Υπολογιστική Γεωμετρία με χρήση της Python

```
from pycompgeom import *
```

Χριστόδουλος Φραγκουδάκης

ΕΜΠ - Κέντρο Υπολογιστών

Άνοιξη 2014

Εισαγωγή στην Python

- Το πρόγραμμα “Hello World!” σε τρεις γλώσσες προγραμματισμού:

C++
```
#include <iostream.h>
void main() {
    cout << "Hello, world!" << endl;
}
```

Java
```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Python
```
print "Hello World!"
```

Εισαγωγή στην Python

- Multi paradigm: object orientation, structural programming, functional programming, …
- Έχει δυναμικούς τύπους και πολύ καλή διαχείριση μνήμης.
- Είναι εύκολα επεκτάσιμη (small core, large standard library, extensible interpreter).
- Η σύνταξη είναι καθαρή, αραιή και άμεση.
- Απορρίπτει το motto της Perl:

  There is more than one way to do it.

  και υϊοθετεί το:

  There sould be one *obvious* way to do it.

Εισαγωγή στην Python

- Οι μεταβλητές δεν έχουν τύπο και δεν χρειάζονται δήλωση.
- Οι μεταβλητές εμφανίζονται όταν καταχωρούμε σε αυτές και εξαφανίζονται όταν δεν τις χρησιμοποιούμε.
- Η καταχώρηση γίνεται με τον τελεστή = και η σύγκριση με τον τελεστή ==
- Μπορούμε να καταχωρίσουμε σε πολλές μεταβλητές ταυτόχρονα:

```
x, y, z = 1, 2, 3
first, second = second, first
a = b = 123
```

Εισαγωγή στην Python

- Τα blocks υποδεικνύονται μόνο μέσω εσοχών (indentation) κειμένου (όχι BEGIN/END ή αγκύλες).
- Συνηθισμένες δομές ελέγχου:

```
if x < 5 or 10 < x < 20:
    print "The value is OK."

for i in [1,2,3,4,5]:
    print "This is iteration number", i

# Print out the values from 0 to 99 inclusive.
for value in range(100):
    print value

x = 10
while x >= 0:
    print "x is still not negative."
    x = x-1
```

Εισαγωγή στην Python

- Αλληλεπίδραση με το χρήστη:

```
x = input("Please enter a number:")
print "The square of that number is", x*x
```

- Αποθήκευση εισόδου αυτολεξεί σαν αλφαριθμητικό και έλεγχος σφάλματος:

```
x = raw_input("Please enter a number:")
try:
    y = int(x)
    print "The square of that number is", y*y
except:
    print "That was not a number!"
```

Εισαγωγή στην Python

- Θεωρήστε τη δομή της λίστας σαν ετερογενή πίνακα:

```
name = ["Poppins", "Mary"]
x = [[1,2,3],[y,z],[[[]]]]
print len(x) # prints 3
```

- Δεικτοδότηση στις λίστες (το πρώτο στοιχείο έχει δείκτη 0):

```
print name[1], name[0] # prints "Mary Poppins"
name[0] = "Shelley"
```

- Τεμαχισμός στις λίστες:

```
x = ["super","cali","fragi","listic","expial","idocious"]
print x[3:5] # prints ["listic", "expial"]
print x[:2]  # prints ["super", "cali"]
print x[4:]  # prints ["expial", "idocious"]
print x[-3]  # prints "listic"
```

Εισαγωγή στην Python

- Τα λεξικά είναι μη ταξινομημένες λίστες και η δεικτοδότηση γίνεται με κλειδιά:

```
book = {'author':"Mary Shelley", 'title':"Frankenstein",\
  'year':1831}
print book['author'] # prints 'Mary Shelley'
book['title'] = "Mary Poppins"
```

- Τα λεξικά μπορούν να περιέχουν άλλα λεξικά ή και λίστες.
- Και οι λίστες μπορούν να περιέχουν λεξικά.

Εισαγωγή στην Python

- Ορισμός συνάρτησης:

```
def square(x):
    return x * x

print square(2) # Prints 4
```

- Οι παράμετροι περνάνε “by assignment”:

```
def f(x):
    x = 0
y = 1
f(y)
print y # Prints 1
```

```
def change(some_list):
    some_list[1] = 4
x = [1,2,3]
change(x)
print x # Prints out [1,4,3]
```

Εισαγωγή στην Python

```
def addition(x,y):
    return x+y

print addition(1,2)                 # prints 3
print addition('dobe','dobedo')     # prints dobedobedo
print addition(1.25, 5.36)          # prints 6.61
```

Εισαγωγή στην Python

- Δήλωση κλάσης:

```
class Basket:

    # Always remember the *self* argument
    def __init__(self, contents=None):
        self.contents = contents or []

    def add(self, element):
        self.contents.append(element)

    def print_me(self):
        result = ""
        for element in self.contents:
            result = result + " " + `element`
        print "Contains:" + result
```

Εισαγωγή στην Python

- Όλες οι μέθοδοι έχουν το έξτρα όρισμα `self`.
- Διάφορα προκαθορισμένα ονόματα μεθόδων έχουν ιδιαίτερη σημασία για τα αντικείμενα.
- Το όνομα `__init__` καθορίζει τη συνάρτηση που θα κληθεί όταν δημιουργηθεί ένα αντικείμενο της κλάσης.
- Τα ορίσματα στις μεθόδους μπορεί να είναι προαιρετικά και να έχουν προκαθορισμένη τιμή:

```
def foo(self, bar=32):
    ...
```

Η `foo` μπορεί να κληθεί με 0 ή 1 όρισμα. Αν δεν δοθεί όρισμα τότε η παράμετρος `bar` θα έχει την τιμή 32.

Εισαγωγή στην Python

- Οι μέθοδοι καλούνται σαν `object.method(arg1,arg2)`
- Τα backquote (``) μετατρέπουν ένα αντικείμενο στην αλφαριθμητική του αναπαράσταση:

```
element = 1
print `element` # prints 1
print 'element' # prints element
```

- Το σύμβολο + ενώνει δύο λίστες.
- Τα αλφαριθμητικά είναι λίστες χαρακτήρων, επιδέχονται δηλαδή δεικτοδότηση, τεμαχισμό και τη χρήση της συνάρτησης `len`

Εισαγωγή στην Python

- Κάθε τιμή μπορεί να χρησιμοποιηθεί σαν λογική τιμή: οι "άδειες" τιμές `[]`, `0`, `""` και `None` αναπαριστούν το false.
- Οι λογικές εκφράσεις αποτιμούνται με lazy evaluation: `a and b`, αν το `a` είναι false επιστρέφει την τιμή του, αλλιώς επιστρέφει την τιμή του `b`. Αντίστοιχα για το `or`.
- Μπορούμε λοιπόν να γράφουμε σύντομα και κομψά:

```
print a or b
```

αντί του

```
if a:
    print a
else:
    print b
```

Εισαγωγή στην Python

- Χρήση της κλάσης `Basket`:

```
b = Basket(['apple','orange'])
b.add("lemon")
b.print_me
```

- Χρήση της προκαθορισμένης μεθόδου `__str__`:

```
Class Basket:
...
def __str__(self):
  result = ""
  for element in self.contents:
      result = result + " " + `element`
  return "Contains:"+result
...
b = Basket(['apple','orange'])
print b
```

Εισαγωγή στην Python

- Χρήσιμες συναρτήσεις και κλάσεις τοποθετούνται σε modules που πρέπει να κληθούν για να χρησιμοποιηθούν:
- Για να χρησιμοποιήσουμε τη μέθοδο `split` από το standard module string:

