Polygan Triangulation

برای پیدا کردن مجموع کوتاهترین قطرها به کمک برنامه نویسی پویا هر چند ضلعی را به بخش تقسیم میکنم و سعی میکنیم تمام قسمت های موجود را حساب کرده و مینیمم آنهارا در هر مرحله برای چند ضلعی بزرگتر در نظر بگیریم

جدولی مانند شکل زیر داریم که در هر بخش i,j مقدار مجموع قطر های مینیمم در چند ضلعی تشکیل شده از i تا j را می نویسیم بنابر این کافی است مقدار 0,n-1 را بدست آوریم که به معنی کمترین مجموع قطر ها در چندضلعی داده شده میباشد

برای بدست آوردن هر یک از این مقادیر چندضلعی i تا j را به سه بخش یک مثلث و دو چند ضلعی تقسیم میکنیم یعنی : چندضلعی i تا k مثلث ikj و چندضلعی k تا j .

بنابر این می توان تعداد قطر های هر چندضلعی ij را به صورت زیر نوشت:

D[i][j] = MIN { D[i][k] + D[k][j] + { مجموع ضلعهای مثلث بین که قطرهستند (i<k<j)

برای تمام k های بین i j مقدار مینیمم را حساب کرده و در D ذخیره میکنیم

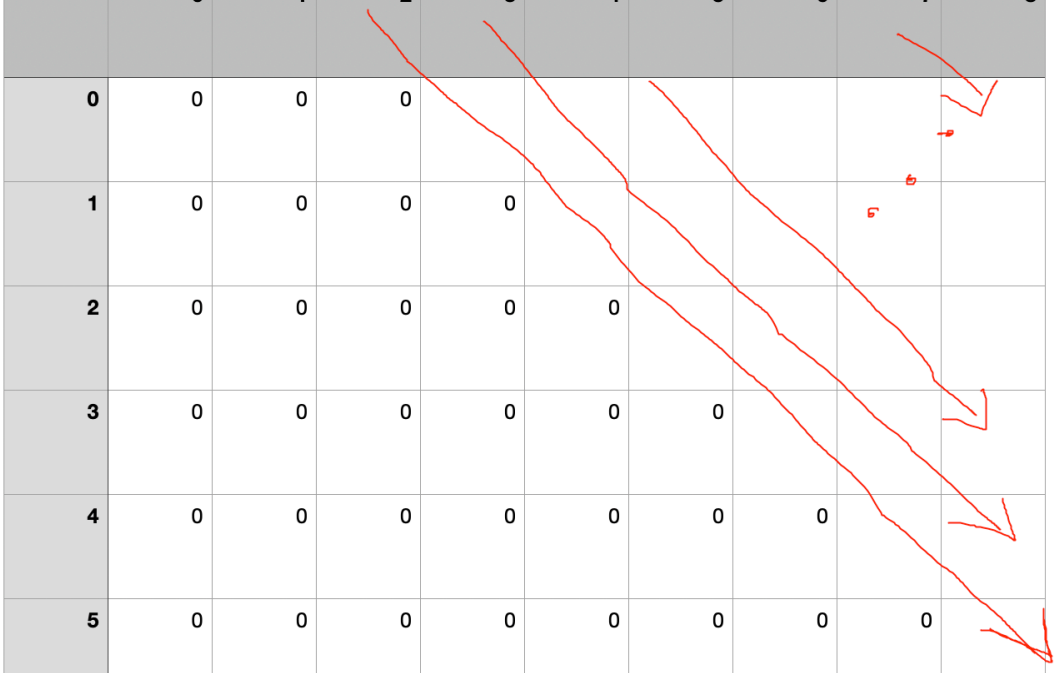
مجموع ضلعهای مثلث بین که قطر هستند یعنی اینکه در مثلث ikj اضلاعی که در چندضلعی i تا j قطر هستند را حساب میکنم. میدانیم که ij یکی از اضلاع است پس کافی است دو ضلع ik و kj راچک کنیم ببینیم که کنار هم هستند یا نه سپس میتوان مقدار D[i][j] را حساب کرد.

