Efficiency

Program dosyayı okuduktan sonra tüm kesişen noktaları bulup o noktalardaki maksimum ve minimum yükseklikleri hesaplıyor. Sonra da bunları standart output'a yazdırıyor ve kesişen noktalardan oluşan grafiği çiziyor.

Harcanan toplam zaman aşağıdaki gibi hesaplanınca O(n^2) olarak hesaplanır.

```
26              // While scan hasNext
27              while(scan.hasNext()){
28                  width.add(scan.nextInt());
29                  height.add(scan.nextInt());
30                  position.add(scan.nextInt());
31              }
32              fr.close(); // Closing fileReader object
```

	Cost	Repetation	Total
while	1	n+1	n+1

Total Time Required

O(n+1) = O(N)

```
34          int cont1,cont2; // control variables
35          for (int i = 0; i < position.size(); i++) {   // Compare all rectangle
36              cont1=0; cont2=0;
37              for (int j = 0; j < position.size(); j++) {
38                  // If any rectangle edge contains the other rectangle's left e
39                  if(position.get(j) < position.get(i) && (position.get(j)+width
40                      cont1=1;
41                      break;
42                  }
43                  // If any rectangle edge contains the other rectangle's left e
44                  if( position.get(j) < (position.get(i)+width.get(i)) && (posit
45                      cont2=1;
46                      break;
47                  }
48              }
49              //If cont1==0 and crtcl_points dont contains position then add thi
50              if(cont1==0)
51                  if(!crtcl_points.contains(position.get(i)))
52                      crtcl_points.add(position.get(i));
53              //If cont2==0 and crtcl_points dont contains position then add thi
54              if(cont2==0)
55                  if(!crtcl_points.contains(position.get(i)+width.get(i)))
56                      crtcl_points.add(position.get(i)+width.get(i));
57          }
58          Collections.sort(crtcl_points); // Sorting Critical points array
```

	Cost	Repetation	Total
for	1	n	n
for	1	n*n	n^2
1. if	7	n*n	7 n^2
2.if	9	n*n	9 n^2
3.if	2	n	2n
4.if	2	n	2n

Total Time Required

O(17n^2+5n) = O(N^2)

```
63          for(Integer k : crtcl_points){                                  // For each c
64              max=0; right_max=0 ; left_max=0;
65              for (int j = 0; j < position.size(); j++) {
66                  if(position.get(j) <= k && (position.get(j)+width.get(j)) >= k ){
67                      if( height.get(j) > max)                            // Find maxim
68                          max = height.get(j);
69                  }
70                  if(position.get(j) <= k+1 && (position.get(j)+width.get(j)) >= k+1 ){
71                      if( height.get(j) > right_max){
72                          right_max = height.get(j);
73                          right_index = j;
74                      }
75                  }
76                  if(position.get(j) <= k-1 && (position.get(j)+width.get(j)) >= k-1 ){
77                      if( height.get(j) > left_max){
78                          left_max = height.get(j);
79                          left_index = j;
80                      }
81                  }
82              }
83              max_pts.add(max);           // Add maximum height of critical points to m
84              if(Objects.equals(max, left_max)){
85                  if( (position.get(right_index))<= k)
86                      min_pts.add(right_max);           // Add minimum height of critic
87                  else
88                      min_pts.add(0);
89              }
90              else if(Objects.equals(max, right_max)){
91                  if( position.get(left_index)+width.get(left_index)>=k )
92                      min_pts.add(left_max);           // Add minimum height of critica
93                  else
94                      min_pts.add(0);
95              }
96              else
97                  min_pts.add(0);
98          }
```

	Cost	Repetation	Total
1.for	1	2n (maks)	2n
2.for	1	2n*n	2n^2
1.if	7	2n*n	14 n^2
2.if	10	2n*n	20 n^2
3.if	10	2n*n	20 n^2
if-else	4(maks)	2n	8n

