algorithm

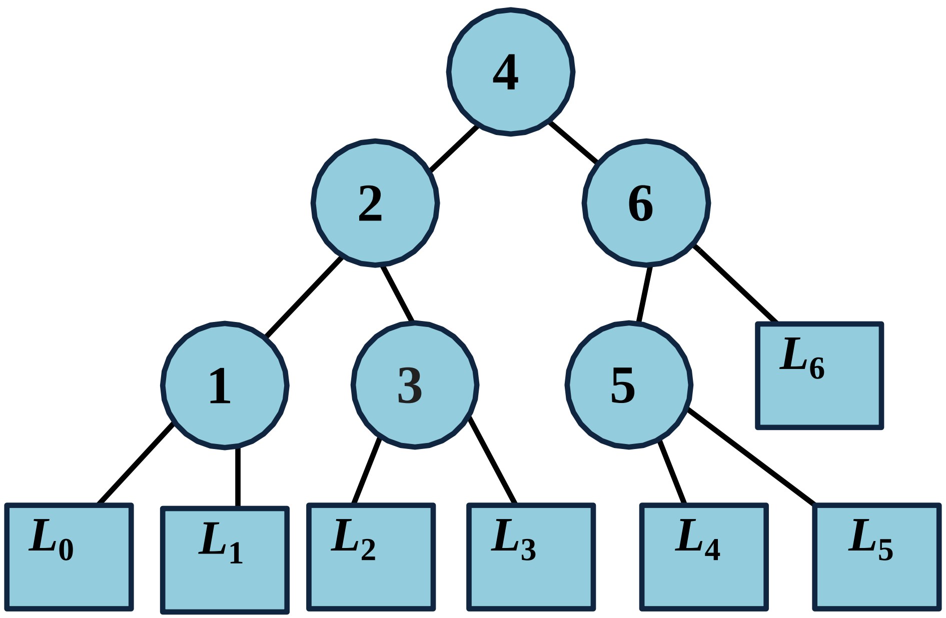
5.动态规划c

本节内容

- 4.10 最优二叉检索树的概念
- 4.11 最优二叉检索树的算法
- 4.12 RNA二级结构预测
- 4.13 序列比对

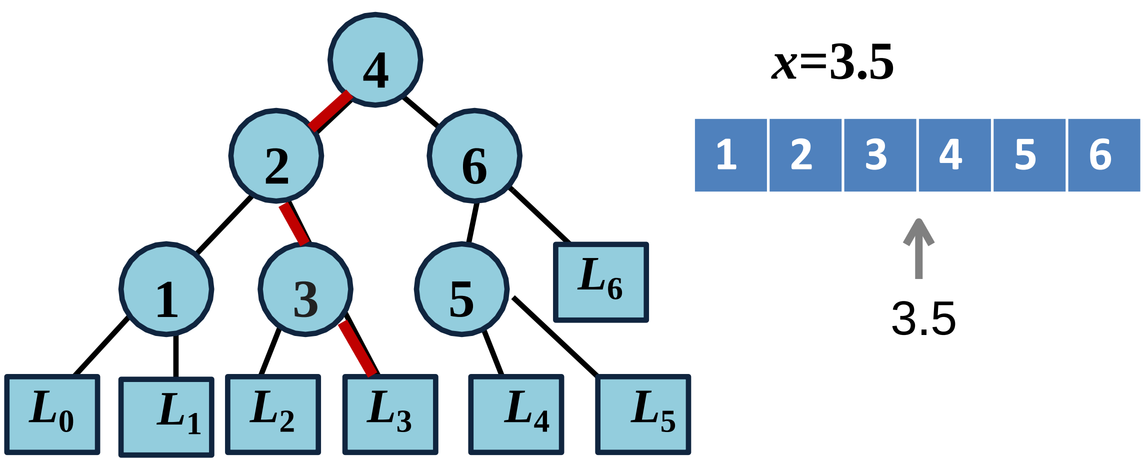
最优二叉检索树

- 二叉检索树
- 集合S为排序的 n 个元素, $x_1<x_2<\ldots<x_n$，将这些元素存储在一棵二叉树的结点上，以查找 x 是否在这些数中
- 如果 x 不在，确定 x 在那个空隙（方结点）
- 实例：S=<1,2,3,4,5,6>



二叉树的检索方法

1. 初始，x与根元素比较；
2. x < 根元素，递归进入左子树；
3. x > 根元素，递归进入右子树；
4. x = 根元素，算法停止，输出 x；
5. x 到叶结点算法停止，输出 x不在数组.



