算法设计与分析

第三章

姓名：张博康

学号：2014211383

班级：2014211309

一、 源代码

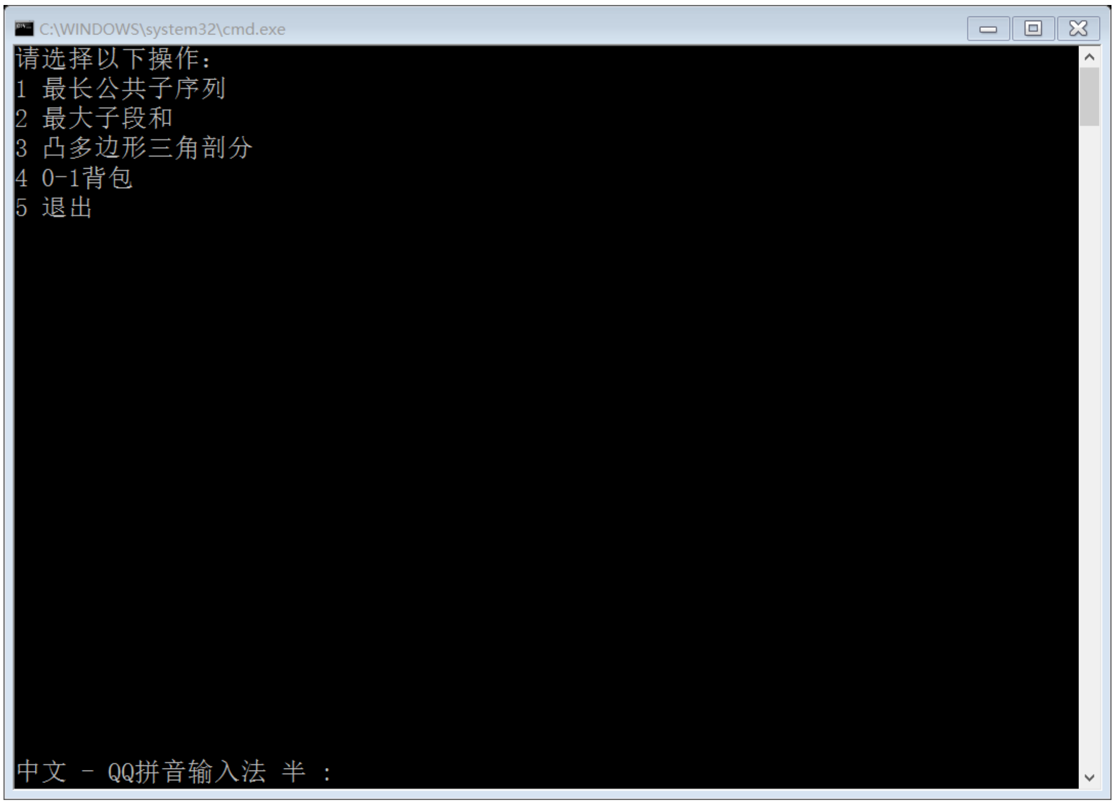
```
#include <iostream>
#include <sstream>
#include <fstream>
#include <cstring>
#include <cstdlib>
#include <vector>
#include <climits>
#include <cmath>

#define MAXSIZE 10000
#define Pi 3.141592657

typedef struct station
{
    long long enodedId;
    double longitude, latitude;
    int index;
} Station;

using namespace std;

const double R = 6378137.0; //地球半径，以m为单位

Station stations[MAXSIZE];
int dp[MAXSIZE][MAXSIZE];
int opr[MAXSIZE][MAXSIZE];
double weight[MAXSIZE][MAXSIZE];

double distance(const Station &u, const Station &v)
{
    double radLat1 = u.latitude * Pi / 180.0;
    double radLat2 = v.latitude * Pi / 180.0;
    double radLon1 = u.longitude * Pi / 180.0;
    double radLon2 = v.longitude * Pi / 180.0;

    return R * acos(cos(radLat1) * cos(radLat2) * cos(radLon1
- radLon2)
        + sin(radLat1) * sin(radLat2));
}

double Weight(int i, int k, int j)
{
        return weight[i][k] + weight[i][j] + weight[k][j];
    }

    string LCS(const string &X, const string &Y)
    {
        memset(dp, 0, sizeof(dp));
        memset(opr, 0, sizeof(opr)); //1 表示从(i-1,j-1)->(i,j);2
表示(i,j-1)->(i,j);3 表示从(i-1,j)->(i,j)

        dp[0][0] = 0;
        for (int i = 1; i <= X.length(); ++i)
        {
            for (int j = 1; j <= Y.length(); ++j)
            {
                //cout << i << j << endl;
                if (X[i-1] == Y[j-1])
                {
                   // cout << X[i-1] << Y[j-1] << endl;;
                    dp[i][j] = dp[i-1][j-1] + 1;
                    opr[i][j] = 1;
                }
                else
                {
                    dp[i][j] = max(dp[i][j-1], dp[i-1][j]);
                    if (dp[i][j] == dp[i][j-1])
                        opr[i][j] = 2;
                    else
                        opr[i][j] = 3;
                }
            }
        }

        string sub;
        int i = X.length();
        int j = Y.length();
        while (i != 0 && j != 0)
        {
            if (opr[i][j] == 1)
            {
                sub = X[i-1] + sub;
                i--;
                j--;
            }
            else if (opr[i][j] == 2)
                j--;
            else
                i--;
        }

        return sub;
    }

    int MaxSum(const vector<int> &a, int &left, int &right)
    {
        int sum = 0;
        int max = -INT_MAX;
        int l, r;

        for (int i = 0; i < a.size(); ++i)
        {
            if (sum + a[i] < 0)
            {
                sum = 0;
                l = i + 1;
            }
            else
            {
                sum += a[i];
                r = i;
            }
            if (sum > max)
            {
                max = sum;
                left = l;
                right = r;
            }
        }

        return max;
    }

    double MinWeightTriangulation(const int &n)
    {
        memset(dp, 0, sizeof(dp));
        memset(opr, 0, sizeof(opr));

        for (int i = 0; i < n; i++)
            for (int j = i; j < n; j++)
```

```
                weight[j][i] = weight[i][j] = 
distance(stations[i], stations[j]);

        for (int r = 2; r <= n; r++)
            for (int i = 1; i <= n-r+1; i++)
            {
                int j = i + r - 1;
                dp[i][j] = dp[i + 1][j] + Weight(i - 1, i, j);
                opr[i][j] = i;
                for (int k = i+1; k < j; k++)
                {
                    int u = dp[i][k] + dp[k + 1][j] + Weight(i - 
1, k, j);
                    if (u < dp[i][j])
                    {
                        dp[i][j] = u;
                        opr[i][j] = k;
                    }
                }
            }

        double len = 0;
        for (int i = 0; i < n-1; i++)
        {
            len += weight[i][i+1];
        }
        len += weight[n-1][0];

        return (len + dp[1][n-1]) / 2;
    }

    void TraceBack(int i, int j)
    {
        if (i == j)
            return;

        TraceBack(i, opr[i][j]);
        TraceBack(opr[i][j] + 1, j);

        cout << "三角剖分顶点：V" << i-1 << ",V" << j << ",V" << 
opr[i][j] << endl;
    }
    vector<bool> Knapspack(const int &c, const vector<int> 
&weight, const vector<int> &value, int &maxn)
```

```
    {
        int n = weight.size();
        vector<bool> result(n, false);
        memset(dp, 0, sizeof(dp));
        memset(opr, 0, sizeof(opr)); //1 表示从(i+1,j)->(i,j);2 表
示(i+1,j)->(i,j-weight[i]);

        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j <= c; j++)
            {
                if (j < weight[i]) {
                    dp[i + 1][j] = dp[i][j];
                    opr[i + 1][j] = 1;
                } else {
                    dp[i + 1][j] = max(dp[i][j], dp[i][j -
weight[i]] + value[i]);
                    if (dp[i + 1][j] == dp[i][j - weight[i]] +
value[i])
                        opr[i + 1][j] = 2;
                    else
                        opr[i + 1][j] = 1;
                }
            }
        }

        int i = n, j = c;
        while(i != 0)
        {
            if(opr[i][j] == 2)
            {
                result[i - 1] = true;
                j = j - weight[i - 1];
            }
            i--;
        }

        maxn = dp[n][c];
        return result;
    }

    int main(int argc, char const *argv[])
    {
        int choose = 0;
        while (choose != 5)
        {
            cout << "请选择以下操作：" << endl;
            cout << "1 最长公共子序列" << endl;
            cout << "2 最大子段和" << endl;
            cout << "3 凸多边形三角剖分" << endl;
            cout << "4 0-1 背包" << endl;
            cout << "5 退出" << endl;

            while (cin >> choose, !(choose >= 1 && choose <= 5))
            {
                cout << "输入不合法，请重新输入" << endl;
                cin.clear();
                cin.sync();
            }
            cout << "--------------------------------------------
--------------------" << endl;

            switch (choose)
            {
                case 1:
                {
                    ifstream in("附件 1.最长公共子序列输入数
据.txt", ios_base::in);
                    if (!in.is_open())
                    {
                        cout << "Error opening file..." << endl;
                        exit(1);
                    }

                    //读入 A，B，C，D 四个串
                    string A, B, C, D;
                    int conditon = 0;
                    char ch;
                    while (in >> ch)
                    {
                        if(ch >= 'A' && ch <= 'D' && in.get() ==
':')
                            conditon = ch;
                        else if(ch != ' ' && ch != '\n' && ch !=
':')
                            switch (conditon)
                            {
                                case 'A':
                                    A += ch;
                                    break;
                                case 'B':
                                    B += ch;
                                    break;
                                case 'C':
