I、作业内容:最长公共子序列

要求:

- 利用附. 1.最长公共子序列输入数据

中给出的字符串 A, B, C, D, 分别找出下列两两字符串间的最长公共子串, 并给出结果:
A-B,
C-D,
A-D,
C-B

程. 源. 码::

```
#include <stdio.h>
#include <string.h>
#define MAXLEN 2500

void LCSLength(char *x, char *y, int m, int n, int c[][MAXLEN], int b[][MAXLEN])
{
	int i, j;

	for(i = 0; i <= m; i++)
		c[i][0] = 0;
	for(j = 1; j <= n; j++)
		c[0][j] = 0;
	for(i = 1; i<= m; i++)
		{
			for(j = 1; j <= n; j++)
				{
					if(x[i-1] == y[j-1])
						{
							c[i][j] = c[i-1][j-1] + 1;
							b[i][j] = 0;
						}
					else if(c[i-1][j] >= c[i][j-1])
						{
							c[i][j] = c[i-1][j];
							b[i][j] = 1;
						}
					else
						{
							c[i][j] = c[i][j-1];
							b[i][j] = -1;
```

```
                                }
                        }
                }
}

void PrintLCS(int b[][MAXLEN], char *x, int i, int j)
{
        if(i == 0 || j == 0)
                return;
        if(b[i][j] == 0)
                {
                        PrintLCS(b, x, i-1, j-1);
                        printf("%c ", x[i-1]);
                }
        else if(b[i][j] == 1)
                PrintLCS(b, x, i-1, j);
        else
                PrintLCS(b, x, i, j-1);
}

int main(int argc, char **argv)
{
        char                                    aa[MAXLEN]                                      =
"3741a169n084+932a968l840g035o126r160i539t644h812m737+375i919s815+210a465n30
2y234+360w485e024l174l698d312e454f025i006n494e376d677+651c373o160m418p173u9
00t034a542t144i134o107n610a628l715+277p841r630o097c046e000d407u018r082e439+43
9t714h726a096t284+739t123a143k178e039s852+378s958o919m728e300+478v048a355l26
0u637e480s348+252a562s569+581i767n340p107u404t588+428a174n336d647+257p541r83
2o112d830u085c018e645s292+782s463o834m765e312+358v268a149l808u058e434s632+1
34a726s023+317o512u829t311p602u526t829627007156527547003539349050502050257 9
513147583774997461448820159492726382418921359620098912229671250429570695 69
378242774277373090915734258858283879475246592101461548221554971247933819 36
182894871058085647631540170472332156666952245092456430497334508433691739 23
255018200425418297776291475671183277874463942273549539106609606561154500 44
675729818927389787909637922348732998482661583751721545046254185729742865 27
180333254951214699929767844857466793016576251068550814957631337704065556 64
200912752567171137041574605106030819094206486753687926520 70";
        char                                    bb[MAXLEN]                                      =
"&(!#a%$(n@*%+!*&a)*$l%&&g))*o)($r&@$i%#!t)$$h^#$m@!!+@&!i$%!s!#(+!)&a)(%n(()y
*)@+%()w^%&e$!!l%@&l*@(d@*)e!(*f#)@i%)^n*%*e^@(d^*%+&^)c@*#o()(m!(^p##)u&
%#t#(^a*&%t@%$i%!@o@*%n^@&a$%)l@($+*%^p##^r@&%o%!$c#^!e%!(d!)!u%$^r%^%
e$@)+@$^t^&&h&!*a!!%t$^*+&^!t#$@a(^!k)(#e$(#s#!(+!#*s%!)o!^%m#!(e$*^+#($v*!*a)#
%l#$#u#)#e&%&s%&$+#^(a@$(s&**+(!@i^$)n$@)p!!*u%%$t&#@+%*!a#%%n!^#d@$$+!*)
p@$%r$$@o*!%d#&(u@&&c!#*e^^%s&$%+@!%s&!^o@(&m&(&e$!*+$!&v%)!a#%^l#(&u
&%$e$@&s)@*+^(*a^!^s)!!+#%#o^%%u&($t%&%p#()u&^#t&^!^@@($*@$(&)&(*!#%^@!
```

```
^^$*&!^*!!))##$*$)#$!^$@$!^&$#(#%(*^@!@!*)^!&$*@$%$%%!(&&^@)&*))&^&&@&)#(
^*$&#^$#%^%#^&@&*#&!(#*%#**$@#)^(*()@(^&!@&(#^#@%$&)^^($!(($(!#^)&!%#@!&
%&@#%!!%(^^&!!#(%*$##&()@!(!*#%*^^!*!*^&*&()#%^*&(&!^@$(@^)@@$$%!$$*$$@)
*#@($(%(*$$##)$!@!)&@@!$!&@&&)()(^&&&&%((!@$%^!@^(!&$@$@^$$#$$!^@))##($
*@$&@^$^$@*^^*!%@$&!$)^$)*#%^!@(@$%*#&&&(^(**)$^)(!)&$^!*$&$@%(^**$%&