数据元素存取概率分布

- 空隙:
  - $(x_0, x_1), (x_1, x_2), \ldots, (x_{n-1}, x_n), (x_n, x_{n+1})$,
  - $x_0 = -\infty$, $x_{n+1} = +\infty$
- 给定序列 $S = < x_1, x_2, \ldots, x_n >$,
  - x 在 x_i 的概率为 b_i ,
  - x 在 (x_i , x_{i+1}) 的概率为 a_i ,
- S 的存取概率分布如下:
  - $P = < a_0, b_1, a_1, b_2, a_2, \ldots, b_n, a_n >$

实例

- 实例:　S = < 1, 2, 3, 4, 5, 6 >
- P = < 0.04, 0.1, 0.01, 0.2, 0.05, 0.2, 0.02,
  0.1, 0.02, 0.1, 0.07, 0.05, 0.04 >

- 1, 2, 3, 4, 5, 6 检索的概率分别为:
- **0.1, 0.2, 0.2, 0.1, 0.1, 0.05**
- 各个空隙的检索概率分别为:
- **0.04, 0.01, 0.05, 0.02, 0.02, 0.07, 0.04**

检索数据的平均时间

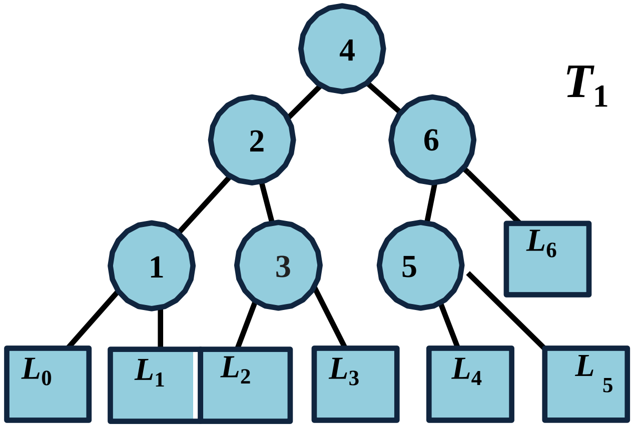
- $S = <1, 2, 3, 4, 5, 6>$
- $P = <0.04, 0.1, 0.01, 0.2, 0.05, 0.2, 0.02,$

  $0.1, 0.02, 0.1, 0.07, 0.05, 0.04>$
- $m(T_1) = [1\times0.1+2\times(0.2+0.05)+3\times(0.1+0.2+0.1)]$

  $+[3\times(0.04+0.01+0.05+0.02+0.02+0.07)$

  $+2\times0.04]$

  $= 1.8 + 0.71 = 2.51$



T_1

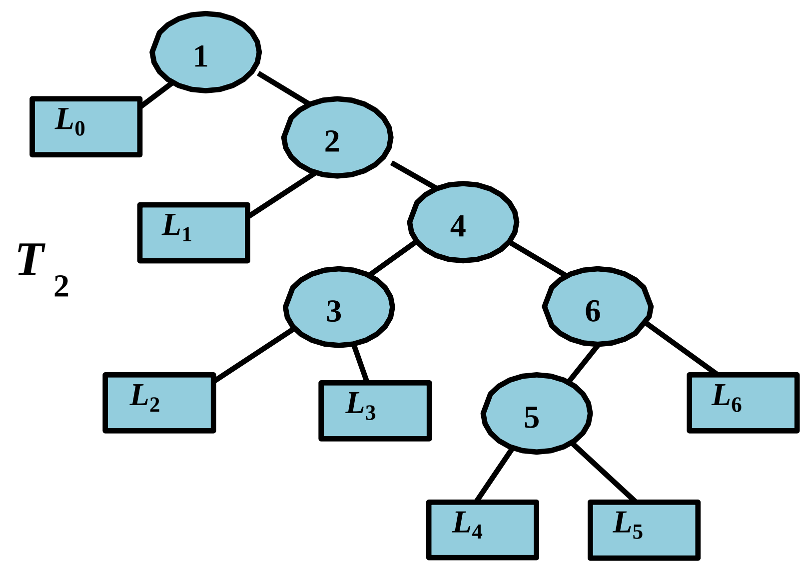
检索数据的平均时间

$S = < 1, 2, 3, 4, 5, 6 >$

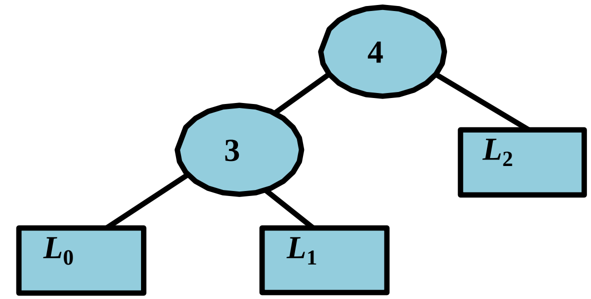
$P = < 0.04, 0.1, 0.01, 0.2, 0.05, 0.2, 0.02,$

