一些说明：
注：为了方便读取，将xls文件改为csv
注：所有题目均在一个cpp文件下，主函数中依次进行各个算法

代码：
```cpp
// Algorithm3.cpp : Designed by Xiao Yunming.
//

#include "stdafx.h" // VS projects head file

#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <cmath>
#include <functional>
#include <time.h>

#define EARTH_RADIUS 6378.137
#define M_PI 3.14159265358979323846

using namespace std;

// ----------------------------- LCS --------------------------------------

void LCSLength(int m, int n, string x, string y, int **c)
{
    int i, j;
    for (i = 0; i <= m; i++)  c[i][0] = 0;
    for (i = 0; i <= n; i++)  c[0][i] = 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;
            }
            else if (c[i - 1][j] > c[i][j - 1]) {
                c[i][j] = c[i - 1][j];
            }
            else {
                c[i][j] = c[i][j - 1];
            }
        }
}

string LCS(int i, int j, string x, int **c)
{
    if (i == 0 || j == 0)
        return "";
    if (c[i][j] == c[i - 1][j - 1] + 1) {
        return LCS(i - 1, j - 1, x, c) + x[i - 1];
    }
    else if (c[i][j] == c[i - 1][j])
        return LCS(i - 1, j, x, c);
    else
        return LCS(i, j - 1, x, c);
}

// --------------------------- MAX_SUM ------------------------------------

template<class Type>
int MaxSum(int n, Type a, int *pos)
{
    int sum = 0, b = 0;
    pos[0] = 0;
    pos[1] = 0;
    for (int i = 0; i < n; i++) {
        if (b > 0) {
            b += a[i];
        }
        else {
            b = a[i];
            pos[0] = pos[1] = i + 1;
        }
        if (b > sum) {
            sum = b;
            pos[1] = i + 1;
        }
    }
    return sum;
}

// ---------------------- MIN_WERIGHT_TRAIAN ------------------------------

class BaseStation
{
public:
    BaseStation(int enodebid, float longitude, float latitude);
    BaseStation(string line);
    BaseStation(const BaseStation &b);

    float longitude, latitude;
    int enodebid;
};

BaseStation::BaseStation(int enodebid, float longitude, float latitude)
{
    this->enodebid = enodebid;
    this->longitude = longitude;
    this->latitude = latitude;
}

BaseStation::BaseStation(string line)
{
    size_t pos = 0;
    size_t len = line.length();
    size_t delim_len = 1;
    // to devide the line by ','
    vector<string> s;
    while (pos < len) {
        int find_pos = line.find(',', pos);
        if (find_pos < 0) {
            s.push_back(line.substr(pos, len - pos));
            break;
        }
        s.push_back(line.substr(pos, find_pos - pos));
        pos = find_pos + delim_len;
    }

    this->enodebid = stoi(s[0]);
    this->longitude = stod(s[1]);
    this->latitude = stod(s[2]);
}

BaseStation::BaseStation(const BaseStation &b)
{
    this->enodebid = b.enodebid;
    this->longitude = b.longitude;
    this->latitude = b.latitude;
}

double GetDistance(BaseStation A, BaseStation B)
{
    auto rad = [](const double& f) {return f * M_PI / 180.0; };
    double radLatA = rad(A.latitude), radLatB = rad(B.latitude);
    double radLonA = rad(A.longitude), radLonB = rad(B.longitude);
    double s = 1000 * EARTH_RADIUS * acos(cos(radLatA) * cos(radLatB) * cos(radLonA - radLonB) +
sin(radLatA) * sin(radLatB));
    return s;
}

double Weight(vector<BaseStation> B, int a, int b, int c)
{
    double ab = GetDistance(B[a], B[b]);
    double ac = GetDistance(B[a], B[c]);
    double bc = GetDistance(B[b], B[c]);
    return ab + ac + bc;
}

void MinWerightTriangulation(int n, double **t, int **s, vector<BaseStation> B)
{
    for (int i = 0; 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(B, i - 1, i, j);
            s[i][j] = i;
            for (int k = i + 1; k < i + r - 1; k++) {
                int u = t[i][k] + t[k + 1][j] + Weight(B, i - 1, k, j);
                if (u < t[i][j]) {