@%#^@@^#(!(%^))*@@###*@^@&*$(#&)%(%!^&!*)^%!@^)%&!$@$$#!)&!!&*!!%)@@&
@(*@(^!$&$#!)))!!^^%@%(**)$*^!!*($!*!)!#^@%*&$$%!%)#$!($()%(%)!#";
	char cc[MAXLEN] =
"7033a825n442+580a140l659g853o200r879i610t403h938m362+054i997s569+165a354n00
9y190+733w699e162l095l606d927e110f262i145n339e295d327+714c538o936m930p658u0
99t346a808t401i238o952n810a973l035+039p915r253o507c888e136d064u955r363e209+66
1t070h489a601t629+855t441a594k460e369s045+872s862o687m130e480+677v536a356l24
0u347e799s394+425a229s234+242i778n153p334u901t846+150a308n134d237+781p077r98
0o514d895u771c528e088s891+366s067o288m878e340+486v518a875l598u023e219s644+2
27a141s199+330o867u339t537p109u343t37029194510158042071986019868543397297983
53643410074012584617254914613054656819690663988290905874978082948294817979
79024459813702517981321228505397454757711562214873601157160013067814068867
50887347346442868260742894435063570740933980500571908580457558339693853778
29932298873979872025119433703590742621878944416385851349190558782743696394
84510531876823560908923262320499393985379237493430407883777369733930995864
05393274595751018549599725926848545 8";
	char dd[MAXLEN] =
"#%#&a!%&n*%*+##%a)$!l*!(g)&*o%&!r&%(i^!(t))%h!(!m!#!+$*$i)!@s*!@+)^(a%)$n)%@y)
^@+*^#w&!^e&&#l#$&l)&$d!&!e@@%f@&&i)#&n^)#e#*&d*!)+^&$c$!)o@&!m^!#p@#!u
*$#t@!(a*^@t##!i#$$o)^#n&@)a()*l@$!+$))p*!*r)$(o#^(c%$#e#@^d!)&u*$#r%)(e&*)+%*
&t@$^h*^$a**&t)#!+(@)t^!)a%@&k!!@e^*&s^#&+%)%s$*(o)@!m%)^e((*+^^$v#^^a*#*l$
&*u)#@e(($s@)*+!!)a)&(s%&@+&!)i%@&n%*)p*%^u(#@t(!!+(&&a^^#n@()d%$%+)$%p&()
r)!!o^&%d%&^u!!!c&&$e*)^s!%!+^!*s!)!o%(*m%((e^@@+@($v!!)a())l*!*u)(!e&*$s)(*+$^!a!
%^s$@(+^!^o)#)u!%$t^^(p)$^u(&$t@^)##@(&))(%)#@*&$$^###&##&#(#%$$!&$#$&*^^!$
(%$#@((@!)!^%$*(&)@*&)#^^@%$*(^&%@^^())^!^$#($%@!@)*@!^#$@@)()#^^)*)@^&(
(**@%**)(#*@^)!)(^%(!!(()#@(&^%&#))*@#(^&(@#$@!%%&#@)^*&^($()^*@!$)*&@%^
@$&^!$(**(&!@&&!#*&))(^$&()#%@$!%!)#)#^!^@@%!)#*)#$*%^@&!*&!%!((!^*&&&@*
$&#!^^*%(((($%!!&^(&)&(#^$%#&*&$)&^!@#)*#)($(#$%^!^)$)*&*!$$*^@^^^))^*$)$!^^%!(!
^)@#*))%^(^^)#*$#%^!!!*!!$)!^^)@#&&$(&*(*$(^$(*))(&^&(%)**)))!((!!&**@@#$#(*)$#$^
#%!^*^*%&!)$(!!(&)^(^#^%(^^*&$)**)(@!#$(@%@$!%*%*%@@%%*)*!)^!#!@!^@##%(@!
*$$^(!***#!#(*!^)(%&^&(&@#%#&^^*$";
	int a[MAXLEN][MAXLEN];
	int b[MAXLEN][MAXLEN];
	int c[MAXLEN][MAXLEN];
	int d[MAXLEN][MAXLEN];
	int m, n,i,j;

	m = strlen(aa);
	n = strlen(bb);
	i=strlen(cc);
```

```
j=strlen(dd);

LCSLength(aa,bb, m, n, b, a);
PrintLCS(a, aa, m, n);
LCSLength(cc, dd, i, j, d, c);
PrintLCS(c, cc, i, j);

char aa2[MAXLEN] =
"3741a169n084+932a968l840g035o126r160i539t644h812m737+375i919s815+210a465n30
2y234+360w485e024l174l698d312e454f025i006n494e376d677+651c373o160m418p173u9
00t034a542t144i134o107n610a628l715+277p841r630o097c046e000d407u018r082e439+43
9t714h726a096t284+739t123a143k178e039s852+378s958o919m728e300+478v048a355l26
0u637e480s348+252a562s569+581i767n340p107u404t588+428a174n336d647+257p541r83
2o112d830u085c018e645s292+782s463o834m765e312+358v268a149l808u058e434s632+1
34a726s023+317o512u829t311p602u526t829627007156527547003539349050502050257 9
513147583774997461448820159492726382418921359620098912229671250429570695 69
378242774277373090915734258858283879475246592101461548221554971247933819 36
182894871058085647631540170472332156666952245092456430497334508433691739 23
255018200425418297776291475671183277874463942273549539106609606561154500 44
675729818927389787909637922348732998482661583751721545046254185729742865 27