                                    C += ch;
                                    break;
                                case 'D':
                                    D += ch;
                                    break;
                            }
                        }

                        cout << "A-B 的最长公共子串：" << LCS(A, B) <<
endl;
                        cout << "C-D 的最长公共子串：" << LCS(C, D) <<
endl;
                        cout << "A-D 的最长公共子串：" << LCS(A, D) <<
endl;
                        cout << "C-B 的最长公共子串：" << LCS(C, B) <<
endl;

                        in.close();
                        cout << "------------------------------------
-----------------------------" << endl;
                        break;
                    }
                    case 2:
                    {
                        ifstream in1("附件 2.最大子段和输入数据-序列
1.txt", ios_base::in);
                        ifstream in2("附件 2.最大子段和输入数据-序列
2.txt", ios_base::in);
                        if (!in1.is_open() || !in2.is_open())
                        {
                            cout << "Error opening file..." << endl;
                            exit(1);
                        }

                        vector<int> a;
                        int num;
                        int left, right;
                        while(in1 >> num)
                            a.push_back(num);
                        int len = MaxSum(a, left, right);

                        cout << "序列 1 的最大子段:" << endl;

                        int sum = 0;
                        for (int i = left; i <= right; ++i)
                        {
                            cout << a[i] << ' ';
                            sum += a[i];
                        }
                        cout << endl << "最大字段为从" << left << "到
    " << right << " 和为：" << sum << endl;

                        a.clear();
                        while(in2 >> num)
                            a.push_back(num);
                        len = MaxSum(a, left, right);

                        cout << endl << "序列 2 的最大子段:" << endl;
                        for (int i = left; i <= right; ++i)
                        {
                            cout << a[i] << ' ';
                        }
                        cout << endl << "最大字段为从" << left << "到
    " << right << " 和为：" << len << endl;

                        in1.close();
                        in2.close();
                        cout << "------------------------------------
    -----------------------------" << endl;
                        break;
                    }
                    case 3:
                    {
                        ifstream in1("附件 3-1.21 个基站凸多边形数
    据.txt", ios_base::in);
                        ifstream in2("附件 3-2.29 个基站凸多边形数
    据.txt", ios_base::in);

                        if (!in1.is_open() || !in2.is_open())
                        {
                            cout << "Error opening file..." << endl;
                            exit(1);
                        }
                        int n = 0;
                        while (in1 >> stations[n].enodedId >>
    stations[n].longitude
                            >> stations[n].latitude >>
    stations[n].index)
                            n++;
                        cout << "21 个基站凸多边形的最优三角剖分值为：
    " << MinWeightTriangulation(n) << endl;
                        cout << "最优三角剖分结构为：" << endl;
                        TraceBack(1, n - 1);

                        n = 0;
                        while (in2 >> stations[n].enodedId >>
    stations[n].longitude
                            >> stations[n].latitude >>
    stations[n].index)
                            n++;
                        cout << endl << "29 个基站凸多边形的最优三角剖
    分值为： " << MinWeightTriangulation(n) << endl;
                        cout << "最优三角剖分结构为：" << endl;
                        TraceBack(1, n - 1);

                        cout << "------------------------------------
    -----------------------------" << endl;
                        break;
                    }
                    case 4:
                  {
                        ifstream in("附件 4.背包问题输入数据.txt",
    ios_base::in);
                        if (!in.is_open())
                        {
                            cout << "Error opening file..." << endl;
                            exit(1);
                        }

                        int c;
                        int num;
                        int maxn = 0;
                        string line;
                        vector<int> weight;
                        vector<int> value;
                        in >> c;
                        in.get(); //读取多余的换行符
                        getline(in, line);
                        istringstream iss(line);
                        while (iss >> num)
                            weight.push_back(num);
                      getline(in, line);
                      iss.clear(); //重置 iss 状态
                        iss.str(line);
                        while (iss >> num)
                            value.push_back(num);

                        vector<bool> result = Knapspack(c, weight,
    value, maxn);

                        cout << "第一组数据的最大背包装载价值: " <<
    maxn << endl;
                        cout << "装载的物品如下:(序号,重量,价值)"
    << endl;
                        for (int i = 0; i < result.size(); ++i)
                        {
                            if (result[i] == true)
                                cout << "(" << i+1 << "," <<
    weight[i] << "," << value[i] << ") ";
                        }
                        cout << endl;

                      weight.clear();
                      value.clear();
                      in >> c;
                        in.get(); //读取多余的换行符
                        getline(in, line);
                        iss.clear();
                        iss.str(line);
                        while (iss >> num)
                            weight.push_back(num);
                      getline(in, line);
                      iss.clear(); //重置 iss 状态
                        iss.str(line);
                        while (iss >> num)
                            value.push_back(num);

                        result = Knapspack(c, weight, value, maxn);
                        cout << endl << "第二组数据的最大背包装载价
    值： " << maxn << endl;
                        cout << "装载的物品如下：(序号，重量，价值)"
    << endl;
                        for (int i = 0; i < result.size(); ++i)
                        {
                            if (result[i] == true)
                                cout << "(" << i+1 << "," <<
    weight[i] << "," << value[i] << ") ";
                        }
                        cout << endl;
                        cout << "------------------------------------
    -----------------------------" << endl;
                        break;
                    }
                    default:
                        break;
                }
            }
            return 0;
    }
```

