### 相似度

#### 相似度的定义

相似度就是比较两个事物的相似性。一般通过计算事物的特征之间的距离，如果距离小，那么相似度大；如果距离大，那么相似度小。

问题定义：有两个对象X,Y,都包含N维特征，X=(x1,x2,x3,……..,xn),Y=(y1,y2,y3,……..,yn),计算X和Y的相似性。

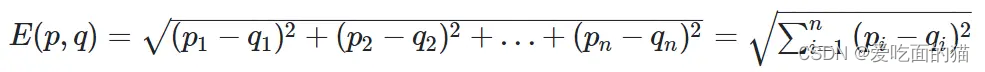
在数据分析和数据挖掘以及搜索引擎中，我们经常需要知道个体间差异的大小，进而评价个体的相似性和类别。常见的比如数据分析中比如相关分析，数据挖掘中的分类聚类（K-Means等）算法，搜索引擎进行物品推荐时。

#### 相似度的计算方法

##### **（1）**欧几里得距离（Eucledian Distance）

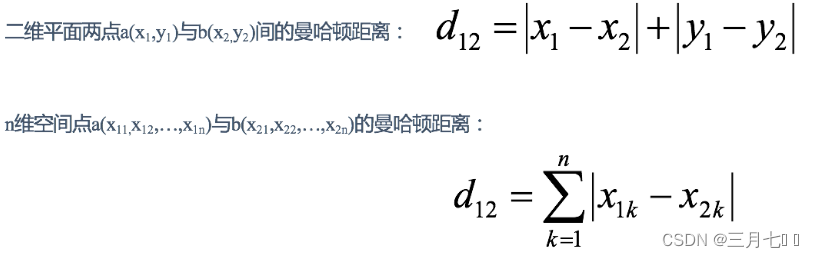
欧氏距离是最常用的距离计算公式，衡量的是多维空间中各个点之间的**绝对距离**，当数据很稠密并且连续时，这是一种很好的计算方式。

因为计算是基于各维度特征的绝对数值，所以欧氏度量需要保证各维度指标在**相同的刻度级别**，比如对身高（cm）和体重（kg）两个单位不同的指标使用欧式距离可能使结果失效。



##### **（2）**曼哈顿距离（Manhattan Distance）

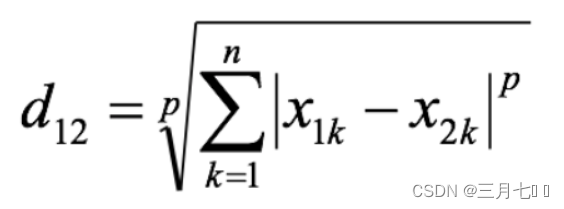
曼哈顿距离——两点在南北方向上的距离加上在东西方向上的距离，即d(i,j)=|xi-xj|+|yi-yj|。对于一个具有正南正北、正东正西方向规则布局的城镇街道，从一点到达另一点的距离正是在南北方向上旅行的距离加上在东西方向上旅行的距离，因此，曼哈顿距离又称为**出租车距离**。曼哈顿距离不是距离不变量，当坐标轴变动时，点间的距离就会不同。曼哈顿距示意图在早期的计算机图形学中，屏幕是由像素构成，是整数，点的坐标也一般是整数，原因是浮点运算很昂贵，很慢而且有误差，如果直接使用AB的欧氏距离（欧几里德距离：在二维和三维空间中的欧氏距离的就是两点之间的距离），则必须要进行浮点运算，如果使用AC和CB，则只要计算加减法即可，这就大大提高了运算速度，而且不管累计运算多少次，都不会有误差。

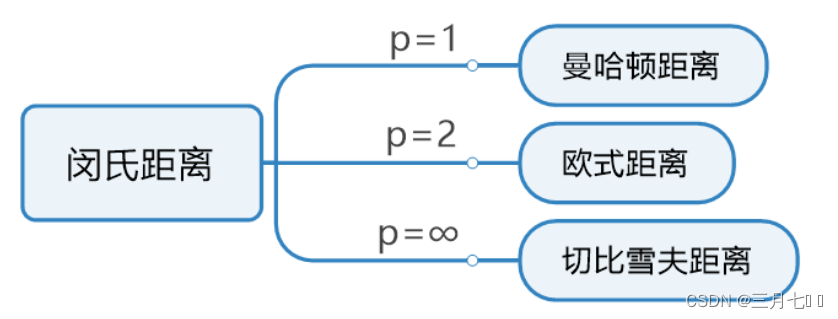
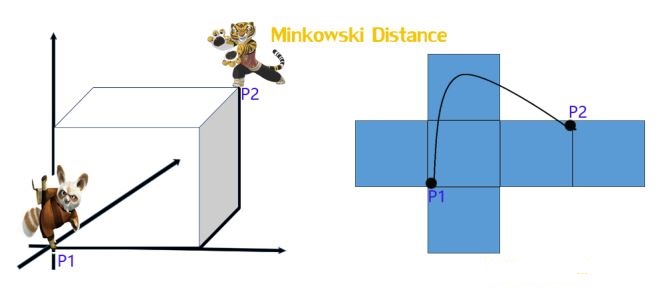


