

# 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.

```

53     @staticmethod
54     def points_to_random_segments(pts: set):
55         segments = []
56         it = pts.__iter__()
57         # todo remove hardcoded
58         count_mock = 1e4
59         while True:
60             if len(segments) > count_mock:
61                 return segments
62             try:
63                 p1 = it.__next__()
64                 p2 = it.__next__()
65                 assert p1 != p2 and p1.color == p2.color
66                 segments.append(Segment(p1, p2, p1.color))
67             except StopIteration:
68                 return segments
69
70     @staticmethod
71     def generate_segments(kw: dict, dist=None):
72         #each segment contains 2 points
73         kw=dict((k,kw[k]*2) for k in kw)
74         #segment points: red, blue
75         pts=Tests.generate_points(kw,dist)
76         sg=Tests.points_to_random_segments
77         return [sg(x) for x in pts]

```

Funkcja `write_test_data` wpisuje dane do pliku

```

80     def write_test_data(kw, test_num=0):
81         section_name= "segments"
82         tw=TestOutputWriter()
83         tw.add_section(section_name)
84         for seg_lst in Tests.generate_segments(kw):
85             for seg in seg_lst:
86                 tw.add_section_value(section_name, seg)
87         fname=Tests.file_name_from_test_num(test_num)
88         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)

```

test\_7.txt — Notatnik

Plik Edycja Format Widok Pomoc

[[segments]				
8534.202	11112.361	27150.841	11209.302	2
14782.677	17163.856	25179.737	12693.554	2
12075.406	6162.141	12535.545	15723.671	1
18838.877	10667.377	34834.789	8099.126	1

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()

```

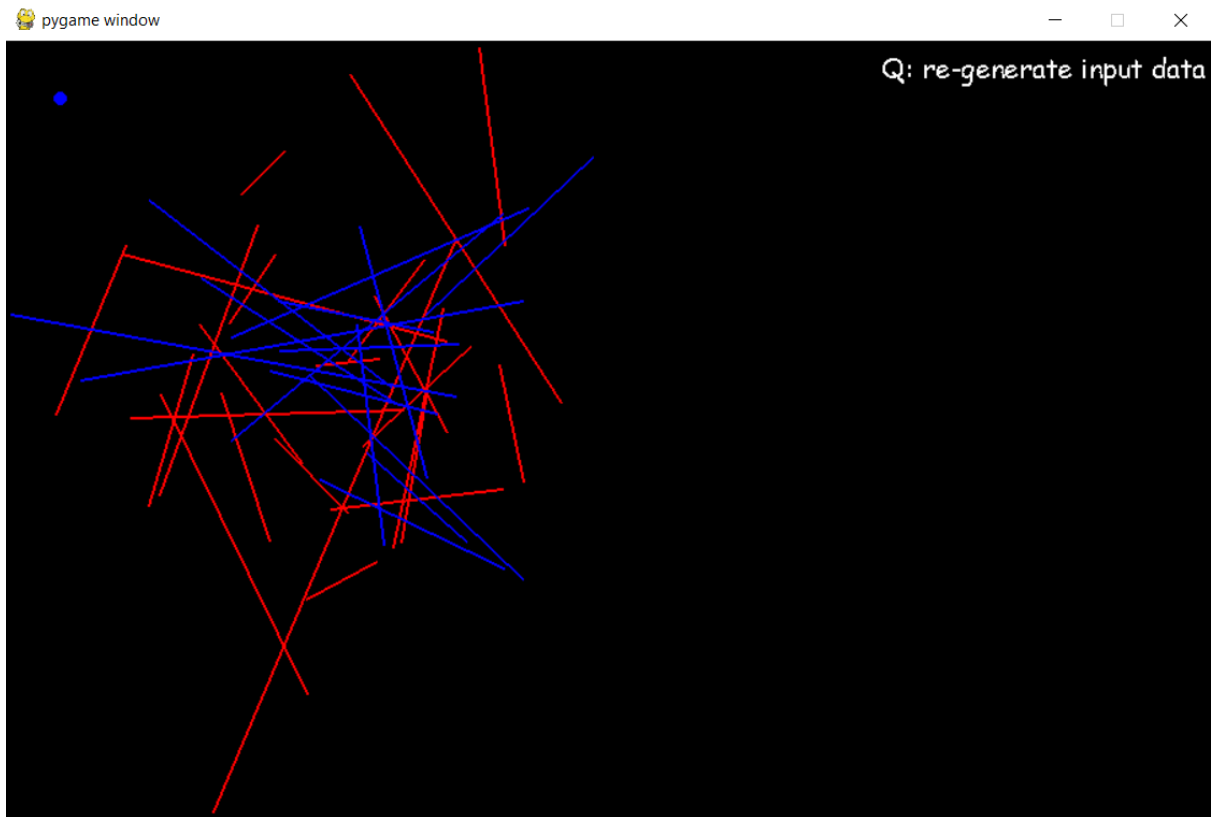
Przykładowe użycie kodu:

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 pracę w czasie  $O(n \log n)$ ).

