

法律声明

□ 本课件包括演示文稿、示例、代码、题库、视频和声音等内容，小象学院和主讲老师拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意及内容，我们保留一切通过法律手段追究违反者的权利。

□ 课程详情请咨询

■ 微信公众号：小象

■ 新浪微博：ChinaHadoop



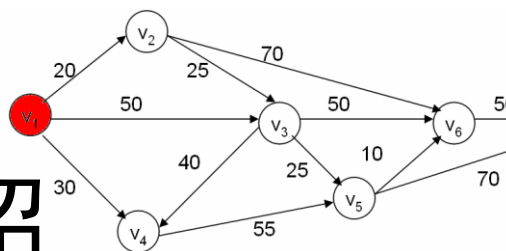
动态规划与隐马尔科夫模型



小象学院
ChinaHadoop.cn

邹博

本课程介绍



$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + c \cdot n$$

$$= 2 \cdot \left(2 \cdot T\left(\frac{n}{4}\right) + c \cdot \frac{n}{2} \right) + c \cdot n = 4T\left(\frac{n}{4}\right) + 2c \cdot n$$

$$= 4 \left(2 \cdot T\left(\frac{n}{8}\right) + c \cdot \frac{n}{4} \right) + 2c \cdot n = 8T\left(\frac{n}{8}\right) + 3c \cdot n = \dots$$

$$= 2^k T(1) + kc \cdot n = an + cn \log_2 n$$

☐ 线性表

☐ 图实践

☐ 递归分治

☐ 查找排序

☐ 字符串

☐ 动态规划

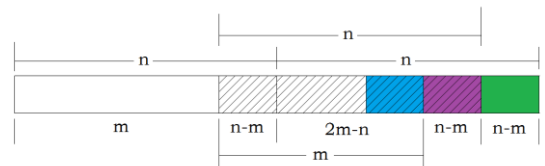
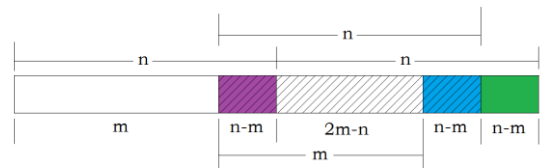
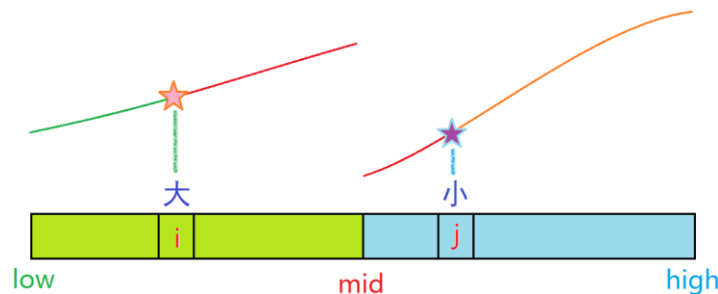
☐ 数组

☐ 概率组合数论

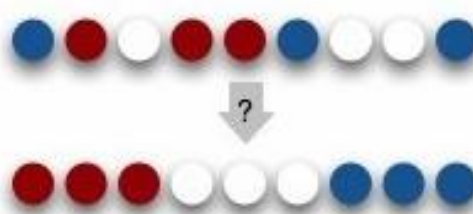
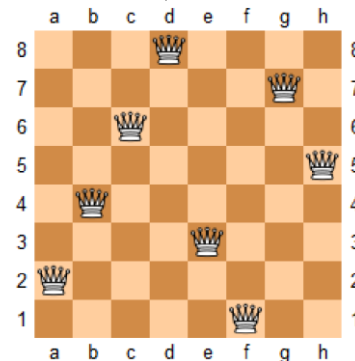
☐ 树

☐ 海量数据处理

☐ 图



	细君	昭君	探春	文成
站军姿 孙武	8小时	6小时	2小时	4小时
踢正步 王蔚	3小时	1小时	3小时	12小时



BAT面试算法特训班

□ 8月29日，12次课24课时

□ 每周二四晚20:00 – 22:00

■ 理论、实践、代码

■ 略有拖堂



BAT 8月29日开课 邹博主讲
面试算法特训班



《BAT面试算法特训班》（七天无理由退款）邹博主讲

原 价 ￥899.00

拼团价 ￥399.00 50人以上

¥299.00 100人以上 (当前价)

团长:小象学院

893 人参团

剩余时间 3天内

参团

主要内容

- 算法总论
 - 算法是有用的
 - 算法的系统性与精巧性
- 字符串循环左移
- 海明距离
- 最长公共子序列
- 字符串的全排列
- 最长回文子串
- Eratosthenes筛法
- 走棋盘问题与应用
- 隐马尔科夫模型

算法口诀

难题首选**动归**，
受阻**贪心暴力**；
考虑**分治**思想，
配合**排序哈希**。

- **动态规划**是解决相当数目问题的法宝；
 - **滚动数组**降低空间复杂度
- **贪心法**并不简单
 - Dijkstra最短路径、最小生成树Prim、Kruskal算法
- **深度优先搜索、广度优先搜索**，都可以归结为**暴力求解**；
 - **分支限界**条件加快搜索效率
- **分治法**在降低问题规模问题上很有效；
 - 快速排序、归并排序——**递归、广义分治法**
- **排序**是为了更好的查找；
 - 各种排序方法的选择
- 实在不行了，**空间换时间**——Hash
 - 深入理解Hash——`int a[65536]/int a[256]`
- 有些题目需要上述两者或者多个技术**综合运用**。

总论

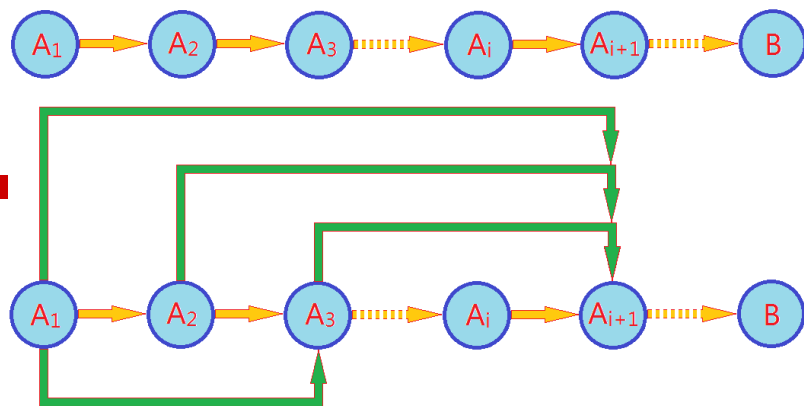
□ 算法包罗万象

- 推理、逻辑、“机智”
- 演绎、归纳、类比
- 严格归纳

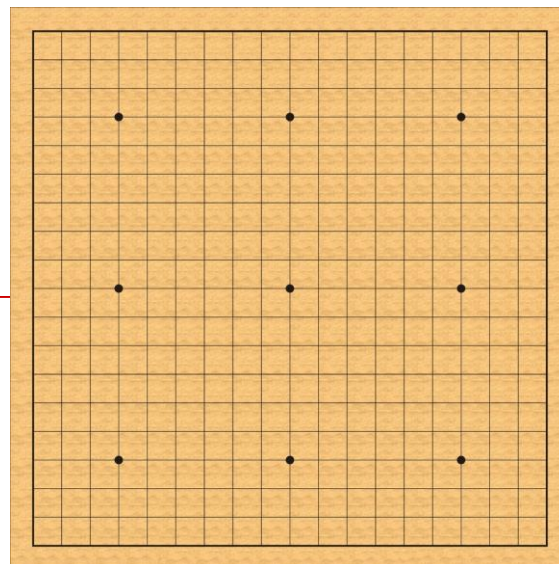
□ 算法是脑力的游戏

□ 合理运用算法，能够获得更高的效率

- 时间复杂度优先
- 空间复杂度优先
- 时间复杂度和空间复杂度的折中



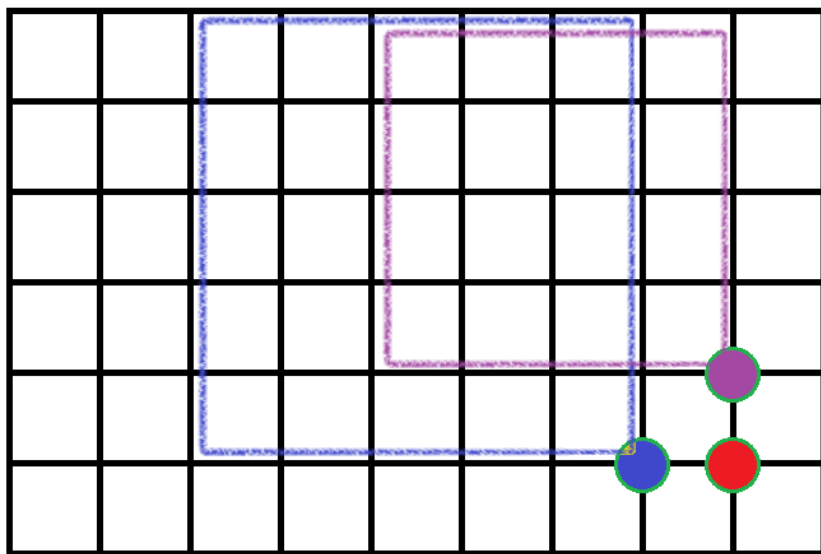
系统的“数数”



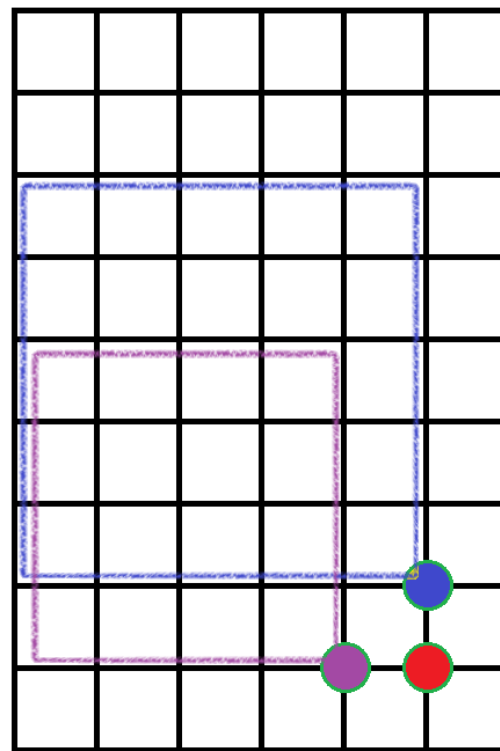
- 围棋棋盘由横纵19*19条线组成，这些线共组成多少个正方形？
- 思路：
 - 边长为1的正方形多少个？边长为2的呢？边长为3、4...的呢？
 - 以某点为右下角的正方形有多少个？把所有点的正方形相加。
- 系统的遍历——不漏不重：
 - 动态规划 Richard Ernest Bellman, 1953
 - 广度优先搜索 Edward Forrest Moore 1959, C.Y.Lee 1961
 - 深度优先搜索 John E. Hopcroft, Robert Endre Tarjan 1971-1972
 - 子集和数问题：给定N个数和某定值sum，从N个数中取若干个数，要求它们的和是sum，输出所有的取法。

算法分析 $f(i, j) = \max(f(i-1, j), f(i, j-1)), i \neq j$

□ 以 (i, j) 为右下角的正方形数目 $f(i, j)$



$$f(i, j) = f(i, j-1), \quad i < j$$



$$f(i, j) = f(i-1, j), \quad i > j$$

求1的个数

- 给定一个32位无符号整数N，求整数N的二进制数中1的个数。
 - 显然：可以通过不断的将整数N右移，判断当前数字的最低位是否为1，直到整数N为0为止。
 - 平均情况下，大约需要16次移位和16次加法。
 - 有其他更精巧的办法吗？

两种常规Code

□ 思路1:

- 每次右移一位
- 奇数则累加1

□ 思路2:

- 每次最低位清0
- 只需要 $n \&= (n-1)$ 即可

```
int OneNumber (int n)
{
    int c = 0;
    while (n != 0)
    {
        c += (n&1); //奇数则累加1
        n >>= 1;
    }
    return c;
}

int OneNumber2 (int n)
{
    int c = 0;
    while (n != 0)
    {
        n &= (n-1); //最低为1的位清0
        c++;
    }
    return c;
}
```