```
import string
x = string.split(y)
```

είτε:

```
from string import split
x = split(y)
```

- Τη στιγμή του import εκτελείται όλος ο κώδικας του module. Αν θέλουμε το πρόγραμμα να είναι και module και εκτελέσιμο τότε προσθέτουμε στο τέλος:

```
if __name__ == "__main__":
    go() # any python code
```

Εισαγωγή στην Python

- Έλεγχος σφαλμάτων:

```
def safe_division(a,b):
    try:
        return a/b
    except ZeroDivisionError:
        return None
```

ή

```
try:
    unsafe_division(a,b):
except ZeroDivisionError:
    print "Something was divided by zero in unsafe_division"
```

- Στις περιπτώσεις όπου “φυσιολογικά” δεν πρέπει να υπάρχει πρόβλημα, αλλά κάποιο πρόβλημα “μπορεί να” υπάρξει η χρήση των exceptions είναι καλή πρακτική.

Εισαγωγή στην Python

Χρήση του ίδιου κώδικα για διαφορετικούς τύπους δεδομένων

```
def insertion_sort(sequence):
  for j in range(1, len(sequence)):
    key = sequence[j]
    i = j-1
    while i >= 0 and sequence[i] > key:
      sequence[i+1] = sequence[i]
      i = i-1
    sequence[i+1] = key
  return sequence

print insertion_sort([2,7,5,3])
print insertion_sort(['s','c','f','l','e','i'])
print insertion_sort([('b',(2,1)),('a',(9,7)),('b',(1,1))])

# prints:
# [2, 3, 5, 7]
# ['c', 'e', 'f', 'i', 'l', 's']
# [('a', (9, 7)), ('b', (1, 1)), ('b', (2, 1))]
```

Υπολογιστική Γεωμετρία

Γιατί με χρήση της Python;

- Σχεδόν μοιάζει με ψευδογλώσσα:

```
def factorial(num):
  return 1 if num==1 else num*factorial(num-1)

def symmetric_difference(set1, set2):
  return [x for x in set1 if x not in set2] +
    [x for x in set2 if x not in set1]
```

- “Οι μπαταρίες περιέχονται μέσα στη συσκευασία”:
  - εξαιρετική τεκμηρίωση μέσα στην ίδια τη γλώσσα,
  - πληθώρα ενσωματωμένων δομών δεδομένων,
  - πληθώρα βιβλιοθηκών για επιστημονικούς υπολογισμούς,
  - ενθουσιώδη κοινότητα.

Αναπαράσταση σημείων

- Μια κλάση της Python που υλοποιεί στιγμιότυπα σημείων στο επίπεδο (περιέχεται στο αρχείο κειμένου `point.py`):

```
class Point2(object):
  def __init__(self, x, y):
    self.x = x
    self.y = y
  def __repr__(self):
    return "Point2(%s, %s)" % (self.x, self.y)
```

Αναπαράσταση σημείων

- Μια κλάση της Python που υλοποιεί στιγμιότυπα σημείων στο επίπεδο (περιέχεται στο αρχείο κειμένου `point.py`):

```
class Point2(object):
  def __init__(self, x, y):
    self.x = x
    self.y = y
  def __repr__(self):
    return "Point2(%s, %s)" % (self.x, self.y)
```

- Παράδειγμα χρήσης στον διερμηνέα:

```
>>> from point import Point2
>>> a, b = Point2(7.32, 0), Point2(0.12342, -10.23241)
>>> print a,b
Point2(7.32, 0) Point2(0.12342, -10.23241)
>>>
```

Αναπαράσταση σημείων

- Εμπλουτισμένη κλάση για τα σημεία:

```
class Point2(object):
  def __init__(self, x, y):
    self.x, self.y = x, y
  def __repr__(self):
    return "Point2(%s, %s)" % (self.x, self.y)
  @classmethod
  def from_point2(cls, point2):
    return cls(point2.x, point2.y)
  @classmethod
  def from_tuple(cls, tup):
    return cls(tup[0], tup[1])
  @property
  def coordinates(self):
    return self.x, self.y
  @coordinates.setter
  def coordinates(self, tup):
    self.x, self.y = tup[0], tup[1]
```

Αναπαράσταση σημείων

- Παράδειγμα χρήσης της εμπλουτισμένης κλάσης στον διερμηνέα:

```
>>> from point import Point2
>>> p1 = Point2(3.5,-4)
>>> p2 = Point2.fromPoint2(p1)
>>> p3 = Point2.fromTuple(p2.coordinates)
>>> print p1, p2, p3
Point2(3.5, -4) Point2(3.5, -4) Point2(3.5, -4)
>>>
```

Αναπαράσταση ευθυγράμμων τμημάτων

- Μια κλάση της Python που υλοποιεί στιγμιότυπα ευθυγράμμων τμημάτων στο επίπεδο:

```
class Segment2(object):
  def __init__(self, start, end):
    self.start = start
    self.end = end
  def __repr__(self):
    return "Segment2(%s, %s)" % (self.start, self.end)
```

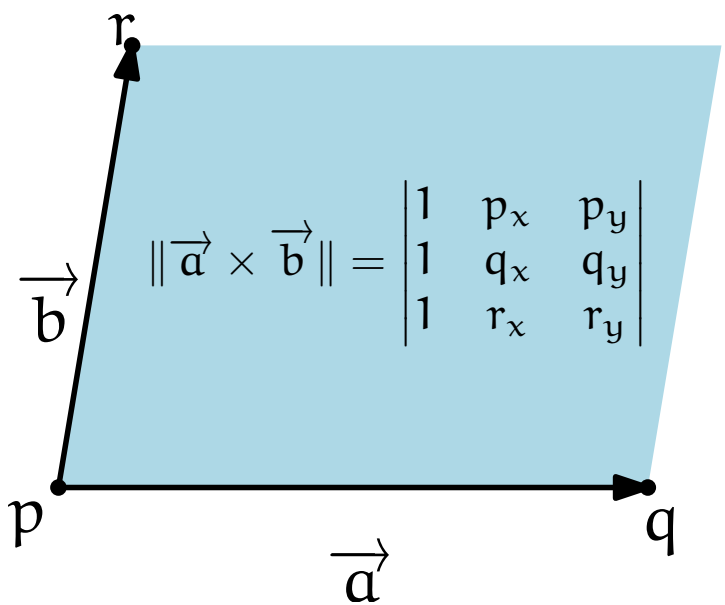
Αναπαράσταση ευθυγράμμων τμημάτων

- Παράδειγμα χρήσης στον διερμηνέα:

