Sprawozdanie do projektu 5

Alexander Krasovskiy
Aleksander Kaluta

Wstęp

W niniejszym sprawozdaniu opisane zostało rozwiązanie projektu 5, które dotyczyło wyznaczenia punktów przecięcia w zbiorze odcinków. Rozwiązanie projektu opisywane w tym sprawozdaniu spełnia wszystkie wymagania podstawowe i wymagania dodatkowe d) oraz e).

Organizacja projektu

Kod źródłowy zawierający implementację wyznaczenia punktów przecięcia w zbiorze odcinków dostępny jest w udostępnionym pliku project5.py. Pliki pomocnicze test.py oraz gui.py tworzą kolejno pliki testowe oraz interfejs graficzny i są wykorzystane w pliku project5.py W folderze test_data zostały umieszczone pliki testowe użyte w implementacji zadania.

Rozwiązanie

W pliku test.py znajduje się kod generujący odcinki i wpisujący go do pliku.
Funkcja random_points generuje losowy punkt, a funkcja generate_points zapisuje ją do listy.

```
36      @staticmethod
37      def random_points(count=1,color=Color.RED,dist=None):
38          t=TestCases()
39          pts=t.generate_normal(maxpoints=count,dist=dist)
40          for p in pts:
41              p.color=color
42          return pts
```

```
44      @staticmethod
45      def generate_points(kw:dict,dist=None):
46          rmp=Tests.random_points
47          points=[]
48          for color in kw:
49              count=kw[color]
50              points.append(rmp(count,color,dist))
51          return points
```

Funkcja points_to_random_segments generuje losowy odcinki, a funkcja generate_segments zapisuje ją do listy.

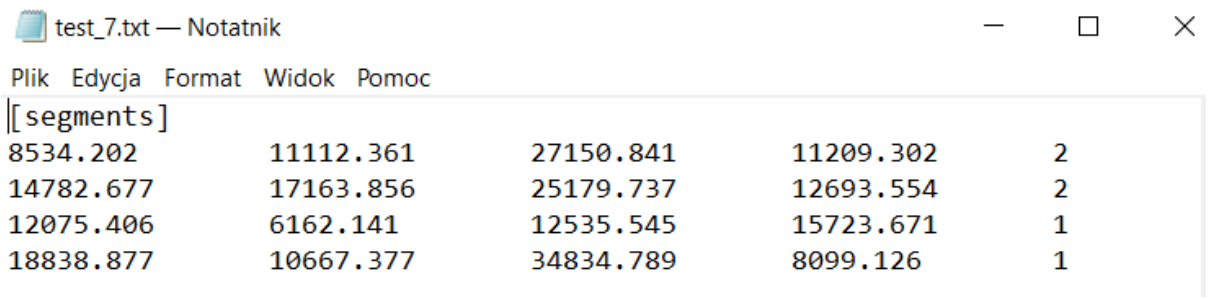
```python
    @staticmethod
    def points_to_random_segments(pts: set):
        segments = []
        it = pts.__iter__()
        # todo remove hardcode
        count_mock = 1e4
        while True:
            if len(segments) > count_mock:
                return segments
            try:
                p1 = it.__next__()
                p2 = it.__next__()
                assert p1 != p2 and p1.color == p2.color
                segments.append(Segment(p1, p2, p1.color))
            except StopIteration:
                return segments

    @staticmethod
    def generate_segments(kw:dict,dist=None):
        #each segment contains 2 points
        kw=dict((k,kw[k]*2) for k in kw)
        #segment points: red, blue
        pts=Tests.generate_points(kw,dist)
        sg=Tests.points_to_random_segments
        return [sg(x) for x in pts]
```

Funkcja write_test_data wpisuje dane do pliku

```python
    def write_test_data(kw,test_num=0):
        section_name= "segments"
        tw=TestOutputWriter()
        tw.add_section(section_name)
        for seg_lst in Tests.generate_segments(kw):
            for seg in seg_lst:
                tw.add_section_value(section_name,seg)
        fname=Tests.file_name_from_test_num(test_num)
        tw.print_to_file(fname)
```