Code

```
int HammingWeight(unsigned int n)
{
    n = (n & 0x55555555) + ((n & 0xaaaaaaaa) >> 1);
    n = (n & 0x33333333) + ((n & 0xcccccccc) >> 2);
    n = (n & 0x0f0f0f0f) + ((n & 0xf0f0f0f0) >> 4);
    n = (n & 0x00ff00ff) + ((n & 0xff00ff00) >> 8);
    n = (n & 0x0000ffff) + ((n & 0xffff0000) >> 16);
    return n;
}
```

字符串循环左移

- 给定一个字符串 $S[0 \dots N-1]$ ，要求把 S 的前 k 个字符移动到 S 的尾部，如把字符串 “**abc**def” 前面的2个字符 ‘a’、 ‘b’ 移动到字符串的尾部，得到新字符串 “**cdefab**”：即字符串循环左移 k 。
 - 循环左移 $n+k$ 位和 k 位的效果相同。
 - 多说一句：循环左移 k 位等价于循环右移 $n-k$ 位。
- 算法要求：
 - 时间复杂度为 $O(n)$ ，空间复杂度为 $O(1)$ 。

暴力无法满足要求

□ 暴力移位法

- 每次循环左移1位，调用k次即可
- 时间复杂度 $O(kN)$ ，空间复杂度 $O(1)$

□ 三次拷贝

- $S[0...k] \rightarrow T[0...k]$
- $S[k+1...N-1] \rightarrow S[0...N-k-1]$
- $T[0...k] \rightarrow S[N-k...N-1]$
- 时间复杂度 $O(N)$ ，空间复杂度 $O(k)$

优雅一点的算法

□ $(X'Y')' = YX$

■ 如：abcdef

■ $X=ab$ $X'=ba$

■ $Y=cdef$ $Y'=fedc$

■ $(X'Y')' = (\text{bafedc})' = \text{cdefab}$

□ 时间复杂度 $O(N)$ ，空间复杂度 $O(1)$

■ 该问题会在“完美洗牌”算法中再次遇到。

Code

```
void ReverseString(char* s,int from,int to)
{
    while (from < to)
    {
        char t = s[from];
        s[from++] = s[to];
        s[to--] = t;
    }
}

void LeftRotateString(char* s,int n,int m)
{
    m %= n;
    ReverseString(s, 0, m - 1);
    ReverseString(s, m, n - 1);
    ReverseString(s, 0, n - 1);
}
```


LCS的定义

- 最长公共子序列，即Longest Common Subsequence, LCS。
- 一个序列S任意删除若干个字符得到新序列T，则T叫做S的子序列；
- 两个序列X和Y的公共子序列中，长度最长的那个，定义为X和Y的最长公共子序列。
 - 字符串13455与245576的最长公共子序列为455
 - 字符串acdfg与adfc的最长公共子序列为adf
- 注意区别最长公共子串(Longest Common Substring)
 - 最长公共子串要求连续

Code

```
int _tmain(int argc, _TCHAR* argv[])
{
    const char* str1 = "TCGGATCGACTT";
    const char* str2 = "AGCCTACGTA";
    string str;
    LCS(str1, str2, str);
    cout << str.c_str() << endl;
    return 0;
}
```

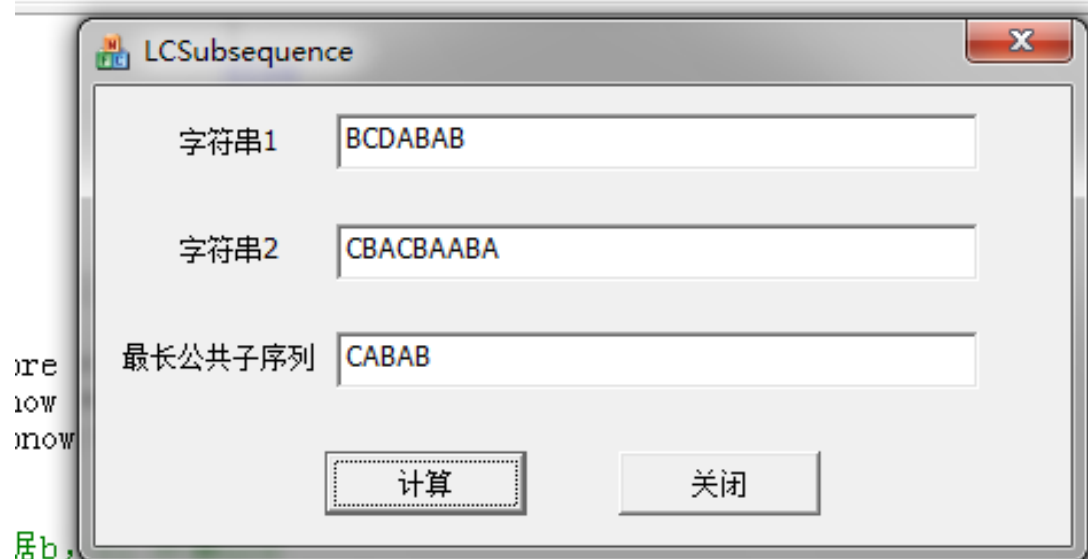
```
void LCS(const char* str1, const char* str2, string& str)
{
    int size1 = (int)strlen(str1);
    int size2 = (int)strlen(str2);
    const char* s1 = str1-1;    //从1开始数，方便后面的代码编写
    const char* s2 = str2-1;
    vector<vector<int>> chess(size1+1, vector<int>(size2+1));
    int i, j;
    for(i = 0; i <= size1; i++) //第0列
        chess[i][0] = 0;
    for(j = 0; j <= size2; j++) //第0行
        chess[0][j] = 0;

    for(i = 1; i <= size1; i++)
    {
        for(j = 1; j <= size2; j++)
        {
            if(s1[i] == s2[j]) //i, j相等
                chess[i][j] = chess[i-1][j-1] + 1;
            else
                chess[i][j] = max(chess[i][j-1], chess[i-1][j]);
        }
    }

    i = size1;
    j = size2;
    while((i != 0) && (j != 0))
    {
        if(s1[i] == s2[j])
        {
            str.push_back(s1[i]);
            i--;
            j--;
        }
        else
        {
            if(chess[i][j-1] > chess[i-1][j])
                j--;
            else
                i--;
        }
    }
    reverse(str.begin(), str.end());
}
```

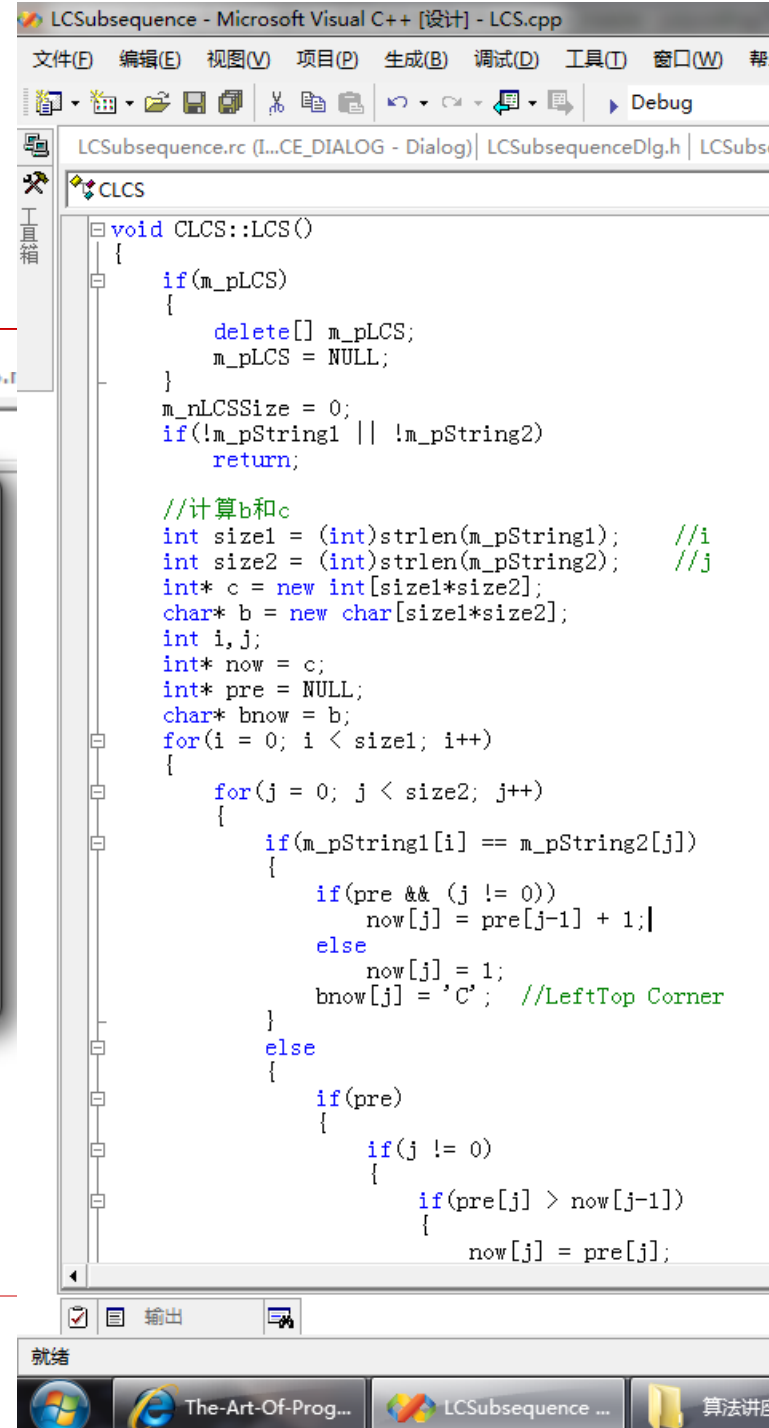
算法实现Demo

LCSubsequence.rc (L...CE_DIALOG - Dialog) | LCSubsequenceDlg.h | LCSubsequenceDlg.cpp | LCS...



re
low
now

居b,
Index = size1*size2-1;
SSize = c[nIndex];
S = new char[m_nLCSSize+1];
S[m_nLCSSize] = 0;



字符串的全排列

- 给定字符串 $S[0\dots N-1]$ ，设计算法，枚举 S 的全排列。

递归Code

```
void Print(const int* a, int size)
{
    for(int i = 0; i < size; i++)
        cout << a[i] << ' ';
    cout << endl;
}

void Permutation(int* a, int size, int n)
{
    if(n == size-1)
    {
        Print(a, size);
        return;
    }
    for(int i = n; i < size; i++)
    {
        swap(a[i], a[n]);
        Permutation(a, size, n+1);
        swap(a[i], a[n]);
    }
}

int main(int argc, char* argv[])
{
    int a[] = {1, 2, 3, 4};
    Permutation(a, sizeof(a)/sizeof(int), 0);
    return 0;
}
```

1234
1243
1324
1342
1432
1423
2134
2143
2314
2341
2431
2413
3214
3241
3124
3142
3412
3421
4231
4213
4321
4312
4132
4123

全排列的非递归算法

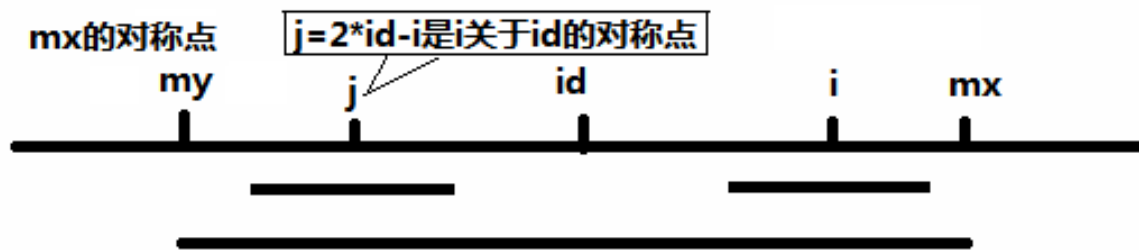
- 起点：字典序最小的排列，例如12345
- 终点：字典序最大的排列，例如54321
- 过程：从当前排列生成字典序刚好比它大的下一个排列
- 如：21543的下一个排列是23145
 - 如何计算？

最长回文子串

- 给定字符串str，若子串s是回文串，称s为str的回文子串。设计算法，计算str的最长回文子串。
 - 枚举所有子串，显然是一种解法。
 - 是否可以有更快的算法呢？