```
>>> from point import Point2
>>> from segment import Segment2
>>> p = Point2(1.2, 3.21)
>>> q = Point2(-2, 2.34)
>>> s = Segment2(p, q)
>>> print s
Segment2(Point2(1.2, 3.21), Point2(-2.0, 2.34))
```

Εμβαδό τριγώνου

- Από το σχολείο: $E = \frac{\text{βάση} \times \text{ύψος}}{2}$
- Το τρίγωνο δίνεται σαν μια τριάδα σημείων $p, q, r = (p_x, p_y), (q_x, q_y), (r_x, r_y)$
- Το εξωτερικό γινόμενο δύο διανυσμάτων έχει μέτρο ίσο με το εμβαδό του παραλληλογράμμου που ορίζουν.



Υπολογισμός του εμβαδού

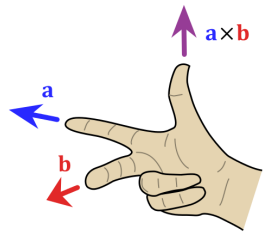
- Μια συνάρτηση Python που υπολογίζει το *διπλάσιο* του εμβαδού του τριγώνου pqr:

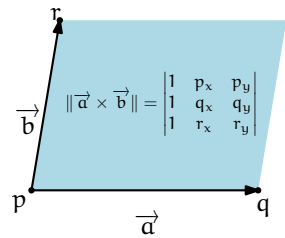
```
def area2(p, q, r):
  return (r.y-p.y) * (q.x-p.x) - (q.y-p.y) * (r.x-p.x)
```

- Παράδειγμα χρήσης στον διερμηνέα:

```
>>> from point import Point2
>>> from utilities import area2
>>> p = Point(0, 0)
>>> q = Point(3, 0)
>>> r = Point(0, 4)
>>> area2(p, q, r)
12.0
>>> area2(q, p, r)
-12.0
>>> area2(p, q, q)
0.0
>>>
```

Πρόσημο του εμβαδού





- Κανόνας του δεξιού χεριού.
- *Θετικό* εμβαδό στο τρίγωνο pqr σημαίνει ότι το διάνυσμα $\overrightarrow{\mathrm{a}} = \overrightarrow{\mathrm{pq}}$ πρέπει να στρίψει αριστερά για να βρει το $\overrightarrow{\mathrm{b}} = \overrightarrow{\mathrm{pr}}$.
- Με άλλα λόγια, θετικό εμβαδό στο τρίγωνο pqr σημαίνει ότι το σημείο r βρίσκεται στα *αριστερά* του $\overrightarrow{\mathrm{pq}}$.
- *Αρνητικό* εμβαδό στο τρίγωνο qpr σημαίνει ότι το σημείο r βρίσκεται στα *δεξιά* του $\overrightarrow{\mathrm{qp}}$
- *Μηδενικό* εμβαδό στο τρίγωνο pqr σημαίνει ότι τα σημεία p, q, r είναι *συνευθειακά*.

Κατηγορήματα Προσανατολισμού

- Συναρτήσεις της Python που υλοποιούν κατηγορήματα προσανατολισμού:

```
def ccw(p, q, r):
  return area2(p, q, r) > 0

def cw(p, q, r):
  return area2(p, q, r) < 0

def collinear(p, q, r):
  return area2(p, q, r) == 0

def between(p, q, r):
  if not collinear(p, q, r):
    return False
  if p.x != q.x:
    return p.x <= r.x <= q.x or p.x >= r.x >= q.x
  else:
    return p.y <= r.y <= q.y or p.y >= r.y >= q.y
```

Κατηγορήματα Προσανατολισμού

- Παράδειγμα χρήσης στον διερμηνέα:

```
>>> from point import Point2
>>> from predicates import *
>>> p = Point2(-0.00342324, 2.03424345)
>>> q = Point2(23.47029054, 3.34344444)
>>> r = Point2(-2.1, -23.00009389)
>>> ccw(p, q, r)
False
>>> cw(p, q, r)
True
>>> collinear(p, q, r)
False
>>> collinear(p, q, q)
True
>>> collinear(q, q, q)
True
```

Διάταξη σημείων

- Υλοποίηση λεξικογραφικής διάταξης για τα στιγμιότυπα της κλάσης `Point`:

```
class Point2(object):
    ...
  def __eq__(self, other):
    return (self.x, self.y) == (other.x, other.y)
  def __ne__(self, other):
    return (self.x, self.y) != (other.x, other.y)
  def __lt__(self, other):
    return (self.x, self.y) < (other.x, other.y)
  def __gt__(self, other):
    return (self.x, self.y) > (other.x, other.y)
  def __le__(self, other):
    return  (self.x, self.y) <= (other.x, other.y)
  def __ge__(self, other):
    return  (self.x, self.y) >= (other.x, other.y)
    ...
```

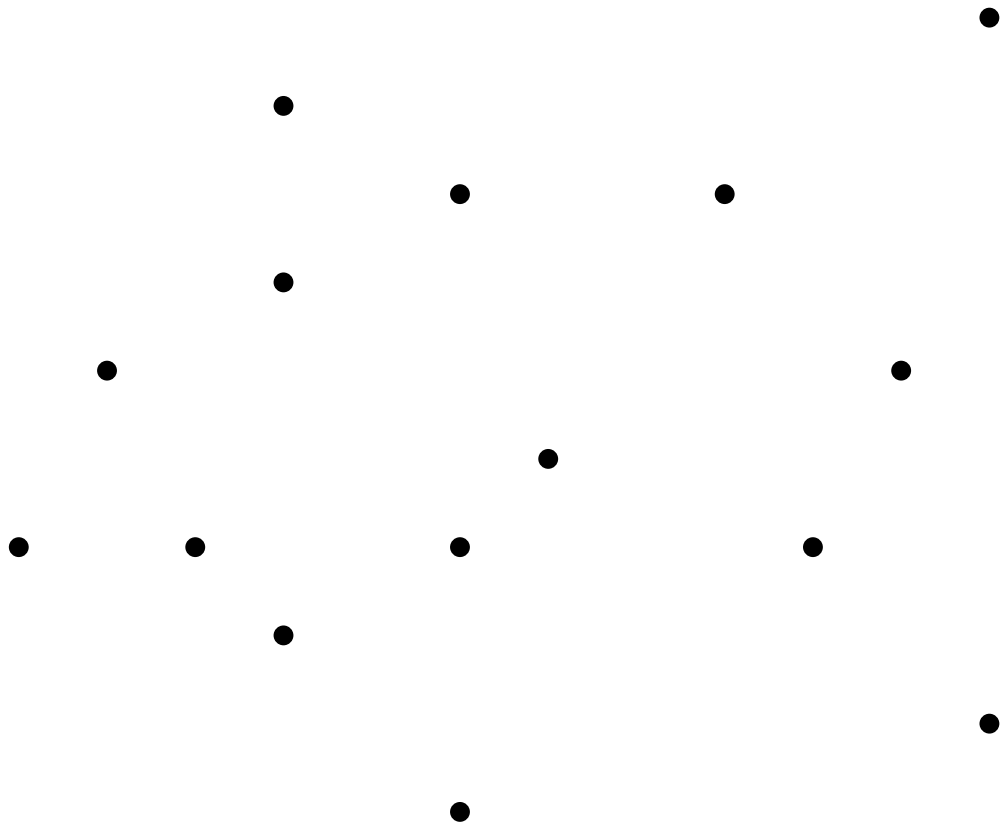
Διάταξη σημείων

- Παράδειγμα χρήσης στον διερμηνέα:

