ΜΙΧΑΗΛ-ΧΡΗΣΤΟΣ ΝΑΒΑΡΟ-ΑΜΑΡΓΙΑΝΟΣ
ΑΜ: 1115202000151

ΥΠΟΛΟΓΙΣΤΙΚΗ ΓΕΩΜΕΤΡΙΑ

Υπολογιστική Εργασία Σεπτεμβρίου

Υλοποίηση Α: Κυρτό περίβλημα

1.
    1) Αυξητικός αλγόριθμος : “A > incr.py”
    2) Αλγόριθμος περιτυλίγματος : “A > g_wrp.py”
    3) Αλγόριθμος Διαίρει και Βασίλευε : “A > div_con.py”
    4) Αλγόριθμος QuickHull : “A > quick_hull.py”

2. Για την εύρεση κυρτού περιβλήματος σε 3 διαστάσεις υλοποίησα και προσάρμοσα τον αλγόριθμο QuickHull : “A > quick_hull_3d.py”

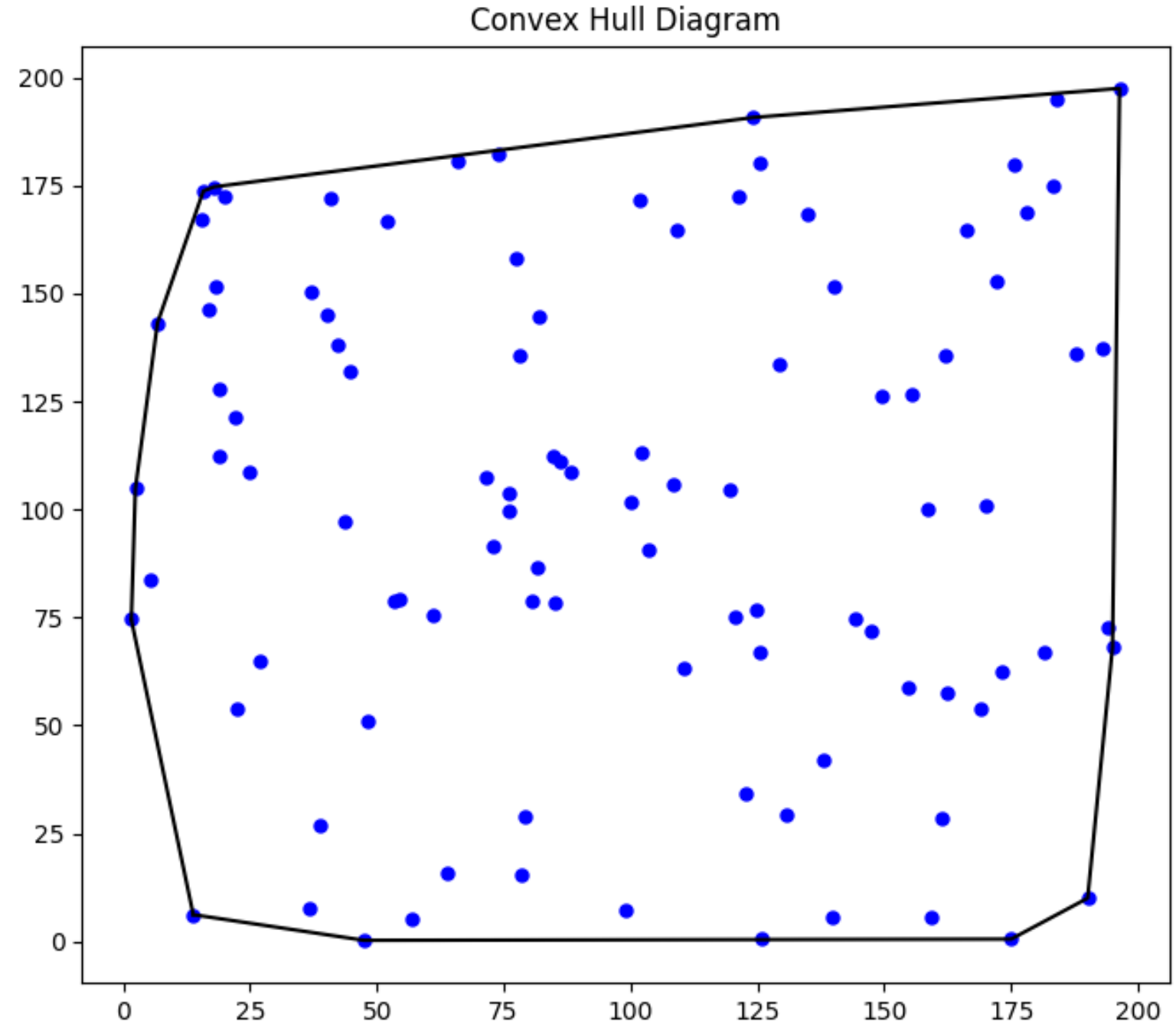
Εφαρμογή Α:

α)
    α.1) Αυξητικός αλγόριθμος : “A > app1.py”
    α.2) Αλγόριθμος περιτυλίγματος : “A > app2.py”
    α.3) Αλγόριθμος Διαίρει και Βασίλευε : “A > app3.py”
    α.4) Αλγόριθμος QuickHull : “A > app4.py”

Στο πρόγραμμα “A > app_all.py” εκτελούνται όλοι οι αλγόριθμοι μαζί με την ίδια είσοδο. Όπως αναμενόταν το Convex Hull είναι το ίδιο, ωστόσο παρατηρούμε πως τα σημεία στην λίστα του Convex Hull είναι τοποθετημένα με διαφορετική σειρά σε κάθε περίπτωση, αφού χρησιμοποιείται διαφορετική μέθοδος υπολογισμού.

Ενδεικτικά αποτελέσματα:

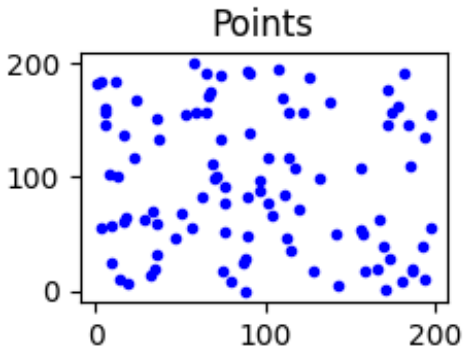
```
Points of convex hull with Incremental:
[[1.27712983e-01 6.88776597e+01]
 [4.32948031e+00 1.29822045e+02]
 [3.99680084e+01 1.88537996e+02]
 [7.32831136e+01 1.96729154e+02]
 [8.38299499e+01 1.99202743e+02]
 [1.97040240e+02 1.92649412e+02]
 [1.97295013e+02 3.95461951e+01]
 [1.96155395e+02 2.63103172e+01]
 [1.71567124e+02 1.14900319e+00]
 [9.60245611e+01 2.93844497e-01]
 [7.69167814e+01 9.19738486e-01]
 [4.28127918e+01 2.19136088e+00]
 [8.31870796e+00 3.38596756e+01]]
Points of convex hull with Gift Wrapping:
[[1.27712983e-01 6.88776597e+01]
 [8.31870796e+00 3.38596756e+01]
 [4.28127918e+01 2.19136088e+00]
 [7.69167814e+01 9.19738486e-01]
 [9.60245611e+01 2.93844497e-01]
 [1.71567124e+02 1.14900319e+00]
 [1.96155395e+02 2.63103172e+01]
 [1.97295013e+02 3.95461951e+01]
 [1.97040240e+02 1.92649412e+02]
 [8.38299499e+01 1.99202743e+02]
 [7.32831136e+01 1.96729154e+02]
 [3.99680084e+01 1.88537996e+02]
 [4.32948031e+00 1.29822045e+02]]
Points of convex hull Divide and conquier:
[[8.38299499e+01 1.99202743e+02]
 [7.32831136e+01 1.96729154e+02]
 [3.99680084e+01 1.88537996e+02]
 [4.32948031e+00 1.29822045e+02]
 [1.27712983e-01 6.88776597e+01]
 [8.31870796e+00 3.38596756e+01]
 [4.28127918e+01 2.19136088e+00]
 [7.69167814e+01 9.19738486e-01]
 [9.60245611e+01 2.93844497e-01]
 [1.71567124e+02 1.14900319e+00]
 [1.96155395e+02 2.63103172e+01]
 [1.97295013e+02 3.95461951e+01]
 [1.97040240e+02 1.92649412e+02]]
Points of convex hull with QuickHull:
[[  4.32948031 129.82204503]
 [ 22.01388693  35.15130341]
 [148.94239094  78.25510644]
 [149.94170563  72.82189082]
 [ 42.81279176   2.19136088]
 [ 25.97673856  94.04327463]
 [ 59.65750458   5.7746344 ]
 [180.87890987 190.82554227]
 [ 52.48567233  11.74091936]
 [163.34738819 192.54586135]
 [178.90847517 129.92978608]
 [164.85123541 185.25881353]
 [186.16013011 105.37892303]]
```

