

PROGRAMMING and ALGORITHMS II



INTRODUCTION TO **DYNAMIC PROGRAMMING II**

Dr Tilo Burghardt

Unit Code COMS10001

Algorithm Design Paradigms

DIVIDE & CONQUER

Break down a problem into independent sub-problems of related type, solve them separately and combine the solutions.

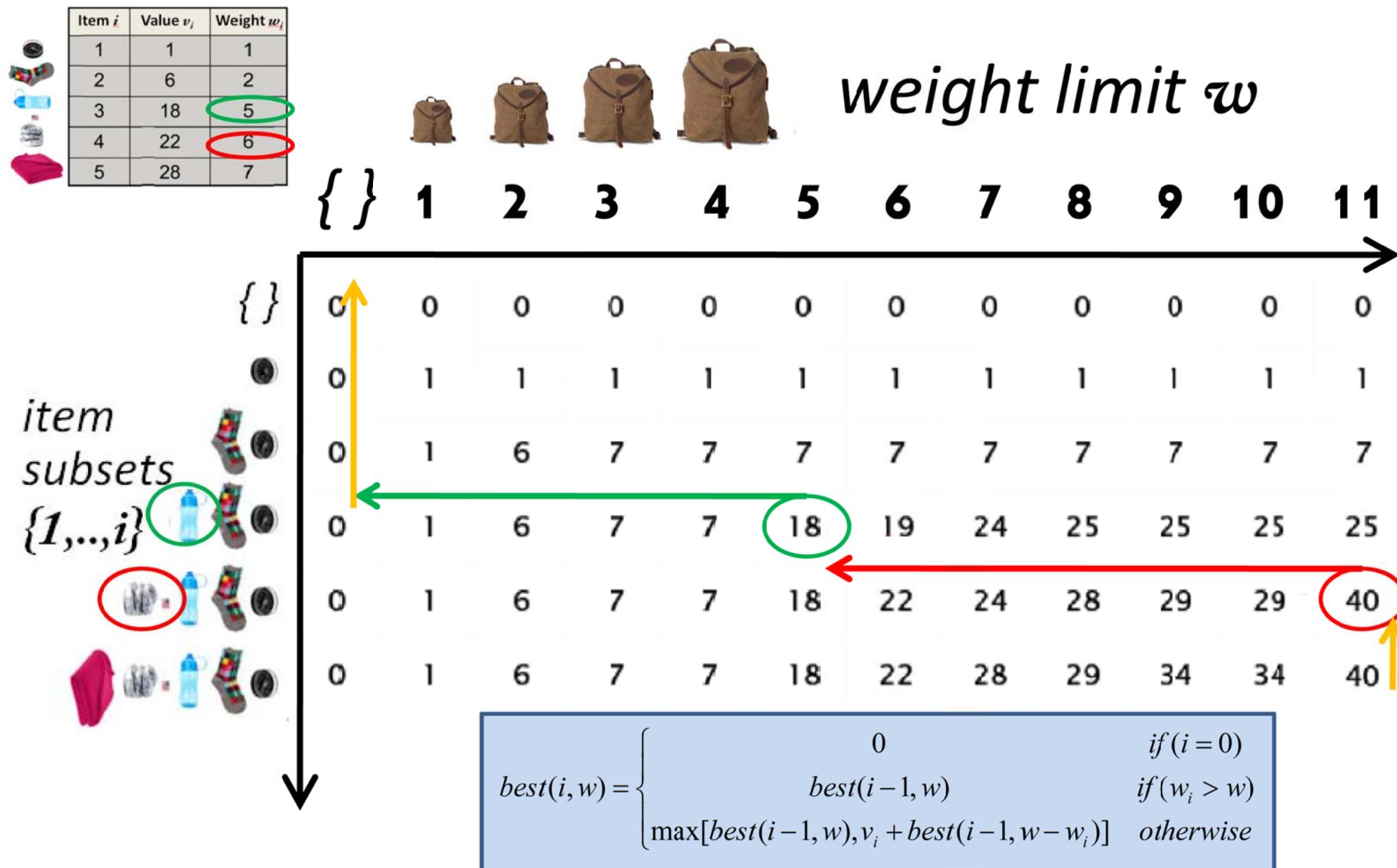
GREEDY APPROACH

Use a sequence of locally optimal decisions incrementally to build up a solution.

DYNAMIC PROGRAMMING

Break down a problem into overlapping sub-problems of related type, build up solutions from larger and larger sub-solutions.

