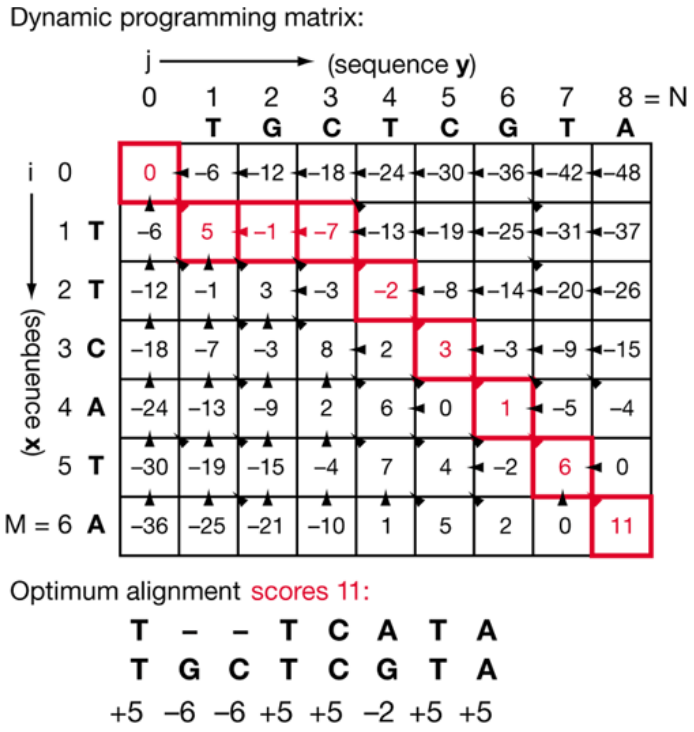
CS342: Bioinformatics
Lecture 10



Dynamic programming matrix:

j → (sequence **y**)

i ↓ (sequence **x**)

i		0	1	2	3	4	5	6	7	8 = N
			T	**G**	**C**	**T**	**C**	**G**	**T**	**A**
0		0	−6	−12	−18	−24	−30	−36	−42	−48
1	**T**	−6	5	−1	−7	−13	−19	−25	−31	−37
2	**T**	−12	−1	3	−3	−2	−8	−14	−20	−26
3	**C**	−18	−7	−3	8	2	3	−3	−9	−15
4	**A**	−24	−13	−9	2	6	0	1	−5	−4
5	**T**	−30	−19	−15	−4	7	4	−2	6	0
M = 6	**A**	−36	−25	−21	−10	1	5	2	0	11

Optimum alignment scores 11:

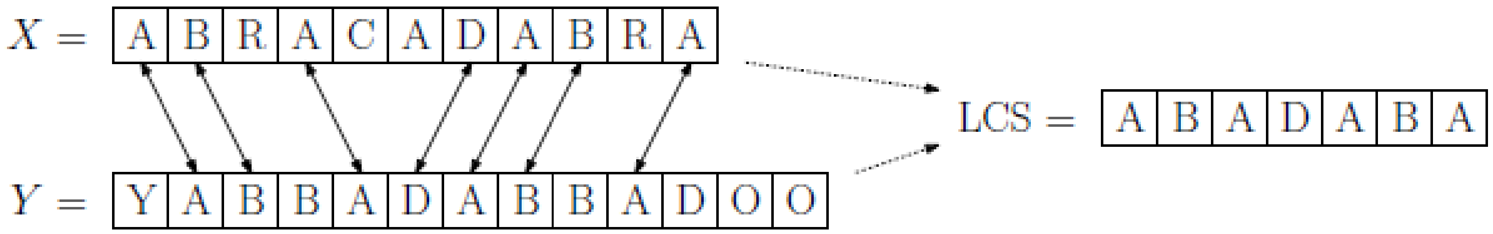
```
 T   –   –   T   C   A   T   A
 T   G   C   T   C   G   T   A
+5  −6  −6  +5  +5  −2  +5  +5
```

