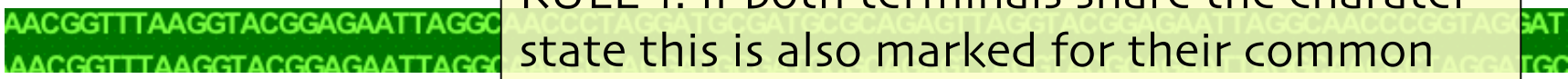


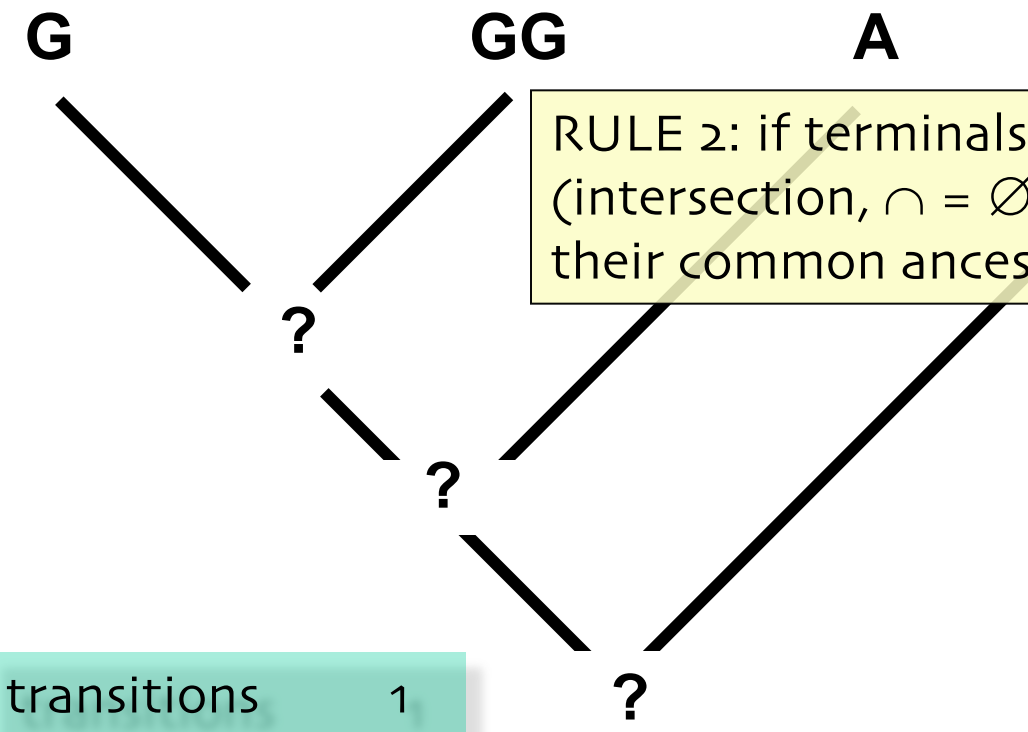
transitions	1
transversions	1
indels	2

```
– G
G G      cost 2

G –
G G      cost 2

G – –
– G G    cost 6

– G –
G – G    cost 6

– – G
G G –    cost 6
```

Direct optimization (DO)

RULE 1: if both terminals share the charater state this is also marked for their common ancestor (intersection, ∩)





RULE 2: if terminals have different character states (intersection, ∩ = ∅) mark their union (∪) for their common ancestor

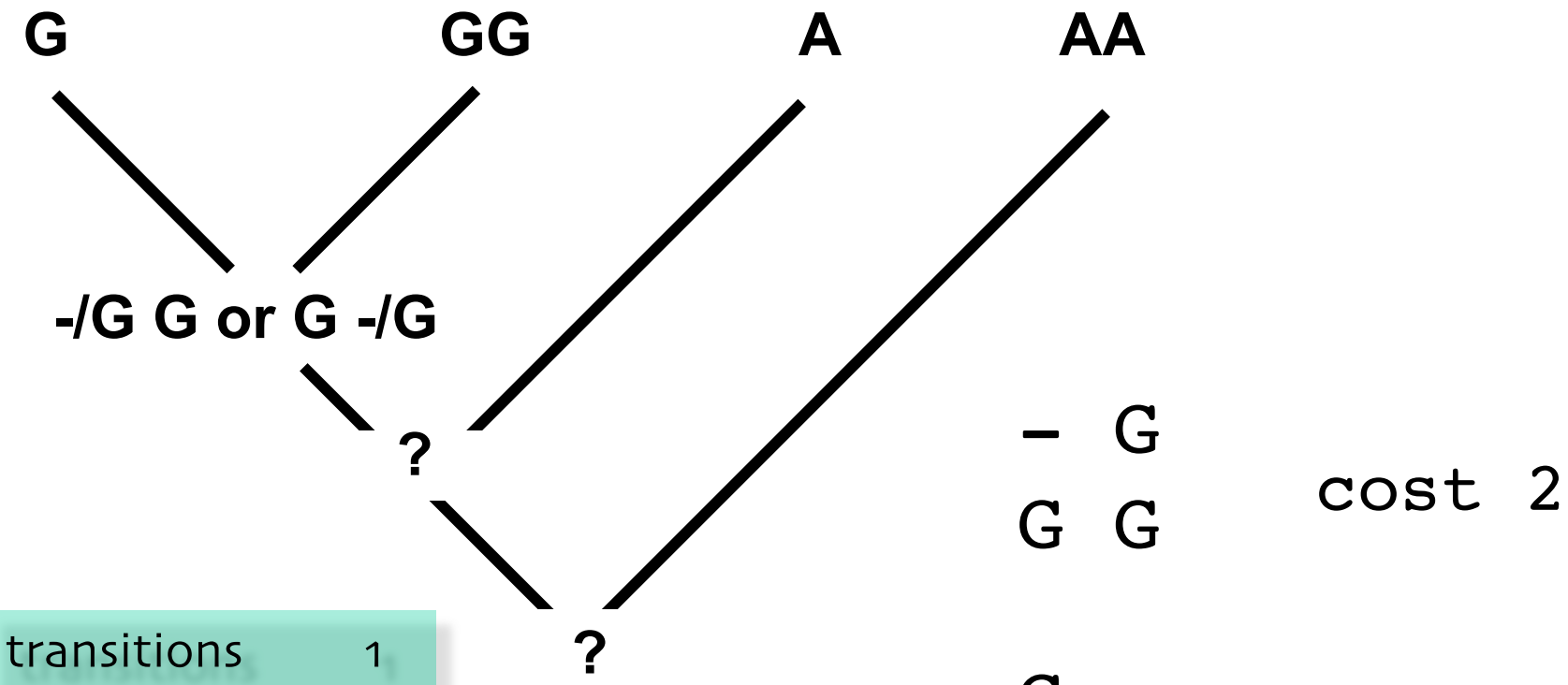
transitions	1
transversions	1
indels	2

```
- G
G G
```
cost 2

```
G -
G G
```
cost 2

Direct optimization (DO)

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT



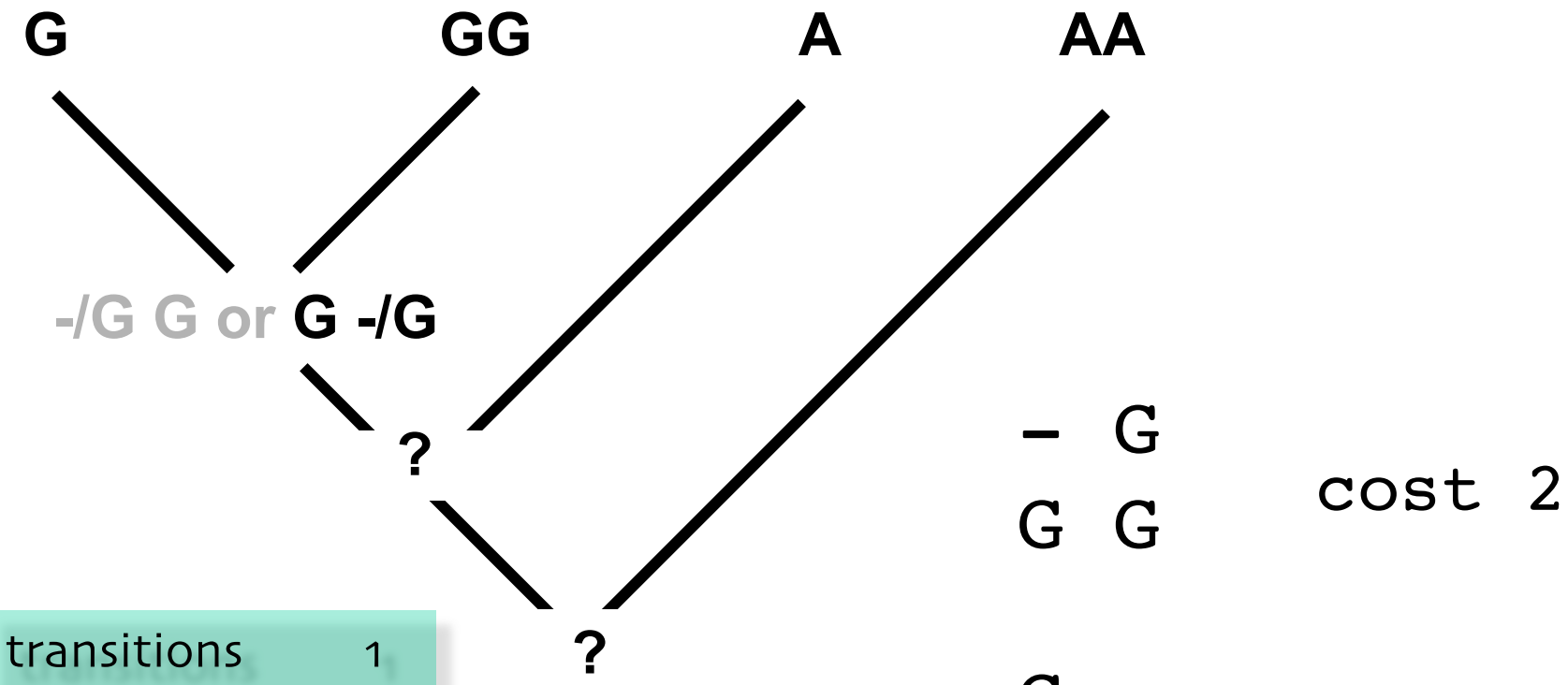
transitions	1
transversions	1
indels	2

```
- G
G G
```
cost 2

```
G -
G G
```
cost 2

Direct optimization (DO)