$0.1, 0.02, 0.1, 0.07, 0.05, 0.04 >$

$$m(T_2) = [1\times0.1+2\times0.2+3\times0.1+4\times(0.2+0.05)+5\times0.1]$$
$$+ [1\times0.04+2\times0.01+4\times(0.05+0.02+0.04)$$
$$+ 5\times(0.02+0.07)]$$
$$= 2.3 + 0.95 = 3.25$$



平均比较次数计算



- 数据集 $S = <x_1, x_2, \ldots, x_n>$
- 存取概率分布
- $P=<a_0, b_1, a_1, b_2, \ldots, a_i, b_{i+1}, \ldots, b_n, a_n>$
- 结点 x_i 在 T 中的深度是 $d(x_i)$, $i=1,2,\ldots,n$,
- 空隙 L_j 的深度为 $d(L_j)$, $j=0,1,\ldots,n$,
- 平均比较次数为:

$$t = \sum_{i=1}^{n} b_i(1 + d(x_i)) + \sum_{j=0}^{n} a_j d(L_j)$$

问题

- 给定数据集

$$S = < x_1, x_2, \ldots, x_n >,$$

- 及 S 的存取概率分布如下：

$$P = < a_0, b_1, a_1, b_2, a_2, \ldots, b_n, a_n >$$

- 求一棵最优的（即平均比较次数最少的）二分检索树

最优二叉检索树的算法

- 关键问题
  - 子问题边界界定
  - 如何将该问题归结为更小的子问题
  - 优化函数的递推方程及初值
  - 计算顺序
  - 是否需要标记函数
  - 时间复杂度分析

子问题划分

- 子问题边界为(i, j)
- 数据集： $S[i, j] = < x_i, x_{i+1}, \ldots, x_j >$
- 存取概率分布：

  $P[i, j]=<a_{i-1}, b_i, a_i, b_{i+1}, \ldots, b_j, a_j>$

- 输入实例:$S = < A, B, C, D, E >$

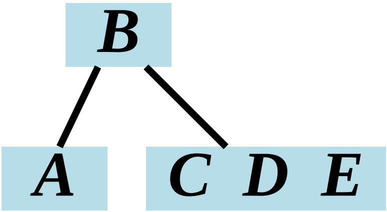
  $P = < 0.04, 0.1, 0.02, 0.3, 0.02, 0.1, 0.05, 0.2, 0.06, 0.1, 0.01>$

- 子问题： $S[2,4] = < B, C, D >$

  $P[2,4]=<0.02,0.3,0.02,0.1,0.05, 0.2,0.06>$

子问题归约

- 以 x_k 作为根归结为子问题：
- $S[\,i, k-1]$, $P[\,i, k-1]$
- $S[\,k+1, j\,]$, $P[\,k+1, j\,]$
- $S[1,5] = <A, B, C, D, E>$
- $P[1,5] = <0.04, 0.1, 0.02, 0.3, 0.02, 0.1, 0.05, 0.2, 0.06, 0.1, 0.01>$
- $S[1,1] = <A>$,
- $P[1,1] = <0.04, 0.1, 0.02>$
- $S[3,5] = <C,D,E>$,
- $P[3,5] = <0.02, 0.1, 0.05, 0.2, 0.06, 0.1, 0.01>$