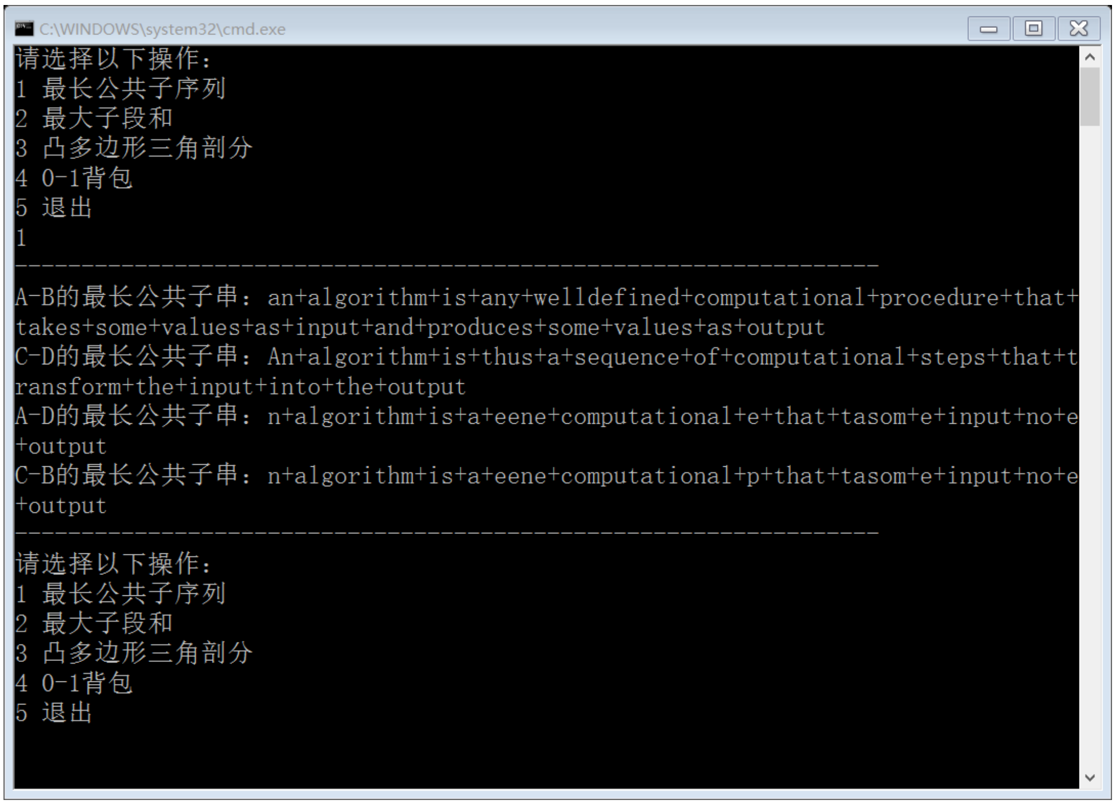
二、 运行结果

1. 开始界面（输入 1-5，选择相应操作）



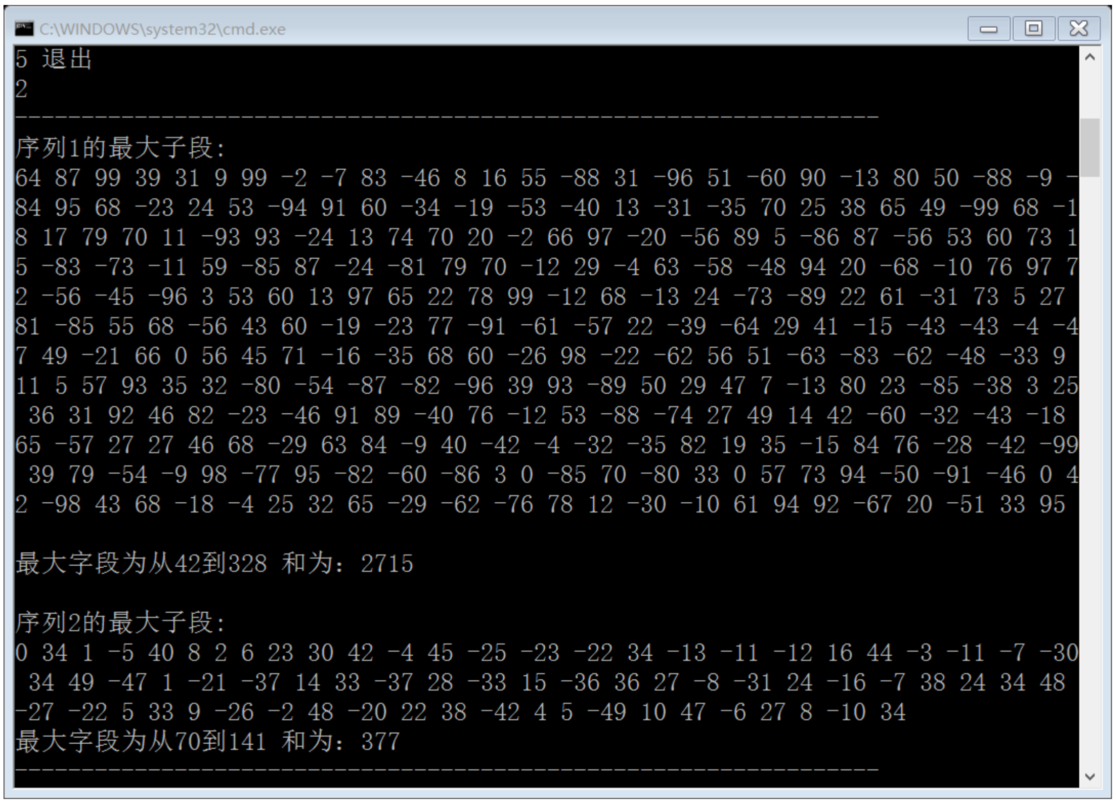
```
请选择以下操作:
1 最长公共子序列
2 最大子段和
3 凸多边形三角剖分
4 0-1背包
5 退出





中文 - QQ拼音输入法 半 :
```

2.　（输入 1）最长公共子序列



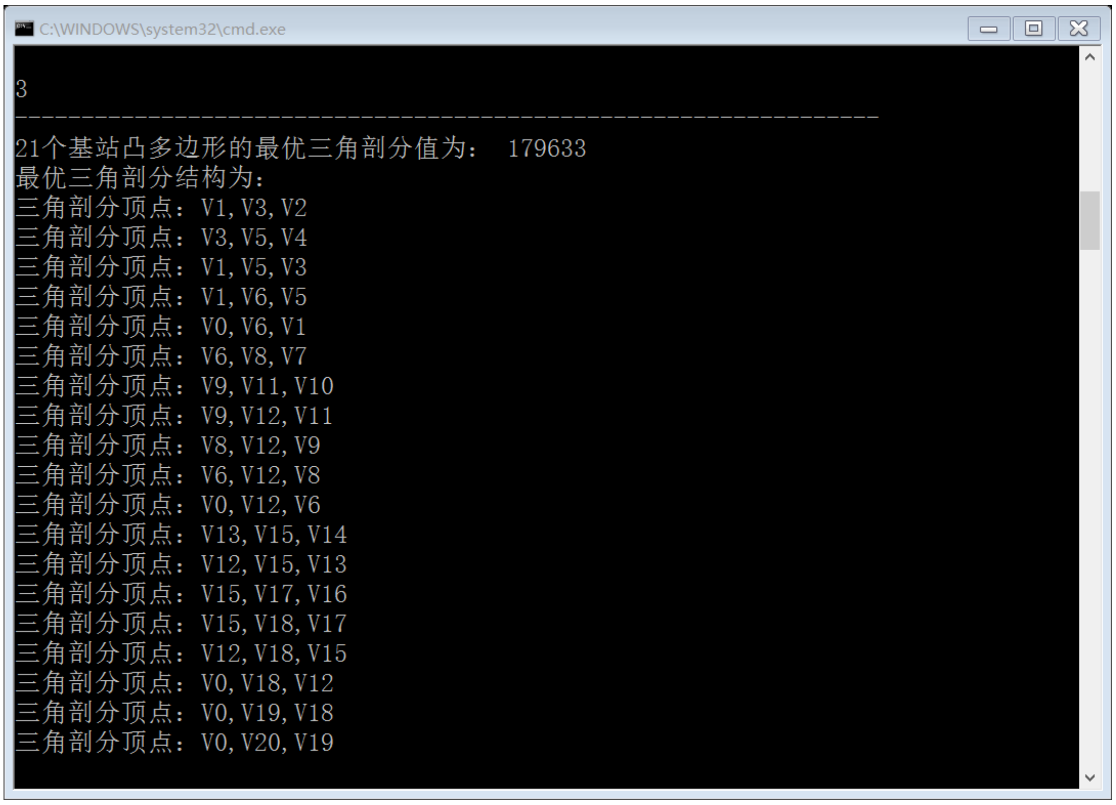
```
请选择以下操作:
1 最长公共子序列
2 最大子段和
3 凸多边形三角剖分
4 0-1背包
5 退出
1
-----------------------------------------------------------------
A-B的最长公共子串: an+algorithm+is+any+welldefined+computational+procedure+that+
takes+some+values+as+input+and+produces+some+values+as+output
C-D的最长公共子串: An+algorithm+is+thus+a+sequence+of+computational+steps+that+t
ransform+the+input+into+the+output
A-D的最长公共子串: n+algorithm+is+a+eene+computational+e+that+tasom+e+input+no+e
+output
C-B的最长公共子串: n+algorithm+is+a+eene+computational+p+that+tasom+e+input+no+e
+output
-----------------------------------------------------------------
请选择以下操作:
1 最长公共子序列
2 最大子段和
3 凸多边形三角剖分
4 0-1背包
5 退出
```