```
>>> from point import Point2
>>> p1 = Point2(3,4)
>>> p2 = Point2(3,3)
>>> p3 = Point2(2,4)
>>> p4 = Point2(2,2)
>>> p5 = Point2(0,4)
>>> point_list = [p1, p2, p3, p4, p5]
>>> for point in sorted(point_list):
...     print point
...
Point2(0, 4)
Point2(2, 2)
Point2(2, 4)
Point2(3, 3)
Point2(3, 4)
```

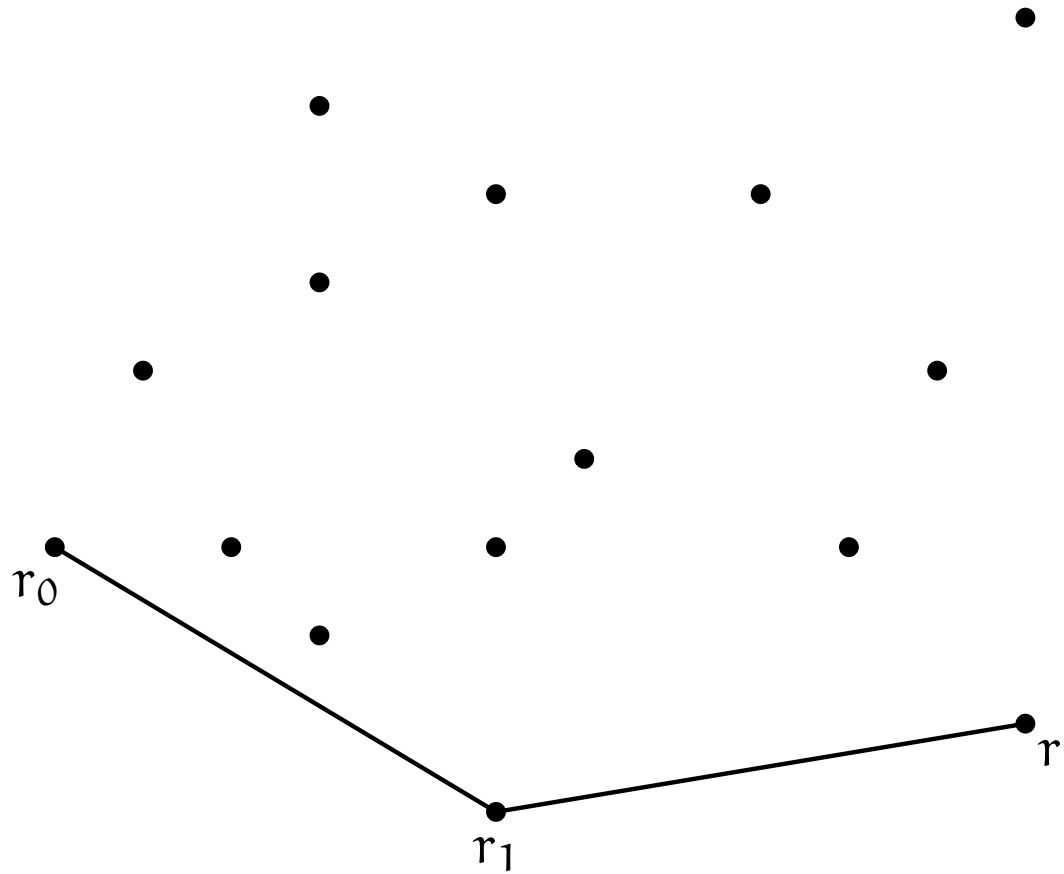