子问题的概率之和

- 子问题界定 $S[i,j]$ 和 $P[i,j]$，令

$$w[i,j]=\sum_{p=i-1}^{j}a_p+\sum_{q=i}^{j}b_q$$

- 是$P[i,j]$中所有概率(数据与空隙)之和
- 实 例 ： $S[2,4] = <B,C,D>$

  $P[2,4] = <0.02,0.3,0.02,0.1,0.05,0.2,0.06>$

  $w[2,4] = (0.3+0.1+0.2) + (0.02+0.02+0.05+0.06) = 0.75$

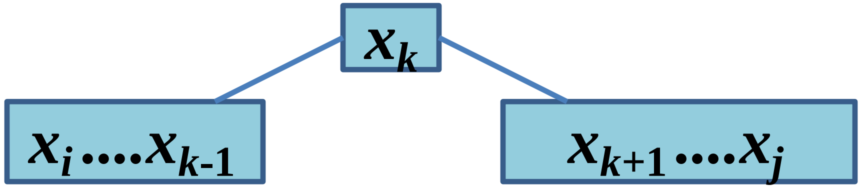
优化函数的递推方程

- 设 $m[i,j]$ 是相对于输入 $S[i,j]$ 和 $P[i,j]$ 的最优二叉搜索树的平均比较次数
- 递推方程：

$$m[i,j]=\min_{i\le k\le j}\{m[i,k-1]+m[k+1,j]+w[i,j]\},$$

$$1\le i\le j\le n$$

$$m[i,i-1]=0,\quad i=1,2,\ldots,\ n$$



$m[\,i,j\,]_k$公式的证明

$m[i,j]_k$：根为 x_k时平均比较次数的最小值

$$
\begin{aligned}
& m[i,j]_k \\
&= (m[i,k-1]+w[i,k-1])+(m[k+1,j]+w[k+1,j])+1\times b_k \\
&= (m[i,k-1]+m[k+1,j])+(w[i,k-1]+b_k+w[k+1,j]) \\
&= (m[i,k-1]+m[k+1,j])+(\sum_{p=i-1}^{k-1} a_p+\sum_{q=i}^{k-1} b_q)+b_k+(\sum_{p=k}^{j} a_p+\sum_{q=k+1}^{j} b_q) \\
&= (m[i,k-1]+m[k+1,j])+\sum_{p=i-1}^{j} a_p+\sum_{q=i}^{j} b_q \\
&= m[i,k-1]+m[k+1,j]+w[i,j]
\end{aligned}
$$

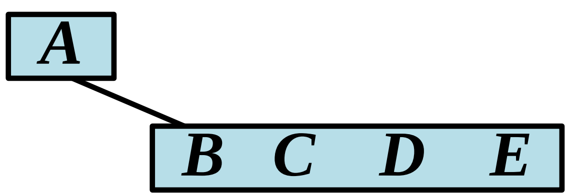
递推方程

$$m[i,j]=\min_{i\le k\le j}\{m[i,k-1]+m[k+1,j]+w[i,j]\},$$

- 平均比较次数：在所有 k 的情况下
- $m[i,j]_k$的最小值

$$m[\,i,j\,]=\min\{\,m[\,i,j\,]_k \mid i\le k\le j\,\}$$

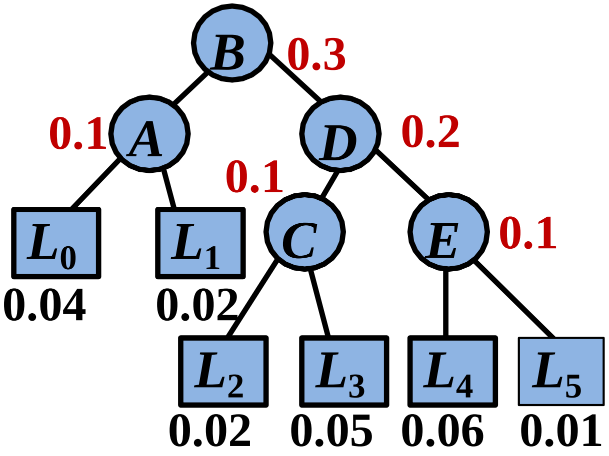
- 初值 $m[i, i\text{-}1]=0$对应于空的子问题，例如 $S=\langle A,B,C,D,E\rangle$，取 A 作根，$i=1$，k=1, 左边子问题为空树，对应于：$S[1,0]$，$m[1,0]$=0的情况