در جدول D مقادیر زیر در حالت اولیه برابر صفر میباشند و کافی است مقادیر بعدی را به صورت قطری و بر حسب مقدار های حساب شده در جدول و به کمک فرمول بالا حساب کرد.

در این جدول مقادیری که شماره ستون کمتر از شماره سطر به اضافه ی ۳ کمتر باشد صفر شده چون یا تشکیل چندضلعی نمی دهند یا مثلث هستند که قطر ندارند بنابراین بقیه مقادیر را پر میکنیم

Table 1

	0	1	2	3	4	5	6	7	8
0	0	0	0						
1	0	0	0	0					
2	0	0	0	0	0				
3	0	0	0	0	0	0			
4	0	0	0	0	0	0	0		
5	0	0	0	0	0	0	0	0	
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0



جهت پر شدن خانه ها نشان داده شده است.

کد:

دارای دو فایل main و Point می باشد که در Main گرافیک و منطق برنامه و در Point کلاس نقطه ها نوشته شده است.

Point:

کلاس ساده ای میباشد که فقط مختصات ایکس و ایگرگ داده شده را ذخیره میکند

مقادیر ایکس و ایگرگ را با مقدار داده شده ضرب در UNIT که برابر تعداد پیکسل های طول یک واحد صفحه کارتزین است ذخیره میکنیم

این کلاس دارای متد distance میباشد یه یک نقطه دیگر دریافت میکند و فاصله تا آن نقطه را بر میگرداند.

Main:

متد start ابتدا صفحه را میکشیم یک خط افقی و یک خط عمودی در مرکز و بقیه خطوط را اضافه میکنیم و در قسمت پایینی صفحه مطلوب مسأله یعنی کمترین مجموع قطر ها را نشان میدهیم.

در متد calculate مطلوب مسأله را حساب میکنیم ابتدا تعداد نقاط سپس مختصات نقاط وارد شده را میگیریم و برای هر کدام نمونه ای از کلاس Point میسازیم

سپس آرایه دو بعدی D که همان جدول صفحه بالا می باشد و آرایه دو بعدی diagonals که شامل لیستی از جنس Line میباشد در نظر میگیریم که diagonal[i][j] نشان دهنده ی خطوط قطر های چندضلعی i تا j میباشد.

و در انتها قصد داریم D[0][n-1] و diagonal[0][n-1] را بدست آوریم و بکشیم.

و آن دو آرایه را مقدار دهی اولیه می کنیم.

```
75      public static double calculate(Group group) {
76          Scanner input = new Scanner(System.in);
77          int n = input.nextInt();
78
79          ArrayList<Point> points = new ArrayList<>();
80          for (int i = 0; i < n; i++) {
81              double x = input.nextInt() * UNIT;
82              double y = input.nextInt() * UNIT;
83
84              points.add(new Point(x, -y));
85          }
86
87          double[][] D = new double[n][n];
88          ArrayList<Line>[][] diagonals = new ArrayList[n][n];
89
90          for (int i = 0; i < n; i++) {
91              for (int j = 0; j < n; j++) {
92                  D[i][j] = 0;
93                  diagonals[i][j] = new ArrayList();
94
95              }
96          }
```

حال شروع میکنیم و در همان جهتی که در جدول کشیده شده است جدول را با فرمول داده شده پر میکنیم ابتدا از خانه هایی شروع میکنیم که اختلافشات (s) ۳ است و تا انتها میرویم مقدار a نشان دهنده ی i و مقدار a+s برابر مقدار j است.

در هر مرحله از i بزرگتر از a تا کوچکتر از a+s مقدار مینیمم حاصل را حساب میکنیم و در

D[i][j] = MIN { D[i][k] + D[k][j] + مجموع ضلعهای مثلث بین که قطرهستند } (i<k<j)

ذخیره میکنیم.