                    t[i][j] = u;
                    s[i][j] = k;
                }
            }
        }
    }
}


// ---------------------------- KNAPSACK ---------------------------------

void Knapsack(vector<int> v, vector<int> w, int c, int **m)
{
    int n = v.size() - 1;
    auto min = [](int w, int c) {return w < c ? w : c; };
    auto max = [](int w, int c) {return w > c ? w : c; };

    int j_max = min(w[n] - 1, c);
    for (int j = 0; j <= j_max; j++)
        m[n][j] = 0;
    for (int j = w[n]; j <= c; j++)
        m[n][j] = v[n];

    for (int i = n - 1; i > 0; i--) {
        j_max = min(w[i] - 1, c);
        for (int j = 0; j <= j_max; j++)
            m[i][j] = m[i + 1][j];
        for (int j = w[i]; j <= c; j++)
            m[i][j] = max(m[i + 1][j], m[i + 1][j - w[i]] + v[i]);
    }
    m[0][c] = m[1][c];
    if (c > w[0])
        m[0][c] = max(m[0][c], m[1][c - w[0]] + v[0]);
}

void TraceBack(int **m, vector<int> w, int c, int *x)
{
    int n = w.size() - 1;
    for (int i = 0; i < n; i++) {
        if (m[i][c] == m[i + 1][c])
            x[i] = 0;
        else {
            x[i] = 1;
            c -= w[i];
        }
    }
    x[n] = (m[n][c] > 0 ? 1 : 0);
}

// ------------------------------ MAIN -----------------------------------

int main()
{
    ofstream fr("Result.txt");
    clock_t start, end;
    double duration;

    // LCS
    // init
    fstream f1("附件1.最长公共子序列输入文件2017-4-26.txt", ios::in | ios::out);
    string a1, b1, c1, d1;
    getline(f1, a1);     getline(f1, a1);
    getline(f1, b1);     getline(f1, b1);     getline(f1, b1);
    getline(f1, c1);     getline(f1, c1);     getline(f1, c1);
    getline(f1, d1);     getline(f1, d1);     getline(f1, d1);
    f1.close();

    int **c11 = new int*[a1.length() + 1];
    int **c12 = new int*[c1.length() + 1];
    int **c13 = new int*[a1.length() + 1];
    int **c14 = new int*[c1.length() + 1];
    for (int i = 0; i < a1.length() + 1; i++)
        c11[i] = new int[b1.length() + 1];
    for (int i = 0; i < c1.length() + 1; i++)
        c12[i] = new int[d1.length() + 1];
    for (int i = 0; i < a1.length() + 1; i++)
        c13[i] = new int[d1.length() + 1];
    for (int i = 0; i < c1.length() + 1; i++)
        c14[i] = new int[b1.length() + 1];

    // the algorithm & output
    start = clock();
    LCSLength(a1.length(), b1.length(), a1, b1, c11);
    LCSLength(c1.length(), d1.length(), c1, d1, c12);
    LCSLength(a1.length(), d1.length(), a1, d1, c13);
    LCSLength(c1.length(), b1.length(), c1, b1, c14);
    string result11 = LCS(a1.length(), b1.length(), b1, c11);
    string result12 = LCS(c1.length(), d1.length(), d1, c12);
    string result13 = LCS(a1.length(), d1.length(), d1, c13);
    string result14 = LCS(c1.length(), b1.length(), b1, c14);
    end = clock();
    duration = (double)(end - start);
    fr << "1.最长公共子序列:" << endl;
    fr << "A-B: " << result11 << endl;
    fr << "C-D: " << result12 << endl;
    fr << "A-D: " << result13 << endl;
    fr << "C-B: " << result14 << endl;
    fr << "耗时" << duration / CLOCKS_PER_SEC << "秒" << endl;
    fr << endl << endl;


    // MAX_SUM
    // init
    fstream f21("附件2.最大子段和输入数据2017-序列1.txt", ios::in | ios::out);
    fstream f22("附件2.最大子段和输入数据2017-序列2.txt", ios::in | ios::out);
    vector<int> v21, v22;
    string temp;
    v21.reserve(300);    v22.reserve(100);
    while (getline(f21, temp) && f21.good())
        v21.push_back(std::stoi(temp));
    while (getline(f22, temp) && f22.good())
        v22.push_back(std::stoi(temp));
    f21.close();
    f22.close();

    // the algorithm & output
    int result21_pos[2], result22_pos[2];
    start = clock();
    int result21 = MaxSum(v21.size(), v21, result21_pos);
    int result22 = MaxSum(v22.size(), v22, result22_pos);
    end = clock();