Αλγόριθμος του Jarvis

Εύρεση επόμενης κορυφής



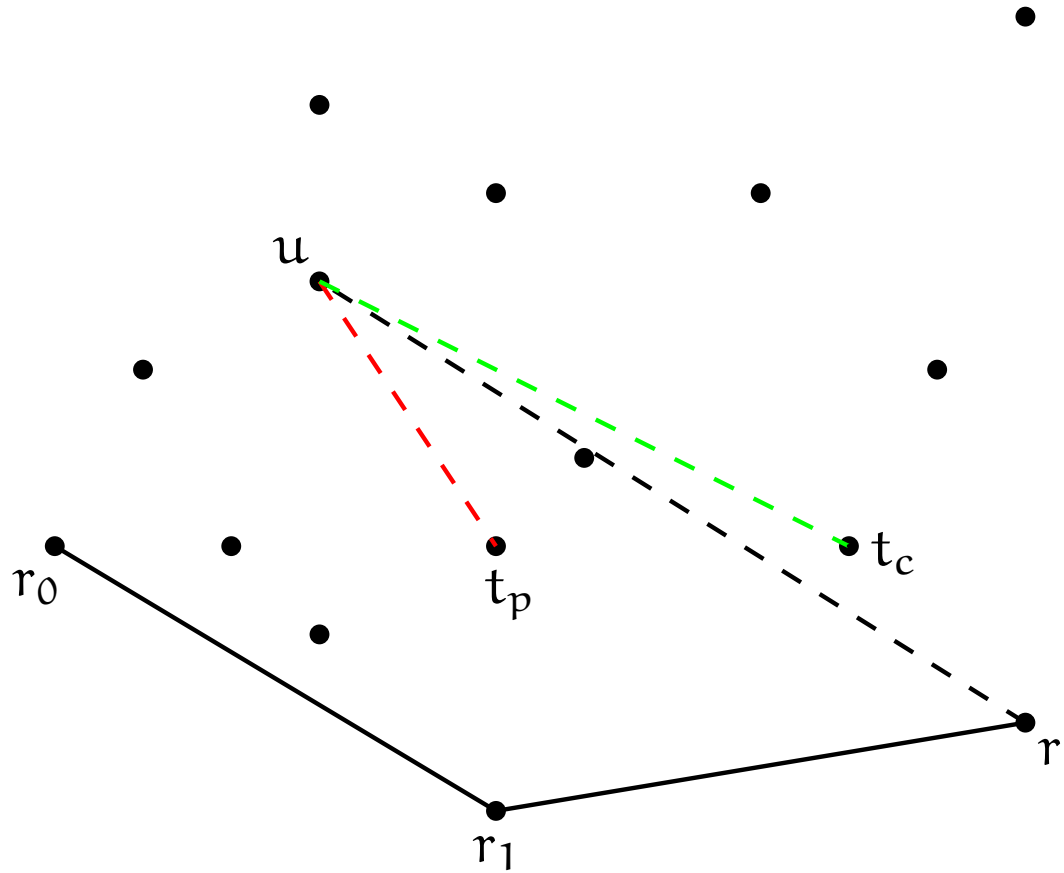
Αλγόριθμος του Jarvis

Εύρεση επόμενης κορυφής



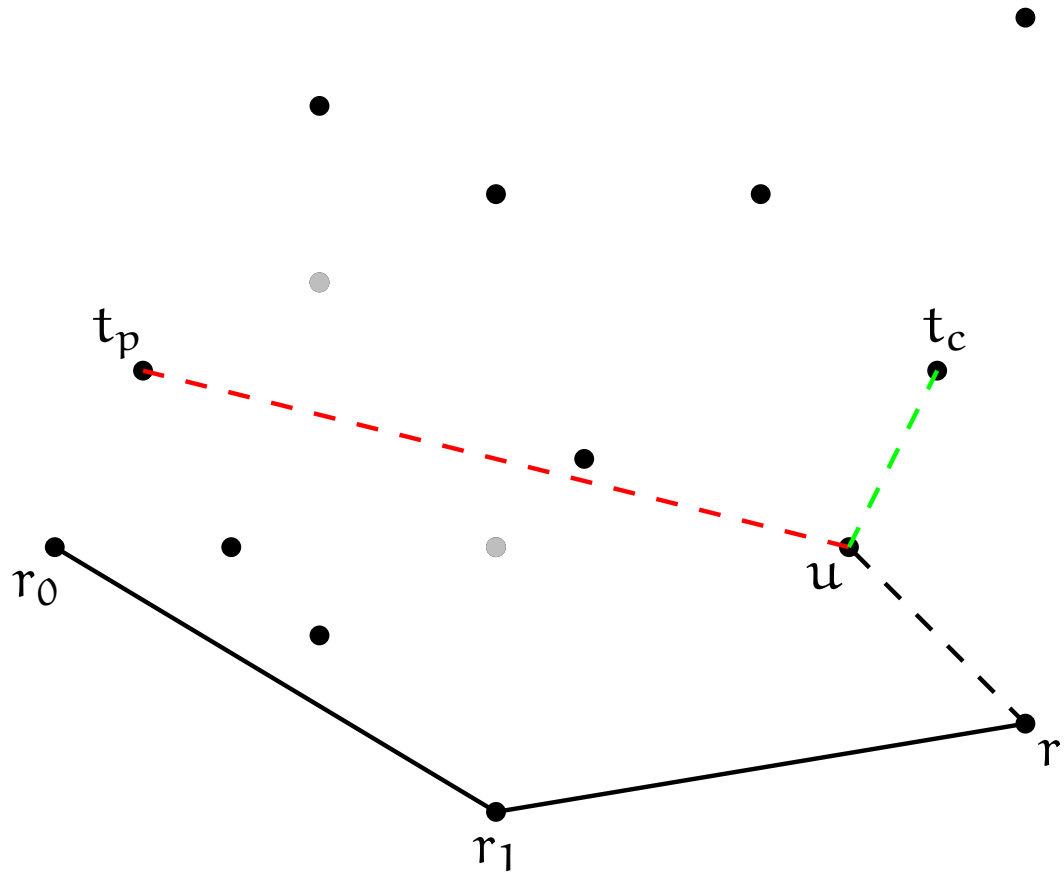
Αλγόριθμος του Jarvis

Εύρεση επόμενης κορυφής



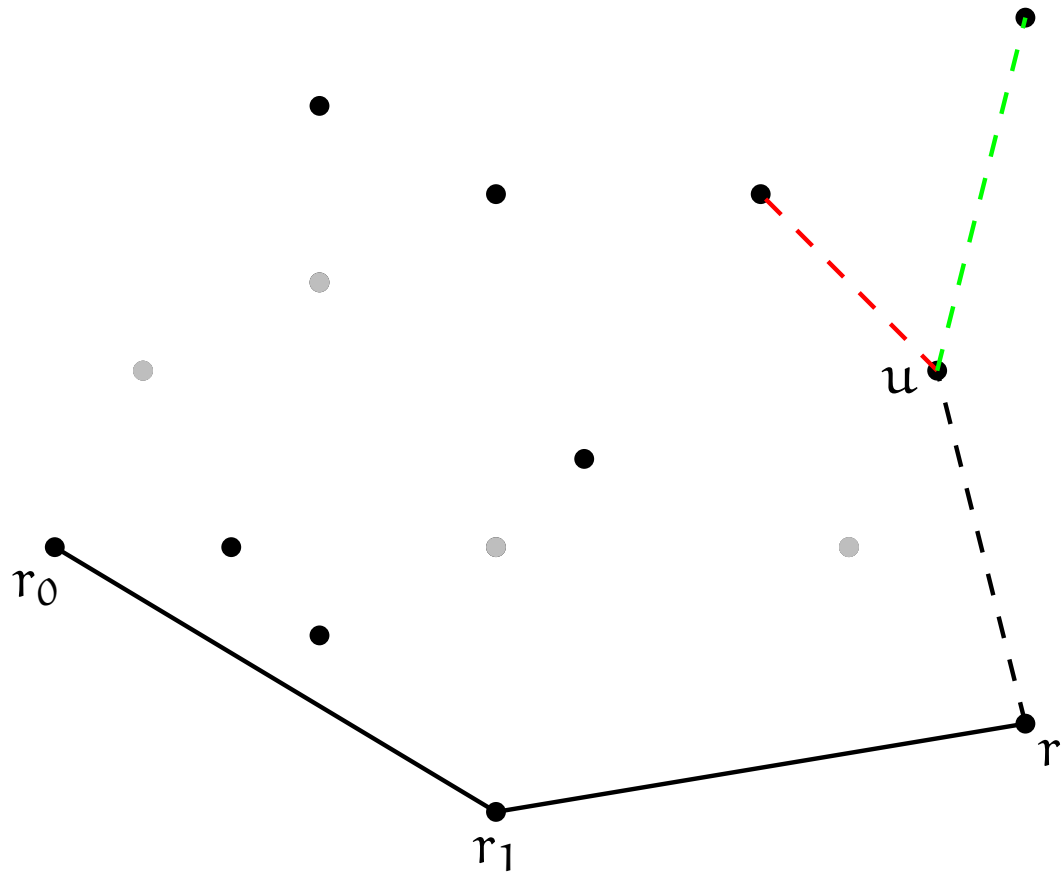
Αλγόριθμος του Jarvis

Εύρεση επόμενης κορυφής



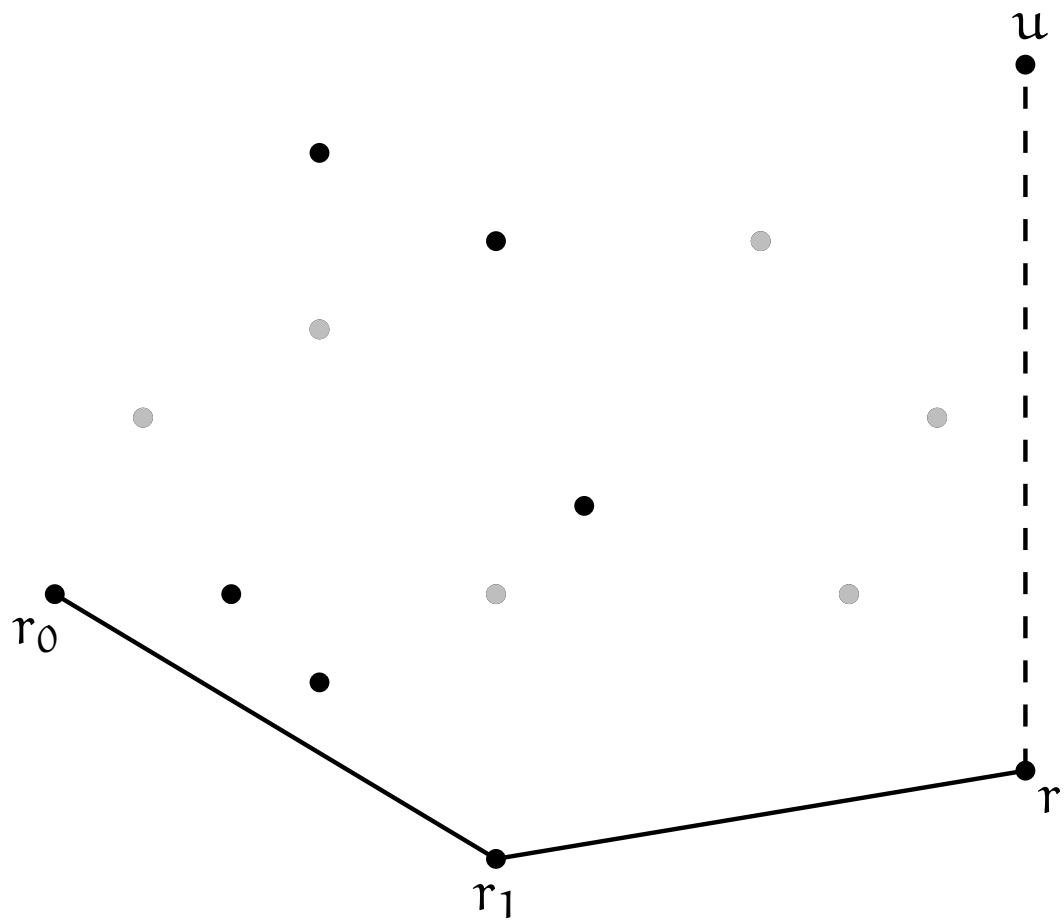
