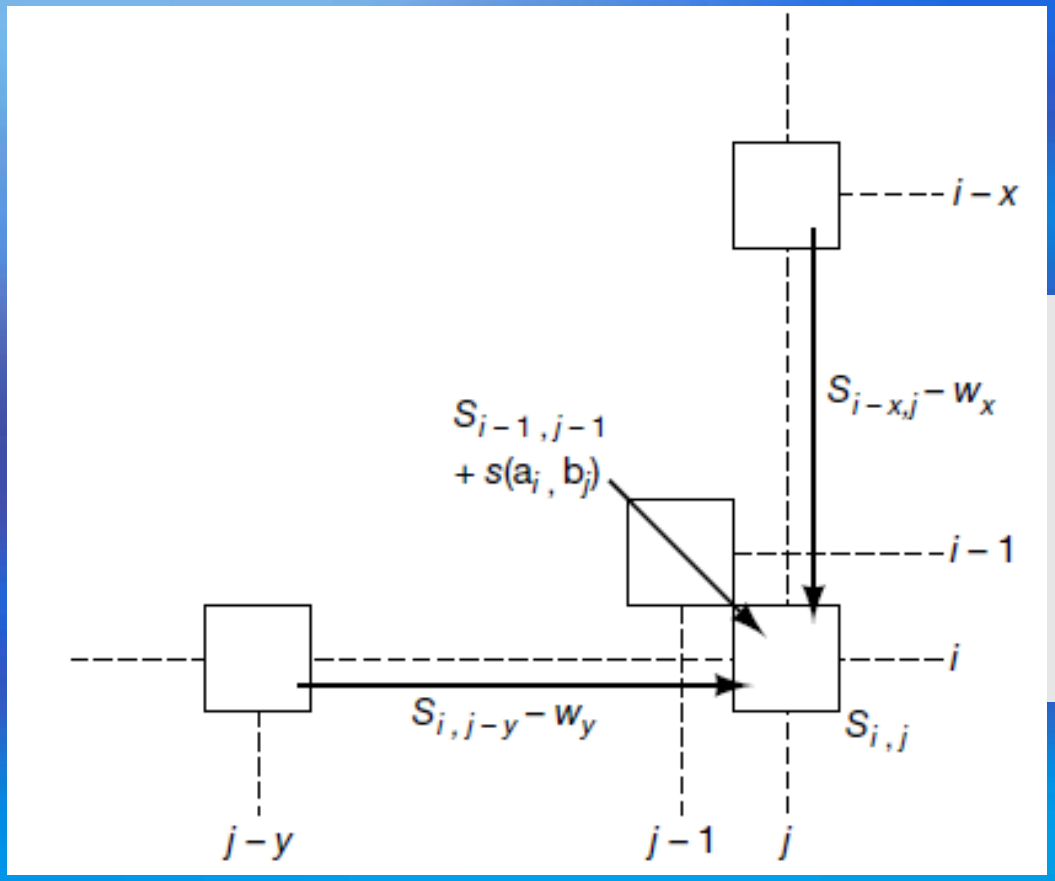
BLAST 算法

刘 欢

09.4.22

动态规划算法
Dynamic Programming Algorithm



$$S_{ij} = \max \{ S_{i-1,j-1} + s(a_i b_j), \max_{x \geq 1} (S_{i-x,j} - w_x), \max_{y \geq 1} (S_{ij-y} - w_y) \}$$

