遗传信息的分析与研究

测序数据序列比对操作

纲要

01 生物序列比对算法

1.1 序列比对简介

序列比对(Alignment)

- 序列比对是指，将两个或多个序列排列在一起，以确定字符或子字符串之间的对应关系，由此判断序列之间的相似性和一致性。
- 该方法在脱氧核糖核酸（DNA）、核糖核酸（RNA）和蛋白质序列分析当中有广泛应用。

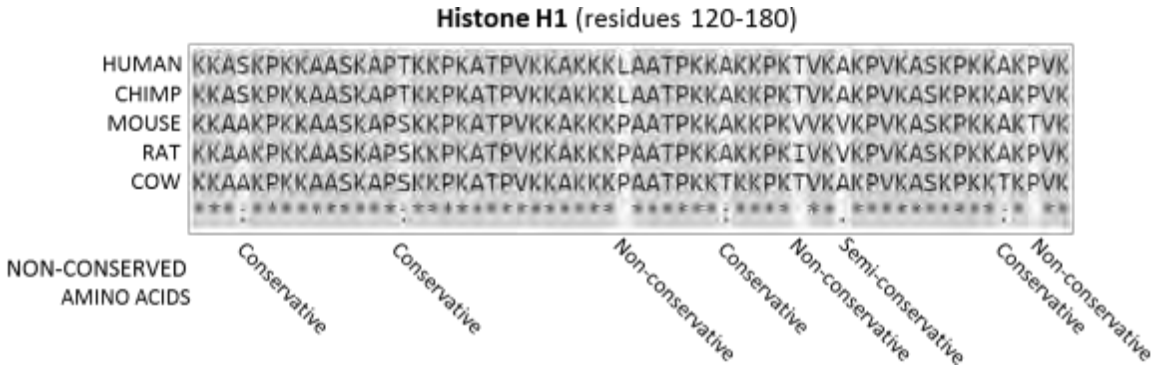
A	A	C	T	T	T	T	G	T
C	A	G	T	T	T	A	G	G

	Score	Expect	Identities	Gaps
Target 1	76.8 bits (41)	2e-19	49/53 (92%)	0/53 (0%)

```
Query  1   GTTGATAACAGCAAGATGGCTTTGAACTCAGAAGCCTTATCAGTTGTGAGTGA  53
           ||||||||||||||||||||||||||||||||||||||||| |||||||| | ||||||||||
Sbjct  68  GTTGATAACAGCAAGATGGCTTTGAACTCAAAAGCCTTAGCCCTTGTGAGTGA  120
```

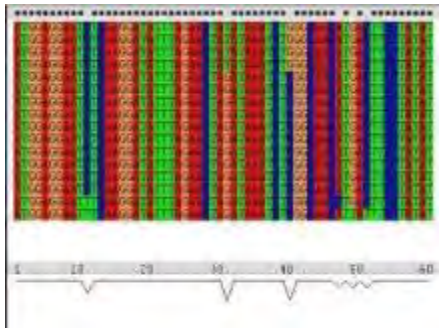
	Score	Expect	Identities	Gaps
Target 3	75.0 bits (40)	8e-19	45/47 (96%)	2/47 (4%)

```
Query  14  CGCCTGGAGCGCGGCAGGAAGCCTTATCAGTTGTGAGTGAGGACCAG  60
           ||||||||  |||||||||  ||||||||||||||||||||||||||||
Sbjct  13  CGCCTGGA-CGCGGCAGG-AGCCTTATCAGTTGTGAGTGAGGACCAG  57
```

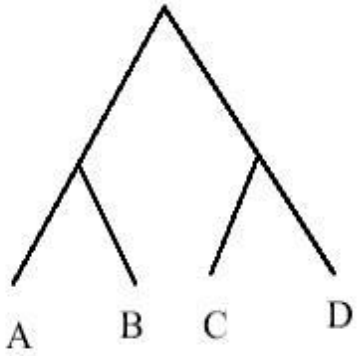


1.1 序列比对简介

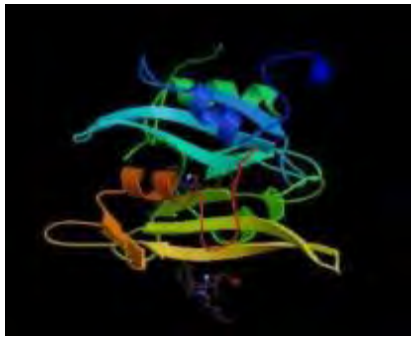
- 序列比对的根本任务是：
  - ●发现序列之间的相似性(Similarity)
  - ●辨别序列之间的差异



- 序列比对的目的是：
  - ●相似序列 → 相似的结构，相似的功能
  - ●判别序列之间的同源性(Homology)
  - ●推测序列之间的进化关系



- 应用方向为：
  - ●寻找新基因
  - ●系统发育学与进化分析
  - ●蛋白结构模建
  - ●功能预测等



- 序列的相似性(Similarity) V.S. 同源性(Homology)

●同源性(Homology)：当两个基因或蛋白在进化上来自同一个祖先，则具有同源性

●同源性与序列相似性紧密相关，一般相似性越高则是同源序列的可能性越高，因此可通过相似性来推断序列是否同源

●通常，DNA序列有超过30%或400bp的序列相似性，或蛋白质序列有超过125个氨基酸相似，可初步认为是同源序列

- 全局序列比对(Global alignment)
  - 对两条或多条完整的序列进行对位排列
  - 如Needleman-Wunsch 算法(1970年)
  - 主要优点是适合较短序列或结构预测
- 局部序列比对(Local alignment)
  - 找出最大相似的子序列
  - 多用于两两成对比较
  - 如Smith-Waterman算法(1981年)
  - 主要优点是适合数据库查询或寻找结构域

1.1 序列比对简介

- 以字符串比较为例：
  - 给定两个字符串，设置一套评分方式，从而评价如何匹配两句话，能够获得最佳评分

全局序列比对(Global alignment)

```
          - C- CC
--T--CC-C-AGT--TATGT-CAGGGGACACG-A-GCATGCAGA-GAC
  |  || |  ||  | | | |||    || | | |  | ||||   |
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG-T-CAGAT--C
```

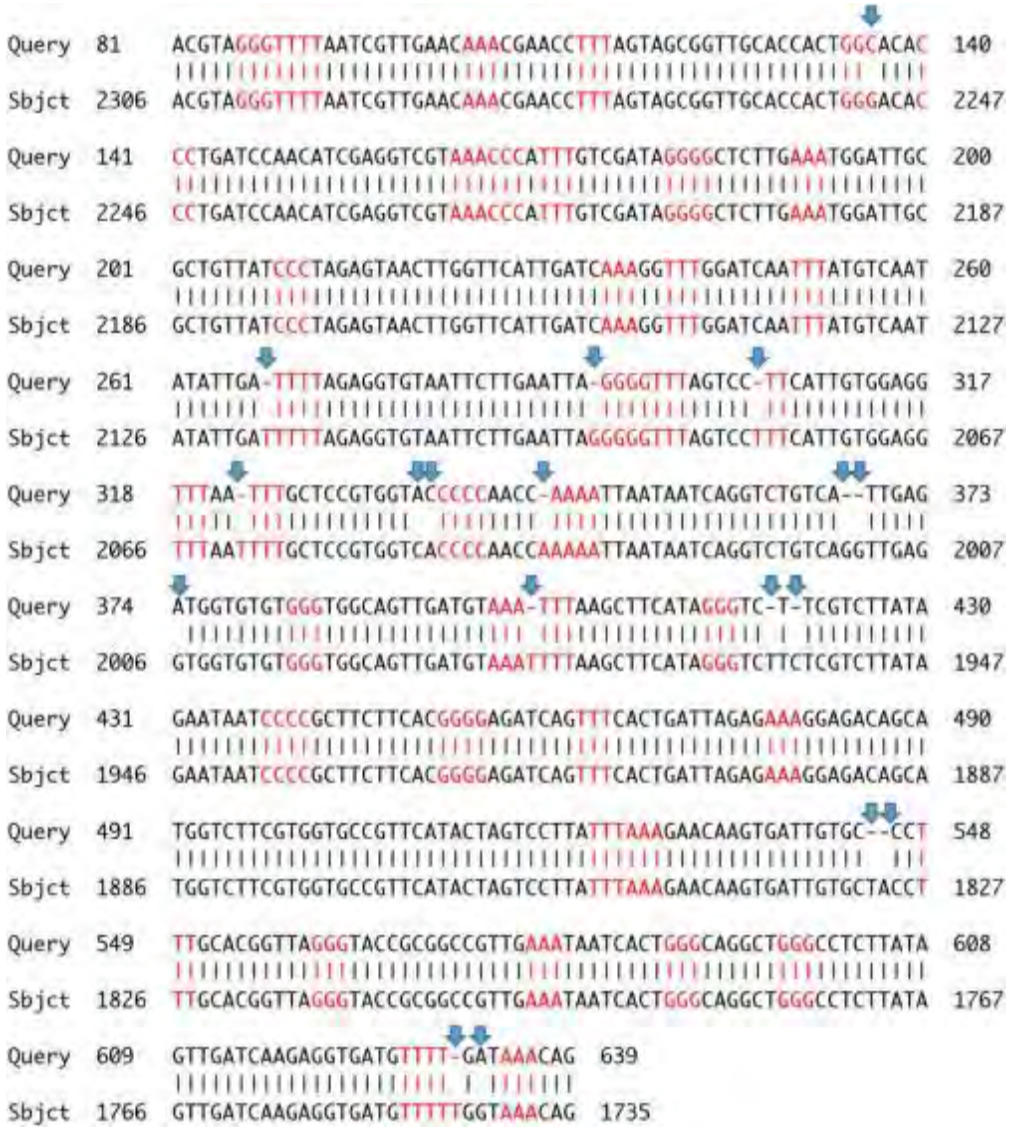
局部序列比对(Local alignment)

```
                    tccCAGTTATGTCAGgggacacgagcatgcagagac
                       ||||||||||||
aattgccgccgtcgttttcagCAGTTATGTCAGatc
```

