1. **页码：前言3 • 行数：20 • 印次 1**

前言第三页应该是：

    读者对所学知识**的掌握**情况，...

1. **页码：5 • 行数：2,5,8,10,11,19 • 印次 1**

2,5行：修改红色部分

total=0; //运行1次

for( i =1; i<= n; i++) //运行n+1次

{

sum=sum+i; //运行n次

for(j =1; j<= n; j++) //运行n\*(n+1)次

total=total+i\*j; //运行n\*n次

}

8-11行：

把算法所有语句的运行次数加起来：1+1+*n*+1+*n*+*n*×(*n*+1)+*n*×*n*，可以用一个函数*T*(*n*)表达：

*T*(*n*)=2*n*2+3*n*+3

当*n*足够大时，例如*n*=105时，*T*(*n*)=2×1010+3×105+3，我们可以看到算法运行时间主要取决于第一项，后面的甚至可以忽略不计。

19行：



1. **页码：13 • 行数：-7 • 印次 1**

int \*a=new int[n+1];//定义一个长度为n+1的数组，0空间未使用

1. **页码：17-19 • 行数： • 印次 1**

算法1-11、算法1-12、算法1-13三个算法末尾大括号前面加一行return 0;

例如：其它两个算法也是如此添加：

//算法1-11

#include<iostream>

int main()

{

…..

return 0;

}

1. **页码：19 • 行数：14 • 印次 1**

直接用下面内容替换掉这段：

（5）问题的进一步讨论

 此题算法还可考虑求**2、3、5、6**的最小公倍数*n*，然后令*t*=*n*−1，判断*t*≡0(*m*od 7)是否成立，若不成立则*t*=*t*+*n*，再进行判别，直到选出满足条件的*t*为止。

解释：因为*n*是2、3、5、6的最小公倍数，减1后，分别除以2、3、5、6，余数必然为1、2、4、5，正好满足前四个条件，再继续判断是否满足第五个条件即可。

2、3、5、6的最小公倍数n=30。

*t*=*n*-1=29，*t*≡0(*m*od 7)不成立；

*t*= *t*+*n*=59，*t*≡0(*m*od 7)不成立；

*t*= *t*+*n*=89，*t*≡0(*m*od 7)不成立；

*t*= *t*+*n*=119，*t*≡0(*m*od 7)成立。

我们可以看到这一算法判断**4次**即成功，但是，求多个数的最小公倍数需要多少时间复杂度，是不是比上面的算法更优呢？结果如何请大家动手试一试。

1. **页码：20 • 行数：24 • 印次 1**

i后面缺少一个左括号：

for(j=2;j<=(int)(sqrt((double)i));j++)

1. **页码：22 • 行数：21 • 印次 1**

修改红色部分：

（2）时间复杂度通常用渐近上界符号***O*(*f*(*n*))**表示。

1. **页码：30 • 行数：26 • 印次 1**

temp去掉e，改为tmp:

double tmp=0.0;

1. **页码：53 • 行数：14，-11 • 印次 1**

14行：!s[j]此处应为!flag[j]。

if(!flag[j]&& map[t][j]<INF) //**!flag[j]**表示j在V-S中

-11行：main函数中，此行多了一个分号，删除

cout <<"请输入小明所在的位置: "<<endl； ；

1. **页码：56 • 行数：4,7,8,11 • 印次 1**

**7**处改动，v改为u：

struct Node{

int **u**,step; // **u**为顶点，step为源点到顶点**u**的最短路径

Node(){};

Node(int a,int sp){

**u** = a; //参数传递，**u**为顶点

step = sp; //参数传递，step为源点到顶点**u**的最短路径

}

bool operator < (const Node& a)const{

return step > a.step; //重载 <，step(源点到顶点**u**的最短路径)最小值优先

}

};

**行数：19,23(4**处改动，v改为u**)**

struct Node{

int **u**,step; // **u**为顶点，step为源点到顶点**u**的最短路径

};