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT



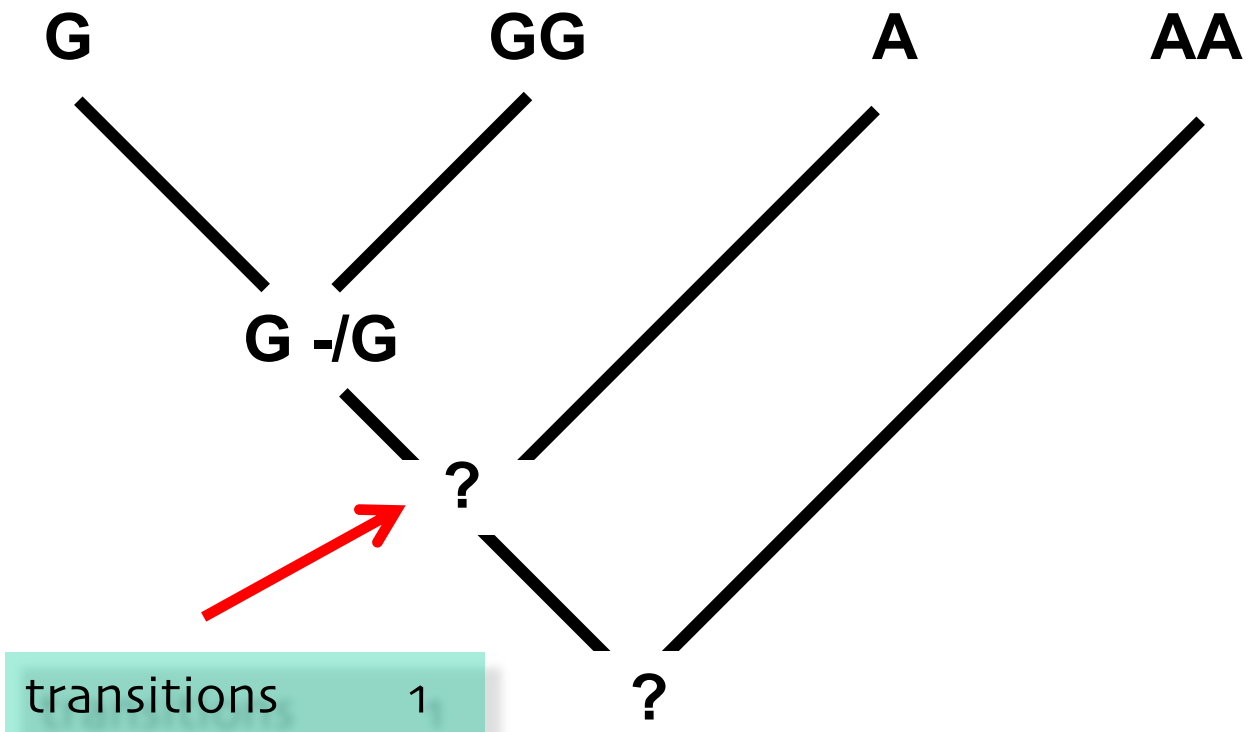
transitions	1
transversions	1
indels	2

```
- G
G G
```
cost 2

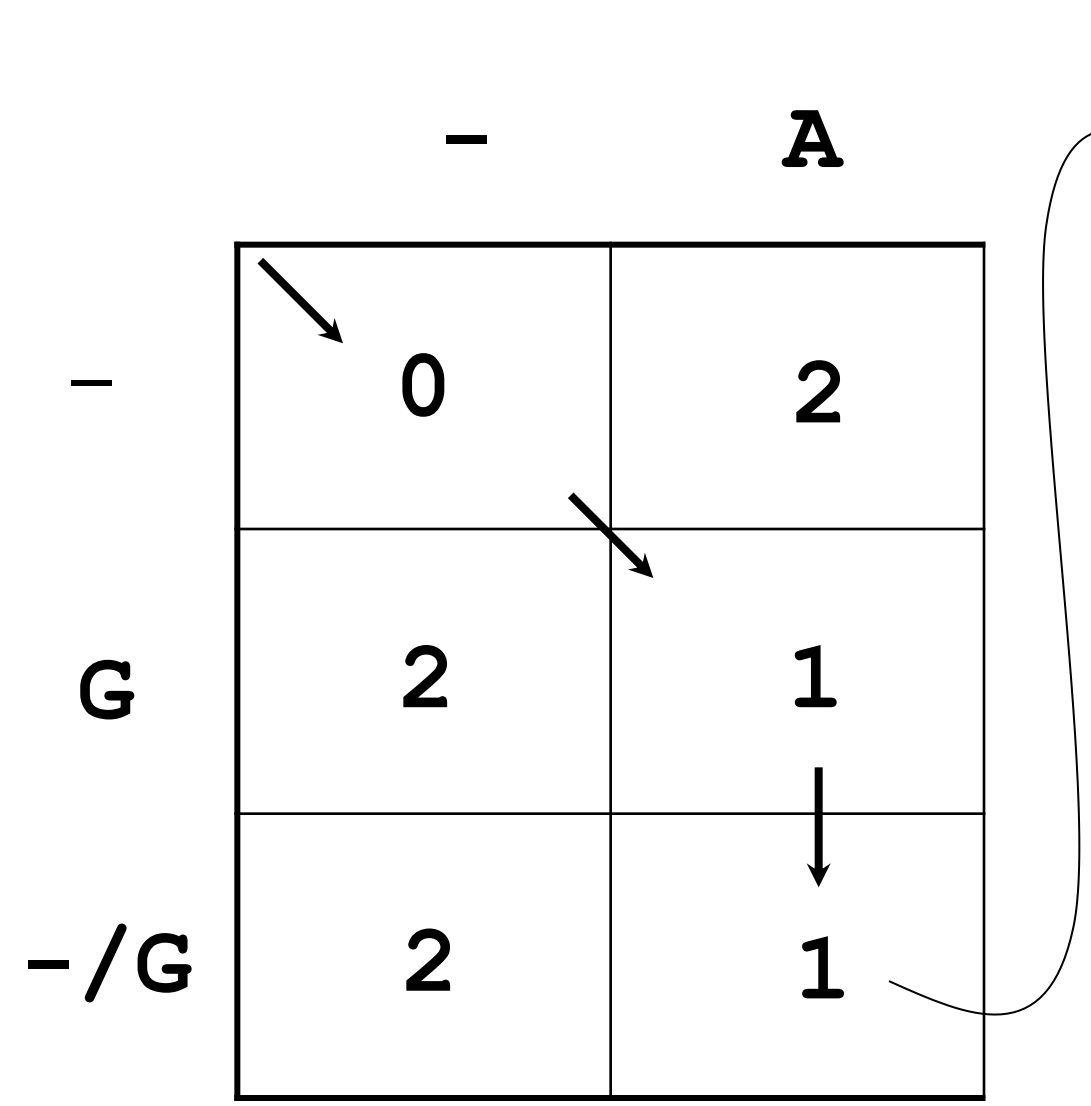
```
G -
G G
```
cost 2

Direct optimization (DO)

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT



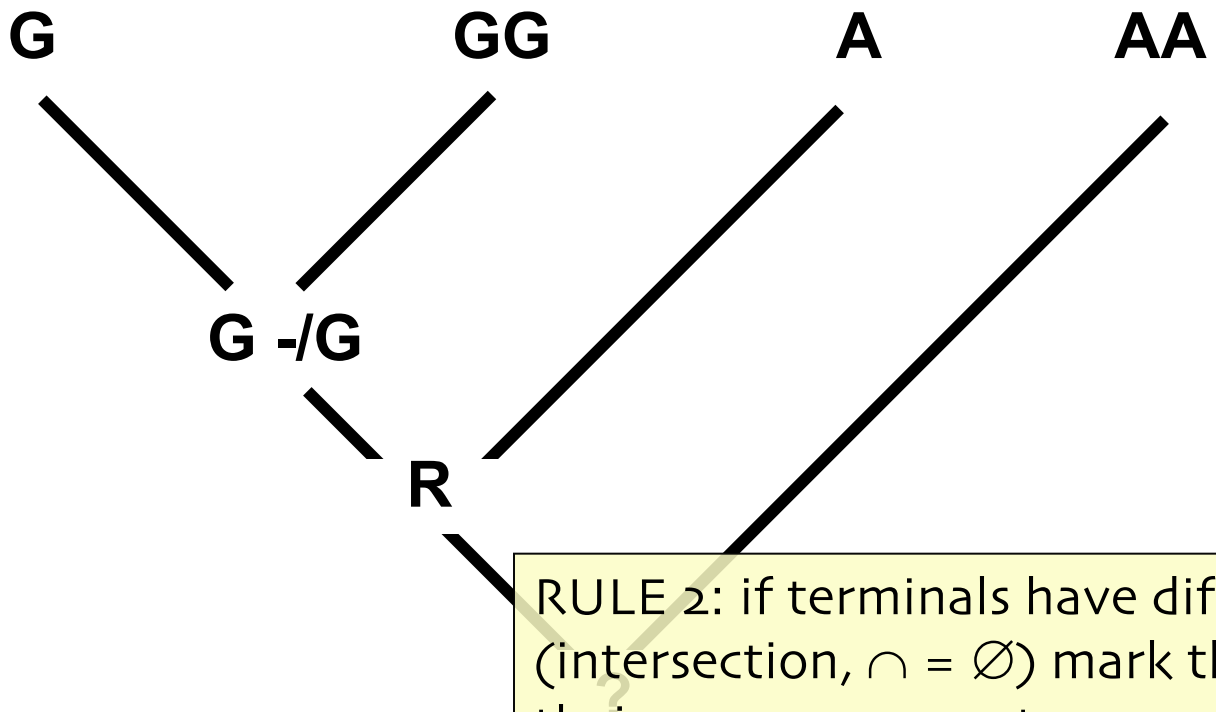
transitions	1
transversions	1
indels	2




```
A  -
G -/G   cost 1

-  A
G -/G   cost 3

A -  -
- G -/G cost 4

- A -
G - -/G cost 4

-  -  A
G -/G -  cost 4
```

Direct optimization (DO)

RULE 1: if both terminals share the charater state this is also marked for their common ancestor (intersection, ∩)



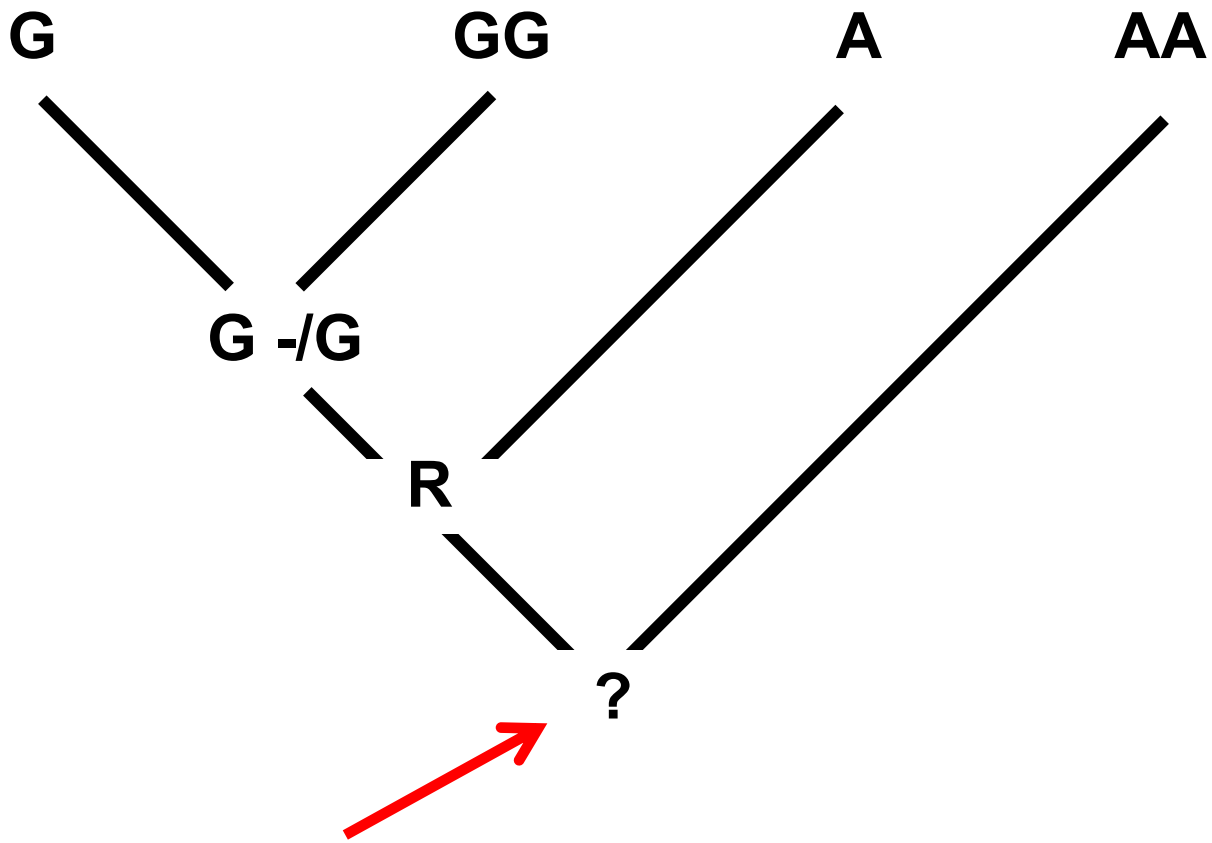
RULE 2: if terminals have different character states (intersection, ∩ = ∅) mark their union (∪) for their common ancestor