Needleman-Wunsch 算法

- 全局比对(global alignment)
- 动态规划算法
- S(0,0)= 0

Needleman-Wunsch 算法

打分矩阵

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

空位罚分（gap penalty） = -5

		A	A	G
	0			
A				
G				
C				

Needleman-Wunsch 算法

打分矩阵

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

		A	A	G
	0 →	-5 →	-10 →	-15
A	-5 ↓			
G	-10 ↓			
C	-15			

空位罚分 = -5

Needleman-Wunsch 算法

打分矩阵

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

		A	A	G
	0 →	-5 →	-10 →	-15
A	-5 ↓	2 →	-3 →	-8
G	-10 ↓	-3 ↓	-3 ↓	-1 ↓
C	-15	-8	-8	**-6**

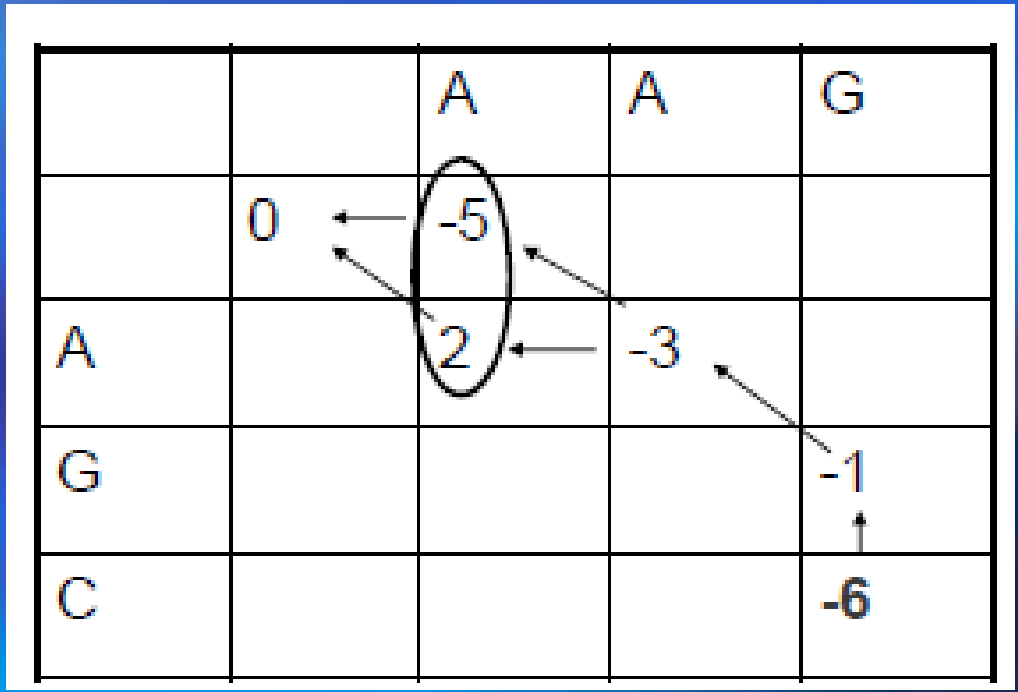
空位罚分 = -5

Needleman-Wunsch 算法

打分矩阵

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

空位罚分 = -5



Smith-Waterman 算法

- 局部比对(local alignment)
- 动态规划算法
- 矩阵 H

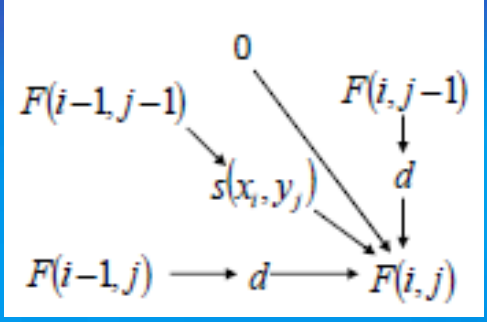
$$H(i,0) = 0,\ 0 \le i \le m$$
$$H(0,j) = 0,\ 0 \le j \le n$$
$$H(i,j) = \max \begin{cases} 0 & \\ H(i-1,j-1) + w(a_i, b_j) & \text{Match/Mismatch} \\ H(i-1,j) + w(a_i, -) & \text{Deletion} \\ H(i,j-1) + w(-, b_j) & \text{Insertion} \end{cases}$$

