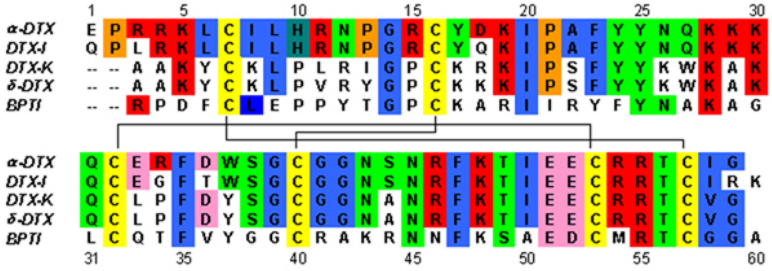
Health Informatics Lab@USTC

第三章
动态规划算法和序列比对



动态规划算法

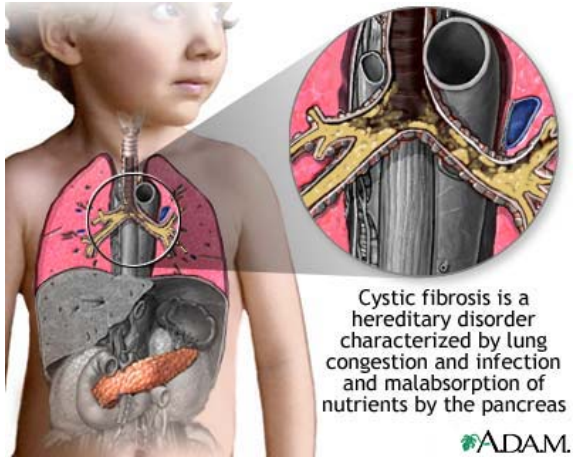
- 思路：问题分解为若干较小的问题
  - 子问题数目多
  - 子问题需要重复求解
- 优点：避免重复计算，节约时间
- 应用实例：
  - 工程领域：最长/短路径问题，背包问题，项目管理，网络流优化等；
  - 生物信息学：序列比对，隐Markov模型（HMM）等；

DNA序列比较

- 对于一个新发现的基因，推断其功能的常用途径是寻找与已知功能基因的序列相似性
- 1984年Doolittle和他的同事发现了致癌基因同血小板生长因子基因(PDGF)之间的相似性

Cystic Fibrosis

- 囊性纤维化(Cystic fibrosis, CF) 是一种慢性、死亡率高的人体粘液腺遗传疾病
- CF主要影响儿童的呼吸系统



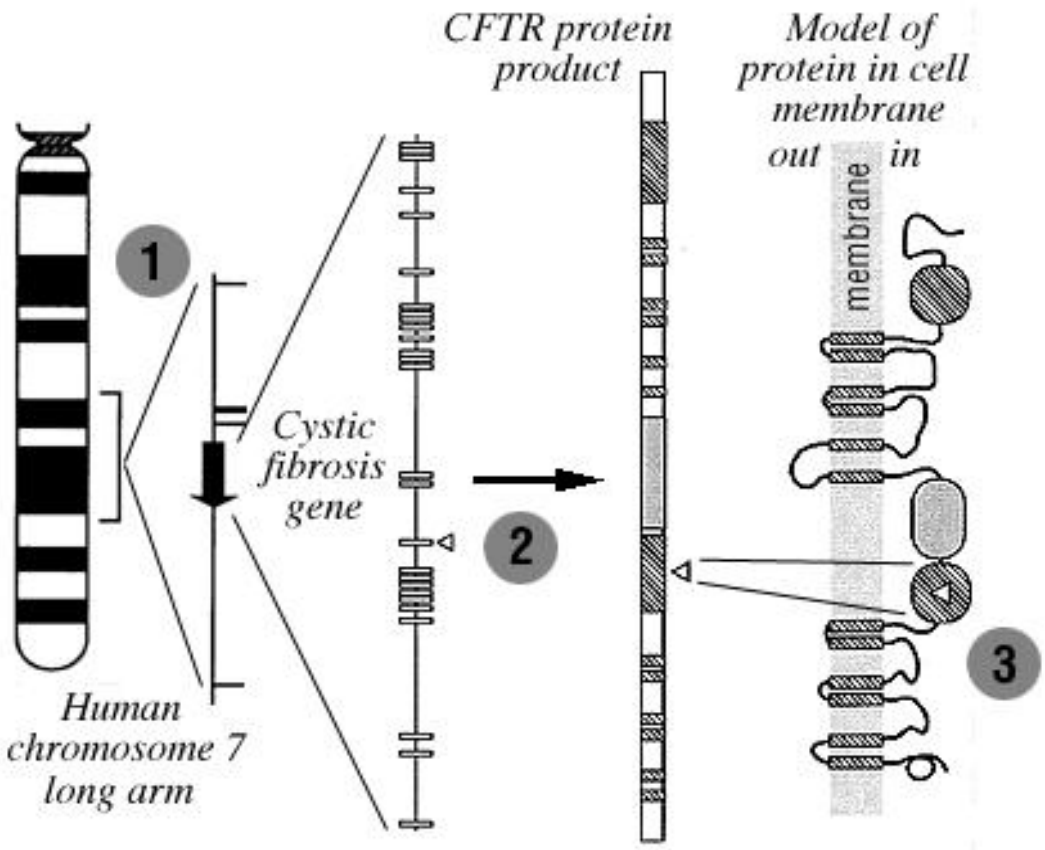
Health Informatics Lab@USTC

CF的遗传特性

- 20世纪80年代初，生物学家推测，CF是由一个未知基因突变引起的一种常染色体隐性遗传疾病，该基因直到1989年才被发现

- 杂合子(Heterozygous)携带者无症状，必须是纯合隐性(homozygous recessive)才能被诊断

寻找CF的致病基因


CFTR protein product
Model of protein in cell membrane
out
in
membrane
1
Cystic fibrosis gene
2
3
Human chromosome 7 long arm

Health Informatics Lab@USTC

寻找CF基因和ATP结合蛋白之间的相似性

- ATP结合蛋白在细胞膜上，起到运输离子的通道作用
- 在1989年，生物学家发现CF基因和ATP结合蛋白之间的相似性
- 因为CF的症状之一就是汗液中异常高的钠离子水平，因此这是一个合理的囊性纤维化基因的功能

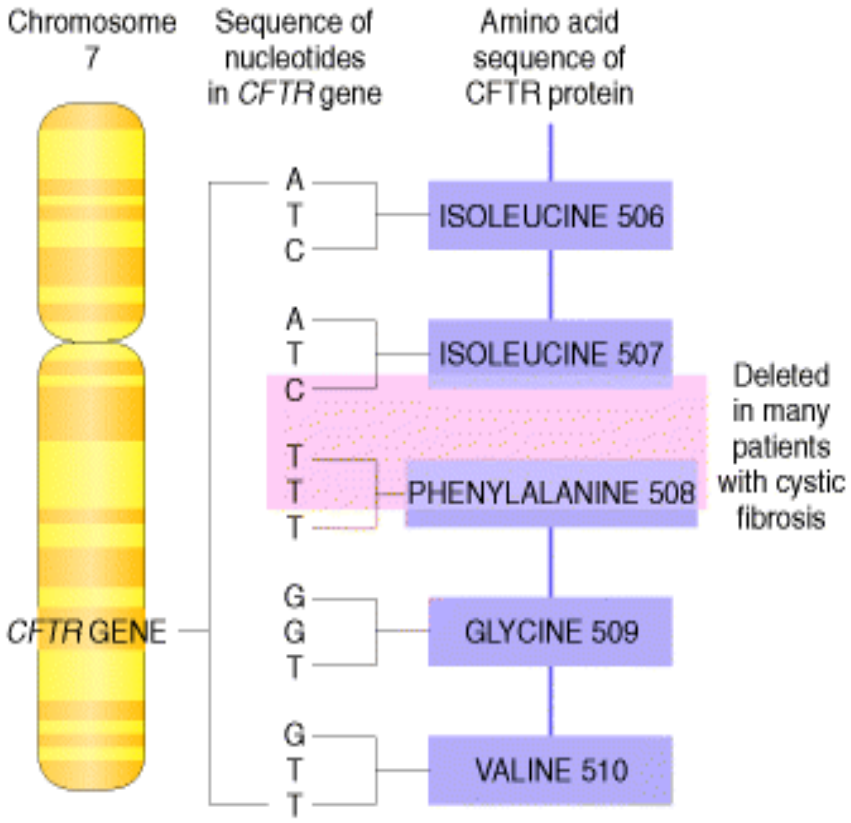
Health Informatics Lab@USTC

CF的突变分析

- 如果高比例的CF患者有特定的基因突变而正常患者没有，则其可能是一个CF突变的指标

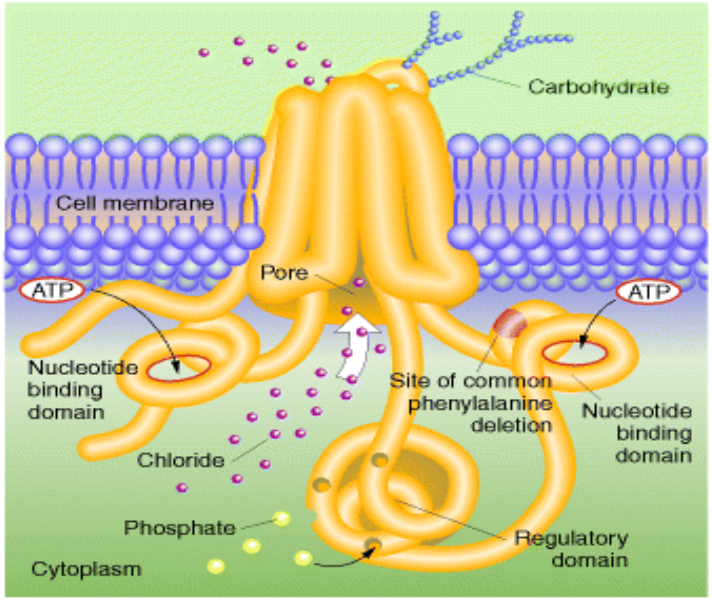
- 70%CF患者中发现了一个特定的基因突变，这是表明它是CF的一个主要遗传诊断标记。

CF和CFTR基因


Chromosome 7
Sequence of nucleotides in CFTR gene
Amino acid sequence of CFTR protein
CFTR GENE
A T C
ISOLEUCINE 506
A T C
ISOLEUCINE 507
T T T
PHENYLALANINE 508
Deleted in many patients with cystic fibrosis
G G T
GLYCINE 509
G T T
VALINE 510

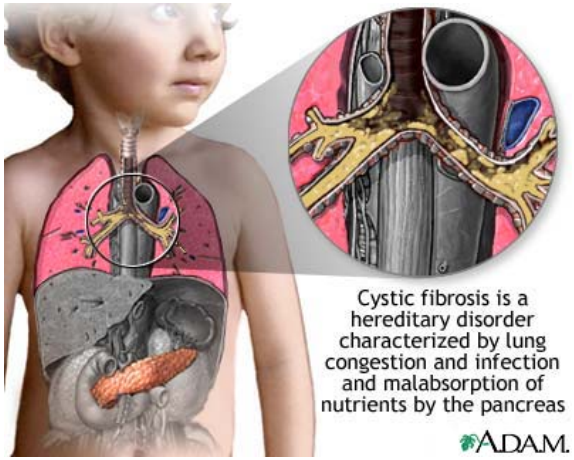
CF和CFTR蛋白

- CFTR蛋白在上皮细胞(功能为分泌粘液)的细胞膜上
- 这些细胞排列于呼吸道的鼻子，肺，胃壁等



CF的致病机制

- CFTR蛋白（包括1480个氨基酸）的功能为调节氯离子通道,调整由细胞分泌液体的含水量
- CF患者体内的CFTR缺少一个氨基酸，引起粘液过稠，从而最终导致发病



生物信息学家的贡献

- 计算两个基因之间的相似性得分，可以说明他们有类似功能的可能性有多大
- 动态规划(dynamic programming)是揭示基因之间的相似之处的技术

寻找序列相似性的问题

- 序列1
  - AGGVLIIQVG
- 序列2
  - AGGVLIQVG

- 由于空位(Gap)的引入，使得寻找两个序列相似性的问题复杂度大大提高
  - 动态规划算法

Health Informatics Lab@USTC

例子1：拿石头的问题

- 游戏规则（小强、旺财）
  - 两堆石头，各10块（记为10+10）
  - 每轮从一堆石头中拿走一块，或者两堆石头中各拿走一块 小强先拿
  - 石头不准放回
  - 谁拿走最后一块即赢得比赛

分析问题

- 简化的问题：2+2
  - 策略：
    - 情况1 小强：一块石头，旺财：同一堆一块石头；-> 小强输旺财赢
    - 情况2 小强：两块石头，旺财：两块石头；-> 小强输旺财赢
  - 结论：2+2 旺财后拿必胜！
  - 3+3?
  - 4+4?
  - …
  - 10+10?

游戏胜负表

	0	1	2	3	4	5	6	7	8	9	10
0	*	←	*	←	*	←	*	←	*	←	*
1	↑	↖	↑	↖	↑	↖	↑	↖	↑	↖	↑
2	*	←	*	←	*	←	*	←	*	←	*
3	↑	↖	↑	↖	↑	↖	↑	↖	↑	↖	↑
4	*	←	*	←	*	←	*	←	*	←	*
5	↑	↖	↑	↖	↑	↖	↑	↖	↑	↖	↑
6	*	←	*	←	*	←	*	←	*	←	*
7	↑	↖	↑	↖	↑	↖	↑	↖	↑	↖	↑
8	*	←	*	←	*	←	*	←	*	←	*
9	↑	↖	↑	↖	↑	↖	↑	↖	↑	↖	↑
10	*	←	*	←	*	←	*	←	*	←	*

问题分解

- 10+10:
  - 9+9，10+9， 9+10
    - …
      - 2+2
- 子问题组合
- 2+2 -> 3+3 ->4+4 ->… ->10+10
- 对于某个子问题n+m，R(n+m)为判断此时玩家胜负的函数
  - R（2，2）= L，lose
  - R（2，1）= W，win

游戏胜负表

	0	1	2	3	4	5	6	7	8	9	10
0		W	L	W	L	W	L	W	L	W	L
1	W	W	W	W	W	W	W	W	W	W	W
2	L	W	L	W	L	W	L	W	L	W	L
3	W	W	W	W	W	W	W	W	W	W	W
4	L	W	L	W	L	W	L	W	L	W	L
5	W	W	W	W	W	W	W	W	W	W	W
6	L	W	L	W	L	W	L	W	L	W	L
7	W	W	W	W	W	W	W	W	W	W	W
8	L	W	L	W	L	W	L	W	L	W	L
9	W	W	W	W	W	W	W	W	W	W	W
10	L	W	L	W	L	W	L	W	L	W	L

伪代码描述问题

- 推广至更一般性的n+m情况

```
ROCKS(n, m)
 1  R_{0,0} = L
 2  for i ← 1 to n
 3       if R_{i-1,0} = W
 4            R_{i,0} ← L
 5       else
 6            R_{i,0} ← W
 7  for j ← 1 to m
 8       if R_{0,j-1} = W
 9            R_{0,j} ← L
10       else
11            R_{0,j} ← W
12  for i ← 1 to n
13       for j ← 1 to m
14            if R_{i-1,j-1} = W and R_{i,j-1} = W and R_{i-1,j} = W
15                 R_{i,j} ← L
16            else
17                 R_{i,j} ← W
18  return R_{n,m}
```

更快速的算法

```
FastRocks(n, m)
1  if  n and m are both even
2        return L
3  else
4        return W
```

- 但是推广性能不如慢速算法

找钱问题

Goal：把钱数M转换为给定面额的硬币，要求使用的硬币数目最少

Input：钱数 M，硬币$c = (c_1, c_2, \cdots, c_d)$，$(c_1 > c_2 > \cdots > c_d)$

Output：整数列 $i_1, i_2, \cdots, i_d$ 使得

$$c_1 i_1 + c_2 i_2 + \cdots + c_d i_d = M$$

并且 $i_1 + i_2 + \cdots + i_d$ 最小

找钱问题：递归

这个例子由以下递推关系表示:

$$\text{minNumCoins}(M) = \text{min of} \begin{cases} \text{minNumCoins}(M-1) + 1 \\ \text{minNumCoins}(M-3) + 1 \\ \text{minNumCoins}(M-7) + 1 \end{cases}$$

找钱问题：递归

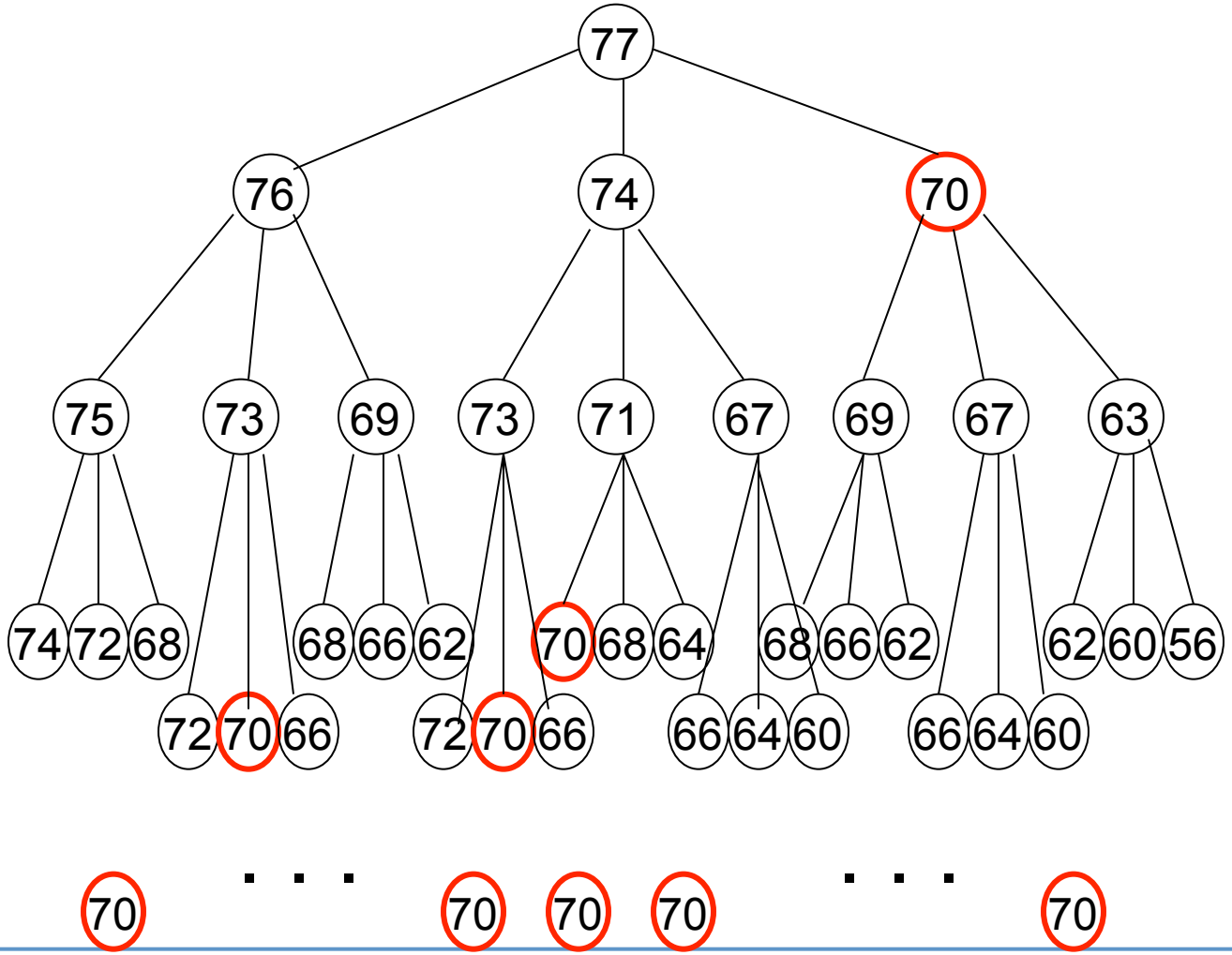
对于面值为c：c_1，c_2，…，c_d的硬币，递归公式为：

$$\text{minNumCoins}(M) = \text{min of} \begin{cases} \text{minNumCoins}(M-c_1) + 1 \\ \text{minNumCoins}(M-c_2) + 1 \\ \cdots \\ \text{minNumCoins}(M-c_d) + 1 \end{cases}$$

递归算法

```
1.  RecursiveChange(M,c,d)
2.    if M = 0
3.      return 0
4.    bestNumCoins ← infinity
5.    for i ← 1 to d
6.      if M ≥ c_i
7.        numCoins ← RecursiveChange(M – c_i , c, d)
8.        if numCoins + 1 < bestNumCoins
9.          bestNumCoins ← numCoins + 1
10.   return bestNumCoins
```

RecursiveChange搜索树



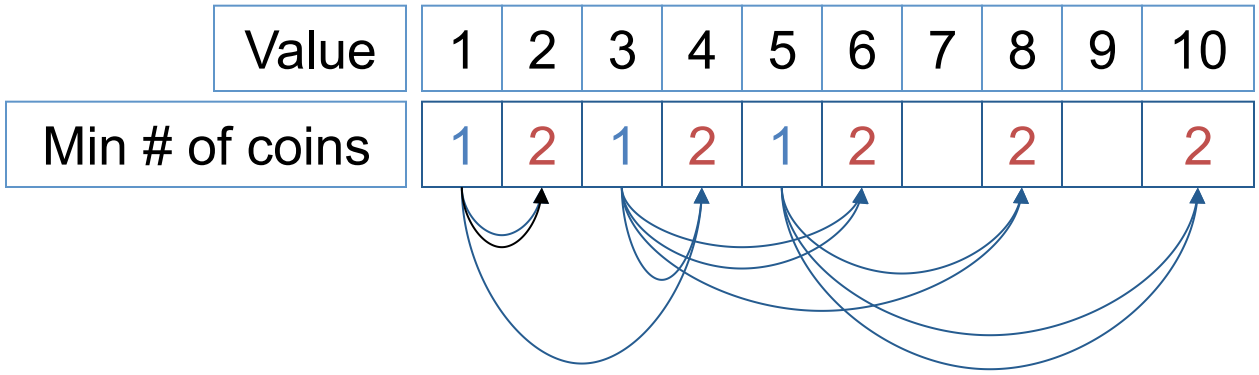
找钱问题举例

使用面额为1，3和5的硬币，对不同数目的零钱，最少需要的硬币数目？

Value	1	2	3	4	5	6	7	8	9	10
Min # of coins	1		1		1					

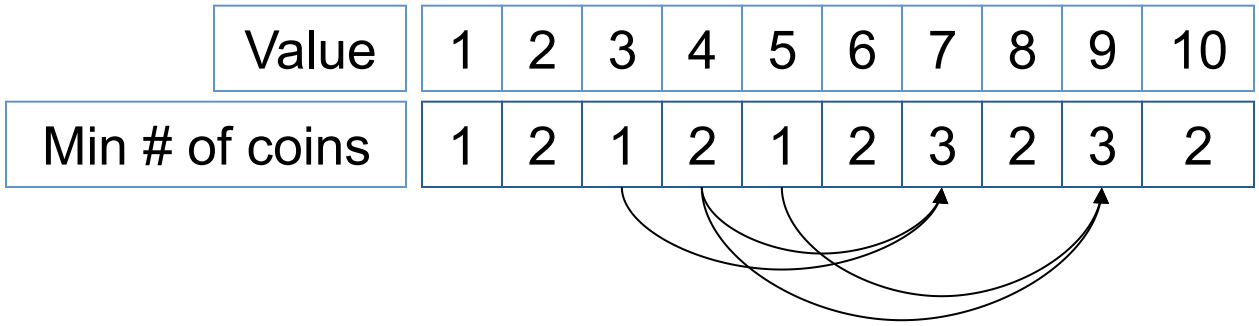
找钱问题举例

使用面额为1，3和5的硬币，对不同数目的零钱，最少需要的硬币数目？



找钱问题举例

使用面额为1，3和5的硬币，对不同数目的零钱，最少需要的硬币数目？



改进思路

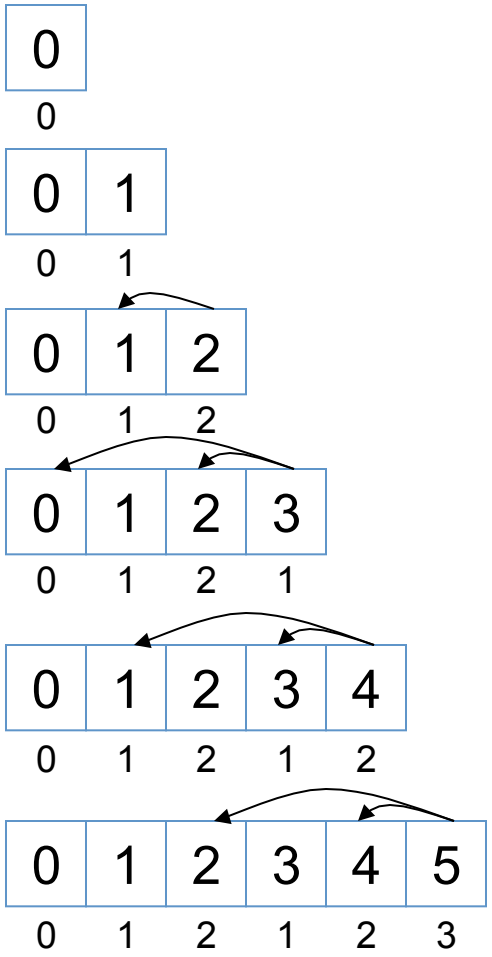
- 算法中对每个子问题进行多次计算
- 保存找钱数从0到M的每次计算结果
- 这样，每次计算时我们仅需要调用原有结果，而不是重新计算

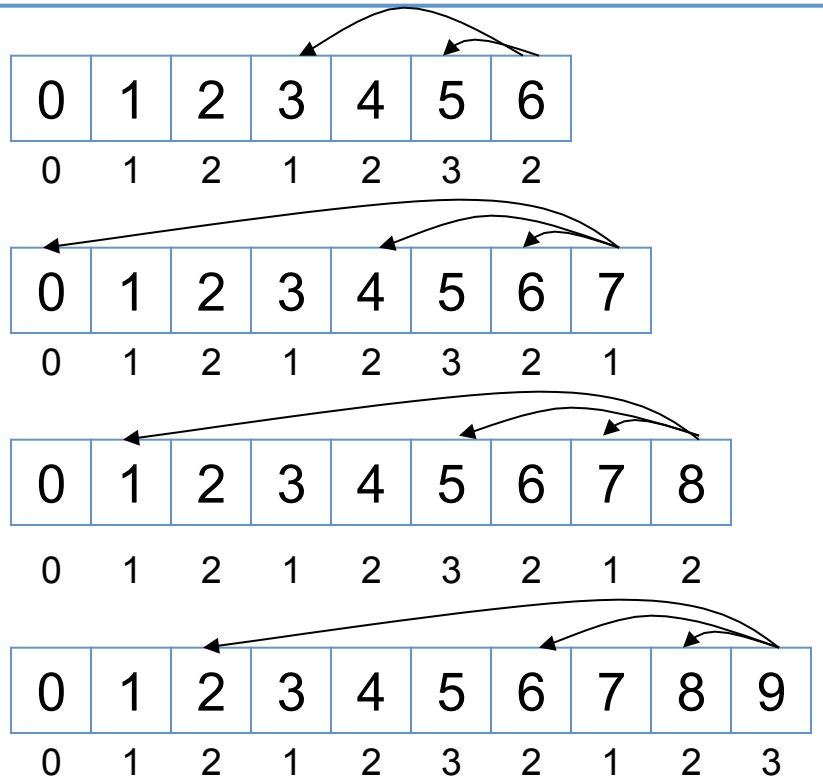
基于动态规划的找钱问题

```
1. DPChange(M,c,d)
2.     bestNumCoins_0 ← 0
3.     for m ← 1 to M
4.         bestNumCoins_m ← infinity
5.         for i ← 1 to d
6.             if m ≥ c_i
7.                 if bestNumCoins_{m - c_i} + 1 < bestNumCoins_m
8.                     bestNumCoins_m ← bestNumCoins_{m - c_i} + 1
9.     return bestNumCoins_M
```