Longest Common Subsequence (LCS)

Given two sequences $X = \langle x_1, x_2, \ldots, x_m \rangle$ and $Z = \langle z_1, z_2, \ldots, z_k \rangle$, we say that Z is a subsequence of X if there is a strictly increasing sequence of k indices $\langle i_1, i_2, \ldots, i_k \rangle$ $(1 \leq i_1 < i_2 < \ldots < i_k \leq m)$ such that $Z = \left\langle x_{i_1}, x_{i_2}, \ldots, x_{i_k} \right\rangle$

For example, let X = <ABRACADABRA> and let Z = <AADAA>, then Z is a subsequence of X.

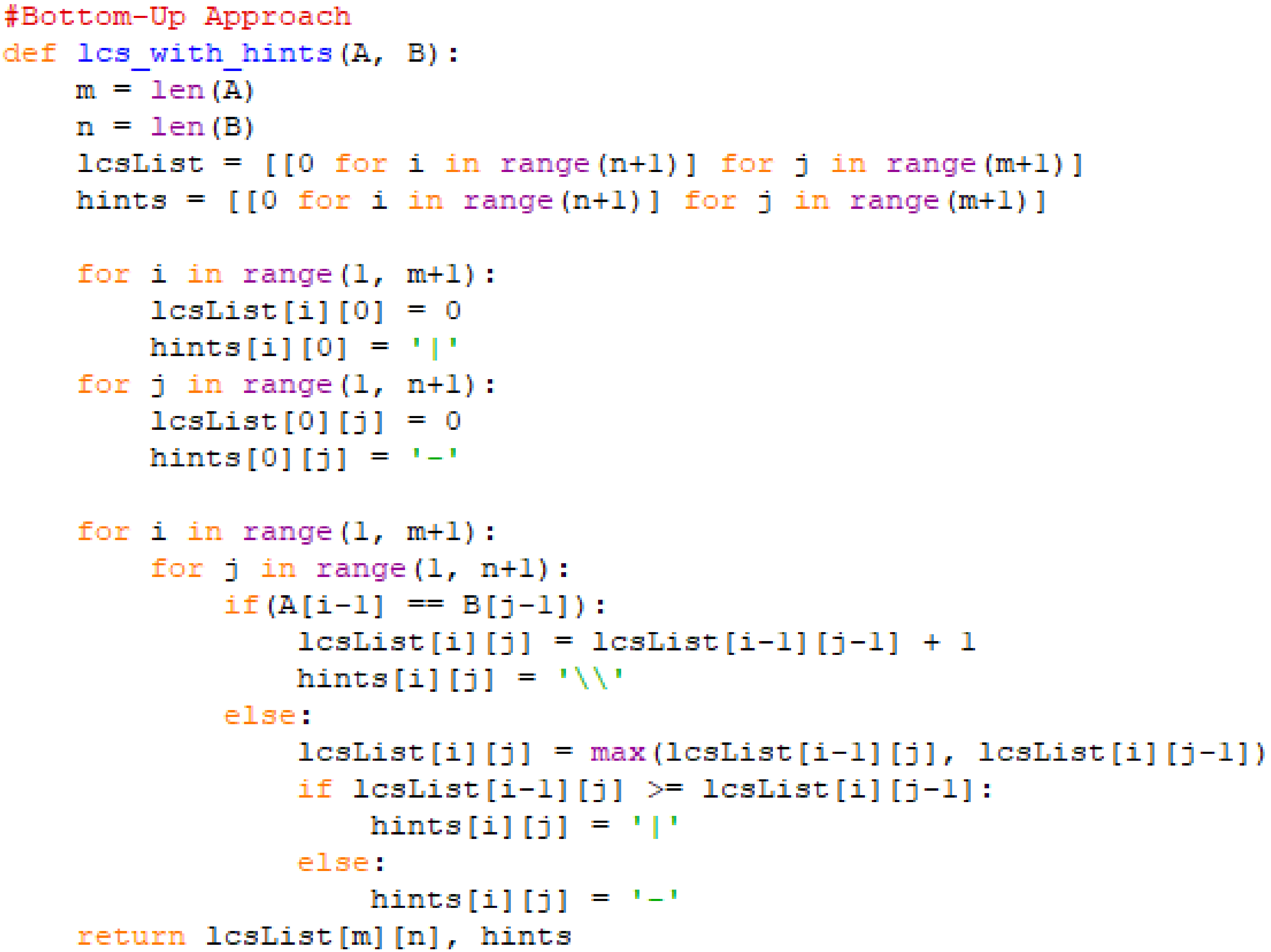
LCS Problem: Given two sequences $X = \langle x_1, \ldots, x_m \rangle$ and $Y = \langle y_1, \ldots, y_n \rangle$ determine the length of their longest common subsequence, and more generally the sequence itself.