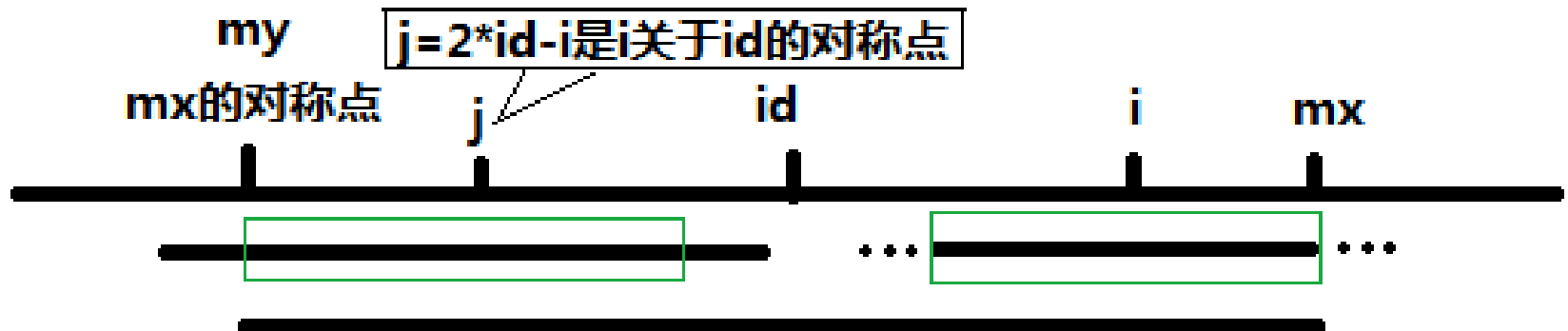
Manacher递推关系

- 记 i 关于 id 的对称点为 $j(=2*id-i)$, 若此时满足条件 $mx-i > P[j]$:
- 记 my 为 mx 关于 id 的对称点($my=2*id-mx$);
- 由于以 $S[id]$ 为中心的最大回文子串为 $S[my+1...id...mx-1]$, 即: $S[my+1...,id]$ 与 $S[id,...,mx-1]$ 对称, 而 i 和 j 关于 id 对称, 因此 $P[i]=P[j]$ ($P[j]$ 是已知的)。



Manacher递推关系

- 记 i 关于 id 的对称点为 $j(=2*id-i)$, 若此时满足条件 $mx-i < P[j]$;
- 记 my 为 mx 关于 id 的对称点($my=2*id-mx$);
- 由于以 $S[id]$ 为中心的最大回文子串为 $S[my+1...id...mx-1]$, 即: $S[my+1...,id]$ 与 $S[id...,mx-1]$ 对称, 而 i 和 j 关于 id 对称, 因此 $P[i]$ 至少等于 $mx-i$ (图中绿色框部分)。



回文对 Palindrome Pairs

- 给定若干单词组成的词典words，找到词典中的所有单词对(i,j)，使得words[i]+words[j]是回文串。
- 如：给定词典words=["abcd", "dcba", "lls", "s", "ssll", "sss"], 则应返回(0,1),(1,0),(3,2),(2,4)，即回文串是："dcbaabcd", "abcd dcba", "slls", "lls sll", "ssss", "ssll sss", "ssss"。

KMP算法

□ 字符串查找问题

- 给定文本串text和模式串pattern，从文本串text中找出模式串pattern **第一次出现**的位置。

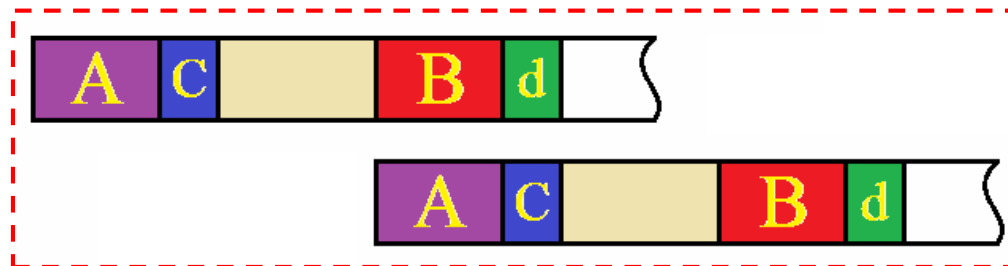
□ 最基本的字符串匹配算法

- 暴力求解(Brute Force)：时间复杂度 $O(m*n)$

□ KMP算法是一种线性时间复杂度的字符串匹配算法，它是对BF算法改进。

□ 记：文本串长度为N，模式串长度为M

- BF算法的时间复杂度 $O(M*N)$ ，空间复杂度 $O(1)$
- KMP算法的时间复杂度 $O(M+N)$ ，空间复杂度 $O(M)$



分析后的结论

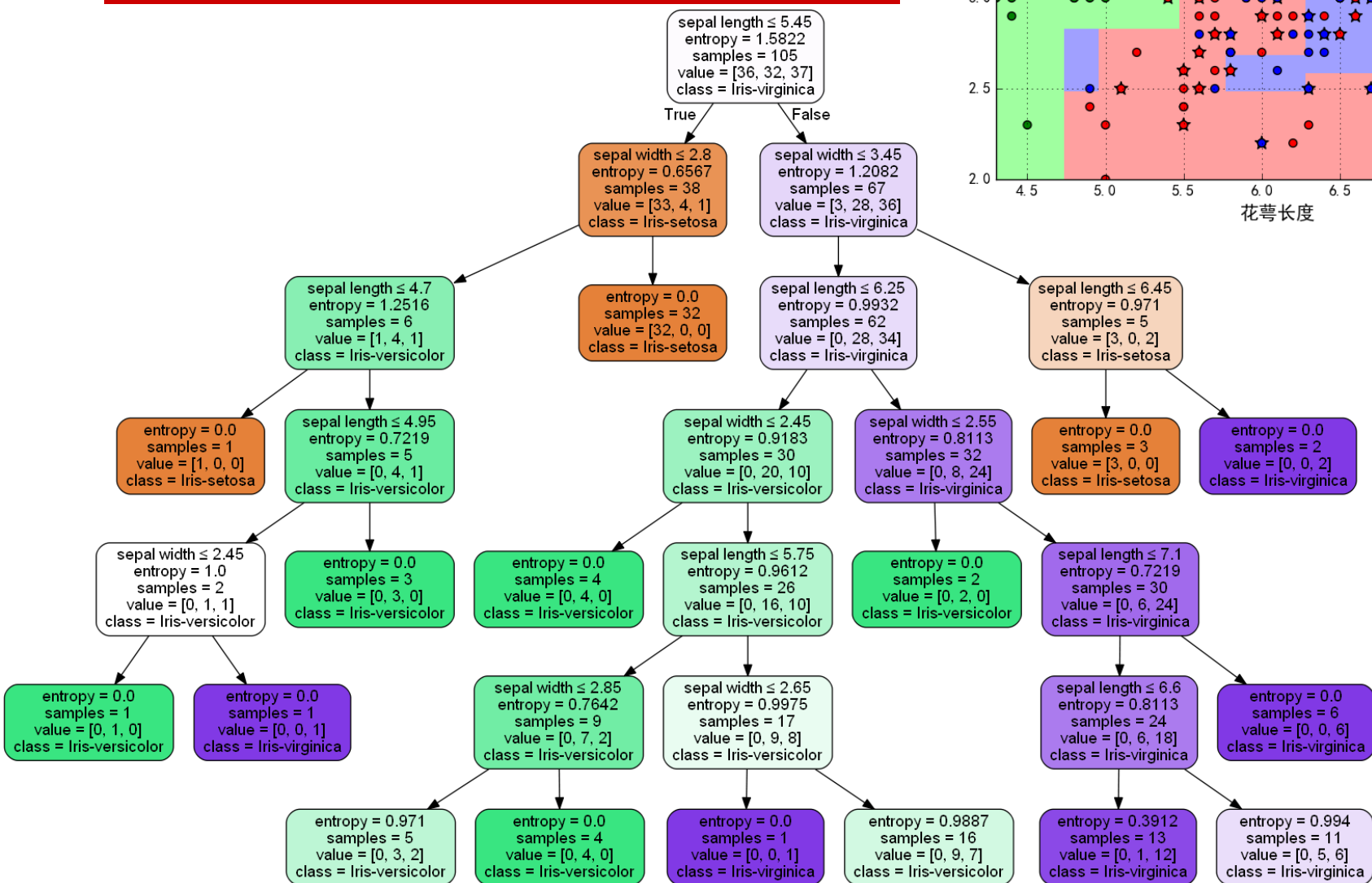
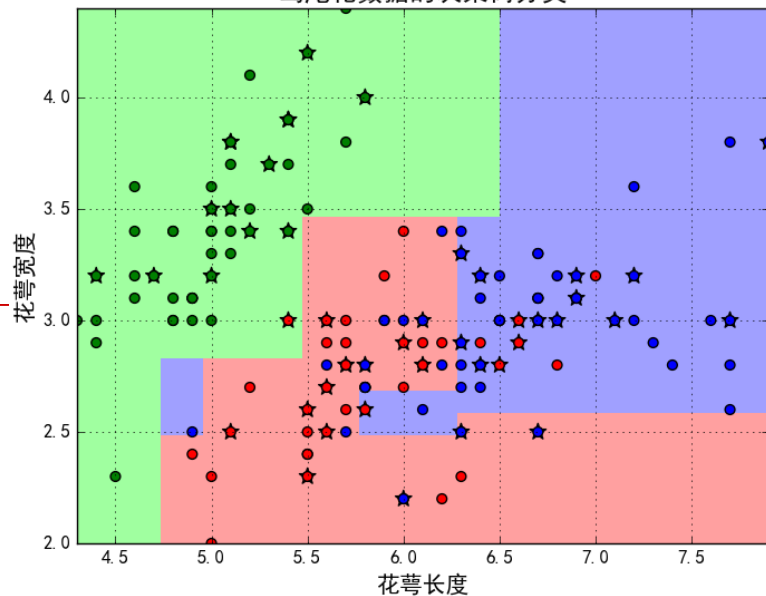
□ 对于模式串的位置 j ，考察 $\text{Pattern}_{j-1} = p_0p_1 \dots p_{j-2}p_{j-1}$ ，查找字符串 Pattern_{j-1} 的最大相等 k 前缀和 k 后缀。

■ 注：计算 $\text{next}[j]$ 时，考察的字符串是模式串的前 $j-1$ 个字符，与 $\text{pattern}[j]$ 无关。

□ 即：查找满足条件的最大的 k ，使得

■
$$p_0p_1 \dots p_{k-2}p_{k-1} = p_{j-k}p_{j-k+1} \dots p_{j-2}p_{j-1}$$

鸢尾花数据决策树



Eratosthenes筛法求素数

- 给定正整数 N ，求小于等于 N 的全部素数。
- Eratosthenes筛法
 - 将2到 N 写成一排；
 - 记排头元素为 x ，则 x 是素数；除 x 以外，将 x 的倍数全部划去；
 - 重复以上操作，直到没有元素被划去，则剩余的即小于等于 N 的全部素数。
 - 为表述方便，将排头元素称为“筛数”。

Eratosthenes筛计算100以内的素数

□ 2 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31
33 35 37 39 41 43 45 47 49 51 53 55 57 59 61
63 65 67 69 71 73 75 77 79 81 83 85 87 89 91
93 95 97 99

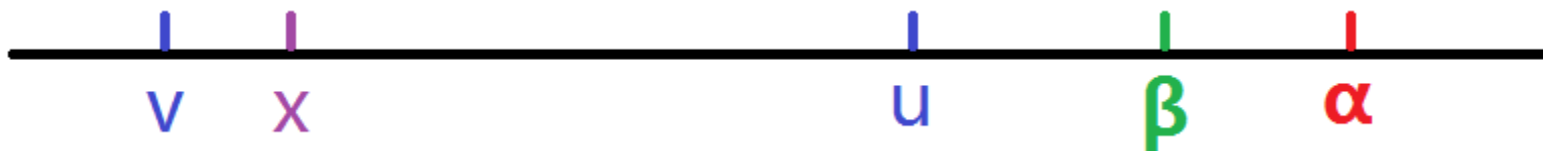
□ 2 3 5 7 11 13 17 19 23 25 29 31 35 37 41 43
47 49 53 55 59 61 65 67 71 73 77 79 83 85 89
91 95 97