Przykładowe użycie kodu i wynik:
Zapisanie do pliku test_7.txt dwóch odcinków czerwonych i dwóch odcinków niebieskich.

```
141  if __name__ == '__main__':
142      Tests.write_test_data({
143          Color.RED:2,
144          Color.BLUE:2
145      },7)
```



Natomiast funkcja write_test_results wpisuje do pliku wczytane z listy punkty wspólne odcinków, liczbę odcinków każdego koloru oraz liczbę punktów wspólnych każdego koloru.

```
101      def write_test_results(segments,intersection_pts,test_num):
102          postf="_intersections.txt"
103          fname=Tests.file_name_from_test_num(test_num)+postf
104          t=TestOutputWriter()
105          sec="intersections"
106          t.add_section(sec)
107          for x in intersection_pts:
108              t.add_section_value(sec,x)
109
110          points_by_color=dict()
111          segments_by_color=dict()
112          for x in intersection_pts:
113              p=x.point
114              if p.color in points_by_color:
115                  points_by_color[p.color]+=1
116              else:
117                  points_by_color[p.color]=1
118          for x in segments:
119              if x.color in segments_by_color:
120                  segments_by_color[x.color]+=1
121              else:
122                  segments_by_color[x.color]=1
```

```
123         info="info"
124         t.add_section(info)
125         t.add_section_value(info,"segments count by color: ")
126         format="%s: %d"
127         for c in segments_by_color:
128             color_name=Tests.colour_name(Color.to_pygame(c))
129             val=format%(color_name,segments_by_color[c])
130             t.add_section_value(info,val)
131 
132         t.add_section_value(info,"\nintersection points count by color: ")
133         for c in points_by_color:
134             color_name = Tests.colour_name(Color.to_pygame(c))
135             val = format%(color_name, points_by_color[c])
136             t.add_section_value(info, val)
137 
138         t.print_to_file(fname)
```

W pliku gui.py znajduje się GUI rysujące wczytane odcinki i punkty wspólne. Funkcja draw_point rysuje koło w punkcie. funkcja draw_line rysuje linie między dwoma punktami. Funkcja _draw rysuje odcinki i punkty wspólne. Natomiast funkcja draw pokazuje je na ekranie.

```
81      def draw_point(self,surface:pygame.Surface,point,color="green"):
82          radius=5
83          scale=self.scale
84          offset=self.obj_offset
85          pygame.draw.circle(surface,color,(scale*(point.x+offset[0]),scale*(point.y+offset[1])),radius)
86 
87      def draw_line(self,surface: pygame.Surface, p1,p2,color="blue"):
88          scale=self.scale
89          offset=self.obj_offset
90          pygame.draw.line(surface, color, (scale*(p1.x+offset[0]), scale*(p1.y+offset[1])),
91                                   (scale*(p2.x+offset[0]), scale*(p2.y+offset[1])),width=2)
```

```
75      def _draw(self,surface):
76          for s in self.segments:
77              self.draw_line(surface,s.A,s.B,color=Color.to_pygame(s.color))
78          for p in self.intersection_points:
79              self.draw_point(surface,p,color=Color.to_pygame(p.color))
```