$X =$ A B R A C A D A B R A

$Y =$ Y A B B A D A B B A D O O

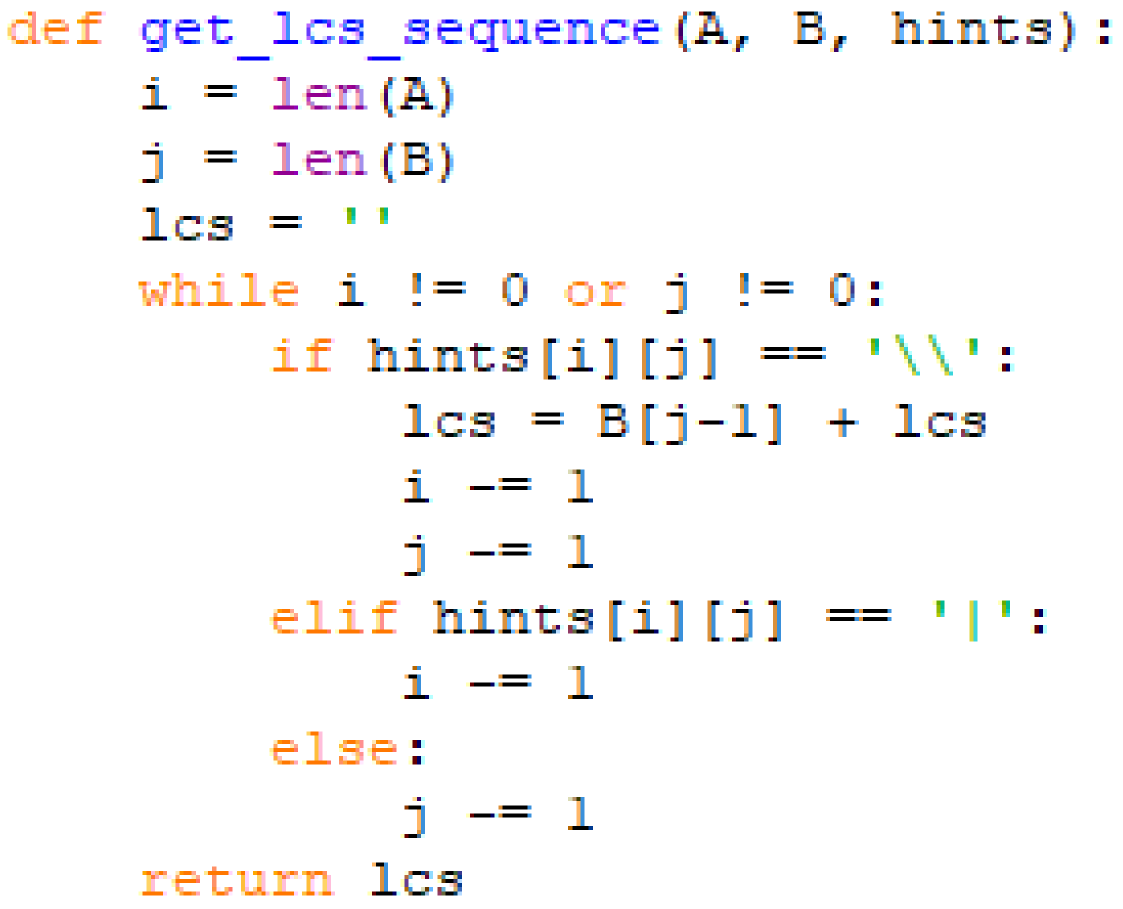
LCS = A B A D A B A



```python
#Bottom-Up Approach
def lcs_with_hints(A, B):
    m = len(A)
    n = len(B)
    lcsList = [[0 for i in range(n+1)] for j in range(m+1)]
    hints = [[0 for i in range(n+1)] for j in range(m+1)]

    for i in range(1, m+1):
        lcsList[i][0] = 0
        hints[i][0] = '|'
    for j in range(1, n+1):
        lcsList[0][j] = 0
        hints[0][j] = '-'

    for i in range(1, m+1):
        for j in range(1, n+1):
            if(A[i-1] == B[j-1]):
                lcsList[i][j] = lcsList[i-1][j-1] + 1
                hints[i][j] = '\\'
            else:
                lcsList[i][j] = max(lcsList[i-1][j], lcsList[i][j-1])
                if lcsList[i-1][j] >= lcsList[i][j-1]:
                    hints[i][j] = '|'
                else:
                    hints[i][j] = '-'
    return lcsList[m][n], hints
```



```python
def get_lcs_sequence(A, B, hints):
    i = len(A)
    j = len(B)
    lcs = ''
    while i != 0 or j != 0:
        if hints[i][j] == '\\':
            lcs = B[j-1] + lcs
            i -= 1
            j -= 1
        elif hints[i][j] == '|':
            i -= 1
        else:
            j -= 1
    return lcs
```