□ 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 49
53 59 61 67 71 73 77 79 83 89 91 97

□ 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53
59 61 67 71 73 79 83 89 97

算法改进：筛选 α 以内的素数

- α 以内素数的最大筛数为 $\sqrt{\alpha}$ ，记 $x = \sqrt{\alpha}$
- 对于 $\beta < \alpha$
- 若 β 为合数，即： $\beta = v \cdot u$
- 显然， u 、 v 不能同时大于 x ，不妨 $v < u$ ，将它们记录在数轴上：



- 在使用 x 作为筛数之前， β 已经被 v 筛掉。

循环染色方案

□ 用红、蓝两种颜色将围成一圈的8个棋子染色。规定：若某两种染色方案通过旋转的方式可以重合，则只算一种。问：一共有多少种不同的染色方案？

总结与思考



□ Burnside定理和Polya计数以置换群为基础给出了该问题的分析过程(N个棋子c种颜色)。

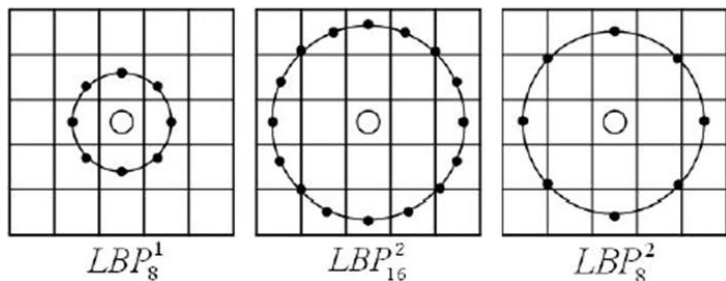
■ 用红蓝两色对正方体六面染色有多少种方案？

□ 该问题也可以看做LBP算子(Local Binary Pattern)的附属题目：

■ 根据图像上某指定像素p和周围像素(如8邻域)的相对强弱赋值为0/1，得到该像素点p的LBP值。

■ 如果循环计算LBP的最小值，则为旋转不变LBP算子。

■ 应用于：指纹识别、字符识别、人脸识别、车牌识别等



求局部最大值

- 给定一个无重复元素的数组 $A[0 \dots N-1]$ ，找到一个该数组的局部最大值。
- 规定：在数组边界外的值无穷小。即： $A[0] > A[-1]$ ， $A[N-1] > A[N]$ 。从而可得如下局部最大值的形式化定义：
$$a^* = one\ of\ \{a[i] \mid a[i] > a[i-1] \text{ 且 } a[i] > a[i+1], 0 \leq i \leq n-1\}$$
- 遍历一遍得全局最大值，它显然是局部最大值
- 可否有更快的办法？

老鼠吃奶酪

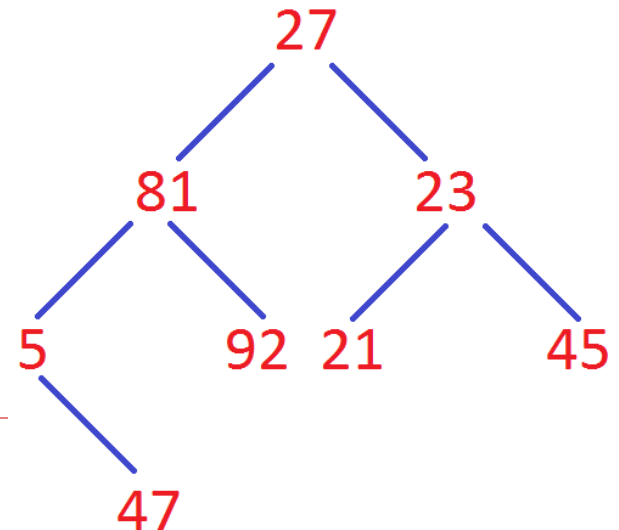
□ 一只老鼠位于迷宫左上角(0, 0)，迷宫中的数字9处有块大奶酪。0表示墙，1表示可通过路径。试给出一条可行的吃到奶酪的路径；若没有返回空。

■ 假定迷宫是4连通的。

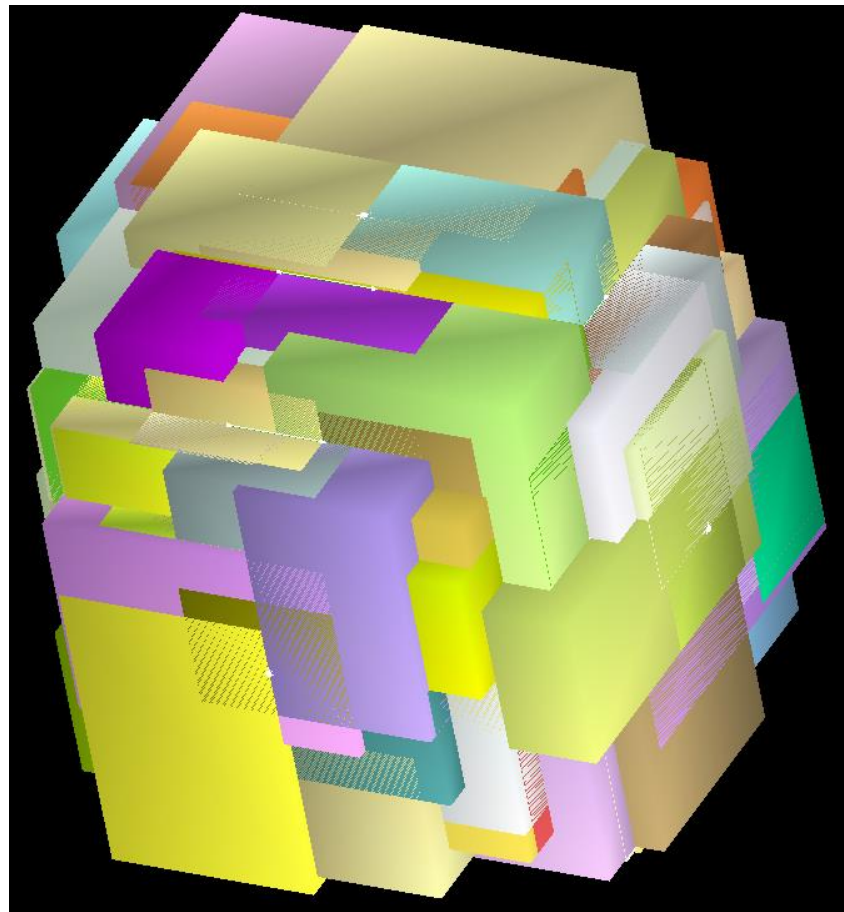
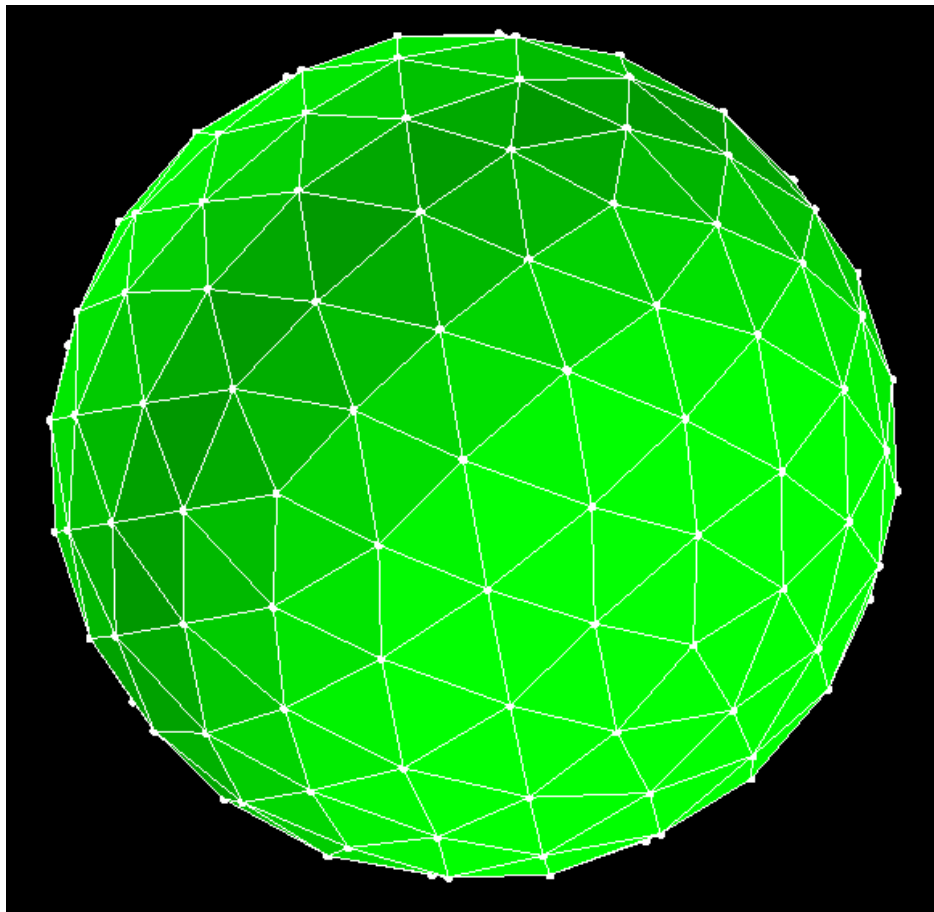
1	1	0	0	0	0	0	1
1	1	1	1	1	1	1	1
1	0	0	0	1	0	0	1
1	1	1	0	1	0	0	1
0	1	0	0	1	1	1	1
0	1	0	0	0	0	0	1
0	1	0	9	1	1	1	1
0	1	1	1	0	0	1	0

Largest BST Subtree

- 给定某二叉树，计算它的最大二叉搜索子树。
返回该最大二叉搜索子树的根结点。
- 规定：
 - 如果某子树拥有更多的结点，则该子树更大。
 - 一颗树的子树，指以某结点为根的所有结点。
- 如右图的二叉树，则返回81



三维空间中的R树



城市名称	中心坐标x	中心坐标y
北京	12956000	4824875
上海	13522000	3641093
广州	12614437	2631281
深圳	12694500	2562500
成都	11585750	3567500
天津	13046000	4714250
西安	12127500	4039281
南京	13222250	3747750
重庆	11857375	3424562
香港	12705437	2539812
澳门	12640003	2518359
合肥	13055906	3722562
安庆	13029953	3550359
蚌埠	13067218	3862125
巢湖	13121625	3690312
池州	13077937	3566328
滁州	13170531	3782062
阜阳	12891343	3859500
亳州	12888750	3988093
淮北	13003281	4000531
淮南	13026219	3825375
黄山	13170781	3445859
六安	12968312	3708375
马鞍山	13190562	3698750
宿州	13021000	3957750
铜陵	13114718	3603000
芜湖	13177156	3654093
宣城	13218718	3604484
福州	13280531	2990640
龙岩	13027531	2870218
南平	13156296	3058828
宁德	13308171	3060968
莆田	13248500	2911625
泉州	13203500	2846875
三明	13093156	3011906
厦门	13147125	2794937
漳州	13097281	2798562
兰州	11554484	4283500
白银	11596093	4350625
定西	11645968	4217000
甘南	11455500	4137875
嘉峪关	10939781	4808312
金昌	11376000	4623625
酒泉	10966171	4799953

启发式搜索：A*算法

- 给定有向图G，求从S到E的**最短路径**。
- 思考：从S到E的路径探索中，假定当前位于结点i，则与i相连的结点中，应选择哪个？
- 定义 $f[j] := g[j] + h[j]$
 - $g[j]$ ：从**起始结点S**到**当前结点j**的距离
 - $h[j]$ ：从**当前结点j**到**终止结点E**的距离
- 则，应选择 **$f[j]$ 最小**的结点作为i的**后继**。

Word Break

□ 分割词汇

□ 给定一组字符串构成的字典dict和某字符串str，将str增加若干空格构成句子，使得str被分割后的每个词都在字典dict中。返回满足要求的分割str后的所有句子。如：

■ str="catsanddog",

■ dict=["cat","cats","and","sand","dog"]