实例

$$m[i,j]=\min_{i\le k\le j}\{m[i,k-1]+m[k+1,j]+w[i,j]\}$$

$$m[i,i-1]=0$$



以B为根

k=2

m[1,1]=0.16

m[3,5]=0.88

$$m[1,5]=1+\min_{k=2,3,4}\{m[1,k-1]+m[k+1,5]\}$$

$$=1+\{m[1,1]+m[3,5]\}=1+\{0.16+0.88\}=2.04$$

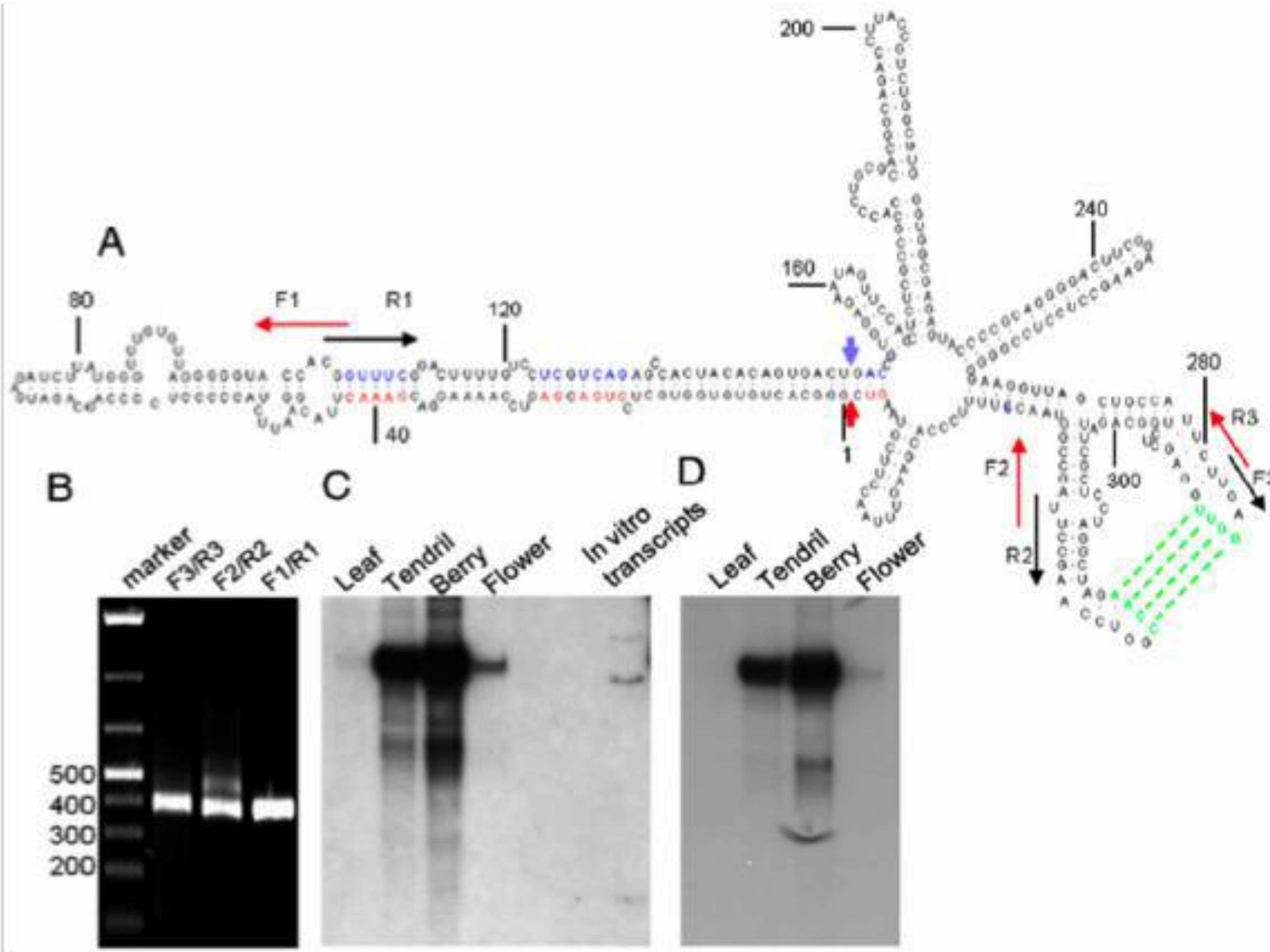
计算复杂性估计

- $m[i,j] = \min\limits_{i \le k \le j} \{m[i,k-1] + m[k+1,j] + w[i,j]\}$

  $1 \le i \le j \le n$
- $m[i,i\text{-}1] = 0, \quad i = 1,2,\ldots,n$
- i, j 的所有组合$O(n^2)$种
- 每种要对不同的 k 进行计算，$k=O(n)$
- 每次计算为常数时间
- 时间复杂性：$T(n) = O(n^3)$
- 空间复杂度：$S(n) = O(n^2)$