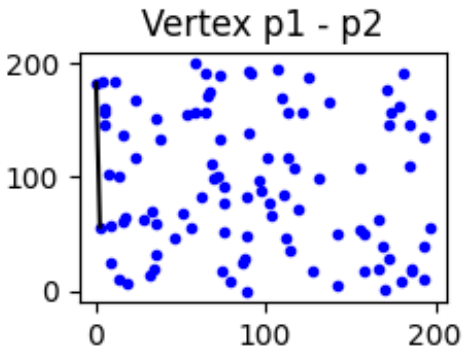


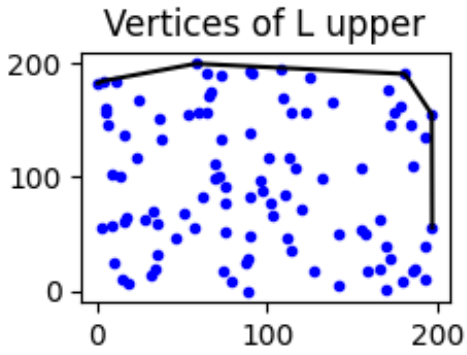
β) Η σύγκριση των αλγορίθμων ως προς τον χρόνο υλοποίησης τους γίνεται με τον πρόγραμμα “A > times.py”.

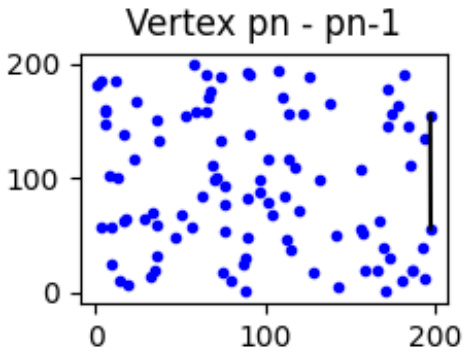
```
Terminal — -zsh — 134×53
miguelnavarro@Miguels MacbookAir a % python times.py
Points              100 points              1000 points             10000 points            100000 points
Incremental         0.0011403560638427734   0.006597995758056641    0.06505131721496582     0.696638822555542
Gift Wrapping       0.002080202102661133    0.029722213745117188    0.3847169876098633      4.790376901626587
Divide and Conquer  0.0012080669403076172   0.02026820182800293     0.14833784103393555     1.5277140140533447
QuickHull           0.0005681514739990234   0.0015521049499511719   0.013859033584594727    0.1358652114868164
miguelnavarro@Miguels MacbookAir a %
```

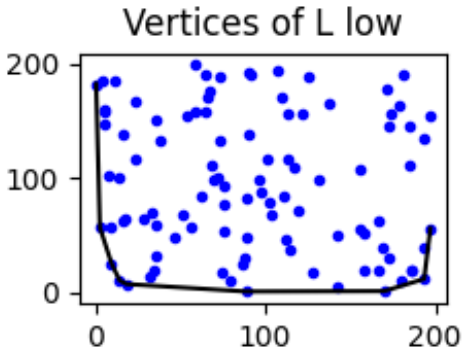
γ) Για την οπτικοποίηση των βημάτων της κατασκευή του κυρτού περιβλήματος, επιλέγουμε τον αυξητικό αλγόριθμο. Η διαδικασία αυτή υλοποιείται με το πρόγραμμα “A > app1_steps.py”