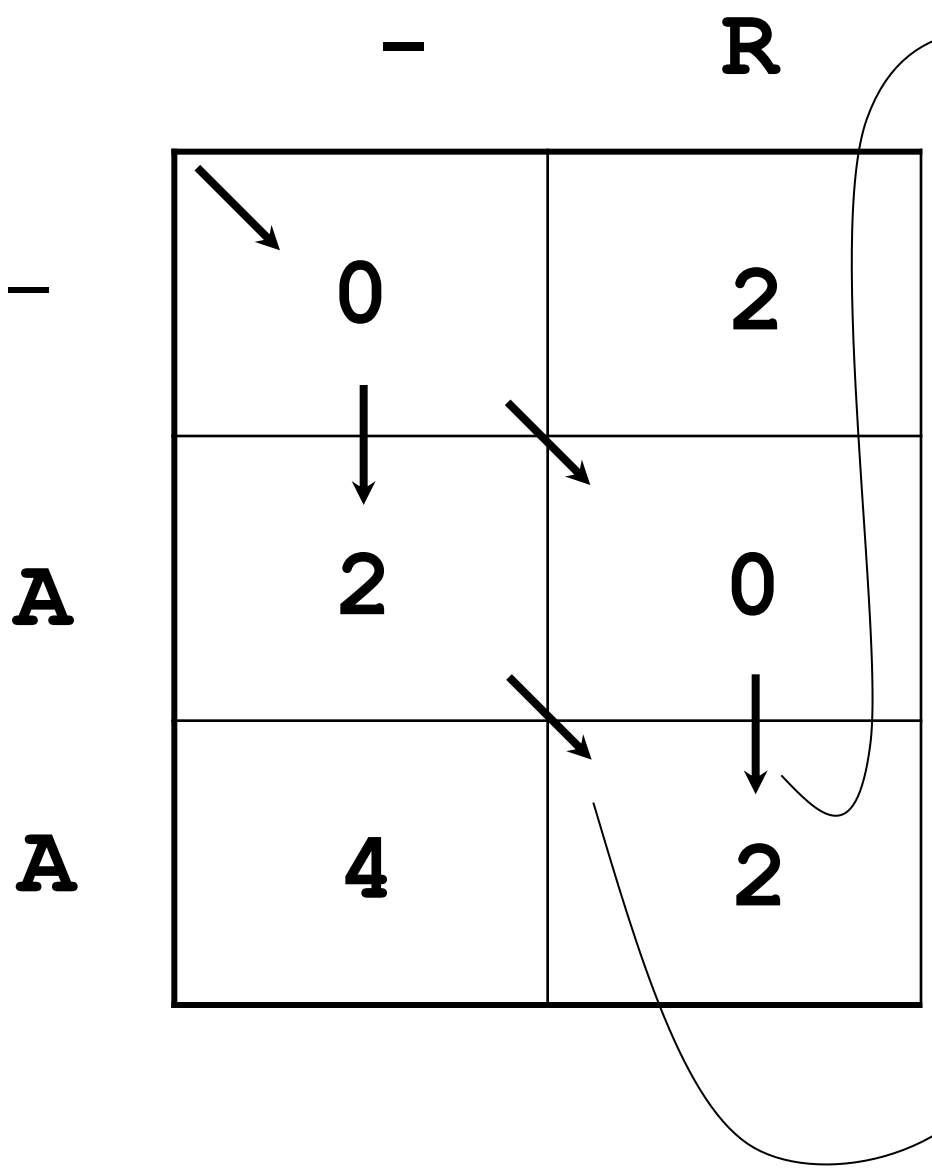
```
A   -
G  -/G   cost 1
```

GA	R
GT	K
GC	S
GAT	D
GAC	V
GTC	B
ATC	H
AT	W
AC	M
CT	Y
ACGT	N
ACGT-	X

Direct optimization (DO)

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT





R -
A A

cost 2

- R
A A

cost 2

R - -
- A A

cost 6

- R -
A - A

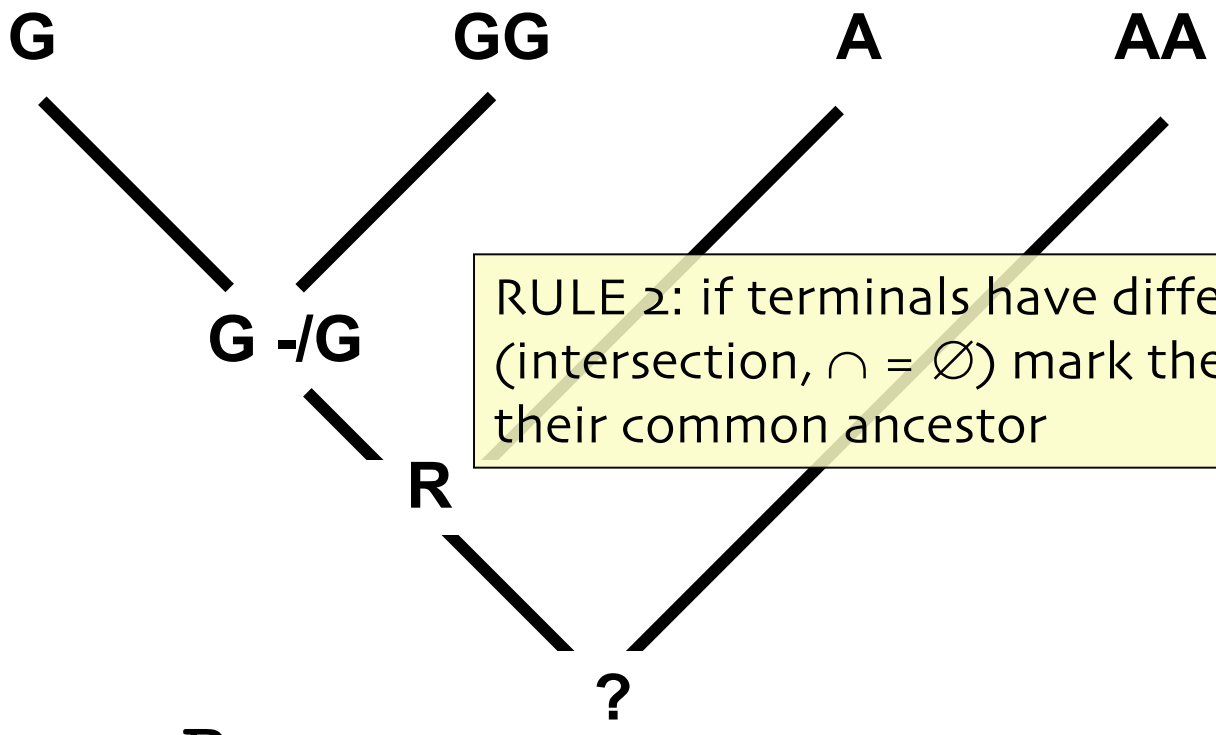
cost 6

- - R
A A -

cost 6

Direct optimization (DO)

RULE 1: if both terminals share the charater state this is also marked for their common ancestor (intersection, ∩)

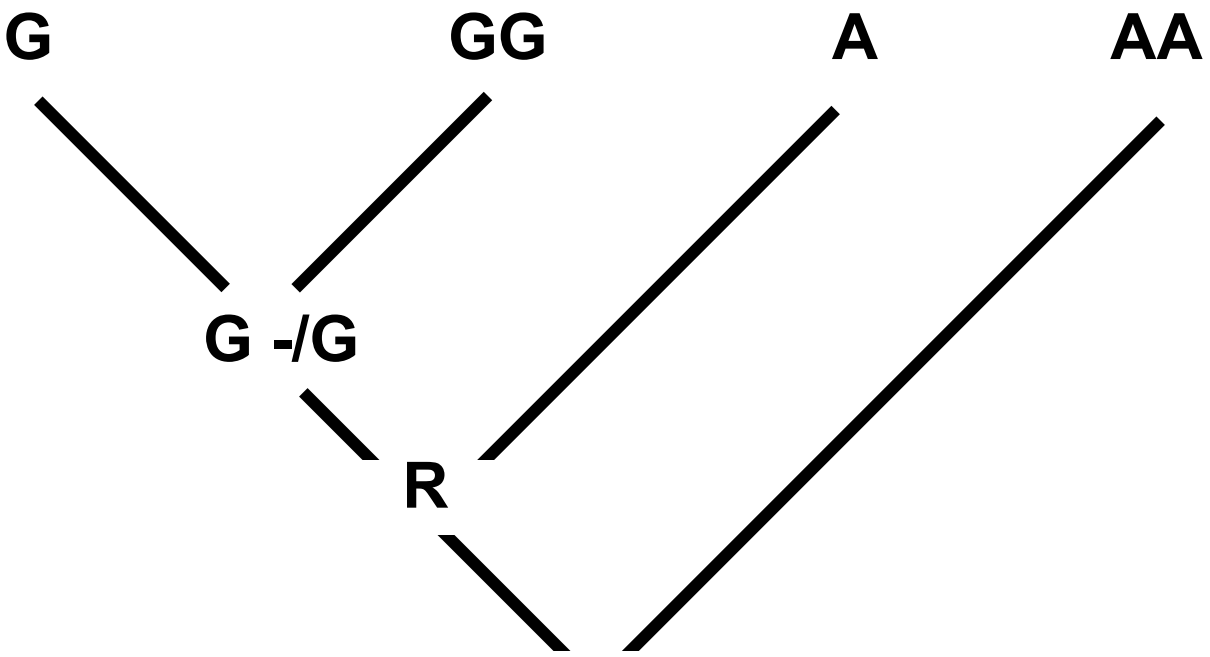


RULE 2: if terminals have different character states (intersection, ∩ = ∅) mark their union (∪) for their common ancestor

GA	R
GC	S
GAT	D
GAC	V
GTC	B
ATC	H
AT	W
AC	M
CT	Y
ACGT	N
ACGT-	X

Direct optimization (DO)