运算复杂度O(M*d)

DPChange举例



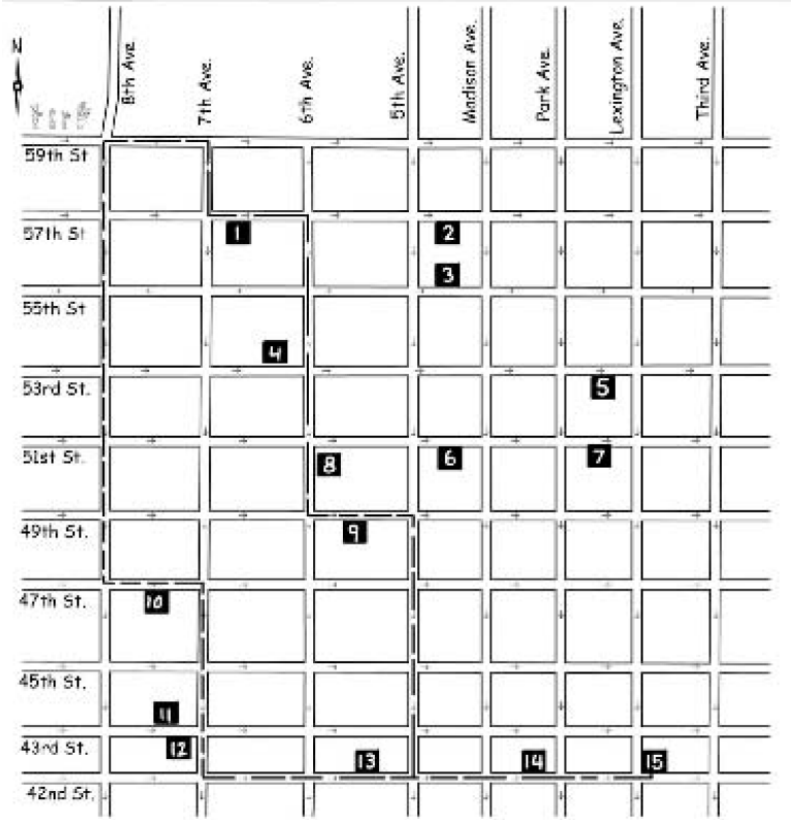


c = (1,3,7)

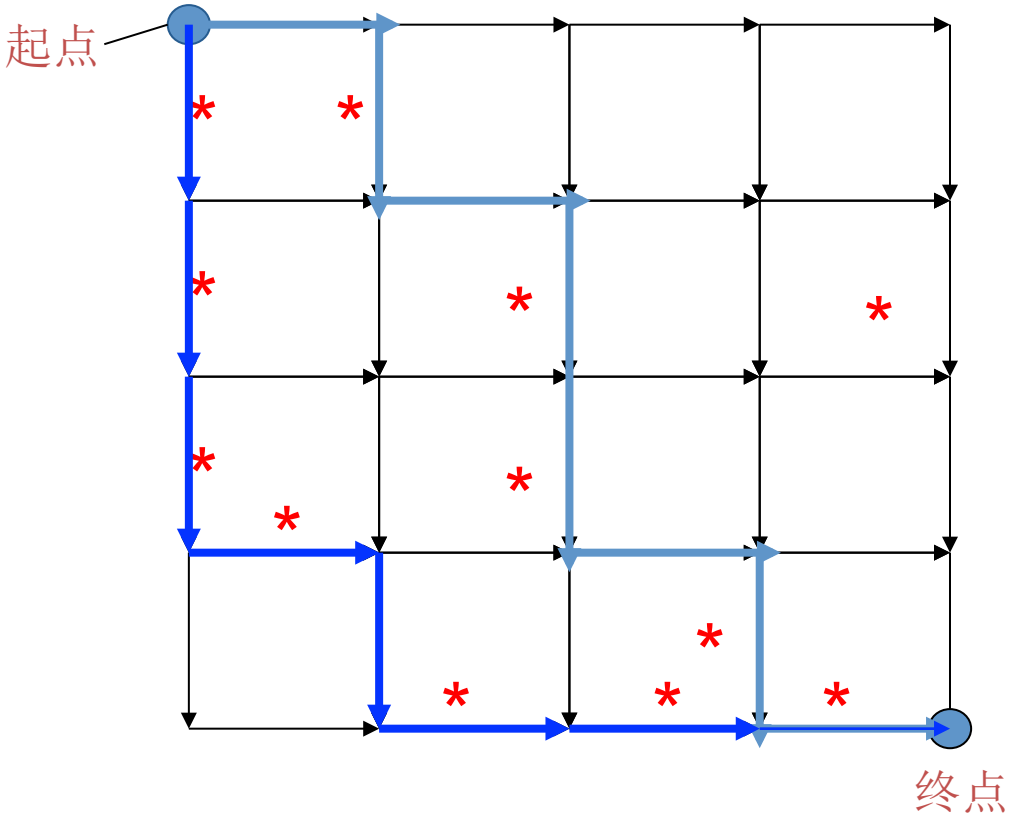
M = 9

Health Informatics Lab@USTC

例子2：曼哈顿游客问题

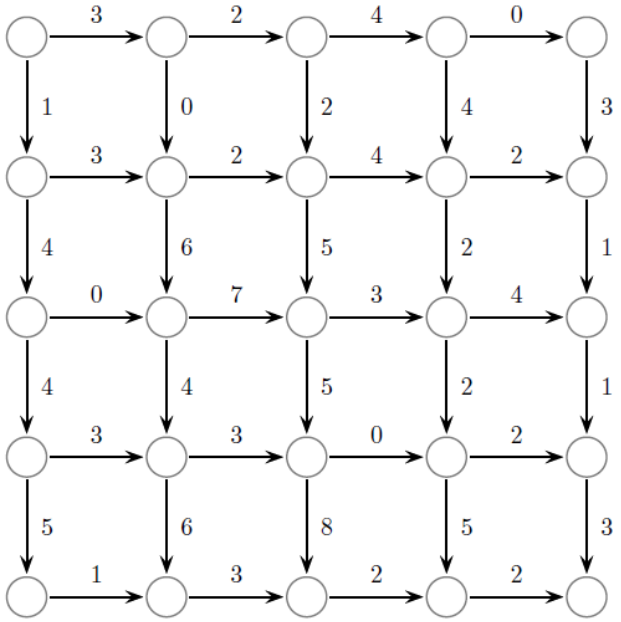


曼哈顿游客问题

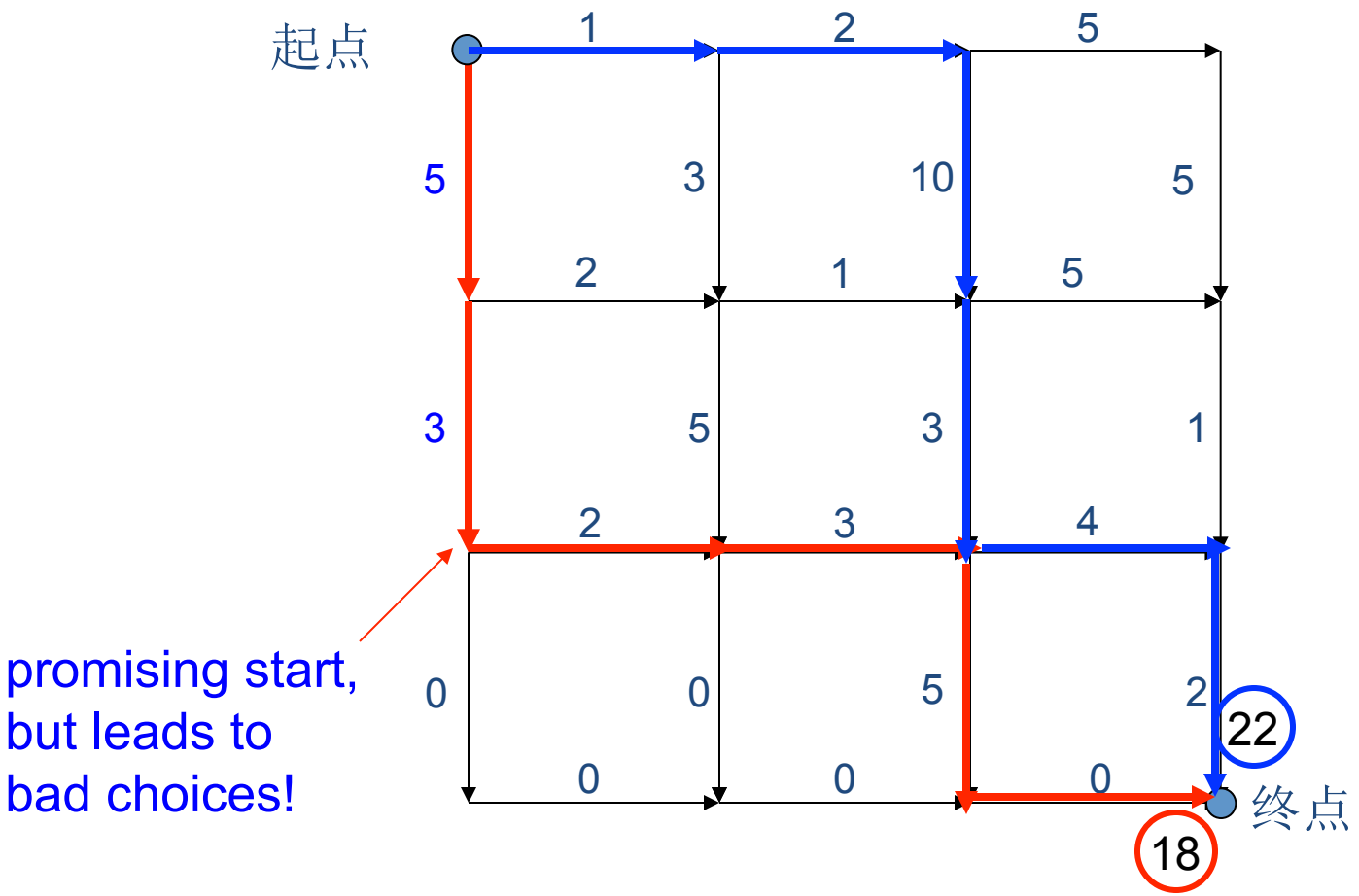


曼哈顿游客问题的描述

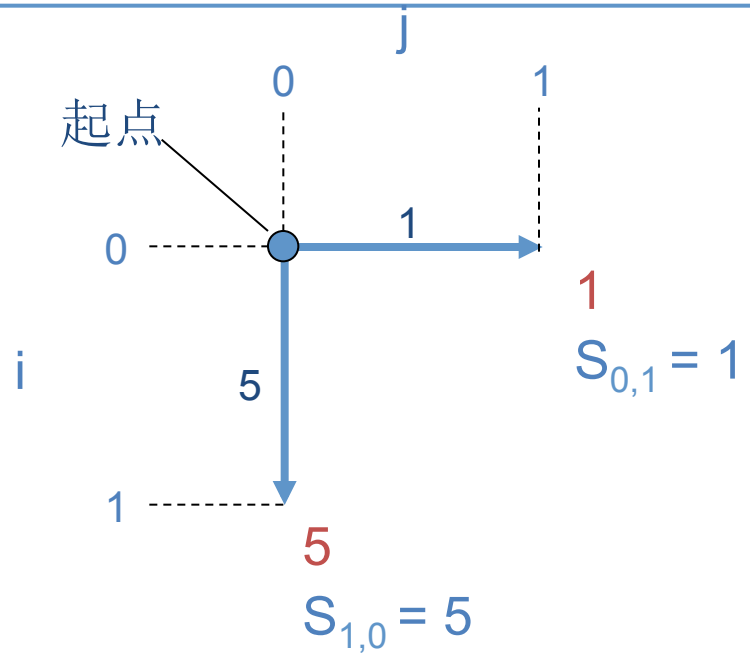
- Goal：在一个带权网格中寻找最长路径
- Input：带权网格G，包括“起点”和“终点”两个顶点
- Output：网格G中包含从起点到终点的最长路径



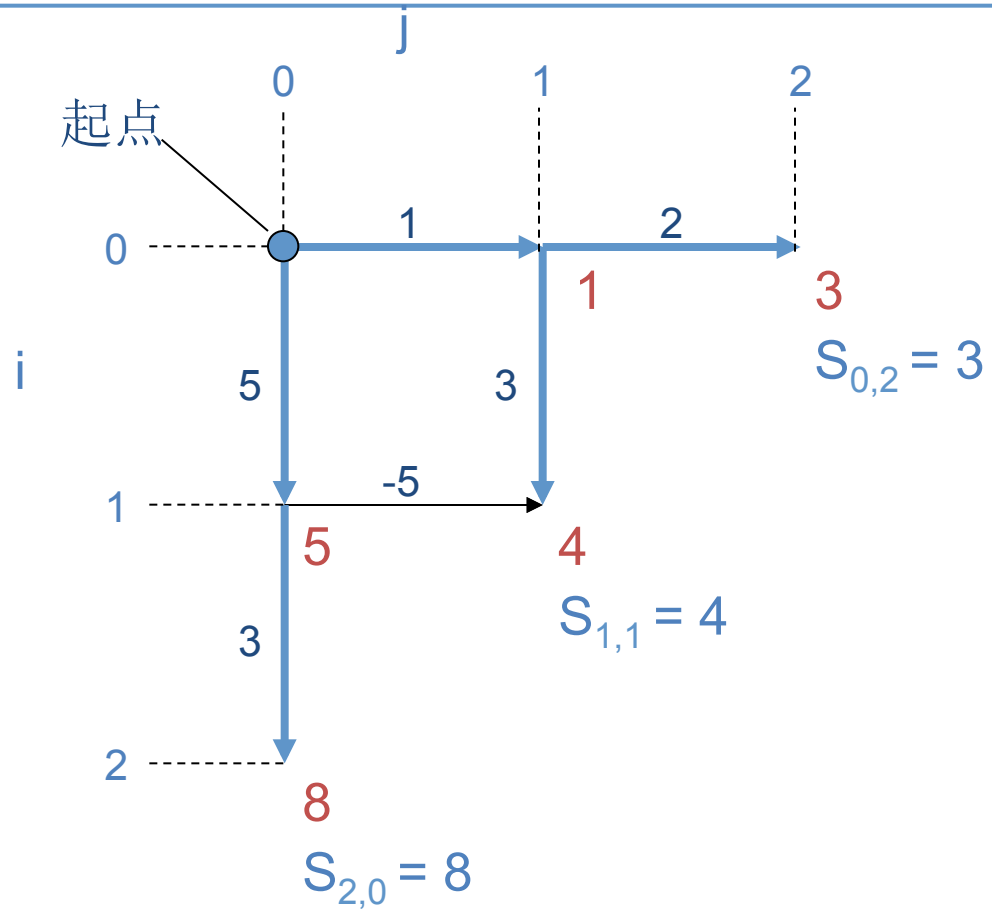
曼哈顿游客问题：贪婪算法无效



动态规划算法



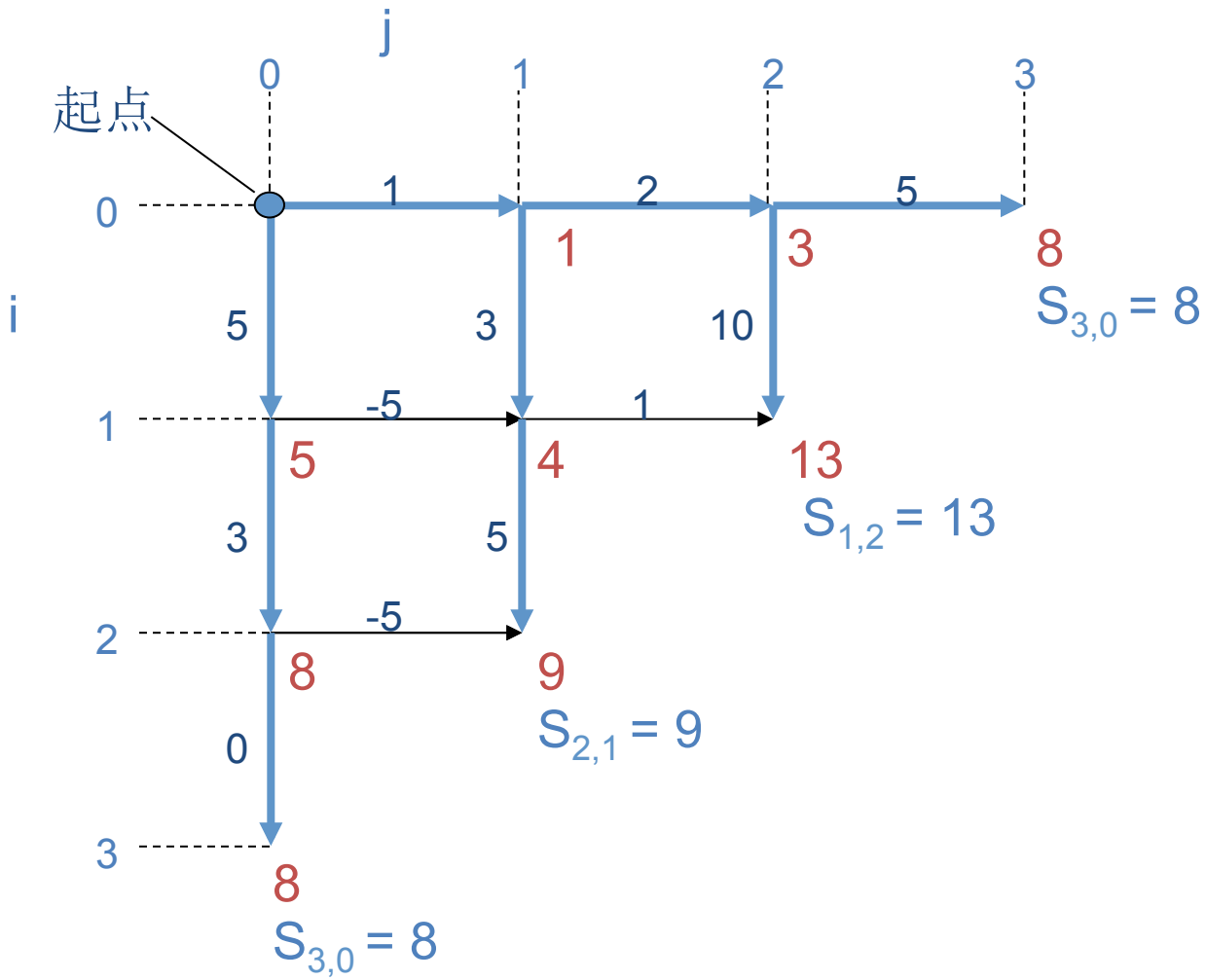
动态规划算法



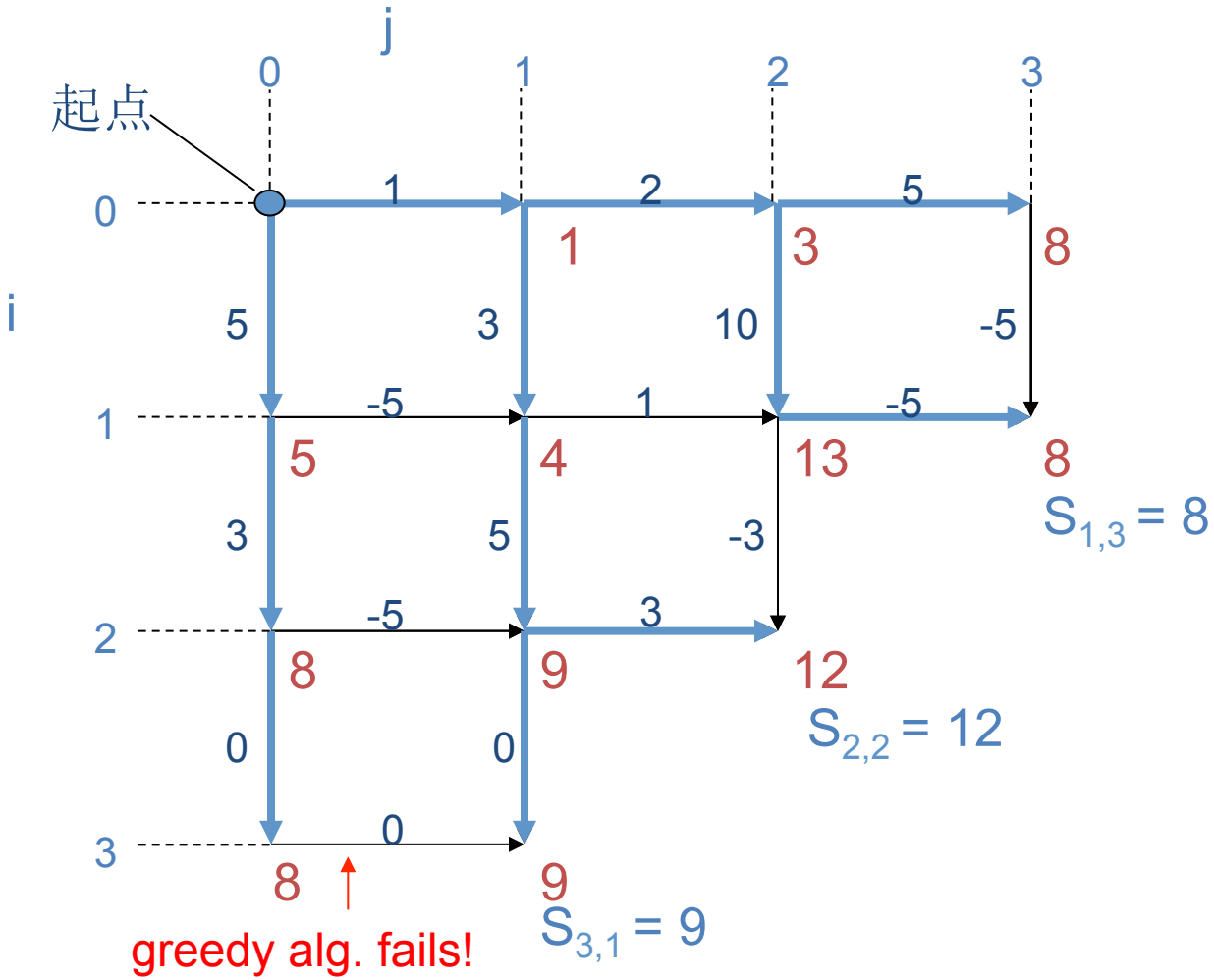
一个简单的递归程序

```
MT(n,m)
  if n=0  or m=0
     return MT(n,m)
  x ← MT(n-1,m)+
                length of the edge from (n- 1,m) to (n,m)
  y ← MT(n,m-1)+
                length of the edge from (n,m-1) to (n,m)
  return max{x,y}
```

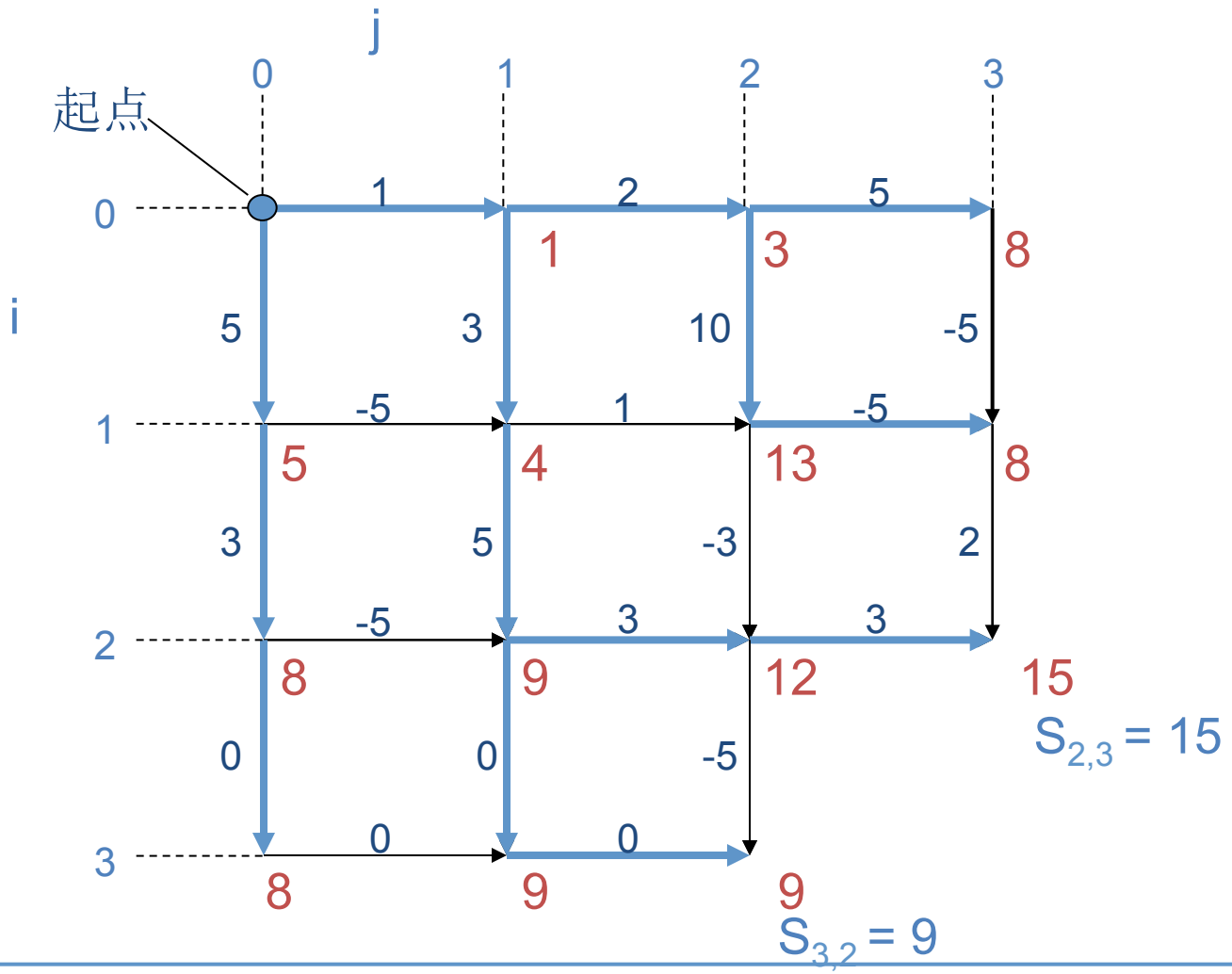
动态规划算法


j
0
1
2
3
起点
i
0
1
2
3
1
2
5
5
3
10
-5
1
3
5
-5
0
1
3
8
$S_{3,0} = 8$
5
4
13
$S_{1,2} = 13$
8
9
$S_{2,1} = 9$
8
$S_{3,0} = 8$