1.1 序列比对简介

- 长序列比对，如何寻找最佳匹配情况?

  - 例：如抑癌基因p53,全长超过25kb，比对情况复杂，如何进行比对评价?



1.2 生物序列格式

DNA核酸标准编码(IUPAC codes)

DNA:

Nucleotide Code:	Base:
A	Adenine
C	Cytosine
G	Guanine
T (or U)	Thymine (or Uracil)
R	A or G
Y	C or T
S	G or C
W	A or T
K	G or T
M	A or C
B	C or G or T
D	A or G or T
H	A or C or T
V	A or C or G
N	any base
. or -	gap

➢蛋白质氨基酸标准编码(IUPAC codes)

Protein:

Amino Acid Code:	Three letter Code:	Amino Acid:
A	Ala	Alanine
B	Asx	Aspartic acid or Asparagine
C	Cys	Cysteine
D	Asp	Aspartic Acid
E	Glu	Glutamic Acid
F	Phe	Phenylalanine
G	Gly	Glycine
H	His	Histidine
I	Ile	Isoleucine
K	Lys	Lysine
L	Leu	Leucine
M	Met	Methionine
N	Asn	Asparagine
P	Pro	Proline
Q	Gln	Glutamine
R	Arg	Arginine
S	Ser	Serine
T	Thr	Threonine
V	Val	Valine
W	Trp	Tryptophan
X	Xaa	Any amino acid
Y	Tyr	Tyrosine
Z	Glx	Glutamine or Glutamic acid

1.2 生物序列格式

FASTA序列格式

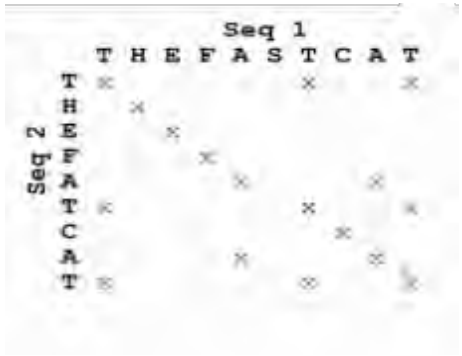
- 文本文件
- 首行开头为标题，以“>”符号开头
- 序列使用IUPAC编码

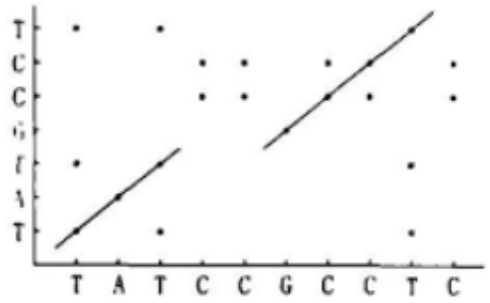
```
>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIGTNLV
EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFLG
LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAGX
IENY
```

GenBank	gi|*gi-number*|gb|*accession*|*locus*
EMBL Data Library	gi|*gi-number*|emb|*accession*|*locus*
DDBJ, DNA Database of Japan	gi|*gi-number*|dbj|*accession*|*locus*
NBRF PIR	pir||*entry*
Protein Research Foundation	prf||*name*
SWISS-PROT	sp|*accession*|*name*
Brookhaven Protein Data Bank (1)	pdb|*entry*|*chain*
Brookhaven Protein Data Bank (2)	*entry*:*chain*|PDBID|CHAIN|SEQUENCE
Patents	pat|country|number
GenInfo Backbone Id	bbs|number
General database identifier	gnl|database|identifier
NCBI Reference Sequence	ref|accession|locus
Local Sequence identifier	lcl|identifier

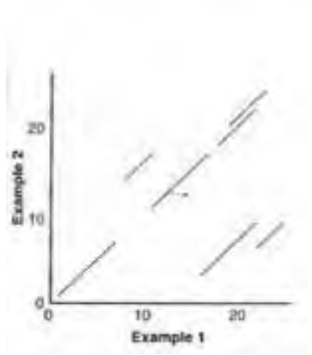
1.3 序列比对打分矩阵

- 传统方法：点阵图





TAT--GCCT
||| ||||
TATCCGCCTC



如何设计算法，快速、精准地完成长序列、多序列等复杂比对计算?

- 可能的位点关系
  - 匹配（|）
  - 替换/错配（*）
  - 空位（插入/缺失 -）

```
Query:  ACAGCTTACGCGAAAACCAAGCAGGGAGTTTGGGAAACCCAACA-T-AGTCGACCCC
        ||||||||||*|||||**|||||**|||||***|||||||-|-||||||||||
Sject:  ACAGCTTACGCCAAAACCCTGCAGGGCTTTTGGGTTTCCCAACAGTAAGTCGACCCC
```

1.3 序列比对打分矩阵

➢核酸打分矩阵(Scoring Matrix)

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

Unitary Matrix
等价矩阵

	A	C	T	G
A	1	-5	-5	-1
C	-5	1	-1	-5
T	-5	-1	1	-5
G	-1	-5	-5	1

Transition-transversion Matrix
转移矩阵(转换-颠换矩阵)

	A	C	T	G
A	5	-4	-4	-4
C	-4	5	-4	-4
T	-4	-4	5	-4
G	-4	-4	-4	5

BLAST Matrix
BLAST矩阵

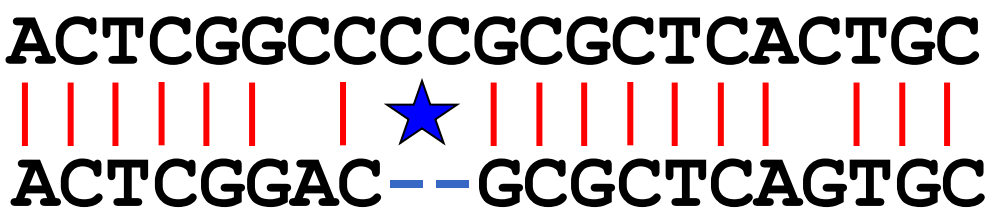
1. 等价矩阵：相同核苷酸之间的匹配得分为1，不同核苷酸之间的替换得分为0。这种打分方式的优点是简单，但由于没有考虑到实际的统计规律或生物演化规律，实际应用较少；

2. 转换-颠换矩阵：DNA序列由4种基本的脱氧核糖核苷酸单分子组成，核苷酸分子的核心结构差异是4种含氮碱基，分别是腺嘌呤（Adenine，简称A），鸟嘌呤（Guanine，简称G），胸腺嘧啶（Thymine，简称T），胞嘧啶（Cytosine，简称C）。在数据分析过程中，我们往往用碱基序列指代DNA序列。在生物演化过程中，4种碱基发生相互替换的概率并不均等，嘌呤容易被嘌呤替换（A G），嘧啶容易被嘧啶替换（C T），这样的替换被称为转换；嘌呤与嘧啶之间发生替换，则被称为颠换。转换比颠换更容易发生，为体现这样的演化规律，在对序列比对进行打分时，通常对转换赋分为-1，对颠换赋分为-5.