3.　（输入 2）最大字段和

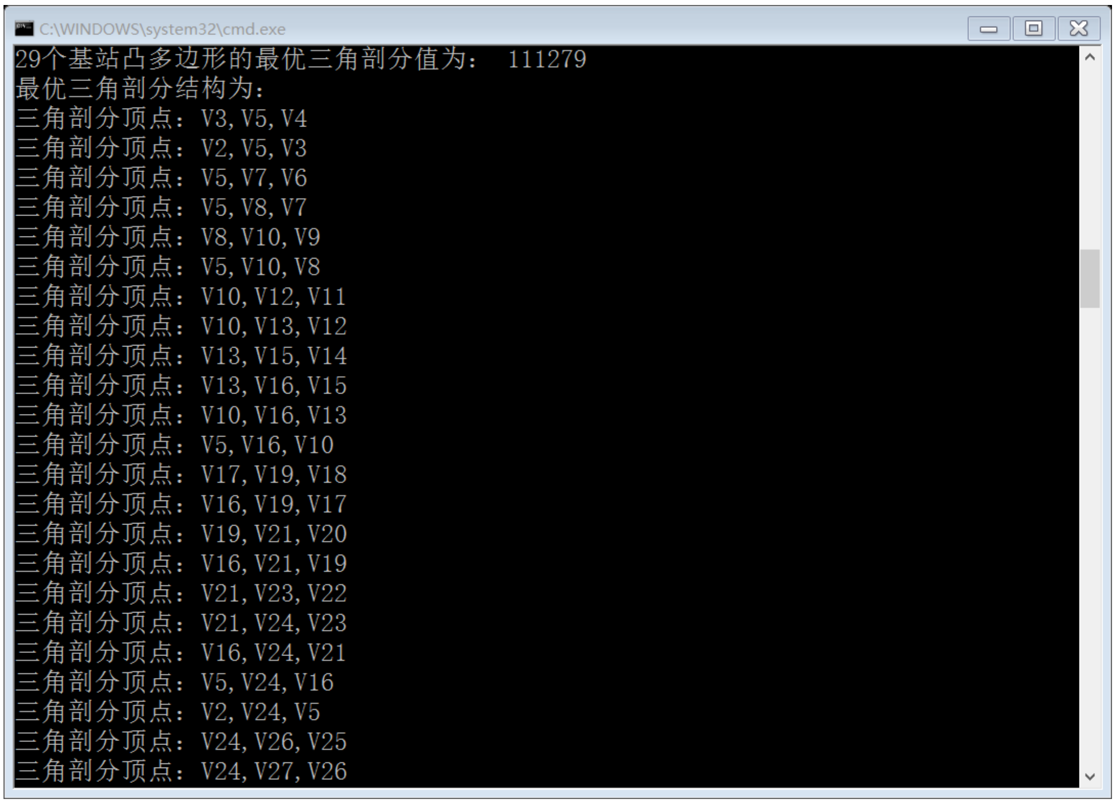


```
5 退出
2
-----------------------------------------------------------------
序列1的最大子段:
64 87 99 39 31 9 99 -2 -7 83 -46 8 16 55 -88 31 -96 51 -60 90 -13 80 50 -88 -9 -
84 95 68 -23 24 53 -94 91 60 -34 -19 -53 -40 13 -31 -35 70 25 38 65 49 -99 68 -1
8 17 79 70 11 -93 93 -24 13 74 70 20 -2 66 97 -20 -56 89 5 -86 87 -56 53 60 73 1
5 -83 -73 -11 59 -85 87 -24 -81 79 70 -12 29 -4 63 -58 -48 94 20 -68 -10 76 97 7
2 -56 -45 -96 3 53 60 13 97 65 22 78 99 -12 68 -13 24 -73 -89 22 61 -31 73 5 27
81 -85 55 68 -56 43 60 -19 -23 77 -91 -61 -57 22 -39 -64 29 41 -15 -43 -43 -4 -4
7 49 -21 66 0 56 45 71 -16 -35 68 60 -26 98 -22 -62 56 51 -63 -83 -62 -48 -33 9
11 5 57 93 35 32 -80 -54 -87 -82 -96 39 93 -89 50 29 47 7 -13 80 23 -85 -38 3 25
 36 31 92 46 82 -23 -46 91 89 -40 76 -12 53 -88 -74 27 49 14 42 -60 -32 -43 -18
65 -57 27 27 46 68 -29 63 84 -9 40 -42 -4 -32 -35 82 19 35 -15 84 76 -28 -42 -99
 39 79 -54 -9 98 -77 95 -82 -60 -86 3 0 -85 70 -80 33 0 57 73 94 -50 -91 -46 0 4
2 -98 43 68 -18 -4 25 32 65 -29 -62 -76 78 12 -30 -10 61 94 92 -67 20 -51 33 95

最大字段为从42到328 和为: 2715

序列2的最大子段:
0 34 1 -5 40 8 2 6 23 30 42 -4 45 -25 -23 -22 34 -13 -11 -12 16 44 -3 -11 -7 -30
 34 49 -47 1 -21 -37 14 33 -37 28 -33 15 -36 36 27 -8 -31 24 -16 -7 38 24 34 48
-27 -22 5 33 9 -26 -2 48 -20 22 38 -42 4 5 -49 10 47 -6 27 8 -10 34
最大字段为从70到141 和为: 377
-----------------------------------------------------------------
```