180333254951214699929767844857466793016576251068550814957631337704065556 64
200912752567171137041574605106030819094206486753687926520 70";
char bb2[MAXLEN] =
"&(!#a%$(n@*%+!*&a)*$l%&&g))*o)($r&@$i%#!t)$$h^#$m@!!+@&!i$%!s!#(+!)&a)(%n(()y
*)@+%()w^%&e$!!l%@&l*@(d@*)e!(*f#)@i%)^n*%*e^@(d^*%+&^)c@*#o()(m!(^p##)u&
%#t#(^a*&%t@%$i%!@o@*%n^@&a$%)l@($+*%^p##^r@&%o%!$c#^!e%!(d!)!u%$^r%^%
e$@)+@$^t^&&h&!*a!!%t$^*+&^!t#$@a(^!k)(#e$(#s#!(+!#*s%!)o!^%m#!(e$*^+#($v*!*a)#
%l#$#u#)#e&%&s%&$+#^(a@$(s&**+(!@i^$)n$@)p!!*u%%$t&#@+%*!a#%%n!^#d@$$+!*)
p@$%r$$@o*!%d#&(u@&&c!#*e^^%s&$%+@!%s&!^o@(&m&(&e$!*+$!&v%)!a#%^l#(&u
&%$e$@&s)@*+^(*a^!^s)!!+#%#o^%%u&($t%&%p#()u&^#t&^!^@@($*@$(&)&(*!#%^@!
^^$*&!^*!!))##$*$)#$!^$@$!^&$#(#%(*^@!@!*)^!&$*@$%$%%!(&&^@)&*))&^&&@&)#(
^*$&#^$#%^%#^&@&*#&!(#*%#**$@#)^(*()@(^&!@&(#^#@%$&)^^($!(($(!#^)&!%#@!&
%&@#%!!%(^^&!!#(%*$##&()@!(!*#%*^^!*!*^&*&()#%^*&(&!^@$(@^)@@$$%!$$*$$@)
*#@($(%(*$$##)$!@!)&@@!$!&@&&)()(^&&&&%((!@$%^!@^(!&$@$@^$$#$$!^@))##($
*@$&@^$^$@*^^*!%@$&!$)^$)*#%^!@(@$%*#&&&(^(**)$^)(!)&$^!*$&$@%(^**$%&
@%#^@@^#(!(%^))*@@###*@^@&*$(#&)%(%!^&!*)^%!@^)%&!$@$$#!)&!!&*!!%)@@&
@(*@(^!$&$#!)))!!^^%@%(**)$*^!!*($!*!)!#^@%*&$$%!%)#$!($()%(%)!#";
char cc2[MAXLEN] =
"7033a825n442+580a140l659g853o200r879i610t403h938m362+054i997s569+165a354n00
9y190+733w699e162l095l606d927e110f262i145n339e295d327+714c538o936m930p658u0
99t346a808t401i238o952n810a973l035+039p915r253o507c888e136d064u955r363e209+66
1t070h489a601t629+855t441a594k460e369s045+872s862o687m130e480+677v536a356l24
0u347e799s394+425a229s234+242i778n153p334u901t846+150a308n134d237+781p077r98
0o514d895u771c528e088s891+366s067o288m878e340+486v518a875l598u023e219s644+2
27a141s199+330o867u339t537p109u343t37029194510158042071986019868543397297 98
```

```
353643410074012584617254914613054656819690663988290905874978082948294817977902445981370251798132122850539745475771156221487360115716001306781406886750887347346442868260742894435063570740933980500571908580457558339693853778299322988739798720251194337035907426218789444163858513491905587827436963948451053187682356090892326232049939398537923749343040788377736973393099586405393274595751018549599725926848545 8";

    char dd2[MAXLEN] = "#%#&a!%&n*%*+##%a)$!l*!(g)&*o%&!r&%(i^!(t))%h!(!m!#!+$*$i)!@s*!@+)^(a%)$n)%@y)^@+*^#w&!^e&&#l#$&l)&$d!&!e@@%f@&&i)#&n^)#e#*&d*!)+^&$c$!)o@&!m^!#p@#!u*$#t@!(a*^@t##!i#$$o)^#n&@)a()*l@$!+$))p*!*r)$(o#^(c%$#e#@^d!)&u*$#r%)(e&*)+%*&t@$^h*^$a**&t)#!+(@)t^!)a%@&k!!@e^*&s^#&+%)%s$*(o)@!m%)^e((*+^^$v#^^a*#*l$&*u)#@e(($s@)*+!!)a)&(s%&@+&!)i%@&n%*)p*%^u(#@t(!!+(&&a^^#n@()d%$%+)$%p&()r)!!o^&%d%&^u!!!c&&$e*)^s!%!+^!*s!)!o%(*m%((e^@@+@($v!!)a())l*!*u)(!e&*$s)(*+$^!a!%^s$@(+^!^o)#)u!%$t^^(p)$^u(&$t@^)##@(&))(%)#@*&$$^###&##&(#%$$!&$#$&*^^!$(%$#@((@!)!^%$*(&)@*&)#^^@%$*(^&%@^^())^!^$#($%@!@)*@!^#$@@)()#^^)*)@^&((**@%**)(#*@^)!)(^%(!!(()#@(&^%&#))*@#(^&(@#$@!%%&#@)^*&^($()^*@!$)*&@%^@$&^!$(**(&!@&&!#*&))(^$&()#%@$!%!)#)#^!^@@%!)#*)#$*%^@&!*&!%!((!^*&&&@*$&#!^^*%(((($%!!&^(&)&(#^$%#&*&$)&^!@#)*#)($(#$%^!^)$)*&*!$$*^@^^^))^*$)$!^^%!(!^)@#*))%^(^^)#*$#%^!!!*!!$)!^^)@#&&$(&*(*$(^$(*))(&^&(%)**)))!((!!&**@@#$#(*)$#$^#%!^*^*%&!)$(!!(&)^(^#^%(^^*&$)**)(@!#$(@%@$!%*%*%@@%%*)*!)^!#!@!^@##%(@!*$$^(!***#!#(*!^)(%&^&(&@#%#&^^*$";

    LCSLength(aa2, dd2, m, j, c, b);
    PrintLCS(a, aa2, m, j);

    LCSLength(bb2,cc2, n, i, c, b);
    PrintLCS(b, bb2, n, i);

    return 0;
}
```