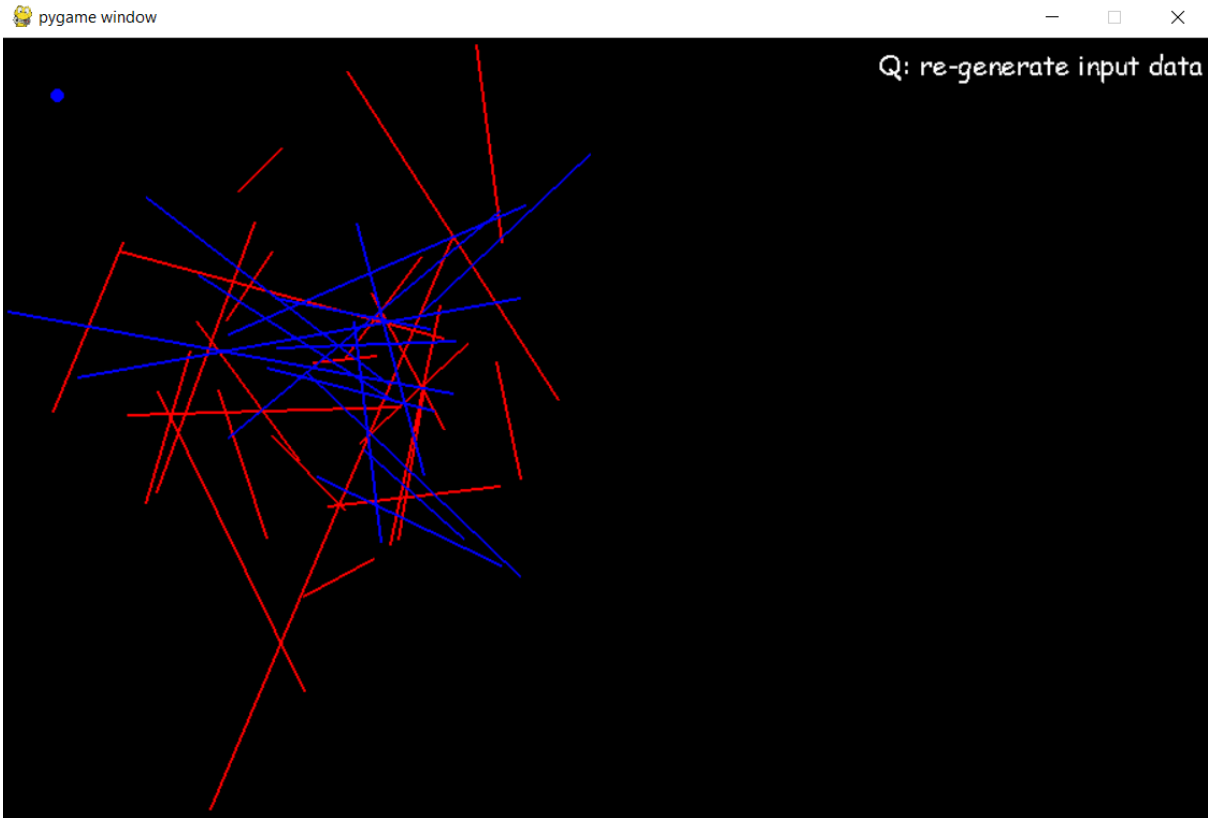
```
66      def draw(self):
67          self._lock.acquire()
68          if self._cached_surface is None:
69              self._cached_surface=pygame.Surface(self.WINDOW_SIZE)
70              self._draw(self._cached_surface)
71          self.screen.blit(self._cached_surface,self.offset)
72          self.screen.blit(self.text_surf,(self.WINDOW_SIZE[0]-self.font.size(self.msg)[0]-self.offset[0],self.offset[1]))
73          self._lock.release()
```

Funkcja mainloop uruchamia cały program.

```
124     def mainloop(self):
125         self.setup()
126         clock = pygame.time.Clock()
127         while self.draw_self:
128             clock.tick(60)
129             for event in pygame.event.get():
130                 if event.type == pygame.QUIT:
131                     self.exit()
132             self.screen.fill((0x00, 0x00, 0x00))
133             self.draw()
134             pygame.display.flip()
```

Przykładowe użycie kodu:
Narysowanie na ekranie 23 odcinków czerwonych i 15 odcinków niebieskich.

```
136 if __name__ == '__main__':
137     from tests import Tests
138     segs=Tests.generate_segments({
139         Color.RED:23,
140         Color.BLUE:15
141     })
142 
143     segs=flat_map(lambda a:a,segs)
144 
145     print(segs)
146     r=DisplaySegmentIntersections(segs,[
147         Point(0, 10), Point(10, 10),
148         Point(10, 0)
149     ],method=None,generator=None)
150     r.mainloop()
```