##### （3）明可夫斯基距离（Minkowski distance）

也被称为 闵氏距离。它不仅仅是一种距离，而是将多个距离公式（曼哈顿距离、欧式距离、切比雪夫距离）总结成为的一个公式。

两个n维变量a(x11,x12,…,x1n)与b(x21,x22,…,x2n)间的闵可夫斯基距离定义为：

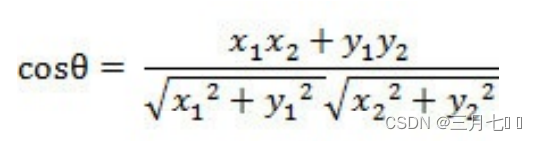




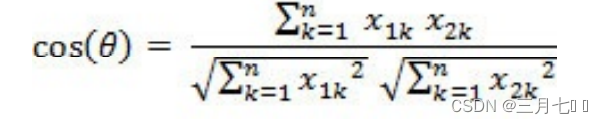
##### **（4）**（余弦相似度）Cosine Similarity

余弦相似度用向量空间中两个向量夹角的余弦值作为衡量两个个体间差异的大小。相比距离度量，余弦相似度更加注重两个向量在方向上的差异，而非距离或长度上。

⼆维空间中向量A(x1,y1)与向量B(x2,y2)的夹⻆余弦公式：



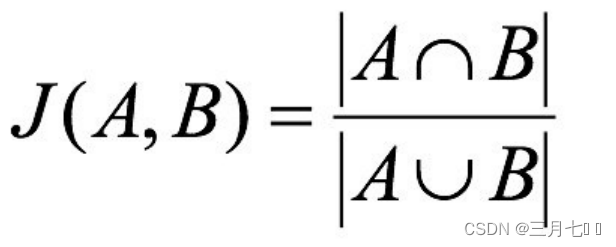
两个n维样本点a(x11,x12,…,x1n)和b(x21,x22,…,x2n)的夹⻆余弦为：



##### **（5）**Jaccard Similarity

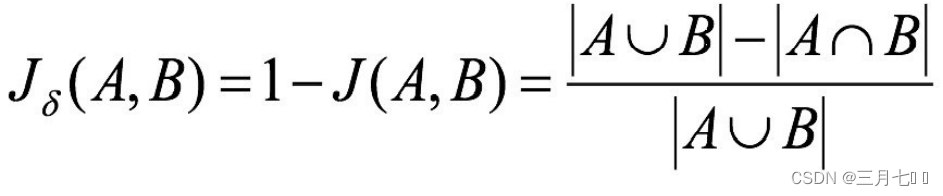
Jaccard系数主要用于计算符号度量或布尔值度量的个体间的相似度，因为个体的特征属性都是由符号度量或者布尔值标识，因此无法衡量差异具 体值的大小，只能获得“是否相同”这个结果，所以Jaccard系数只关心个体间共同具有的特征是否一致这个问题。

两个集合A和B的交集元素在A，B的并集中所占的⽐例，称为两个集合的杰卡德相似系数，⽤符号J(A,B)表示：



杰卡德距离(Jaccard Distance)：与杰卡德相似系数相反，⽤两个集合中不同元素占所有元素

的⽐例来衡量两个集合的区分度：



##### **（6）**皮尔森相关系数(Pearson Correlation Coefficient)

又称相关相似性，通过Peason相关系数来度量两个用户的相似性。计算时，首先找到两个用户共同评分过的项目集，然后计算这两个向量的相关系数。

IMG_256

### 旋转有序序列的查找

#### **1.**理解问题

对于旋转有序序列（即围绕有序序列中某个元素旋转得到如7,8,9,1,2,3,4,5,6），设计一个查找算法，查找成功返回它的位置，否则返回失败标志，并分析算法。

因为序列一定可以被分成两部分有序的序列，所以想到二分查找的方法。

例：依次输入数组元素个数、数组中每个元素、要查询的元素。

9 7 8 9 1 2 3 4 5 6 10

#### **2.**算法设计

While循环条件：L<=R

进入循环先计算mid

如果arr[mid]==x，直接返回mid  
如果arr[mid]<arr[R]说明右侧有序，如果arr[mid]<x&&x<=arr[R]，则L=mid+1;否则R=mid-1

另一侧同理

#### 算法代码

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int BiSearch(int arr[],int length,int x)

{

    int L=1,R=length,mid=(L+R)/2;

    while (L<=R)

    {

        mid=(L+R)/2;

        if (arr[mid]==x)return mid;

        if (arr[mid]<arr[R])  //右侧有序

        {

            if (arr[mid]<x&&x<=arr[R])

            {

                L=mid+1;

            }else R=mid-1;

        }else{              //左侧有序

            if (arr[mid]>x&&arr[L]<=x)

            {

                R=mid-1;

            }else L=mid+1;

        }

    }

    return 0;

}

int main(){

    int length;

    printf("请依次输入元素个数、元素、待查询的元素：\n");

    scanf("%d", &length);

    int arr[100];

    for (int i = 1; i <= length; i++)

    {

        scanf("%d", &arr[i]);

    }

    int x;

    scanf("%d", &x);

    int res=BiSearch(arr, length,x);

    if(res==0)printf("Not Found!\n");

    else printf("%d\n", res);

}

测试结果：