3. BLAST矩阵：在实际使用过程中，根据统计经验，将碱基完全相同的情况赋分为5，比对不上的情况赋分为-4，计分矩阵计算所得结果较好，因此将此方法广泛应用于DNA序列比对。

1.3 序列比对打分矩阵

- 插入Gap思路
  - 当出现错配时，插入gap进行调整

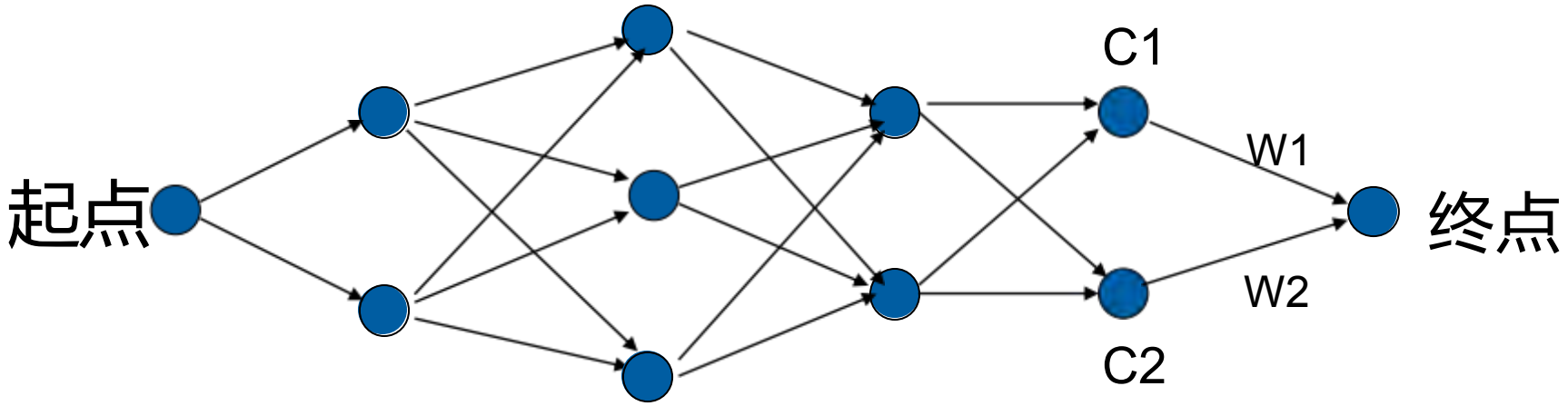


  - Gap需要罚分(penalties):
    - 插入首个gap (GAP OPENING) : 高罚分 (例如： -2)
    - 插入后续gap (GAP EXTENSION): 低罚分 (例如：每个gap -1)
  - 插入多个Gap将会造成分值过低

1.4 序列比对算法

核心问题：动态规划算法（ Dynamic Programming ）

- 寻找最长公共子序列(LCS，Longest Common Subsequence)
- 两条序列的最长公共子序列，就是指这两条序列共有的子序列，且长度最长。



路径1：C1 + W1 ？

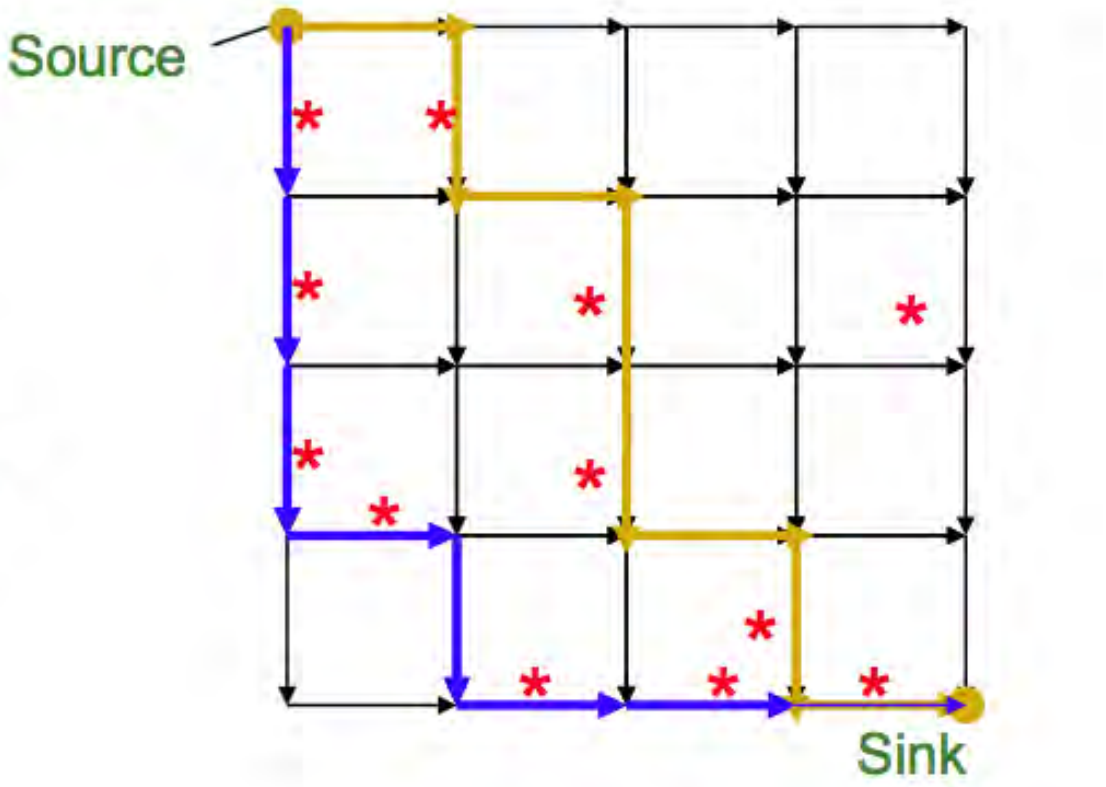
路径2：C2 + W2 ？

取路径最小值

- 思路：大问题分解为小问题，从起点到终点逐层计算最短路径，最终拼装解决大问题。
- 具体到序列上就是先把大序列从头看成小片段，在优化了小片段的结果之后，再逐步得出整个比对结果。

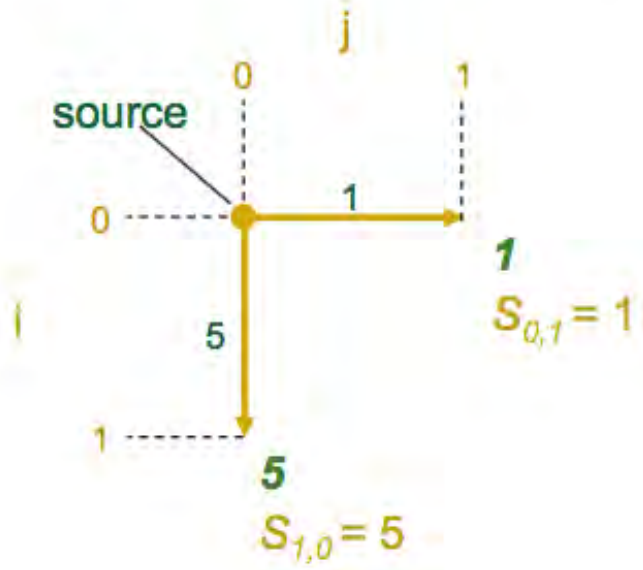
The Manhattan Tourist Problem

Imagine seeking a path (from source to sink) to travel (only eastward and southward) with the most number of attractions (*) in the Manhattan grid