W pliku project5.py znajdują się algorytmy naiwny i bentleya-ottmana znajdujące punkty wspólne odcinków (algorytm bentleya-ottmana wykonuje prace w czasie O(nlogn)).

```
21  def seg_intersection_naive(iterable, collector, intersection_point_collector=None):
22      ipc=intersection_point_collector
23      for seg1 in iterable:
24          s = iterable.__iter__()
25          s.__next__()
26          for seg2 in s:
27              p = Intersection(seg1, seg2)
28              if not p.is_definite():
29                  continue
30              c=Color.combine(seg1.color,seg2.color)
31              p.color=c
32              if ipc is not None and not p in collector:
33                  ipc.add(Intersection_Point(seg1,seg2,p))
34              collector.add(p)
```

```python
def _bentley_ottman(segments,collector,step=None,seg_range=None,sort=None,intersection_point_collector=None):
    #O(n*log(n))
    intersections=collector
    seg_sorted=seg_sort(segments,sort)
    i=0
    status=deque()
    status_min,status_max=seg_range(seg_sorted[0])[0],seg_range(seg_sorted[len(seg_sorted)-1])[1]

    if step is None:
        step=abs(status_max-status_min)/1000
    for status_point in np.arange(status_min,status_max,step):
        if i>=len(seg_sorted):
            break
        status.clear()
        rng=seg_range(seg_sorted[i])
        while i<len(seg_sorted) and rng[0]<=status_point<=rng[1]:
            status.append(seg_sorted[i])
            i+=1
        if len(status)==0:
            continue
        seg_intersection_naive(status,intersections,intersection_point_collector)
```

```python
def bentley_ottman(segments, step=None, intersection_point_collector=None):
    ipc=intersection_point_collector
    intersections=set()
    seg_x_range = lambda seg: (seg.A.x, seg.B.x)
    seg_y_range = lambda seg: (seg.A.y, seg.B.y)
    sort_x = lambda seg: seg.A.x
    sort_y = lambda seg: seg.A.y
    _bentley_ottman(segments,intersections,step,seg_x_range,sort_x,intersection_point_collector=ipc)
    _bentley_ottman(segments,intersections,step,seg_y_range,sort_y,intersection_point_collector=ipc)
    return intersections
```

Po skomplilowaniu pliku project5.py program pokazuje wczytane z podanego pliku odcinki i pokazuje je na ekranie razem z częściami wspólnymi. Program daje też możliwość użytkownikowi wygenerowanie nowego pliku odcinków oraz zapisania do pliku z dopiskiem _intesetion wczytane z listy punkty wspólne odcinków, liczbę odcinków każdego koloru oraz liczbę punktów wspólnych każdego koloru.