```
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)
```

```

36 def _bentley_ottman(segments, collector, step=None, seg_range=None, sort=None, intersection_point_collector=None):
37     #O(n*log(n))
38     intersections=collector
39     seg_sorted=seg_sort(segments, sort)
40     i=0
41     status=deque()
42     status_min, status_max=seg_range(seg_sorted[0])[0], seg_range(seg_sorted[len(seg_sorted)-1])[1]
43
44     if step is None:
45         step=abs(status_max-status_min)/1000
46     for status_point in np.arange(status_min, status_max, step):
47         if i>len(seg_sorted):
48             break
49         status.clear()
50         rng=seg_range(seg_sorted[i])
51         while i<len(seg_sorted) and rng[0]<=status_point<=rng[1]:
52             status.append(seg_sorted[i])
53             i+=1
54         if len(status)==0:
55             continue
56         seg_intersection_naive(status, intersections, intersection_point_collector)
57
58 def bentley_ottman(segments, step=None, intersection_point_collector=None):
59     ipc=intersection_point_collector
60     intersections=set()
61     seg_x_range = lambda seg: (seg.A.x, seg.B.x)
62     seg_y_range = lambda seg: (seg.A.y, seg.B.y)
63     sort_x = lambda seg: seg.A.x
64     sort_y = lambda seg: seg.A.y
65     _bentley_ottman(segments, intersections, step, seg_x_range, sort_x, intersection_point_collector=ipc)
66     _bentley_ottman(segments, intersections, step, seg_y_range, sort_y, intersection_point_collector=ipc)
67     return intersections

```

Po skompiłowaniu 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.

```

103 gen = lambda: flat_map(lambda _: _, Tests.generate_segments(data_def, dist=dist))
104 IntersectionPts = type('IntersectionPts', (object,), {
105     "write_test_res": write_test_res,
106     "data": set(),
107     "add": lambda self, obj: self.data.add(obj) if self.write_test_res else None
108 })
109 intersection_points=IntersectionPts()
110
111 method = lambda segments: bentley_ottman(segments, intersection_point_collector=intersection_points)
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

82  else:
83      generate = False
84      test_num = None
85  data_def = {
86      Color.RED: 73,
87      Color.BLUE: 44
88  }
89  data_def1 = {
90      Color.RED: 50,
91      Color.BLUE: 50,
92      Color.generic_from_tuple((0, 255, 0)): 50,
93  }
94  data_def=data_def1
95  from scipy.stats import norm, expon, uniform
96  # X coordinate - exponentially distributed
97  d_uni = lambda a, b: uniform.rvs(loc=0, scale=10000)
98  d_exp = lambda a, b: expon.rvs(loc=b - a, scale=b)
99  # Y coordinate: normally distributed
100 d_norm = lambda a, b: norm.rvs(loc=b - a, scale=b)
101 dist = (d_norm, d_exp)

