# **高斯消元快速入门**

## **一、基本描述**

学习一个算法/技能，首先要知道它是干什么的，那么高斯消元是干啥的呢？

****高斯消元主要用来求解线性方程组****，也可以****求解矩阵的秩，矩阵的逆****。在ACM中是一个****有力的数学武器****.

它的时间复杂度是****n^3****，主要与方程组的个数，未知数的个数有关。

那么什么是****线性方程组呢?****   
简而言之就是有多个未知数，并且每个未知数的次数均为一次，这样多个未知数组成的方程组为线性方程组。

## **二、算法过程**

其实高斯消元的过程就是****手算解方程组的过程****，回忆一下小的时候怎么求解方程组：加减消元，消去未知数，如果有多个未知数，就一直消去，****直到得到类似kx=b（k和b为常数，x为未知数）的式子****，就可以求解出未知数x，然后我们****回代****，依次求解出各个未知数的值，就解完了方程组。   
换句话说，分两步：

1. 加减消元

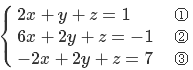
2. 回代求未知数值

高斯消元就是这样的一个过程。

下面通过一个小例子来具体说明

### **0.求解方程组**

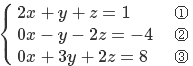
**有这样一个三元一次方程组：**

****

### **1.消去x**

**①×(−3)+②得到  0x−y−2z=−4**

**①+③得到  0x+3y+2z=8**

**从而得到**

### **2.消去y**

**②×3+③得到  0x+0y−4z=−4**

**进而得到**

****

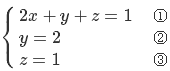
**至此，我们已经求解出来了  z=1**

**下一步我们进行回代过程**

### **3.回代求解y**

**将z=1带入②，求得 y=2**

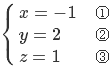
**进而得到**

****

### **4.回代求解x**

**将z=1,y=2带入①，求得 x=−1**

**最终得到**

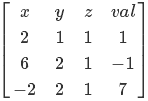
****

**至此，整个方程组就求解完毕了。**

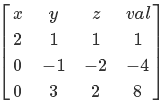
## **三、再解算法**

**对于方程组，其系数是具体存在矩阵(数组)里的，下面在给出实际在矩阵中的表示**

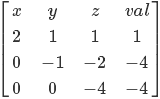
### **0.求解方程组**

****

### **1.消去x**

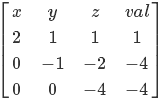
****

### **2.消去y**

****

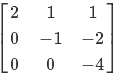
### **3.回代求解y**

回代的时候，记录各个变量的结果将保存在另外一个数组当中，故保存矩阵的数组值不会发生改变，该矩阵主要进行消元过程。

****

## **四、再再解算法**

说了这么多，其实有一些情况我们还没有说到。   
通过上述的消元方法，其实我们比较希望得到的是一个上三角阵(省去了最后的val)

****

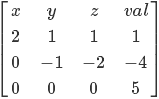
**下面问题来了：**

**Q1：系数不一定是整数啊？**

**A1：这时候数组就要用到浮点数了！不能是整数！**

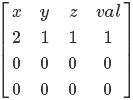
**Q2：什么时候无解啊？**

**A2：消元完了，发现有一行系数都为0，但是常数项不为0，当然无解啦！比如：**

****

**Q3：什么时候多解啊？**

**A3：消元完了，发现有好几行**系数为0，常数项也为0，这样就多解了！有几行为全为0，就有几个****自由元****，所谓自由元，就是这些变量的值可以随意取，有无数种情况可以满足给出的方程组，比如：

****

**您说这x,y,z不是无数组解嘛！**

**Q4：那什么时候解是唯一的啊！**

**A4：您做一下排除法，不满足2和3的，不就是解释唯一的嘛！其实也就是说我们的系数矩阵可以**化成****上三角阵****。

## **五、代码实现**

****例：**[ZOJ3645](http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemCode=3645)**

****题意：****高斯消元模板题(浮点型)

高斯消去法的主元，但绝对值很小时，用绝对值小的数做除数，会导致其它元素数量级的严重增长和传入误差的扩大

/\*\*

高斯消元求解线性方程组.