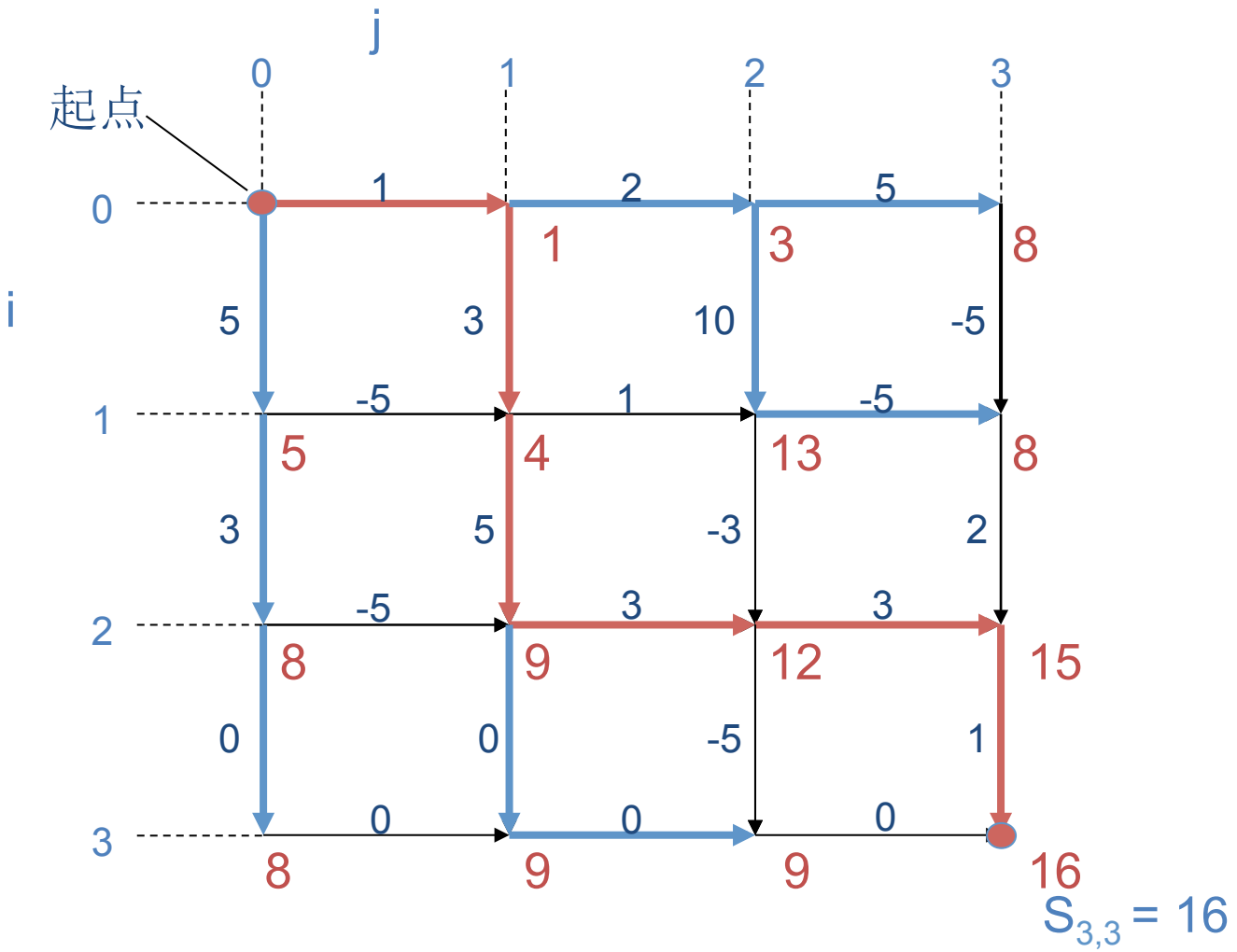
动态规划算法


j
i
起点
0
1
2
3
1
2
5
5
3
10
-5
-5
1
-5
5
4
13
8
$S_{1,3} = 8$
3
5
-3
-5
3
8
9
12
$S_{2,2} = 12$
0
0
0
8
9
$S_{3,1} = 9$
greedy alg. fails!

动态规划算法


j
i
起点
0
1
2
3
1
2
5
5
3
10
-5
-5
1
-5
3
5
-3
2
-5
3
3
0
0
-5
0
0
1
3
8
5
4
13
8
8
9
12
15
8
9
9
$S_{2,3}$ = 15
$S_{3,2}$ = 9

动态规划算法


j
i
起点
0
1
2
3
1
2
5
1
3
8
5
3
10
-5
-5
1
-5
5
4
13
8
3
5
-3
2
-5
3
3
8
9
12
15
0
0
-5
1
0
0
0
8
9
9
16
$S_{3,3}$ = 16

递归关系

- 根据递归关系计算点(i, j)的分数

$$s_{i,j} = \max \begin{cases} s_{i-1,j} + \text{weight of the edge between (i-1, j) and (i, j)} \\ s_{i,j-1} + \text{weight of the edge between (i, j-1) and (i, j)} \end{cases}$$

- $\downarrow \mathbf{w}$ 南北方向边的权重二维数组
- $\rightarrow \mathbf{w}$ 东西方向边的权重二维数组

曼哈顿游客问题的动态规划算法

MANHATTANTOURIST($\overset{\downarrow}{\mathbf{w}}, \overrightarrow{\mathbf{w}}, n, m$)

1 $s_{0,0} \leftarrow 0$

2 **for** $i \leftarrow 1$ **to** n

3 $\quad s_{i,0} \leftarrow s_{i-1,0} + \overset{\downarrow}{w}_{i,0}$

4 **for** $j \leftarrow 1$ **to** m

5 $\quad s_{0,j} \leftarrow s_{0,j-1} + \overrightarrow{w}_{0,j}$

6 **for** $i \leftarrow 1$ **to** n

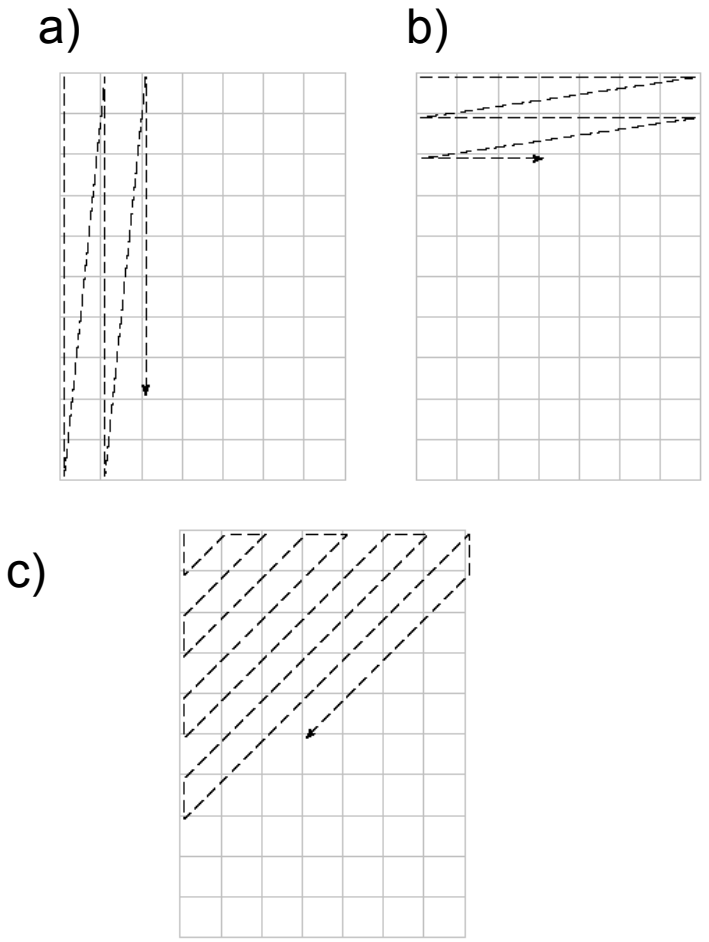
7 $\quad$ **for** $j \leftarrow 1$ **to** m

8 $$s_{i,j} \leftarrow \max \begin{cases} s_{i-1,j} + \overset{\downarrow}{w}_{i,j} \\ s_{i,j-1} + \overrightarrow{w}_{i,j} \end{cases}$$

9 **return** $s_{n,m}$

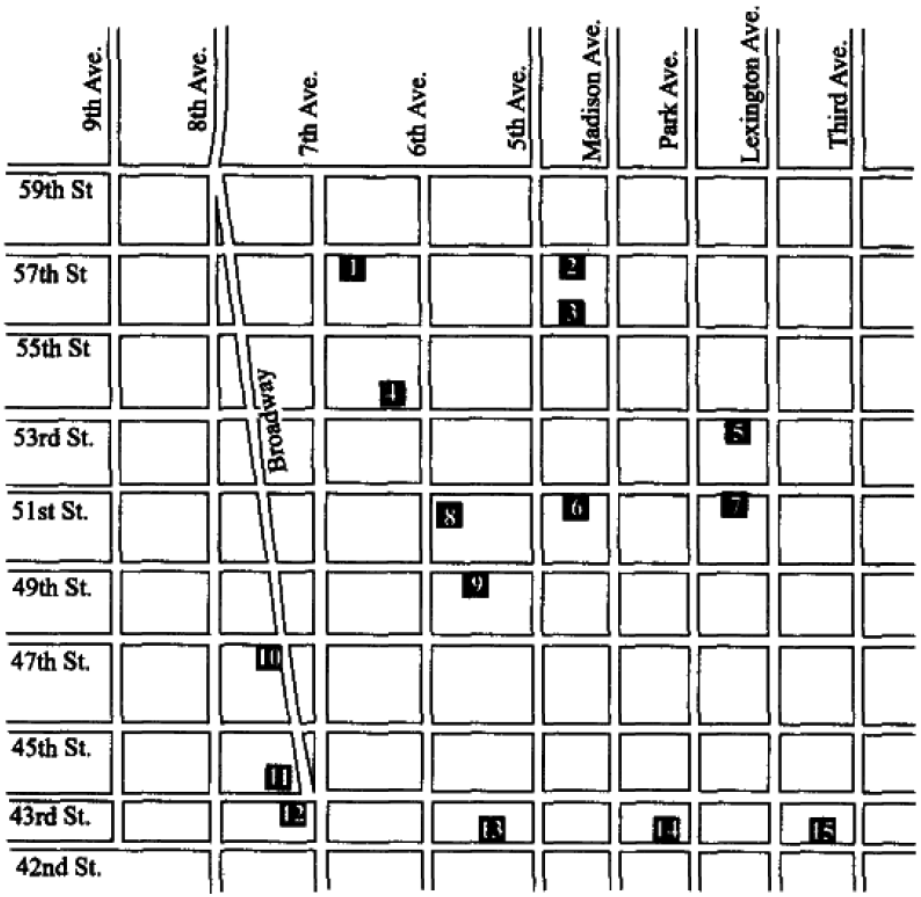
网格遍历策略

- 必须确定顶点访问的顺序
- 在顶点x被分析之前，其前导y应已经被计算，否则算法将产生问题
- 因此我们需要按照某种顺序访问顶点

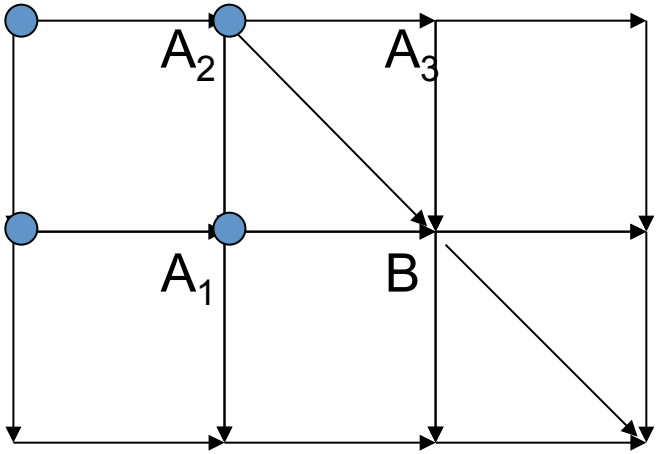


Health Informatics Lab@USTC

实际的曼哈顿地图



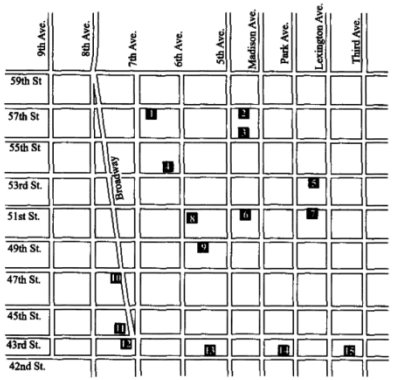
非理想网格的情况

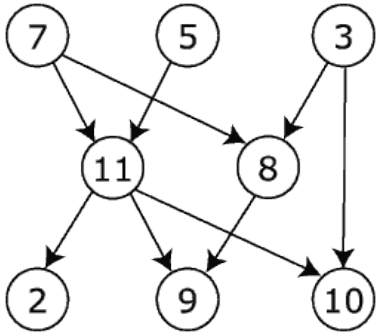


- B点的迭代公式变为：

$$s_B = \max \text{ of} \begin{cases} s_{A1} + \text{weight of the edge } (A_1, B) \\ s_{A2} + \text{weight of the edge } (A_2, B) \\ s_{A3} + \text{weight of the edge } (A_3, B) \end{cases}$$

有向无圈图（DAG）





- 有向无圈图(directed acyclic graphs，DAG)
  - 假定城市街区对应于有向边(directed edge)
  - 图中没有有向圈(directed cycles)
- 图G表示为两个集合组成的对：G=(V，E)
  - V为顶点的集合，E为边的集合

Health Informatics Lab@USTC

DAG的最长路径问题

- Goal:在一个带权DAG中寻找最长路径
- Input:带权DAG ，包括“起点”和“终点”两个顶点
- Output:带权DAG中包含从起点到终点的最长路径

DAG问题：动态规划算法

- S设顶点v的入度为3，前导集合 $\{u_1, u_2, u_3\}$
- 顶点v的最长路径为：

$$s_v = \max \text{ of } \begin{cases} s_{u_1} + \text{weight of edge from } u_1 \text{ to v} \\ s_{u_2} + \text{weight of edge from } u_2 \text{ to v} \\ s_{u_3} + \text{weight of edge from } u_3 \text{ to v} \end{cases}$$

- 更具有一般性：

$$s_v = \max_u (s_u + \text{weight of edge from u to v})$$

序列比对的例子

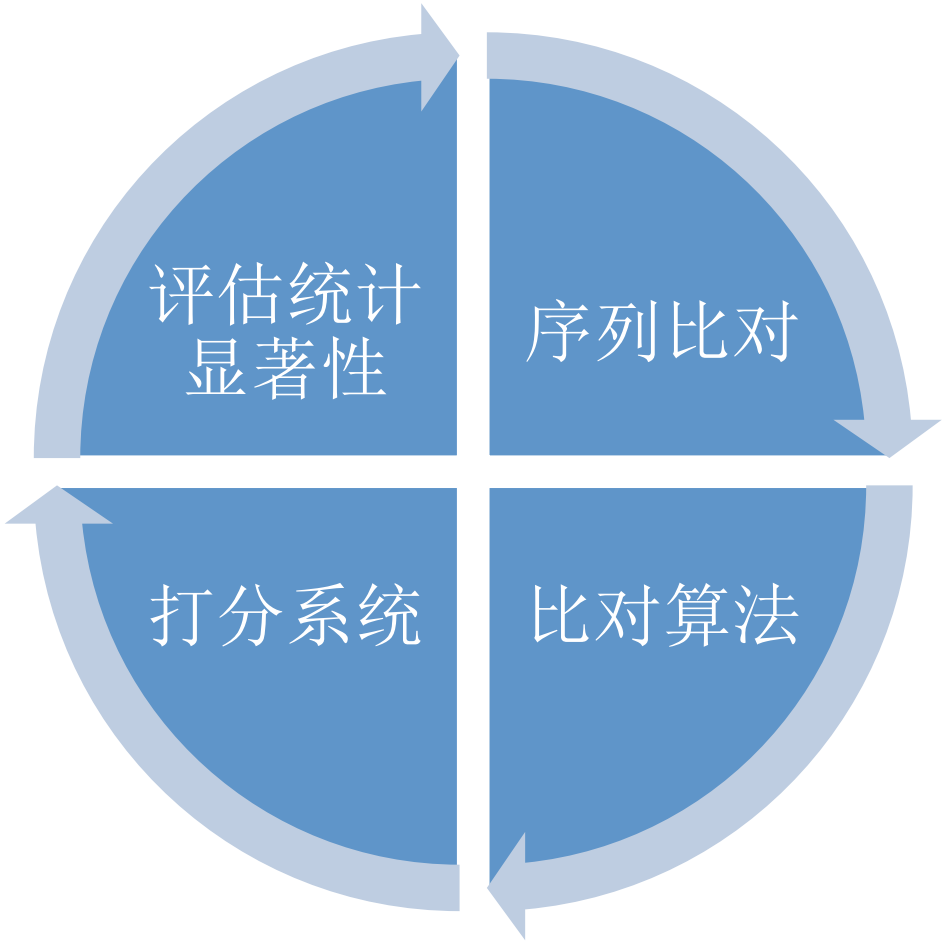
```
(a)
HBA_HUMAN  GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKL
           G+ +VK+HGKKV  A+++++AH+D++ +++++LS+LH  KL
HBB_HUMAN  GNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKL

(b)
HBA_HUMAN  GSAQVKGHGKKVADALTNAVAHV---D--DMPNALSALSDLHAHKL
           ++ ++++H+ KV   + +A  ++          +L+ L+++H+ K
LGB2_LUPLU NNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG

(c)
HBA_HUMAN  GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSD----LHAHKL
           GS+ + G +   +D L  ++ H+ D+  A +AL D    ++AH+
F11G11.2   GSGYLVGDSLTFVDLL--VAQHTADLLAANAALLDEFPQFKAHQE
```

Health Informatics Lab@USTC

序列比对的关键问题



汉明距离

- 汉明距离(Hamming distance)：是两个字符串对应位置的不同字符的个数
  - 没有删除和插入的比对方式
- 对于DNA序列v和w:

v : ATATATAT
w: TATATATA

dH(v, w)=8，但序列具有很大的相似性

删除和插入的比对方式

- 通过简单的移位

v： ATATATAT--
w： --TATATATA

- 汉明距离忽略了DNA中的插入、删除和替换现象

编辑距离(Edit Distance)

- 编辑距离(Edit Distance)：将一个字符串变为另一个的最少基本操作(包括插入、删除、替代）数目

编辑距离vs 汉明距离

Hamming distance always compares

i-th letter of v with

i-th letter of w

```
V = ATATATAT
    | | | | | | |
W = TATATATA
```

Hamming distance:

d(v, w)=8

Computing Hamming distance is a trivial task.

Health Informatics Lab@USTC

编辑距离vs 汉明距离

Hamming distance always compares i^{-th} letter of v with i^{-th} letter of w

```
V = ATATATAT
    |||| |||
W = TATATATA
```

Just one shift

Make it all line up

Edit distance may compare i^{-th} letter of v with j^{-th} letter of w

```
V = - ATATATAT
    | | | | | |
W = TATATATA
```

Hamming distance:

d(v, w)=8

Computing Hamming distance is a trivial task

Edit distance:

d(v, w)=2

Computing edit is a non-trivial task

Health Informatics Lab@USTC

编辑距离vs 汉明距离

Hamming distance always compares i^{-th} letter of v with i^{-th} letter of w

```
V = ATATATAT
    |||| |||
W = TATATATA
```

Hamming distance:

d(v, w)=8

Edit distance may compare i^{-th} letter of v with j^{-th} letter of w

```
V = - ATATATAT
    |||| |||
W = TATATATA
```

Edit distance:

d(v, w)=2

(one insertion and one deletion)

How to find what j goes with what i ???

编辑距离计算举例

TGCATAT → ATCCGAT in 5 steps

TGCATAT → (delete last T)
TGCATA → (delete last A)
TGCAT → (insert A at front)
ATGCAT → (substitute C for 3rd G)
ATCCAT → (insert G before last A)
ATCCGAT (Done)

编辑距离是5吗？

编辑距离计算举例

TGCATAT → ATCCGAT in 4 steps

TGCATAT → (insert A at front)
ATGCATAT → (delete 6th T)
ATGCATA → (substitute G for 5th A)
ATGCGTA → (substitute C for 3rd G)
ATCCGAT (Done)

编辑距离：例子

TGCATAT → ATCCGAT in 4 steps

TGCATAT → (insert A at front)
ATGCATAT → (delete 6th T)
ATGCATA → (substitute G for 5th A)
ATGCGTA → (substitute C for 3rd G)
ATCCGAT (Done)

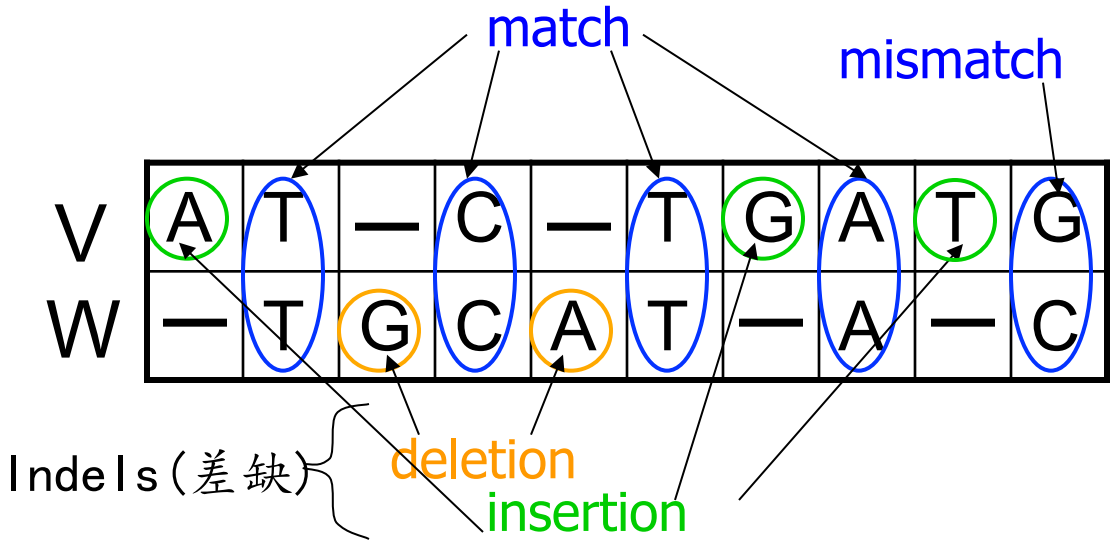
Can it be done in 3 steps???