那么在变量参数赋值时，需要这样赋值：

Node vs ; //先定义一个Node结点类型变量

 vs. **u** =3 ,vs.step = 5; //分别对该变量的两个成员进行赋值

**行数：36(1**处改动，v改为u)

上面语句等价于：

vs. **u** =3 ,vs.step = 5;

1. **页码：57 • 行数：3 • 印次 1**

//参数i传递给顶点**u**， dist[i]传递给step

1. **页码：66 • 行数：-7 • 印次 1**

左儿子改为左孩子：

的左孩子为ti，右孩子为tj，zk的权值为ti和tj的权值之和。

1. **页码：75 • 行数：6 • 印次 1**

0改为-1:

x1=x2= **-1**;

1. **页码：79 • 行数：12 • 印次 1**

2-68应该是2-6**0**

1. **页码：80 • 行数：22 • 印次 1**

添加“为空”两个字：

"如果集合V-U**为空**，算法结束"。

1. **页码：82 • 行数：16 • 印次 1**

26改为28：

(5)更新

。。。。6号顶点到U集合的最邻近点为1，最邻近距离为28，

（6）找最小

1. **页码：87 • 行数：3，-9 • 印次 1**
2. 初始化。。。；初始化数组lowcost,

（3）更新lowcost和closest数组

s[t] = true; //否则， 将 t 加入集合U

1. **页码：89 • 行数：24 • 印次 1**

注释少了一个t

应为“//计算最后的lowcos**t**的总和”。

1. **页码：90 • 行数：-6 • 印次 1**

构造最小生成树还有一种算法，Kruskal算法：

1. **页码：95 • 行数：47 • 印次 1**

1改为0，删除=:

for(int i=**0**; **i<m**; i++)

cin >> e[i].u>> e[i].v >>e[i].w;

1. **页码：97 • 行数：22 • 印次 1**

1改为0，删除=:

for(int i=**0**; **i<m**; i++)

cin >> e[i].u>> e[i].v >>e[i].w;

1. **页码：103 • 行数：5 • 印次 1**

low改为high:

围缩小到子问题*S*[*middle*+1…*high*]，令*low*=*middle*+1，如图3-6所示。

1. **页码：105 • 行数：28 • 印次 1**

1739中间缺少空格，改为：

“请依次输入数列中的元素： 60 **17 39** 15 8 34 30 45 5 52 25”。

1. **页码：106 • 行数：32，34 • 印次 1**

只修改红色部分即可，high改为middle，两个return后面加空格：

else if(x<s[middle]) //x 小于s[middle]，则从前半部分查找

return recursionBS (s[],x, low, **middle** -1)

else

return recursionBS (s[],x, middle +1low, high)

1. **页码：111 • 行数：7,8,9,10 • 印次 1**

7,8,9,10行缩进两个空格，10行后面插入一行delete []B;

改为：

while(i <= mid) B[k++] = A[i++]; //对子序列A[low:middle]剩余的依次处理

while(j <= high) B[k++] = A[j++]; //对子序列A[middle+1:high]剩余的依次处理

for(i = low, k = 0; i <= high; i++) //将合并后的序列复制到原来的A[]序列

A[i] = B[k++];

**delete []B;**

1. **页码：111 • 行数：-3 • 印次 1**

**-3**行后面插入一行delete []B;

改为：

for(i=low, k=0; i<= high; i++)

A[i] = B[k++];

**delete []B;**

1. **页码：112 • 行数：3 • 印次 1**

缩进问题，3行的”{“向右缩进4位，与if对齐：

if(low < high)

**{**

1. **页码：127 • 行数： 3 • 印次 3**

图3-53中，改动红色部分：

图3-54中，改动红色部分：



1. **页码：129 • 行数：23和27 • 印次 1**

23行注释： //相当于cp(**&a,**&ah,2,3);

27行注释： //相当于cp(**&a,**&al,0,2);