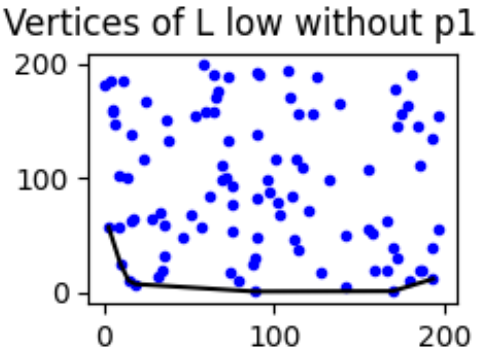


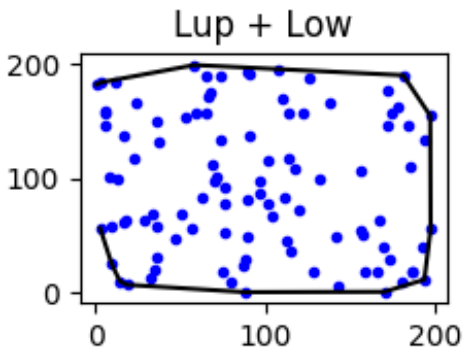


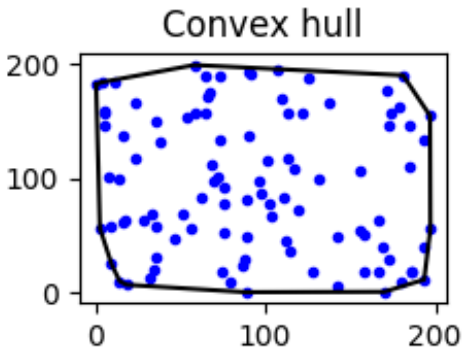






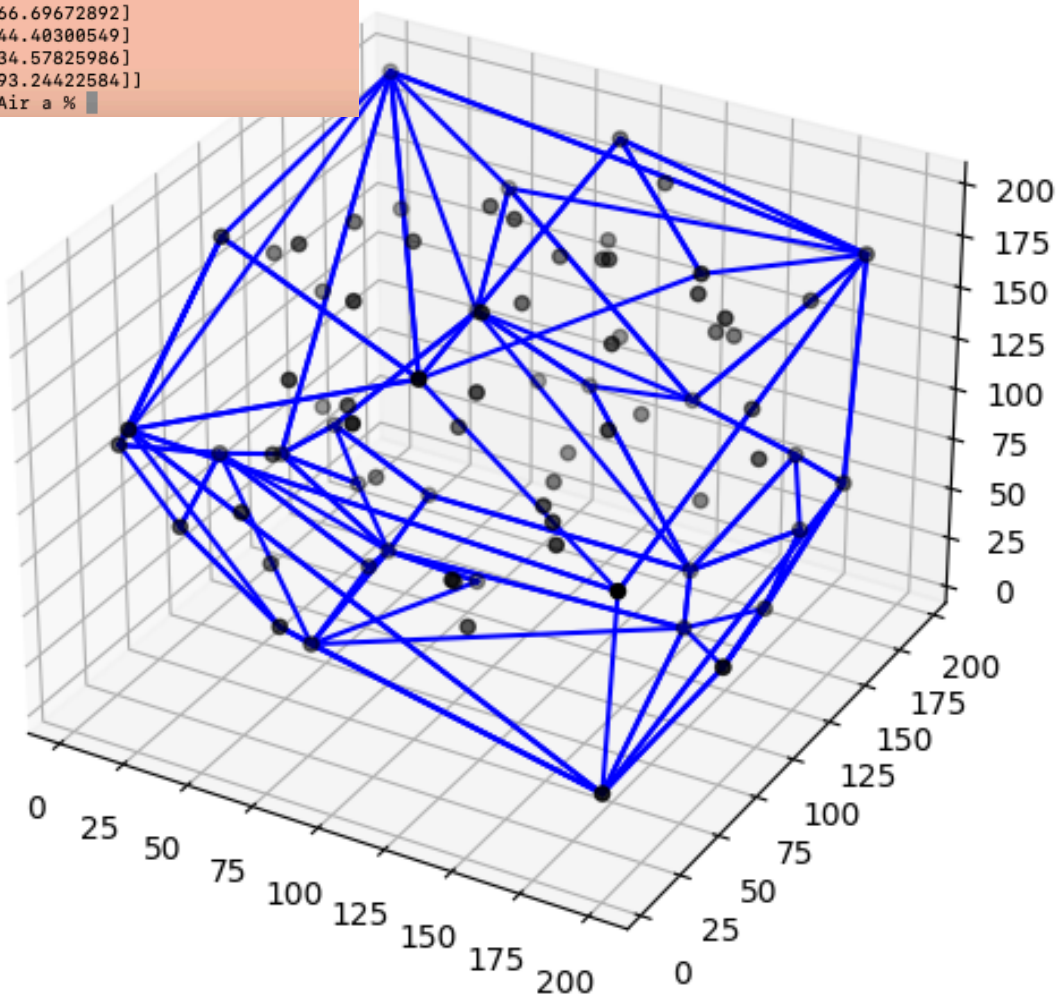






2. Για να βρω το κυρτό περίβλημα στον χώρο του R^3 χρησιμοποιώ τον αλγόριθμο QuickHull, ο οποίος έχει υλοποιηθεί στο πρόγραμμα “A > quick_hull_3d.py”. Το ερώτημα της άσκησης έχει υλοποιηθεί στο πρόγραμμα “A > app_quickhull_3d.py”. Ενδεικτικά αποτελέσματα:

```
List_points of convex hull:
[[162.29163842 125.77906328 190.68205479]
 [171.05796813 177.8903683   71.87073478]
 [ 26.84659267  25.76679187  85.61324218]
 [199.70301759   8.51152213 134.08777694]
 [152.21455726 137.05546291   5.46290399]
 [ 55.03869721  94.38821277  28.82672039]
 [194.84404507   8.73828261  35.87481135]
 [  2.67934622  27.55905678 115.22921099]
 [187.28229018 192.89946016 166.85135444]
 [ 50.64530356 192.52653041 153.27663893]
 [ 13.04740393  74.18326641 189.04060523]
 [146.06112086 151.86181905  21.81121339]
 [ 35.06241801 119.6985338   45.76675028]
 [191.26990402  92.70570487  33.42000511]
 [ 97.17923369 186.74179813 196.29344388]
 [ 91.35056064 178.06472052  77.44394467]
 [ 21.94855624 125.89962099  66.61967581]