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT




```
-   R
A   A
```

cost 2

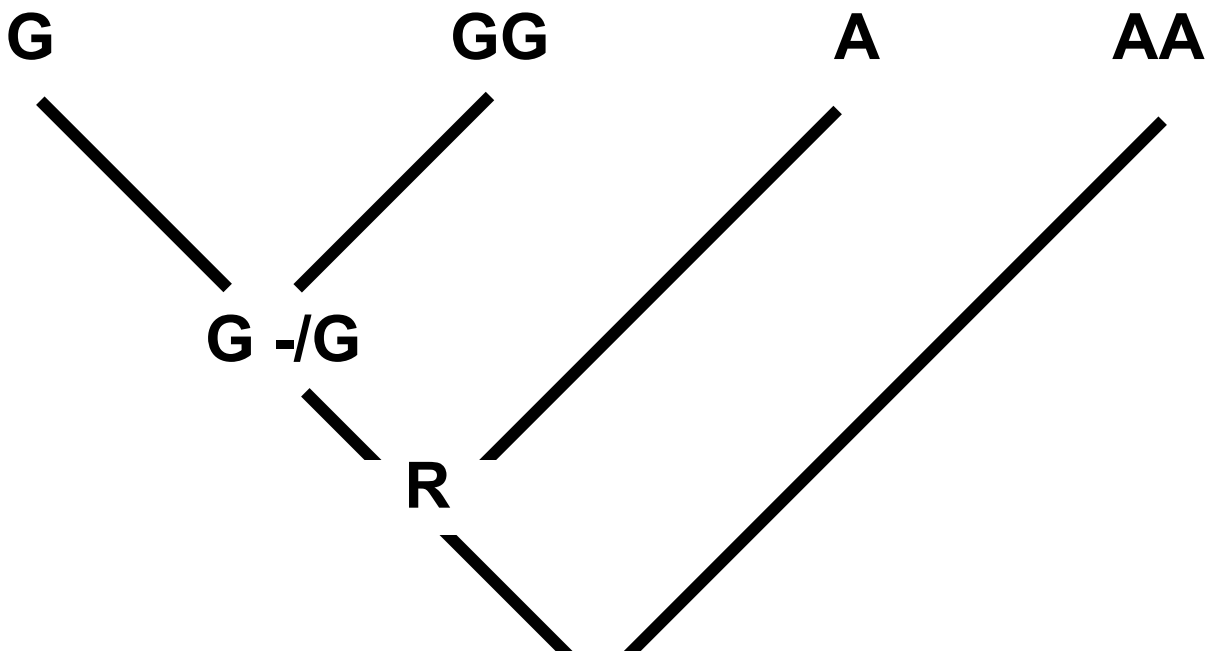
```
R   -
A   A
```

cost 2

GA	R
GT	K
GC	S
GAT	D
GAC	V
GTC	B
ATC	H
AT	W
AC	M
CT	Y
ACGT	N
ACGT-	X

Direct optimization (DO)

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT




```
-   R
A   A
```

cost 2

```
R   -
A   A
```

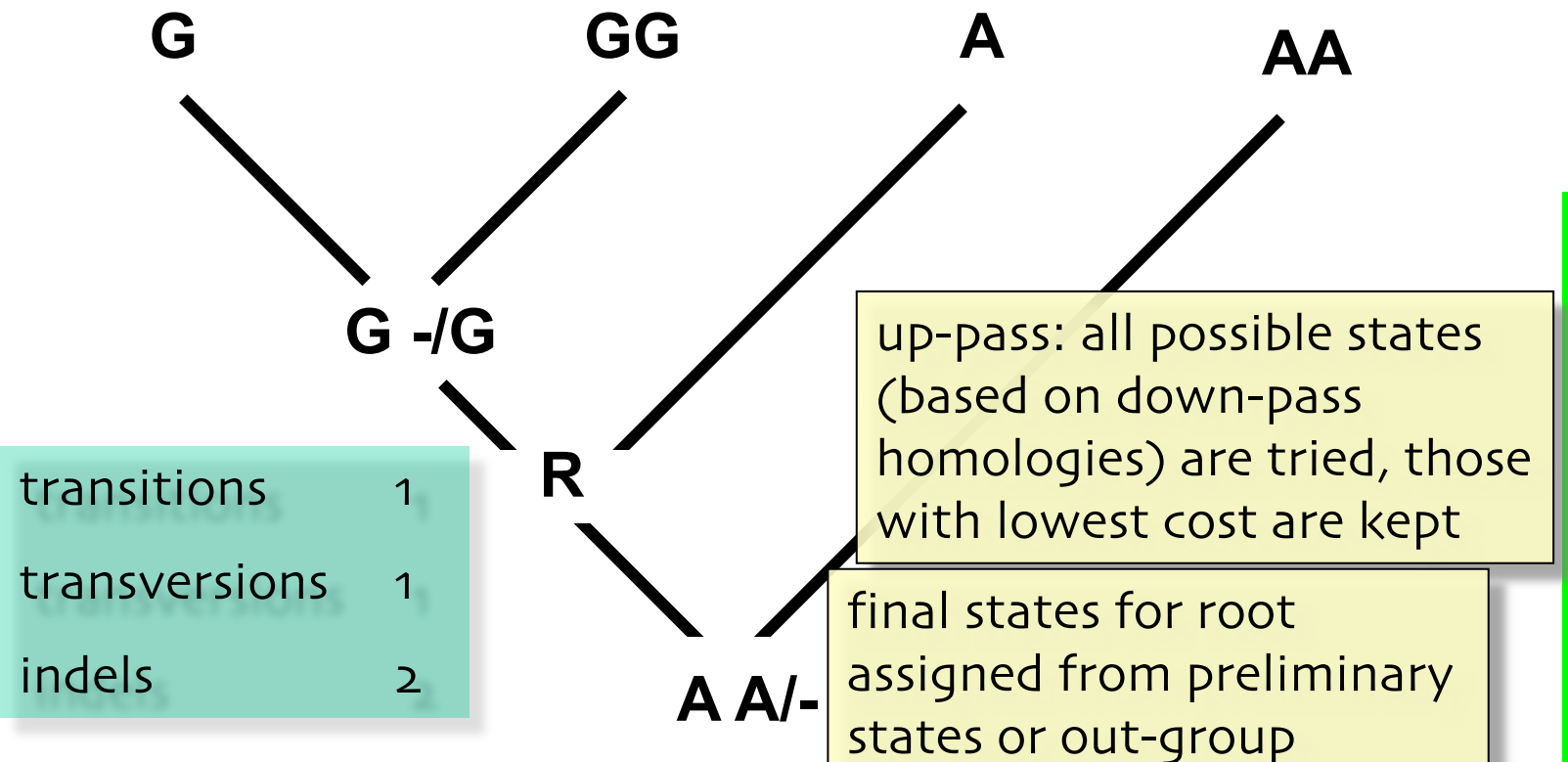
cost 2

GA	R
GT	K
GC	S
GAT	D
GAC	V
GTC	B
ATC	H
AT	W
AC	M
CT	Y
ACGT	N
ACGT-	X

Direct optimization (DO)

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCTAGGATGC



transitions	1
transversions	1
indels	2

up-pass: all possible states (based on down-pass homologies) are tried, those with lowest cost are kept

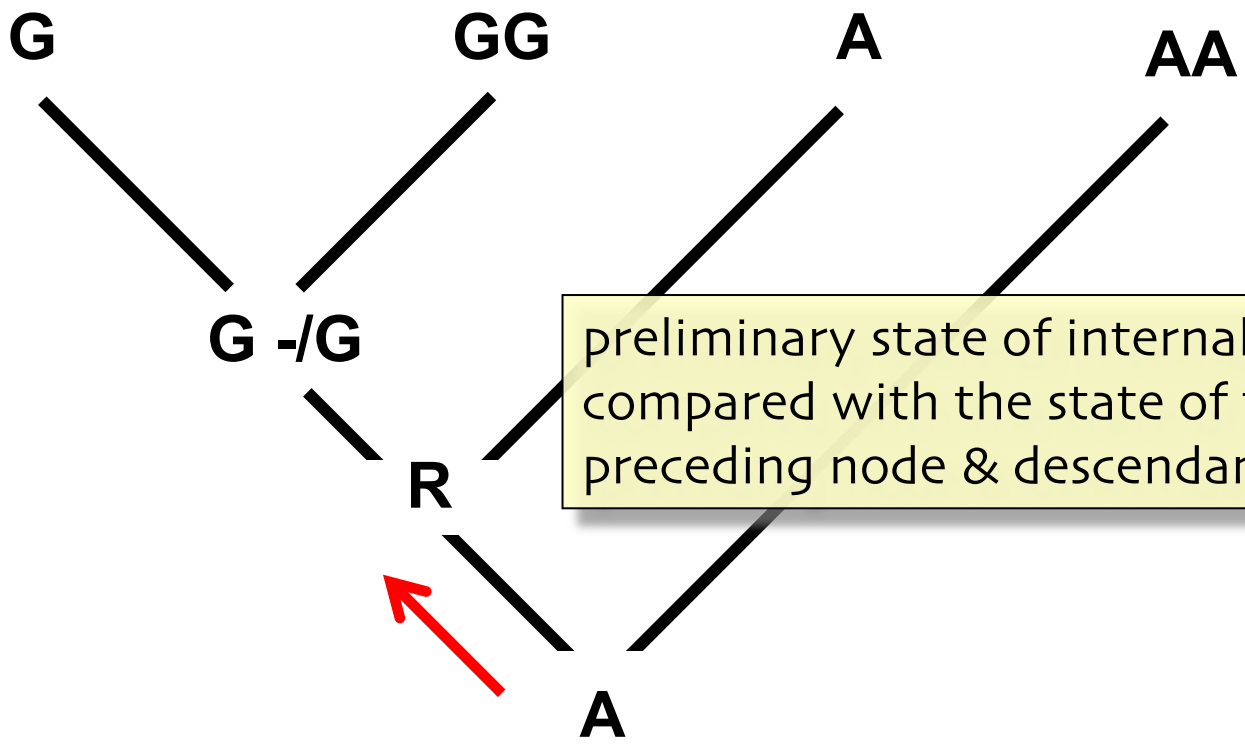
final states for root assigned from preliminary states or out-group

GA	R
GT	K
GC	S
GAT	D
GAC	V
GTC	B
ATC	H
AT	W
AC	M
CT	Y
ACGT	N
ACGT-	X

total cost 2+1+2=5

Direct optimization (DO)