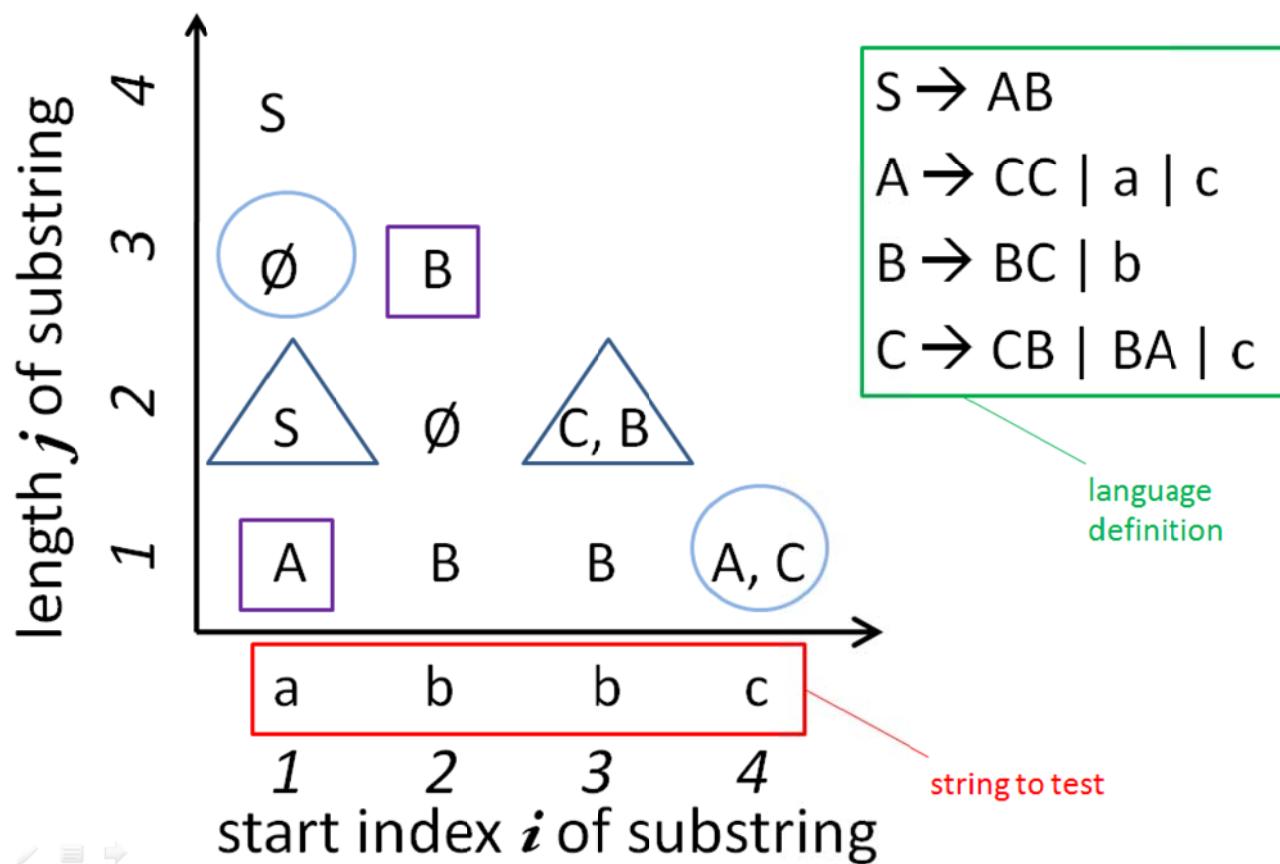
Recap: Knapsack Problem



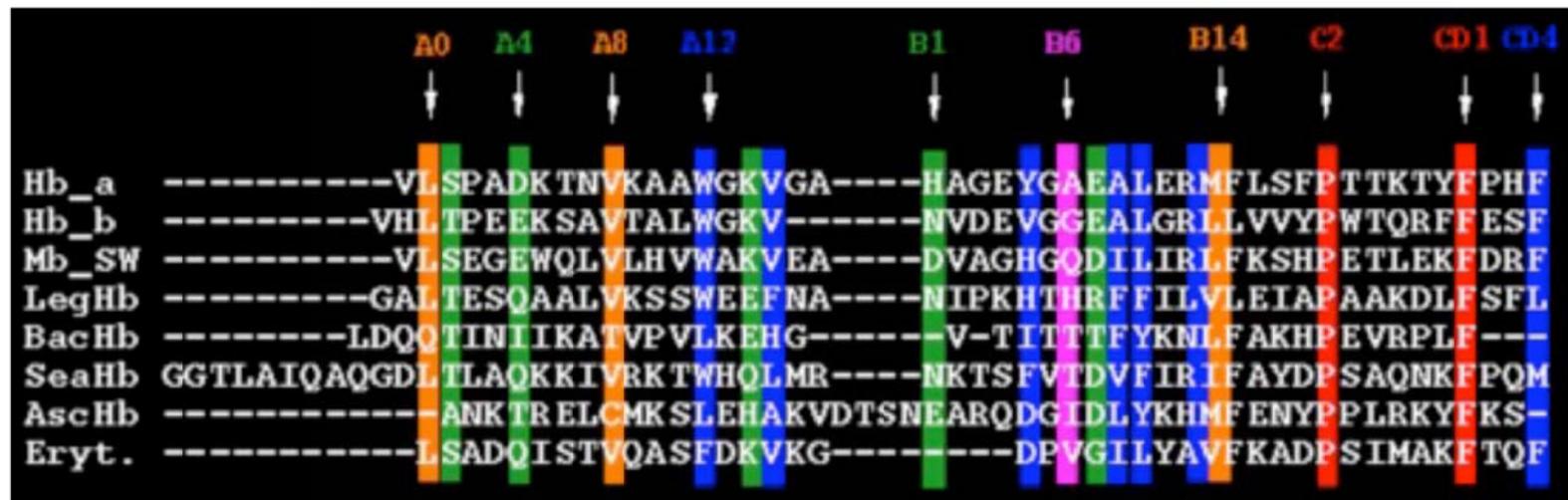
Recap: CYK Algorithm

$$table(i,1) = \{R \mid \exists_{(R \rightarrow w[i]) \in P}\}$$

$$table(i,j) = \bigcup_{x=1}^{x=j-1} \{R \mid \exists_{(R \rightarrow AB) \in P} : A \in table(i,x) \wedge B \in table(i+x, j-x)\}$$



- In 1984 R Doolittle et al. found similarities between a cancer-causing gene and a normal growth factor gene searching a database
- Today, finding sequence similarities with genes of known function is a common approach to infer the function of newly sequenced genes



The LCS Problem

Given: two strings $x[1\dots m]$ and $y[1\dots n]$

Goal: find a longest subsequence of characters (not necessarily continuous, but in correct order) common to both strings

Example: $x = A B C B D A B$

$y = B D C A B A$

$\rightarrow \text{LCS}(x, y) = B C B A$

- generate every subsequence of $x[1\dots m]$ and check if it is also a subsequence of $y[1\dots n]$
 - a single check will take up to n steps, this will give $O(n)$ for checking a subsequence
 - there are 2^m sub-sequences to check
 - Worst-case runtime in $O(n2^m)$