RNA二级结构预测

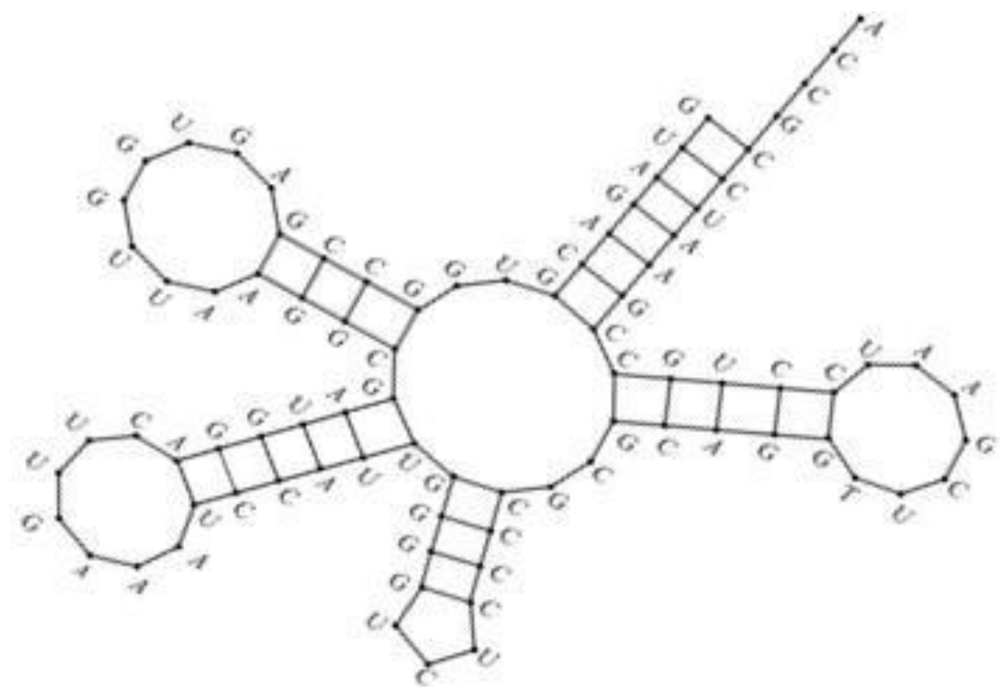
- 375nt 的环形类病毒GHVd的 RNA二级结构预测和RT-PCR、Northern blot检测结果



RNA二级结构

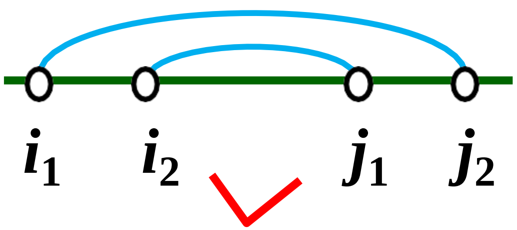
- 一级结构：由字母 *A*, *C*, *G*, *U* 标记
- 的核苷酸构成的一条链
- 实例： *A-C-C-G-C-C-U-A-A-G-C-C-G-U-C-C-U-A-A-G-* ...