Αλγόριθμος του Jarvis

Εύρεση επόμενης κορυφής



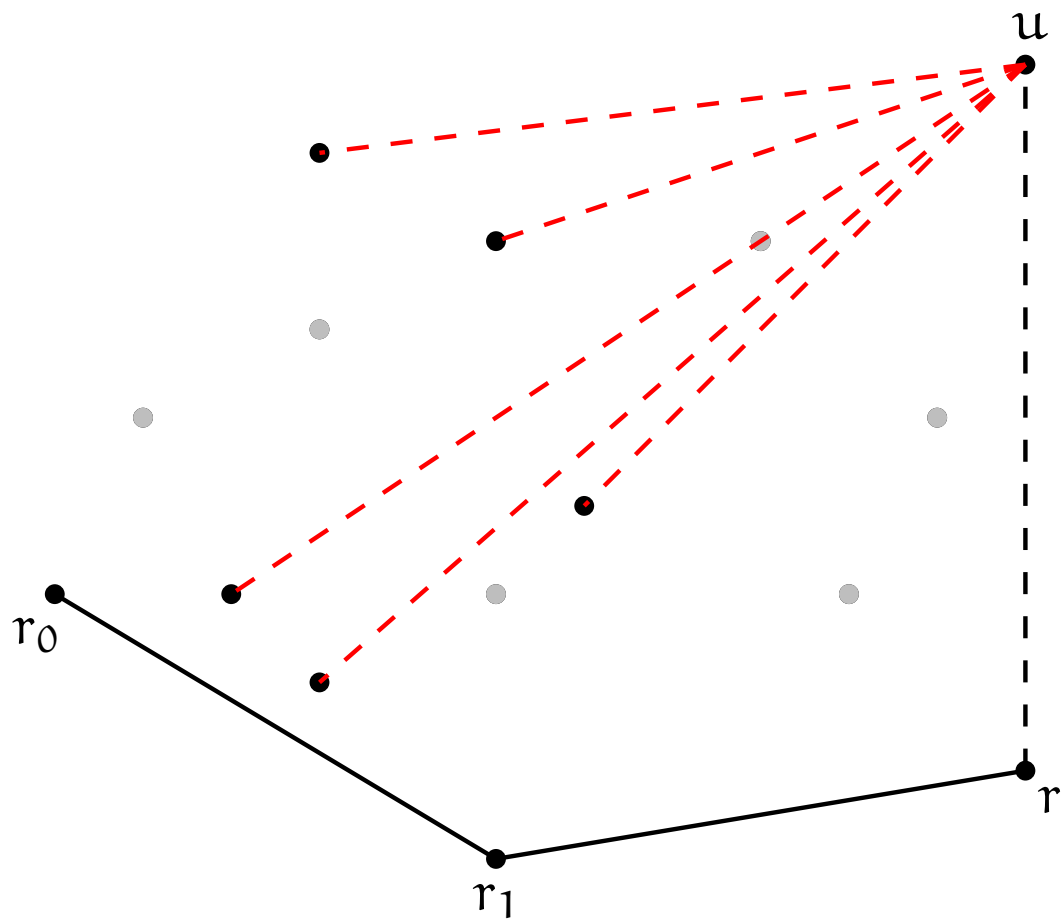
Αλγόριθμος του Jarvis

Εύρεση επόμενης κορυφής



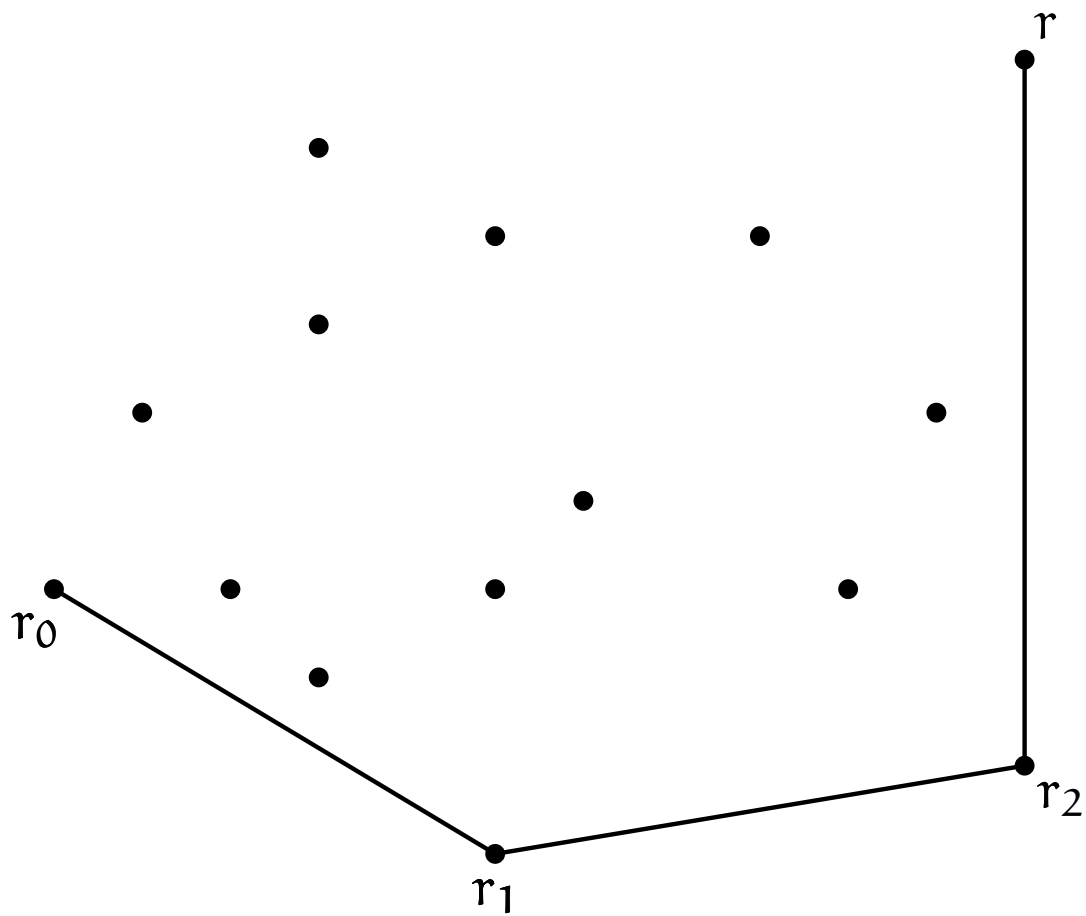
Αλγόριθμος του Jarvis

Εύρεση επόμενης κορυφής



Αλγόριθμος του Jarvis

Εύρεση επόμενης κορυφής



Αλγόριθμος του Jarvis για ΚΠ2

Υλοποίηση στην Python