1. **页码：131 • 行数：16，-10 • 印次 1**

16行注释改为：//b数加上次幂的总长度，上例中**b**len=5。

-10行b改为pb：

if(i <pb->l)

tb = pb->s[i]; //从b数组中第0位开始取数字

1. **页码：147 • 行数：5 • 印次 1**

缺少一个字“自”：

(3)自底向上计算最优值，并记录最优值和最优策略

1. **页码：149 • 行数：5 • 印次 1**

j=2: A=A, 则取左上角数值加1，c[1][2]=c[0][1]+1= **1**, 最优策略来源b[1][2] = 1,

1. **页码：152 • 行数：1-9 • 印次 1**

改动红色部分：

b[7][5]=1，向左上角找b[6][4]，返回时输出s1[6]=“B”;

b[6][4]=3，向上找b[5][4]；

b[5][4]=1，向左上角找b[4][3]，返回时输出s1[4]=“D”;

b[4][3]=2，向左找b[4][2]；

b[4][2]=1，向左上角找b[3][1]，返回时输出s1[3]=“A”;

b[3][1]=3，向上找b[2][1]；

b[2][1]=1，向左上角找，返回时输出s1[1]=“B”;

b[1][0]中列为0，算法停止，返回，输出最长公共子序列为BADB。如图4-16所示。

1. **页码：152-156 • 印次 1**

这几页的代码中，所有大写字母'I'都应改为小写字母'i'。

152页3处：

Void LCSL()

{

int **i**,j;

for(**i** = 1; **i** <= len1;i++) //控制s1序列

153页2处：

Void print(int **i**, int j)//根据记录下来的信息构造最长公共子序列（从b[i][j]开始递推）

{

if(i==0 || j==0) return;

if(b[i][j]==1)

{

print(i-1,j-1);

cout<<s1[i-1];

}

else if(b[i][j]==2)

print(**i**,j-1);

else print(i-1,j);

}

154页6处：

void LCSL()

{

int **i**,j;

for(**i** = 1; **i** <= len1;i++)//控制s1序列

for(j = 1;j <= len2;j++)//控制s2序列

{

if(s1[i-1]==s2[j-1])

{//如果当前字符相同，则公共子序列的长度为该字符前的最长公共序列+1

c[i][j] = c[i-1][j-1]+1;

b[i][j] = 1;

}

else

{

if(c[i][j-1]>=c[i-1][j])

{

c[i][j] = c[i][j-1];

b[i][j] = 2;

}

else

{

c[i][j] = c[i-1][j];

b[i][j] = 3;

}

}

}

}

void print(int **i**, int j)//根据记录下来的信息构造最长公共子序列（从b[i][j]开始递推）

{

if(i==0 || j==0) return;

if(b[i][j]==1)

{

print(i-1,j-1);

cout<<s1[i-1];

}

else if(b[i][j]==2)

print(**i**,j-1);

else

print(i-1,j);

}

int main()

{

int **i**,j;

cout << “输入字符串s1:”<<endl;

cin >> s1;

cout << “输入字符串s2:”<<endl;

cin >> s2;

155页2处：

len1 = strlen(s1);//计算两个字符串的长度

len2 = strlen(s2);

for(**i** = 0; **i** <= len1;i++)

1. **页码：154 • 行数：1 • 印次 1**

\*ons tint N=1002;

改为:

**const** int N=1002;

1. **页码：156 • 行数：12-17 • 印次 1**

例如两个字符串FAMILY和FRAME，有两种对齐方式：

改为：

例如两个字符串FAMILY和FRAME，有**多**种对齐方式，**例：**

F - A M I L Y - F A M I L Y - F A M I L Y

F R A M E F R A M E F R A M - - E

第一种对齐需要付出的代价：4，插入R，I替换为E，删除L,Y。

第二种对齐需要付出的代价：5，插入**F**，F替换为R，I替换为E，删除L,Y。

**第三种对齐需要付出的代价：5，插入F，F替换为R，删除I, L，Y替换为E。**

1. **页码：159-160 • 行数：12-21 • 印次 3**

159页12行：“加”字改为“**+**”，“这”字前面添加逗号“**，**”：

即：取上面+1，左侧+1，左上角数值**+**diff[i][j]**，**这3个数当中的最小值，相等时取后者。

159页13，15，17，19，21行：“左上角数值加”语句中的 “加”字改为“**+**”，共5处，13行添加“**取**”字：

* j = 1：F =F：diff[1][1]=0，左上角数值**+**diff[1][1]=0，左侧+1=上面+1=2，**取**3个数当中的最小值，d[1][1] =0，如图4-23所示。
* j = 2：F≠R：diff[1][2]=1，左上角数值+diff[1][2]=2，…
* j = 3：F ≠A：diff[1][3]=1，左上角数值+diff[1][3]=3，…
* j = 4：F ≠ M：diff[1][4]=1，左上角数值+diff[1][4]=4，…
* j = 5：F ≠ E：diff[1][5]=1，左上角数值+diff[1][5]=5，…

160页-4行：“加”字改为“**+**”，并在前面添加逗号“**，**”：

即：取上面+1，左侧+1，左上角数值**+**diff[i][j]**，**3个数当中的最小值，相等时取后者。

1. **页码：167 • 行数：8 • 印次 1**

·i=3，j=5：m[3][4]+ m[4][5] =7> m[3][5]=6，不做改变。

·i=4，j=6：m[4][5]+ m[5][6]=9 > m[4][6]=8，不做改变。

改为

·i=3，j=5：m[3][4]+ m[4][5]=**8** > m[3][5]=6，不做改变。

·i=4，j=6：m[4][5]+ m[5][6]=**11** > m[4][6]=8，不做改变。

1. **页码：173 • 行数：3 • 印次 1**

图4-40替换为：



图4-40 矩阵乘法

1. **页码：175 • 行数：19 • 印次 1**

当i>j时， 改为：当i**<**j时,

1. **页码：185 • 行数：18 • 印次 1**

当i>j时， 改为：当i**<**j时,

1. **页码：210 • 行数：-7 • 印次 1**

（参看2.4.6节）改为：（参看2.3.6节）

1. **页码：211 • 行数： 12 • 印次 1**

该问题是否具有最优子结构性质。将“是否”二字删除，改为：

该问题具有最优子结构性质。

1. **页码：222 • 行数： 1 • 印次 1**

图4-100，4-101，4-102图标二叉搜索数树，删除数字：

1. **页码：225-236 • 行数： 6 • 印次 1**

225页6行：公式内里面改为+号:



226页-3行：公式内里面改为+号:



227页3，7，11行：公式内里面改为+号:

*c*[1][1]= min{*c*[1][0]**+***c*[2][1] }+ *w*[1][1] =0.18；

*c*[2][2]= min{*c*[2][1]**+***c*[3][2] }+ *w*[2][2] =0.27；

*c*[3][3]= min{*c*[3][2]**+***c*[4][3] }+ *w*[3][3] =0.25；

228页1行：改动红色部分：

*c*[4][4]= min{*c*[4][3]+*c*[5][4] }+ *w*[4][4] =0.14；

228页5，9行：公式内里面改为+号:

*c*[5][5]= min{*c*[5][4]**+***c*[6][5] }+ *w*[5][5] =0.22；

*c*[6][6]= min{*c*[6][5]**+***c*[7][6] }+ *w*[6][6] =0.29；

229页2行：公式内里面改为+号:



231页1行：公式内里面改为+号:



233页2行：公式内里面改为+号:



235页4行：公式内里面改为+号:



236页-2行：公式内里面改为+号:



1. **页码：305 • 行数： 6 • 印次 1**

零件*Ji*需要机器1、机器2的处理时间为*t*1*i*、*t*2*i*。

1. **页码：333 • 行数： -1，-4 • 印次 1**

-1行：添加阶乘符号，改为：叶子个数为(*n*-1)**!**，

-4行：改为：除了最后一层外，有**1+(*n*-1)** +…+(*n*-1)(*n*-2)...2≤*n*(*n*-1)!