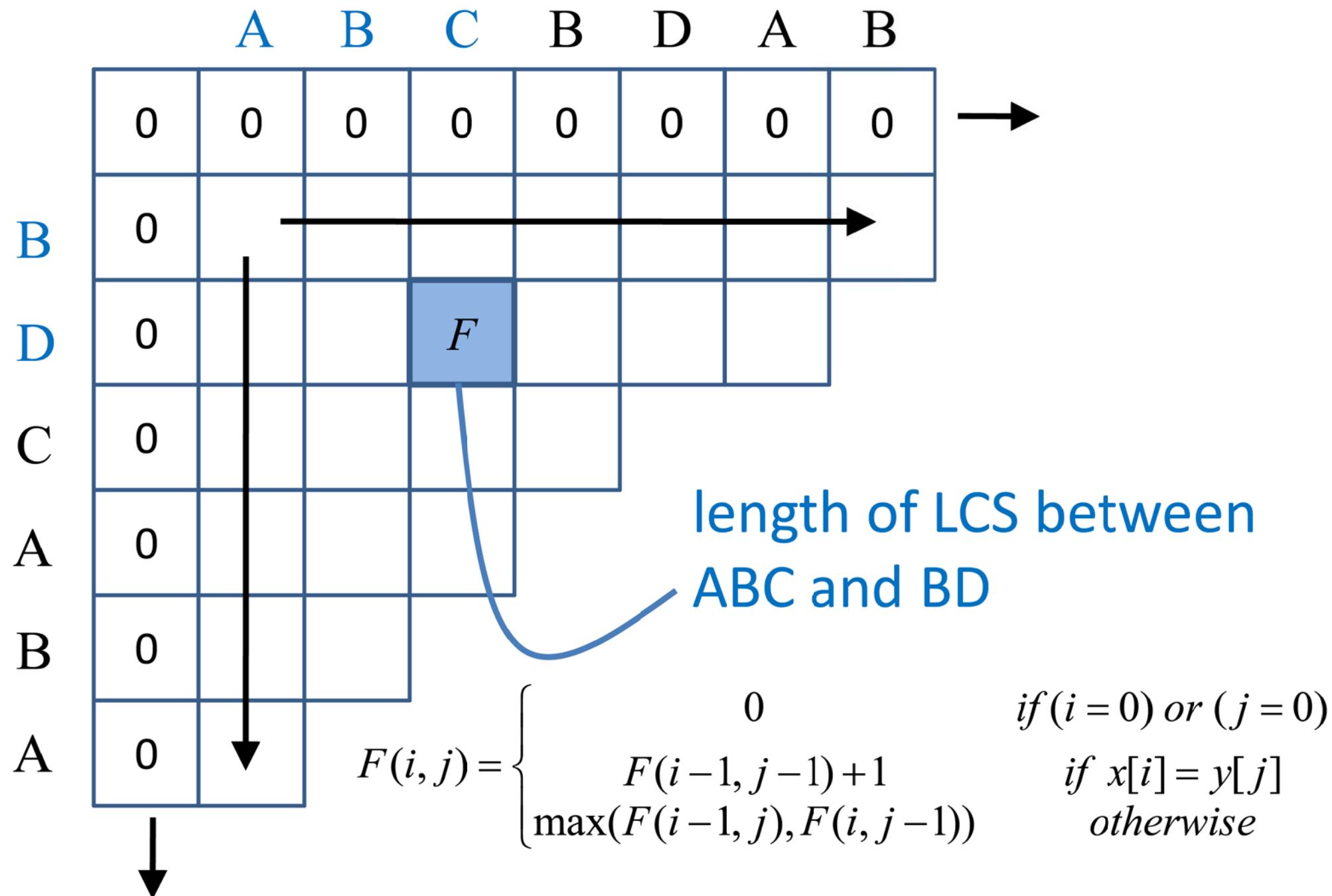


Bottom-up Approach

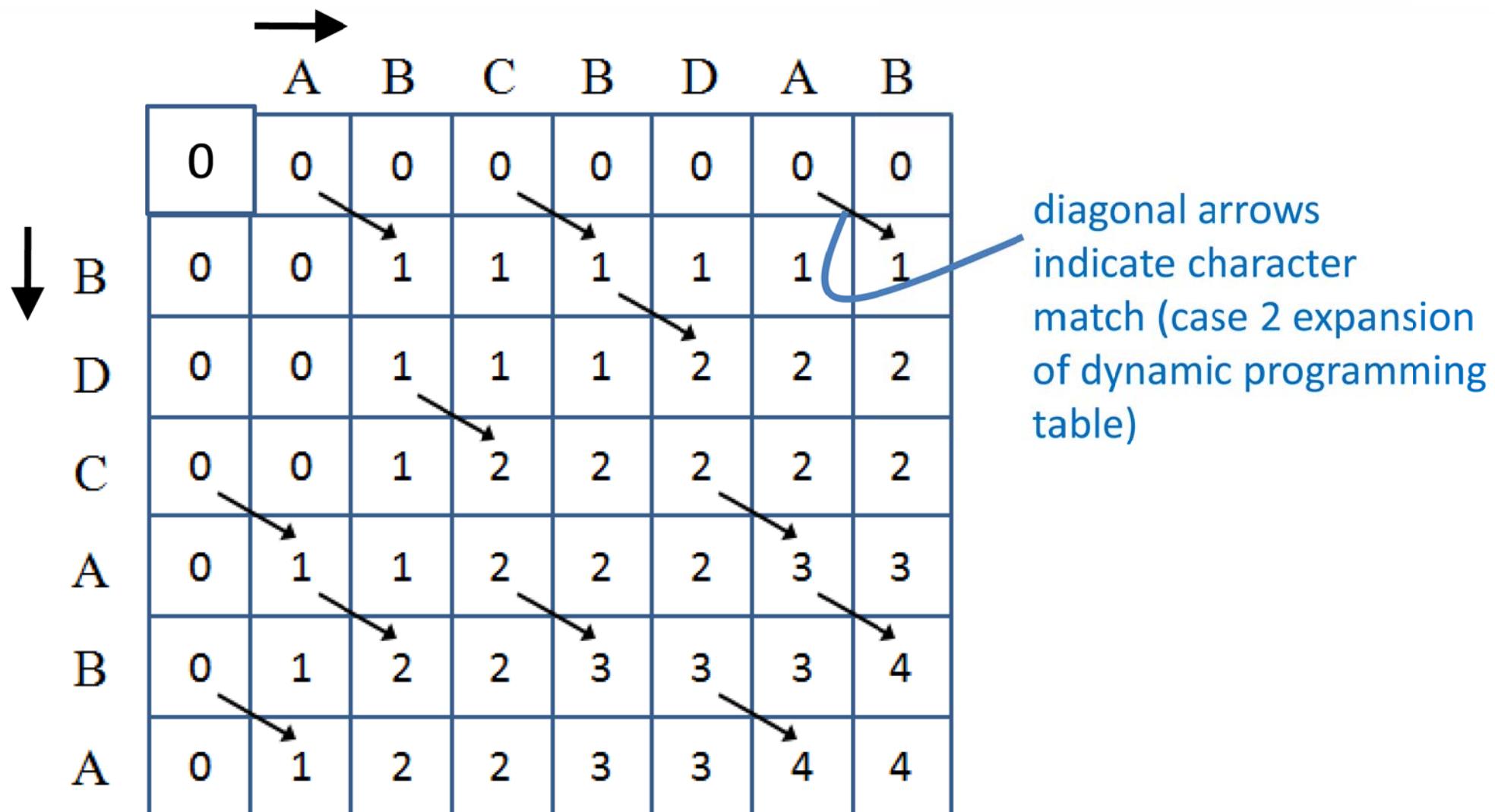


University of
BRISTOL

Department of
Computer Science



Computing Sub-solutions



	A	B	C	B	D	A	B
0	0	0	0	0	0	0	0
B	0	0	1	1	1	1	1
D	0	0	1	1	1	2	2
C	0	0	1	2	2	2	2
A	0	1	1	2	2	2	3
B	0	1	2	2	3	3	3
A	0	1	2	2	3	3	4

diagonal arrows indicate character match (case 2 expansion of dynamic programming table)

$$F(i, j) = \begin{cases} 0 & \text{if } (i = 0) \text{ or } (j = 0) \\ F(i-1, j-1) + 1 & \text{if } x[i] = y[j] \\ \max(F(i-1, j), F(i, j-1)) & \text{otherwise} \end{cases}$$

Tracing LCS Solutions

Matching alternative 1:

$$\text{LCS}_1(x,y)=\text{BCBA}$$

	A	B	C	B	D	A	B
A	0	0	0	0	0	0	0
B	0	0	1	1	1	1	1
C	0	0	1	1	1	2	2
D	0	0	1	1	2	2	2
C	0	0	1	2	2	2	2
A	0	1	1	2	2	2	3
B	0	1	2	2	3	3	4
A	0	1	2	2	3	3	4

Matching
alternative 2:
 $\text{LCS}_2(x,y)=\text{BDAB}$

Overall, LCS has
4 characters in
any case.