    duration = (double)(end - start);
    fr << "2.最大子段和:" << endl;
    fr << "序列1：从第 " << result21_pos[0] << " 位到第 " << result21_pos[1] << " 位的和为最大和: " << result21
<< endl;
    fr << "序列1：从第 " << result22_pos[0] << " 位到第 " << result22_pos[1] << " 位的和为最大和: " << result22
<< endl;
    fr << "耗时" << duration / CLOCKS_PER_SEC << "秒" << endl;
    fr << endl << endl;


    // MIN_WERIGHT_TRAIAN
    // init
    fstream f31("附件3-1.21个基站凸多边形数据2017.csv", ios::in | ios::out);
    fstream f32("附件3-2.29个基站凸多边形数据2017.csv", ios::in | ios::out);
    vector<BaseStation> b21, b22;
    getline(f31, temp);      getline(f32, temp); // file header
    while (getline(f31, temp) && f31.good())
        b21.push_back(BaseStation(temp));
    while (getline(f32, temp) && f32.good())
        b22.push_back(BaseStation(temp));
    f31.close();
    f32.close();

    double **result31_t = new double*[b21.size()], **result32_t = new double*[b22.size()];
    int **result31_s = new int*[b21.size()], **result32_s = new int*[b22.size()];

    for (int i = 0; i < b21.size(); i++) {
        result31_t[i] = new double[b21.size()];
        result31_s[i] = new int[b21.size()];
    }
    for (int i = 0; i < b22.size(); i++) {
        result32_t[i] = new double[b22.size()];
        result32_s[i] = new int[b22.size()];
    }

    // the algorithm
    start = clock();
    MinWerightTriangulation(b21.size() - 1, result31_t, result31_s, b21);
    MinWerightTriangulation(b22.size() - 1, result32_t, result32_s, b22);
    end = clock();
    duration = (double)(end - start);

    // output
    std::function<void(int, int, double**, int**, ofstream&)> ShowResult3;
    ShowResult3 = [&ShowResult3](int i, int j, double **t, int **s, ofstream &fr)
    {
        int k = s[i][j];
        fr << "(" << i - 1 << ", " << k << ", " << j << "), weight = " << t[i][j] << endl;
        if (k - i >= 2)
            ShowResult3(i, k, t, s, fr);
        else if (k - i == 1)
            fr << "(" << i - 1 << ", " << i << ", " << k << "), weight = " << t[i][k] << endl;
        if (j - k >= 2)
            ShowResult3(k + 1, j, t, s, fr);
    };

    double result31_w, result32_w;
    fr << "3.凸多边形最优三角部分:" << endl;
    fr << "21凸多边形：（每个三维序列表示一个三角剖分的三个顶点）" << endl;
    // 获得边长
    ShowResult3(1, b21.size() - 1, result31_t, result31_s, fr);
    result31_w = result31_t[1][b21.size()-1];
    auto it1 = b21.begin();
    while ((it1+1) != b21.end()) {
        result31_w += GetDistance(*it1, *(it1 + 1));
        it1++;
    }
    result31_w += GetDistance(b21[0], *it1);
    result31_w = result31_w / 2;
    fr << "最小边长弦长和为：" << result31_w << endl << endl;
    fr << "29凸多边形：（每个三维序列表示一个三角剖分的三个顶点）" << endl;
    ShowResult3(1, b22.size() - 1, result32_t, result32_s, fr);
    result32_w = result32_t[1][b22.size() - 1];
    // 获得边长
    auto it2 = b22.begin();
    while ((it2 + 1) != b22.end()) {
        result32_w += GetDistance(*it2, *(it2 + 1));
        it2++;
    }
    result32_w += GetDistance(b21[0], *it2);
    result32_w = result32_w / 2;
    fr << "最小边长弦长和为：" << result32_w << endl;
    fr << "耗时" << duration / CLOCKS_PER_SEC << "秒" << endl;
    fr << endl << endl;

    // KNAPSACK
    // init, kind of complex
    int c41, c42;
    vector<int> w41, w42;
    vector<int> v41, v42;
    fstream f4("附件4.背包问题输入数据2017.txt", ios::in | ios::out);

    std::function<vector<int>(string, string)> StringParse; // to parse the long line to pieces with "delim"
    StringParse = [](string s, string delim)
    {
        vector<string> str;
        vector<int>result_int;
        size_t pos = 0;
        size_t len = s.length();
        size_t delim_len = delim.length();
        while (pos < len) {
            int find_pos = s.find(delim, pos);
            if (find_pos < 0) {
                str.push_back(s.substr(pos, len - pos));