比对(alignment)

V = ATCTGATG　　n = 8

W = TGCATAC　　m = 7

4 matches（匹配）
1 mismatches（错配）
2 deletions（缺失）
3 insertions（插入）



比对矩阵

比对计算举例

- 对于两个DNA序列v and w:

v : ATCTGAT m = 7
w : TGCATA n = 6

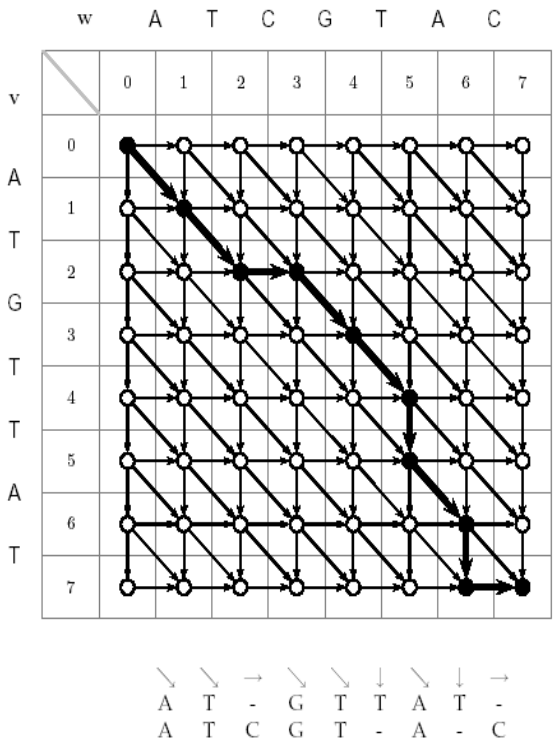
- 比对矩阵:

v	A	T	--	G	T	T	A	T	--
w	A	T	C	G	T	--	A	--	C

4 matches | 2 insertions | 2 deletions

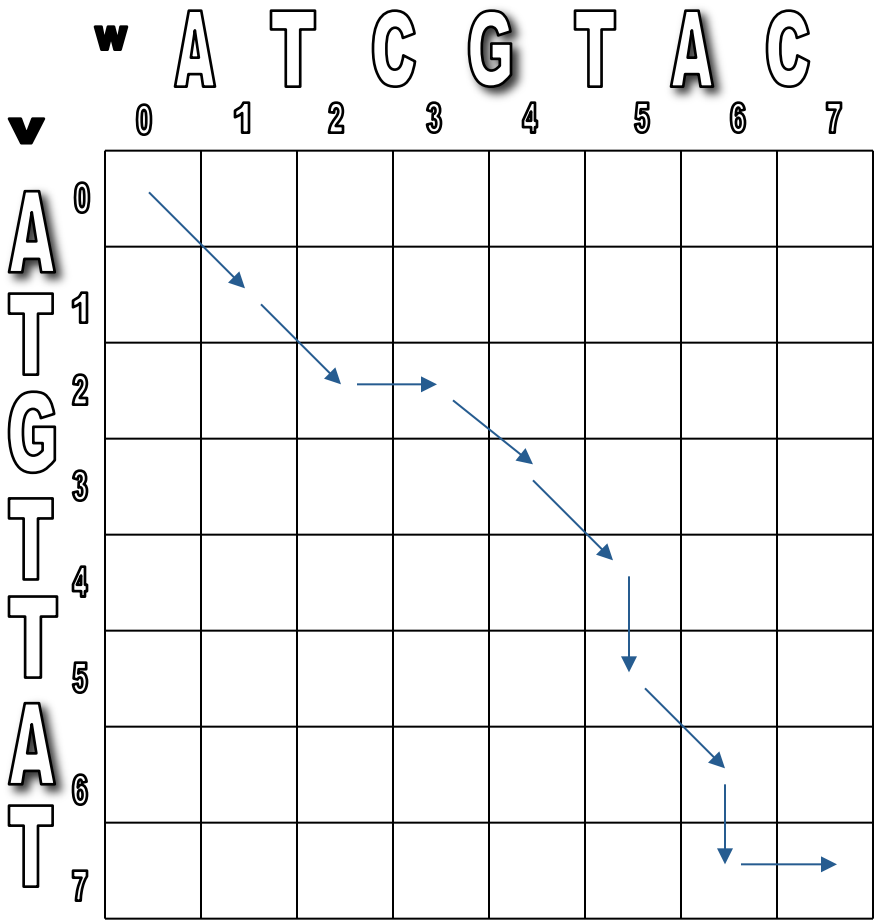
比对网格

- 通过对网格中街道的每一个交叉点引入一个顶点，可以构建一个图，称之为编辑图(edit graph)



Health Informatics Lab@USTC

比对网格

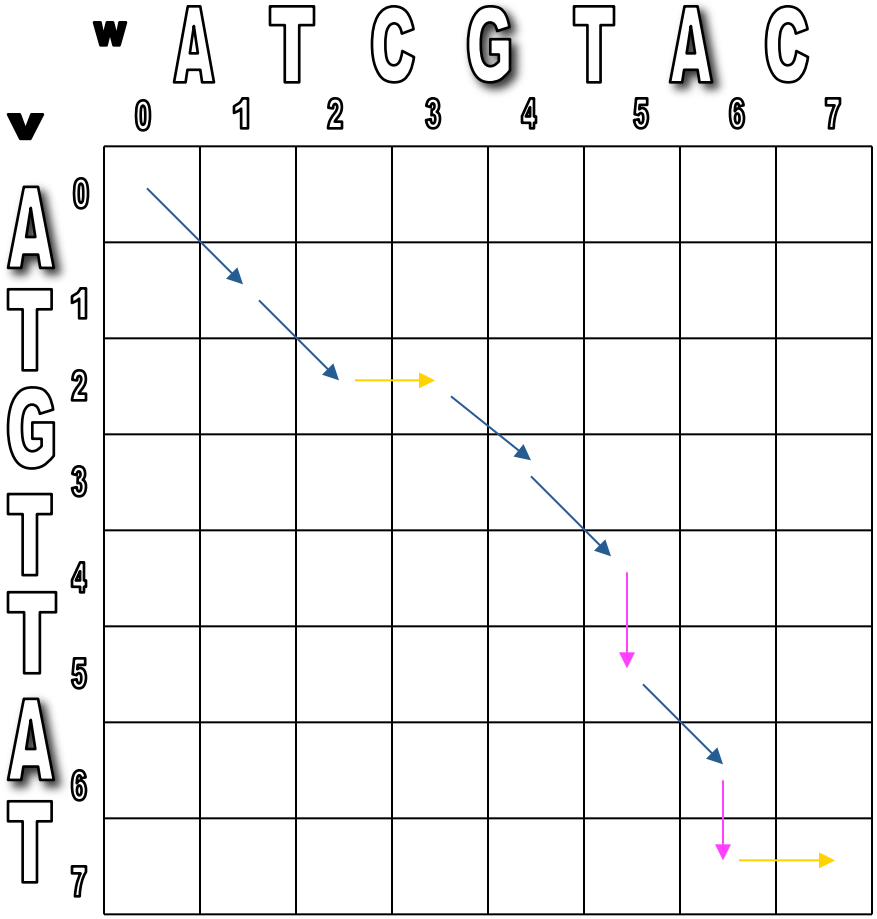


0	1	2	2	3	4	5	6	7	7
	A	T	_	G	T	T	A	T	_
	A	T	C	G	T	_	A	_	C
0	1	2	3	4	5	5	6	6	7

- 相应的路径为 -

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

比对网格



0122345677

v= AT_GTTAT_

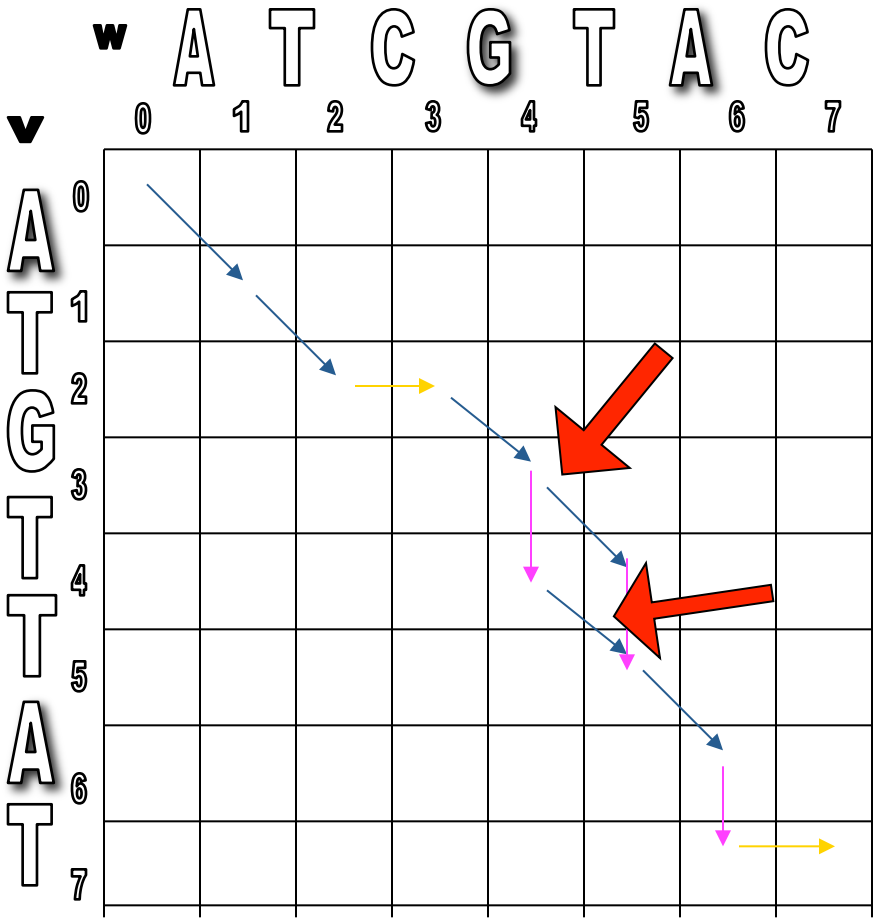
w= ATCGT_A_C

0123455667

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

Health Informatics Lab@USTC

比对网格



比对1

0122345677

v= AT_GTTAT_

w= ATCGT_A_C

0123455667

比对2

0122345677

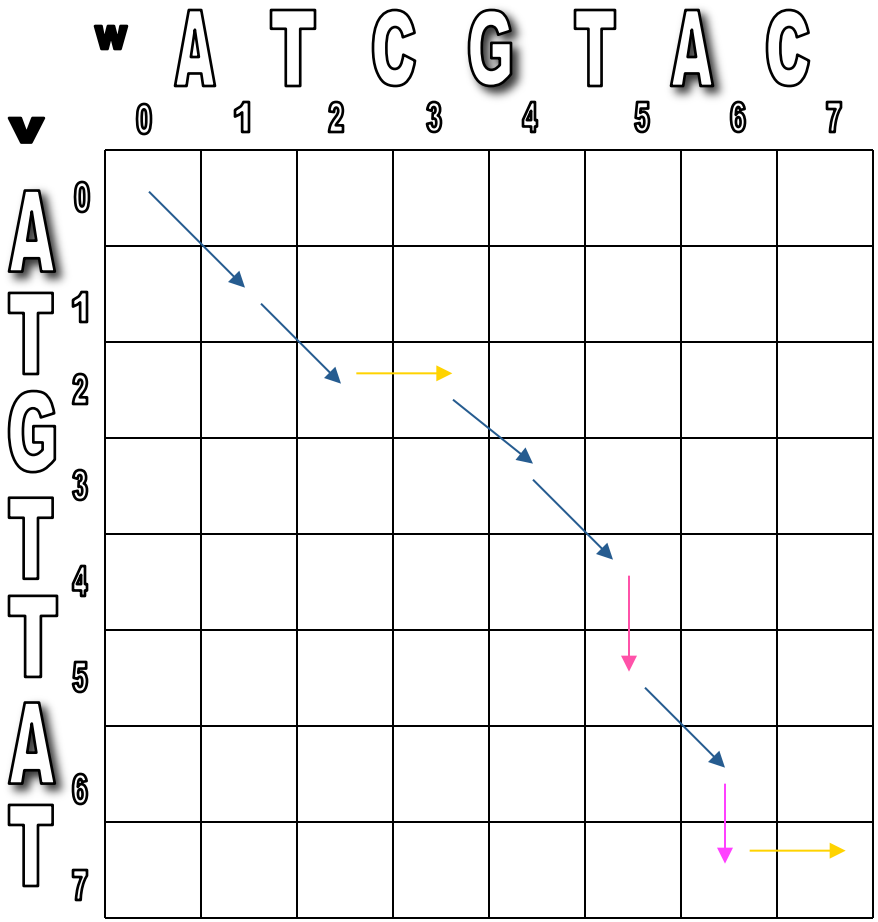
v= AT_GTTAT_

w= ATCG_TA_C

0123445667

Health Informatics Lab@USTC

比对打分



↓和→代表插删得分为0

↘代表匹配得分为1

图中路径所代表的比对得分为5

最长共同子序列

- 最长共同子序列(Longest Common Subsequence, LCS)
  - 对于两个序列 $v = v_1 v_2 \cdots v_m$ and $w = w_1 w_2 \cdots w_n$
  - v和w的LCS 对应于v的一个位置序列 $v: 1 \leq i_1 < i_2 < \cdots < i_t \leq m$
  - w的一个位置序列 $w: 1 \leq j_1 < j_2 < \cdots < j_t \leq n$
  - 使得$v(i_t)=w(j_t)$，且t最大

Health Informatics Lab@USTC

LCS举例

i coords:	0	1	2	2	3	3	4	5	6	7	8
elements of v		A	T	--	C	--	T	G	A	T	C
elements of 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)

LCS: 'TCTAC'

positions in v: 2 < 3 < 4 < 6 < 8

positions in w: 1 < 3 < 5 < 6 < 7

$$d(\mathbf{v},\mathbf{w}) = n + m - 2s(\mathbf{v},\mathbf{w})$$

Health Informatics Lab@USTC

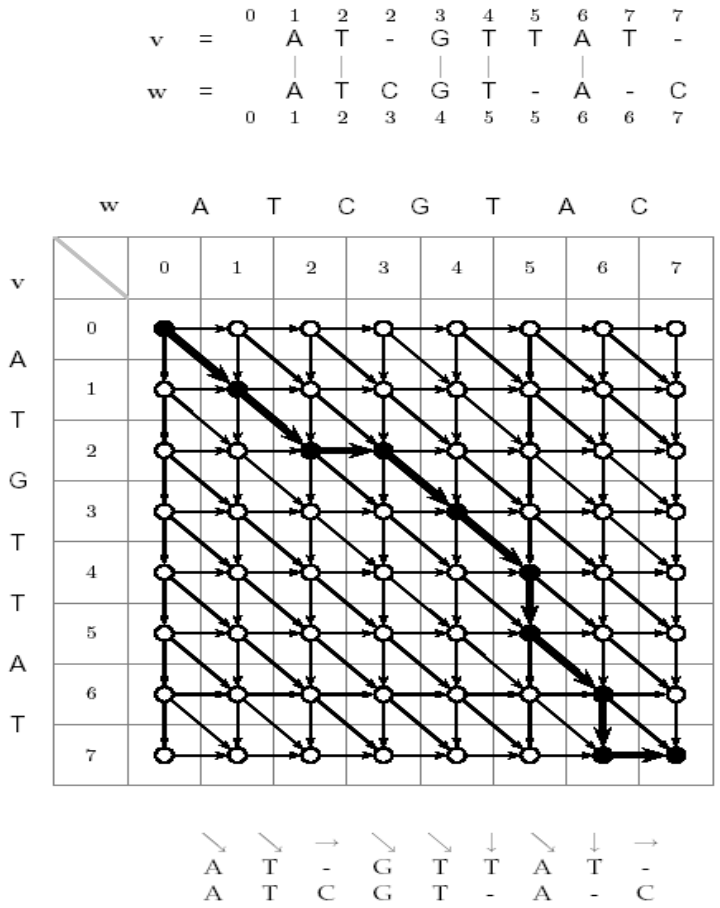
LCS问题的动态规划算法

Goal：寻找两个字符串的LCS

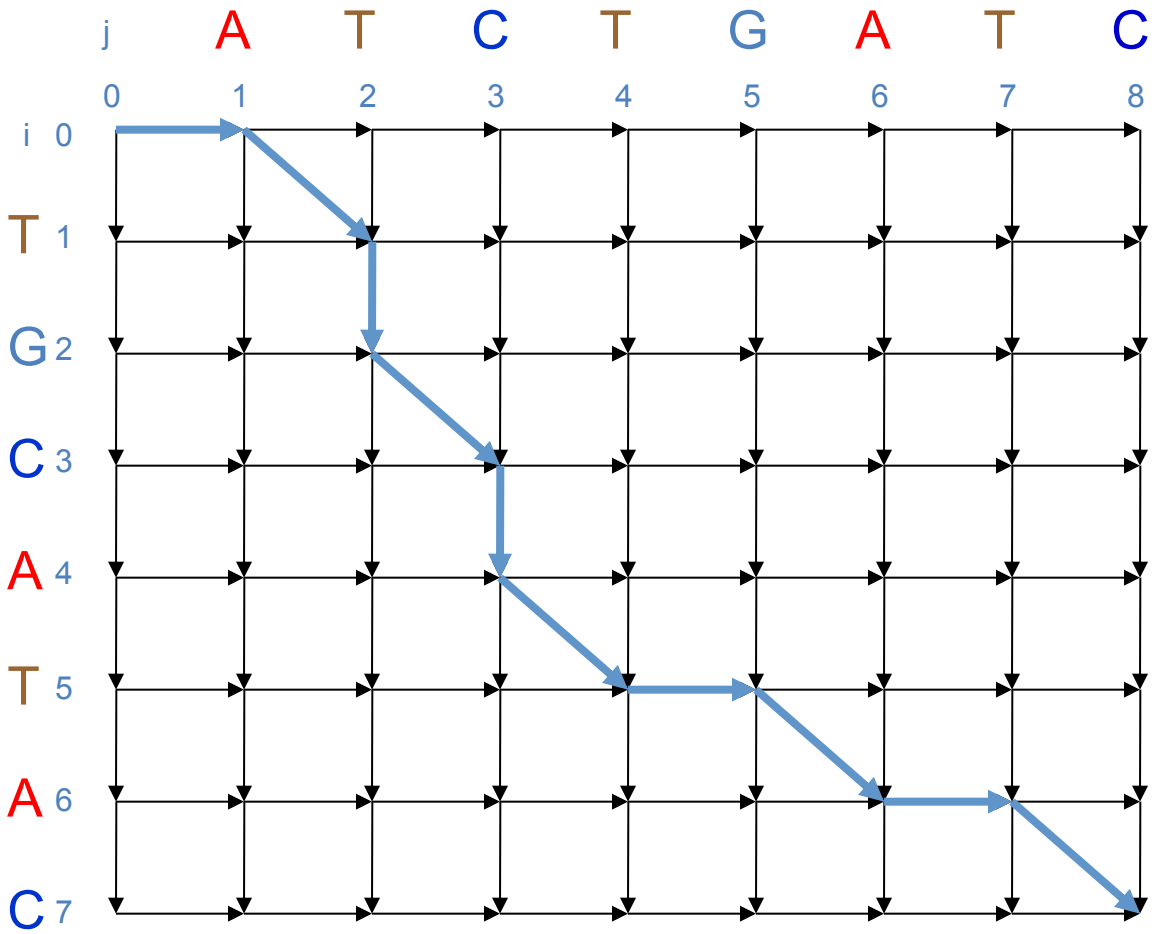
Input:带权DAG 包括“起点”和“终点”两个顶点

Output：带权DAG中包含从起点到终点的最长路径

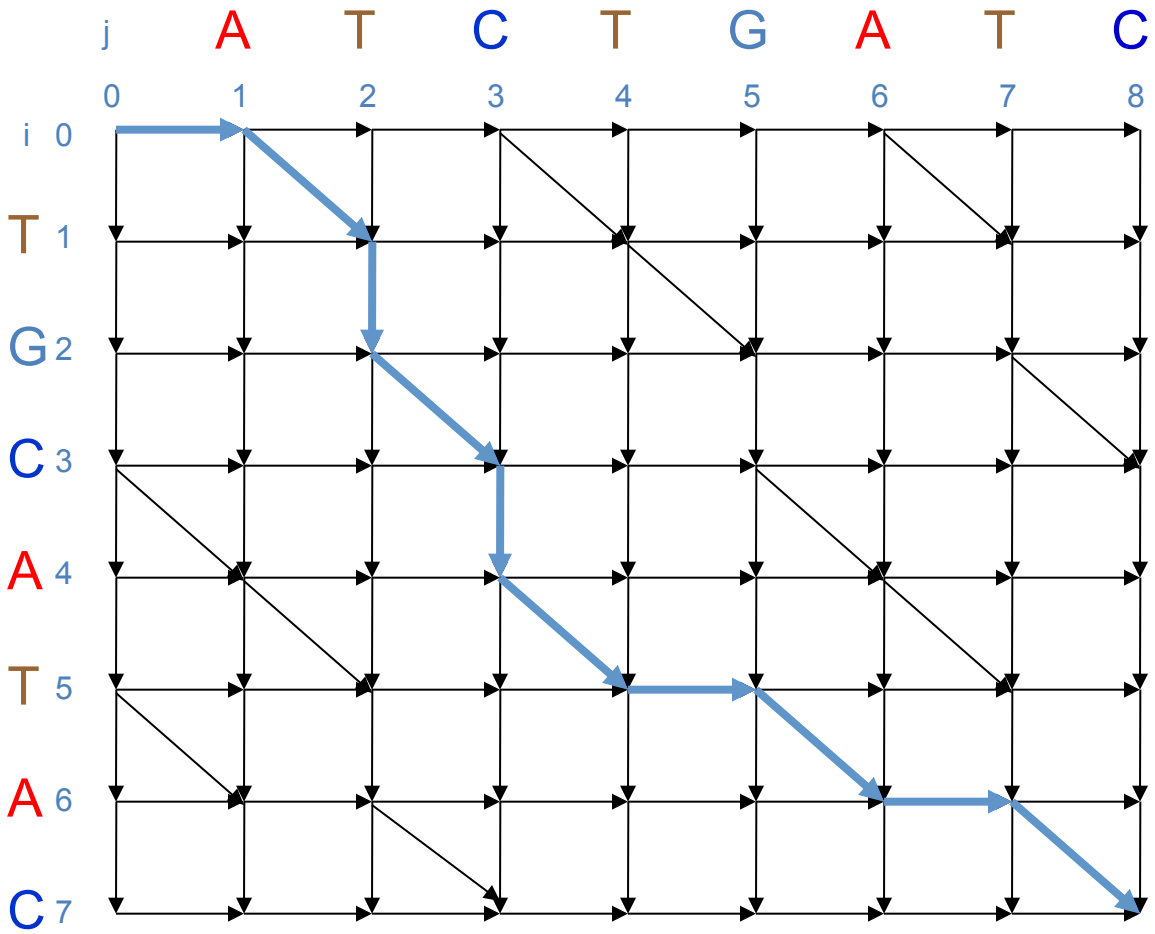
思路：利用编辑图(edit graph)求解



LCS问题与曼哈顿游客问题

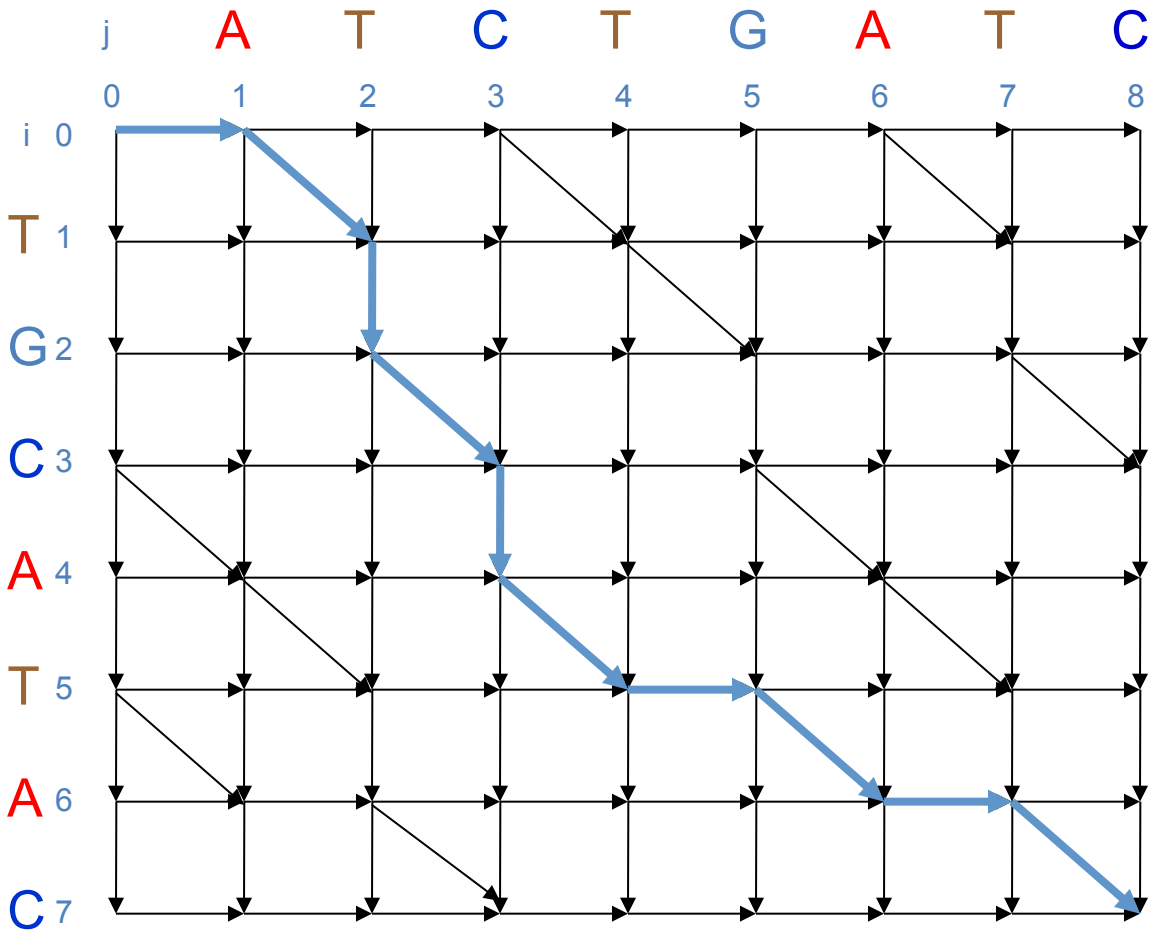


LCS问题的编辑图



Health Informatics Lab@USTC

LCS问题的编辑图



每条路径都有一个共同子序列

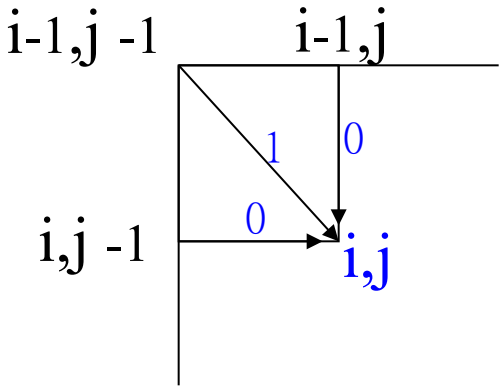
每经过一个对角边，共同子序列增加一个元素

LCS问题转化为寻找具有最多对角边的路径

LCS的动态规划算法

- LCS(v_i, w_j) 的长度计算公式如下：

$$s_{i,j} = \text{MAX} \begin{cases} s_{i-1,j} + 0 \\ s_{i,j-1} + 0 \\ s_{i-1,j-1} + 1, & \text{if } v_i = w_j \end{cases}$$



Health Informatics Lab@USTC

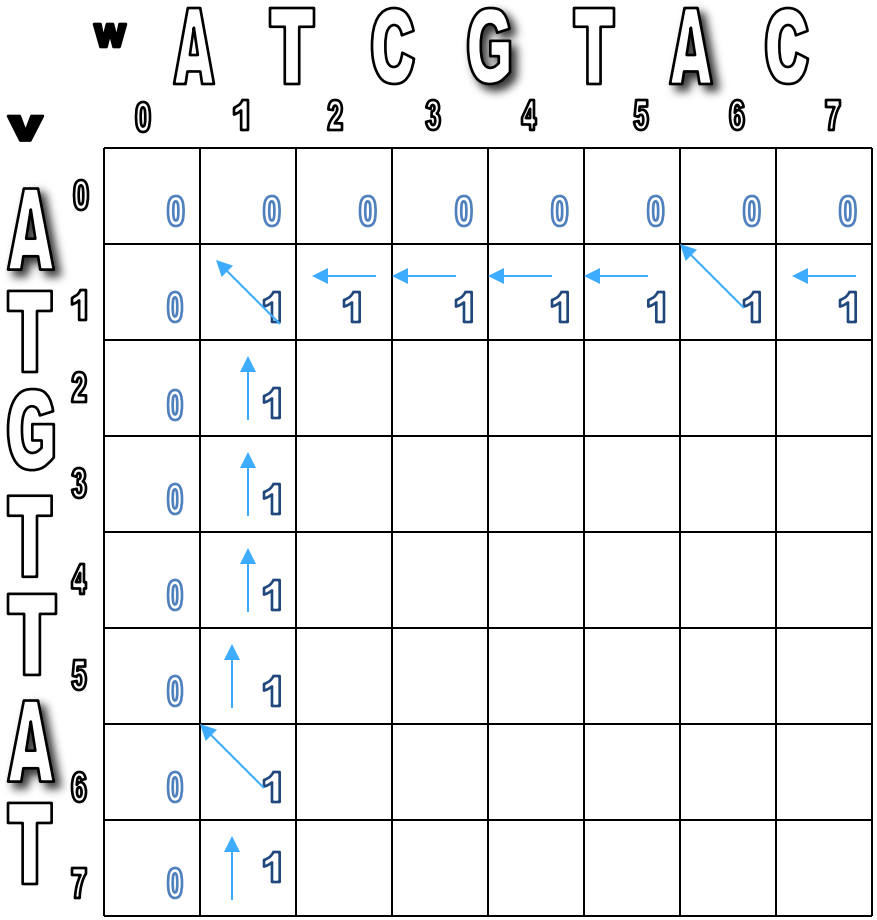
LCS的计算举例

v \ w		A	T	C	G	T	A	C
	0	1	2	3	4	5	6	7
A 0	0	0	0	0	0	0	0	0
T 1	0							
G 2	0							
T 3	0							
T 4	0							
A 5	0							
T 6	0							
7	0							