2. 结果

```
an+algorithm+is+any+welldefined+computational+procedure+that+takes+some+values+as+input+and+produces+some+values+as+output
122
n+algorithm+is+a+eene+computational+p+that+tasom+e+input+no+e+output
68
n+algorithm+is+a+eene+computational+e+that+tasom+e+input+no+e+output
68
An+algorithm+is+thus+a+sequence+of+computational+steps+that+transform+the+input+into+the+output
95
```

II、作业:最大子段和

■ 针对“附件 2.最大子段和输入数据-序列 1”、“附件 2.最大子段和输入数据-序列 2”中给出的序列 1、序列 2，分别计算其最大子段和

源程序代码：

```
#include <cstdio>
#include <iostream>
```

```
#include <cstdlib>
#include <cstring>
using namespace std;
const int MAXN = 500;
int MaxSum(int n, int *a,int &l, int &r)
{
int sum=0, b=0;
int pos = 1;
for(int i=1; i<=n; i++) {
if(b>0) {
b += a[i];
}
else {
b = a[i];
pos = i;
}
if(b>sum) {
sum = b;
l = pos;
r = i;
}
}
return sum;
}
int main()
{
freopen("附件 2.最大子段和输入数据-序列 1.txt","r",stdin);
freopen("maxsubsum1.txt","w",stdout);
int n = 1;
int a[MAXN];
while(scanf("%d",&a[n]) != EOF) {
n++;
}
n--;
int l,r;
```

```
printf("最大字段和为:%d\n",MaxSum(n,a,l,r));
printf("最大字段位于第%d 个元素到第%d 个元素\n",l,r);
return 0;
}
```