LCS Example

LCS length

		0	1	2	3	$4 = n$
			B	D	C	B
0		0	0	0	0	0
1	B	0	1	1	1	1
2	A	0	1	1	1	1
3	C	0	1	1	2	2
4	D	0	1	2	2	2
$m = 5$	B	0	1	2	2	3

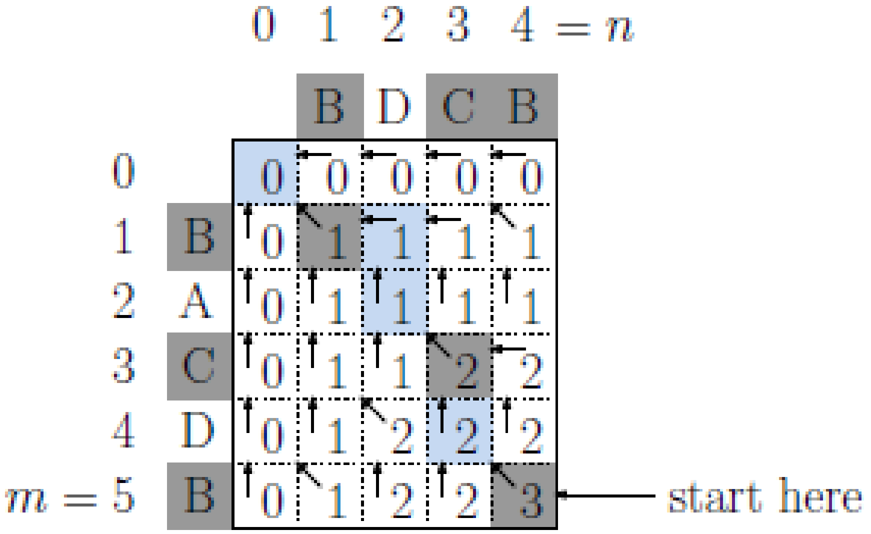
(a)