برای "مجموع ضلعهای مثلث بین که قطرهستند " از متد dist استفاده میکنیم که شماره سه راس را (I a a+s) را به همراه لیست نقاط برای یافتن مختصات آنها به آن میدهیم و چک میکند که آیا این نقاط همسایه هستند یا نه (در صورت همسایه بودن قطر نیستند ضلع میباشند) سپس مجموع قطر ها را بر میگرداند تا مقدار فرمول بالا محاسبه شود.

```
155     public static double dist(int a, int k, int b, ArrayList<Point> points) {
156         double sum = 0;
157         if (k - a > 1) {
158             sum += points.get(k).distance(points.get(a),UNIT);
159         }
160 
161         if (b - k > 1) {
162             sum += points.get(k).distance(points.get(b),UNIT);
163         }
164 
165         return sum;
166     }
167 }
168 
```

و سپس برای قطر ها هم به همین صورت قطر های موجود در چندضلعی a تا a+s برابر مینیمم مجموع قطر های aتا i و قطرهای i تا a+s و اضلاع مثلث a i a+s که میتواند قطر باشد میباشد.

```
98          for (int s = 3; s < n; s++) {
99              for (int a = 0; a < n - s; a++) { // x = a , y = s + a
100                 D[a][s + a] = Integer.MAX_VALUE;
101                 for (int i = a + 1; i < s + a; i++) {
102                     if (D[a][s + a] > D[a][i] + D[i][s + a] + dist(a, i, b: s + a, points)) {
103                         D[a][s + a] = D[a][i] + D[i][s + a] + dist(a, i, b: s + a, points);
104                         diagonals[a][s + a].clear();
105                         diagonals[a][s + a].addAll(diagonals[a][i]);
106                         diagonals[a][s + a].addAll(diagonals[i][a + s]);
107                         addDiagonal(a, i, b: a + s, points, diagonals[a][s + a]);
108                     }
109                 }
110             }
111         }
```

برای اینکه بدانیم کدام یک از اضلاع ai یا i(a+s) میتواند قطر باشد از متد addDiagonal استفاده میکنیم و شماره سه راس را (l a a+s) را به همراه لیست نقاط برای یافتن مختصات آنها و لیست قطر ها به آن میدهیم تا در لیست قطر ها ذخیره کند

```
135    public static void addDiagonal(int a, int k, int b, ArrayList<Point> points, ArrayList<Line> d) {
136        if (k - a > 1) {
137            Line l = new Line(points.get(a).getX(), points.get(a).getY(),
138                    points.get(k).getX(), points.get(k).getY());
139            l.setStroke(Color.GREEN);
140            l.setStyle("   -fx-stroke-width: 1.5; ");
141
142            d.add(l);
143        }
144
145        if (b - k > 1) {
146            Line l = new Line(points.get(k).getX(), points.get(k).getY(),
147                    points.get(b).getX(), points.get(b).getY());
148            l.setStroke(Color.GREEN);
149            l.setStyle("   -fx-stroke-width: 1.5; ");
150
151            d.add(l);
152        }
153    }
```

چک میکند که آیا راس وسط یعنیi با راس های کناری اش پشت سر هم هستند (در این صورت قطر نیستند جز راس های چند ضلعی اند) در غیر این صورت یک خط میسازد و در لیست قطر های آن چند ضلعی ذخیره میکند.