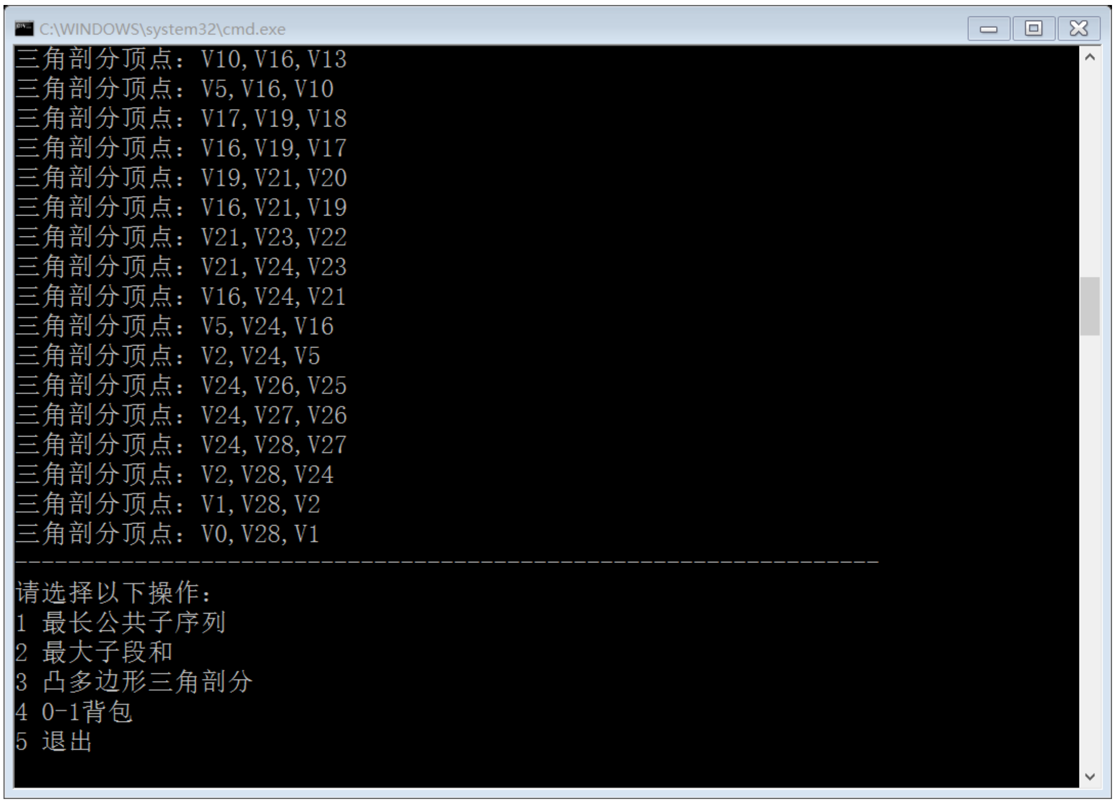
4. （输入 3）凸多边形三角剖分（后附剖分结果图，顶点用 A，B，C……代替）