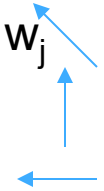
- 第一行第一列初始化为0，即$s_{0,j}=0$，$s_{i,0}=0$

Health Informatics Lab@USTC

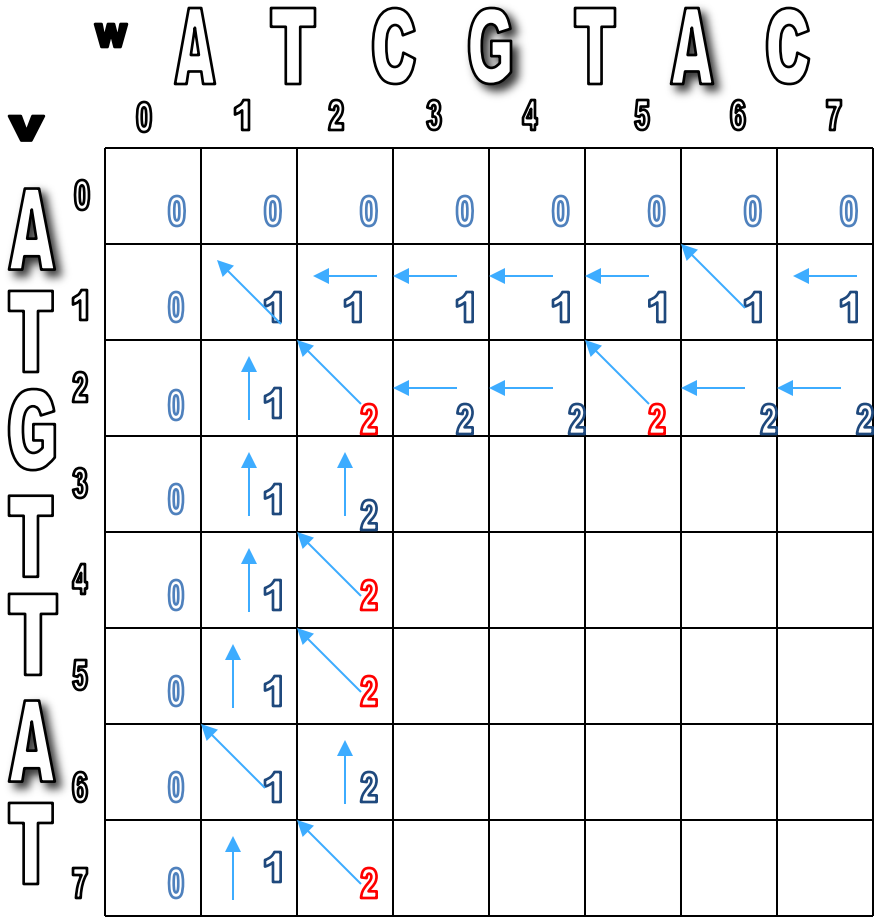
LCS的计算举例



$$S_{i,j} = \max \begin{cases} S_{i-1,\,j-1} & \text{value from NW +1, if } v_i = w_j \\ S_{i-1,\,j} & \text{value from North (top)} \\ S_{i,\,j-1} & \text{value from West (left)} \end{cases}$$



LCS的计算举例



寻找第二行和第二列的匹配

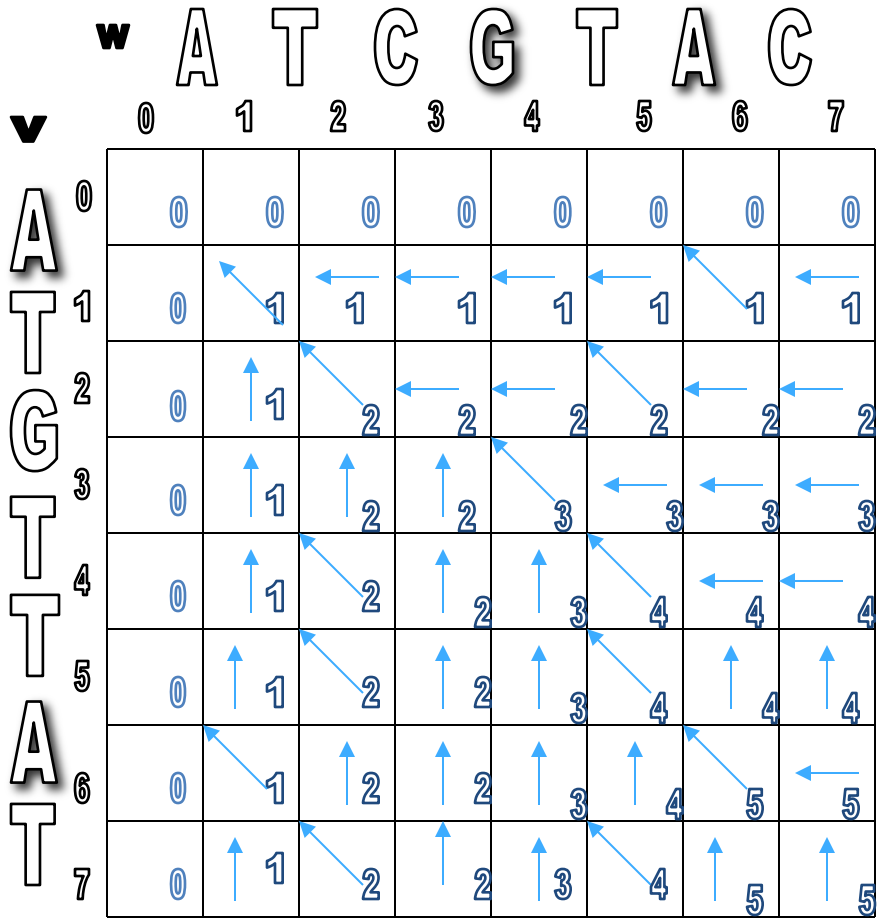
i=2, j=2, 5 为匹配(T).
j=2, i=4, 5, 7 为匹配 (T).

由于$v_i = w_j$, $s_{i,j} = s_{i-1,j-1} + 1$
因此有

$s_{2,2} = [s_{1,1} = 1] + 1$
$s_{2,5} = [s_{1,4} = 1] + 1$
$s_{4,2} = [s_{3,1} = 1] + 1$
$s_{5,2} = [s_{4,1} = 1] + 1$
$s_{7,2} = [s_{6,1} = 1] + 1$

Health Informatics Lab@USTC

LCS的计算举例



Health Informatics Lab@USTC

LCS 算法

1. LCS(v,w)
2. for $i \leftarrow 1$ to n
3. $s_{i,0} \leftarrow 0$
4. for j ← 1 to m
5. $s_{0,j} \leftarrow 0$
6. for $i \leftarrow 1$ to n
7. for $j \leftarrow 1$ to m
8. $$s_{i,j} \leftarrow \max \begin{cases} s_{i-1,j} \\ s_{i,j-1} \\ s_{i-1,j-1} + 1, \text{ if } v_i = w_j \end{cases}$$
9. $$b_{i,j} \leftarrow \begin{cases} \text{“}\uparrow\text{“} & \text{if } s_{i,j} = s_{i-1,j} \\ \text{“}\leftarrow\text{“} & \text{if } s_{i,j} = s_{i,j-1} \\ \text{“}\nwarrow\text{“} & \text{if } s_{i,j} = s_{i-1,j-1} + 1 \end{cases}$$
10. return $(s_{n,m}, b)$

Health Informatics Lab@USTC

Printing LCS: 回溯

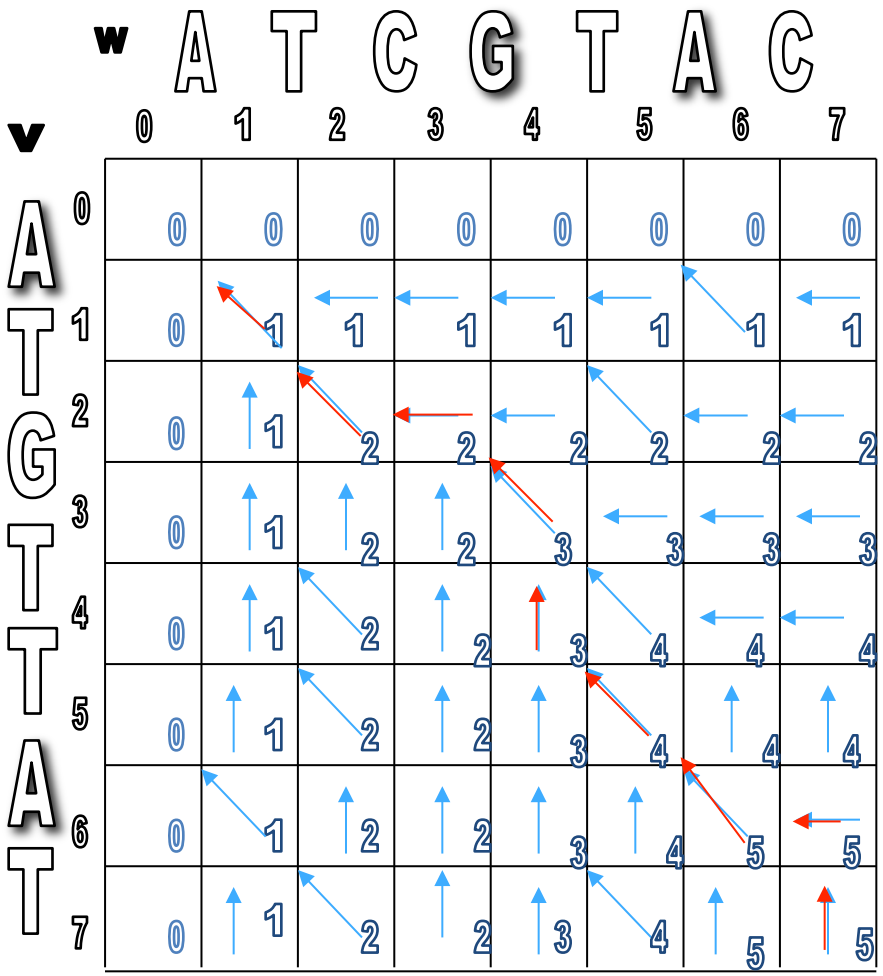
```
1.  PrintLCS(b,v,i,j)
2.      if  i = 0 or j = 0
3.          return
4.      if b_{i,j} = "↖"
5.          PrintLCS(b,v,i−1,j−1)
6.          print v_i
7.      else
8.          if b_{i,j} = "↑"
9.              PrintLCS(b,v,i−1,j)
10.         else
11.             PrintLCS(b,v,i,j−1)
```

Health Informatics Lab@USTC

LCS结果

- 编辑距离=?
- 能否用动态规划算法直接计算?
- $d_{i,0}=i$， $d_{0,j}=j$

$$d_{i,j}=\min\begin{cases} d_{i-1,j}+1 \\ d_{i,j-1}+1 \\ d_{i-1,j-1} \end{cases} \quad \text{if} \quad v_i=w_j$$



Health Informatics Lab@USTC

从LCS到比对: 改变打分方式

- 最长共同子序列问题的本质
  - 最简单的序列比对
  - 打分简单(0-1)，对插缺没有惩罚
  - 不考虑错配
- 更好的打分方式:
  - +1 ： 匹配
  - $-\mu$ ： 错配罚分
  - $-\sigma$ ： 插缺罚分
  - 最后分数为：#matches $-\ \mu$ (#mismatches) $-\ \sigma$ (#indels)

Health Informatics Lab@USTC

全局匹配问题

给定打分方式，寻找两个字符串之间的最佳比对

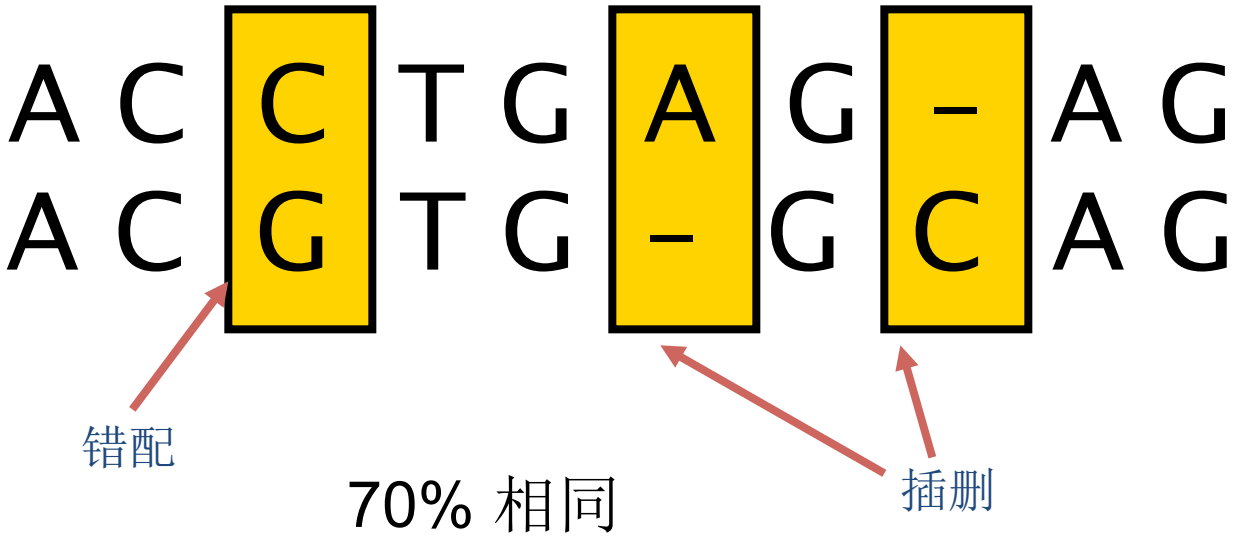
输入：字符串**v**和**w**以及打分方式
输出：最大比对得分所对应的比对结果

递归过程公式：

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

相似性度量

- 评价两条序列相似程度的指标
  - 基于序列中相同氨基酸的比例
  - 基于序列的保守性



序列的保守性

- 同源序列(Homologous Sequences)：
  - 定义：从某一共同祖先经进化而形成的不同序列
  - 进化过程中发生了多次突变：插入、删除、替代
  - 序列之间具有相似性，但非完美匹配

突变和替换

- DNA序列随机突变可能改变编码的蛋白质氨基酸序列
  - 若显著改变蛋白质结构→功能损害→蛋白质的适应性下降
  - 氨基酸替换出现的频率不一致
- 常见的氨基酸替换：
  - 极性：天门冬氨酸 → 谷氨酸
  - 非极性：丙氨酸 → 缬氨酸

打分矩阵(Scoring Matrix)

- 反映同源序列进化过程中由于突变产生差异的生物学依据
  - DNA序列比较的打分矩阵通常只由错配罚分与插缺罚分定义
  - 蛋白质序列比较的常用打分矩阵反映了在相关序列的进化中发生氨基酸替换的频率

打分矩阵举例

	A	R	N	K
A	5	-2	-1	-1
R	-	7	-1	3
N	-	-	7	0
K	-	-	-	6

AKRANR

KAAANK

-1 + (-1) + (-2) + 5 + 7 + 3 = 11

- 注意到尽管R和K是不同的氨基酸，但是他们之间的替换打分>0.
- R和K均为带正电的氨基酸→替换不会显著改变蛋白质的功能.

Health Informatics Lab@USTC

打分矩阵的数学描述

考虑一对长度分别为m和n的序列x和y，x_i为x的第i个字符，y_j为y的第j个字符

- 无关或随机模型(假设字母a以频率q_a独立地出现)：

$$R(x,y\mid R)=\prod_i q_{x_i}\prod_j q_{y_j}$$

- 匹配模型（比对的残基对以联合概率p_{ab}出现）：

$$R(x,y\mid M)=\prod_i p_{x_i y_i}$$

打分矩阵的数学描述

- 两种模型的似然比（odds ratio）：

$$\frac{R(x,y|M)}{R(x,y|R)} = \frac{\prod_i p_{x_i y_i}}{\prod_i q_{x_i} \prod_i q_{y_i}} = \prod_i \frac{p_{x_i y_i}}{q_{x_i} q_{y_i}}$$

- 对数几率比（log-odds ratio）：

$$S = \sum_i s(x_i, y_i), \quad s(a,b) = \log(\frac{p_{ab}}{q_a q_b})$$

Health Informatics Lab@USTC

打分矩阵的数学描述

- 如何估计比对中符合出现的概率？
  - 在已知比对中统计比对符合(氨基酸/空位）出现的频率
  - 最大似然估计

- 但是有两个问题：
  - 如何获得比对数据并对其进行采样?
  - 不同序列间分化程度不同

PAM打分矩阵

- 可接受点突变（Point Accepted Mutation）
  - 基于进化的点突变模型，如果两种氨基酸替换频繁，说明自然界接受这种替换，那么这对氨基酸替换得分就高。一个PAM(即PAM1)就是一个进化的变异单位，即1%的氨基酸改变
  - 但这并不意味100次PAM后，每个氨基酸都发生变化，因为其中一些位置可能会经过多次突变，甚至可能会变回到原来的氨基酸
  - PAM矩阵是从蛋白质序列的全局比对结果推导出来的

PAM_X

- $PAM_x = PAM_1^x$
  - $PAM_{250} = PAM_1^{250}$
- 常用的PAM_{250} 打分矩阵：

	Ala A	Arg R	Asn N	Asp D	Cys C	Gln Q	Glu E	Gly G	His H	Ile I	Leu L	Lys K	...
Ala A	13	6	9	9	5	8	9	12	6	8	6	7	...
Arg R	3	17	4	3	2	5	3	2	6	3	2	9	
Asn N	4	4	6	7	2	5	6	4	6	3	2	5	
Asp D	5	4	8	11	1	7	10	5	6	3	2	5	
Cys C	2	1	1	1	52	1	1	2	2	2	1	1	
Gln Q	3	5	5	6	1	10	7	3	7	2	3	5	
...													
Trp W	0	2	0	0	0	0	0	0	1	0	1	0	
Tyr Y	1	1	2	1	3	1	1	1	3	2	2	1	
Val V	7	4	4	4	4	4	4	4	5	4	15	10	

Health Informatics Lab@USTC

BLOSUM矩阵

- 模块替代矩阵（Blocks Substitution Matrix）
  - BLOSUM矩阵则是从蛋白质序列块（短序列）比对而推导出来的
- 矩阵名代表进化距离
  - BLOSUM-62矩阵适于用来比较大约具有62%相似度的序列，而BLOSUM-80矩阵更适合于相似度为80%左右的序列

BLOSUM矩阵

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	5	-2	-1	-2	-1	-1	-1	0	-2	-1	-2	-1	-1	-3	-1	1	0	-3	-2	0	-2	-1	-1	-5
R	-2	7	-1	-2	-4	1	0	-3	0	-4	-3	3	-2	-3	-3	-1	-1	-3	-1	-3	-1	0	-1	-5
N	-1	-1	7	2	-2	0	0	0	1	-3	-4	0	-2	-4	-2	1	0	-4	-2	-3	4	0	-1	-5
D	-2	-2	2	8	-4	0	2	-1	-1	-4	-4	-1	-4	-5	-1	0	-1	-5	-3	-4	5	1	-1	-5
C	-1	-4	-2	-4	13	-3	-3	-3	-3	-2	-2	-3	-2	-2	-4	-1	-1	-5	-3	-1	-3	-3	-2	-5
Q	-1	1	0	0	-3	7	2	-2	1	-3	-2	2	0	-4	-1	0	-1	-1	-1	-3	0	4	-1	-5
E	-1	0	0	2	-3	2	6	-3	0	-4	-3	1	-2	-3	-1	-1	-1	-3	-2	-3	1	5	-1	-5
G	0	-3	0	-1	-3	-2	-3	8	-2	-4	-4	-2	-3	-4	-2	0	-2	-3	-3	-4	-1	-2	-2	-5
H	-2	0	1	-1	-3	1	0	-2	10	-4	-3	0	-1	-1	-2	-1	-2	-3	2	-4	0	0	-1	-5
I	-1	-4	-3	-4	-2	-3	-4	-4	-4	5	2	-3	2	0	-3	-3	-1	-3	-1	4	-4	-3	-1	-5
L	-2	-3	-4	-4	-2	-2	-3	-4	-3	2	5	-3	3	1	-4	-3	-1	-2	-1	1	-4	-3	-1	-5
K	-1	3	0	-1	-3	2	1	-2	0	-3	-3	6	-2	-4	-1	0	-1	-3	-2	-3	0	1	-1	-5
M	-1	-2	-2	-4	-2	0	-2	-3	-1	2	3	-2	7	0	-3	-2	-1	-1	0	1	-3	-1	-1	-5
F	-3	-3	-4	-5	-2	-4	-3	-4	-1	0	1	-4	0	8	-4	-3	-2	1	4	-1	-4	-4	-2	-5
P	-1	-3	-2	-1	-4	-1	-1	-2	-2	-3	-4	-1	-3	-4	10	-1	-1	-4	-3	-3	-2	-1	-2	-5
S	1	-1	1	0	-1	0	-1	0	-1	-3	-3	0	-2	-3	-1	5	2	-4	-2	-2	0	0	-1	-5
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	2	5	-3	-2	0	0	-1	0	-5
W	-3	-3	-4	-5	-5	-1	-3	-3	-3	-3	-2	-3	-1	1	-4	-4	-3	15	2	-3	-5	-2	-3	-5
Y	-2	-1	-2	-3	-3	-1	-2	-3	2	-1	-1	-2	0	4	-3	-2	-2	2	8	-1	-3	-2	-1	-5
V	0	-3	-3	-4	-1	-3	-3	-4	-4	4	1	-3	1	-1	-3	-2	0	-3	-1	5	-4	-3	-1	-5
B	-2	-1	4	5	-3	0	1	-1	0	-4	-4	0	-3	-4	-2	0	0	-5	-3	-4	5	2	-1	-5
Z	-1	0	0	1	-3	4	5	-2	0	-3	-3	1	-1	-4	-1	0	-1	-2	-2	-3	2	5	-1	-5
X	-1	-1	-1	-1	-2	-1	-1	-2	-1	-1	-1	-1	-1	-1	-2	-2	-1	0	-3	-1	-1	-1	-1	-5
*	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	1

BLOSUM50打分矩阵

局部比对的生物学意义

- 两个不同物种的基因可能仅在部分保守区域相似，而在其他部分缺乏相似性
- 同源异型框(homeobox)基因
  - 存在一种叫做同源异型域(homeodomain)的高度保守区域
  - 同源异型框基因在不同物种中有很大差异
  - 全局比对算法找不到同源异型域的保守区段

局部比对举例

- 全局比对

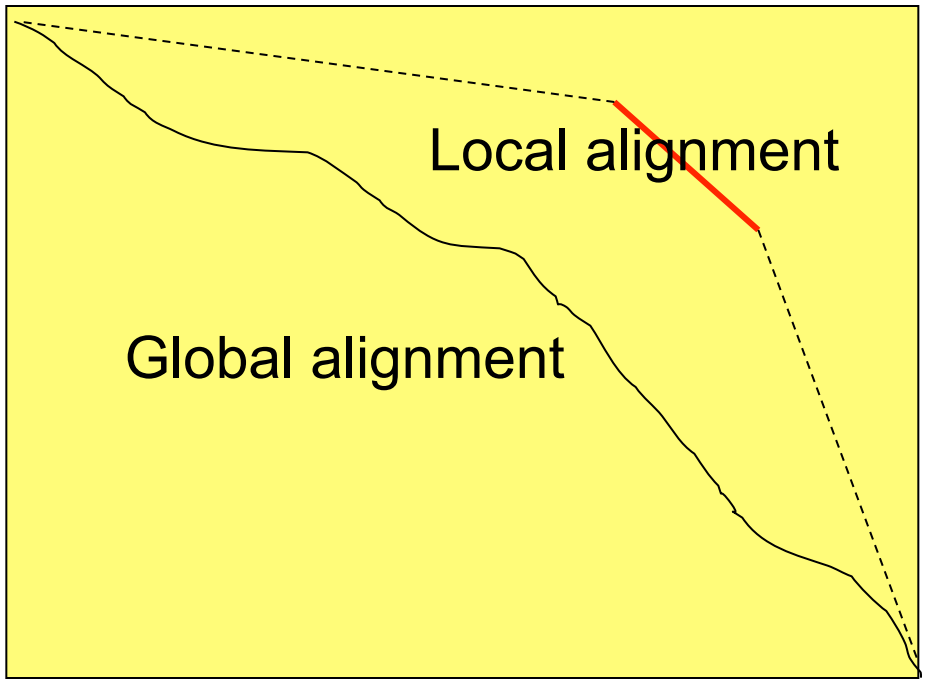
```
--T--CC-C-AGT--TATGT-CAGGGGACACG-A-GCATGCAGA-GAC
  |   ||  |   ||   | | | |||     || | | |   | ||||    |
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG-T-CAGAT--C
```

- 局部比对-找出保守区域的效果更好

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

- 局部比对得分>全局比对得分!