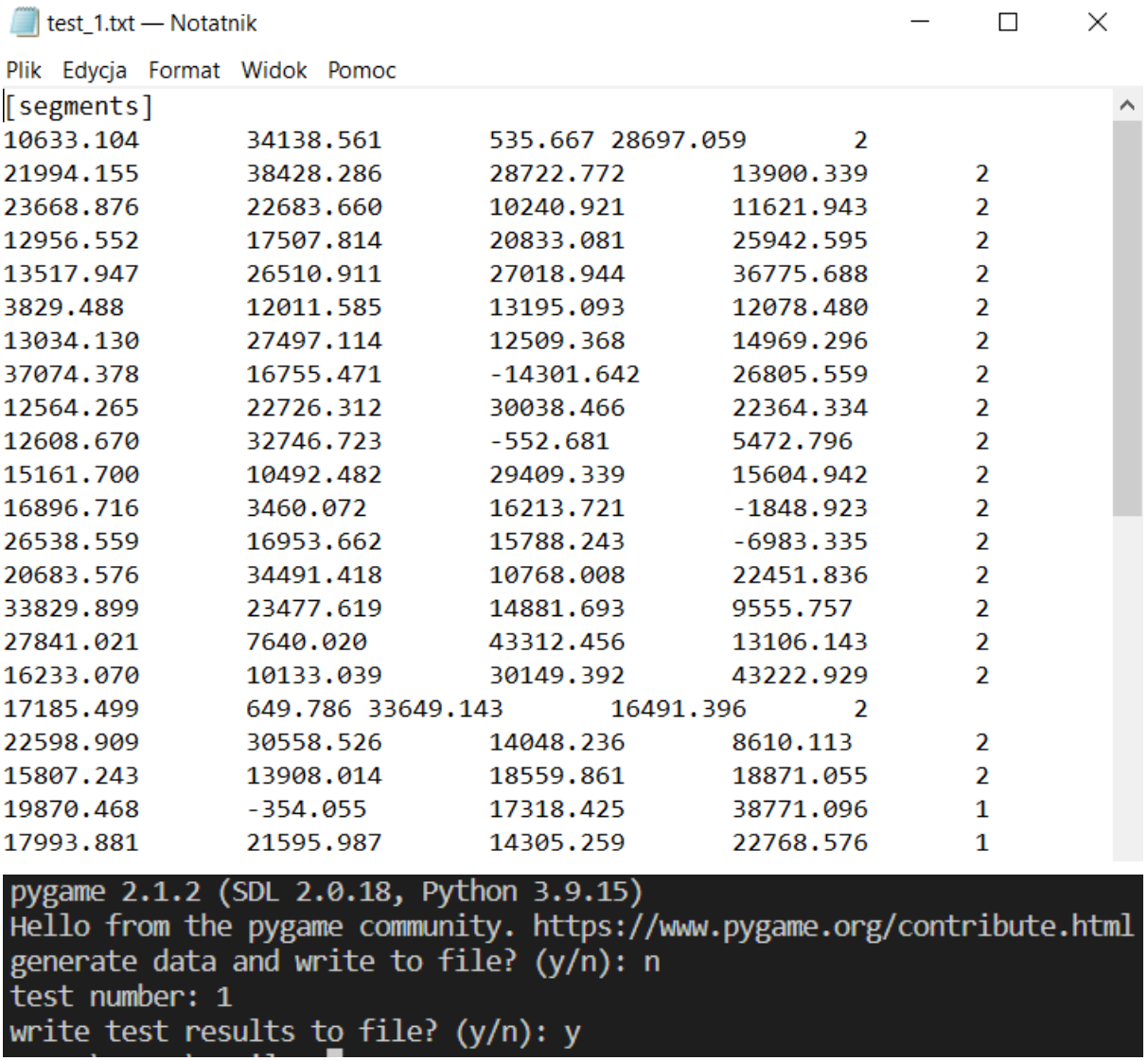
```python
    gen = lambda: flat_map(lambda _: _, Tests.generate_segments(data_def, dist=dist))
    IntersectionPts = type('IntersectionPts', (object,), {
        "write_test_res":write_test_res,
        "data":set(),
        "add":lambda self,obj: self.data.add(obj) if self.write_test_res else None
    })
    intersection_points=IntersectionPts()

    method = lambda segments: bentley_ottman(segments,intersection_point_collector=intersection_points)
```

```python
def main():
    #from Project5 import flat_map
    from gui import DisplaySegmentIntersections
    flat_map = lambda f, it: [y for ys in it for y in f(ys)]
    inp = True
    write_test_res=False
    if inp:
        generate = input("generate data and write to file? (y/n): ").lower()[:1]
        generate = len(generate) != 0 and generate[0] == 'y'
        test_num = input("test number: ")
        test_num = None if len(test_num) == 0 else int(test_num)
        write_test_res = input("write test results to file? (y/n): ").lower()[:1]
        write_test_res = len(write_test_res) != 0 and write_test_res[0] == 'y'
```