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT



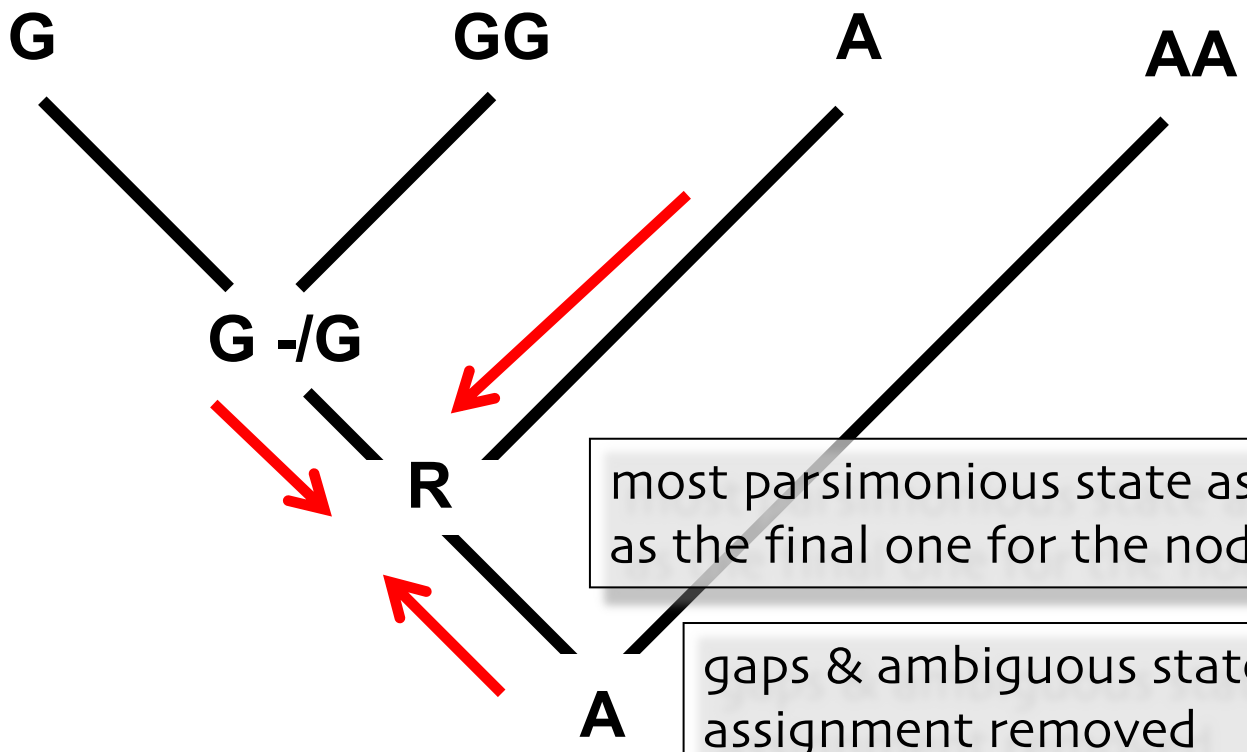
preliminary state of internal node compared with the state of the preceding node & descendant nodes

GA	R
GT	K
GC	S
GAT	D
GAC	V
GTC	B
ATC	H
AT	W
AC	M
CT	Y
ACGT	N
ACGT-	X

Direct optimization (DO)

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCTAGGATGC



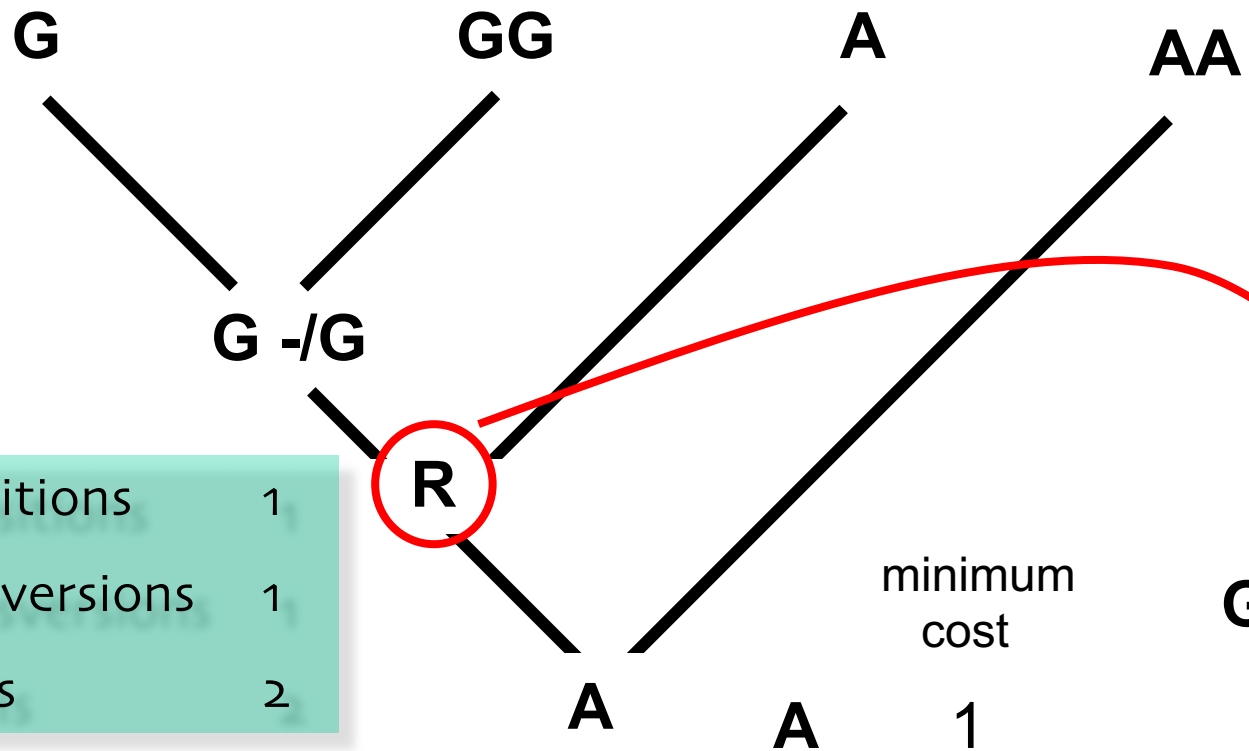
most parsimonious state assigned as the final one for the node

gaps & ambiguous state assignment removed

GA	R
GT	K
GC	S
GAT	D
GAC	V
GTC	B
ATC	H
AT	W
AC	M
CT	Y
ACGT	N
ACGT-	X

Direct optimization (DO)

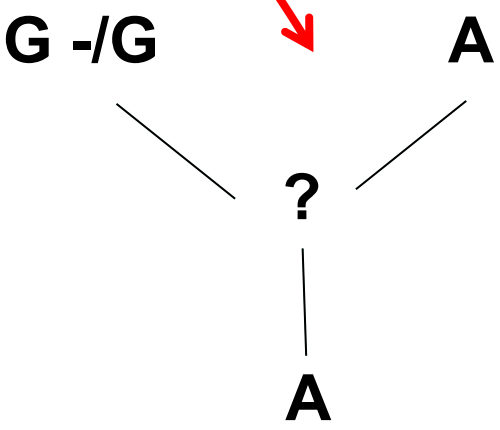
GA	R
GT	K
GC	S
GAT	D
GAC	V
GTC	B
ATC	H
AT	W
AC	M
CT	Y
ACGT	N
ACGT-	X



transitions	1
transversions	1
indels	2

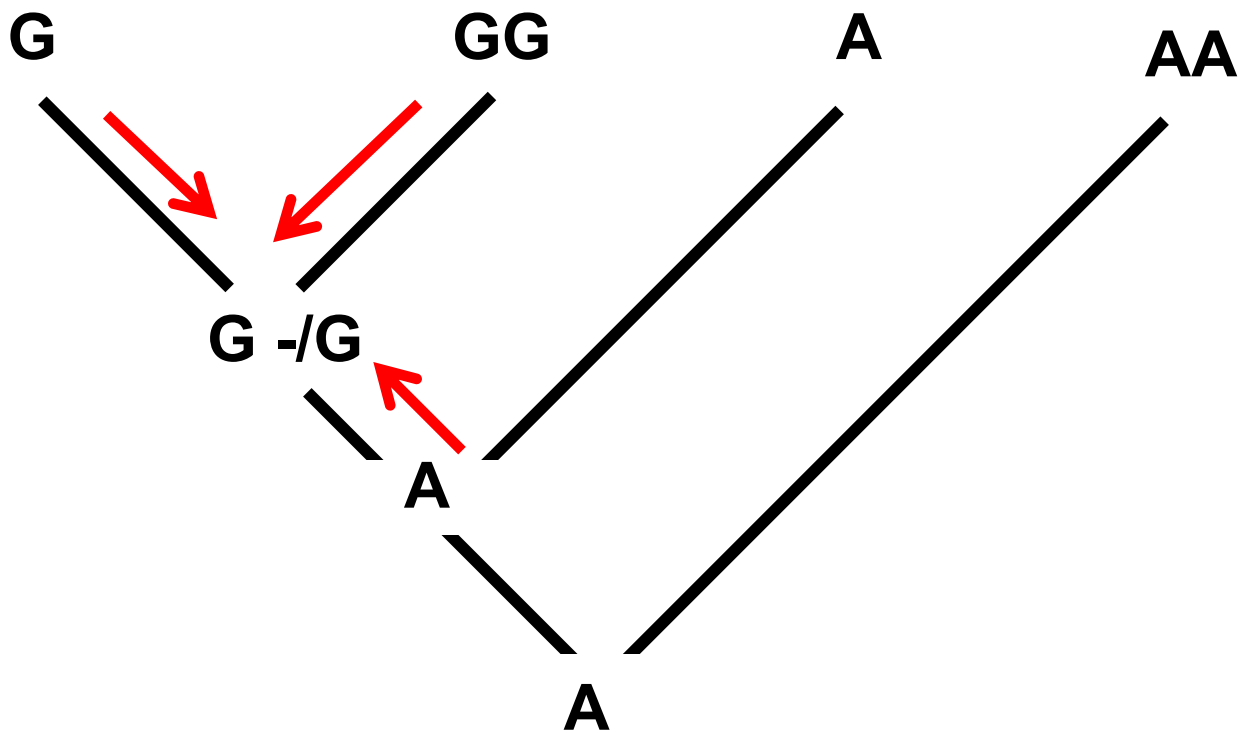
minimum cost

A	1
G	2
GG	6



Direct optimization (DO)

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT

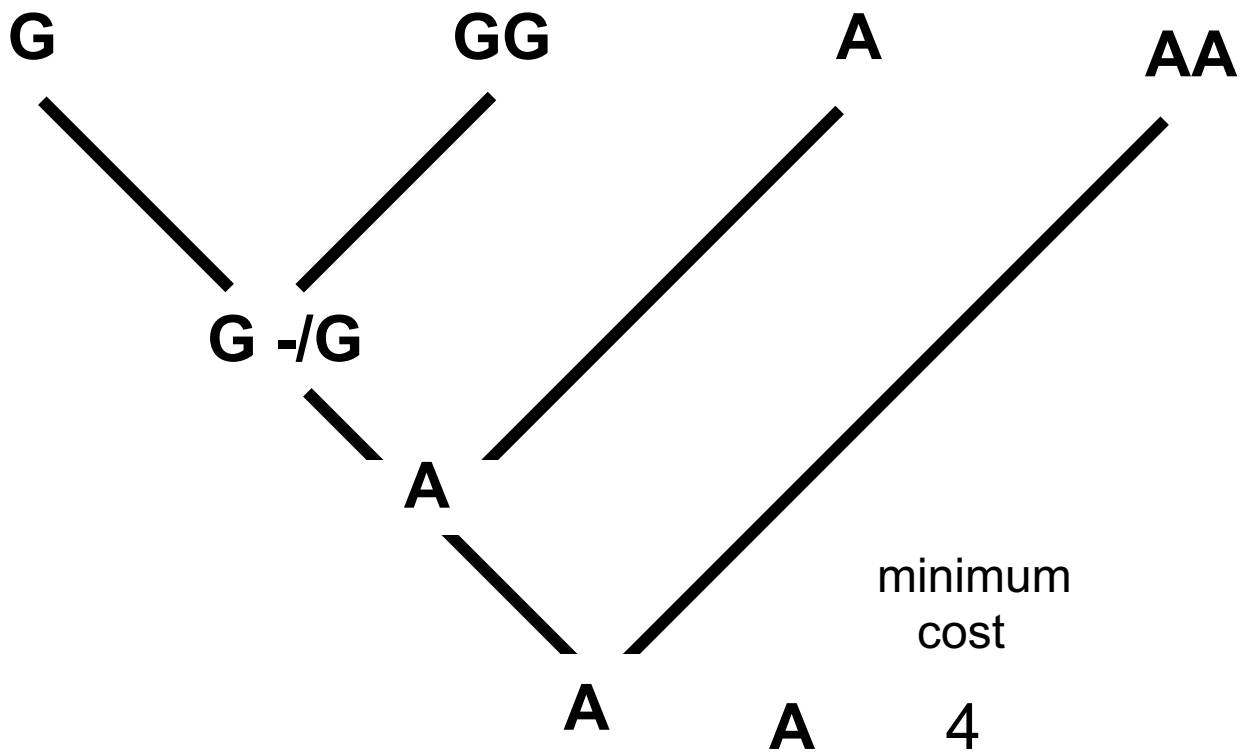


GA	R
GT	K
GC	S
GAT	D
GAC	V
GTC	B
ATC	H
AT	W
AC	M
CT	Y
ACGT	N
ACGT-	X

Direct optimization (DO)