                break;
            }
            str.push_back(s.substr(pos, find_pos - pos));
            pos = find_pos + delim_len;
        }

        vector<string>::iterator it = str.begin();
        while (it != str.end()) {
            it->erase(0, it->find_first_not_of(' '));
            result_int.push_back(stoi(*it));
            it++;
        }
        return result_int;
    };

    getline(f4, temp);
```

```
    getline(f4, temp);
    while (temp[0] < '0' || temp[0] > '9')temp = temp.substr(1, temp.length() - 1);
    while (temp[temp.length() - 1] < '0' || temp[temp.length() - 1] > '9')  temp = temp.substr(0, 
temp.length() - 1);
    c41 = stoi(temp);
    getline(f4, temp);       getline(f4, temp);
    getline(f4, temp);
    w41 = StringParse(temp, " ");
    getline(f4, temp);       getline(f4, temp);       getline(f4, temp);
    v41 = StringParse(temp, " ");

    getline(f4, temp);   getline(f4, temp);   getline(f4, temp);
    getline(f4, temp);   getline(f4, temp);   getline(f4, temp);
    getline(f4, temp);
    while (temp[0] < '0' || temp[0] > '9')temp = temp.substr(1, temp.length() - 1);
    while (temp[temp.length() - 1] < '0' || temp[temp.length() - 1] > '9')  temp = temp.substr(0, 
temp.length() - 1);
    c42 = stoi(temp);
    getline(f4, temp);       getline(f4, temp);
    getline(f4, temp);
    w42 = StringParse(temp, " ");
    getline(f4, temp);       getline(f4, temp);       getline(f4, temp);
    v42 = StringParse(temp, " ");

    int **m41 = new int*[w41.size() + 1], **m42 = new int*[w42.size() + 1];
    for (int i = 0; i < w41.size() + 1; i++)
        m41[i] = new int[c41 + 1];
    for (int i = 0; i < w42.size() + 1; i++)
        m42[i] = new int[c42 + 1];
    int *result41 = new int[w41.size()], *result42 = new int[w42.size()];

    // the algorigthm
    start = clock();
    Knapsack(v41, w41, c41, m41);
    TraceBack(m41, w41, c41, result41);
    Knapsack(v42, w42, c42, m42);
    TraceBack(m42, w42, c42, result42);
    end = clock();
    duration = (double)(end - start);

    // output
    int total_weight1 = 0, total_weight2 = 0, total_value1 = 0, total_value2 = 0;
    fr << "4.0-1背包问题:" << endl;
    fr << "第1组放入的序号:";
    for (int i = 0; i < w41.size(); i++) {
        if (result41[i] == 1) {
            fr << i + 1 << " ";
            total_weight1 += w41[i];
            total_value1 += v41[i];
        }
    }
    fr << endl << "物品总重为:" << total_weight1 << endl;
    fr << "物品总价值为:" << total_value1 << endl;
    fr << endl;

    fr << "第2组放入的序号:";
    for (int i = 0; i < w42.size(); i++) {
        if (result42[i] == 1) {
            fr << i + 1 << " ";
            total_weight2 += w42[i];
            total_value2 += v42[i];
        }
    }
    fr << endl << "物品总重为:" << total_weight2 << endl;
    fr << "物品总价值为:" << total_value2 << endl;
    fr << "耗时" << duration / CLOCKS_PER_SEC << "秒" << endl;
    fr << endl;
    fr.close();
    for (int i = 0; i < a1.length() + 1; i++)
        delete c11[i];
    for (int i = 0; i < c1.length() + 1; i++)
        delete c12[i];
    for (int i = 0; i < a1.length() + 1; i++)
        delete c13[i];
    for (int i = 0; i < c1.length() + 1; i++)
        delete c14[i];
    delete c11;
    delete c12;
    delete c13;
    delete c14;

    for (int i = 0; i < b21.size(); i++) {
        delete result31_t[i];
        delete result31_s[i];
    }
    for (int i = 0; i < b22.size(); i++) {
        delete result32_t[i];
        delete result32_s[i];
    }
    delete result31_s;
    delete result31_t;
    delete result32_s;
    delete result32_t;

    for (int i = 0; i < w41.size() + 1; i++)
        delete m41[i];
    for (int i = 0; i < w42.size() + 1; i++)
        delete m42[i];
    delete result41;
    delete result42;
}
```