运行结果：

附件 2.最大子段和输入数据-序列 1：

最大字段和为:2715

最大字段位于第 43 个元素到第 329 个元素

附件 2.最大子段和输入数据-序列 2：

最大字段和为:377

最大字段位于第 72 个元素到第 142 个元素

第三题—贪心实现

```
#include<iostream>
#include<fstream>
#include<cmath>
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
const double Earth_R=6378.137;
struct Node
{
int ENodeID,seq;
double LongiTude,LatiTude;
double radLat,radLon;
Node(){
}
Node(int ID,double Long,double Lati,int ord):ENodeID(ID),
LongiTude(Long),LatiTude(Lati),seq(ord){
radLat=LatiTude*M_PI/180.0;//构造结构体的同时直接计算弧度
radLon=LongiTude*M_PI/180.0;
}
}node[50];
double dp[50][50],s[50][50],cunt;
double solve(int n);
double Weight(int a,int b,int c);
double dist(int a,int b)
{
struct Node p1=node[a],p2=node[b];
return Earth_R*acos(cos(p1.radLat)*cos(p2.radLat)*cos(p1.radLon-p2.radLon)
+sin(p1.radLat)*sin(p2.radLat));
}
```

```
void Recall(int h,int n);
ofstream out("最优三角剖分结果.txt");
int main()
{
ifstream in("附件 3-2.29 个基站凸多边形数据.txt",ios::in);
int Cun,ID,i,j;
double Long,Lati,ans,sum=0.0;
while(!in.eof())
{
in>>ID>>Long>>Lati>>i;
node[i]=Node(ID,Long,Lati,i);
memset(dp[i],0,sizeof(dp[i]));
}
Cun=i-1;
//for(i=0;i<Cun;i++)
//sum+=dist(i,i+1);//求边长总和
ans=solve(Cun);
out<<"最优三角形的剖分结果为"<<endl;
Recall(1,Cun);
printf("%.6f 千米",ans);
out<<"最小边长弦长总和为"<<ans<<"千米"<<endl;
out.close();
in.close();
return 0;
}
double Weight(int a,int b,int c)//返回构造的三角形边长
{
return dist(a,b)+dist(a,c)+dist(b,c);
}
void Recall(int h,int n)
{
if(n<=h) return;
cout<<h-1<<" "<<s[h][n]<<" "<<n<<endl;
out<<"点"<<h-1<<", "<<s[h][n]<<", "<<n<<endl;
Recall(h,s[h][n]);
Recall(s[h][n]+1,n);
}
double solve(int n)
{
for(int r=2;r<=n;r++)//枚举多边形的边数
{
for(int i=1;i<=n-r+1;i++)//枚举子问题的起始节点
{
int j=i+r-1;//选取子问题终点
```

```
dp[i][j]=dp[i+1][j]+Weight(i-1,i,j);//初始化该子问题的值=某其子局面的最优解+新增点产生的三角形。
s[i][j]=i;//显然此时 k=i；
for(int k=i+1;k<j;k++)//枚举 i+1~j-1 中的第三个点（必须给剩余的部分至少留一个三角形
{
double temp=dp[i][k]+dp[k+1][j]+Weight(i-1,k,j);
if(temp<dp[i][j])
{
dp[i][j]=temp;//更新最优解
s[i][j]=k;//记录剖分点
}
}
}
}
return dp[1][n];//0~n 点构建的三角剖分中所有内弦长的最优解
}
```

三·最优三角算法启发式

```
#include<iostream>
#include<math.h>
using namespace std;
double getDistance(double LON[],double LAT[],int a,int b)
{
	double LATa = 3.1415*LAT[a]/180;
	double LATb = 3.1415*LAT[b]/180;
	double LONa = 3.1415*LON[a]/180;
	double LONb = 3.1415*LON[b]/180;
	return acos(cos(LATa)*cos(LATb)*cos(LONa-LONb)+sin(LATa)*sin(LATb));
}


double weight(double LON[] ,double LAT[] ,int i,int k,int j)
{
	double x1 = getDistance(LON,LAT,i,k);
	double x2 = getDistance(LON,LAT,i,j);
	double x3 = getDistance(LON,LAT,k,j);
	return x1+x2+x3;
}

void minweight(double LON[] ,double LAT[] ,int n,double t[][29] ,int s[][29] )