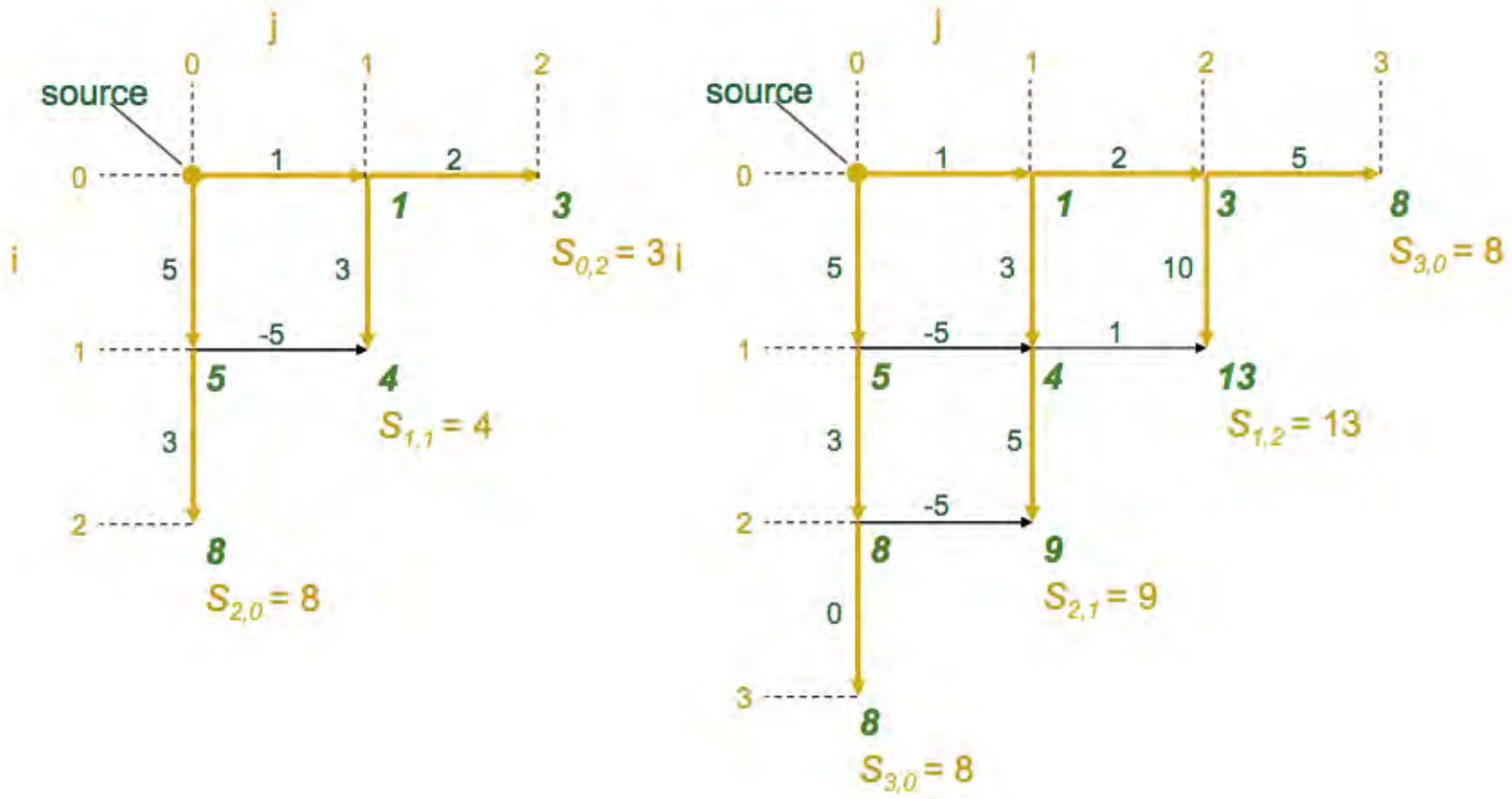
The Manhattan Tourist Problem

- Dynamic programming: identify the distinct subproblems and solve them in the right order.

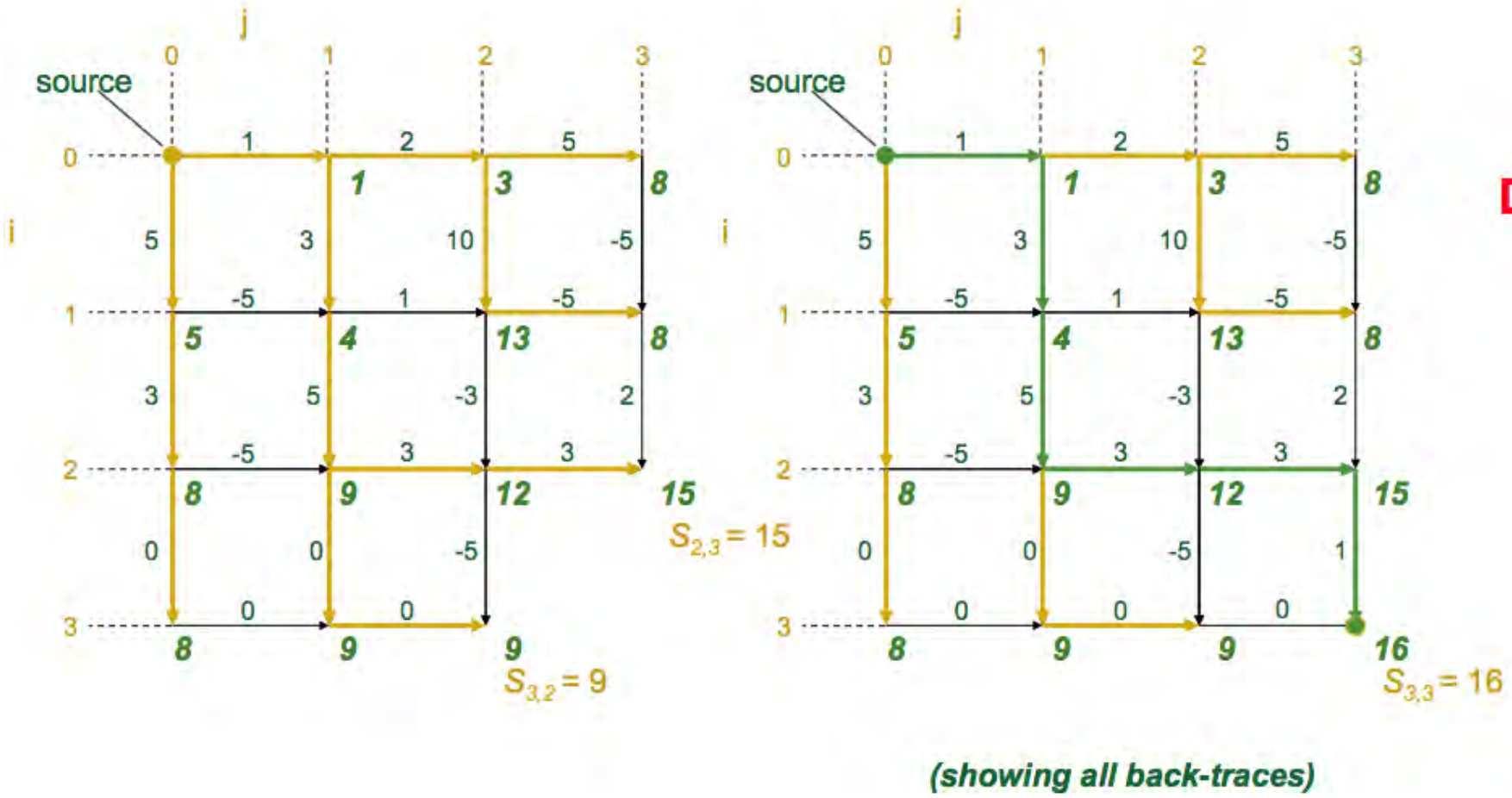


Each node's score is the maximum of the prior node score plus the weight of the respective edge in between

The Manhattan Tourist Problem



The Manhattan Tourist Problem



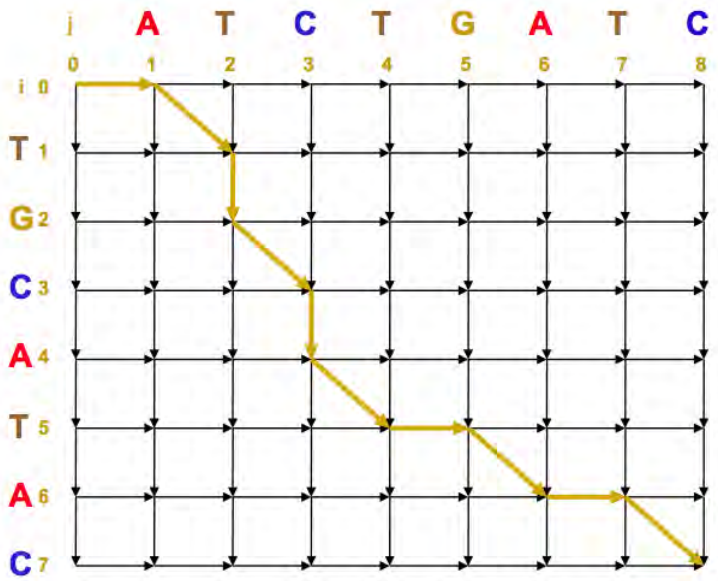
DNA 序列比对

序列1：ATCTGATC

m = 8

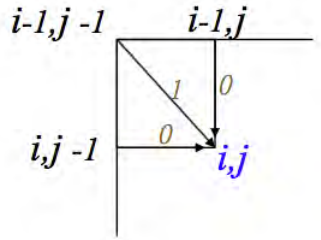
序列2：TGCATAC

n = 7



The length of LCS(v_i,w_j) is computed by:

$$s_{i,j} = \max \begin{cases} s_{i-1,j} \\ s_{i,j-1} \\ s_{i-1,j-1} + 1 \text{ if } v_i = w_j \end{cases}$$



LCS Problem: *Find a path with maximum number of diagonal edges*

i coords:	0	1	2	2	3	3	4	5	6	7	8
v		A	T	--	C	--	T	G	A	T	C
w		--	T	G	C	A	T	--	A	--	C
j coords:	0	0	1	2	3	4	5	5	6	6	7

(0,0)→(1,0)→(2,1)→(2,2)→(3,3)→(3,4)→(4,5)→(5,5)→(6,6)→(7,6)→(8,7)

DNA 序列比对

The Longest Common Subsequence (LCS) problem—the simplest form of sequence alignment – allows only insertions and deletions (no mismatches).