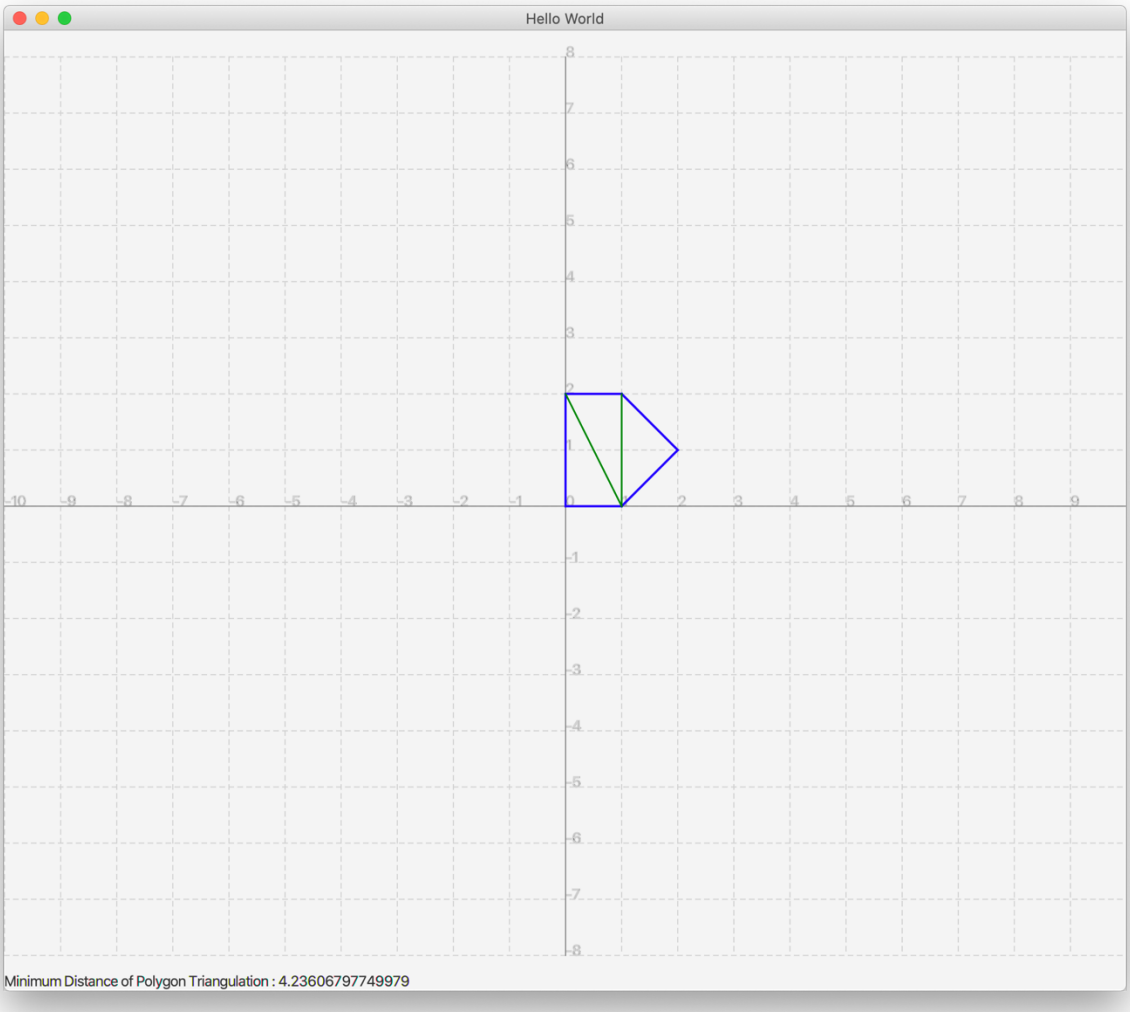
و در انتها چندضلعی را میکشیم سپس قطر ها و مجموع مینیمم مطلوب را میکشیم و بر میگردانیم D[0][n-1] و diagonal[0][n-1] .

```
113        for (int i = 0; i < points.size(); i++) {
114            Line l = new Line(points.get(i).getX(),
115                    points.get(i).getY(),
116                    points.get((i + 1) % points.size()).getX(),
117                    points.get((i + 1) % points.size()).getY());
118            l.setStroke(Color.BLUE);
119            l.setStyle("   -fx-stroke-width: 2; ");
120            group.getChildren().add(l);
121
122        }
123
124        ArrayList<Line> ds = diagonals[0][n - 1];
125
126        for (Line l : ds) {
127            group.getChildren().add(l);
128        }
129
130        return D[0][n - 1];
```

اجرا:

ابتدا در ترمینال تعداد نقاط و سپس مختصات نقاط را میدهیم سپس شکل نمایش داده می شود.

```
5
0 0
1 0
2 1
1 2
0 2
Minimum Distance of Polygon Triangulation : 4.23606797749979
```



نمونه دیگر:

مقدار UNIT را کمتر میکنیم که شکل بزرگتر به خوبی نشان داده شود (UNIT = 20)

```
8
-5 -5
5 -5
10 0
10 5
5 10
-5 10
-10 5
-10 0
Minimum Distance of Polygon Triangulation : 70.38843615231784
```