结果部分：

1.最长公共子序列:

A-B:

an+algorithm+is+any+elldefined+computational+procedure+that+takes+some+values+as+input+and+produces+some+values+as+output

C-D:

an+algorithm+is+any+elldefined+computational+procedure+that+takes+some+values+as+input+and+produces+some+values+as+output

A-D:

an+algorithm+is+any+elldefined+computational+procedure+that+takes+some+values+as+input+and+produces+some+values+as+output

C-B:

an+algorithm+is+any+elldefined+computational+procedure+that+takes+some+values+as+input+and+produces+some+values+as+output

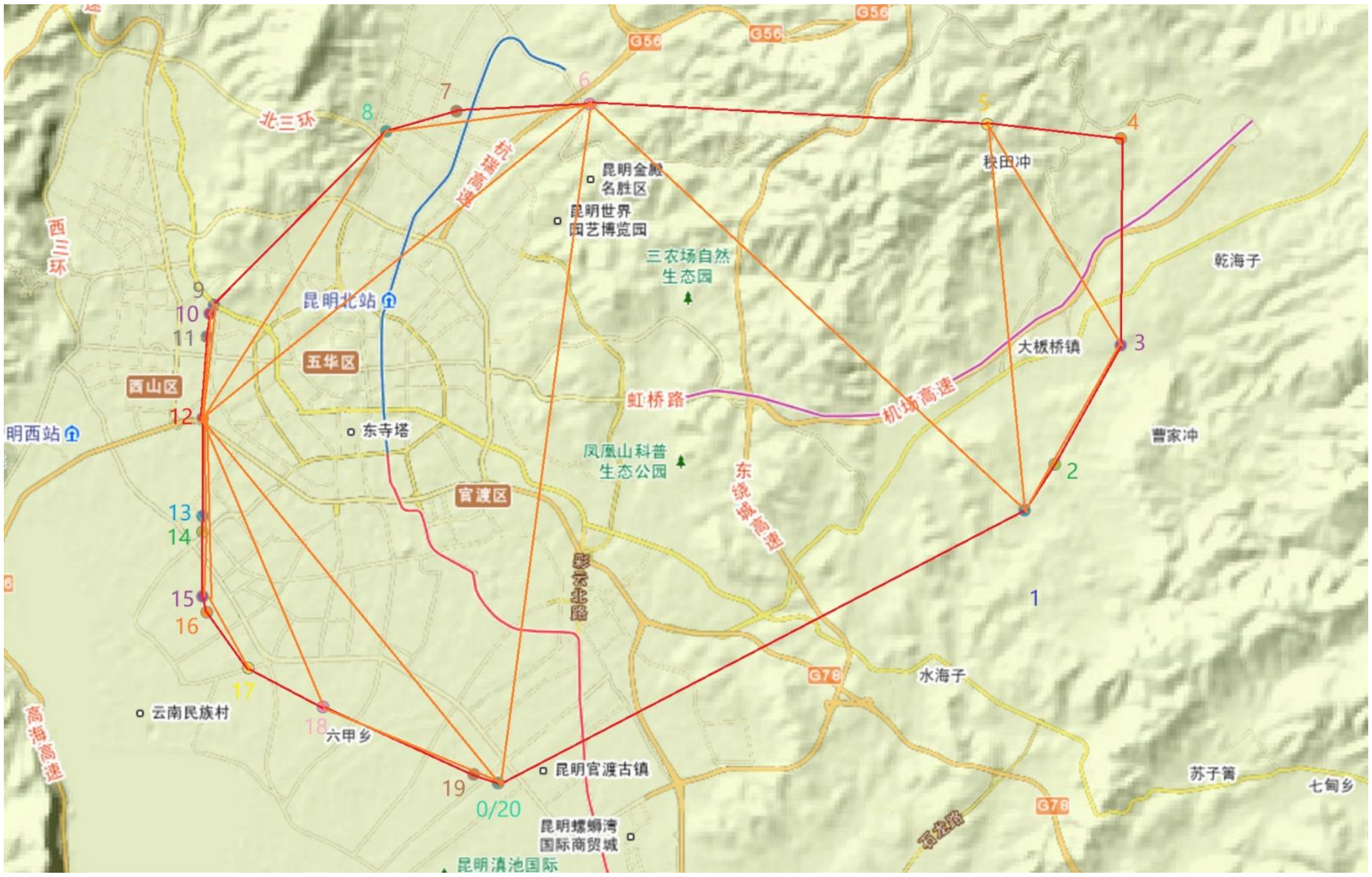
耗时 1.972 秒

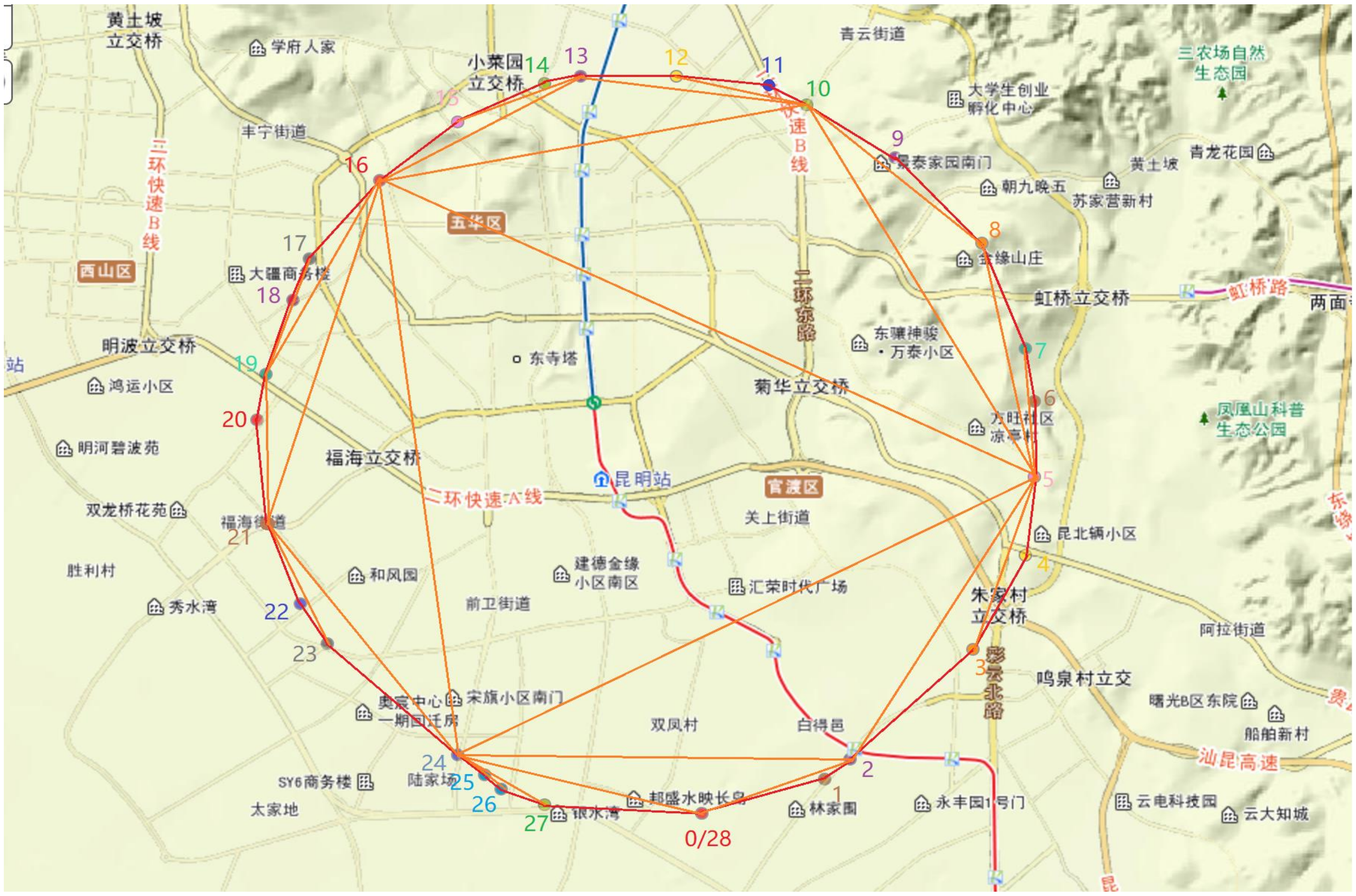
2.最大子段和:

```
序列 1：从第  43  位到第  329  位的和为最大和: 2715
序列 1：从第  72  位到第  142  位的和为最大和: 377
耗时 0 秒
```