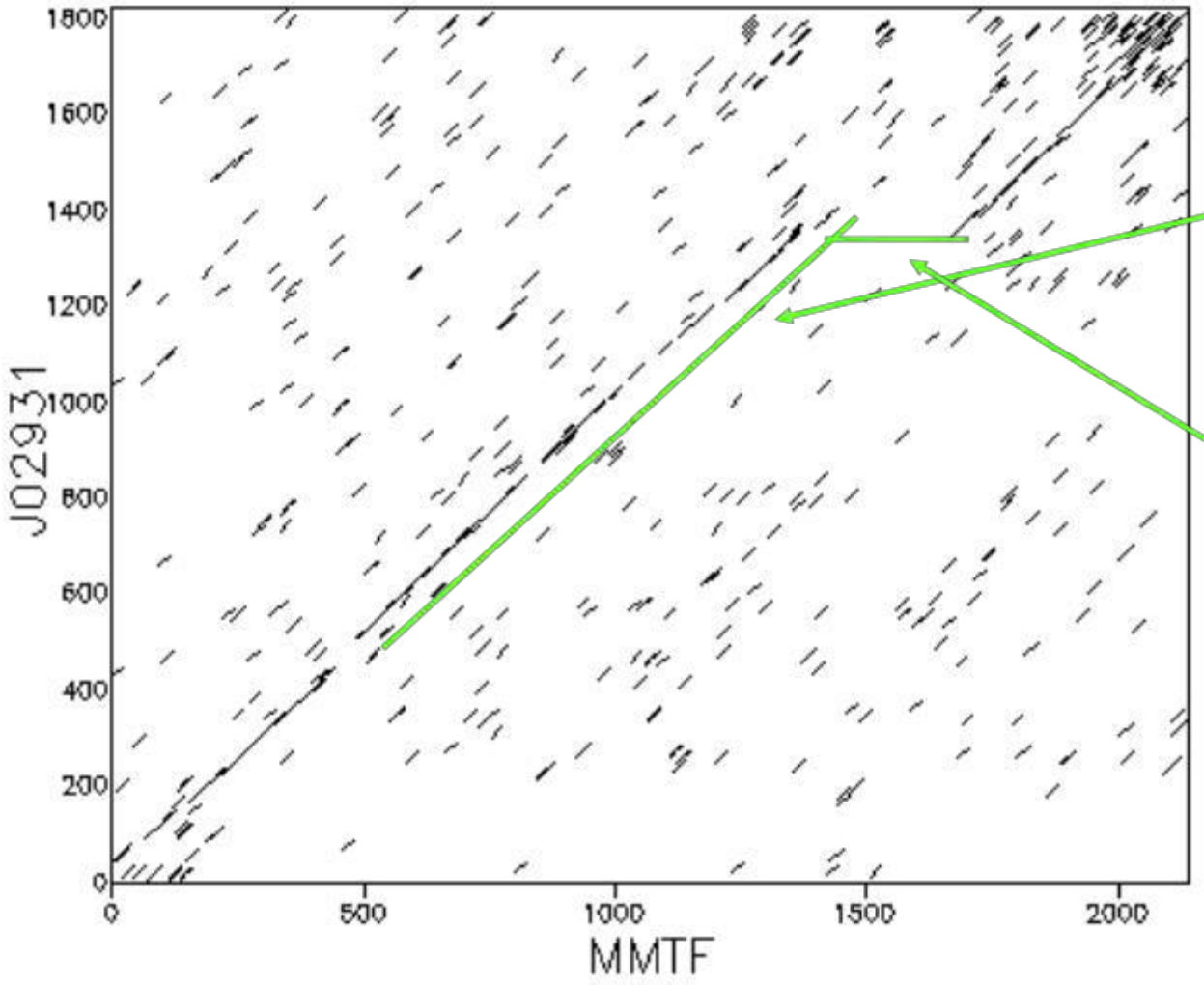
In the LCS Problem, we scored 1 for matches and 0 for indels

- 替换积分矩阵
- Gap罚分

	A	G	C	T	-
A	10	-1	-3	-4	
G	-1	7	-5	-3	
C	-3	-5	9	0	-5
T	-4	-3	0	8	
-	-5				

替换记分矩阵

- 点阵图法序列比对可视化(Dot Plot)：



对角线

两条序列相似区域出现对角线

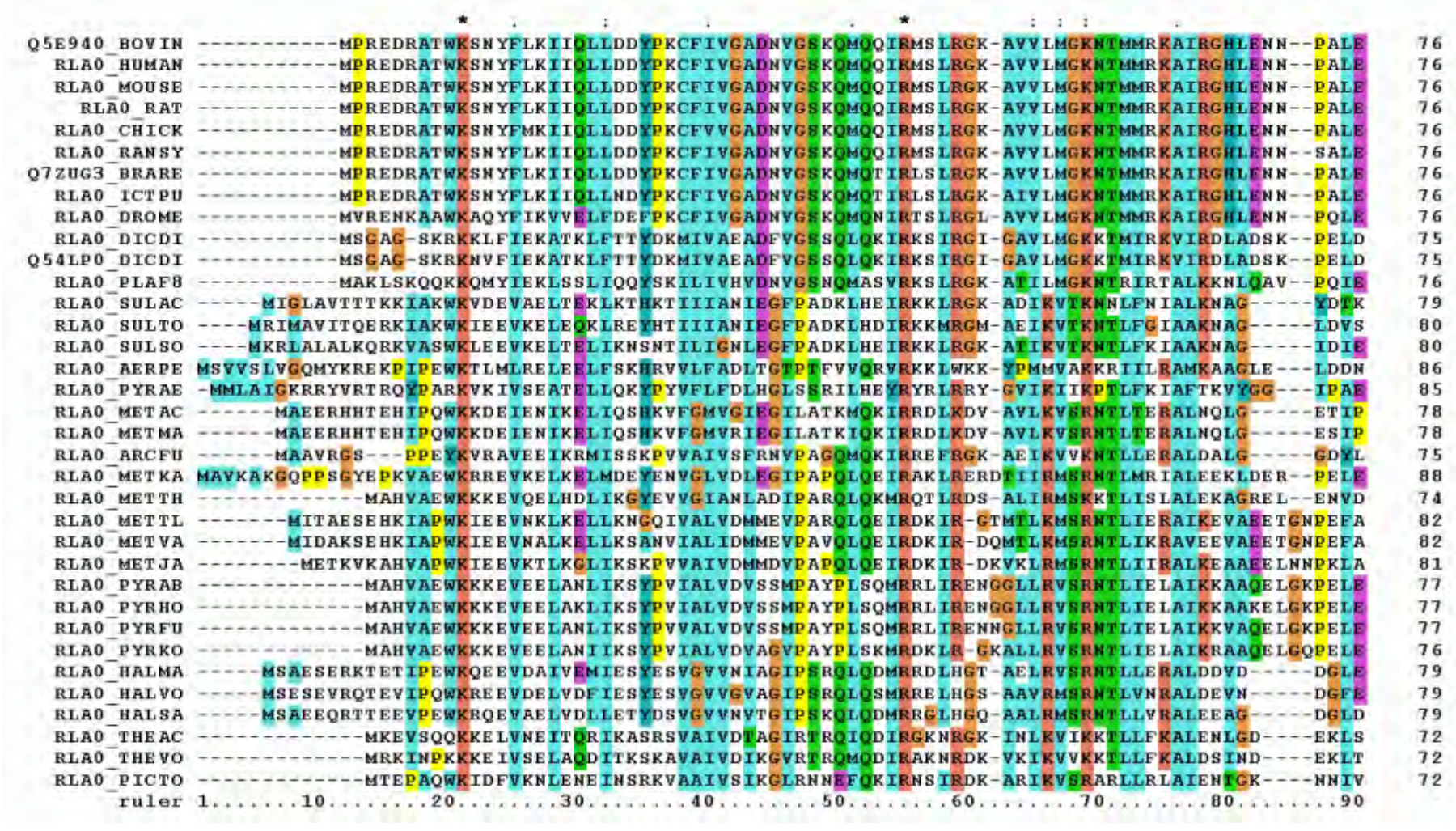
Break

当一条序列相对于另一条序列存在插入或缺失时点阵图中出现break (断裂)