 [  2.0036775  195.20892069 193.60749164]
 [ 72.15417918  14.30308682  62.54501414]
 [183.90811613 106.69945766  20.47643752]
 [ 53.99038215 134.45831129  37.09708722]
 [183.88860755 134.48410161  29.78439219]
 [ 45.03070605 180.65194918  97.06630735]
 [180.98475947 163.53444877  47.86401118]
 [ 85.06219613 114.63688936  18.69044051]
 [128.44364491 183.26292925  80.11391681]
 [196.29865585 164.26558721  76.47986931]
 [ 53.33858035 109.30684693  25.68882197]
 [  9.90956125  74.42794313  82.33268293]
 [ 16.7091812  100.94595441  66.69672892]
 [ 22.21161922   5.62563358 144.40300549]
 [ 67.96971905  38.81120783  34.57825986]
 [120.52276497  21.87645276 193.24422584]]
miguelnavarro@Miguels MacbookAir a %
```



3. Θεωρητικά και οι τέσσερις αλγόριθμοι (Incremental, Gift Wrapping, Divide and Conquer, QuickHull) μπορούν να υλοποιηθούν και να γενικευτούν σε περισσότερες από 3 διαστάσεις. Ωστόσο όσο μεγαλώνουν οι διαστάσεις τόσο αυξάνεται η πολυπλοκότητα των αλγόριθμων και η δυσκολία να αναπαρασταθούν γραφικά. Συνεπώς, πολλές φορές όταν έχουμε μεγάλο αριθμό διαστάσεων οι αλγόριθμοι αυτοί ενδέχεται να μην είναι πρακτικοί, καθώς έχουμε εκθετική αύξηση των πράξεων και των πιθανών συνδυασμών.