3.凸多边形最优三角部分:

```
21 凸多边形：(每个三维序列表示一个三角剖分的三个顶点)
(0, 19, 20), weight = 295839
(0, 18, 19), weight = 294569
(0, 12, 18), weight = 285341
(0, 6, 12), weight = 221436
(0, 1, 6), weight = 122289
(1, 5, 6), weight = 77131
(1, 3, 5), weight = 43854
(1, 2, 3), weight = 9172.32
(3, 4, 5), weight = 14487.7
(6, 8, 12), weight = 59255
(6, 7, 8), weight = 9987.46
(8, 9, 12), weight = 23997.6
(9, 11, 12), weight = 7116
(9, 10, 11), weight = 1581.64
(12, 15, 18), weight = 40413
(12, 13, 15), weight = 12379
(13, 14, 15), weight = 3831.62
(15, 17, 18), weight = 12239
(15, 16, 17), weight = 4154.68
最小边长弦长和为：179633
```



```
29 凸多边形：（每个三维序列表示一个三角剖分的三个顶点）
(0, 1, 28), weight = 194320
(1, 2, 28), weight = 191369
(2, 24, 28), weight = 187667
(2, 5, 24), weight = 168803
(2, 3, 5), weight = 12648.9
(3, 4, 5), weight = 4470.42
(5, 16, 24), weight = 140122
(5, 10, 16), weight = 70756
(5, 8, 10), weight = 25036
(5, 7, 8), weight = 8989
(5, 6, 7), weight = 3148.76
(8, 9, 10), weight = 5279.92
(10, 13, 16), weight = 27078
(10, 12, 13), weight = 8432
(10, 11, 12), weight = 3128.23
(13, 15, 16), weight = 8340
(13, 14, 15), weight = 3039.24
(16, 21, 24), weight = 46470
(16, 19, 21), weight = 20910
(16, 17, 19), weight = 8460.79
(17, 18, 19), weight = 3000.26
(19, 20, 21), weight = 3591.39
(21, 23, 24), weight = 10554
(21, 22, 23), weight = 3288.36
(24, 27, 28), weight = 9570
(24, 26, 27), weight = 3661
(24, 25, 26), weight = 1296.56
最小边长弦长和为：112926
耗时 0.073 秒
```


```
4.0-1 背包问题:
第 1 组放入的序号:1 2 4 8 9 11 18 20 23 24 25 26 32 33 35 38 43 44 45 50
物品总重为:298
物品总价值为:1085

第 2 组放入的序号:1 6 9 11 21 22 23 27 31 42 49 52 54 55 58 60 61 70 77 80 81 82 83 84 88 89 95
物品总重为:600
物品总价值为:1568
耗时 0.033 秒
```