Sequence Alignment

	1.....10.....20.....30.....40.....50.....60.....70.....80.....90
Q5E940_BOVIN	-M P R E D R A T W K S N Y F L K I I Q L L D D Y P K C F I V G A D N V G S K O M O Q I R M S L R G K - A V V L M G K N T M M R K A I R G H L E N N -- P A L E 76
RLAO_HUMAN	-M P R E D R A T W K S N Y F L K I I Q L L D D Y P K C F I V G A D N V G S K O M O Q I R M S L R G K - A V V L M G K N T M M R K A I R G H L E N N -- P A L E 76
RLAO_MOUSE	-M P R E D R A T W K S N Y F L K I I Q L L D D Y P K C F I V G A D N V G S K O M O Q I R M S L R G K - A V V L M G K N T M M R K A I R G H L E N N -- P A L E 76
RLAO_RAT	-M P R E D R A T W K S N Y F L K I I Q L L D D Y P K C F I V G A D N V G S K O M O Q I R M S L R G K - A V V L M G K N T M M R K A I R G H L E N N -- P A L E 76
RLAO_CHICK	-M P R E D R A T W K S N Y F M K I I Q L L D D Y P K C F V V G A D N V G S K O M O Q I R M S L R G K - A V V L M G K N T M M R K A I R G H L E N N -- P A L E 76
RLAO_RANSY	-M P R E D R A T W K S N Y F L K I I Q L L D D Y P K C F I V G A D N V G S K O M O Q I R M S L R G K - A V V L M G K N T M M R K A I R G H L E N N -- S A L E 76
Q7ZUG3_BRARE	-M P R E D R A T W K S N Y F L K I I Q L L D D Y P K C F I V G A D N V G S K O M O T I R L S L R G K - A V V L M G K N T M M R K A I R G H L E N N -- P A L E 76
RLAO_ICTPU	-M P R E D R A T W K S N Y F L K I I Q L L N D Y P K C F I V G A D N V G S K O M O T I R L S L R G K - A V V L M G K N T M M R K A I R G H L E N N -- P A L E 76
RLAO_DROME	-M V R E N K A A W K A Q Y F I K V V E L F D E F P K C F I V G A D N V G S K O M O N I R T S L R G L - A V V L M G K N T M M R K A I R G H L E N N -- P Q L E 76
RLAO_DICDI	-M S G A G - S K R K K L F I E K A T K L F T T Y D K M I V A E A D F V G S S O L Q K I R K S I R G I - G A V L M G K K T M I R K V I R D L A D S K -- P E L D 75
Q54LP0_DICDI	-M S G A G - S K R K N V F I E K A T K L F T T Y D K M I V A E A D F V G S S O L Q K I R K S I R G I - G A V L M G K K T M I R K V I R D L A D S K -- P E L D 75
RLAO_PLAF8	-M A K L S K Q Q K K Q M Y I E K L S S I I Q Q Y S K I L I V H V D N V G S N Q M A S V R K S L R G K - A T I L M G K N T R I R T A L K K N I Q A V -- P Q I E 76
RLAO_SULAC	-M I G L A V T T T K K I A K W K V D E V A E L T E K L K T H K T I I I A N I E G F P A D K L H E I R K K L R G K - A D I K V T K N N L F N I A L K N A G -- Y D T K 79
RLAO_SULTO	-M R I M A V I T Q E R K I A K W K I E E V K E L E O Q K L R E Y H T I I I A N I E G F P A D K L H D I R K K M R G M - A E I K V T K N T L F G I A A K N A G -- L D V S 80
RLAO_SULSO	-M K R L A L A L K Q R K V A S W K L E E V K E L T E I L K N S N T I I L G N L E G F P A D K L H E I R K K L R G K - A T I K V T K N T L F G I A A K N A G -- I D I E 80
RLAO_AERPE	-M S V V S I L V Q Q M Y K R E K P I P E W K T L M L R E L E E L F S K H R V V L F A D L T G T P T F V V Q R V R K K L W K K - Y P M M V A K K R I I L R A M K A A G L E -- L D D N 86
RLAO_PYRAE	-M M L A I G K R R Y V R T R Q Y P A R K V K I V S E A T E L L Q K Y P Y V F L F D L H G L S S R I L H E Y R Y R L R R Y - G V I K I I K P T L F K I A F T K V Y G G -- I P A E 85
RLAO_METAC	-M A E E R H H T E H I P Q W K K D E I E N I K E L I Q S H K V F C M V G I E G I L A T K M O K I R R D L K D V - A V L K V S R N T L T E R A L N Q L G -- E T I P 78
RLAO_METMA	-M A E E R H H T E H I P Q W K K D E I E N I K E L I Q S H K V F G M V R I E G I L A T K I O K I R R D L K D V - A V L K V S R N T L T E R A L N Q L G -- E S I P 78
RLAO_ARCFU	-M A A V R G S -- P P E Y K V R A V E E I K R M I S S K P V V A I V S F R N V P A G Q M O K I R R E F R G K - A E I K V V K N T L L E R A L D A L G -- G D Y L 75
RLAO_METKA	-M A V K A K Q P P S G Y E P K V A E W K R R E V K E L K E L M D E Y E N V G L V D L E G I P A P O L O E I R A K L R E R D I I R M S R N T L M R I A L E E K L D E R -- P E L E 88
RLAO_METTH	-M A H V A E W K K E V Q E L H D L I K G Y E V V G I A N L A D I P A R O L O K M R Q T L R D S - A L I R M S K K T L I S L A L E K A G R E L -- E N V D 74
RLAO_METTL	-M I T A E S E H K I A P W K I E E V N K L K E L L K N G Q I V A L V D M M E V P A R O L Q E I R D K I R - G T M T L K M S R N T L I E R A I K E V A E E T G N P E F A 82
RLAO_METVA	-M I D A K S E H K I A P W K I E E V N A L K E L L K S A N V I A L I D M M E V P A V O L Q E I R D K I R - D Q M T L K M S R N T L I K R A V E E V A E E T G N P E F A 82
RLAO_METJA	-M E T K V K A H V A P W K I E E V K T L K G L I K S K P V V A I V D M M D V P A P O L Q E I R D K I R - D K V K L R M S R N T L I I R A L K E A A E E L N N P K L A 81
RLAO_PYRAB	-M A H V A E W K K E V E E L A N L I K S Y P V I A L V D V S S M P A Y P L S Q M R R L I R E N G G L L R V S R N T L I E L A I K K A A K E L G K P E L E 77
RLAO_PYRHO	-M A H V A E W K K E V E E L A K L I K S Y P V I A L V D V S S M P A Y P L S Q M R R L I R E N G G L L R V S R N T L I E L A I K K A A K E L G K P E L E 77
RLAO_PYRFU	-M A H V A E W K K E V E E L A N L I K S Y P V V A L V D V S S M P A Y P L S Q M R R L I R E N N G L L R V S R N T L I E L A I K K V A Q E L G K P E L E 77
RLAO_PYRKO	-M A H V A E W K K E V E E L A N L I K S Y P V I A L V D V A C V P A Y P L S K M R D K I R - G K A L L R V S R N T L I E L A I K R A A Q E L G Q P E L E 76
RLAO_HALMA	-M S A E S E R K T E T I P E W K Q E E V D A I V E M I E S Y E S V G V V N I A G I P S R Q L Q D M R R D L H G T - A E L R V S R N T L L E R A L D D V D -- D G L E 79
RLAO_HALVO	-M S E S E V R Q T E V I P Q W K R E E V D E L V D F I E S Y E S V G V V G V A G I P S R Q L Q S M R R E L H G S - A A V R M S R N T L V N R A L D E V N -- D G F E 79
RLAO_HALSA	-M S A E E Q R T T E E V P E W K R Q E V A E L V D L L E T Y D S V G V V N V T G I P S R Q L Q D M R R G L H G Q - A A L R M S R N T L L V R A L E E A G -- D G L D 79
RLAO_THEAC	-M K E V S Q Q K K E L V N E I T O R I K A S R S V A I V D V A G I R T R Q I Q D I R G K N R G K - I N L K V I K K T L L F K A L E N L G D -- E K L S 72
RLAO_THEVO	-M R K I N P K K K E I V S E L A Q D I T K S K A V A I V D I K G V R I T Q M Q D I R A K N R D K - V K I K V V K K T L L F K A L D S I N D -- E K L T 72
RLAO_PICTO	-M T E P A Q W K I D F V K N L E N E I N S R K V A A I V S I K G L R N N E F Q K I R N S I R D K - A R I K V S R A R L L R A I E N T G K -- N N I V 72

First 90 positions of a protein multiple sequence alignment of instances of the acidic ribosomal protein P0 (L10E) from several organisms.