$X = \langle \text{BACDB} \rangle$

$Y = \langle \text{BDCB} \rangle$

$\text{LCS} = \langle \text{BCB} \rangle$

... with hints



		0	1	2	3	$4 = n$
			B	D	C	B
0		0	←0	←0	←0	←0
1	B	↑0	↖1	←1	←1	↖1
2	A	↑0	↑1	↑1	↑1	↑1
3	C	↑0	↑1	↑1	↖2	←2
4	D	↑0	↑1	↖2	↑2	↑2
$m = 5$	B	↑0	↖1	↑2	↑2	↖3

(b)

Biology

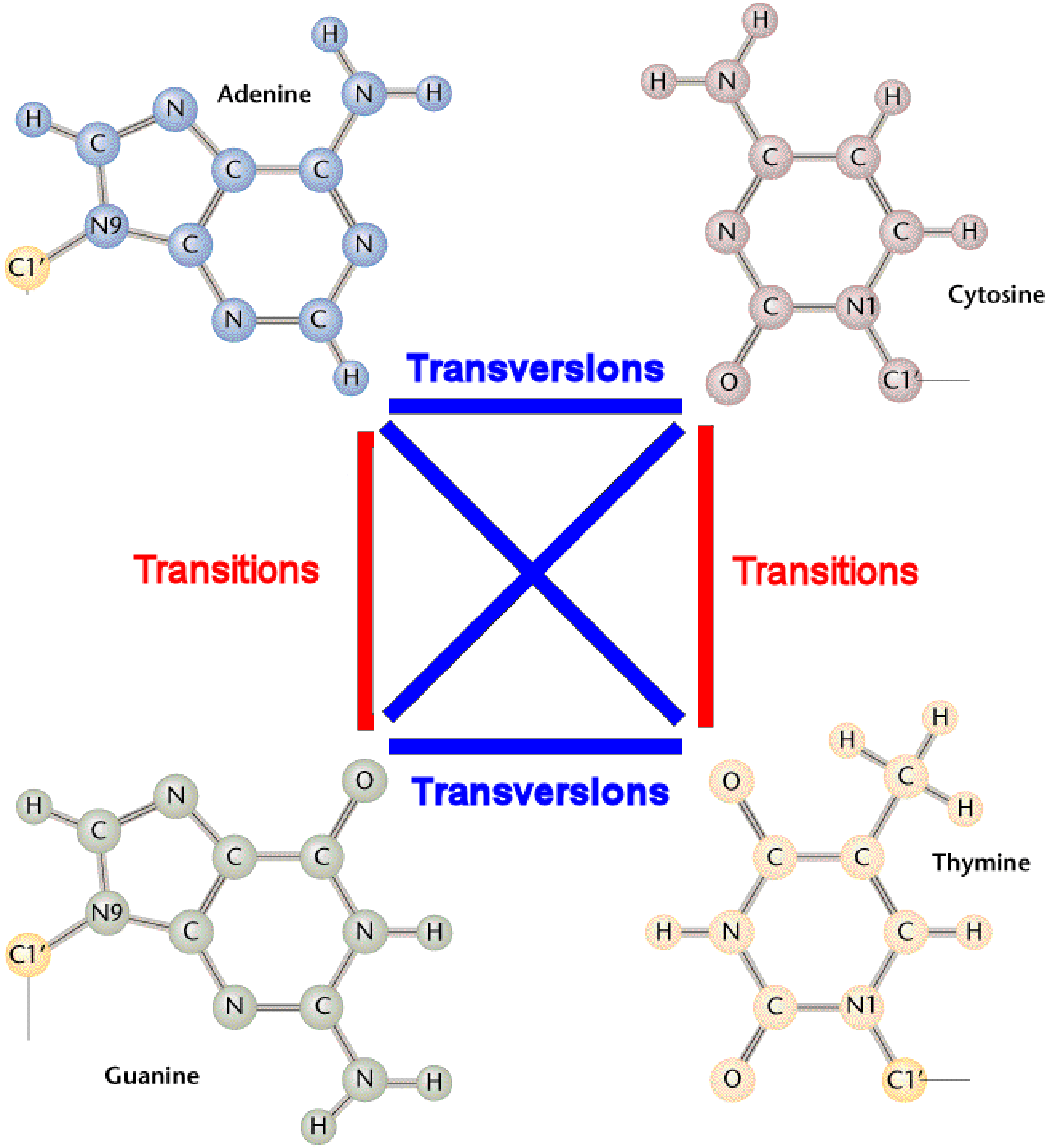
Transitions: A ← → G, C←→T

Transversions: A ← → C, A←→T , G ←→ C, G ←→ T

Transitions are interchanges of two-ring purines (e.g., A ← → G) or one ring pyrimidines (C←→T).

Transversions are interchanges of purine for pyrimidines, so change of one ring for two ring structures.

Takeaway: Transitions happen more frequently than transversions, and are less likely to result in an amino acid substitution.