Smith-Waterman 算法

打分矩阵

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

空位罚分 = -5



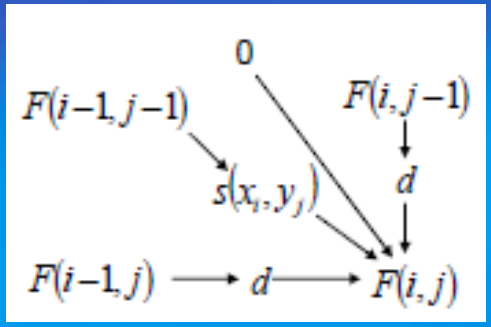
		A	A	G
A				
G				
C				

Smith-Waterman 算法

打分矩阵

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

空位罚分 = -5



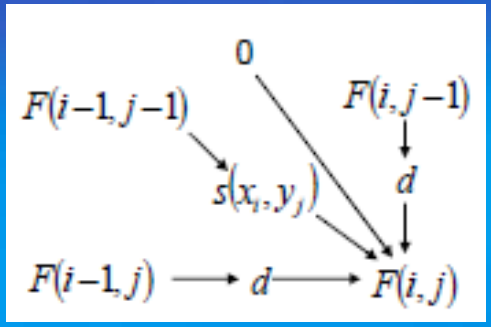
		A	A	G
	0	0	0	0
A	0			
G	0			
C	0			

Smith-Waterman 算法

打分矩阵

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

空位罚分 = -5



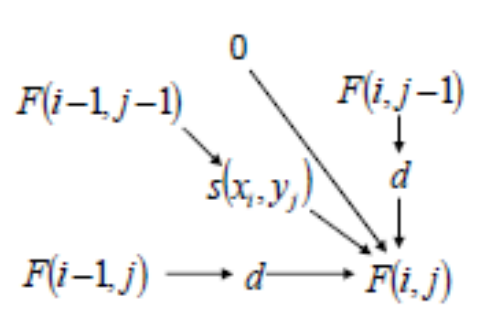
		A	A	G
	0	0	0	0
A	0	2	2	0
G	0	0	0	4
C	0	0	0	0

Smith-Waterman 算法

打分矩阵

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

空位罚分 = -5



		A	A	G
	0	0	0	0
A	0	2	2	0
G	0	0	0	4
C	0	0	0	0

Smith-Waterman 算法

- 时间和占用资源的代价
  O(mn)

- 代价太高!

Blast的技术性创新

- 启发式近似算法（Heuristic approach）
- 不如Smith-Waterman算法精确，但比Smith-Waterman算法快50倍
- 速度 & 相对高的准确性

Blast 算法

1.移除查询序列中的低复杂度区域或重复序列 。

蛋白序列: SEG program

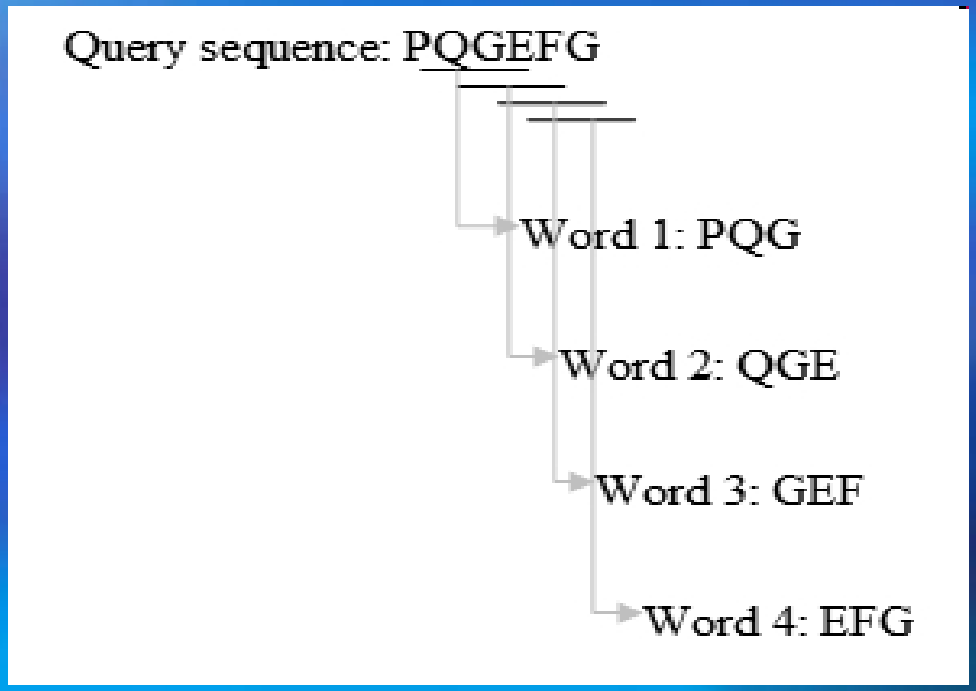
DNA序列: DUST program

串联重复: XNU program

Blast算法

2做查询序列的k-字母单词表

k=3



DNA 序列: K通常取11

Blast 算法

3 列出可能的匹配k-字母

- 比较所有的k-字母和在步骤2中列出的k-字母
- 运用打分矩阵为比较的每个残基对打分
- 邻近字母打分阈值 T(neighborhood word score)
- 例如: PQG & PEG(15), PQA(12)