1. **页码：360，363 • 行数： -6 • 印次 1**

360页-6行，363页-14行： 计算上界函数Bound中，while循环语句添加一行t++;提供的源码中无错误，仅书中缺少。

while(t<=n&&w[t]<=left)

{

 maxvalue+=v[t];

left-=w[t];

t++;

 }

1. **页码：411 • 行数：7,9 • 印次 1**

删除“非”字，“入”改为“离”：

选取“常数列元素/入基列元素”正比值的最小者，所对应的**非**基本变量*xk*为离基变量。*xk*对应的行向量[*ak*1，*ak*2，…，*akn*]为离基行。

图7-3中，常数列元素/入基列元素”正比值的最小者，所对应的基本变量*x*5为**离**基变量。*x*5对应的行向量为离基行，如图7-4所示。

1. **页码：417 • 行数： • 印次 1**



图7-16 新的单纯形表

1. **页码：419 • 行数：-7，-12 • 印次 1**

for(i=0;i<=n;i++) //计算特殊位, **入基列**的元素

{

if(i!=k)

kernel[i][e]=-kernel[i][e]/kernel[k][e];

}

for(j=0;j<=m;j++) //计算特殊位, **离基行**的元素

1. **页码：421 • 行数：16 • 印次 1**

16行for(i=1;i<=n;i++)之前插入代码：（源码更新已发布在博客和群里面）

for(j=1;j<=m;j++) //判断正检验数对应的列如果都小于等于0，则无界解

{

            max2=0;

           if(kernel[0][j]>0)

            {

               for(i=1;i<=n;i++) //搜索正检验数对应的列

                 if(max2<kernel[i][j])

                    max2=kernel[i][j];

               if(max2==0)

                {

                  cout<<"解无界"<<endl;

                  return; //退出函数，不能用break，因为它只是退出当前循环

                }

            }

 }

16-35行改为：

for(i=1;i<=n;i++) //找离基行(常数列/入基列正比值最小对应的行)

{

**~~if(max2<kernel[i][e])~~**

**~~{~~**

**~~max2=kernel[i][e];~~**

**~~}~~**

float temp=kernel[i][0]/kernel[i][e]; //常数项在前,temp=fabs(temp);

if(temp>0&&temp<min) //找离基变量

{

min=temp;

k=i;

}

}

cout<<"**入基**变量："<<"x"<<FJL[e]<<" ";

cout<<"离基变量："<<"x"<<JL[k]<<endl;

**i~~f(max2==0)~~**

**~~{~~**

**~~cout<<"解无界"<<endl;~~**

**~~break;~~**

**~~}~~**

//变基变换(转轴变换)

1. **页码：422 • 行数：6，12 • 印次 1**

for(i=0;i<=n;i++) //计算特殊位, **入基列**的元素

{

if(i!=k)

kernel[i][e]=-kernel[i][e]/kernel[k][e];

}

for(j=0;j<=m;j++) //计算特殊位, **离基行**的元素

1. **页码：423 • 行数：19，27 • 印次 1**

**入基**变量：x5 离基变量：x6

----------单纯形表如下：----------

b x2 x4 x6

c 915000 2.5 2.8 -76.25

x1 60000 1 0 5

x3 24000 0 1 2

x5 12000 0 0 1

x7 400 0.1 0.08 -0.05

**入基**变量：x4 离基变量：x7

1. **页码：436 • 行数：12 • 印次 1**

仍以图7-**7**7为例，其对应的残余网络如图7-79所示。

1. **页码：450 • 行数：6• 印次 1**

（9）找可增广路。

从源点开始，沿着高度*h*(*u*)=*h*(*v*)+1且有可行邻接边的方向前进，*h*(1)=4，*h*(2)=4，虽然*h*(3)=3，但已经没有可增流量，不可行。**重贴标签**，令*h*(**1**)= *h*(2) +1=5，

1. **页码：452 • 行数：12• 印次 1**

/\* j^1表示j和1的“**异或**运算”，因为创建边时是成对创建的，

1. **页码：535 • 行数：**最后一行**• 印次 1**

多了一个右括号，把最后的右括号去掉：

*an* = (*x*+*y*)*an*-1 - *xyan*-2