Υλοποίηση Β: Γεωμετρική αναζήτηση

1. Για την ορθογώνια γεωμετρική αναζήτηση σε ένα σύνολο σημείων στο επίπεδο υλοποίησα την αναζήτησή μέσω kd-δέντρου.

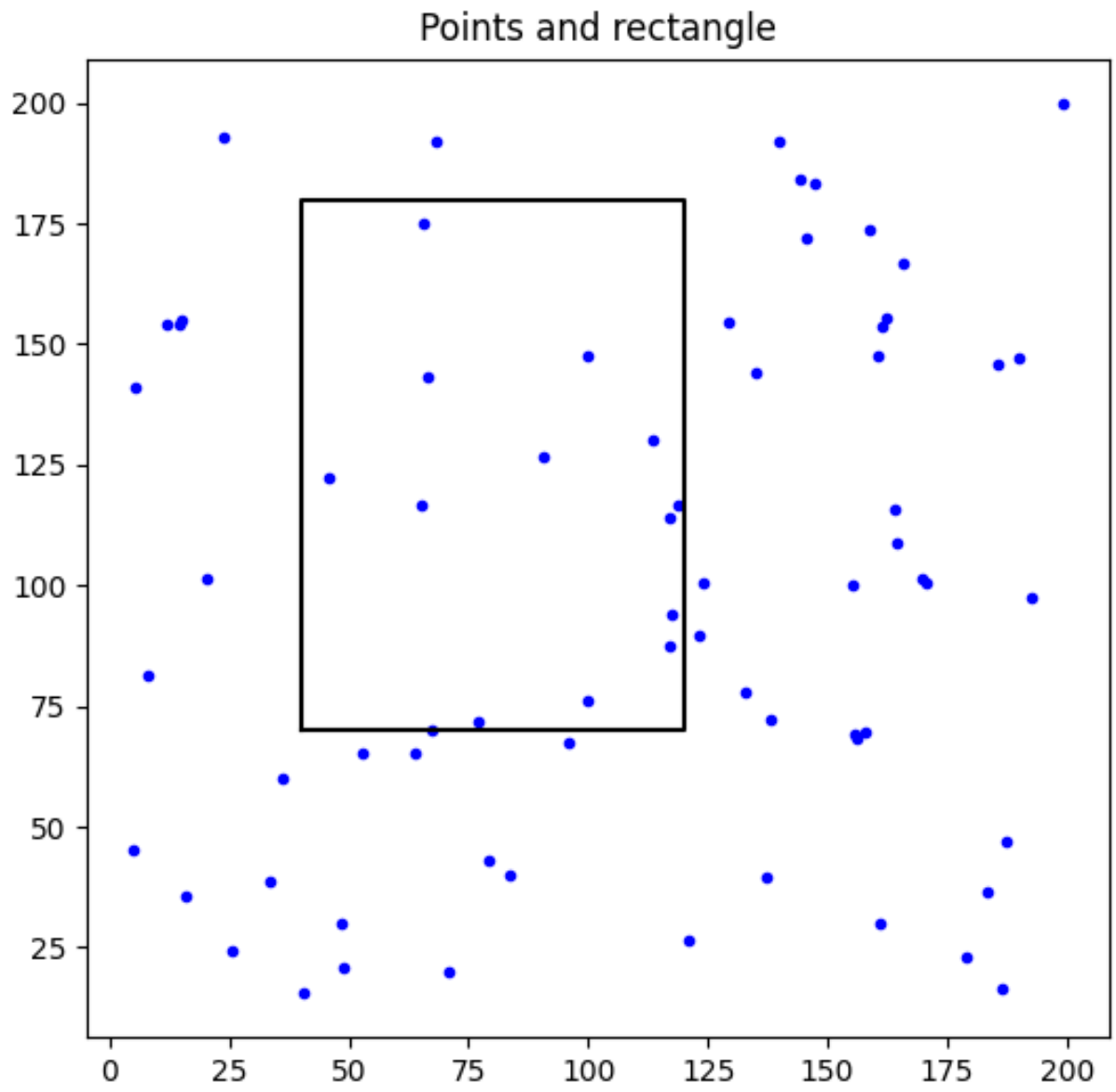
Στο αρχείο “B > search_kd.py”, υλοποιήθηκε και η κατασκεύη και η αναζήτηση σε kd-δέντρου.