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCTAGGATGC



minimum cost

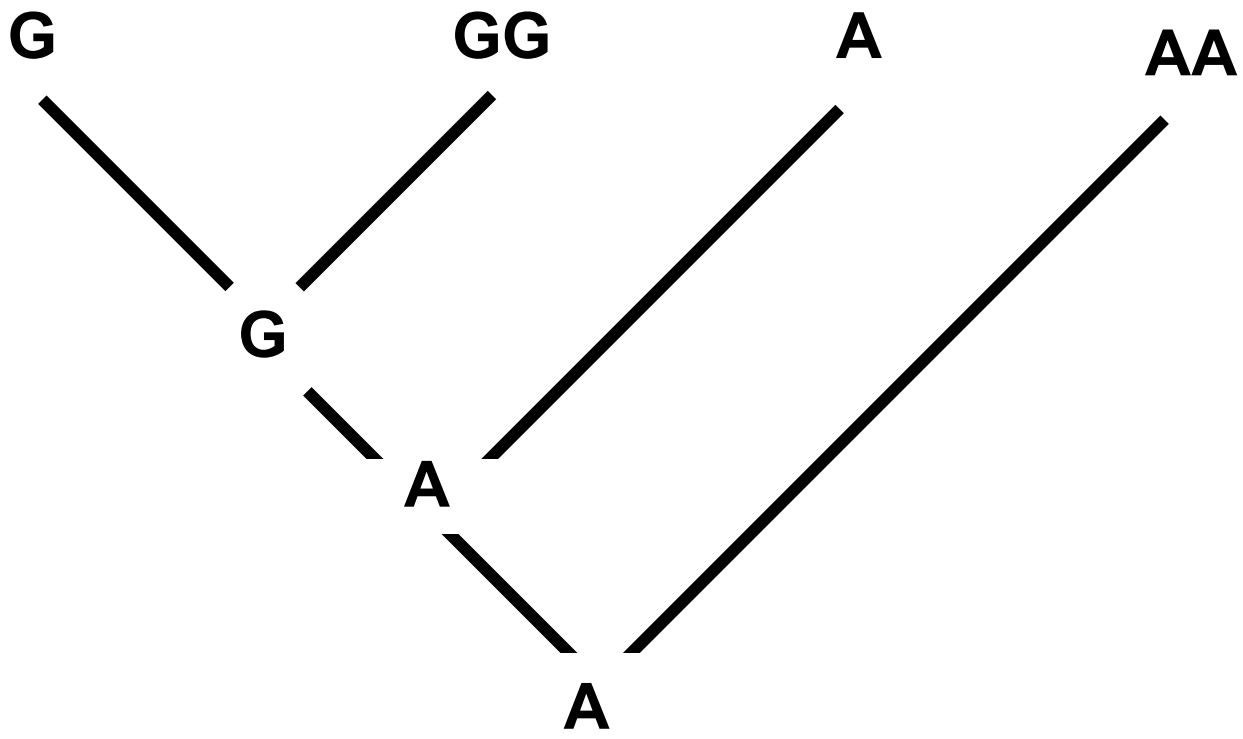
A	4
G	3
GG	5

GA	R
GT	K
GC	S
GAT	D
GAC	V
GTC	B
ATC	H
AT	W
AC	M
CT	Y
ACGT	N
ACGT-	X

Direct optimization (DO)

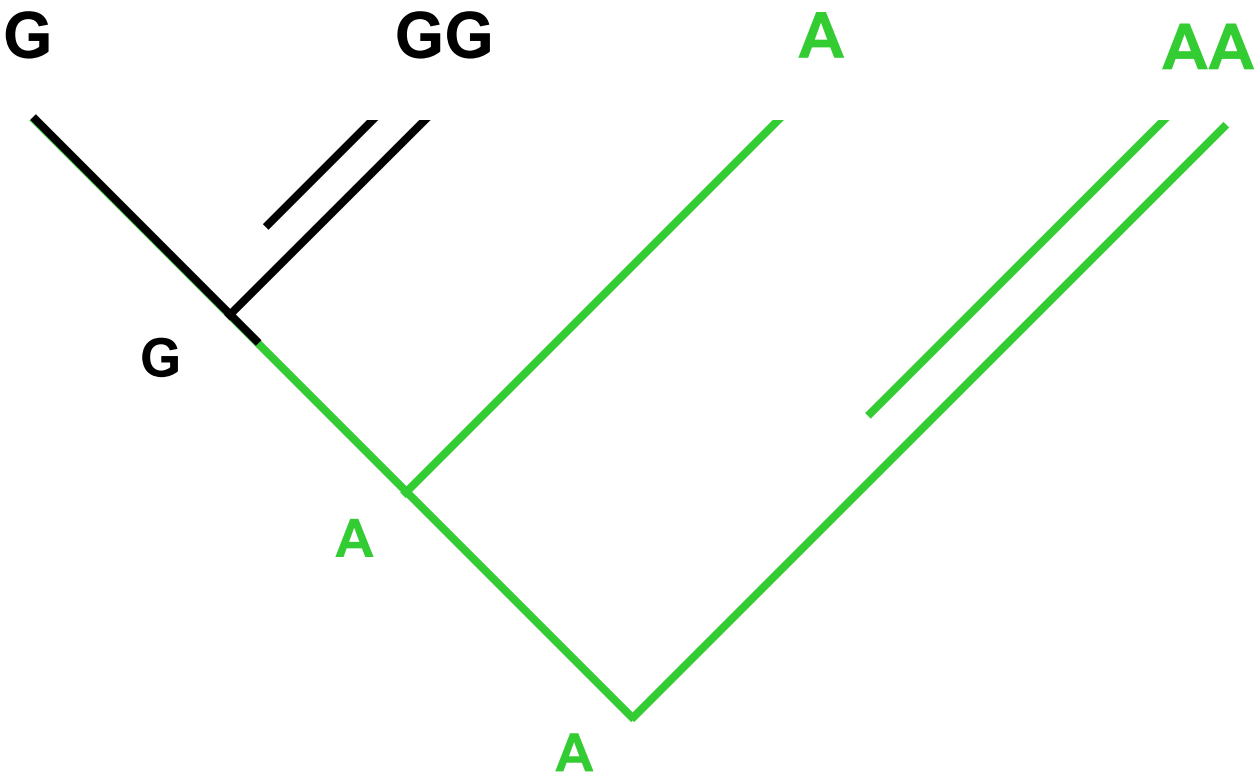
AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCTAGGATGC



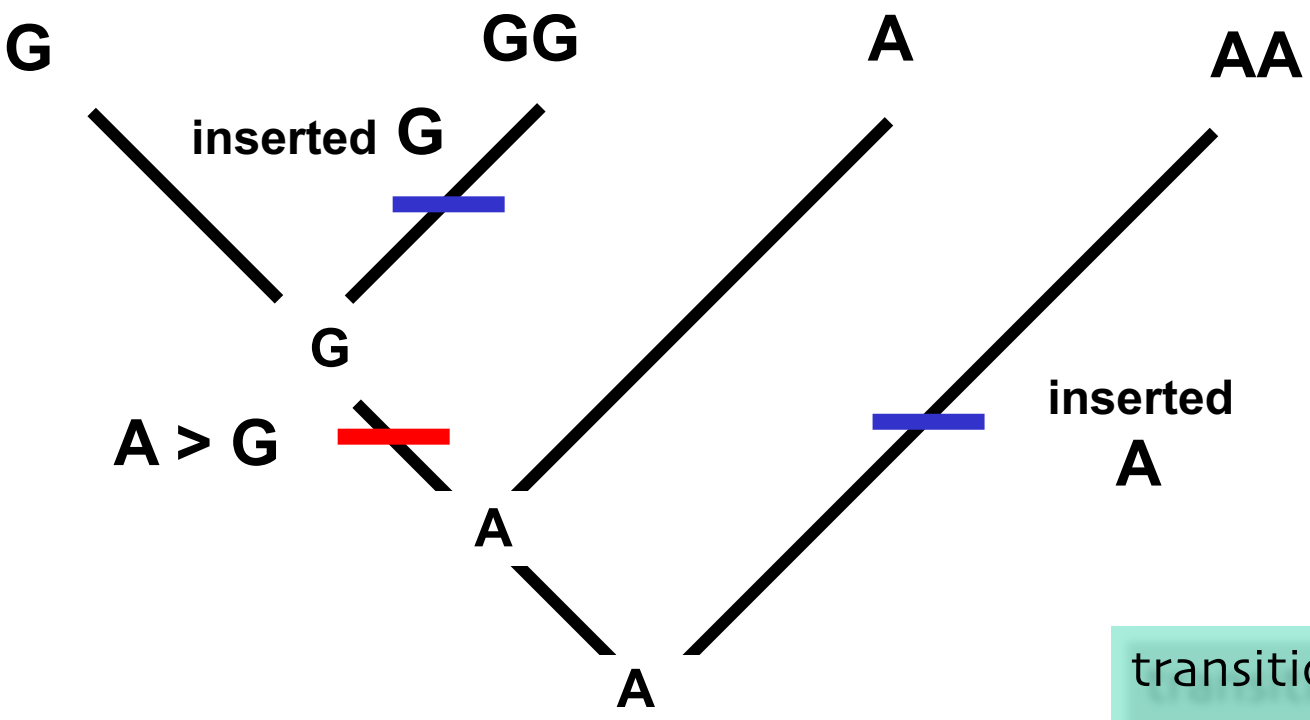
Direct optimization (DO)

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT



Direct optimization (DO)