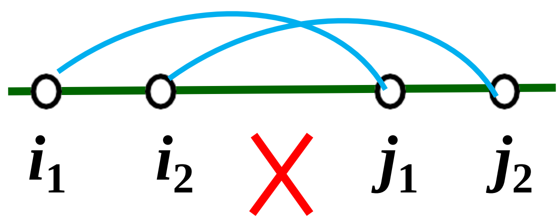
- 二级结构：
  - 核苷酸相互
  - 匹配构成的
  - 平面结构

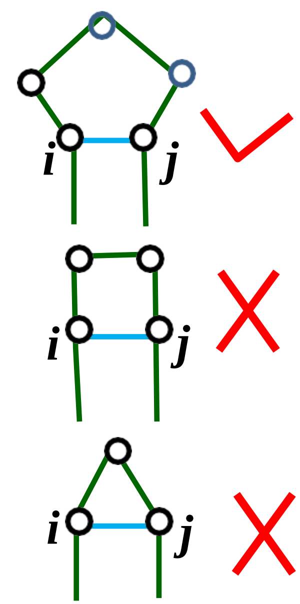


匹配原则

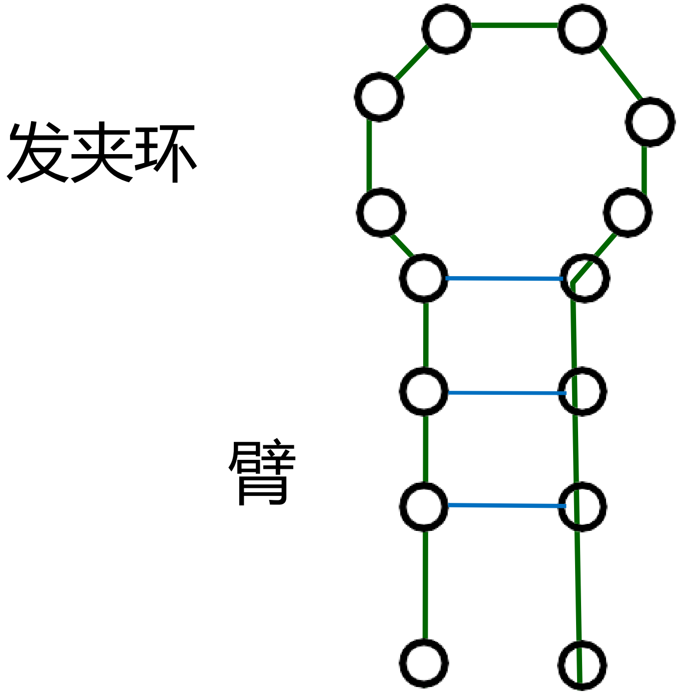
- 配对 *U-A, C-G*
- 末端不出现“尖角”，位置 i-j 配对，则 $i \le j - 4$
- 每个核苷酸只能参加一个配对
- 不允许交叉，即如果位置 i_1, i_2, j_1, j_2 满足 $i_1<i_2<j_1<j_2$，不允许 i_1-j_1, i_2-j_2 配对，但可以允许 i_1-j_2，i_2-j_1 配对.

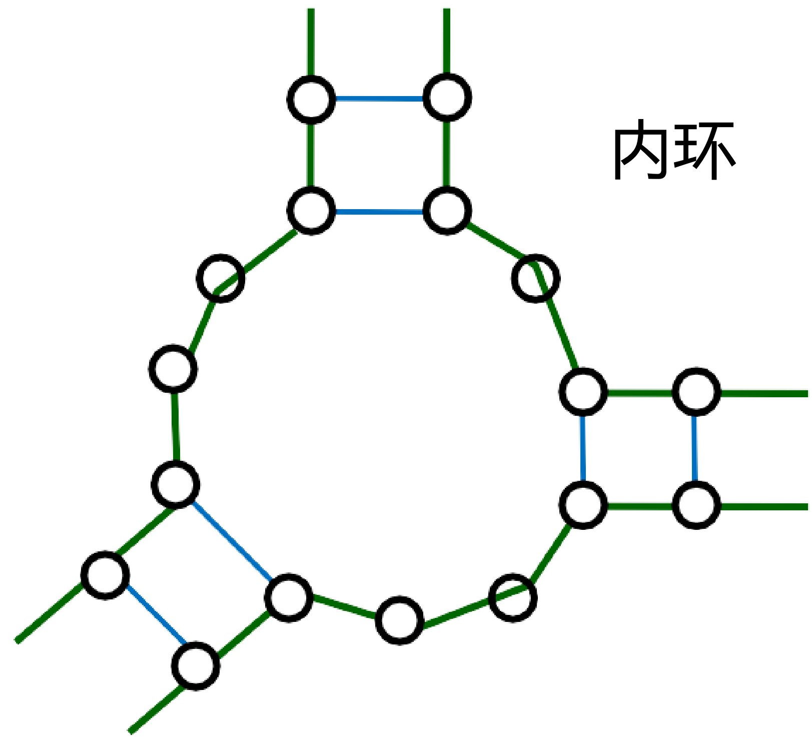






匹配的结构



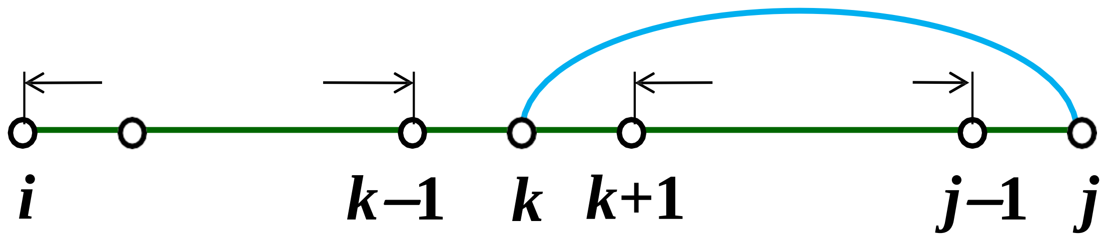


RNA二级结构问题

- 给定RNA的一条链（一级结构），预测它的可能的稳定的二级结构
- 稳定二级结构满足的条件
  - 生物学条件：具有最小自由能
  - 简化条件：具有最多的匹配对数
- 问题：给定RNA链，求具有最多匹配对数的二级结构，即最优结构