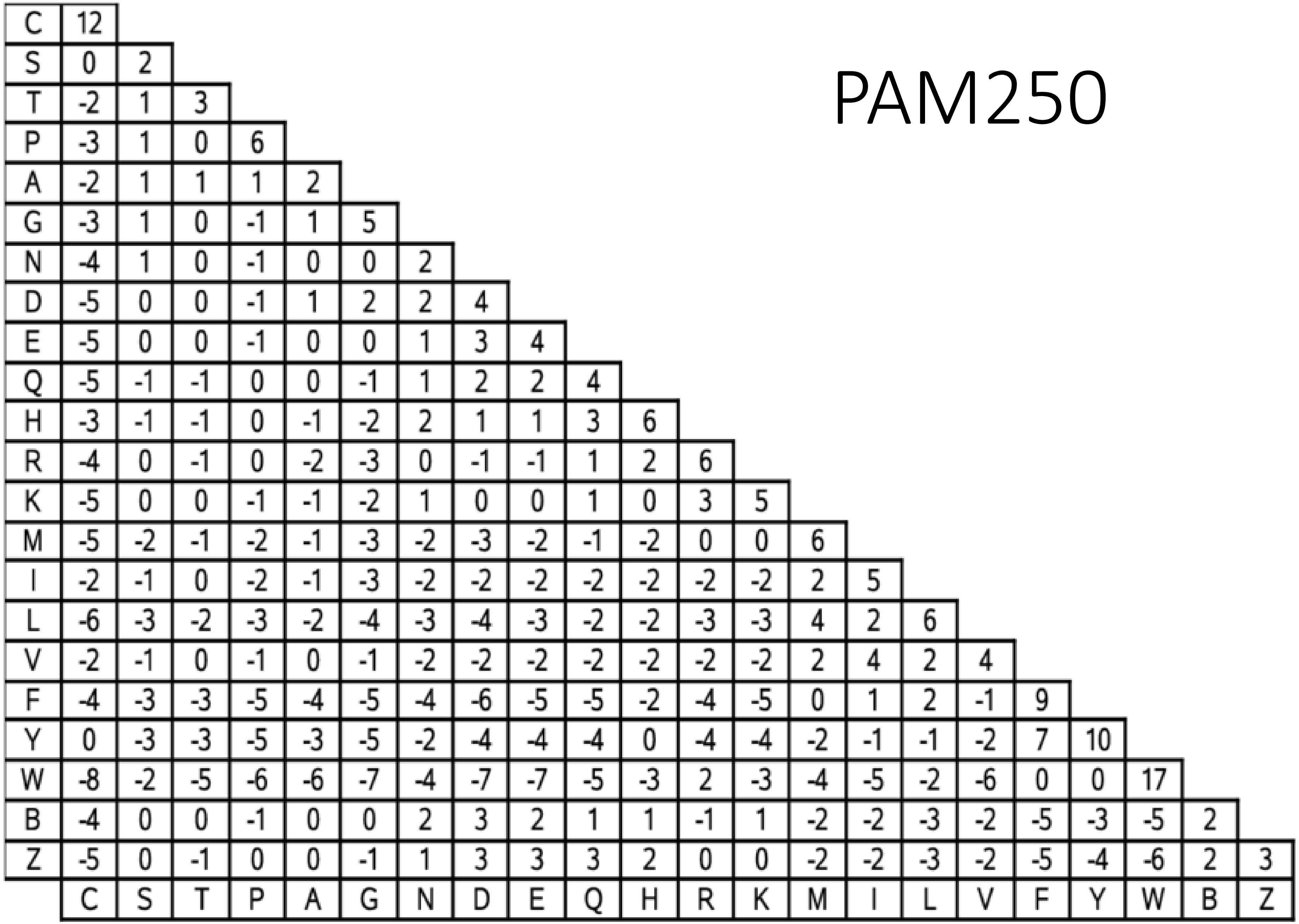
BLOSUM 62 scoring matrix

(positive values are shaded)

A	**4**																			
R	-1	**5**																		
N	-2	0	**6**																	
D	-2	-2	**1**	**6**																
C	0	-3	-3	-3	**9**															
Q	-1	**1**	0	0	-3	**5**														
E	-1	0	0	**2**	-4	**2**	**5**													
G	0	-2	0	-1	-3	-2	-2	**6**												
H	-2	0	**1**	-1	-3	0	0	-2	**8**											
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	**4**										
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	**2**	**4**									
K	-1	**2**	0	-1	-3	**1**	**1**	-2	-1	-3	-2	**5**								
M	-1	-1	-2	-3	-1	0	-2	-3	-2	**1**	**2**	-1	**5**							
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	**6**						
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	**7**					
S	**1**	-1	**1**	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	**4**				
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	**1**	**5**			
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	**1**	-4	-3	-2	**11**		
Y	-2	-2	-2	-3	-2	-1	-2	-3	**2**	-1	-1	-2	-1	**3**	-3	-2	-2	**2**	**7**	
V	0	-3	-3	-3	-1	-2	-2	-3	-3	**3**	**1**	-2	**1**	-1	-2	-2	0	-3	-1	**4**
	A	**R**	**N**	**D**	**C**	**Q**	**E**	**G**	**H**	**I**	**L**	**K**	**M**	**F**	**P**	**S**	**T**	**W**	**Y**	**V**