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT

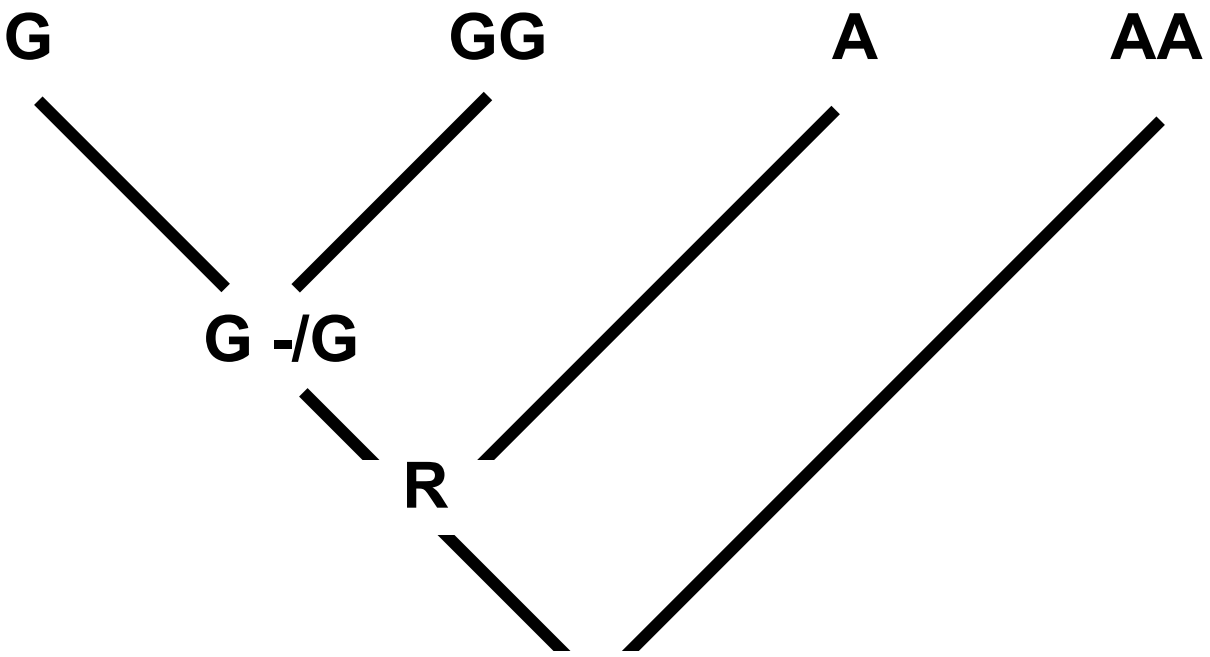


transitions	1
transversions	1
indels	2

total cost 2+1+2=5

Direct optimization (DO)

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT




```
-   R
A   A
```

cost 2

```
R   -
A   A
```

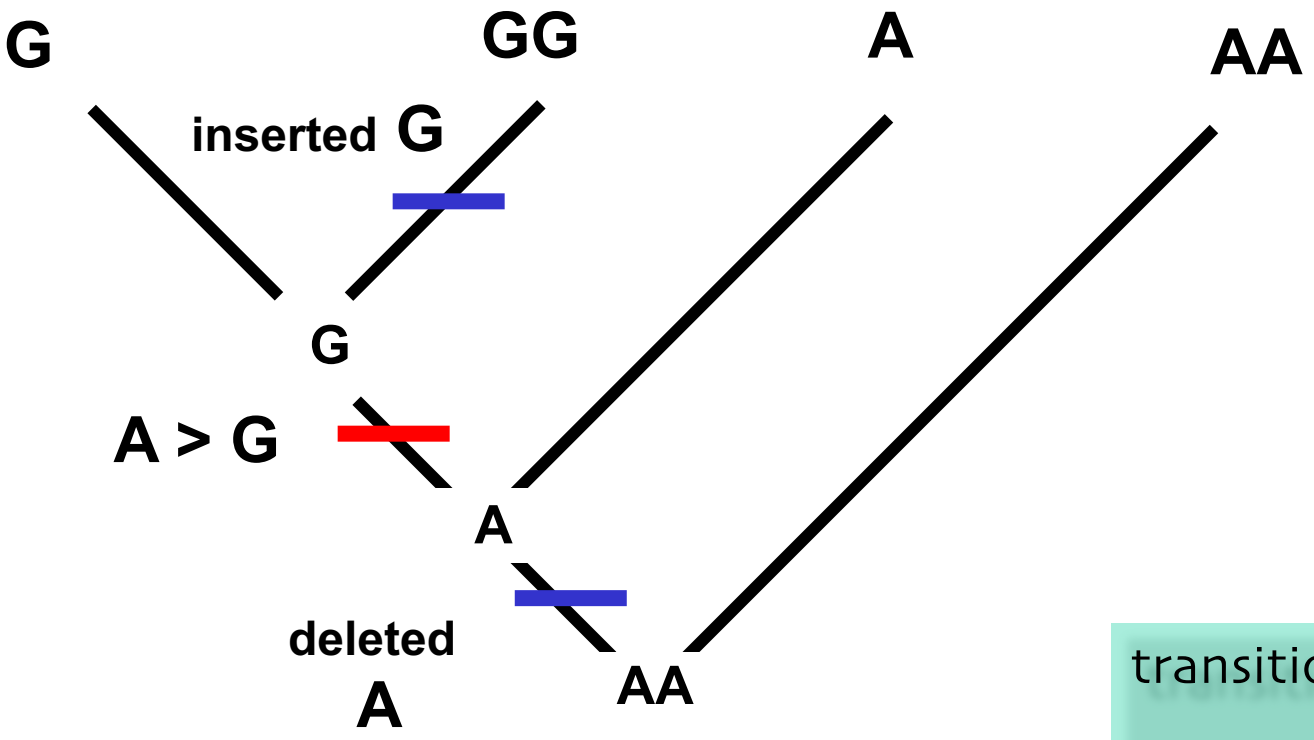
cost 2

GA	R
GT	K
GC	S
GAT	D
GAC	V
GTC	B
ATC	H
AT	W
AC	M
CT	Y
ACGT	N
ACGT-	X

Direct optimization (DO)

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCTAGGATGC



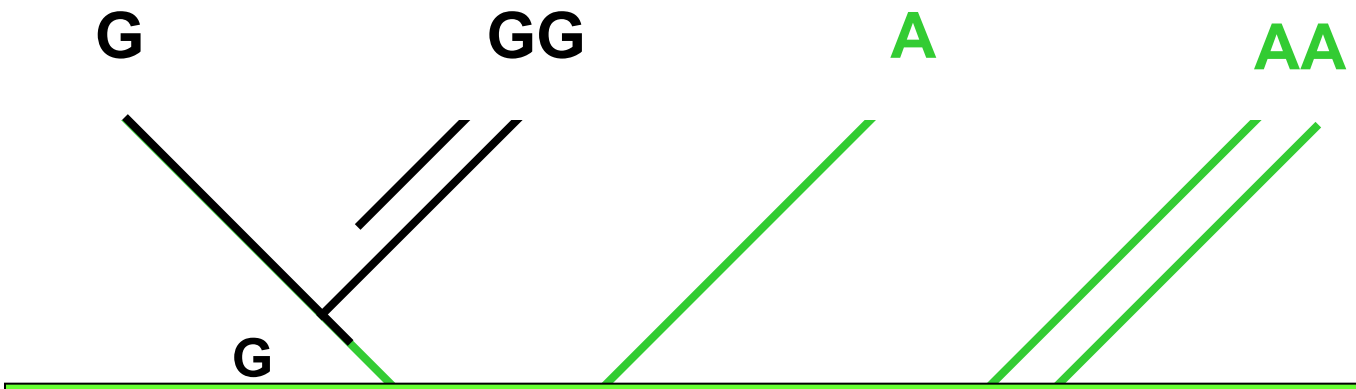
transitions	1
transversions	1
indels	2

total cost 2+1+2=5

Direct optimization (DO)

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCTAGGATGC



...sequence data do not present themselves in neat packages....

....homologies are **TOPOLOGY SPECIFIC**...

Wheeler, W. 2006. Dynamic homology and the likelihood criterion. *Cladistics* 22: 157-170.

AA

ADDITIONAL INFORMATION:

Phillips, A.J. 2006. Homology assessment and molecular sequence alignment. *J. Biomedical Informatics* 39: 18-33.

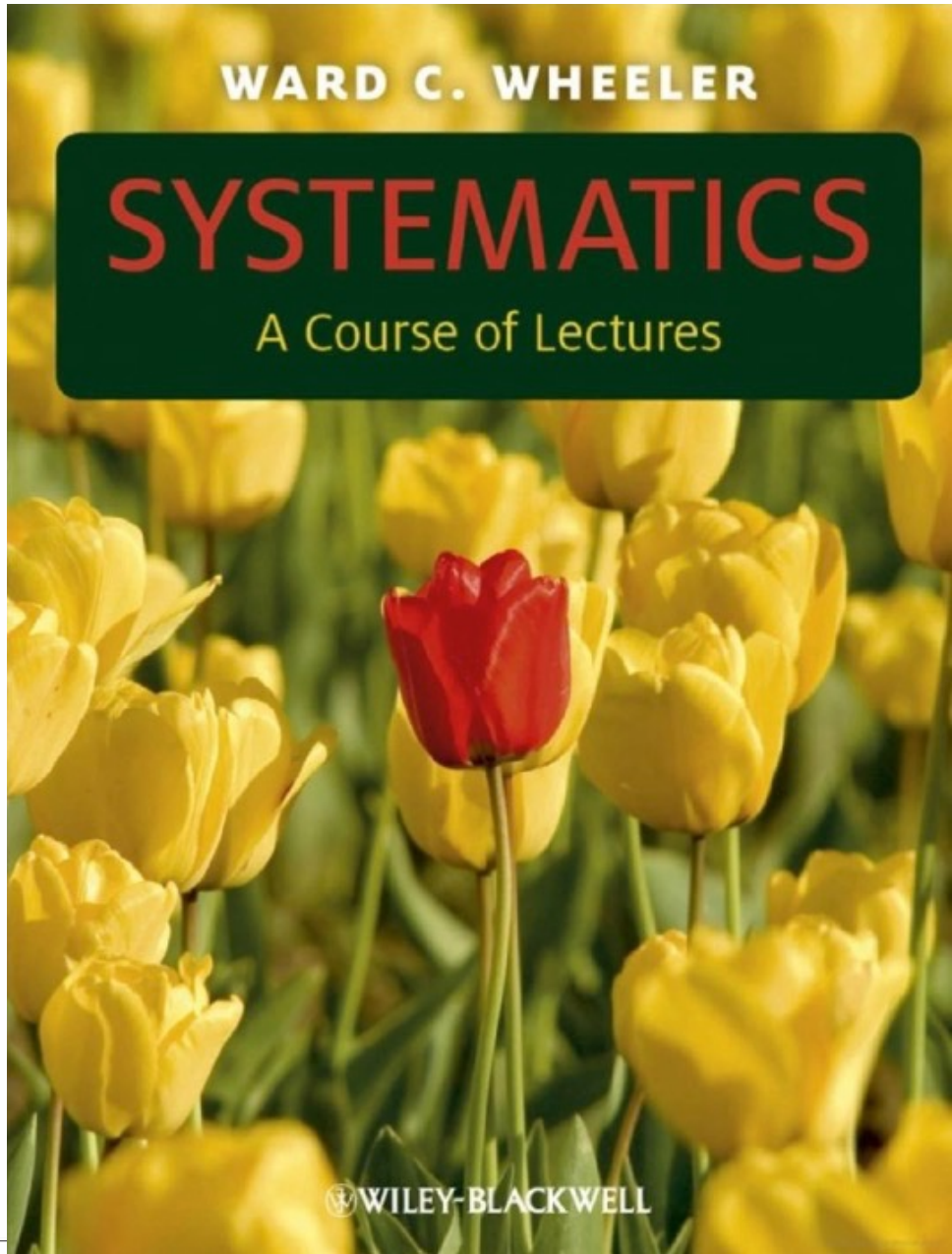
Wheeler, W.C. 2002. "Optimization alignment: down, up, error, and improvements." Pp. 55-69. in R. Desalle et al. eds. Techniques in Molecular Systematics and Evolution. Birkhäuser Verlag, Basel Switzerland.