局部比对举例



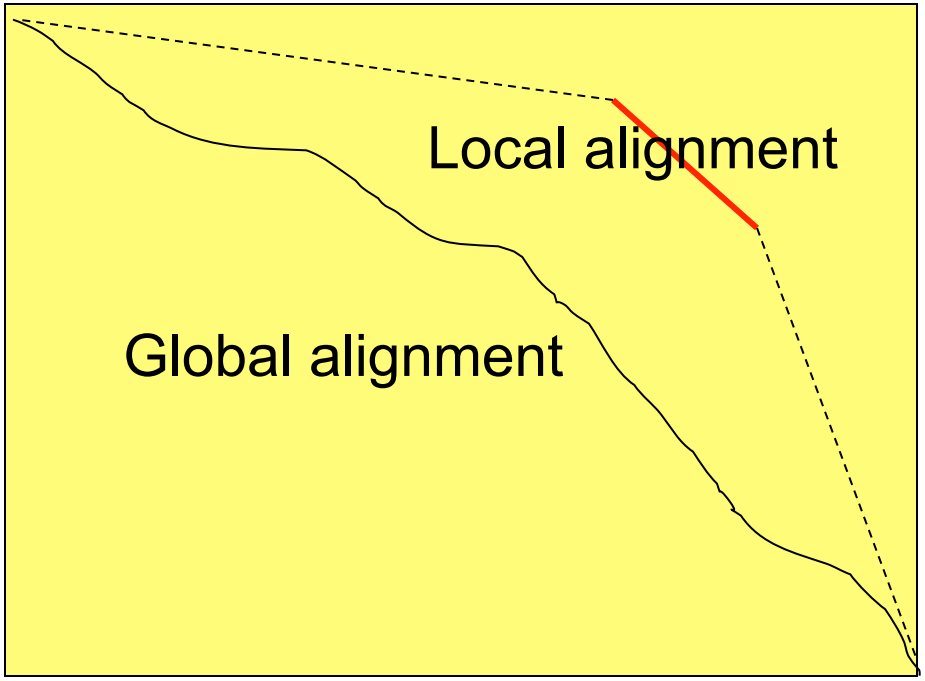
局部比对问题

- 目标：找到两条序列中最好局部比对
- 输入：序列v，w和打分矩阵 δ
- 输出:满足下述条件的v与w的子字符串：在 δ 下的全局比对得分是v与w的所有子字符串的全局比对得分为最大者。

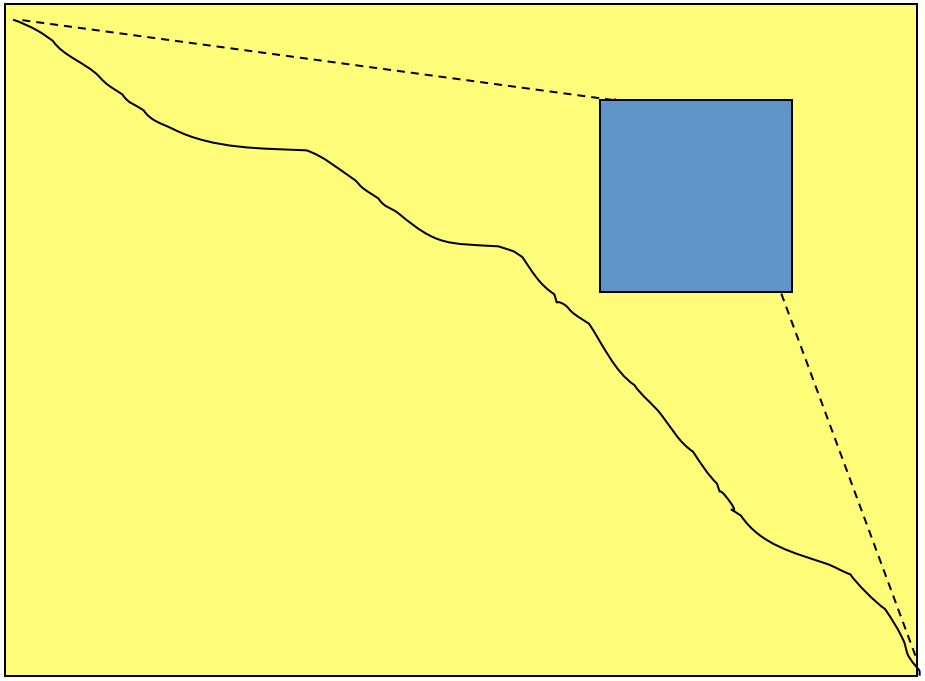
局部比对的困难

- 全局比对问题求解编辑图中位于顶点 $(0, 0)$ 和 (n, m) 之间的最长路径
- 局部比对问题求解编辑图中任意顶点 (i, j) 和 (i', j') 之间的所有路径中寻找一条最长路径
  - 遍历算法：每一对顶点之间的最长路径，然后选择其中最长的一条路

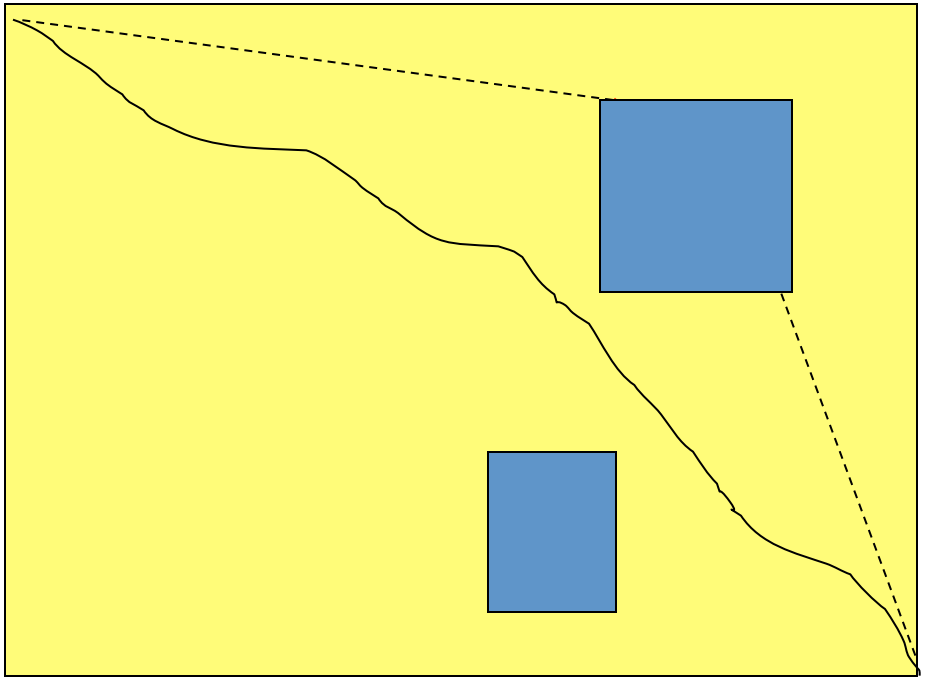
局部比对的遍历算法



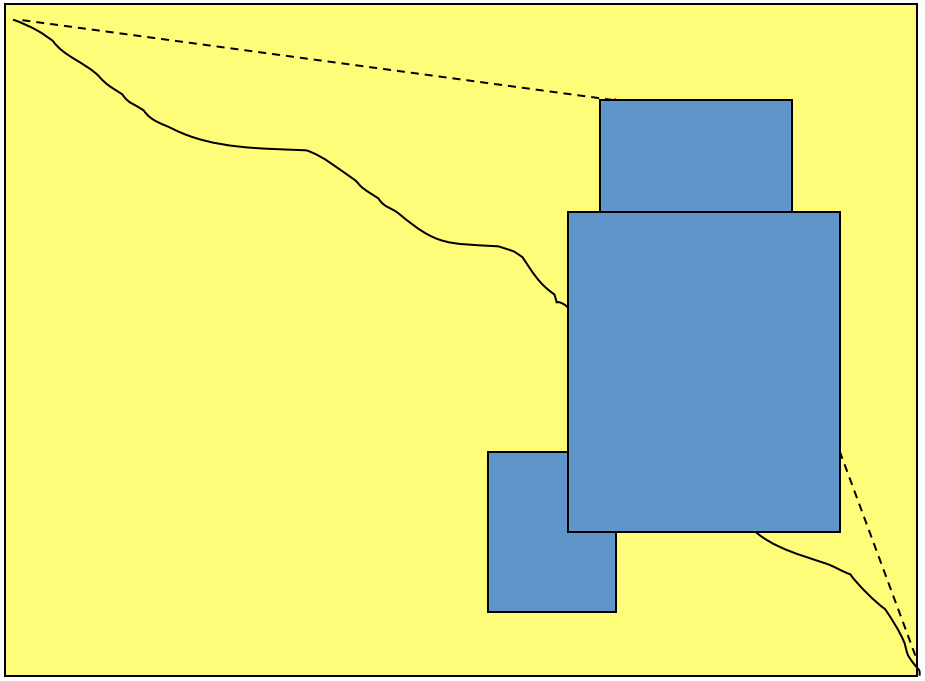
局部比对的遍历算法



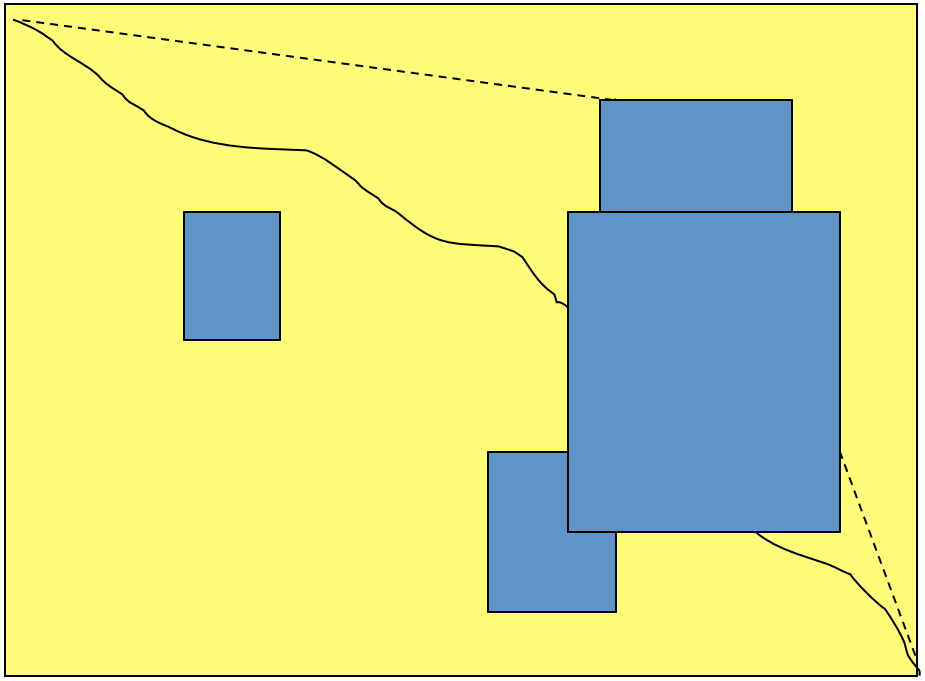
局部比对的遍历算法



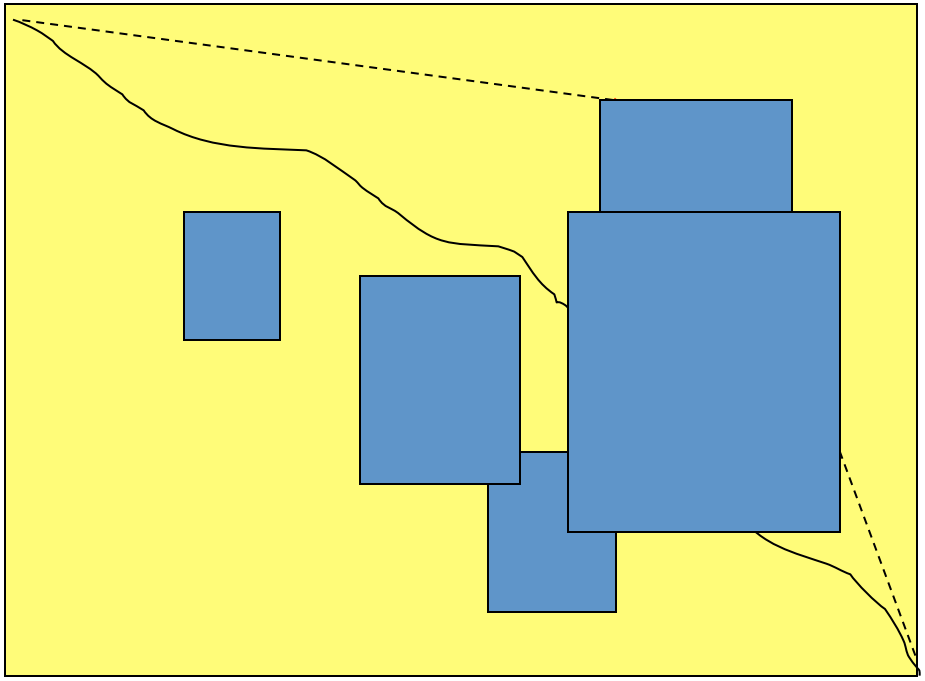
局部比对的遍历算法



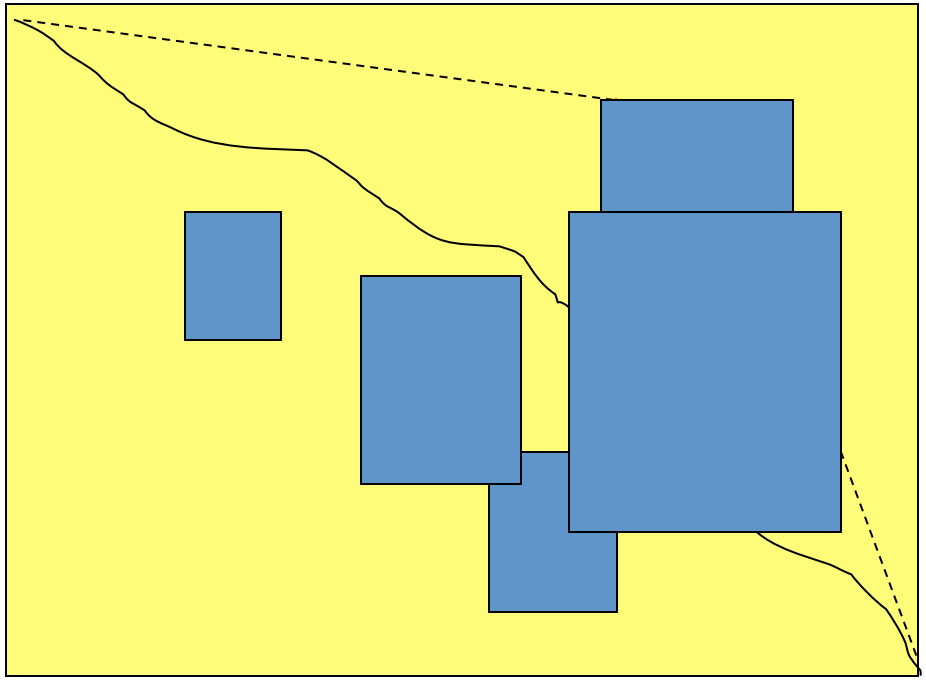
局部比对的遍历算法



局部比对的遍历算法



局部比对的遍历算法



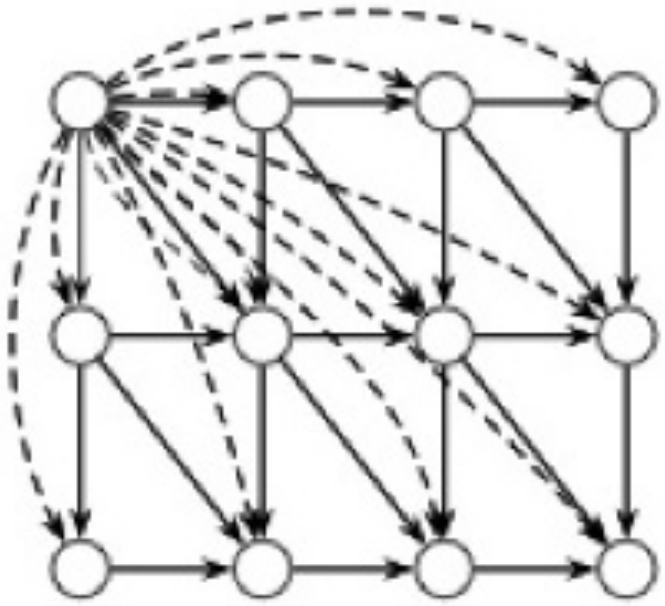
- 算法的复杂度为 $O(n^4)$
  -

Health Informatics Lab@USTC

局部比对的解决方案 (Smith-Waterman算法)

- 通过在编辑图上加权值为0的边来寻找一条从源顶点(0，0)到其他每一个顶点的最长路径

$$s_{i,j} = max \begin{cases} 0 \\ s_{i-1,j-1} + \delta(v_i, w_j) \\ s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \end{cases}$$



局部比对算法的分析

1）选项0对应于开始一个新比对

- 如果达到某一位置的最优比对出现负分，那么最好开始一个新的比对而不是继续延伸

2）比对可以终止于矩阵中任意位置

- 不再以矩阵右下角元素值F(n,m)作为最好分值而是从整个矩阵中寻找F(i,j)的最高值

3）随机比对的期望分值必须为负

- 一个真实的子序列比对很可能因为长度的原因被一个更长但错误的比对所掩盖

4）替换矩阵中必须存在某些大于0的s(a，b)

- 该算法将根本无法找到任何比对

期望分值为负的解释

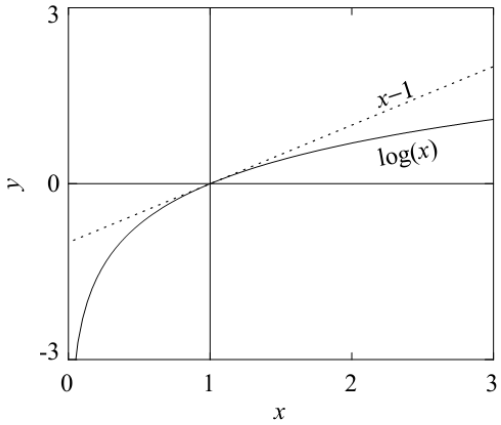
- 考虑无空位的情况下：

$$\sum_{a,b} q_a q_b s(a,b) = -\sum_{a,b} q_a q_b \log \frac{q_a q_b}{p_{ab}}$$

- 相对熵：

$$H(P \| Q) = \sum_i P(x_i) \log \frac{P(x_i)}{Q(x_i)}$$

  - 相对熵总是大于等于0



Health Informatics Lab@USTC

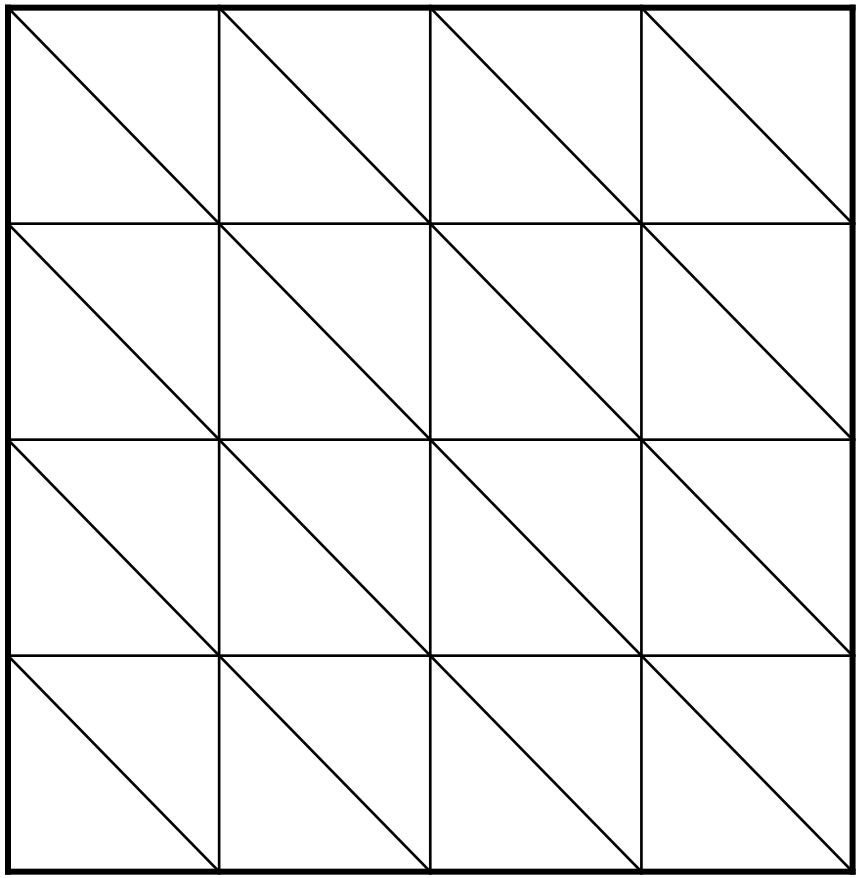
仿射缺口罚分（Affine Gap Penalties)

- 突变常常是由DNA复制过程中出错引起的，经常删除或插入整个子序列
- 缺口(gap)：就是比对中某一行序列中的一段连续空格
- 传统打分方法：
  - 长度为1的缺口，罚分为-1 σ
  - 长度为2的缺口，罚分为-2 σ
  - …
  - 长度为100的缺口，罚分为-100 σ！

仿射缺口罚分策略

- 对于长度为x的缺口罚分为：$-(\rho+\sigma x)$
  - $\rho>0$ 引入缺口的罚分
  - $\sigma>0$ 缺口中每一个字符的罚分
- σ通常要小于ρ，所以能降低对缺口的整体罚分

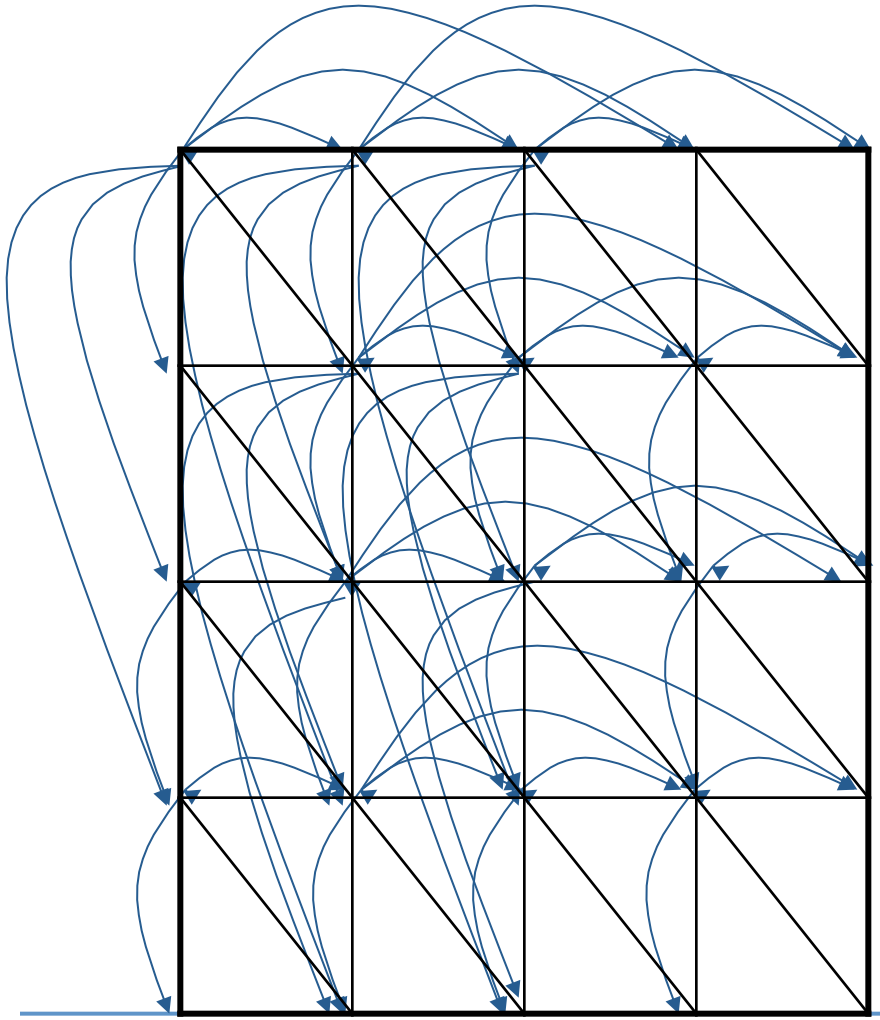
体现仿射缺口罚分的编辑图



为了在编辑图中反映仿射缺口罚分，从局部比对的思路出发，在编辑图中增加垂直或平行的边（长度为x），相应的罚分为：

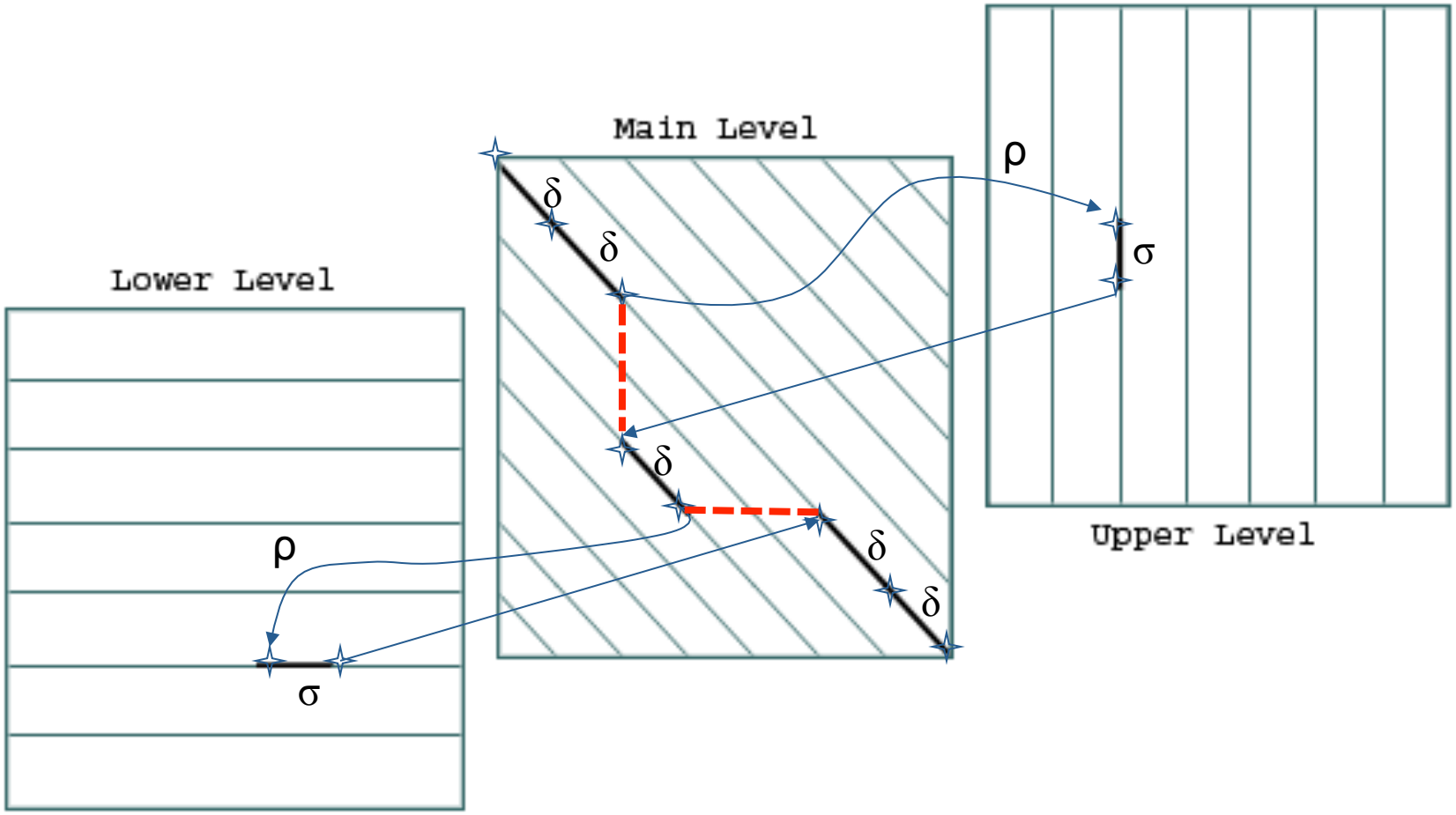
$$-\rho - x * \sigma$$

体现仿射缺口罚分的编辑图



如果通过这样的添加方式，那么算法的运行复杂度将会从 $O(n^2)$ 增加到 $O(n^3)$

编辑图分拆



动态规划的递归公式

上层：

$$\overset{\downarrow}{s}_{i,j} = \max \begin{cases} \overset{\downarrow}{s}_{i-1,j} - \sigma \\ s_{i-1,j} - (\rho+\sigma) \end{cases}$$

w中空位继续增加(删除)

w中开始一个空位(删除)←中层

下层：

$$\vec{s}_{i,j} = \max \begin{cases} \vec{s}_{i,j-1} - \sigma \\ s_{i,j-1} - (\rho+\sigma) \end{cases}$$

v中空位继续增加(插入)

v中开始一个空位(插入)←中层

中层：

$$s_{i,j} = \max \begin{cases} s_{i-1,j-1} + \delta(v_i, w_j) \\ \overset{\downarrow}{s}_{i,j} \\ \vec{s}_{i,j} \end{cases}$$