■ 返回：["cats and dog","cat sand dog"]。

Aux Code

```
void AddAnswer(const string& str, const vector<int>& oneBreak, vector<string>& answer)
{
    int s = (int)oneBreak.size();
    int size = (int)str.length();
    answer.push_back(string());
    string& sentence = answer.back();
    sentence.reserve(size+s); //申请足够的内容长度
    int start = 0, end = 0;
    for(int i = s-2; i >= 0; i--) //oneBreak[size-1]==0, 特殊处理
    {
        end = oneBreak[i]; //别忘了, k=oneBreak[i]的值表示在string[k]的前面添加break
        sentence += str.substr(start, end-start);
        sentence += ' ';
        start = end;
    }
    sentence += str.substr(start, size-start); //最后一个break
}
```

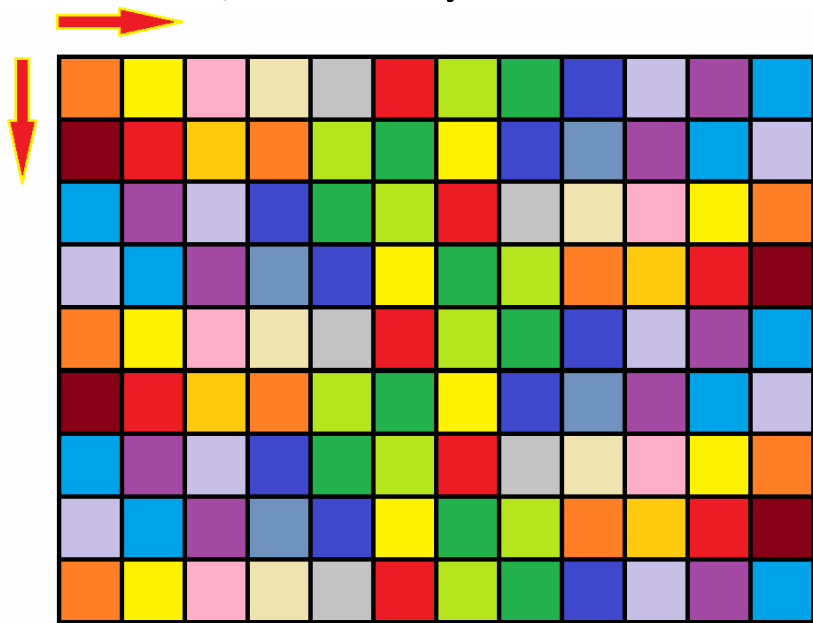
下雨天. 留客. 天留. 我不留
下雨天. 留客天. 留. 我不留
下雨天. 留客天. 留我不. 留

```
void Print(const vector<string>& answer)
{
    vector<string>::const_iterator itEnd = answer.end();
    for(vector<string>::const_iterator it = answer.begin();
        it != itEnd; it++)
        cout << *it << endl;
    cout << endl;
}

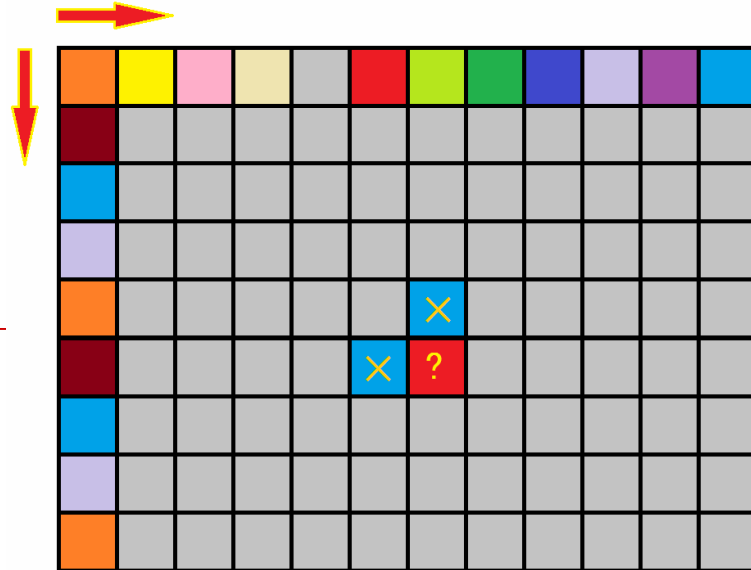
int _tmain(int argc, _TCHAR* argv[])
{
    set<string> dict;
    dict.insert("下雨天");
    dict.insert("留客");
    dict.insert("留客天");
    dict.insert("天留");
    dict.insert("留我不");
    dict.insert("我不留");
    dict.insert("留");
    dict.insert("dog");
    string str = "下雨天留客天留我不留";
    vector<string> answer;
    WordBreak(dict, str, answer);
    Print(answer);
    return 0;
}
```

走棋盘/格子取数

- 给定 $m \times n$ 的矩阵，每个位置是一个非负整数，在左上角放一机器人，它每次只能朝右和下走，直到右下角，求所有路径中，总和最小的那条路径。



状态转移方程



□ 走的方向决定了同一个格子不会经过两次。

■ 若当前位于 (x,y) 处，它来自于哪些格子呢？

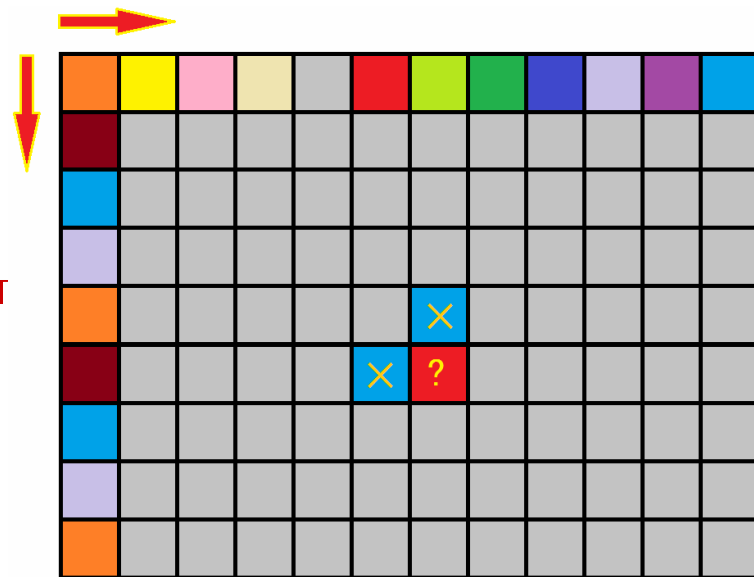
■ $dp[0,0]=a[0,0]$ / 第一行(列)累积

■ $dp[x,y] = \min(dp[x-1,y]+a[x,y], dp[x,y-1]+a[x,y])$

■ 即： $dp[x,y] = \min(dp[x-1,y], dp[x,y-1]) + a[x,y]$

□ 思考：若将上述问题改成“求从左上到右下的最大路径”呢？

状态转移方程



□ 状态转移方程：

$$\begin{cases} dp(i, 0) = \sum_{k=0}^i chess[k][0] \\ dp(0, j) = \sum_{k=0}^j chess[0][k] \\ dp(i, j) = \min(dp(i-1, j), dp(i, j-1)) + chess[i][j] \end{cases}$$

□ 滚动数组：

$$\begin{cases} dp(j) = \sum_{k=0}^j chess[0][k] \\ dp(j) = \min(dp(j), dp(j-1)) + chess[i][j] \end{cases}$$

Code

```
int MinPath(vector<vector<int> >& chess, int M, int N)
{
    vector<int> pathLength(N);
    int i, j;

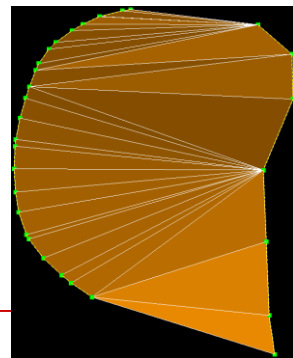
    //初始化
    pathLength[0] = chess[0][0];
    for(j = 1; j < N; j++)
        pathLength[j] = pathLength[j-1] + chess[0][j];

    //依次计算每行
    for(i = 1; i < M; i++)
    {
        pathLength[0] += chess[i][0];
        for(j = 1; j < N; j++)
        {
            if(pathLength[j-1] < pathLength[j])
                pathLength[j] = pathLength[j-1] + chess[i][j];
            else
                pathLength[j] += chess[i][j];
        }
    }
    return pathLength[N-1];
}

int _tmain(int argc, _TCHAR* argv[])
{
    const int M = 10;
    const int N = 8;
    vector<vector<int> > chess(M, vector<int>(N));

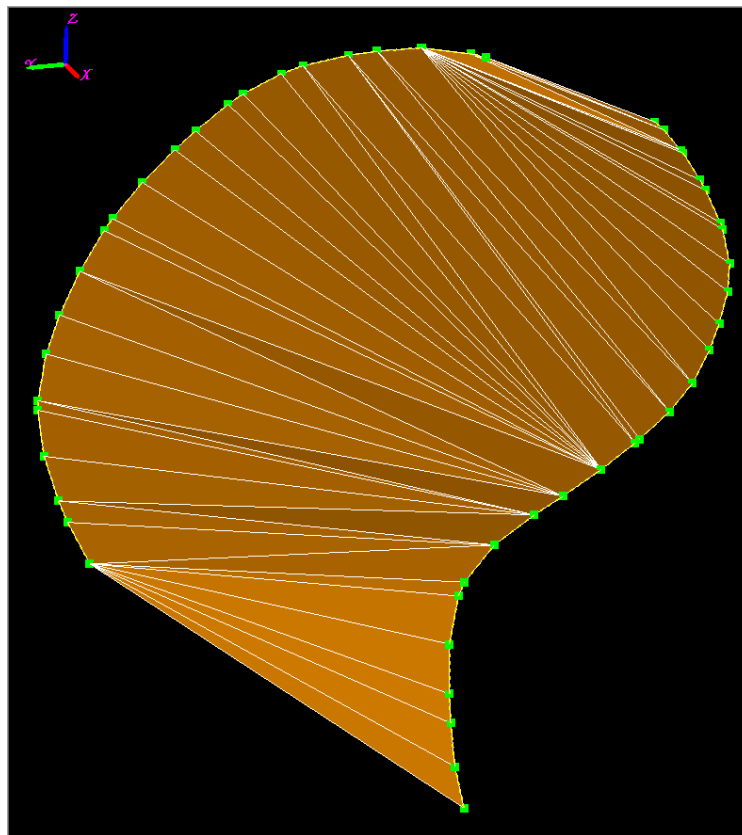
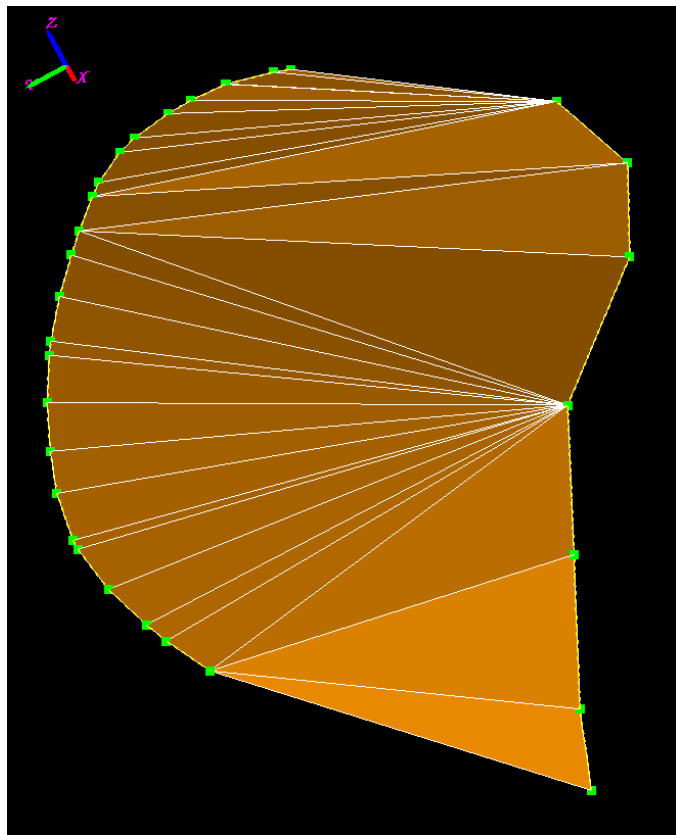
    //初始化棋盘：(随机给定)
    int i, j;
    for(i = 0; i < M; i++)
    {
        for(j = 0; j < N; j++)
            chess[i][j] = rand() % 100;
    }
    cout << MinPath(chess, M, N) << endl;
    return 0;
}
```