Wheeler, W. 1996. Optimization alignment: the end of multiple sequence alignment in phylogenetics? *Cladistics* 12: 1-9.

wardwheeler.wordpress.com

Algorithm 10.6: DirectOptimizationFirstPass

Data: Input strings X and Y of lengths $|X|$ and $|Y|$
Data: Element cost matrix σ' of elements nearest to all pairs of elements in Σ and λ (indel) based on element cost matrix σ of Algorithm 8.1
Result: Median cost.
Initialize first row and column of matrices;
$direction\,[0]\,[0] \leftarrow$ '$\searrow$';
$cost\,[0]\,[0] \leftarrow 0$;
$length\,[0]\,[0] \leftarrow 0$;
for $i = 1$ **to** $|X|$ **do**
  $cost\,[i]\,[0] \leftarrow cost\,[i-1]\,[0] + \sigma'_{X_i,\lambda}$;
  $direction\,[i]\,[0] \leftarrow$ '$\rightarrow$';
  $length\,[i]\,[0] \leftarrow length\,[i-1]\,[0] + 1$;
end
for $j = 1$ **to** $|Y|$ **do**
  $cost\,[0]\,[j] \leftarrow cost\,[0]\,[j-1] + \sigma'_{Y_j,\lambda}$;
  $direction\,[0]\,[j] \leftarrow$ '$\downarrow$';
  $length\,[0]\,[j] \leftarrow length\,[0]\,[j-1] + 1$;
end
Update remainder of matrices *cost*, *direction*, and *length*;
for $i = 1$ **to** $|X|$ **do**
  for $j = 1$ **to** $|Y|$ **do**
    $ins \leftarrow cost\,[i-1]\,[j] + \sigma'_{X_i,\lambda}$;
    $del \leftarrow cost\,[i]\,[j-1] + \sigma'_{Y_j,\lambda}$;
    $sub \leftarrow cost\,[i-1]\,[j-1] + \sigma'_{X_i,Y_j}$;
    $cost\,[i]\,[j] \leftarrow \min(ins, del, sub)$;
    if $cost\,[i]\,[j] = ins$ **then**
      $direction\,[i]\,[j] \leftarrow$ '$\rightarrow$';
      $length\,[i]\,[j] \leftarrow length\,[i-1]\,[j] + 1$;
    else if $cost\,[i]\,[j] = del$ **then**
      $direction\,[i]\,[j] \leftarrow$ '$\downarrow$';
      $length\,[i]\,[j] \leftarrow length\,[i]\,[j-1] + 1$;
    else
      $direction\,[i]\,[j] \leftarrow$ '$\searrow$';
      $length\,[i]\,[j] \leftarrow length\,[i-1]\,[j-1] + 1$;
    end
  end
end
return $cost\,[|X|]\,[|Y|]$



Homology - static or dynamic?

conventional alignment

direct optimization

```
terminal   1     TATACTTT
           2     ----CT-C
           3     TA--C---
```

POSITIONS of nucleotides **FIXED before** phylogenetic analysis

Homology - static or dynamic?

AACGGTTTAAGGTACGGAGAATTAGGCAACCCTAGGATGCGATGCGCAGAGTTAGGTACGGAGAATTAGGCAACCCGGTAGGAT

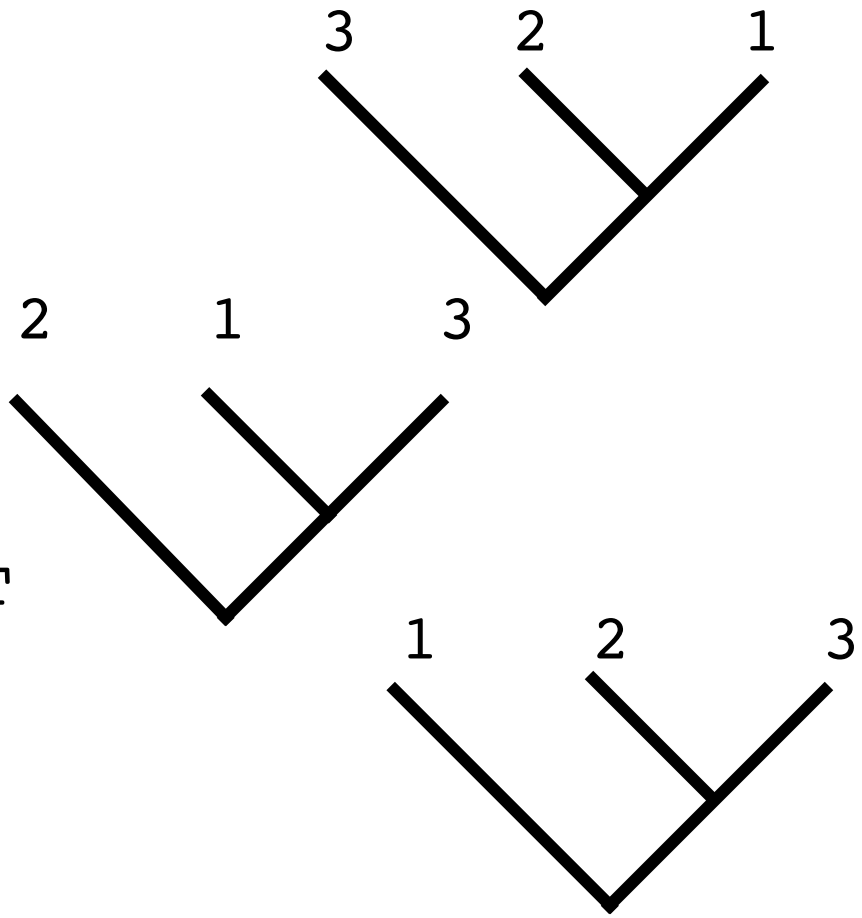
conventional alignment

direct optimization

terminal 1 TATACTTT
2 ----CT-C

positions of nucleotides ARE *FREE to change during phylogenetic analysis*

```
terminal   1     TATACTTT
           2     CTC
           3     TAC
```



Number of possible alignments

f(n, m) for $1 \leq n \leq 10$; $1 \leq m \leq 5$

n\m	2	3	4	5
1	3	13	75	541
2	13	409	23917	2244361
3	63	16081	10681263	14638756721
4	321	699121	5552351121	117629959485121
5	1683	32193253	3147728203035	1.05×10^{18}
10	8097453	985034974418272 9	3.32×10^{26}	1.35×10^{38}

Slowinski (1998)

Finding THE optimal alignment for even a small number of short sequences is IMPOSSIBLE

T	B(T)
3	1
4	3
5	15
6	105
7	945
8	10 395
9	135 135
10	2×10^6
15	8×10^{12}
20	2×10^{20}
50	3×10^{74}

$$B(n) = (2n-5)!!$$

in DO these
2 NP complete
problems are coupled

computationally **VERY**
DEMANDING

only heuristic "solutions"

Direct optimization (DO)

do we search homology

between nucleotides

AAATCGCGGATT

AACTCCCGGAGT

AACTCGC-GAGT

base to base homology

or between STRETCHES of nucleotides

fragment homology

AAATCGCGGATT

AACTCCCGGAGT

AACTCGCGAGT

Direct optimization

base to base homology

POSITION of single nucleotide = character

different nucleotides (A C G T) plus alignment gaps = character states

AAATCGCGGATT

AACTCCCGGAGT

AACTCGC-GAGT

12 characters

A C G T & - 5 character states

Fixed states optimization (FSO)

Wheeler, W.C. 1999. Cladistics 15: 379-385

fragment homology

stretch of DNA = character

homologous stretches in different terminals
= character states

AAATCGCGGATTT

AACTCGCGGAGTT

AACTCGCGAGTT

1 character

number of character states $x \leq n$

n = number of terminals studied

Direct optimization (DO)

we try to figure out homologies at the level alphabets

P A N T S	P - A N T S
A N T S	- - A N T S
P L A N T	P L A N T S
R A T S	R - A - T S
B A T S	B - A - T S

Fixed states optimization (FSO)

we do NOT try to figure out homologies between alphabets

instead we are comparing whole WORDS

each stretch to be compared potentially different character state

number of terminals to be compared = n

number of possible characters states = x

$x \leq n$

P A N T S

D U N G A R E E S

T R O U S E R S

S L A C K S

J E A N S

Fixed states optimization (FSO)

1) pair-wise comparison of sequences

2) cost of transformation determined between each pair
(= cost between different character STATES)

this is based on the nucleotide composition of each stretch

3) each stretch (= character state) located on internal nodes of trees

4) the tree with lowest overall cost is chosen

Fixed states optimization (FSO)

Advantage: large number of character states
origin of exactly same kind of sequence in 2 unrelated terminals is VERY unlikely

in DO only 5 states

A C G T -

Disadvantage: subjectivity in determination of characters, how long, what kind of stretches?

in DO this is NOT a problem, nucleotides CANNOT be reduced any further

METRICITY is required between character states

Fixed state optimization (FSO)

AATTT

ATTAA

TT

terminals with numerous missing entries might have SAME EDIT cost with MANY other terminals

this might lead to violation of triangle inequality

$d(x,y) \leq d(x,z) + d(z,y)$

determination of adjacency of character states mathematically IMPOSSIBLE

TAATT

…optimization will lead to inconsistent results & thus characters states have to fulfill the following requirements:

$\forall x$ $\quad d(x,x) = 0$

$\forall x, y; x \neq y$ $\quad d(x, y) > 0$

$\forall x, y$ $\quad d(x,y) = d(y,x)$

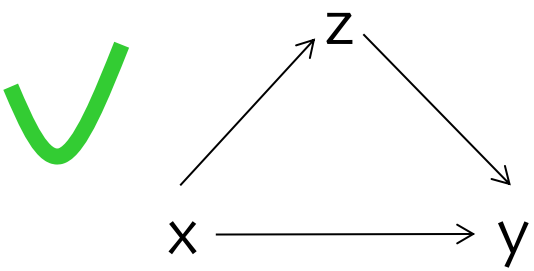
$\forall x, y, z$ $\quad d(x,y) \leq d(x, z) + d(z, y)$

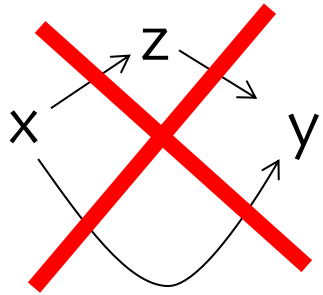
x, y, z character states

d change from ch. state to another

i.e.

1) change from one ch. state to another > 0
2) change x → y = y → x (changes symmetrical)
3) change x → y ≤ change x → z + change z → y (triangle inequality)





SUMMARY

number of possible alignments is **VAST**

problems of multiple alignment & tree construction are inextricably linked

dynamic homology enables more parsimonious interpretation of sequence level data

DO computationally demanding

new algorithms & techniques are continuously developed

HOME "QUIZ"

Why, in this age of whole genome sequences, it is still relevant to study & use also morphological characters in phylogenetic analyses

Why does cutting of continuous sequence stretches within CONSERVED lead into huge speed-up in DO analyses?

come up with arguments for the 1st demo Mon 14.xi.