匹配或错配

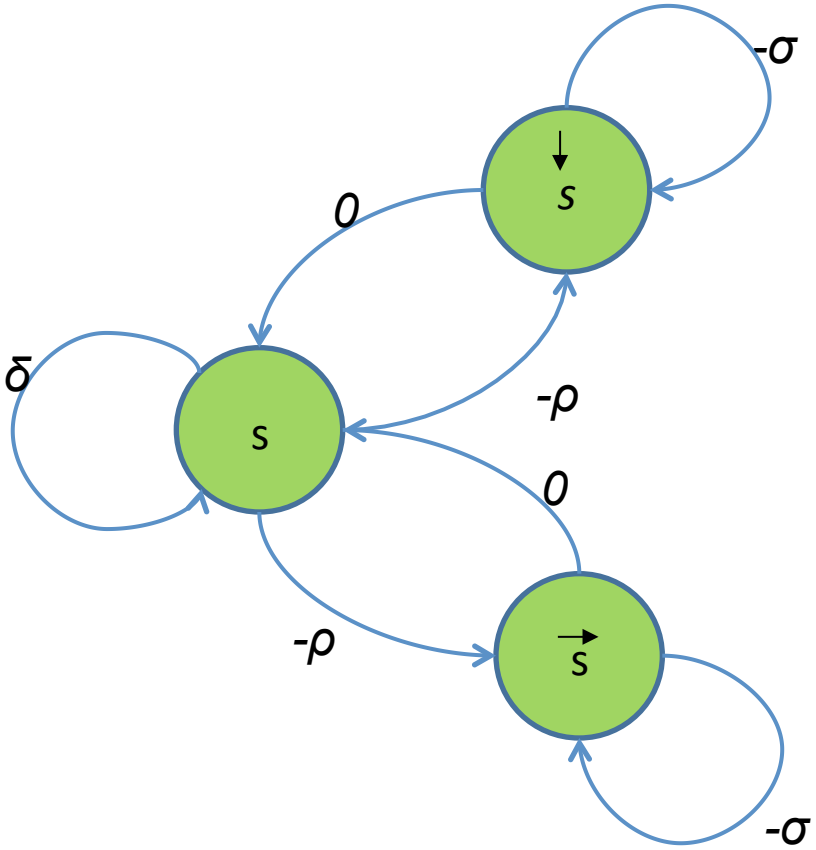
缺口结束(删除)←上层

缺口结束(插入)←下层

Health Informatics Lab@USTC

方程的抽象图形化描述

- 每种矩阵值都被显示为一个状态（上、中、下）
- 箭头表示状态间的转移
- 每个状态变量的新值就是到达这个状态的转移对应的最大分值



- 有限状态自动机(FSA)

总结

- 动态规划算法是序列比对的理论基础
- 合适的打分函数非常重要

- 序列比对中的扩展问题
  - 比对分值的统计显著性
  - 数据库搜索的比对算法
  - 多序列比对
  - …

比对分值的统计显著性

- 比对分值的统计显著性
  - 比对算法确定最优比对，但如何评估分值的显著性?
  - 如何判断如下假设检验:
    - H_0: 序列x和y是两条无关序列，不存在同源性
- 两种解决方案:
  - 贝叶斯方法: 模型比较
  - 经典方法: 极值分布

贝叶斯方法

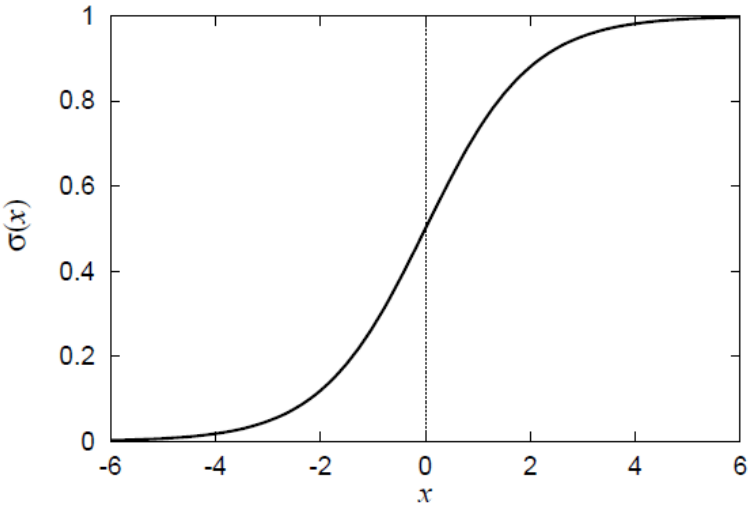
$$P(M \mid x, y) = \frac{P(x, y \mid M)P(M)}{P(x, y)}$$

$$= \frac{P(x, y \mid M)P(M)}{P(x, y \mid M)P(M) + P(x, y \mid R)P(R)}$$

$$= \frac{P(x, y \mid M)P(M) / P(x, y \mid R)P(R)}{1 + P(x, y \mid M)P(M) / P(x, y \mid R)P(R)}$$

$$= \frac{e^{S'}}{1 + e^{S'}} = \sigma(s'),$$

$$s' = \log(\frac{P(x, y \mid M)}{P(x, y \mid R)}) + \log(\frac{P(M)}{P(R)})$$



极值分布

- 经典方法：极值分布

$$P(M_N \le x) \approx \exp(-KNe^{\lambda(x-u)})$$

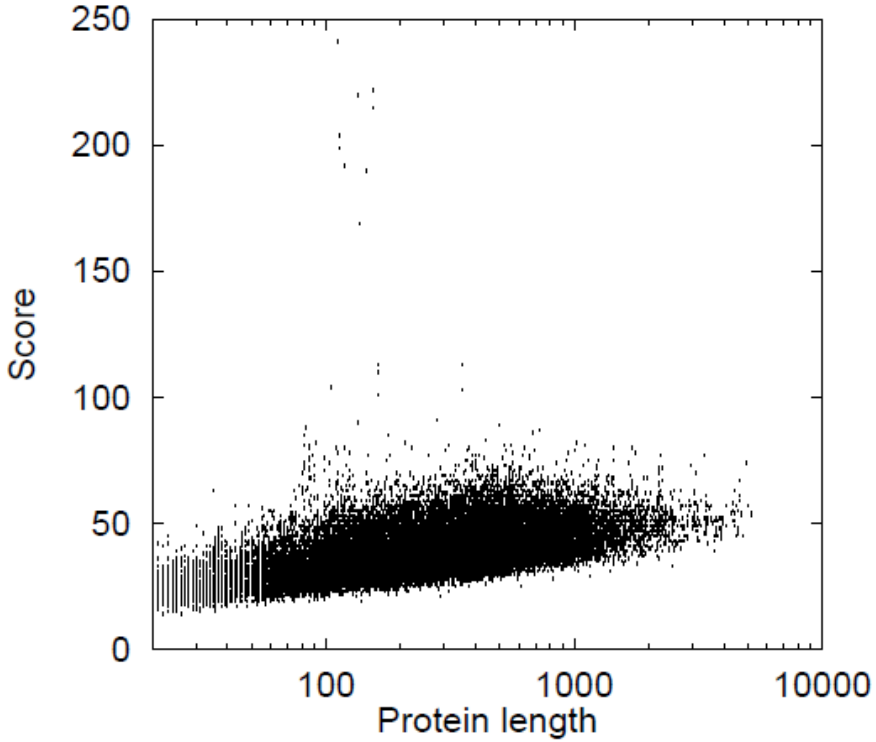
- 局部比对：

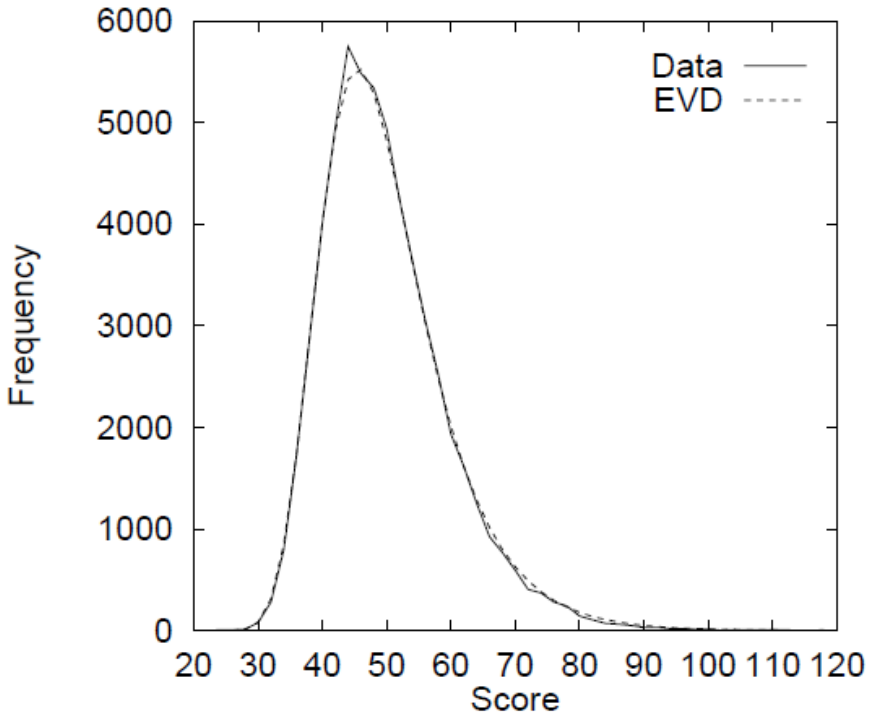
$$E(S) = Kmne^{-\lambda S}$$

$$P(x > S) = 1 - e^{-E(S)}$$

$$S > T + \frac{\log mn}{\lambda}$$

长度修正


Score
Protein length
0
50
100
150
200
250
100
1000
10000

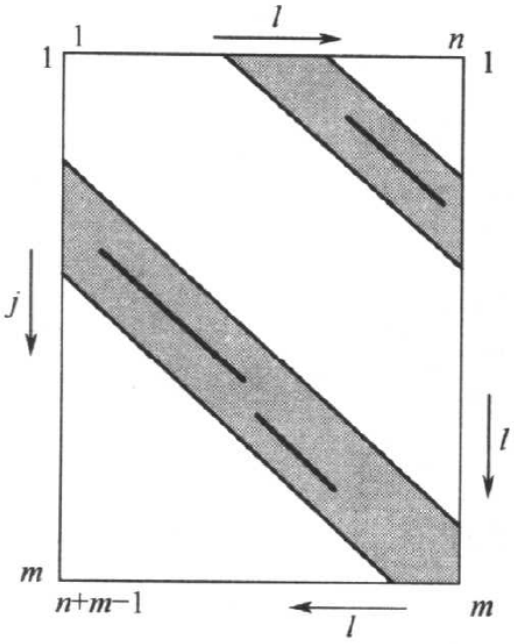

Frequency
Score
Data
EVD
0
1000
2000
3000
4000
5000
6000
20
30
40
50
60
70
80
90
100
110
120

数据库搜索的比对算法

- 数据库搜索的比对算法
- 核酸或蛋白质序列数据库中寻找出与待检序列具有一定相似程度的目标序列
- 动态规划算法复杂度高：O(mn)
- 近似算法：在考虑所有高分比对的同时，搜索动态规划矩阵单元中尽可能小的部分

数据库搜索的比对算法

- FASTA算法
  - 在对角线周围限定了动态规划算法需要搜索的一个区域,从而缩小了动态规划算法所需的计算时间
  - 只搜索很短一段相同的序列片段，称为K元组(k-tuple)，然后将位于同一对角线上的小片段连接成相似性的长片段

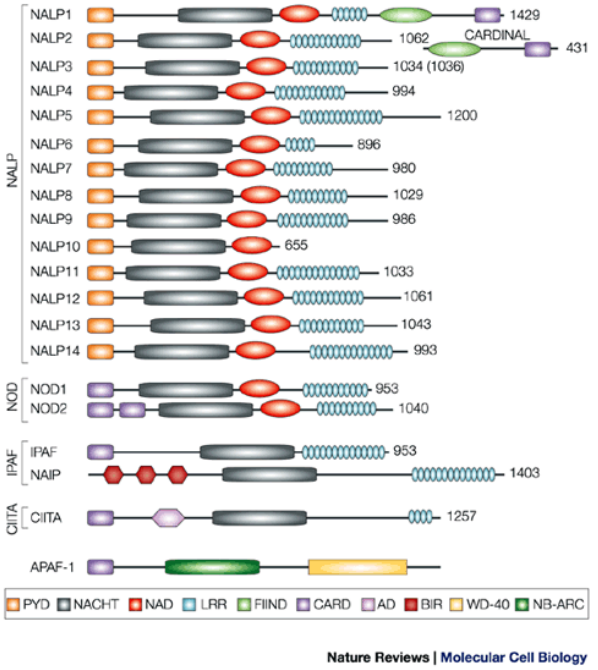


数据库搜索的比对算法

- BLAST算法
  - 真实匹配很有可能包含一小段等同位点或分值极高的短连续片段对(segment pair)
  - 建立包含长度为W的词和相似词的列表，被称为高分值片段对（HSP）
  - 寻找最好的HSP，动态规划算法对此片段进行两边延伸

蛋白质家族

- 蛋白质家族
  - 亚家族：由几乎完全相同的序列组成
  - 家族：由功能上非常相似，而氨基酸序列有50%以上相同残基的序列组成
  - 超家族：序列相似性不是基于氨基酸的完全相同，而是根据PAM计分矩阵
- 序列模体往往是蛋白质分子的重要功能位点所在



Nature Reviews | Molecular Cell Biology

多序列比对的意义

- 对于蛋白质家族的成组序列，必须进行多序列比对才能揭示家族的特性
- 多序列比对有着广泛的应用
  - 蛋白质家族中的保守区域（模体）
  - 蛋白质的聚类、分类
  - 点突变检测
  - 推断进化关系 和构建系统发育树
  - 预测蛋白质结构

多序列比对举例

- 两条序列比对可以用一个二维矩阵表示
- 三条序列比对则可以用一个三维矩阵表征

	A	--	T	G	C

	A	A	T	--	C

	--	A	T	G	C

比对的路径

0	1	1	2	3	4
	A	--	T	G	C

	A	A	T	--	C

	--	A	T	G	C

x坐标

比对的路径

0	1	1	2	3	4
	A	--	T	G	C

x坐标

0	1	2	3	3	4
	A	A	T	--	C

y坐标

	--	A	T	G	C

比对的路径

0	1	1	2	3	4
	A	--	T	G	C
0	1	2	3	3	4
	A	A	T	--	C
0	0	1	2	3	4
	--	A	T	G	C

x坐标

y坐标

z坐标

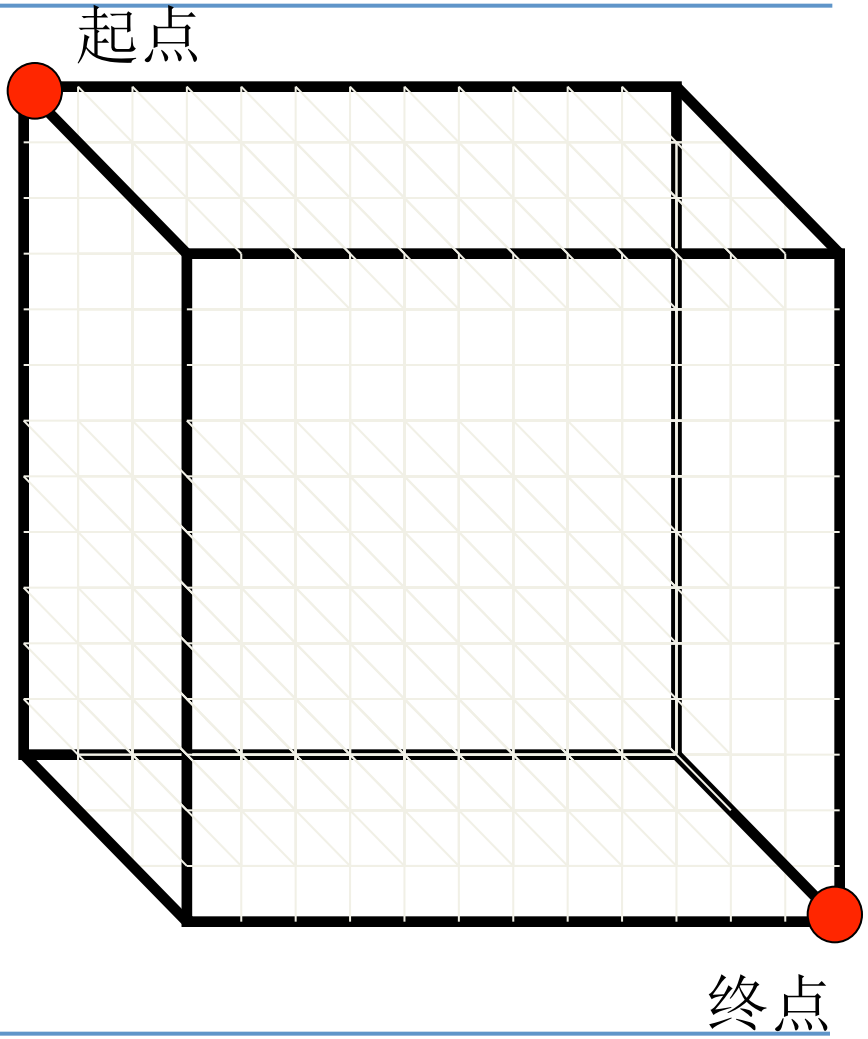
- 在(x, y, z)空间的路径为：

(0,0,0)→(1,1,0)→(1,2,1) →(2,3,2) →(3,3,3) →(4,4,4)

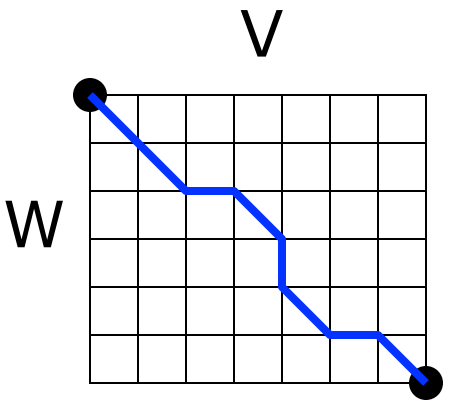
Health Informatics Lab@USTC

三序列比对

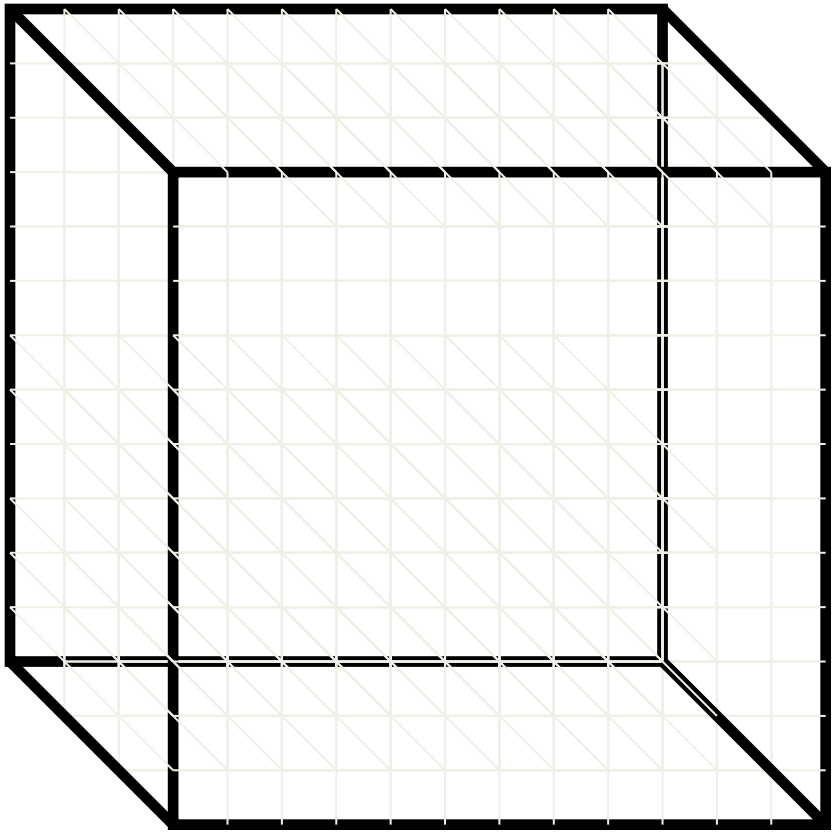
- 同双序列对比的思路一致
- 使用3维曼哈顿图，每一维表征一条待比对的序列
- 寻找一条从三维空间起点到终点的最长路径



2维和3维编辑图

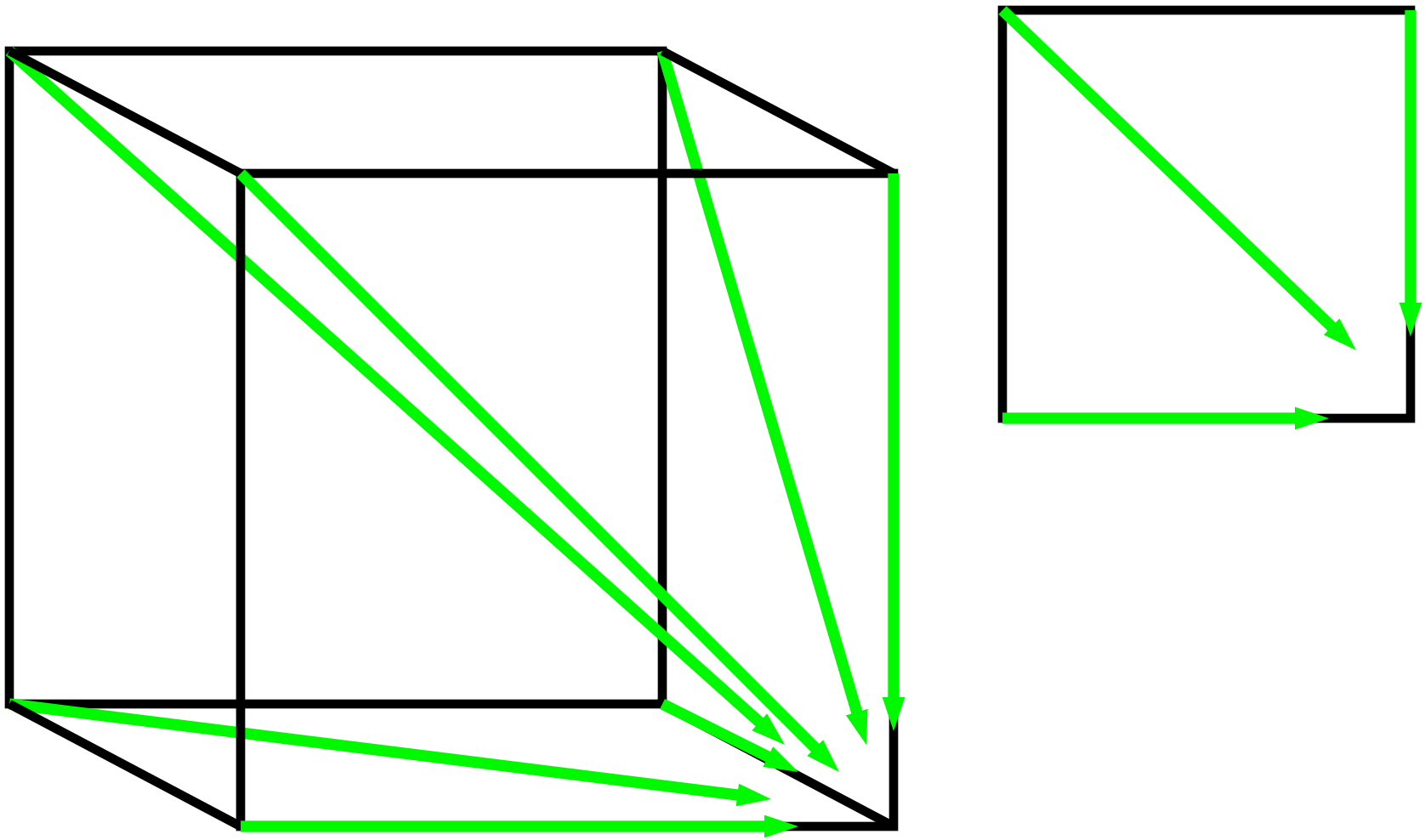


2-D 编辑图

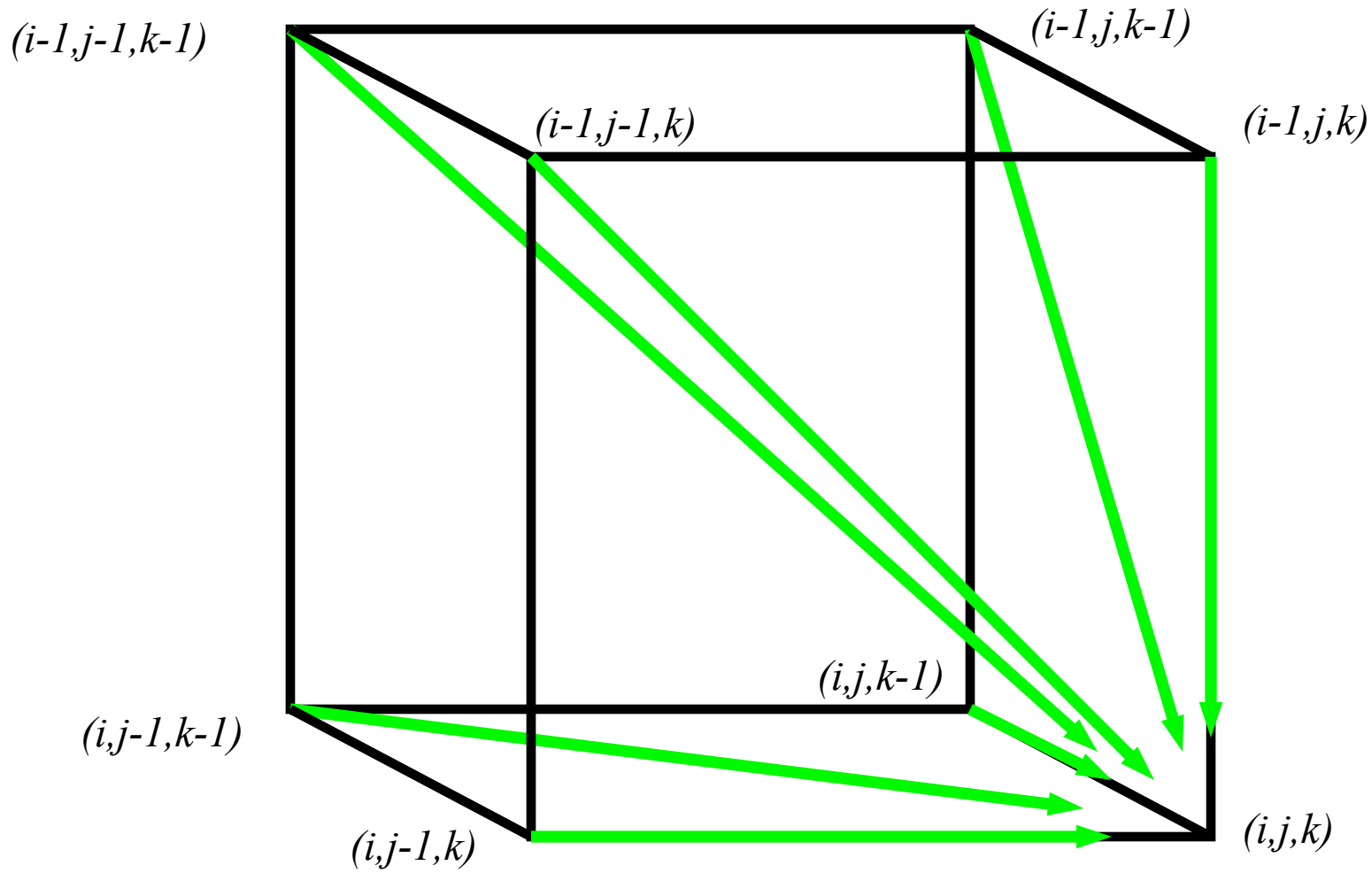


3-D 编辑图

2维和3维路径比较



3维编辑图的结构



多序列比对的动态规划算法

- $s_{i,j,k} = \max$

$$
\begin{cases}
s_{i-1,j-1,k-1} + \delta(v_i, w_j, u_k) & \text{cube diagonal: no indels} \\
s_{i-1,j-1,k} + \delta(v_i, w_j, \_) & \\
s_{i-1,j,k-1} + \delta(v_i, \_, u_k) & \text{face diagonal: one indel} \\
s_{i,j-1,k-1} + \delta(\_, w_j, u_k) & \\
s_{i-1,j,k} + \delta(v_i, \_, \_) & \\
s_{i,j-1,k} + \delta(\_, w_j, \_) & \text{edge diagonal: two indels} \\
s_{i,j,k-1} + \delta(\_, \_, u_k) &
\end{cases}
$$