{
	for(int i=1; i<=n; i++) t[i][i] = 0;
	for(int r=2; r<=n; r++)
```

```
            for(int i=1; i<n-r+1; i++) //< <=
                {
                    int j = i+r-1;
                    t[i][j] = t[i+1][j]+weight(LON,LAT,i-1,i,j);
                    s[i][j] = i;
                    for(int k=i+1; k<i+r-1; k++)
                        {
                            double u = t[i][k]+t[k+1][j]+weight(LON,LAT,i-1,k,j);
                            if(u<t[i][j])
                                {
                                    t[i][j] = u;
                                    s[i][j] = k;
                                }
                        }
                }
}



int main()
{
    // TODO Auto-generated method stub

    int ENODEBID1[] =
{565466,565026,568209,33237,565025,568313,566770,566871,567422,566368,566343,331
22,565884,566091,33181,567580,566022,566038,567506,565676,565466};
    int ENODEBID2 []=
{565668,565756,565494,565532,565578,566971,567007,567009,566720,568098,566643,56
6604,566740,566666,567498,565645,566354,566090,567212,565822,567353,565998,56715
9,567266,565841,567491,33192,565393,565668};
    double LONGITUDE1[] =
{102.746,102.872,102.879,102.895,102.895,102.863,102.768,102.736,102.719,102.678,102.
677,102.676,102.675,102.675,102.675,102.675,102.676,102.686,102.704,102.74,102.746};
    double LONGITUDE2[] =
{102.732,102.746,102.749,102.763,102.769,102.77,102.77,102.769,102.764,102.754,102.74
4,102.74,102.729,102.718,102.714,102.704,102.695,102.687,102.685,102.682,102.681,102.
682,102.686,102.689,102.704,102.707,102.709,102.714,102.732};
    double LATITUDE1[] =
{24.9555,25.01466,25.024444,25.050191,25.095,25.09831,25.10243,25.10088,25.096666,25.
05902,25.05719,25.05218,25.03436,25.01318,25.00996,24.99597,24.99248,24.98032,24.97
184,24.95721,24.9555};
    double LATITUDE2[] =
{24.98221,24.98604,24.988197,25.0001,25.01036,25.01896,25.027222,25.033055,25.04446,
25.05378,25.05965,25.06162,25.06275,25.06268,25.061944,25.05773,25.05141,25.04281,2
5.038284,25.03012,25.0252,25.01405,25.005124,25.00074,24.98862,24.9864,24.98496,24.9
```

```
832,24.98221};
        double t1[22][29] ;
        double t2[29][29];
        int s1[22][29] ;
        int s2[29][29];

        minweight(LONGITUDE1,LATITUDE1,21,t1,s1);
        minweight(LONGITUDE2,LATITUDE2,28,t2,s2);
        for(int i=0; i<28; i++)
            {
                for(int j=0; j<28; j++)
                    {
                        cout<<s2[i][j];
                        cout<<" ";
                    }
                cout<<"\n";
            }

}
```