\*/

#include <iostream>

#include <cstdio>

#include <cstring>

#include <cmath>

#include <algorithm>

using namespace std;

///高斯消元模板

const double eps = 1e-12;

const int Max\_M = 15; ///m个方程,n个变量

const int Max\_N = 15;

int m, n;

double Aug[Max\_M][Max\_N+1]; ///增广矩阵

bool free\_x[Max\_N]; ///判断是否是不确定的变元

double x[Max\_N]; ///解集

int sign(double x){ return (x>eps) - (x<-eps);}

/\*\*

返回值:

-1 无解

0 有且仅有一个解

>=1 有多个解,根据free\_x判断哪些是不确定的解

\*/

int Gauss(){

int i,j;

int row,col,max\_r;

for(row=0,col=0; row<m&&col<n; row++,col++) {

max\_r = row;

for(i = row+1; i < m; i++){ ///找到当前列所有行中的最大值(做除法时减小误差)

if(sign(fabs(Aug[i][col])-fabs(Aug[max\_r][col]))>0)

max\_r = i;

}

if(max\_r != row) ///将该行与当前行交换

for(j = row; j < n+1; j++)

swap(Aug[max\_r][j],Aug[row][j]);

if(sign(Aug[row][col])==0){ ///当前列row行以下全为0(包括row行)

row--;

continue;

}

for(i = row+1; i < m; i++){

if(sign(Aug[i][col])==0)

continue;

double ta = Aug[i][col]/Aug[row][col];

for(j = col; j < n+1; j++)

Aug[i][j] -= Aug[row][j]\*ta;

}

}//上面循环正常结束时，row==m，如果row<m,说明存在0...0的情况

for(i = row; i < m; i++) //col=n存在0...0,a的情况,无解

if(sign(Aug[i][col]))

return -1;

if(row < n) { ///存在0...0,0的情况,有多个解,自由变元个数为n-row个

for(i = row-1; i >=0; i--){

int free\_num = 0; ///自由变元的个数

int free\_index; ///自由变元的序号

for(j = 0; j < n; j++)

if(sign(Aug[i][j])!=0 && free\_x[j])

free\_num++,free\_index=j;

if(free\_num > 1) continue; ///该行中的不确定的变元的个数超过1个,无法求解,它们仍然为不确定的变元

///只有一个不确定的变元free\_index,可以求解出该变元,且该变元是确定的

double tmp = Aug[i][n];

for(j = 0; j < n; j++)

if(sign(Aug[i][j])!=0 && j!=free\_index)

tmp -= Aug[i][j]\*x[j];

x[free\_index] = tmp/Aug[i][free\_index];

free\_x[free\_index] = false;

}

return n-row;

}

///有且仅有一个解,严格的上三角矩阵(n==m)

for(i = n-1; i >= 0; i--){

double tmp = Aug[i][n];

for(j = i+1; j < n; j++)

if(sign(Aug[i][j])!=0)

tmp -= Aug[i][j]\*x[j];

x[i] = tmp/Aug[i][i];

}

return 0;

}

///模板结束

int main(){

int i,j;

int t;

double a[12][12];

scanf("%d",&t);

while(t--){

memset(Aug,0.0,sizeof(Aug));

memset(x,0.0,sizeof(x));

memset(free\_x,1,sizeof(free\_x)); ///都是不确定的变元

for(i = 0; i < 12; i++)

for(j = 0; j < 12; j++)

scanf("%lf",&a[i][j]);

double sum=0;

for(int i=0;i<11;i++)

sum+=a[11][i]\*a[11][i];

for(int i=0;i<11;i++){

for(int j=0;j<11;j++){

Aug[i][j]=2\*(a[i][j]-a[11][j]);

Aug[i][11]+=a[i][j]\*a[i][j];

}

Aug[i][11]+=-a[i][11]\*a[i][11]+a[11][11]\*a[11][11]-sum;

}

m = n = 11;

Gauss();

for(int i = 0; i < n; i++){

printf("%.2lf",x[i]);

printf("%c",i==n-1?'\n':' ');

}

}

return 0;

}