```
from pycompgeom.predicates import *
import random

def jarvis(points):
  r = r0 = min(points)
  hull = [r0]
  u = None
  while u <> r0:
    u = random.choice(points)
    for t in points:
      if cw(r,u,t) or collinear(r,u,t) and between(r,t,u):
        u = t
    r = u
    points.remove(r)
    hull.append(r)
  return hull
```

Αλγόριθμος του Jarvis για ΚΠ2

Σύγκριση αλγόριθμου και υλοποίησης

Αλγόριθμος

```
Είσοδος: Σύνολο S αποτελούμενο από n σημεία στο επίπεδο.
Έξοδος: Η αλυσίδα των ακμών και των κορυφών του ΚΠ2.


1. Τρέχουσα κορυφή r=r0 είναι το "μικρότερο" σημείο.

2. Αρχικοποίηση αλυσίδας κορυφών με r. S=S-{r}.


3. Έστω r η τρέχουσα κορυφή και u που ανήκει στο S ένα
   οποιοδήποτε σημείο που δεν έχει επιλεγεί ως κορυφή.

   Για κάθε σημείο t που ανήκει στο S-{u}:
      αν ισχύει CW(r,u,t),
      ή αν r,u,t συνευθειακά και u εσωτερικό του (r,t)
   τότε θέσε u=t

4. Αν u=r0 τερμάτισε, αλλιώς
      r=u, S=S-{r},
      πρόσθεσε στην αλυσίδα των κορυφών το r,
      συνέχισε στο βήμα 3.
```

Υλοποίηση στην Python

```
from pycompgeom import *
import random
def jarvis(S):

  r = r0 = min(S)

  hull = [r0]
  u = None

  while u <> r0:
      u = random.choice(S)

      for t in S:
          if cw(r,u,t) or \
            collinear(r,u,t) and between(r,t,u):
            u = t


      r = u
      S.remove(r)
      hull.append(r)