运行结果：

附件 3-1.21 个基站凸多边形数据：

最优三角剖分为 534895

剖分点为 0、1、20

剖分点为 1、2、20

剖分点为 2、3、20

剖分点为 3、4、20

剖分点为 4、5、20

剖分点为 5、6、20

剖分点为 6、7、20

剖分点为 7、8、20

剖分点为 8、9、20

剖分点为 9、10、20

剖分点为 10、11、20

剖分点为 11、12、20

剖分点为 12、13、20

剖分点为 13、14、20

剖分点为 14、15、20

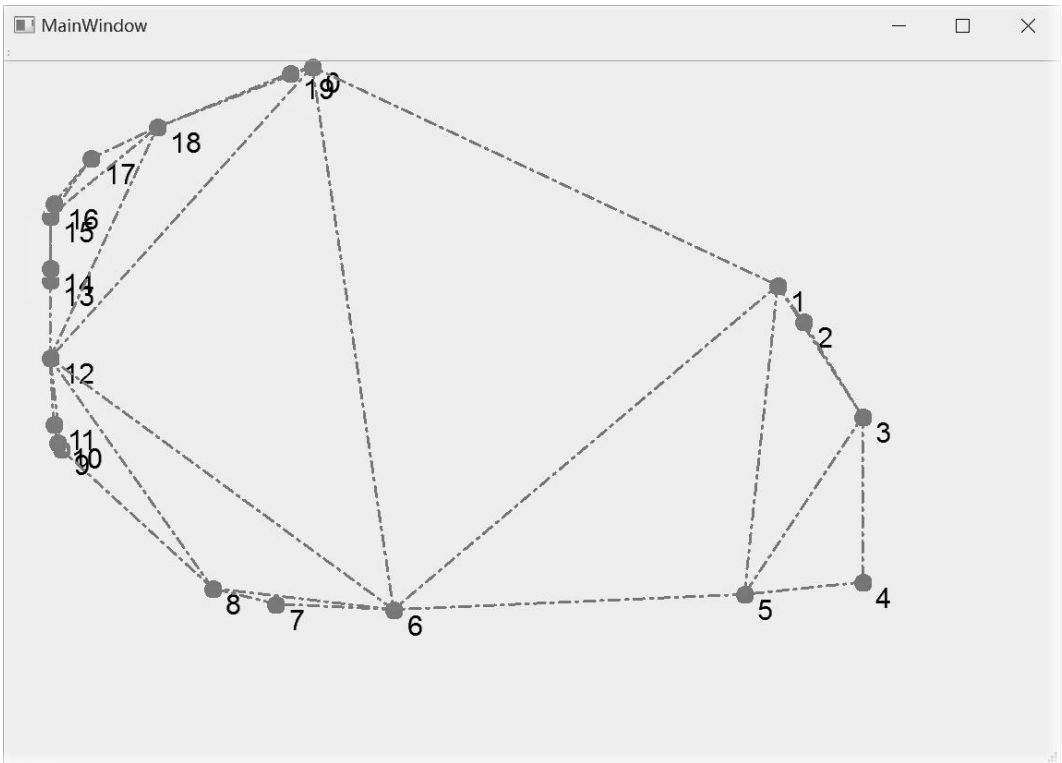
剖分点为 15、16、20

剖分点为 16、17、20

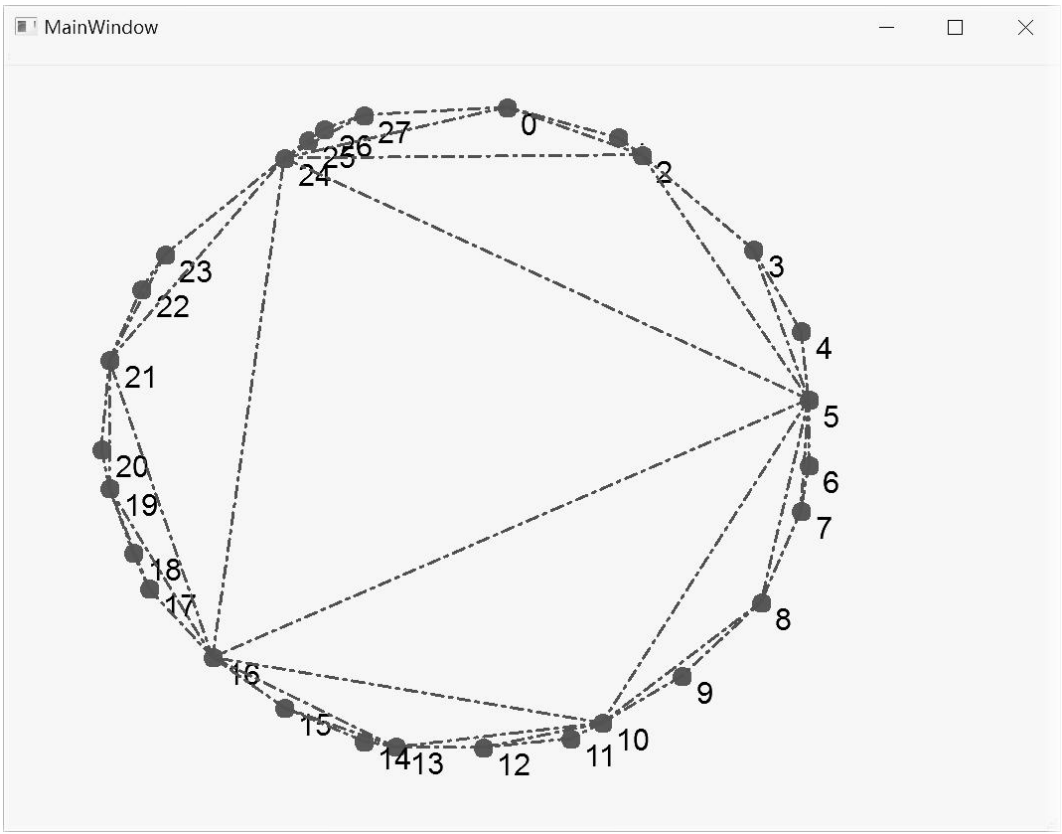
剖分点为 17、18、20

剖分点为 18、19、20

附件 3-2.29 个基站凸多边形数据：


MainWindow
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1

最优三角剖分为 353926
剖分点为 0、1、28
剖分点为 1、2、28
剖分点为 2、3、28
剖分点为 3、4、28
剖分点为 4、5、28
剖分点为 5、6、28
剖分点为 6、7、28
剖分点为 7、8、28
剖分点为 8、9、28
剖分点为 9、10、28
剖分点为 10、11、28
剖分点为 11、12、28
剖分点为 12、13、28
剖分点为 13、14、28
剖分点为 14、15、28
剖分点为 15、16、28
剖分点为 16、17、28
剖分点为 17、18、28
剖分点为 18、19、28
剖分点为 19、20、28
剖分点为 20、21、28
剖分点为 21、22、28
剖分点为 22、23、28
剖分点为 23、24、28
剖分点为 24、25、28
剖分点为 25、26、28
剖分点为 26、27、28
最小边长弦长总和为 191.378 千米