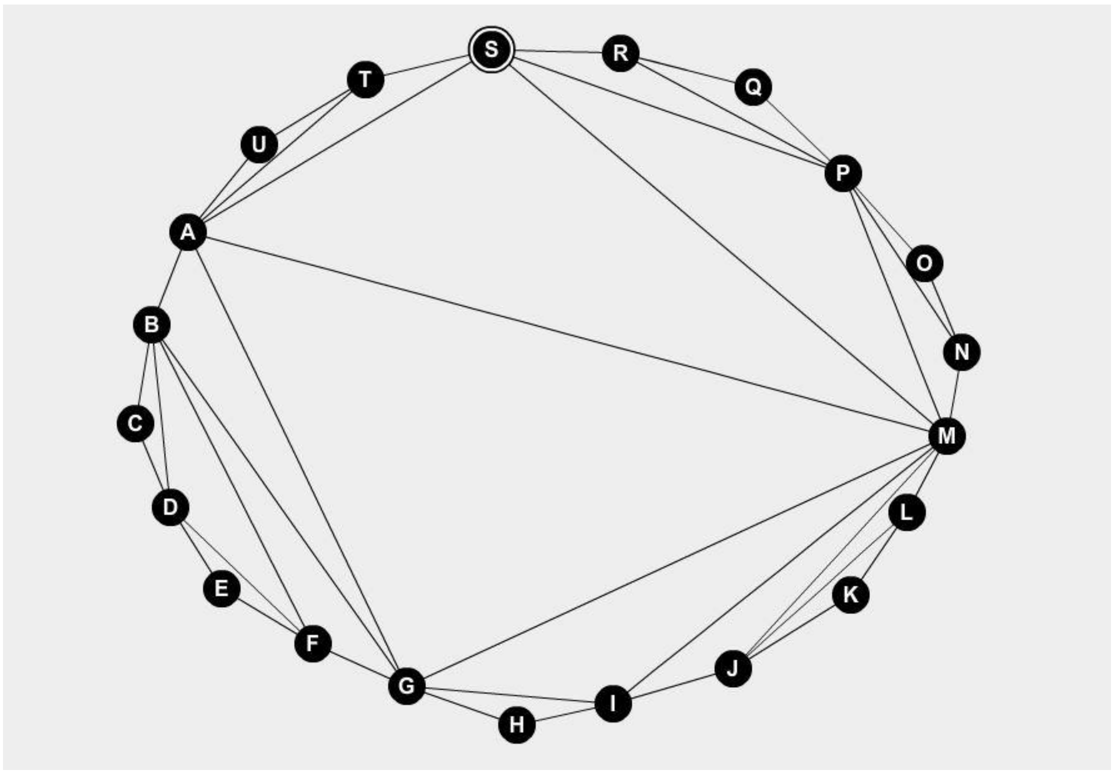
```
3
-----------------------------------------------------------------
21个基站凸多边形的最优三角剖分值为：  179633
最优三角剖分结构为：
三角剖分顶点：V1,V3,V2
三角剖分顶点：V3,V5,V4
三角剖分顶点：V1,V5,V3
三角剖分顶点：V1,V6,V5
三角剖分顶点：V0,V6,V1
三角剖分顶点：V6,V8,V7
三角剖分顶点：V9,V11,V10
三角剖分顶点：V9,V12,V11
三角剖分顶点：V8,V12,V9
三角剖分顶点：V6,V12,V8
三角剖分顶点：V0,V12,V6
三角剖分顶点：V13,V15,V14
三角剖分顶点：V12,V15,V13
三角剖分顶点：V15,V17,V16
三角剖分顶点：V15,V18,V17
三角剖分顶点：V12,V18,V15
三角剖分顶点：V0,V18,V12
三角剖分顶点：V0,V19,V18
三角剖分顶点：V0,V20,V19
```

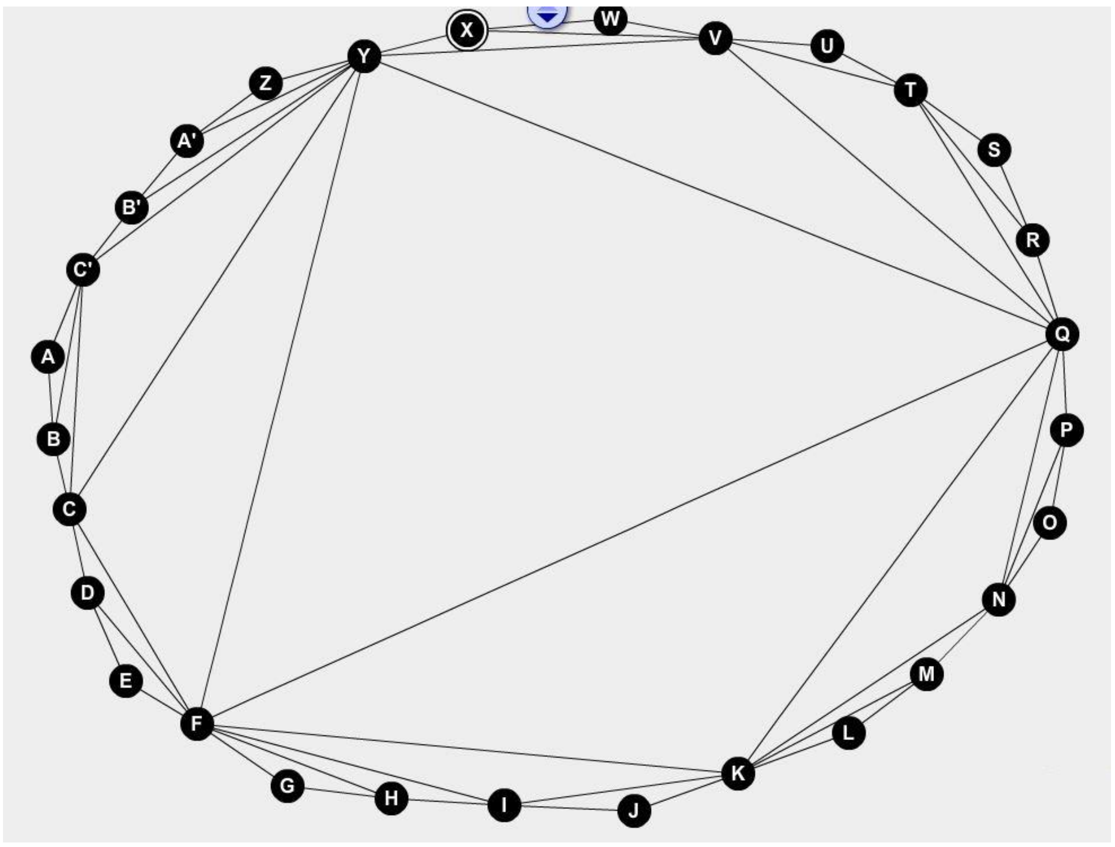


```
29个基站凸多边形的最优三角剖分值为：  111279
最优三角剖分结构为：
三角剖分顶点：V3,V5,V4
三角剖分顶点：V2,V5,V3
三角剖分顶点：V5,V7,V6
三角剖分顶点：V5,V8,V7
三角剖分顶点：V8,V10,V9
三角剖分顶点：V5,V10,V8
三角剖分顶点：V10,V12,V11
三角剖分顶点：V10,V13,V12
三角剖分顶点：V13,V15,V14
三角剖分顶点：V13,V16,V15
三角剖分顶点：V10,V16,V13
三角剖分顶点：V5,V16,V10
三角剖分顶点：V17,V19,V18
三角剖分顶点：V16,V19,V17
三角剖分顶点：V19,V21,V20
三角剖分顶点：V16,V21,V19
三角剖分顶点：V21,V23,V22
三角剖分顶点：V21,V24,V23
三角剖分顶点：V16,V24,V21
三角剖分顶点：V5,V24,V16
三角剖分顶点：V2,V24,V5
三角剖分顶点：V24,V26,V25
三角剖分顶点：V24,V27,V26
```