  T=13

Blast 算法

4 将剩余的高打分k-字母组织成有效的查询树

Blast 算法

5 对于每一个3-字母词，重复步骤1-4

Blast 算法

6 对于剩余的高打分词，在数据库序列中搜索精确匹配的

- BLAST程序在数据库序列中的每一个位置，寻找高打分的词。如果找到一个精确的匹配，那么这个匹配将被作为种子。

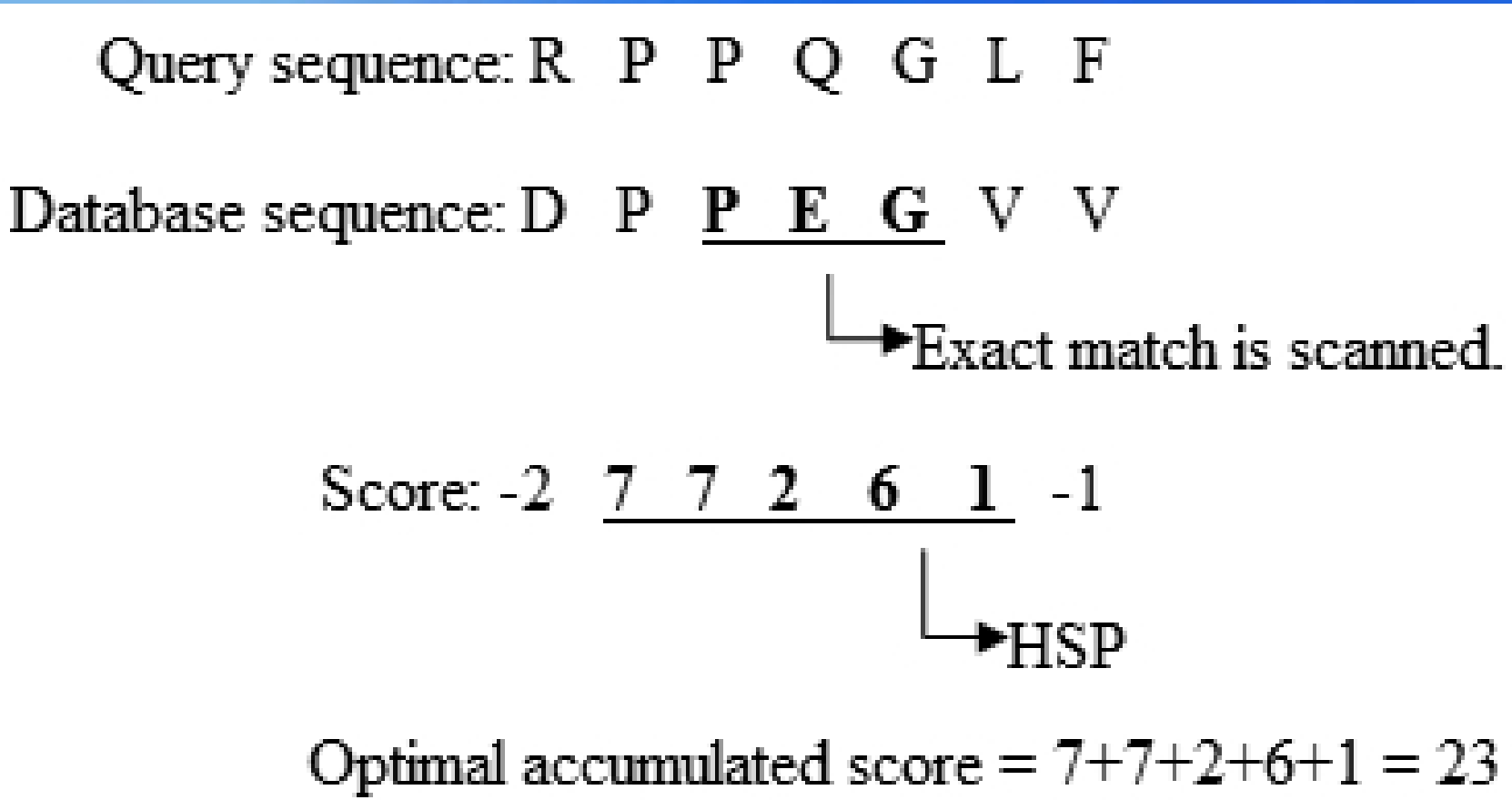
- 基本思想:seeding-and-extending

Blast算法

7 将精确匹配（种子）延伸成高打分匹配片断（high-scoring segment pair, HSP）

- 将种子向左右两侧延伸
- 直到最高的打分停止

Blast算法


Query sequence: R P P Q G L F
Database sequence: D P P E G V V
Exact match is scanned.
Score: -2 7 7 2 6 1 -1
HSP
Optimal accumulated score = 7+7+2+6+1 = 23