IV、作业内容：**0-1** 背包问题

程. 源. 码

```
#include<iostream>
#include<cstring>
#include<algorithm>
#include<fstream>
#include<queue>
using namespace std;
queue<int>result;
const int MaxNum=110;
const int MaxCap=1010;
int dp[MaxCap+10];
int Weigh[MaxNum],Value[MaxNum],Record[MaxNum][MaxCap];
int sumWEI,sumVal;
void zero_onePacket(int N,int C);
void Recall(int n,int u);
int main()
{
int Cap,i,Num;
ofstream out("0-1 背. 方. 结. .txt",ios::out);
while(1)
{
```

```
cout<<"请. 入. 包. 容. ,输. -1 结. 程. "<<endl;
cin>>Cap;//in>>Cap;
if(Cap==-1) break;
cout<<"请. 入. 品. 数. "<<endl;
cin>>Num;
memset(dp,-1,sizeof(dp));//初. 化
memset(Record,0,sizeof(Record));
cout<<"请. 此. 入. 品. 重. ， . 格. 开"<<endl;
for(i=1;i<=Num;i++)
cin>>Weigh[i];//in>>Weigh[i];
cout<<"请. 此. 入. 品. 价. ， . 格. 开"<<endl;
for(i=1;i<=Num;i++)
```

```
cin>>Value[i];//in>>Value[i];
zero_onePacket(Num,Cap);
int mark=0,maxx=0;
for(i=1;i<=Cap;i++)//遍. 寻. 最. 方.
{
if(maxx<dp[i])
{
maxx=dp[i]; mark=i;
}
}
cout<<"最. 装. 方. 的. 价. 为:"<<maxx<<endl<<"最. 装. 方. 的. 重.
为:"<<mark<<endl;
out<<"最. 装. 方案. 总. 值. ： "<<maxx<<endl<<"最. 装. 方. 的总. 量. ：
"<<mark<<endl;
if(mark!=0)//回. 记. 最. 方. 的. 置. 况
{
sumWEI=sumVal=0;
cout<<"最. 装. 方. 放. 的. 品. ： "<<endl;
out<<"最. 装. 方. 放. 的. 品. ： "<<endl;
Recall(Num,mark);
}
while(!result.empty())//输. 最. 方. 计. 结.
{
int head=result.front();
cout<<"第"<<head<<"个. 品, 重. "<<Weigh[head]<<"  价. "<<Value[head]<<endl;
out<<"第"<<head<<"个. 品, 重. "<<Weigh[head]<<"  价. "<<Value[head]<<endl;
result.pop();
}
cout<<endl;
out<<endl;
}
out.close();
return 0;
}
void Recall(int n,int u)
{
if(u==0) return;
Recall(Record[n][u]-1,u-Weigh[Record[n][u]]);
```

```
result.push(Record[n][u]);
return;
}
```

```
void zero_onePacket(int N,int C)
{
dp[0]=0;
for(int i=1;i<=N;i++)
{
for(int j=C;j>=Weigh[i];j--)
{
if(dp[j-Weigh[i]]>-1)//当. 量. j-Weigh[i]有. 品.
{
if(dp[j]<dp[j-Weigh[i]]+Value[i])//当. 优. 可. 新. 即. 新
{
dp[j]=dp[j-Weigh[i]]+Value[i];
Record[i][j]=i;//更. 放. 标.
}
}
if(Record[i][j]==0) Record[i][j]=Record[i-1][j];
}
for(int j=Weigh[i]-1;j>0;j--)
Record[i][j]=Record[i-1][j];
}
}
```

运. 结.

数. **1**：

最. 装. 方. 的. 价. 为:1085
最. 装. 方. 的. 重. 为:298
最. 装. 方. 放. 的. 品. ：
第 1 个. 品, 重. 14 价. 50
第 2 个. 品, 重. 11 价. 72
第 4 个. 品, 重. 17 价. 69
第 8 个. 品, 重. 26 价. 59
第 9 个. 品, 重. 10 价. 49
第 11 个. 品, 重. 16 价. 36
第 18 个. 品, 重. 19 价. 71
第 20 个. 品, 重. 29 价. 61
第 23 个. 品, 重. 13 价. 63
第 24 个. 品, 重. 15 价. 59
第 25 个. 品, 重. 9 价. 48
第 26 个. 品, 重. 10 价. 41

第 32 个. 品, 重. 8 价. 50
第 33 个. 品, 重. 11 价. 48
第 35 个. 品, 重. 11 价. 22
第 38 个. 品, 重. 8 价. 51
第 43 个. 品, 重. 28 价. 72
第 44 个. 品, 重. 16 价. 46
第 45 个. 品, 重. 9 价. 41
第 50 个. 品, 重. 18 价. 77
数. **2**
最. 装. 方. 的. 价. 为:1568
最. 装. 方. 的. 重. 为:598
最. 装. 方. 放. 的. 品. :
第 1 个. 品, 重. 10 价. 61
第 5 个. 品, 重. 33 价. 61
第 6 个. 品, 重. 44 价. 79
第 9 个. 品, 重. 15 价. 59
第 11 个. 品, 重. 12 价. 30
第 21 个. 品, 重. 40 价. 72
第 22 个. 品, 重. 20 价. 74
第 23 个. 品, 重. 18 价. 46
第 25 个. 品, 重. 10 价. 16
第 27 个. 品, 重. 28 价. 51
第 31 个. 品, 重. 24 价. 64
第 42 个. 品, 重. 26 价. 54
第 49 个. 品, 重. 9 价. 19
第 52 个. 品, 重. 18 价. 67
第 54 个. 品, 重. 42 价. 73
第 55 个. 品, 重. 15 价. 44
第 58 个. 品, 重. 11 价. 36
第 60 个. 品, 重. 18 价. 49
第 61 个. 品, 重. 8 价. 79
第 77 个. 品, 重. 18 价. 74
第 80 个. 品, 重. 20 价. 46
第 81 个. 品, 重. 11 价. 35
第 82 个. 品, 重. 23 价. 46
第 83 个. 品, 重. 26 价. 73
第 84 个. 品, 重. 9 价. 39
第 88 个. 品, 重. 36 价. 74
第 89 个. 品, 重. 24 价. 67
第 95 个. 品, 重. 30