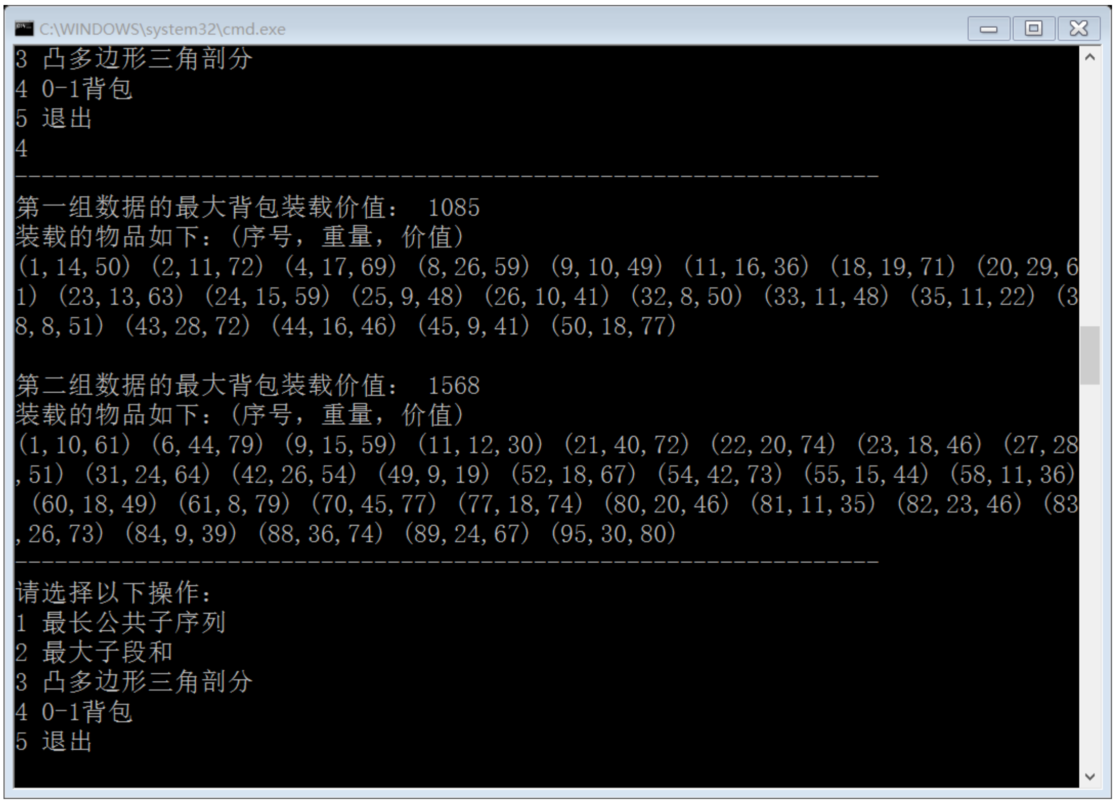


```
三角剖分顶点: V10,V16,V13
三角剖分顶点: V5,V16,V10
三角剖分顶点: V17,V19,V18
三角剖分顶点: V16,V19,V17
三角剖分顶点: V19,V21,V20
三角剖分顶点: V16,V21,V19
三角剖分顶点: V21,V23,V22
三角剖分顶点: V21,V24,V23
三角剖分顶点: V16,V24,V21
三角剖分顶点: V5,V24,V16
三角剖分顶点: V2,V24,V5
三角剖分顶点: V24,V26,V25
三角剖分顶点: V24,V27,V26
三角剖分顶点: V24,V28,V27
三角剖分顶点: V2,V28,V24
三角剖分顶点: V1,V28,V2
三角剖分顶点: V0,V28,V1
-----------------------------------------------------------------
请选择以下操作:
1 最长公共子序列
2 最大子段和
3 凸多边形三角剖分
4 0-1背包
5 退出
```





5.　（输入 4）0-1 背包



```
3 凸多边形三角剖分
4 0-1背包
5 退出
4
-----------------------------------------------------------------
第一组数据的最大背包装载价值： 1085
装载的物品如下：(序号, 重量, 价值)
(1,14,50) (2,11,72) (4,17,69) (8,26,59) (9,10,49) (11,16,36) (18,19,71) (20,29,6
1) (23,13,63) (24,15,59) (25,9,48) (26,10,41) (32,8,50) (33,11,48) (35,11,22) (3
8,8,51) (43,28,72) (44,16,46) (45,9,41) (50,18,77)

第二组数据的最大背包装载价值： 1568
装载的物品如下：(序号, 重量, 价值)
(1,10,61) (6,44,79) (9,15,59) (11,12,30) (21,40,72) (22,20,74) (23,18,46) (27,28
,51) (31,24,64) (42,26,54) (49,9,19) (52,18,67) (54,42,73) (55,15,44) (58,11,36)
 (60,18,49) (61,8,79) (70,45,77) (77,18,74) (80,20,46) (81,11,35) (82,23,46) (83
,26,73) (84,9,39) (88,36,74) (89,24,67) (95,30,80)
-----------------------------------------------------------------
请选择以下操作：
1 最长公共子序列
2 最大子段和
3 凸多边形三角剖分
4 0-1背包
5 退出
```

6.　（输入 5）退出