- $\delta(x, y, z)$ is an entry in the 3-D scoring matrix

Health Informatics Lab@USTC

多序列比对的运行时间

- 对于长度为n的3条序列，算法复杂度为$7n^3$；$0(n^3)$
- 对于k条序列，构建一个k维的网格，运行时间为$(2^k-1)(n^k)$；$0(2^kn^k)$
- 利用动态规划算法求解k个序列的多序列比对问题，可以由两序列比对的算法简单推广，但是复杂度过高在实际中无法使用

ClustalW

- 目前常用的多序列比对算法
- ‘W’代表‘weighted’（比对的不同部分有不同的权重).
- 主要分为三步：

  1.） 构建两序列比对

  2.） 建立系统发生树

  3.） 在树的指导下进行渐进比对

第一步：双序列比对

- 序列两两比对得到相似性矩阵

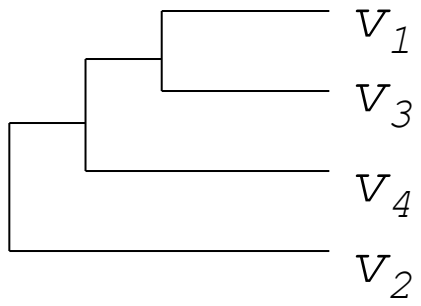
	v_1	v_2	v_3	v_4
v_1	-			
v_2	.17	-		
v_3	.87	.28	-	
v_4	.59	.33	.62	-

(.17 =17 % 残基相同)

Health Informatics Lab@USTC

第二步：系统发生树

	v_1	v_2	v_3	v_4
v_1	-			
v_2	.17	-		
v_3	.87	.28	-	
v_4	.59	.33	.62	-



计算：

$v_{1,3}$ = alignment (v_1, v_3)

$v_{1,3,4}$ = alignment(($v_{1,3}$),v_4)

$v_{1,2,3,4}$ = alignment(($v_{1,3,4}$),v_2)

Health Informatics Lab@USTC

第三步：渐进比对

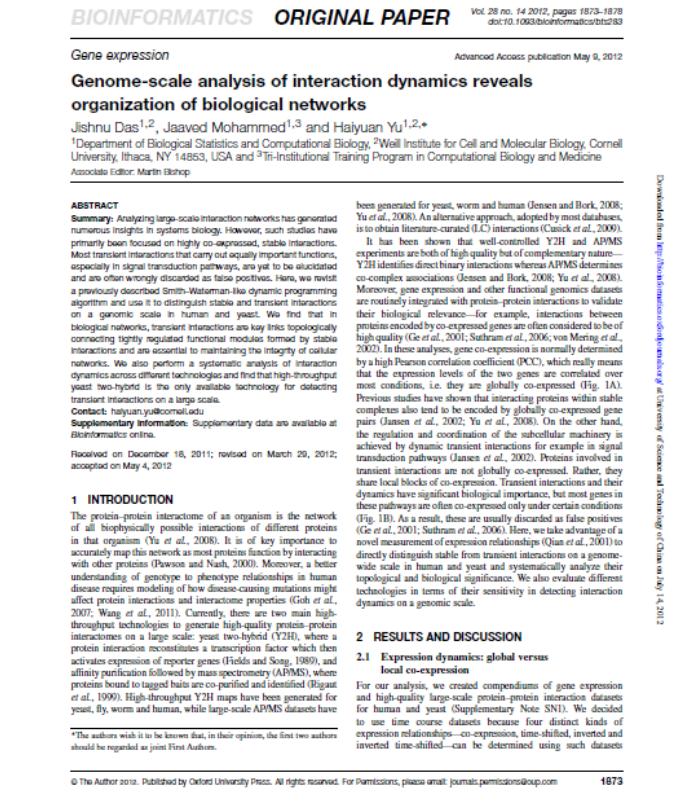
- 对最相似的两条序列进行比对
- 使用系统发生树，逐步加入更多序列，与现有比对结果进行进一步比对
- 加入必须的空位

```
FOS_RAT          PEEMSVTS-LDLTGGLPEATTPESEEAFTLPLLNDPEPK-PSLEPVKNISNMELKAEPFD
FOS_MOUSE        PEEMSVAS-LDLTGGLPEASTPESEEAFTLPLLNDPEPK-PSLEPVKSISNVELKAEPFD
FOS_CHICK        SEELAAATALDLG----APSPAAAEEAFALPLMTEAPPAVPPKEPSG--SGLELKAEPFD
FOSB_MOUSE       PGPGPLAEVRDLPG-----STSAKEDGFGWLLPPPPPPP-----------------LPFQ
FOSB_HUMAN       PGPGPLAEVRDLPG-----SAPAKEDGFSWLLPPPPPPP-----------------LPFQ
                 .   . :   ** .     :..  *:.*   *   . *                  **:
```

保守的位点

动态规划算法的其他应用

- Genome-scale analysis of interaction dynamics reveals organization of biological networks



BIOINFORMATICS ORIGINAL PAPER

Vol. 28 no. 14 2012, pages 1873–1878
doi:10.1093/bioinformatics/bts283

Gene expression

Advanced Access publication May 9, 2012

Genome-scale analysis of interaction dynamics reveals organization of biological networks

Jishnu Das[1,2], Jaaved Mohammed[1,3] and Haiyuan Yu[1,2,*]

[1]Department of Biological Statistics and Computational Biology, [2]Weill Institute for Cell and Molecular Biology, Cornell University, Ithaca, NY 14853, USA and [3]Tri-Institutional Training Program in Computational Biology and Medicine

Associate Editor: Martin Bishop

ABSTRACT

Summary: Analyzing large-scale interaction networks has generated numerous insights in systems biology. However, such studies have primarily been focused on highly co-expressed, stable interactions. Most transient interactions that carry out equally important functions, especially in signal transduction pathways, are yet to be elucidated and are often wrongly discarded as false positives. Here, we revisit a previously described Smith–Waterman-like dynamic programming algorithm and use it to distinguish stable and transient interactions on a genomic scale in human and yeast. We find that in biological networks, transient interactions are key links topologically connecting tightly regulated functional modules formed by stable interactions and are essential to maintaining the integrity of cellular networks. We also perform a systematic analysis of interaction dynamics across different technologies and find that high-throughput yeast two-hybrid is the only available technology for detecting transient interactions on a large scale.

Contact: haiyuan.yu@cornell.edu

Supplementary Information: Supplementary data are available at Bioinformatics online.

Received on December 16, 2011; revised on March 29, 2012; accepted on May 4, 2012

1 INTRODUCTION

The protein–protein interactome of an organism is the network of all biophysically possible interactions of different proteins in that organism (Yu *et al.*, 2008). It is of key importance to accurately map this network as most proteins function by interacting with other proteins (Pawson and Nash, 2000). Moreover, a better understanding of genotype to phenotype relationships in human disease requires modeling of how disease-causing mutations might affect protein interactions and interactome properties (Goh *et al.*, 2007; Wang *et al.*, 2011). Currently, there are two main high-throughput technologies to generate high-quality protein–protein interactomes on a large scale: yeast two-hybrid (Y2H), where a protein interaction reconstitutes a transcription factor which then activates expression of reporter genes (Fields and Song, 1989), and affinity purification followed by mass spectrometry (APMS), where proteins bound to tagged baits are co-purified and identified (Rigaut *et al.*, 1999). High-throughput Y2H maps have been generated for yeast, fly, worm and human, while large-scale APMS datasets have been generated for yeast, worm and human (Jensen and Bork, 2008; Yu *et al.*, 2008). An alternative approach, adopted by most databases, is to obtain literature-curated (LC) interactions (Cusick *et al.*, 2009).

It has been shown that well-controlled Y2H and APMS experiments are both of high quality but of complementary nature—Y2H identifies direct binary interactions whereas APMS determines co-complex associations (Jensen and Bork, 2008; Yu *et al.*, 2008). Moreover, gene expression and other functional genomics datasets are routinely integrated with protein–protein interactions to validate their biological relevance—for example, interactions between proteins encoded by co-expressed genes are often considered to be of high quality (Ge *et al.*, 2001; Suthram *et al.*, 2006; von Mering *et al.*, 2002). In these analyses, gene co-expression is normally determined by a high Pearson correlation coefficient (PCC), which really means that the expression levels of the two genes are correlated over most conditions, i.e. they are globally co-expressed (Fig. 1A). Previous studies have shown that interacting proteins within stable complexes also tend to be encoded by globally co-expressed gene pairs (Jansen *et al.*, 2002; Yu *et al.*, 2008). On the other hand, the regulation and coordination of the subcellular machinery is achieved by dynamic transient interactions for example in signal transduction pathways (Jansen *et al.*, 2002). Proteins involved in transient interactions are not globally co-expressed. Rather, they share local blocks of co-expression. Transient interactions and their dynamics have significant biological importance, but most genes in these pathways are often co-expressed only under certain conditions (Fig. 1B). As a result, these are usually discarded as false positives (Ge *et al.*, 2001; Suthram *et al.*, 2006). Here, we take advantage of a novel measurement of expression relationships (Qian *et al.*, 2001) to directly distinguish stable from transient interactions on a genome-wide scale in human and yeast and systematically analyze their topological and biological significance. We also evaluate different technologies in terms of their sensitivity in detecting interaction dynamics on a genomic scale.

2 RESULTS AND DISCUSSION

2.1 Expression dynamics: global versus local co-expression

For our analysis, we created compendiums of gene expression and high-quality large-scale protein–protein interaction datasets for human and yeast (Supplementary Note SN1). We decided to use time course datasets because four distinct kinds of expression relationships—co-expression, time-shifted, inverted and inverted time-shifted—can be determined using such datasets

*The authors wish it to be known that, in their opinion, the first two authors should be regarded as joint First Authors.

背景介绍

- It is of key importance to accurately map this network as most proteins function by interacting with other proteins

- Moreover, a better understanding of genotype to phenotype relationships in human disease requires modeling of how disease-causing mutations might affect protein interactions and interactome properties

背景介绍

- Moreover, gene expression and other functional genomics datasets are routinely integrated with protein-protein interactions to validate their biological relevance

- Interactions between proteins encoded by co-expressed genes are often considered to be of high quality

背景介绍

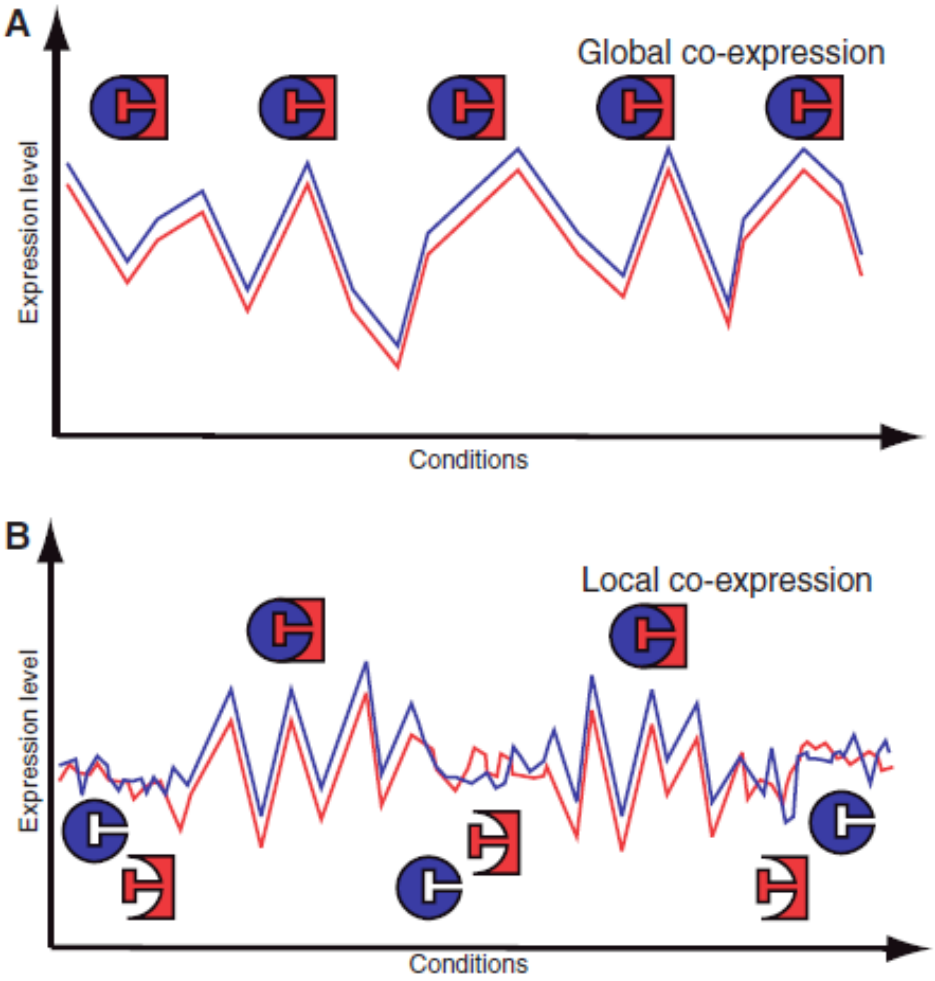
- Gene co-expression is normally determined by a high Pearson correlation coefficient (PCC), which really means that the expression levels of the two genes are correlated over most conditions, i.e. they are globally co-expressed

- Previous studies have shown that interacting proteins within stable complexes also tend to be encoded by globally co-expressed gene pairs

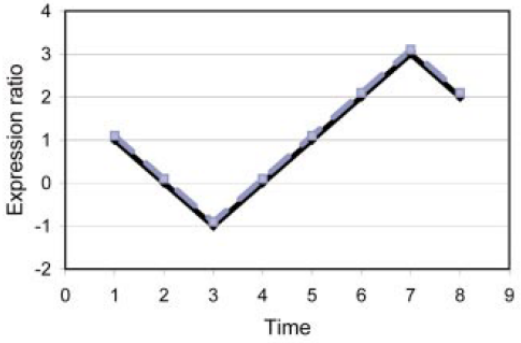
背景介绍

- On the other hand, the regulation and coordination of the subcellular machinery is achieved by dynamic transient interactions
  - signal transduction pathways
- Proteins involved in transient interactions are not globally co-expressed but locally co-expressed

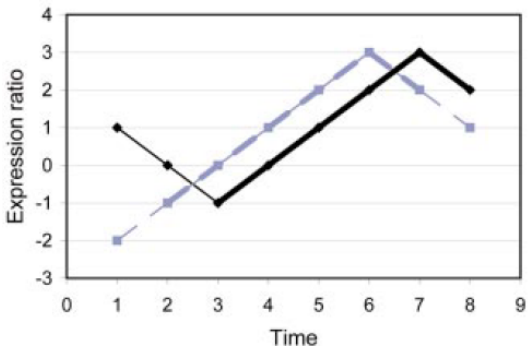
全局相关 vs 局部相关


A
Global co-expression
Expression level
Conditions
B
Local co-expression
Expression level
Conditions

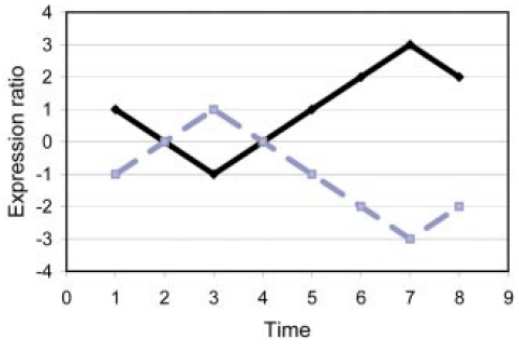
相关的类型



Simultaneous correlation



Time-delayed correlation



Inverted correlation

Health Informatics Lab@USTC

Local alignment between pairs of expression profiles

- Use a degenerate dynamic programming algorithm to find time-shifted and inverted correlations between expression profiles
  - expression ratio is normalized in "Z-score" fashion
  - score matrix: a matrix of all possible similarities between the expression ratio for gene x and gene y

$$M(x_i, y_j) = x_i y_j$$

  - The central idea is to find a local segment that has the maximal aggregated score

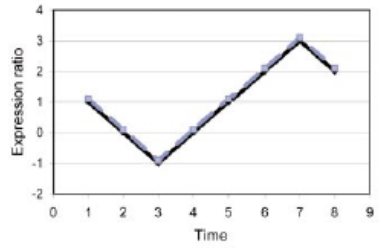
$$E_{i,j} = \max(E_{i-1,j-1} + M_{ij}, 0)$$

$$D_{i,j} = \max(D_{i-1,j-1} - M_{ij}, 0)$$

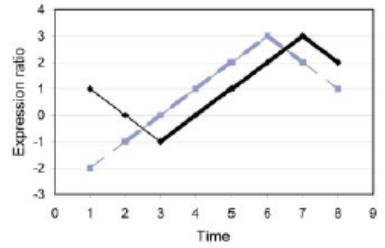
Health Informatics Lab@USTC

Results

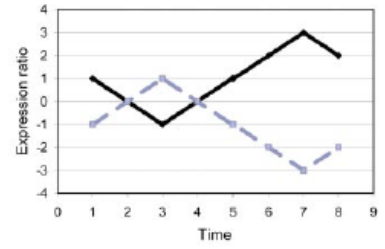
A



B



C



D

	1	0	-1	0	1	2	3	2
	0	0	0	0	0	0	0	0
1	0	1	0	0	0	1	2	3
0	0	0	1	0	0	0	1	2
-1	0	0	0	2	0	0	0	0
0	0	0	0	0	2	0	0	0
1	0	1	0	0	0	3	2	3
2	0	2	1	0	0	2	7	8
3	0	3	2	0	0	3	8	16
2	0	2	3	0	0	2	7	14

$E_{i,j} = \max(E_{i-1,j-1} + x_i \ y_j, 0)$

E

	-2	-1	0	1	2	3	2	1
	0	0	0	0	0	0	0	0
1	0	0	0	0	1	2	3	2
0	0	0	0	0	0	1	2	3
-1	0	2	1	0	0	0	0	2
0	0	0	2	1	0	0	0	0
1	0	0	0	2	2	2	3	2
2	0	0	0	0	4	6	8	7
3	0	0	0	0	3	10	15	14
2	0	0	0	0	2	7	16	19

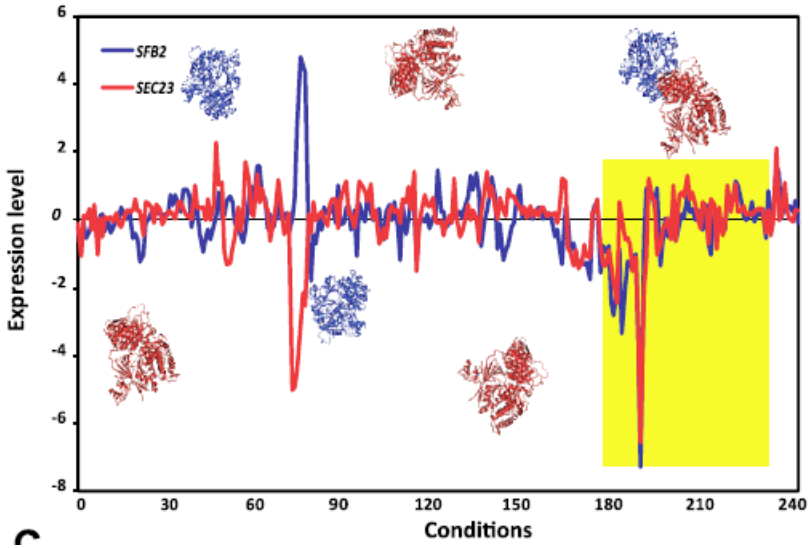
$E_{i,j} = \max(E_{i-1,j-1} + x_i \ y_j, 0)$

F

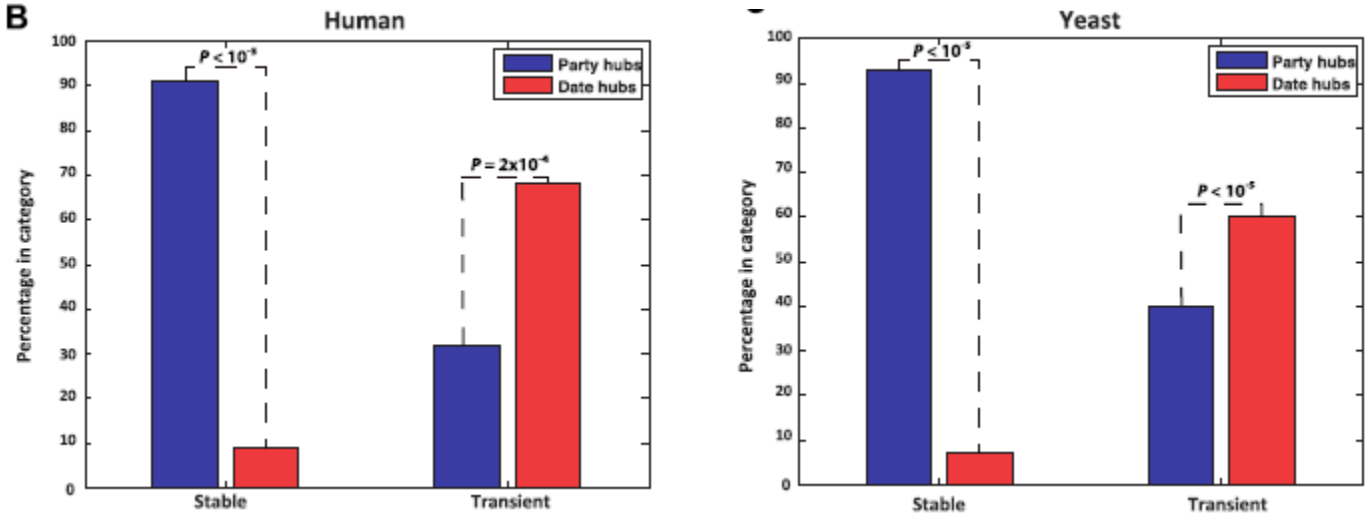
	-1	0	1	0	-1	-2	-3	-2
	0	0	0	0	0	0	0	0
1	0	1	0	0	0	1	2	3
0	0	0	1	0	0	0	1	2
-1	0	0	0	2	0	0	0	0
0	0	0	0	0	2	0	0	0
1	0	1	0	0	0	3	2	3
2	0	2	1	0	0	2	7	8
3	0	3	2	0	0	3	8	16
2	0	2	3	0	0	2	7	14

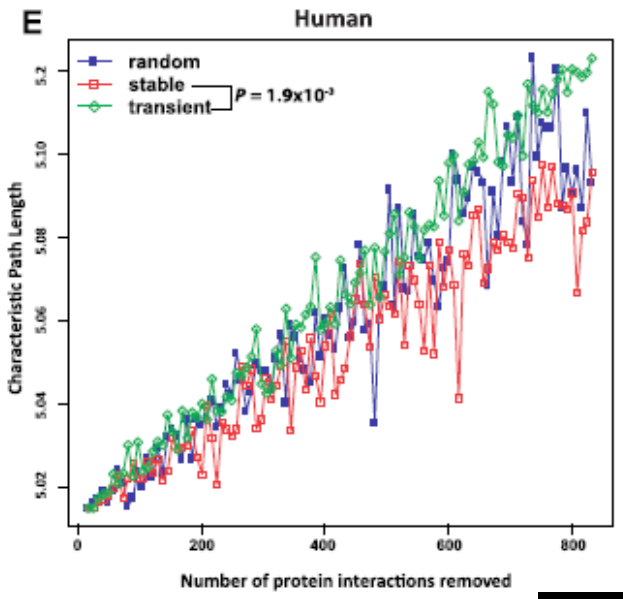
$D_{i,j} = \max(D_{i-1,j-1} - x_i \ y_j, 0)$

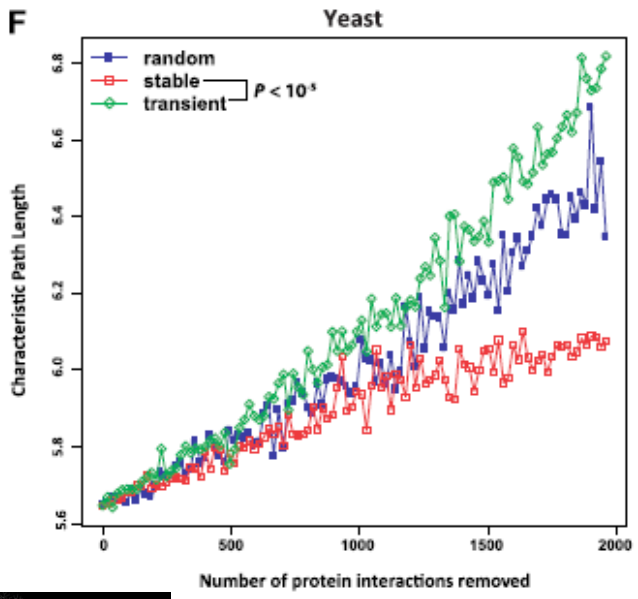
Results


SFB2
SEC23
Expression level
Conditions
6
4
2
0
-2
-4
-6
-8
0
30
60
90
120
150
180
210
240

Results


B
Human
Party hubs
Date hubs
P < 10^-5
P = 2x10^-4
Percentage in category
Stable
Transient
Yeast
Party hubs
Date hubs
P < 10^-5
P < 10^-5
Percentage in category
Stable
Transient


E
Human
random
stable
transient
P = 1.9x10⁻³
Characteristic Path Length
5.02
5.04
5.06
5.08
5.10
5.2
0
200
400
600
800
Number of protein interactions removed


F
Yeast
random
stable
transient
P < 10⁻⁵
Characteristic Path Length
5.6
5.8
6.0
6.2
6.4
6.6
6.8
0
500
1000
1500
2000
Number of protein interactions removed