实践应用

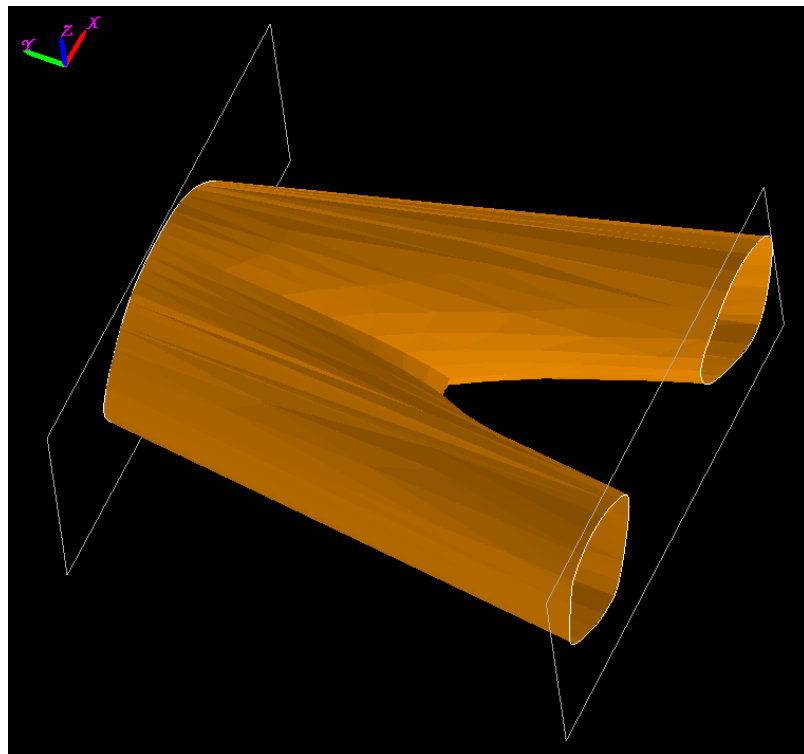
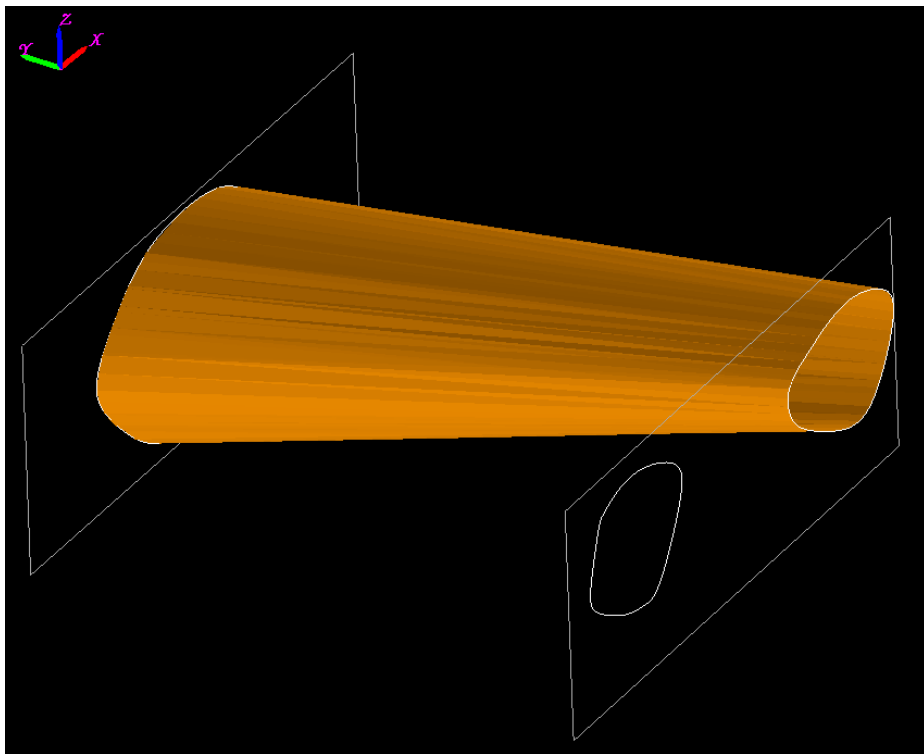


- 给定空间两条曲线，设计经过这两条曲线的曲面，使得该曲面是“最合理”的。
- 探讨如下方案：两条曲线上的点序列，记做 $P1[0...N-1]$ 和 $P2[0...M-1]$ ，计算 $P1[i]$ 和 $P2[j]$ 的距离，从而得到二维表格 $T[N][M]$ 。从 $T[0][0]$ 开始出发，只能向右和向下走(曲面不能自相交)，从 $(0,0)$ 到 $(N-1,M-1)$ 的最短路径，就是曲面的一种“合理连接”。

实践：实践中的应用

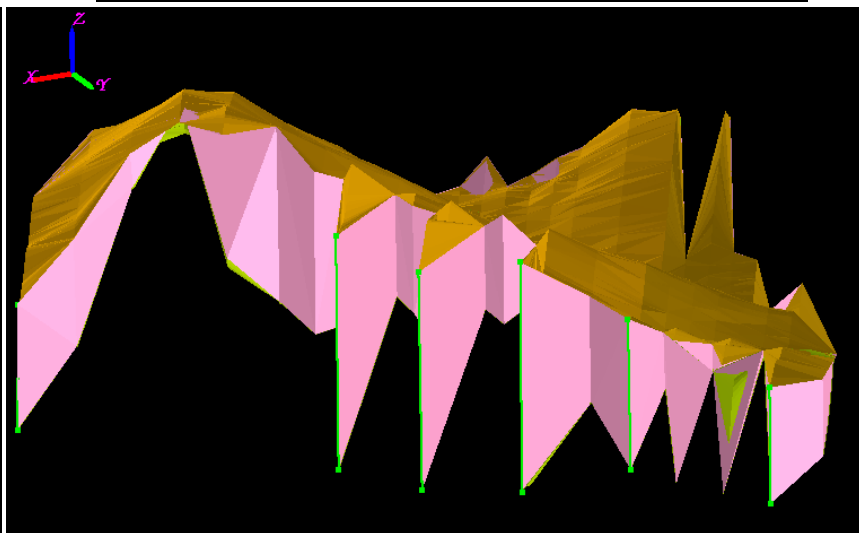
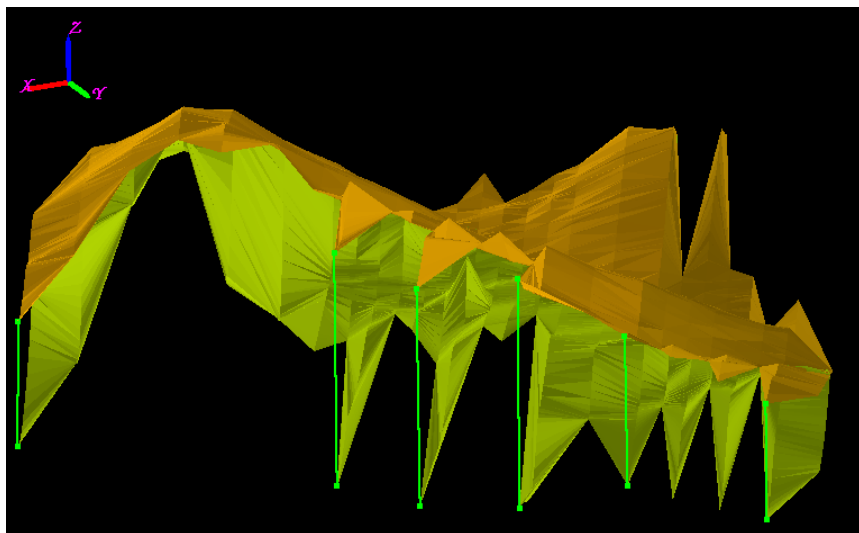
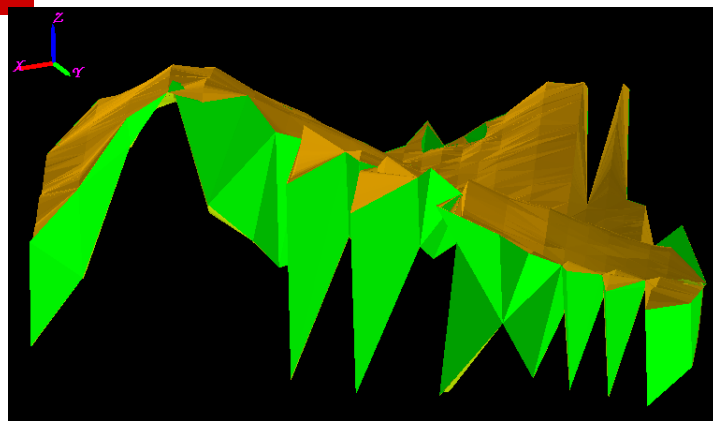


如果三维曲线是封闭线...



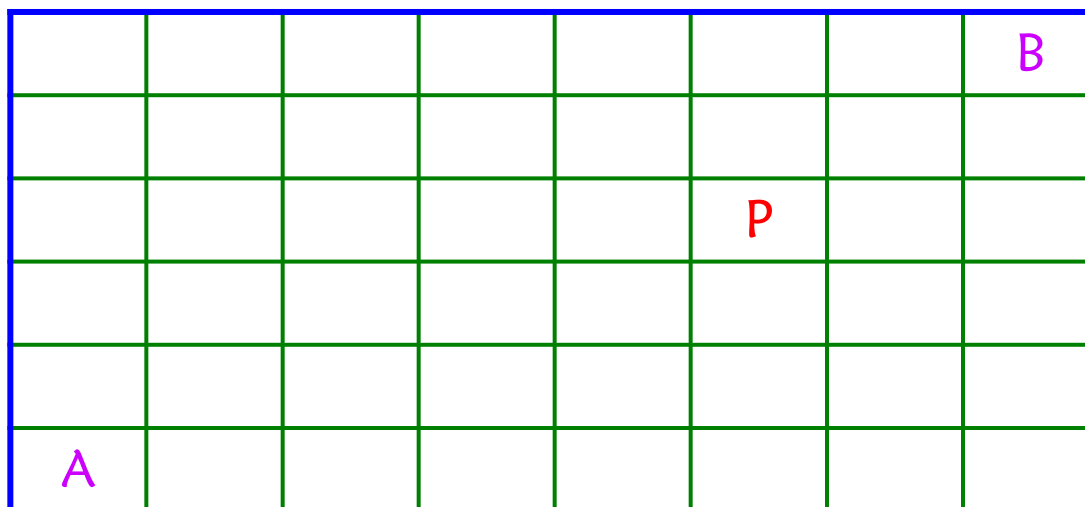
经过引导线的曲面——带约束的走棋盘

- 右上：未使用引导线
- 左下：输入的引导线
- 右下：过引导线的曲面



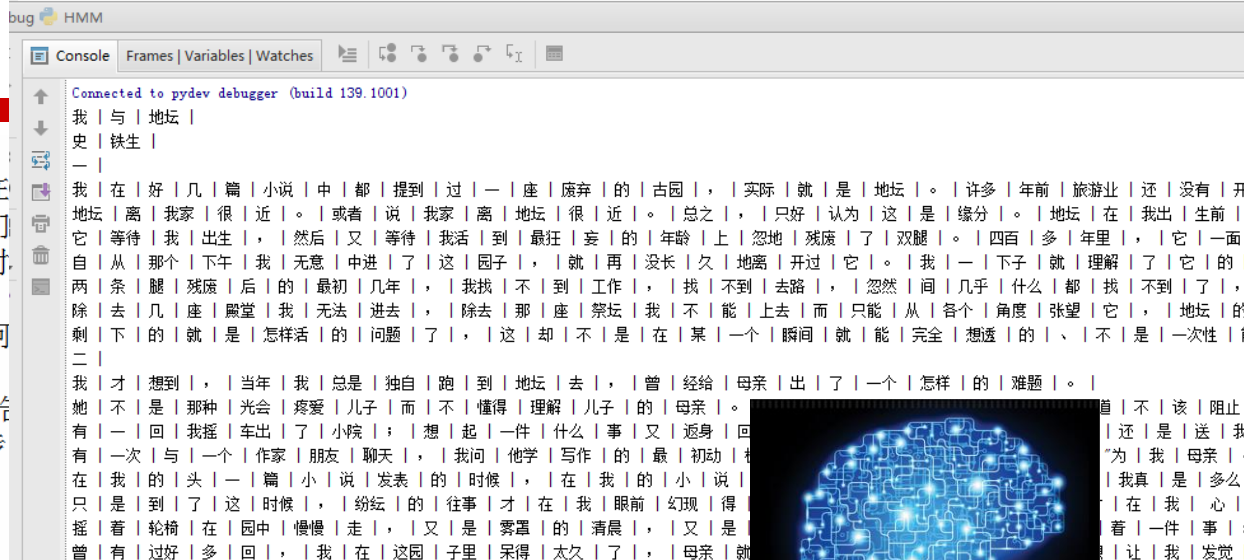
思考：陷阱走棋盘

- 在 8×6 的矩阵中，每次只能向上或向右移动一格，并且不能经过P。试计算从A移动到B一共有多少种走法。



中文分词

```
if __name__ == "__main__":
    pi, A, B = load_train()
    f = file("../text\\novel.txt")
    data = f.read()[3:].decode('utf-8')
    f.close()
    decode = viterbi(pi, A, B, data)
    segment(data, decode)
```