建模

- 子问题界定：前边界 i , 后边界 j
- 若 j与k (所有可能)位置匹配，归约为
- 子问题 1：i 到 k-1的链
- 子问题 2：k+1到 j-1的链



- 若 j 不参与匹配，则原问题归约为 i 到 j-1 的子问题

优化函数的递推方程

- 令$C[i,j]$是序列$S[i..j]$的最大匹配对数

$$C[i,j]=\max\{C[i,j-1],\max_{i\le k\le j-4}\{1+C[i,k-1]+C[k+1,j-1]\}\}$$

$$1\le i,\quad j\le n,\quad j-i\ge 4$$

$$C[i,j]=0\quad j-i<4$$

- 满足优化原则
- 计算顺序：按照子问题长度计算

计算复杂度分析

- 子问题个数： i, j 对的组合有 $O(n^2)$ 个
- 对于给定的 i 和 j ， j 需要考察与所有可能的 k 是否匹配，其中 $i \leq k \leq j-4$，需要 $O(n)$ 时间.
- 算法时间复杂度是 $O(n^3)$.

序列比对

- 为确定两个序列之间的相似性或同源性，将它们按照一定的规律排列，进行比对
- 应用：
- 生物信息学中用于研究同源性，如蛋白质序列或 DNA序列. 在比对中，错配与突变相对应，空位与插入或缺失相对应
- 计算语言学中用于语言进化或文本相 似性的研究

序列之间的编辑距离

- 编辑距离：
- 给定两个序列 S_1 和 S_2，
- 通过一系列字符编辑（插入、删除、替换）等操作，将S_1转变成 S_2
- 完成这种转换所需要的最少的编辑操作个数称为 S_1 和 S_2 的编辑距离

实例

- vintner转变成writers
- 编辑距离≤6

```
               vintner
• 删除v:       -intner
• 插入w:       wintner
• 插入r:       wrintner
• 删除n:       wri-tner
• 删除n:       writ-er
• 插入s:       writners
```

子问题界定和归约

- $S_1[1..n]$ 和 $S_2[1..m]$ 表示两个序列
- 子问题：$S_1[1..i]$ 和 $S_2[1..j]$，边界(i, j)

操作	归约子问题	编辑距离
删除$S_1[i]$	**$(i\text{-}1, j)$**	**+1**
$S_1[i]$后插入$S_2[j]$	**$(i, j\text{-}1)$**	**+1**
$S_1[i]$替换为$S_2[j]$	**$(i\text{-}1, j\text{-}1)$**	**+1**
$S_1[i]$=$S_2[j]$	**$(i\text{-}1, j\text{-}1)$**	**+0**

优化函数的递推方程

$C[i,j]$： $S_1[1..i]$ 和 $S_2[1..j]$ 的编辑距离

$$C[i,j]=\min\{C[i-1,j]+1, C[i,j-1]+1, C[i-1,j-1]+t[i,j]\}$$

$$t[i,j]=\begin{cases}0 & S_1[i]=S_2[j] \\ 1 & S_1[i]\neq S_2[j]\end{cases}$$

$$C[0,j]=j,$$
$$C[i,0]=i$$

- 子问题由 i, j界定，有$O(m\,n)$ 个子问题
- 每个子问题的计算为常数时间
- 算法的时间复杂度是$O(n\,m)$

动态规划算法设计要点

- 引入参数来界定子问题的边界．注意子问题的重叠程度．
- 给出带边界参数的优化函数定义与优化函数的递推关系，找到递推关系的初值．
- 判断该优化问题是否满足优化原则．
- 考虑是否需要标记函数．
- 采用自底向上的实现技术，从最小的子问题开始迭代计算，计算中用备忘 录保留优化函数和标记函数的值．
- 动态规划算法的时间复杂度是对所有子问题(备忘录)的计算工作量求和(可能需要追踪解的工作量)
- 动态规划算法一般使用较多的存储空间，这往往成为限制动态规划算法使用的瓶颈因素．