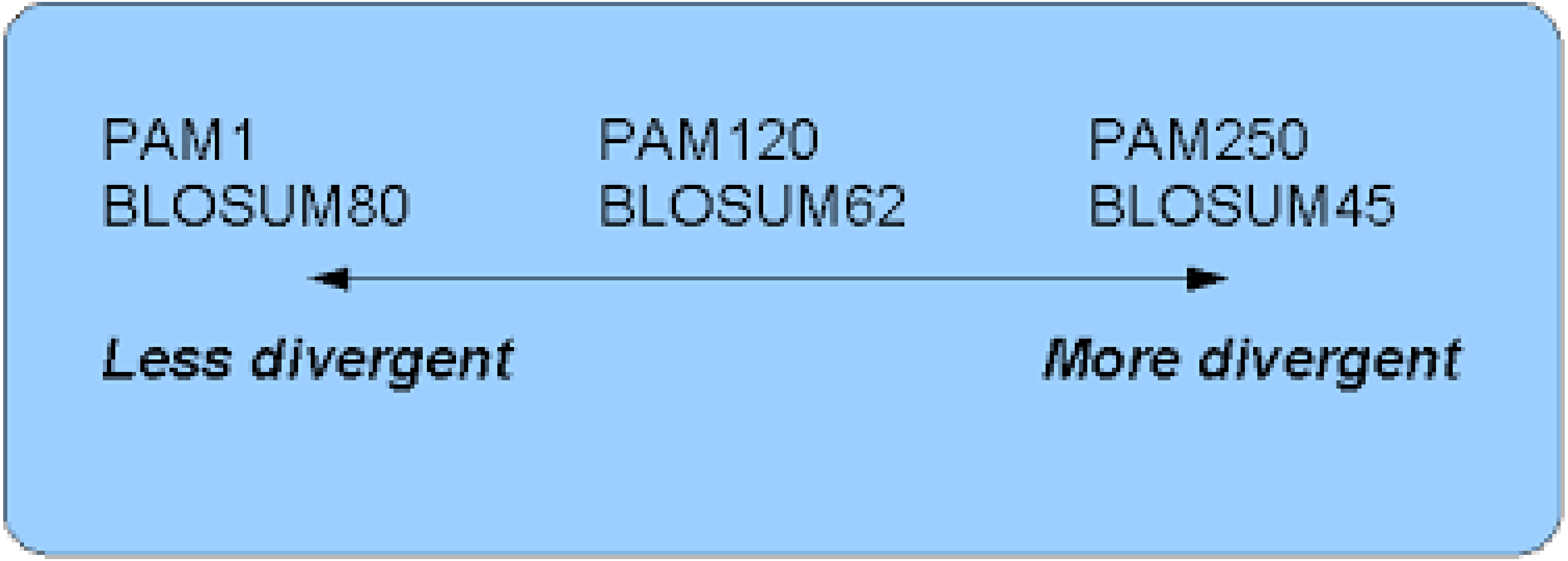
The values for amino acid substitutions were obtained from Henikoff S & Henikoff JG (1992) Amino acid substitutions matrices from protein blocks. *Proc. Natl. Acad. Sci.* **89**: 10915-10919.

PAM250



C	12																					
S	0	2																				
T	-2	1	3																			
P	-3	1	0	6																		
A	-2	1	1	1	2																	
G	-3	1	0	-1	1	5																
N	-4	1	0	-1	0	0	2															
D	-5	0	0	-1	1	2	2	4														
E	-5	0	0	-1	0	0	1	3	4													
Q	-5	-1	-1	0	0	-1	1	2	2	4												
H	-3	-1	-1	0	-1	-2	2	1	1	3	6											
R	-4	0	-1	0	-2	-3	0	-1	-1	1	2	6										
K	-5	0	0	-1	-1	-2	1	0	0	1	0	3	5									
M	-5	-2	-1	-2	-1	-3	-2	-3	-2	-1	-2	0	0	6								
I	-2	-1	0	-2	-1	-3	-2	-2	-2	-2	-2	-2	-2	2	5							
L	-6	-3	-2	-3	-2	-4	-3	-4	-3	-2	-2	-3	-3	4	2	6						
V	-2	-1	0	-1	0	-1	-2	-2	-2	-2	-2	-2	-2	2	4	2	4					
F	-4	-3	-3	-5	-4	-5	-4	-6	-5	-5	-2	-4	-5	0	1	2	-1	9				
Y	0	-3	-3	-5	-3	-5	-2	-4	-4	-4	0	-4	-4	-2	-1	-1	-2	7	10			
W	-8	-2	-5	-6	-6	-7	-4	-7	-7	-5	-3	2	-3	-4	-5	-2	-6	0	0	17		
B	-4	0	0	-1	0	0	2	3	2	1	1	-1	1	-2	-2	-3	-2	-5	-3	-5	2	
Z	-5	0	-1	0	0	-1	1	3	3	3	2	0	0	-2	-2	-3	-2	-5	-4	-6	2	3
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	B	Z

PAM and BLOSUM



PAM1 BLOSUM80

PAM120 BLOSUM62

PAM250 BLOSUM45

Less divergent

More divergent