113 if generate:
114     Tests.write_test_data(data_def, test_num=test_num)
115 if test_num is not None:
116     segs = read_test_segments(test_num)
117 else:
118     segs = gen()
119 pts = method(segs)
120 if write_test_res:
121     if test_num is None:
122         test_num=-1
123     intersection_points=intersection_points.data
124     Tests.write_test_results(segs,intersection_points,test_num)
125 gui = DisplaySegmentIntersections(segs, pts, method=method, generator=gen)
126 gui.mainloop()

```

Przykładowe działanie programu:



[segments]

10633.104	34138.561	535.667	28697.059	2
21994.155	38428.286	28722.772	13900.339	2
23668.876	22683.660	10240.921	11621.943	2
12956.552	17507.814	20833.081	25942.595	2
13517.947	26510.911	27018.944	36775.688	2
3829.488	12011.585	13195.093	12078.480	2
13034.130	27497.114	12509.368	14969.296	2
37074.378	16755.471	-14301.642	26805.559	2
12564.265	22726.312	30038.466	22364.334	2
12608.670	32746.723	-552.681	5472.796	2
15161.700	10492.482	29409.339	15604.942	2
16896.716	3460.072	16213.721	-1848.923	2
26538.559	16953.662	15788.243	-6983.335	2
20683.576	34491.418	10768.008	22451.836	2
33829.899	23477.619	14881.693	9555.757	2
27841.021	7640.020	43312.456	13106.143	2
16233.070	10133.039	30149.392	43222.929	2
17185.499	649.786	33649.143	16491.396	2
22598.909	30558.526	14048.236	8610.113	2
15807.243	13908.014	18559.861	18871.055	2
19870.468	-354.055	17318.425	38771.096	1
17993.881	21595.987	14305.259	22768.576	1

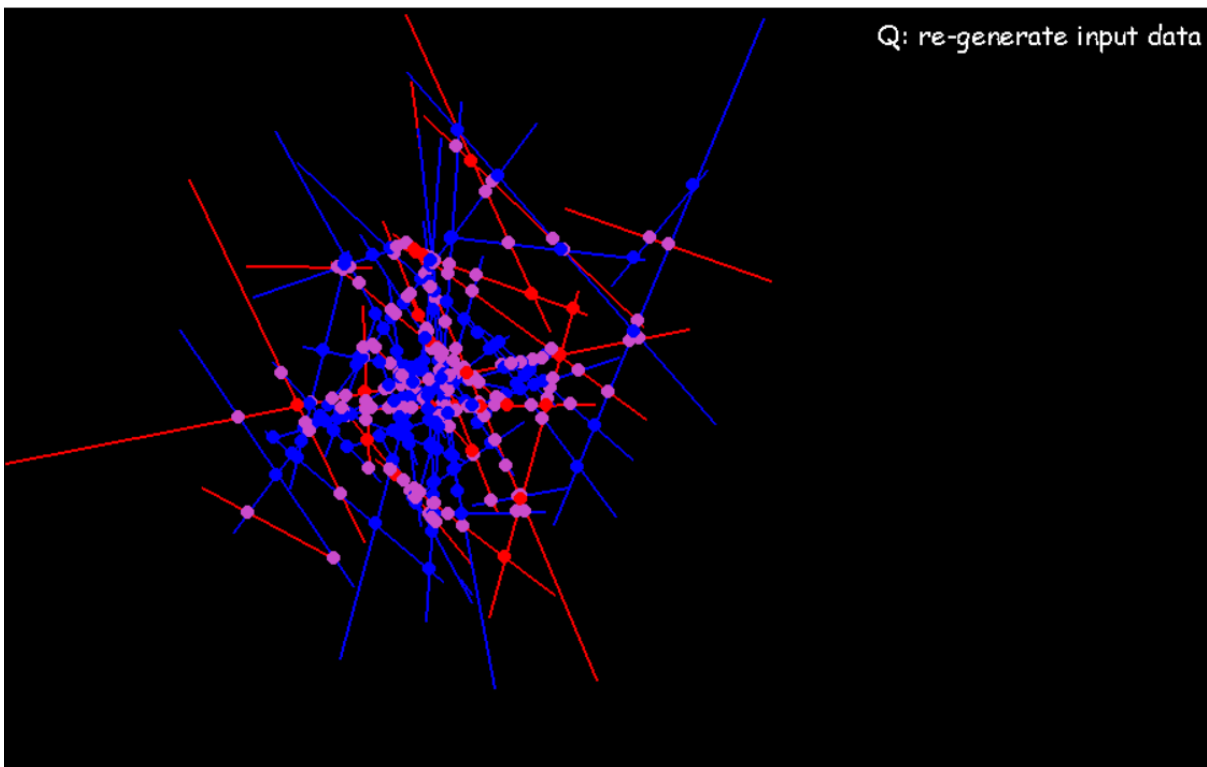
pygame 2.1.2 (SDL 2.0.18, Python 3.9.15)

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



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	14638.244	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	17808.194	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

Plik Edycja Format Widok Pomoc

16233.070	10133.039	30149.392	43222.929	2	23
19870.468	-354.055	17318.425	38771.096	1	20
26538.559	16953.662	15788.243	-6983.335	2	33
3829.488	12011.585	13195.093	12078.480	2	59
13858.484	28205.511	5751.546	19247.833	1	37
16233.070	10133.039	30149.392	43222.929	2	23

[info]

segments count by color:

red: 20

blue: 40

intersection points count by color:

mediumorchid: 188

blue: 186

red: 31