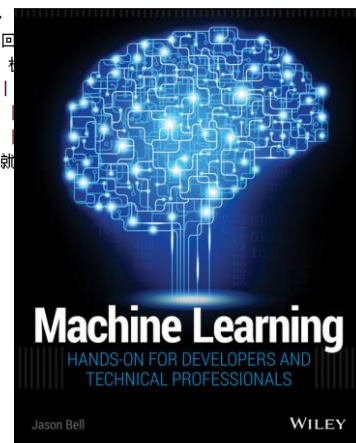
```
bug HMM
Console Frames Variables Watches
Connected to pydev debugger (build 139.1001)
我 | 与 | 地 | 坛 |
史 | 铁 | 生 |
-
我 | 在 | 好 | 几 | 篇 | 小 | 说 | 中 | 都 | 提 | 到 | 过 | 一 | 座 | 废 | 弃 | 的 | 古 | 园 |， | 实 | 际 | 就 | 是 | 地 | 坛 |。 | 许 | 多 | 年 | 前 | 旅 | 游 | 业 | 还 | 没 | 有 | 开 | 发 |。 | 离 | 我 | 家 | 很 | 近 |。 | 或 | 者 | 说 | 我 | 家 | 离 | 地 | 坛 | 很 | 近 |。 | 总 | 之 |， | 只 | 好 | 认 | 为 | 这 | 是 | 缘 | 分 |。 | 地 | 坛 | 在 | 我 | 出 | 生 | 前 | 它 | 等 | 待 | 我 | 出 | 生 |， | 然 | 后 | 又 | 等 | 待 | 我 | 活 | 到 | 最 | 狂 | 妄 | 的 | 年 | 龄 | 上 | 忽 | 地 | 残 | 废 | 了 | 双 | 腿 |。 | 四 | 百 | 多 | 年 | 里 |， | 它 | 一 | 面 | 自 | 从 | 那 | 个 | 下 | 午 | 我 | 无 | 意 | 中 | 进 | 了 | 这 | 园 | 子 |， | 就 | 再 | 没 | 长 | 久 | 地 | 离 | 开 | 过 | 它 |。 | 我 | 一 | 下 | 子 | 就 | 理 | 解 | 了 | 它 | 的 | 两 | 条 | 腿 | 残 | 废 | 后 | 的 | 最 | 初 | 几 | 年 |， | 我 | 找 | 不 | 到 | 工 | 作 |， | 找 | 不 | 到 | 去 | 路 |， | 忽 | 然 | 间 | 几 | 乎 | 什 | 么 | 都 | 找 | 不 | 到 | 了 |， | 除 | 去 | 几 | 座 | 殿 | 堂 | 我 | 无 | 法 | 进 | 去 |， | 除 | 去 | 那 | 座 | 祭 | 坛 | 我 | 不 | 能 | 上 | 去 | 而 | 只 | 能 | 从 | 各 | 个 | 角 | 度 | 张 | 望 | 它 |， | 地 | 坛 | 的 | 剩 | 下 | 的 | 就 | 是 | 怎 | 样 | 活 | 的 | 问 | 题 | 了 |， | 这 | 却 | 不 | 是 | 在 | 某 | 一 | 个 | 瞬 | 间 | 就 | 能 | 完 | 全 | 想 | 透 | 的 |、 | 不 | 是 | 一 | 次 | 性 | 前 | 二 |
我 | 才 | 想 | 到 |， | 当 | 年 | 我 | 总 | 是 | 独 | 自 | 跑 | 到 | 地 | 坛 | 去 |， | 曾 | 经 | 给 | 母 | 亲 | 出 | 了 | 一 | 个 | 怎 | 样 | 的 | 难 | 题 |。 |
她 | 不 | 是 | 那 | 种 | 会 | 疼 | 爱 | 儿 | 子 | 而 | 不 | 懂 | 得 | 理 | 解 | 儿 | 子 | 的 | 母 | 亲 |。 |
有 | 一 | 回 | 我 | 摇 | 车 | 出 | 了 | 小 | 院 |； | 想 | 起 | 一 | 件 | 什 | 么 | 事 | 又 | 返 | 身 | 回 |
有 | 一 | 次 | 与 | 一 | 个 | 作 | 家 | 朋 | 友 | 聊 | 天 |， | 我 | 问 | 他 | 学 | 写 | 作 | 的 | 最 | 初 | 动 | 机 |
在 | 我 | 的 | 头 | 一 | 篇 | 小 | 说 | 发 | 表 | 的 | 时 | 候 |， | 在 | 我 | 的 | 小 | 说 | 只 | 是 | 到 | 了 | 这 | 时 | 候 |， | 纷 | 纷 | 的 | 往 | 事 | 才 | 在 | 我 | 眼 | 前 | 幻 | 现 | 得 |
摇 | 着 | 轮 | 椅 | 在 | 园 | 中 | 慢 | 慢 | 走 |， | 又 | 是 | 雾 | 罩 | 的 | 清 | 晨 |， | 又 | 是 |
曾 | 有 | 过 | 好 | 多 | 回 |， | 我 | 在 | 这 | 园 | 子 | 里 | 呆 | 得 | 太 | 久 | 了 |， | 母 | 亲 | 就 |
```

前言 |

数据 |， | 数据 |， | 数据 |！ | 想 | 必 | 在 | 等 | 媒 | 介 | 的 | 持 | 续 | 冲 | 击 | 下 |， | 人 | 们 | 的 | 洗 | 礼 |。 | 现 | 实 | 需 | 求 | 推 | 动 | 了 | 对 | 这 | 些 | 数 | 据 | 来 | 自 | 于 | 社 | 交 | 媒 | 体 |、 | “ | 物 | 联 | 网 | ” |) |、 | 传 | 感 | 器 | 等 | 任 | 何 | 大 | 多 | 数 | 数 | 据 | 挖 | 掘 | 的 | 宣 | 传 | 着 | 数 | 据 | 洪 | 水 (| data flood) | 的 | 预 | 言 | 数 | 据 |， | 硬 | 件 | 推 | 销 | 人 | 员 | 会 | 进 | 一 | 步 | 能 | 够 | 满 | 足 | 处 | 理 | 速 | 度 | 的 | 要 | 求 |。 | 对 | 的 |， | 但 | 是 | 我 | 们 | 值 | 得 | 停 | 下 | 务 | 进 | 行 | 适 | 当 | 的 | 再 | 认 | 识 |。 |

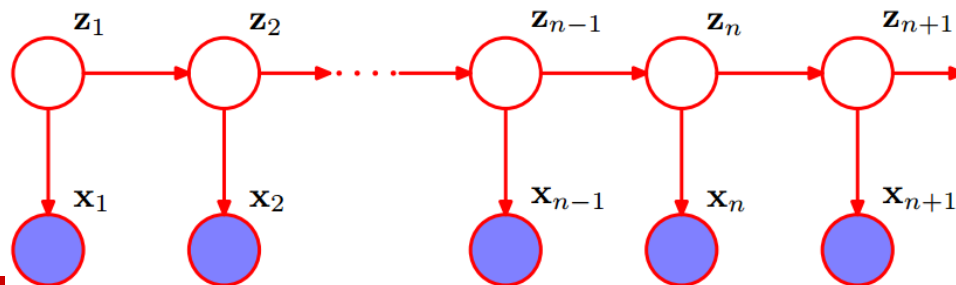
近 | 年 | 来 |， | 数 | 据 | 挖 | 掘 | 和 | 机 | 器 | 学 | 习 | 在 | 我 | 们 | 周 | 围 | 持 | 续 | 火 | 爆 |， | 各 | 种 | 媒 | 体 | 也 | 不 | 断 | 推 | 送 | 着 | 海 | 量 | 的 | 数 | 据 |。 | 仔 | 细 | 观 | 察 | 就 | 能 | 发 | 现 |， | 实 | 际 | 应 | 用 | 中 | 的 | 那 | 些 | 机 | 器 | 学 | 习 | 算 | 法 | 与 | 多 | 年 | 前 | 并 | 没 | 有 | 什 | 么 | 两 | 样 |； | 它 | 们 | 只 | 是 | 在 | 应 | 用 | 的 | 数 | 据 | 规 | 模 | 上 | 有 | 些 | 不 | 同 |。 | 历 | 数 | 一 | 下 | 产 | 生 | 数 | 据 | 的 | 组 | 织 |， | 至 | 少 | 在 | 我 | 看 | 来 |， | 数 | 目 | 其 | 实 | 并 | 不 | 多 |。 | 无 | 非 | 是 | Google |、 | Facebook |、 | Twitter |、 | Netflix | 以 | 及 | 其 | 他 | 为 | 数 | 不 | 多 | 的 | 机 | 构 | 在 | 使 | 用 | 若 | 干 | 学 | 习 | 算 | 法 | 和 | 工 | 具 |， | 这 | 些 | 算 | 法 | 和 | 工 | 具 | 使 | 得 | 他 | 们 | 能 | 够 | 对 | 数 | 据 | 进 | 行 | 测 | 试 | 分 | 析 |。 | 那 | 么 |， | 真 | 正 | 的 | 问 | 题 | 是 |： | “ | 对 | 于 | 其 | 他 | 人 |， | 大 | 数 | 据 | 框 | 架 | 下 | 的 | 算 | 法 | 和 | 工 | 具 | 的 | 作 | 用 | 是 | 什 | 么 | 呢 |？ | ” |

我 | 承 | 认 | 本 | 书 | 将 | 多 | 次 | 提 | 及 | 大 | 数 | 据 | 和 | 机 | 器 | 学 | 习 | 之 | 间 | 的 | 关 | 系 |， | 这 | 是 | 我 | 无 | 法 | 忽 | 视 | 的 | 一 | 个 | 客 | 观 | 问 | 题 |； | 但 | 是 | 它 | 只 | 是 | 一 | 个 | 很 | 小 | 的 | 因 | 素 |， | 终 | 极 | 目 | 标 | 是 | 如 | 何 | 利 | 用 | 可 | 用 | 数 | 据 | 获 | 取 | 数 | 据 | 的 | 本 | 质 |



Jason Bell. *Machine Learning: Hands-On for Developers and Technical Professionals*. Wiley.2015

HMM定义



- 隐马尔科夫模型(HMM, Hidden Markov Model)可用标注问题，在语音识别、NLP、生物信息、模式识别等领域被实践证明是有效的算法。
- HMM是关于时序的概率模型，描述由一个隐藏的马尔科夫链生成不可观测的状态随机序列，再由各个状态生成观测随机序列的过程。
- 隐马尔科夫模型随机生成的状态随机序列，称为状态序列；每个状态生成一个观测，由此产生的观测随机序列，称为观测序列。
 - 序列的每个位置可看做是一个时刻。

HMM的3个基本问题

□ 概率计算问题：前向-后向算法——动态规划

■ 给定模型 $\lambda = (A, B, \pi)$ 和观测序列 $O = \{o_1, o_2, \dots, o_T\}$ ，计算模型 λ 下观测序列 O 出现的概率 $P(O|\lambda)$