Εφαρμογή Β:

Η εφαρμογή “β” βρίσκεται στο αρχείο “B > app2.py”

Ενδεικτικά αποτελέσματα:

```
miguelnavarro@Miguels MacbookAir b % python app2.py
List of points:
[[ 96.07196452  67.57184802]
 [164.71632508 108.66662551]
 [  5.473095   141.16929872]
 [165.83363786 166.65244274]
 [ 79.36363044  43.07269873]
 [ 68.03821197 192.1537074 ]
 [185.68930285 145.63380593]
 [ 15.72118231  35.84124885]
 [161.25156709 153.84792388]
 [187.32214793  47.09906542]
 [124.17947548 100.5304152 ]
 [ 20.13203185 101.48070176]
 [117.00766148 114.25091319]
 [ 36.15275716  59.80614004]
 [186.57229056  16.38172585]
 [156.00073097  68.40368905]
 [135.19400339 143.88969299]
 [ 14.75432351 154.9655943 ]
 [161.01106178  29.75913653]
 [ 40.26658417  15.46667157]
 [ 90.56640138 126.47276295]
 [132.78727071  78.06219427]
 [169.79939798 101.61569124]
 [144.45348868 184.19586778]
 [192.75213005  97.55778345]
 [170.8411577  100.50249904]
 [ 76.80307256  71.84315566]
 [ 11.903498   154.25131896]
 [139.75324806 192.20781966]
 [100.03739576  76.13347969]
 [163.87786832 115.64944976]
 [ 70.71602904  20.04134093]
 [ 63.78877119  65.03450415]
 [157.86528676  69.70834848]
 [  5.06090578  45.17043005]
 [ 83.55678787  40.19924022]
 [ 25.42807335  24.30095934]
 [ 65.18063319 116.71692491]
 [ 48.56544751  29.95968227]
 [137.41632349  39.70521686]
 [ 33.35531163  38.57687777]
 [138.00275494  72.10775094]
 [123.12797802  89.56529532]
 [  7.81921879  81.22760656]
 [120.81656523  26.62440994]
 [118.94690617 116.66654416]
 [178.83487147  22.93764535]
 [ 23.98569424 192.76902142]
 [155.54479241  69.14735354]
 [147.19383237 183.35278633]
 [199.36507927 199.96420215]
 [158.76401768 173.52338315]
 [ 99.67135213 147.59100586]
 [129.37038295 154.37426596]
 [189.92746086 146.97859331]
 [183.47644639  36.44991266]
 [113.35465668 130.00566711]
 [ 48.7741731   20.96684462]
 [162.41411137 155.31705948]
 [ 14.49599042 154.29406589]
 [155.23815787 100.0490859 ]
 [ 45.9798281  122.5283789 ]
 [117.43828869  94.15069856]
 [160.49561108 147.66316726]
 [145.52745698 172.15181516]
 [ 66.46356249 143.42455567]
 [ 67.11082084  69.83114737]
 [ 53.00542659  65.19606725]
 [116.80912951  87.67846154]
 [ 65.51912396 175.25021079]]
Number of points into the rectangle:  13
Points into the rectangle:
(76.80307255845354, 71.84315566001752)
(100.03739576491417, 76.13347968639108)
(65.18063319196068, 116.7169249090706)
(45.97982809500969, 122.52837890249606)
(65.51912396381083, 175.25021079091172)
(116.80912951265876, 87.67846154180 62)
(90.56640138267198, 126.47276295152751)
(117.43828868852839, 94.15069856179925)
(117.00766148430124, 114.25091319454526)
(66.46356249219025, 143.4245556733434)
(113.35465667675761, 130.0056671148408)
(99.6713521332036, 147.59100585742237)
(118.94690616577526, 116.66654415971742)
```