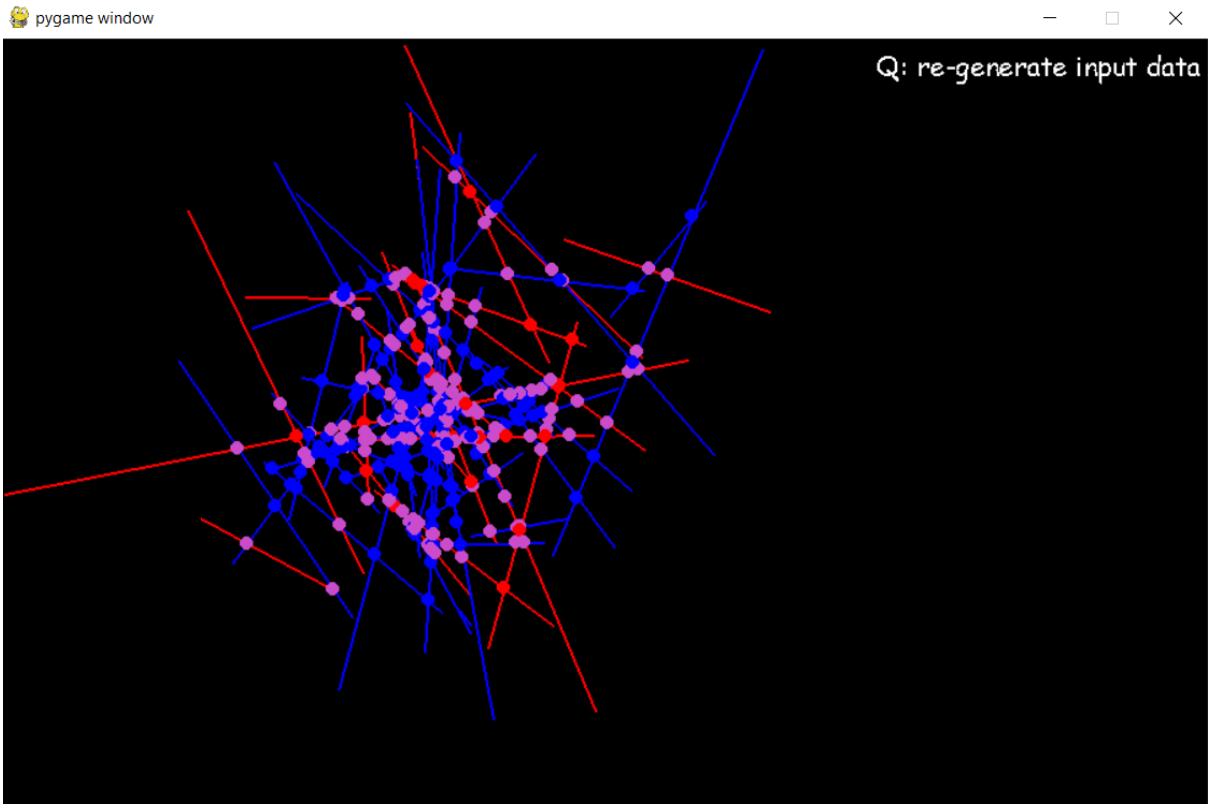
```
    else:
        generate = False
        test_num = None
    data_def = {
        Color.RED: 73,
        Color.BLUE: 44
    }
    data_def1 = {
        Color.RED: 50,
        Color.BLUE: 50,
        Color.generic_from_tuple((0, 255, 0)): 50,
    }
    data_def=data_def1
    from scipy.stats import norm, expon, uniform
    # X coordinate - exponentially distributed
    d_uni = lambda a, b: uniform.rvs(loc=0, scale=10000)
    d_exp = lambda a, b: expon.rvs(loc=b - a, scale=b)
    # Y coordinate: normally distributed
    d_norm = lambda a, b: norm.rvs(loc=b - a, scale=b)
    dist = (d_norm, d_exp)
```

```
    if generate:
        Tests.write_test_data(data_def, test_num=test_num)
    if test_num is not None:
        segs = read_test_segments(test_num)
    else:
        segs = gen()
    pts = method(segs)
    if write_test_res:
        if test_num is None:
            test_num=-1
        intersection_points=intersection_points.data
        Tests.write_test_results(segs,intersection_points,test_num)
    gui = DisplaySegmentIntersections(segs, pts, method=method, generator=gen)
    gui.mainloop()
```