Blast算法

8 列出数据库中所有打分足够高的**HSPs**

- S: cutoff score
- 通过比较随机序列比较的匹配打分的分布，可确定阈值S
- 保证剩余HSP的显著性

Blast算法

9 评定HSP打分的显著性

- 两个随机序列的打分服从Gumbel极值分布(EVD)
- 概率密度函数 $f(x) = e^{-x}e^{-e^{-x}}.$
- 根据EVD,观察到打分S大于等于x的概率为

$$p(S \geq x) = 1 - \exp\left(-e^{-\lambda(x-\mu)}\right)$$

$$\mu = {[\log(Km'n')]}/{\lambda}$$

Blast算法

- λ 和 K :依赖于替换矩阵，gap罚分和序列组成
- m’和n’分别是查询序列和数据库序列的有效长度

$$m' \approx m - {(\ln Kmn)}/{H}$$
$$n' \approx n - {(\ln Kmn)}/{H}$$

- Altschul and G[illegible]为打分矩阵时，无空位局部比对的λ = 0.318, K = 0.13, H = 0.40。

Blast算法

- 期望打分E:一个数据库匹配的期望打分E是在随机的情况下，一个无关序列库获得打分S高于x的次数。在一个含有D条序列的数据库中搜索时E为：

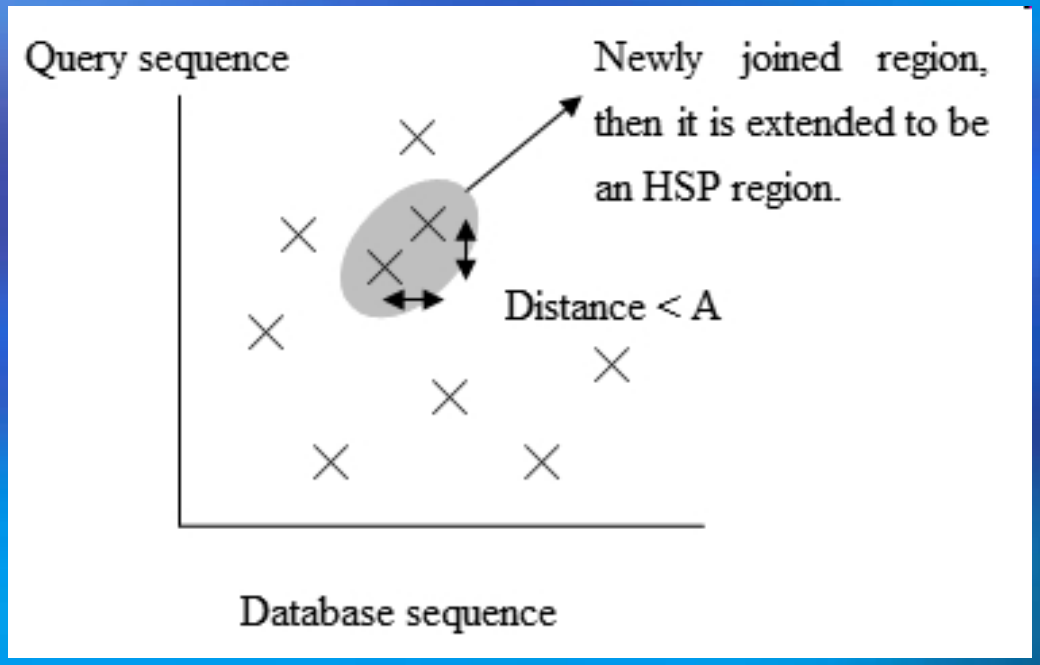
$$E \approx 1 - e^{-p(s>x)D}$$

- p<0.1 时， E~Poisson Distribution

$$E \approx pD$$

Blast算法

10 将两个或多个HSP区域构成一个长的匹配 。



Blast算法

- 两种方法比较新形成的HSP区域的显著性：

Poisson method

（65，40）和（52，45）

倾向于最低打分较高的（45 > 40）

sum-of-scores method

倾向于总打分最高的（60+45 > 52+45）

Blast算法

11 显示查询序列与每一个匹配数据库序列的有空位**Smith-Waterman**局部比对结果

12 根据**E**值列出匹配

谢谢大家!