Total Time Required

O(56n^2+10n) = O(N^2)

```
107             for (int i = 1; i < crtcl_points.size(); i++) {                    // Print corners to
108                 switch(up){                                                    // add ordered corn
109                     case 1:
110                         if( Objects.equals(max_pts.get(i), max_pts.get(i-1)) ){
111                             System.out.printf(", (%d, %d)",  crtcl_points.get(i), max_pts.get(i));
112                             System.out.printf(", (%d, %d)",  crtcl_points.get(i), min_pts.get(i));
113                             que_pts.add(max_pts.get(i));
114                             que_pts.add(min_pts.get(i));
115                             up=2;
116                         }
117                         else if ( Objects.equals(min_pts.get(i), max_pts.get(i-1)) ){
118                             System.out.printf(", (%d, %d)",  crtcl_points.get(i), min_pts.get(i));
119                             System.out.printf(", (%d, %d)",  crtcl_points.get(i), max_pts.get(i));
120                             que_pts.add(min_pts.get(i));
121                             que_pts.add(max_pts.get(i));
122                             up = 1;
123                         }
124                         break;
125                     case 2:
126                         if( Objects.equals(max_pts.get(i), min_pts.get(i-1)) ){
127                             System.out.printf(", (%d, %d)",  crtcl_points.get(i), max_pts.get(i));
128                             System.out.printf(", (%d, %d)",  crtcl_points.get(i), min_pts.get(i));
129                             que_pts.add(max_pts.get(i));
130                             que_pts.add(min_pts.get(i));
131                             up=2;
132                         }
133                         else if ( Objects.equals(min_pts.get(i), min_pts.get(i-1)) ){
134                             System.out.printf(", (%d, %d)",  crtcl_points.get(i), min_pts.get(i));
135                             System.out.printf(", (%d, %d)",  crtcl_points.get(i), max_pts.get(i));
136                             que_pts.add(min_pts.get(i));
137                             que_pts.add(max_pts.get(i));
138                             up = 1;
139                         }
140                         break;
141                 }
142             }
143             new Graphic_Window( crtcl_points, max_pts, min_pts, que_pts); // Call Grapic_Window.java fo
144         }
```

	Cost	Repetation	Total
for	1	2n (maks)	2n
switch	1	2n	2n
if-else	2	2n	2n

Total Time Required

O(6n) = O(N)

For Graphic_window.java

```
150
151     public static void BilesenleriEkle(Container pencere, ArrayList<Integer> x, ArrayList<Integer> y, ArrayL
152     {
153             for (int i = 0; i < x.size(); i++) {
154                 JPanel Panel = new JPanel();
155                 pencere.add(Panel);
156                 Panel.setBackground(Color.RED);
157                 Panel.setBounds(x.get(i)*20, 400-20*y.get(i), 5, 20*(y.get(i)-z.get(i)) );
158             }
159             int high=0, k=1;
160             for (int i = 0; i < x.size()-1; i++) {
161                 JPanel Panel = new JPanel();
162                 pencere.add(Panel);
163                 Panel.setBackground(Color.RED);
164                 high +=(que_pts.get(k)-que_pts.get(k-1));
165                 Panel.setBounds(x.get(i)*20, 400-20*high, 20*(x.get(i+1)-x.get(i))+5, 5 );
166                 k+=2;
167             }
168     }
```

	Cost	Repetation	Total
for	1	2n (maks)	2n
	5	2n	10n
for	1	2n	2n
	8	2n	16n

Total Time Required

O(30n) = O(N)

Time Complexity

Tüm programdaki sabitleri hesaba katmayıp döngülerin ve içindeki işlemlerin maliyetlerini topladığımızda

M = 73n^2 + 53n buluruz. (Efficiency)

Karmaşıklık sınıflarından büyük-o (big-oh) sınıfına göre analiz yapıldığında hatırlarsak (en kötü durum analizi yapıldığı için, asimptotik üst sınır (aysmptotic upper bound) alınır)

M = O(n^2) (n data.txt den okunan dikdörtgen sayısıdır.)

Olarak maliyeti bulunmuş olunur.