Alignment Scoring

- Example Scoring: **MATCH = +10**
MISMATCH = -2
GAP = -5
- Example Alignment:

x =	C	T	G	T	C	G	C	T	G	C	
y =		T	G	C		C	G	T	G		
	-5	+10	+10	-2	-5	-2	-5	-5	+10	+10	-5

Total Score = 11

Task: find the highest scoring overall alignment

Idea: extend the concept of LCS by allowing for different scores for **matches** and **mismatches** as well as a **gap penalties**:

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + \text{match score}(x_i, y_j) \\ F(i-1, j) - \text{gap penalty} \\ F(i, j-1) - \text{gap penalty} \end{cases}$$

Recurrence Relations

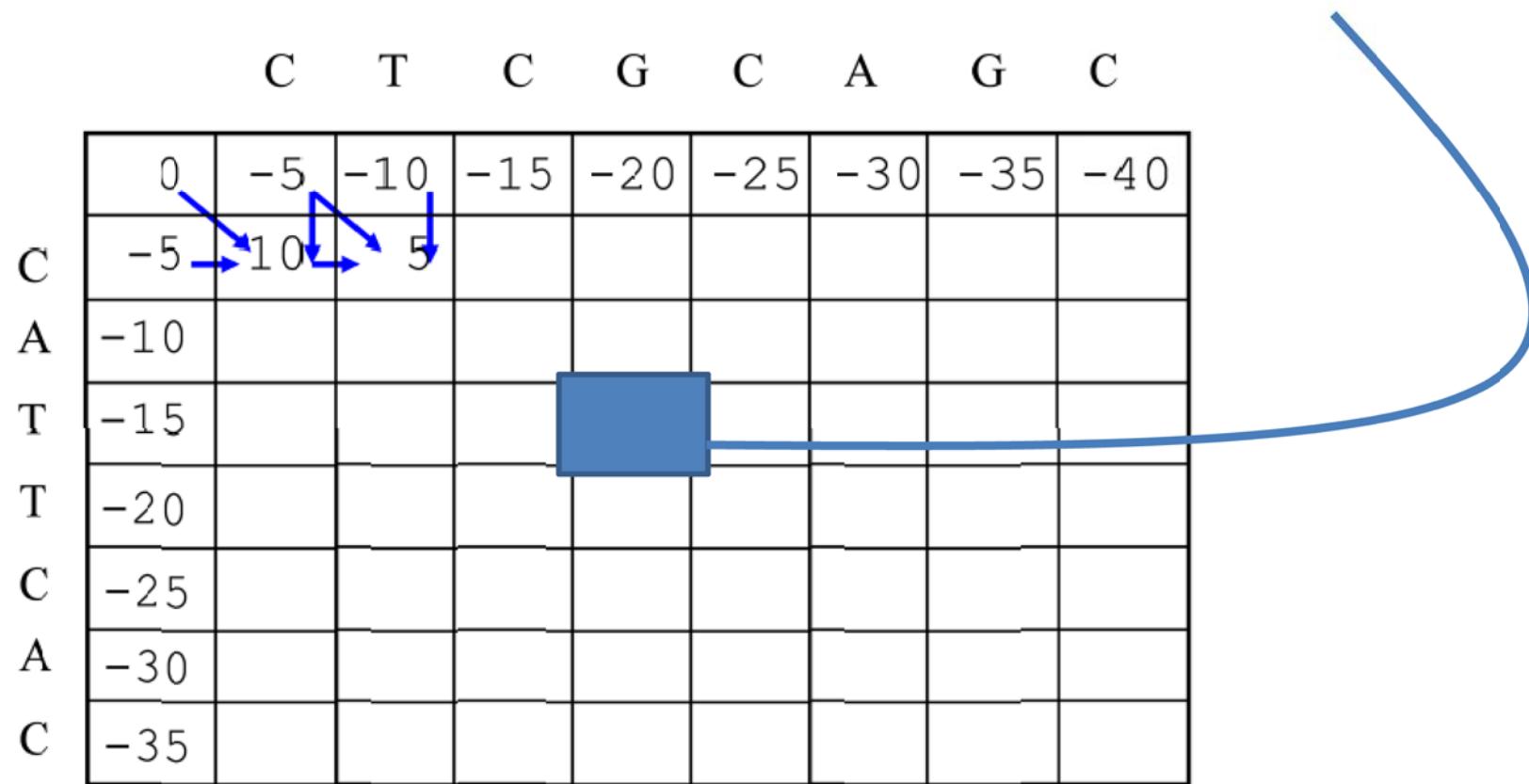
$$F(i, j) = \max \begin{cases} F(i - 1, j - 1) + \text{score}(x_i, y_j) \\ F(i - 1, j) + 0 \\ F(i, j - 1) + 0 \end{cases}$$



$$F(i, j) = \max \begin{cases} F(i - 1, j - 1) + \text{match score}(x_i, y_j) \\ F(i - 1, j) - \text{gap penalty} \\ F(i, j - 1) - \text{gap penalty} \end{cases}$$