Przykładowe działanie programu:




```
pygame 2.1.2 (SDL 2.0.18, Python 3.9.15)
Hello from the pygame community. https://www.pygame.org/contribute.html
generate data and write to file? (y/n): n
test number: 1
write test results to file? (y/n): y
```



test_1.txt_intersections.txt — Notatnik

Plik Edycja Format Widok Pomoc

```
[intersections]
3829.488        12011.585       13195.093       12078.480       2       4336.540        14263.513       16066.453       10206.096       1       10704.849       12060.693       (203, 76, 203
10818.735       41556.722       18033.883       14713.605       1       20683.576       34491.418       10768.008       22451.836       2       14677.876       27199.237       (203, 76, 203
16233.070       10133.039       30149.392       43222.929       2       26973.256       22398.184       7710.489        4157.982        1       17697.295       13614.638       (203, 76, 203
19870.468       -354.055        17318.425       38771.096       1       21547.518       11279.105       16644.876       29314.841       1       18342.623       23069.205       1
16233.070       10133.039       30149.392       43222.929       2       31821.353       18849.315       11407.683       25590.735       1       21352.744       22306.474       (203, 76, 203
2867.335        32081.073       25619.008       1245.166        1       38975.003       23876.303       15911.240       -2621.596       1       22709.335       5188.718        1
10818.735       41556.722       18033.883       14713.605       1       20700.231       37307.697       13836.282       24772.835       1       14838.244       26602.607       1
19870.468       -354.055        17318.425       38771.096       1       21986.648       23210.105       15931.406       19529.434       1       18472.753       21074.189       1
2867.335        32081.073       25619.008       1245.166        1       37074.378       16755.471       -14301.642      26805.559       2       10312.390       21990.605       (203, 76, 203
19870.468       -354.055        17318.425       38771.096       1       22598.909       30558.526       14048.236       8610.113        2       18534.601       20126.002       (203, 76, 203
10818.735       41556.722       18033.883       14713.605       1       21986.648       23210.105       15931.406       19529.434       1       16625.960       19951.618       1
16233.070       10133.039       30149.392       43222.929       2       33829.899       23477.619       14881.693       9555.757        2       16486.027       10734.513       2
2867.335        32081.073       25619.008       1245.166        1       33829.899       23477.619       14881.693       9555.757        2       17868.194       11750.036       (203, 76, 203
10818.735       41556.722       18033.883       14713.605       1       22598.909       30558.526       14048.236       8610.113        2       17377.456       17155.766       (203, 76, 203
16233.070       10133.039       30149.392       43222.929       2       37074.378       16755.471       -14301.642      26805.559       2       20390.760       20019.092       2
19870.468       -354.055        17318.425       38771.096       1       23668.876       22683.660       10240.921       11621.943       2       18638.092       18539.388       (203, 76, 203
2867.335        32081.073       25619.008       1245.166        1       33801.489       11428.259       18792.367       9768.022        1       19289.973       9823.065        1
10818.735       41556.722       18033.883       14713.605       1       23668.876       22683.660       10240.921       11621.943       2       17301.505       17438.330       (203, 76, 203
16559.468       26888.359       13447.897       17937.322       1       17386.475       19017.308       10947.355       23505.138       1       14518.238       21016.364       1
19870.468       -354.055        17318.425       38771.096       1       23812.909       27770.348       12354.686       9624.076        1       18576.864       19478.076       1
2867.335        32081.073       25619.008       1245.166        1       26973.256       22398.184       7710.489        4157.982        1       16987.995       12942.992       1
```

test_1.txt_intersections.txt — Notatnik

Plik Edycja Format Widok Pomoc

```
16233.070       10133.039       30149.392       43222.929       2       2:
19870.468       -354.055        17318.425       38771.096       1       2(
26538.559       16953.662       15788.243       -6983.335       2       3:
3829.488        12011.585       13195.093       12078.480       2       5(
13858.484       28205.511       5751.546        19247.833       1       3:
16233.070       10133.039       30149.392       43222.929       2       2:

[info]
segments count by color:
red: 20
blue: 40

intersection points count by color:
mediumorchid: 188
blue: 186
red: 31
```