1.5 序列比对结果

典型的序列比对结果

星号* (asterisk) 保守区，单一氨基酸； **冒号**: (colon) 半保守区(PAM250 similar properties - scoring > = 0.5).
句号. (period) 低保守区(PAM250 similar properties - scoring < 0.5). **无符号**为非保守区。



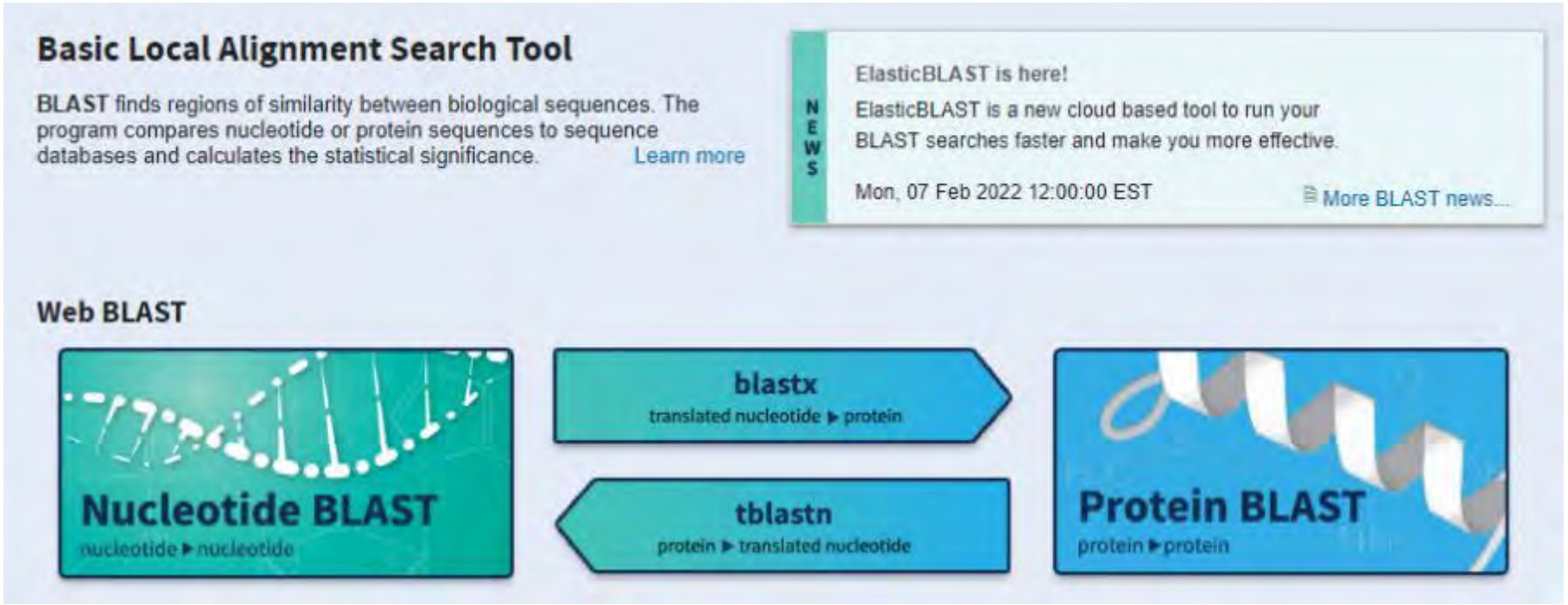
02

NCBI-BLAST的使用

2.1 BLAST概述

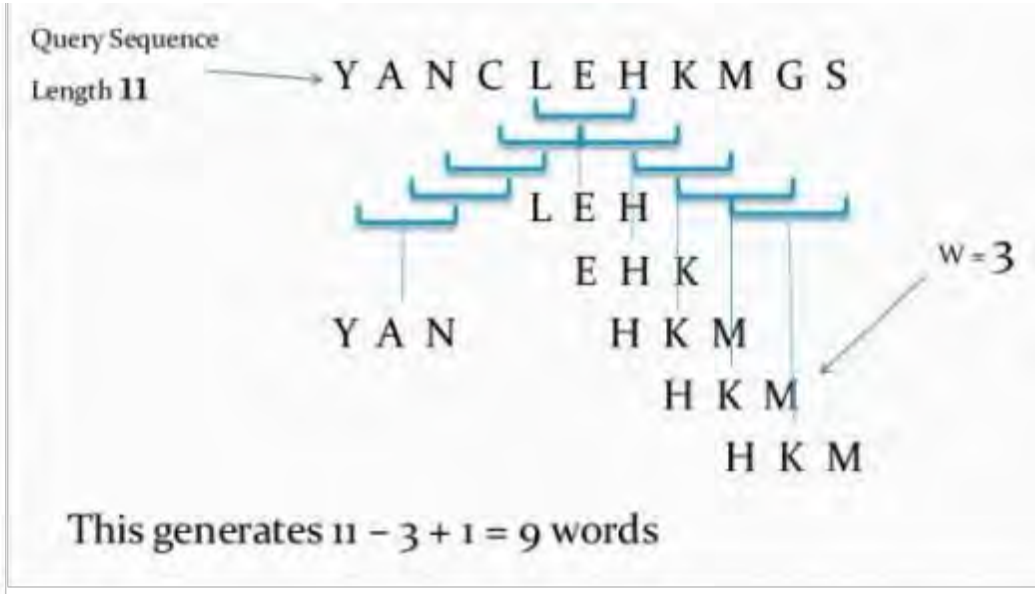
Basic Local Alignment Search Tool

- 1990年由美国国立生物技术信息中心（NCBI）开发的，一个**基于序列相似性的数据库搜索程序。**
- https://blast.ncbi.nlm.nih.gov/Blast.cgi