Alignment Scoring Table

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + \text{match score}(x_i, y_j) \\ F(i-1, j) - \text{gap penalty} \\ F(i, j-1) - \text{gap penalty} \end{cases}$$



+10 for match, -2 for mismatch, -5 for space

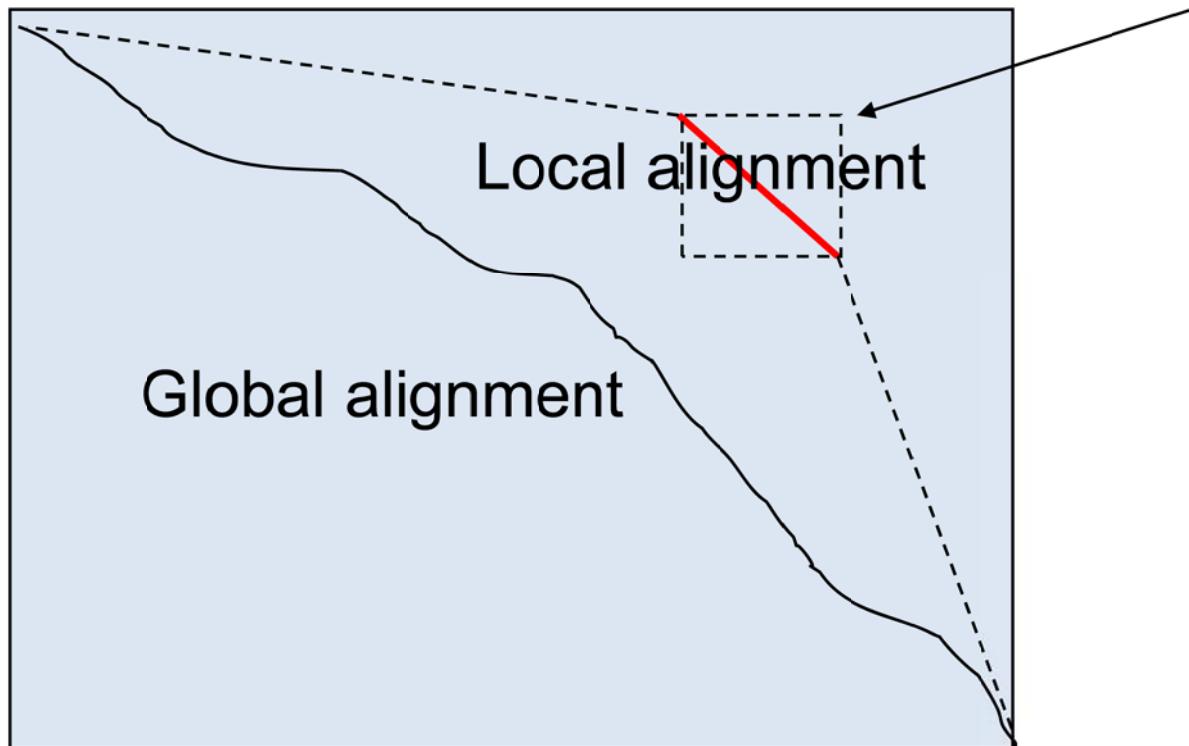
Alignment Scoring Example

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + \text{match score}(x_i, y_j) \\ F(i-1, j) - \text{gap penalty} \\ F(i, j-1) - \text{gap penalty} \end{cases}$$

λ	C	T	C	G	C	A	G	C	
C	0	-5	-10	-15	-20	-25	-30	-35	-40
A	-5	10	5	0	-5	-10	-15	-20	-25
T	-10	5	8	3	-2	-7	0	-5	-10
T	-15	0	15	10	5	0	-5	-2	-7
T	-20	-5	10*	13	8	3	-2	-7	-4
C	-25	-10	5	20	15	18	13	8	3
A	-30	-15	0	15	18	13	28	23	18
C	-35	-20	-5	10	13	28	23	26	33

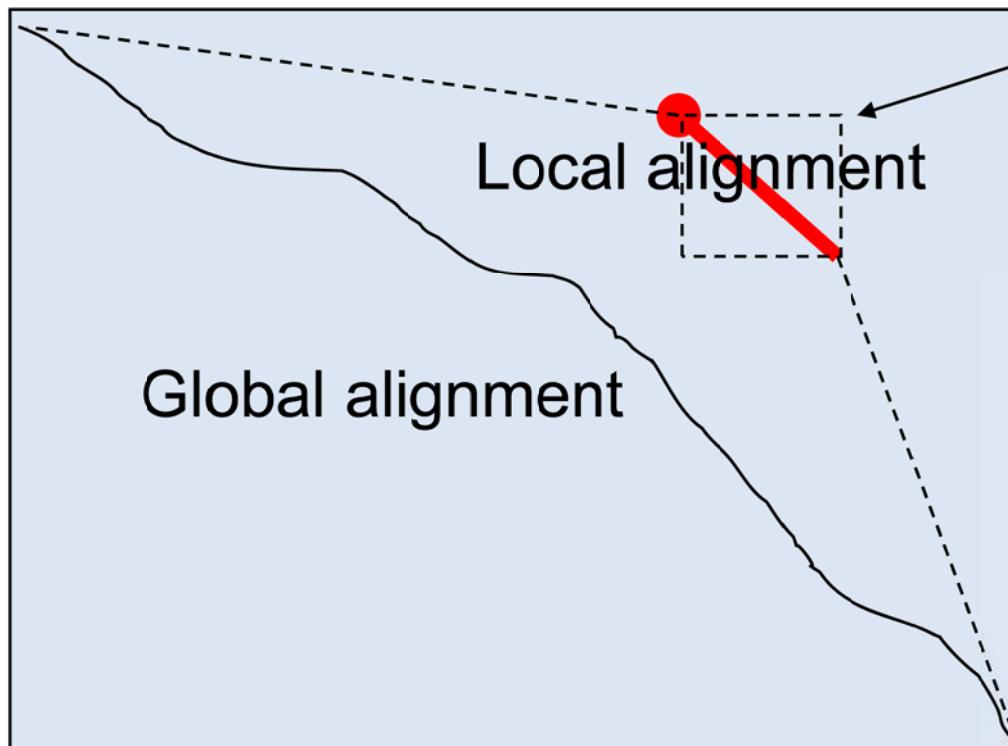
+10 for match, -2 for mismatch, -5 for space