□ 学习问题：Baum-Welch算法(状态未知)——EM

■ 已知观测序列 $O = \{o_1, o_2, \dots, o_T\}$ ，估计模型 $\lambda = (A, B, \pi)$ 的参数，使得在该模型下观测序列 $P(O|\lambda)$ 最大

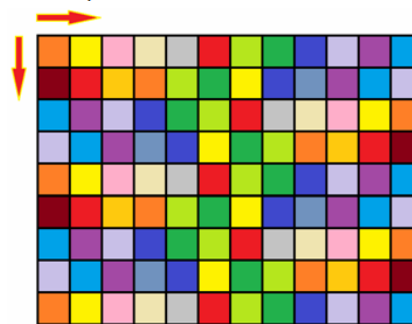
□ 预测问题：Viterbi算法——动态规划

■ 解码问题：已知模型 $\lambda = (A, B, \pi)$ 和观测序列 $O = \{o_1, o_2, \dots, o_T\}$ 求给定观测序列条件概率 $P(I|O, \lambda)$ 最大的状态序列 I

Viterbi算法

- Viterbi算法实际是用动态规划解HMM预测问题，用DP求概率最大的路径(最优路径)，这是一条路径对应一个状态序列。
- 定义变量 $\delta_t(i)$ ：在时刻t状态为i的所有路径中，概率的最大值。

给定m*n的矩阵，每个位置是一个非负整数，从左上角开始，每次只能朝右和下走，走到右下角，求总和最小的路径。



Viterbi算法

□ 定义：

$$\delta_t(i) = \max_{i_1, i_2, \dots, i_{t-1}} P(i_t = i, i_{t-1}, \dots, i_1, o_t, \dots, o_1 | \lambda)$$

□ 递推：

$$\delta_1(i) = \pi_i b_{io_1}$$

$$\delta_{t+1}(i) = \max_{i_1, i_2, \dots, i_t} P(i_{t+1} = i, i_t, \dots, i_1, o_{t+1}, \dots, o_1 | \lambda)$$

$$= \max_{1 \leq j \leq N} (\delta_t(j) a_{ji}) b_{io_{t+1}}$$

□ 终止：

$$P^* = \max_{1 \leq i \leq N} \delta_T(i)$$

例

□ 考察盒子球模型，观测向量 O = “红白红”，试求最优状态序列。

$$\pi = \begin{pmatrix} 0.2 \\ 0.4 \\ 0.4 \end{pmatrix} \quad A = \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix} \quad B = \begin{bmatrix} 0.5 & 0.5 \\ 0.4 & 0.6 \\ 0.7 & 0.3 \end{bmatrix}$$

解：观测向量 O ="红白红" $\pi = \begin{pmatrix} 0.2 \\ 0.4 \\ 0.4 \end{pmatrix}$ $B = \begin{bmatrix} 0.5 & 0.5 \\ 0.4 & 0.6 \\ 0.7 & 0.3 \end{bmatrix}$

□ 初始化：

□ 在 $t=1$ 时，对于每一个状态 i ，求状态为 i 观测到 o_1 =红的概率，记此概率为 $\delta_1(t)$

$$\delta_1(i) = \pi_i b_{io_1} = \pi_i b_{i\text{红}}$$

□ 求得 $\delta_1(1)=0.1$

□ $\delta_1(2)=0.16$

□ $\delta_1(3)=0.28$

解：观测向量 O ="红白红" $A = \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix} B = \begin{bmatrix} 0.5 & 0.5 \\ 0.4 & 0.6 \\ 0.7 & 0.3 \end{bmatrix}$

- 在 $t=2$ 时，对每个状态 i ，求在 $t=1$ 时状态为 j 观测为红并且在 $t=2$ 时状态为 i 观测为白的路径的最大概率，记概率为 $\delta_2(t)$ ，则：

$$\delta_{t+1}(i) = \max_{1 \leq j \leq 3} (\delta_1(j) a_{ji}) b_{io_2} = \max_{1 \leq j \leq 3} (\delta_1(j) a_{ji}) b_{i白}$$

- 求得

$$\delta_2(1) = \max_{1 \leq j \leq 3} (\delta_1(j) a_{j1}) b_{1白}$$

$$= \max \{0.10 \times 0.5, 0.16 \times 0.3, 0.28 \times 0.2\} \times 0.5 = 0.028$$

- 同理：

■ $\delta_2(2) = 0.0504$

■ $\delta_2(3) = 0.042$

解：观测向量 O ="红白红"

□ 同理，求得

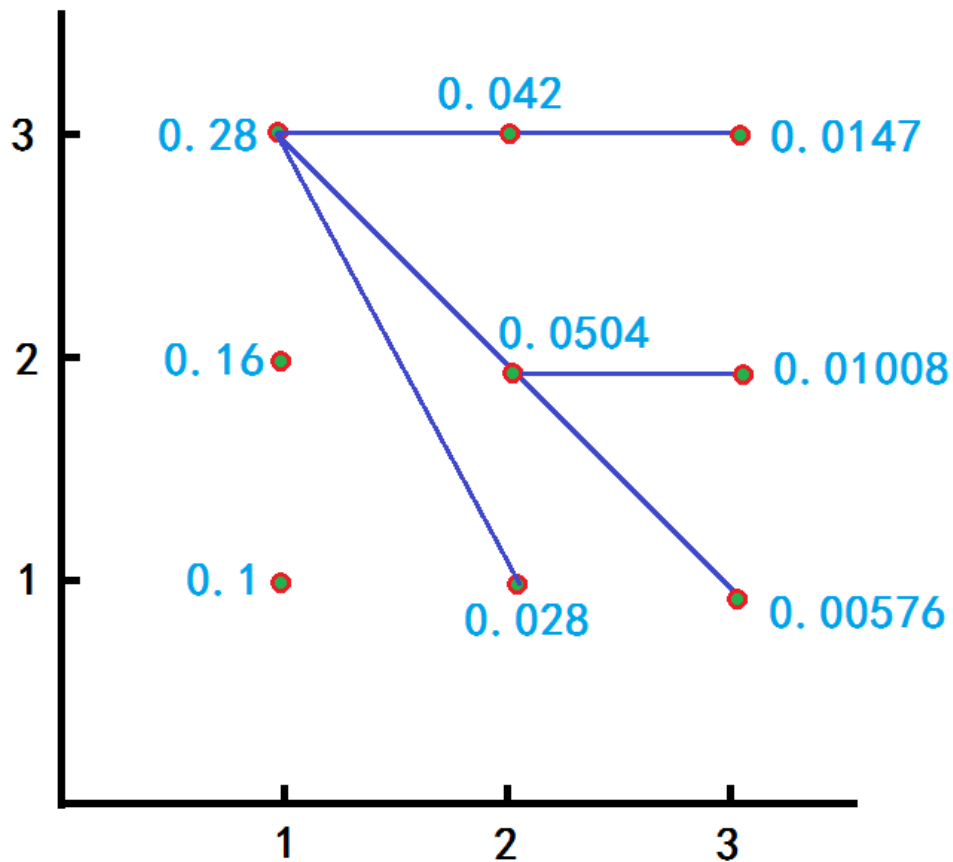
□ $\delta_3(1)=0.00756$

□ $\delta_3(2)=0.01008$

□ $\delta_3(3)=0.0147$

□ 从而，最大是 $\delta_3(3)=0.0147$ ，根据每一步的最大，得到序列是(3,3,3)

求最优路径图解



思考题：百数问题

□ 在1,2,3,4,5,6,7,8,9(顺序不能变)数字之间插入运算符+或者运算符-或者什么都不插入，使得计算结果是100。

■ 如： $1+2+34-5+67-8+9=100$

□ 请输出所有的可行运算符方式。

猜猜看

□ 最终的拼团人数是素数还是合数？

□ 思考：

■ 筛法求素数

■ 赔率



微信扫码 立即参团

BAT 8月29日开课 邹博主讲 面试算法特训班



《BAT面试算法特训班》（七天无理由退款）邹博主讲

原 价 ￥899.00

拼团价 ￥399.00 50人以上

¥299.00 100人以上 (当前价)

团长:小象学院

893 人参团

剩余时间 3天内

参团

我们在这里

□ <http://wenda.ChinaHadoop.cn>

■ 视频/课程/社区

□ 微博

■ @ChinaHadoop

■ @邹博_机器学习

□ 微信公众号

■ 小象学院

■ 大数据分析挖掘

互联网新技术在线教育领航者

小象问答 搜索标题、用户 全站内容搜索 提问 首页 动态 发现 话题 通知

全部 招聘求职 机器学习 大数据平台技术 DCon 大数据行业应用 NoSQL数据库 数据科学 江湖救急

发现 最新 推荐 热门 等待回复

graphviz has no attribute 'write' 贡献
邹博 回复了问题 • 2 人关注 • 1 个回复 • 3 次浏览 • 2017-04-09 15:48

sklearn中如何理解Pipeline机制 贡献
数据分析与数据挖掘 邹博 回复了问题 • 2 人关注 • 1 个回复 • 28 次浏览 • 2017-04-09 15:39

关于9.Logistic回归的ppt中第9页的对数线性函数 贡献
机器学习 邹博 回复了问题 • 3 人关注 • 3 个回复 • 39 次浏览 • 2017-04-09 15:35

关于“贝叶斯估计中，最大后验概率估计就是结构化风险最小化的例子：当模型是条件概率分布，损失函数为对数损失函数，模型的复杂度由模型的先验概率表示，结构化风险最小化就等价于最大后验概率估计” 贡献
机器学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 26 次浏览 • 2017-04-09 15:27

关于连续值的预测 贡献
咨询 邹博 回复了问题 • 2 人关注 • 1 个回复 • 31 次浏览 • 2017-04-09 15:24

拉格朗日对偶函数为什么一定是凸函数 贡献
数据科学 邹博 回复了问题 • 2 人关注 • 2 个回复 • 26 次浏览 • 2017-04-09 15:20

梯度下降公式中的斯堪J 是 贡献
机器学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 29 次浏览 • 2017-04-09 15:17

深度学习适合做预测吗？ 贡献
深度学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 27 次浏览 • 2017-04-09 15:15

关于6.4PCA_FeatureSelection.py中plt.legend的参数疑问 贡献
机器学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 28 次浏览 • 2017-04-09 15:04

@邹博 有哪些可以下载数据源的网站？ 贡献
数据分析与数据挖掘 邹博 回复了问题 • 4 人关注 • 1 个回复 • 31 次浏览 • 2017-04-09 14:53

LDA主题模型 贡献
机器学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 29 次浏览 • 2017-04-09 14:45

代码10.6bagging_ridged老师提到了采样率设为0.2能够使峰值部分的数据被体现出来。这是为什么呢？ 贡献
机器学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 22 次浏览 • 2017-04-09 14:26

GraphViz's executables not found 贡献
机器学习 邹博 回复了问题 • 3 人关注 • 2 个回复 • 23 次浏览 • 2017-04-09 13:47

决策树中关于feature_importances代码的问题 贡献
机器学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 6 次浏览 • 2017-04-09 13:11

专题
招聘求职
大数据行业应用
数据科学
系统与编程
云计算技术

热门话题 更多 >
机器学习 907 个问题, 230 人关注
spark 387 个问题, 172 人关注
hadoop 1059 个问题, 155 人关注
python数据分析 171 个问题, 28 人关注
数据分析与数据挖掘 54 个问题, 111 人关注

热门用户 更多 >
小心巴 14 个问题, 0 次赞同
又又V 45 个问题, 22 次赞同
铁甲无声 10 个问题, 0 次赞同
带刀锦衣卫 13 个问题, 0 次赞同

感谢大家！

恳请大家批评指正！

课前甜点 – 七夕

2017七夕直男考卷

你和女朋友约好一起去逛超市，她画好了妆，从衣柜里拿出两件衣服，问你哪件更好看，你随口说了句“都好看”。出门后两人来到了超市，路过冰淇淋柜时，女朋友脚步放慢了，看的出来她很想吃，但你想到她最近一直说要减肥，就说“别吃了，你不是说要减肥吗？”



2017七夕直男考卷

于是两人回了家，女朋友说“好热啊！”，“那你开空调就好了啊”你随口回答，然后你便打开电视播到中央五，看了一会足球，女朋友有点不高兴了，说“一天就知道看足球，吵死了，就不能不看足球吗？”你想可能吵到女朋友了，就戴上耳机用手机看起了游戏直播，这时，女朋友摔门而出……