2.1 BLAST概述

- BLAST 算法流程



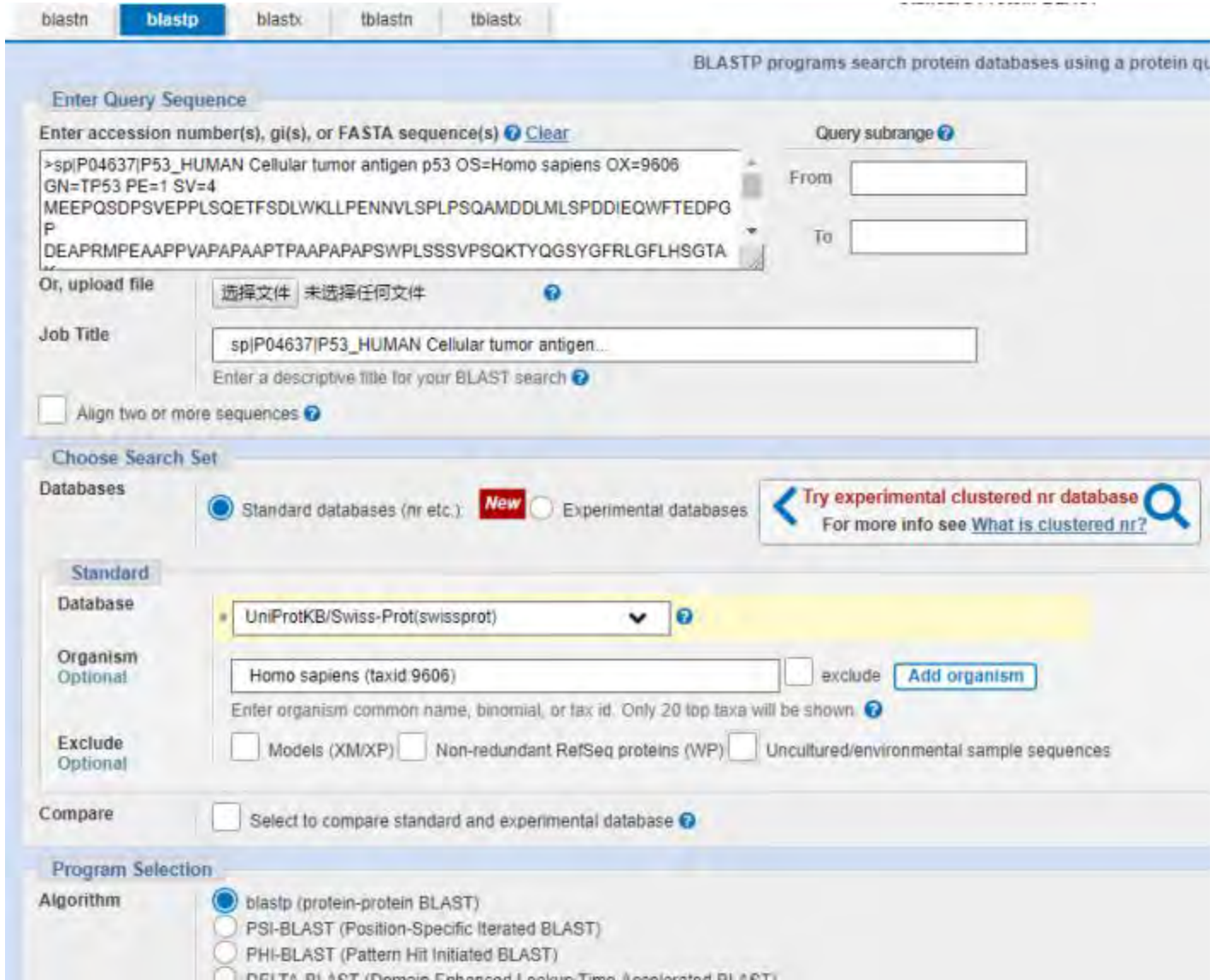
2.1 BLAST概述

- 主要的BLAST程序包

程序名	查询序列	数据库	搜索方法
Blastn	核酸	核酸	核酸序列搜索逐一核酸数据库中的序列
Blastp	蛋白质	蛋白质	蛋白质序列搜索逐一蛋白质数据库中的序列
Blastx	核酸	蛋白质	核酸序列翻译成蛋白质序列后和蛋白质数据库中的序列逐一搜索。
tblastn	蛋白质	核酸	蛋白质序列和核酸数据库中的核酸序列翻译后的蛋白质序列逐一比对。
tblastx	核酸	核酸	核酸序列翻译成蛋白质序列，再和核酸数据库中的核酸序列框翻译成的蛋白质序列逐一进行比对。

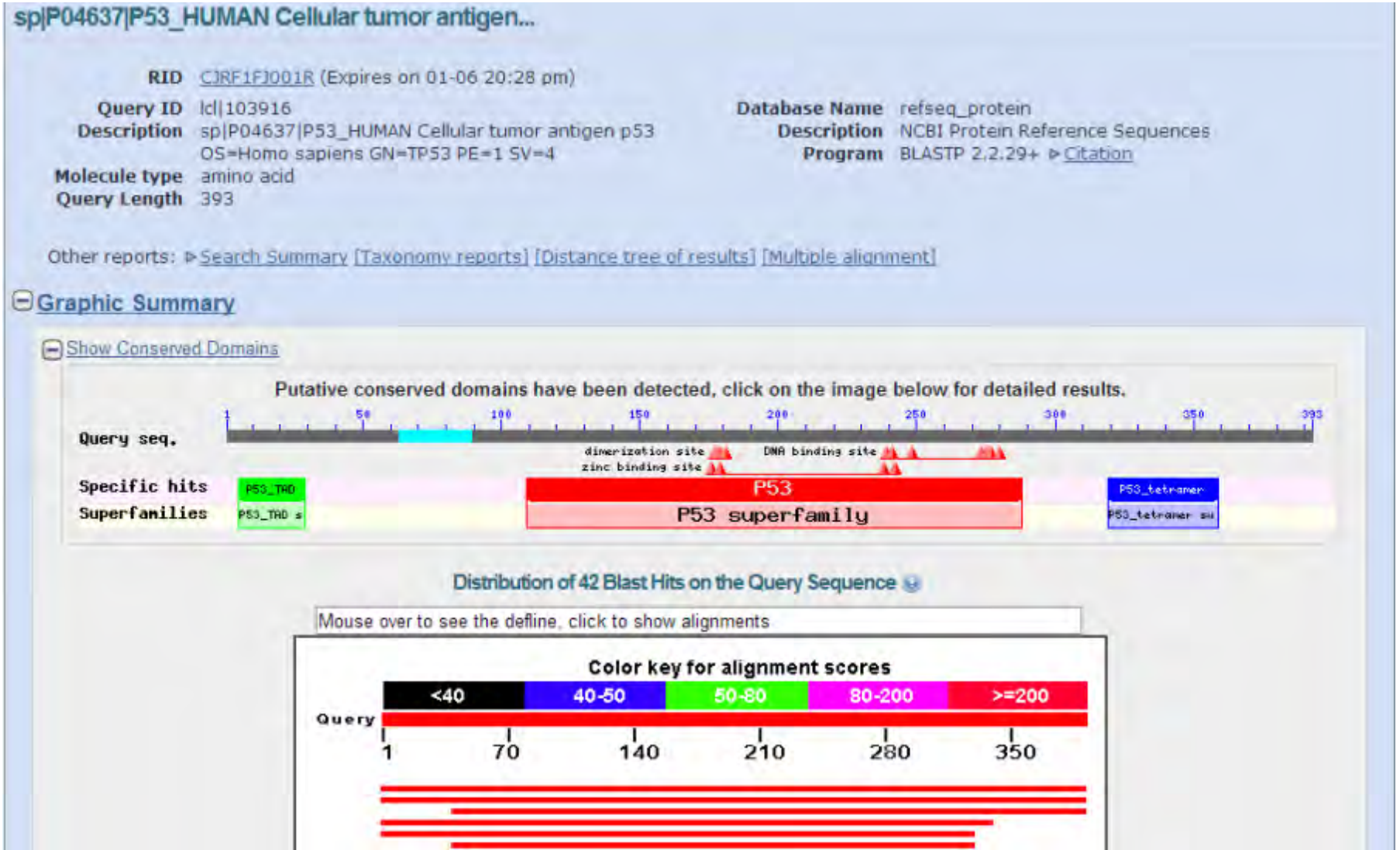
2.2 BLAST操作实例

- BLASTP实例：人类TP53蛋白



2.2 BLAST操作实例

- BLAST结果：任务信息与图形界面



2.2 BLAST操作实例

- BLAST结果：描述信息

Descriptions

Sequences producing significant alignments:

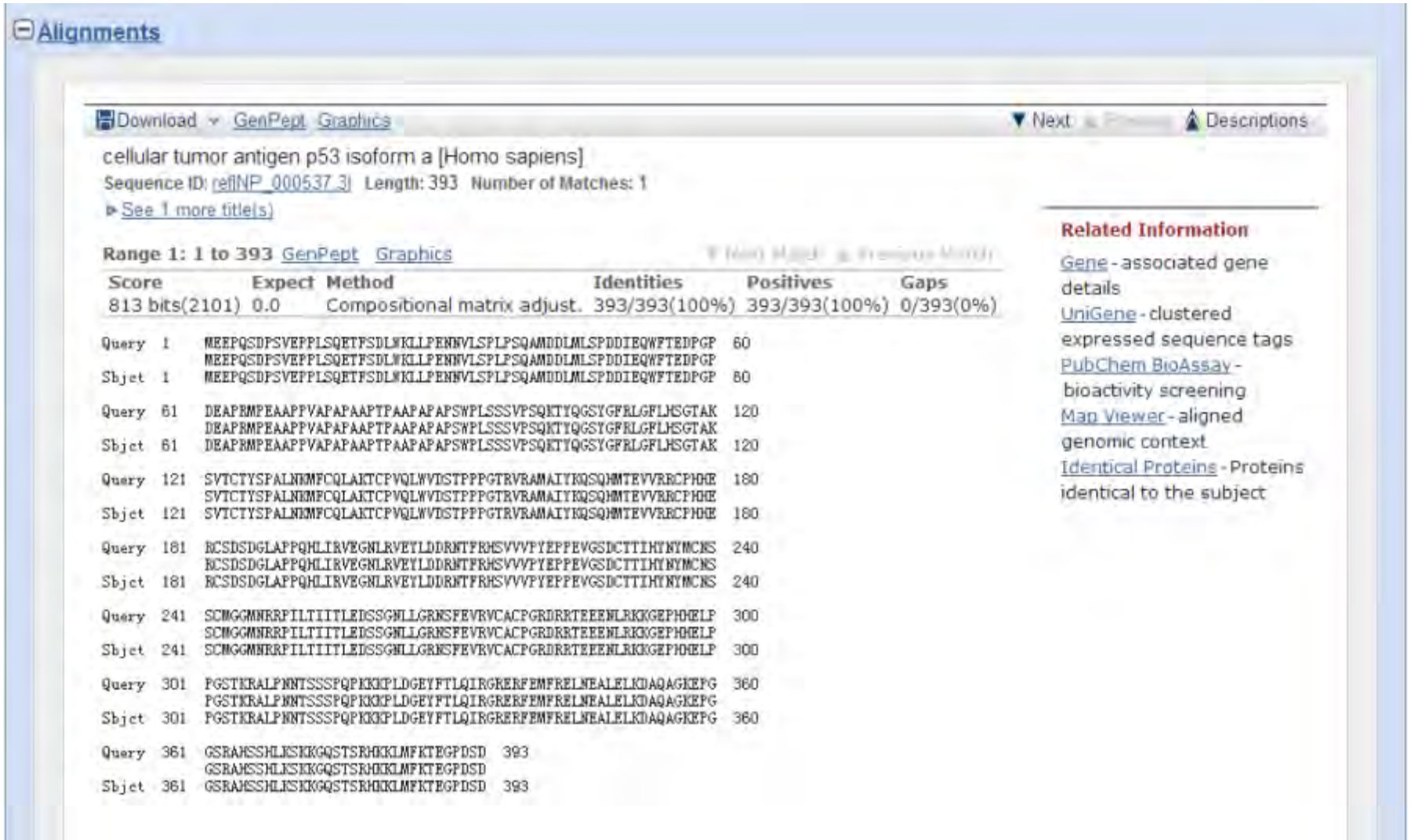
Select: All None Selected:0

Alignments Download GenPept Graphics Distance tree of results Multiple alignment

	Description	Max score	Total score	Query cover	E value	Ident	Accession
☐	cellular tumor antigen p53 isoform a [Homo sapiens] >ref|NP_001119584.1| cellular tumor anti	813	813	100%	0.0	100%	NP_000537.3
☐	PREDICTED: cellular tumor antigen p53 isoform X1 [Homo sapiens]	763	763	100%	0.0	95%	XP_005256835.1
☐	cellular tumor antigen p53 isoform g [Homo sapiens] >ref|NP_001263689.1| cellular tumor anti	737	737	90%	0.0	100%	NP_001119590.1
☐	cellular tumor antigen p53 isoform c [Homo sapiens]	689	689	86%	0.0	98%	NP_001119585.1
☐	cellular tumor antigen p53 isoform b [Homo sapiens]	688	688	84%	0.0	100%	NP_001119586.1
☐	cellular tumor antigen p53 isoform i [Homo sapiens]	612	612	74%	0.0	100%	NP_001263625.1
☐	cellular tumor antigen p53 isoform h [Homo sapiens]	612	612	76%	0.0	97%	NP_001263624.1
☐	cellular tumor antigen p53 isoform d [Homo sapiens]	549	549	66%	0.0	100%	NP_001119587.1
☐	cellular tumor antigen p53 isoform j [Homo sapiens]	491	491	59%	2e-174	100%	NP_001263626.1
☐	cellular tumor antigen p53 isoform e [Homo sapiens]	422	422	50%	8e-148	100%	NP_001119588.1
☐	cellular tumor antigen p53 isoform f [Homo sapiens]	422	422	53%	1e-147	96%	NP_001119589.1
☐	cellular tumor antigen p53 isoform k [Homo sapiens]	365	365	43%	1e-125	100%	NP_001263627.1
☐	cellular tumor antigen p53 isoform l [Homo sapiens]	364	364	46%	2e-125	96%	NP_001263628.1
☐	tumor protein p73 isoform e [Homo sapiens]	273	273	74%	2e-87	48%	NP_001191118.1
☐	tumor protein p73 isoform i [Homo sapiens]	275	275	74%	3e-87	48%	NP_001191115.1
☐	tumor protein p73 isoform d [Homo sapiens]	273	273	74%	2e-86	48%	NP_001119714.1

2.2 BLAST操作实例

- BLAST结果：比对信息



03

测序数据的序列比对

3.1 短序列比对软件BWA

BWA： Burrows-Wheeler Aligner

Burrows-**W**heeler **A**ligner

Bwa软件主页：http://bio-bwa.sourceforge.net/

linux环境下，进行下列四步操作就可以使用了。

①下载软件安装包:

wget http://jaist.dl.sourceforge.net/project/bio-bwa/bwa-0.7.12.tar.bz2

②解压缩：tar jxf bwa-0.7.12.tar.bz2

③进入解压缩目录：cd bwa-0.7.12

④编译安装：make

使用时写明全路径/home/teacher/wcz/bin/bwa-0.7.12/bwa

或者设置环境变量（export PATH="/home/yourname/bin/bwa-0.7.12:$PATH"，也可写入~/.bashrc并source）后仅使用程序名称。

3.1 短序列比对软件BWA

BWA： Burrows-Wheeler Aligner

首先要建立参考序列索引
bwa index -a bwtsw ref.fa

BWA中有三种比对算法: `mem', `bwasw', and `aln/samse/sampe'.
If you are not sure which to use, try `bwa mem' first.

①mem算法
bwa mem ref.fa se.fq.gz > aln-se.sam
②bwasw算法
bwa bwasw ref.fa long_read.fq > aln.sam
③aln/samse/sampe算法
先找坐标
bwa aln ref.fa SEread.fq > SEread.sai
再输出为sam格式
bwa samse -f single.sam reference.fa single.sai single.fastq

3.2 SAM文件分析汇总工具

SAMTOOLS：处理SAM文件

Samtools软件主页
http://samtools.sourceforge.net/
SAM 格式规范:
http://samtools.github.io/hts-specs/SAMv1.pdf

samtools 输入bam文件，导出sam文件。同时可以进行排序，合并，建立索引等功能，并支持从特定区域内查找reads。可以对比对结果进行分类统计。
samtools 可以应用于linux的命令管道里。以“-”作为标准输入或输出。

view 从bam/sam文件中提取/打印部分比对结果。默认为所有的区域，也可以染色体区域（1-based，须sort并index）。
mpileup 列举每条reads比对的indel，SNP等信息。

3.2 SAM文件分析汇总工具

SAMTOOLS用法示例

转换sam为bam：

```
samtools view -bS aln.sam > aln.bam
```

排序：

```
samtools sort aln.bam aln.sorted #得到aln.sorted.bam
```

索引：

```
samtools index aln.sorted.bam
```

找SNP：

```
samtools mpileup -f ref.fa aln.bam > rawSNP.xls
```

统计比对记录数

```
samtools view -c aln.bam
```

只抽取mapped reads

```
samtools view -c -F 4 aln.bam
```

只抽取unmapped reads

```
samtools view -c -f 4 aln.bam
```

简单统计

```
samtools flagstat aln.bam
```

3.3 测序数据序列比对

1. 建立索引：
$ /Pipeline/FIS.Traits/tools/bwa-mem2 index ./00_ref/MGI358.SNP.fa

2. 序列比对生成SAM
$ /Pipeline/FIS.Traits/tools/bwa-mem2 mem -M -Y -t 1 ./00_ref/MGI358.SNP.fa ./01_clean/test_clean.fq.gz > ./02_align/test.clean.sam

3. 节省存储空间生成BAM
$ samtools sort ./02_align/test.clean.sam -o ./02_align/test.clean.sort.bam

4. 为建立的bam文件建立索引
$ samtools index ./02_align/test.clean.sort.bam

$ samtools view -F 256 -hb ./02_align/test.clean.sort.bam > ./02_align/test.clean.sort.uniq.bam

3.3 测序数据序列比对

5. 查看生成BAM文件
$ samtools view -S ./02_align/test.clean.sort.bam|less -S

6. 统计覆盖深度
$ /Pipeline/FIS.Traits/tools/bamdst -p ./00_ref/target.358.SE50.subSNP.bed -o ./02_align ./02_align/test.clean.sort.uniq.bam

7. 查看覆盖深度
$ less ./02_align/depth.tsv.gz