Global vs Local Alignment



Computation of a
regional Global
Alignment to get
Local Alignment

Global vs Local Alignment

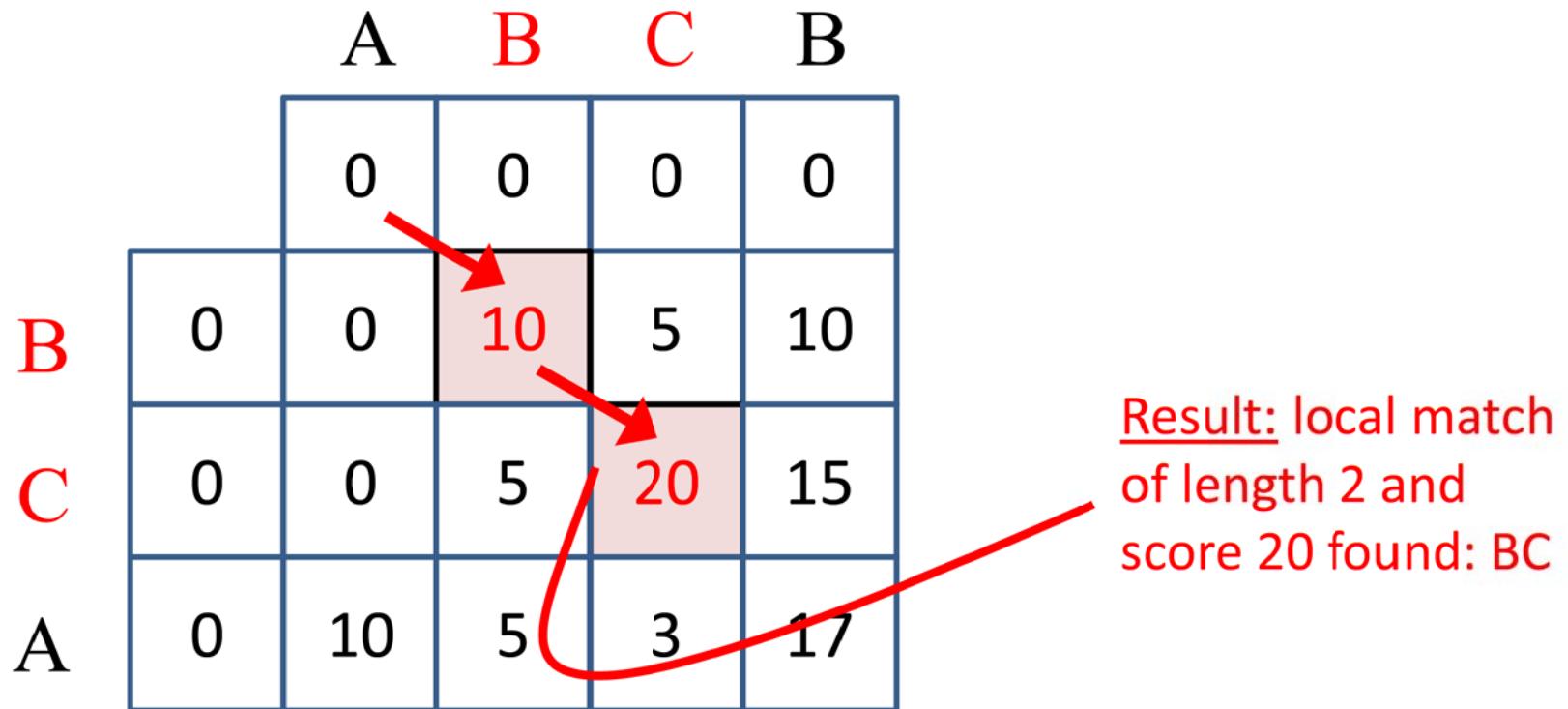


Idea: suppress
penalty from outside
window
→ limit reach of
penalties

Key Idea: we extend the algorithm by Needleman-Wunsch via enforcing a **positive matrix**:

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + \text{match score}(x_i, y_j) \\ F(i-1, j) - \text{gap penalty} \\ F(i, j-1) - \text{gap penalty} \\ 0 \end{cases}$$

Local Alignment Example



$$F(i, j) = \max \begin{cases} F(i-1, j-1) + \text{match score}(x_i, y_j) \\ F(i-1, j) - \text{gap penalty} \\ F(i, j-1) - \text{gap penalty} \\ 0 \end{cases}$$

+10 for match, -2 for mismatch, -5 for space

Step1: Determine Structure

Characterize the structure of an optimal solution by showing that it can be decomposed into optimal sub-problems.

Step2: Find Recurrence Relation

Recursively define the value of an optimal solution by expressing it in terms of optimal solutions for smaller problems.

Step 3: Bottom-up Computation

Compute the value of an optimal solution in a bottom-up fashion by using a table structure.

Step 4: Construction of Optimal Solution

Construct an optimal solution from computed information.

Algorithm Design Paradigms

DIVIDE & CONQUER

Break down a problem into independent sub-problems of related type, solve them separately and combine the solutions.

GREEDY APPROACH

Use a sequence of locally optimal decisions incrementally to build up a solution.

DYNAMIC PROGRAMMING

Break down a problem into overlapping sub-problems of related type, build up solutions from larger and larger sub-solutions.