Εφαρμογή Γ:

Μερικές πραγματικές εφαρμογές όπου χρησιμοποιούνται τα kd-δέντρα είναι εφαρμογές της ρομποτικής και των αυτοματοποιημένων συστημάτων.
Στις εφαρμογές αυτές τα kd-δέντρα χρησιμοποιούνται για την εύρεση ενός μονοπατιού και την αποφυγή εμποδίων.
Τα kd-δέντρα είναι πολύ αποτελεσματικά, αφού μας επιτρέπουν να κάνουμε γρήγορες αναζητήσεις σε πολυδιάστατους χώρους.

Έστω ένα ρομπότ το οποίο χρησιμοποιεί ένα ραντάρ. Το ραντάρ αυτό χρησιμεύει για τον εντοπισμό των εμποδίων που βρίσκονται σε επικίνδυνη περιοχή για το ρομπότ. Το ραντάρ είναι τετραγωνοποιημένο και διαστάσεων 4x4, το ρομπότ θεωρούμε πως βρίσκεται στο μέσο του τετραγώνου ραντάρ.

Έστω ένα δωμάτιο διαστάσεων 15x15 το οποίο περιέχει 50 αντικείμενα στο πάτωμα. Πρέπει να υλοποιήσουμε ένα πρόγραμμα το οποίο για μια δεδομένη θέση του ρομπότ θα μας επιστρέφει όλα τα αντικείμενα του δωματίου που βρίσκονται εντός του ραντάρ του. Να γίνει και οπτικοποίηση της κατάστασης.
Το πρόγραμμα υπάρχει στο “B > robot.py”

Ενδεικτικά αποτελέσματα:

```
List of Obstacles:
[[ 6.81889761 13.42446153]
 [ 4.65248436 11.20740763]
 [ 8.96296847  0.24502555]
 [ 6.28452303  2.4049559 ]
 [12.1030081  13.29206214]
 [12.94060819  4.42764048]
 [14.75228659 14.75977903]
 [12.46033022  6.58837029]
 [ 4.7440923  13.20018887]
 [ 4.22634545 12.43304475]
 [ 9.09632256 14.1105124 ]
 [10.34875462  0.20351384]
 [ 2.33811402  3.67958915]
 [ 8.68138518 12.04270173]
 [12.33223809  4.39226285]
 [ 8.45968641  5.36808742]
 [ 7.90442353  1.49056412]
 [ 1.17039584  0.68612682]
 [ 3.28929321  6.87175012]
 [13.32872822  6.2234668 ]
 [ 3.41098658  1.86232579]
 [11.07528215 14.46614181]
 [ 1.15354683  9.64680642]
 [ 2.6985208   8.09058731]
 [ 8.92401879 14.20829618]
 [ 8.02422506 14.62944002]
 [ 0.78425408 13.91062293]
 [13.0681275  11.28577815]
 [10.64625105  4.92287407]
 [ 5.14815837 14.8161364 ]
 [ 9.84628115  2.80546105]
 [ 2.90913502  9.55004758]
 [ 5.88437666  7.64551479]
 [12.262793    4.57483398]
 [ 8.8860738   9.28138199]
 [14.64534151  5.72408051]
 [ 0.48750213  7.42526301]
 [ 9.60150569  2.49667217]
 [ 7.62616801 14.39704879]
 [ 8.99820302  4.46618468]
 [ 8.74535581  4.79035492]
 [ 8.94225697 10.7869319 ]
 [ 2.91914104  8.04687193]
 [ 8.71638637  2.02584599]
 [12.88959913 12.58526324]
 [ 0.36121355  1.03274379]
 [11.6600948   0.90277103]
 [ 1.92725311  2.1287351 ]
 [ 8.47784684 14.56394714]
 [12.20865142  1.55814932]]
Number of obstacles very near the robot:  1
Obstacles into the radar:
(8.886073795708706, 9.28138199100819)
```