  return hull
```

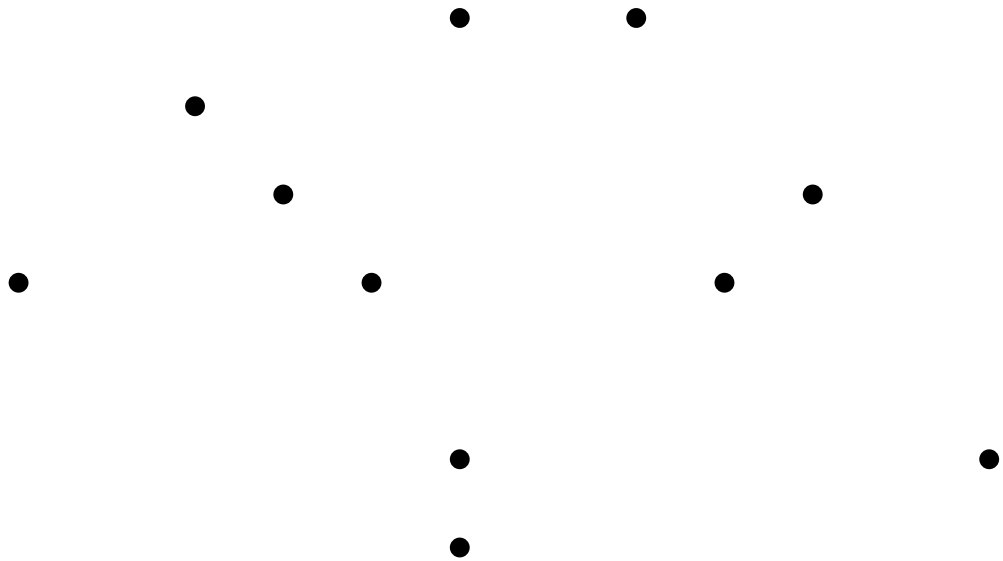
Αλγόριθμος του Andrew για ΚΠ2

Ιδέα

- Εξετάζουμε τα σημεία με αύξουσα λεξικογραφική σειρά.
- Τα δύο πρώτα σημεία αρχικοποιούν το άνω περίβλημα (αΠ) και το κάτω περίβλημα (κΠ).
- Σε κάθε βήμα τηρούμε το αΠ (κΠ) σαν μια cw (ccw) λίστα σημείων.
- Κάθε επόμενο σημείο p ενημερώνει το αΠ και το κΠ εξετάζοντας τη στροφή των δύο τελευταίων κάθε λίστας προς το p:
  - Όσο η στροφή είναι ccw κάνουμε pop από τη λίστα του αΠ.
  - Όσο η στροφή είναι cw κάνουμε pop από τη λίστα του κΠ.
  - Προσθέτουμε το p στο τέλος των λιστών αΠ και κΠ.
- Επιστρέφουμε τις λίστες αΠ και κΠ ή τις συγχωνεύουμε κατάλληλα για να πάρουμε το κυρτό περίβλημα.

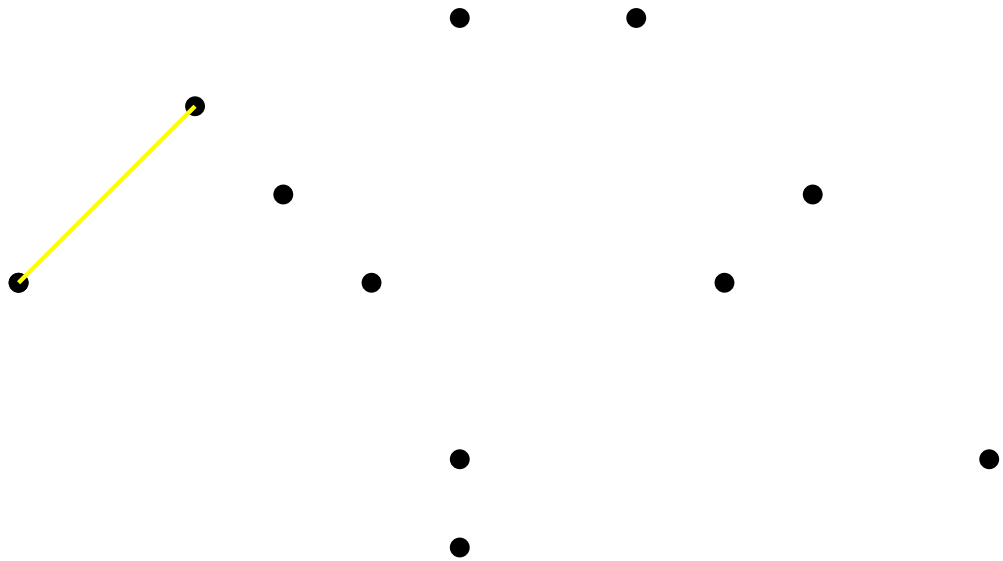
Αλγόριθμος του Andrew

Παράδειγμα



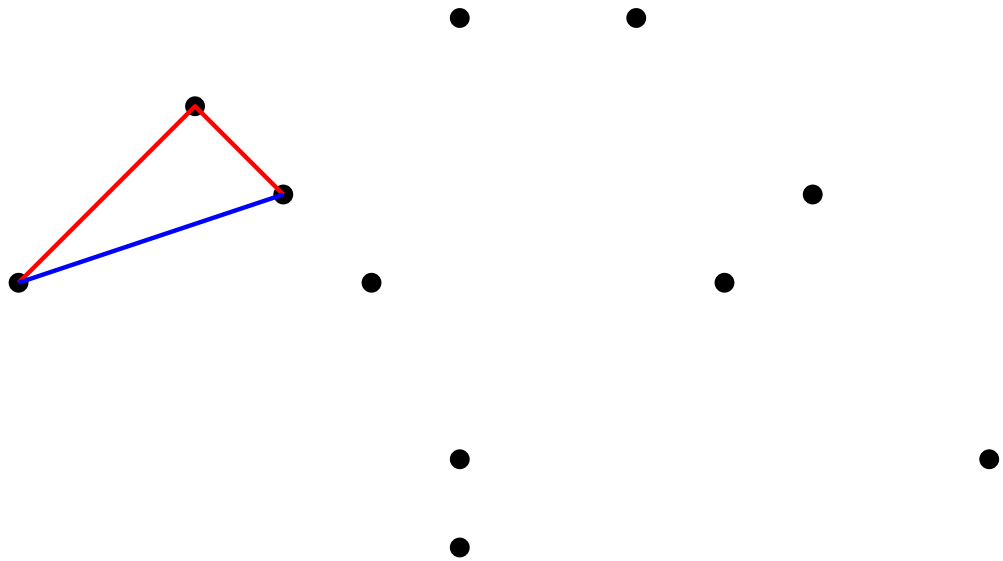
Αλγόριθμος του Andrew

Παράδειγμα



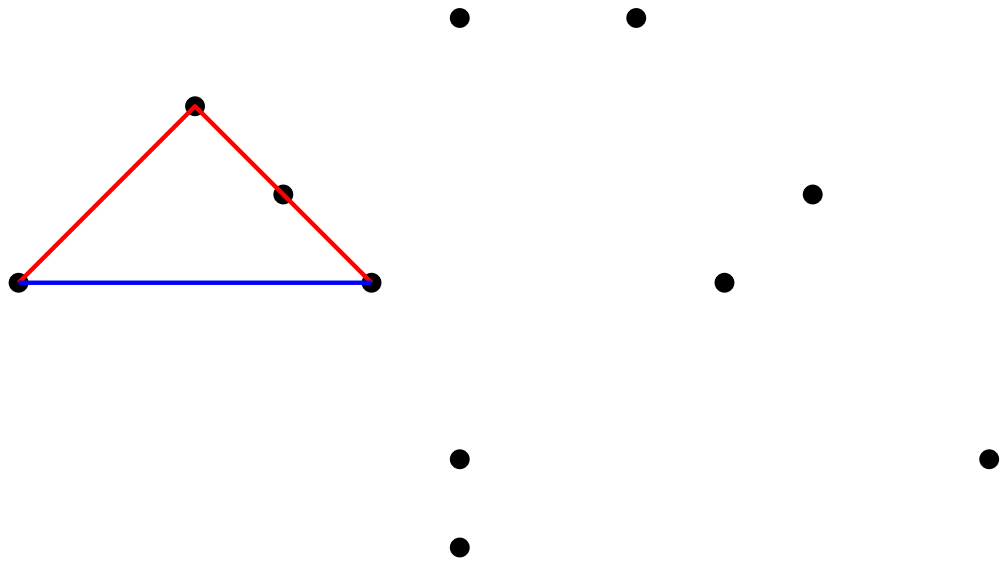
Αλγόριθμος του Andrew

Παράδειγμα



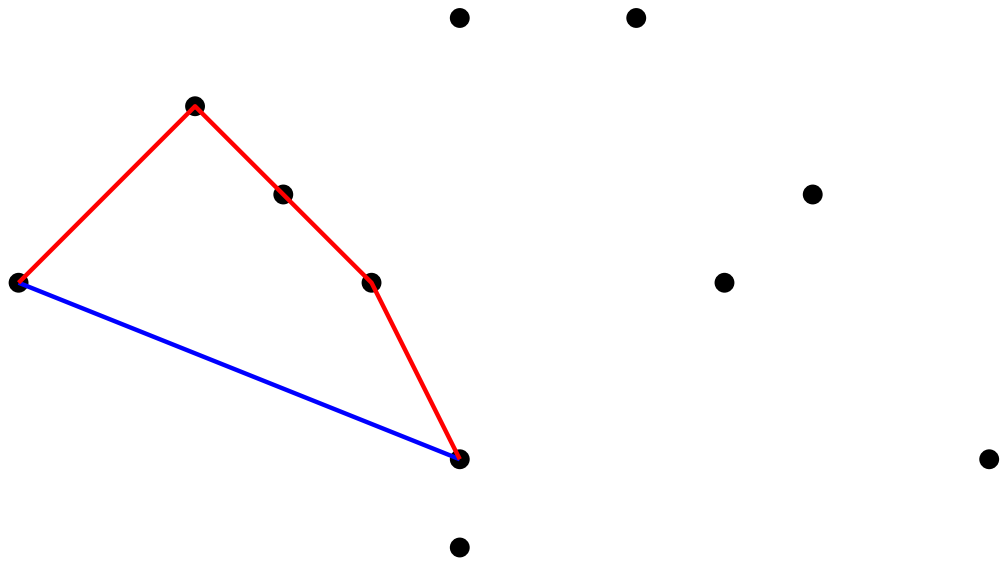
Αλγόριθμος του Andrew

Παράδειγμα



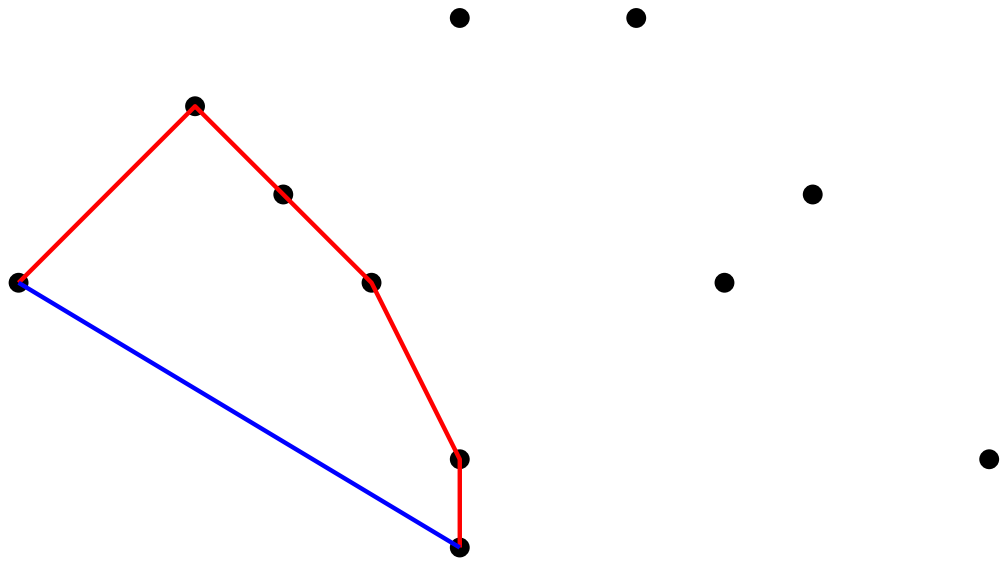
Αλγόριθμος του Andrew

Παράδειγμα



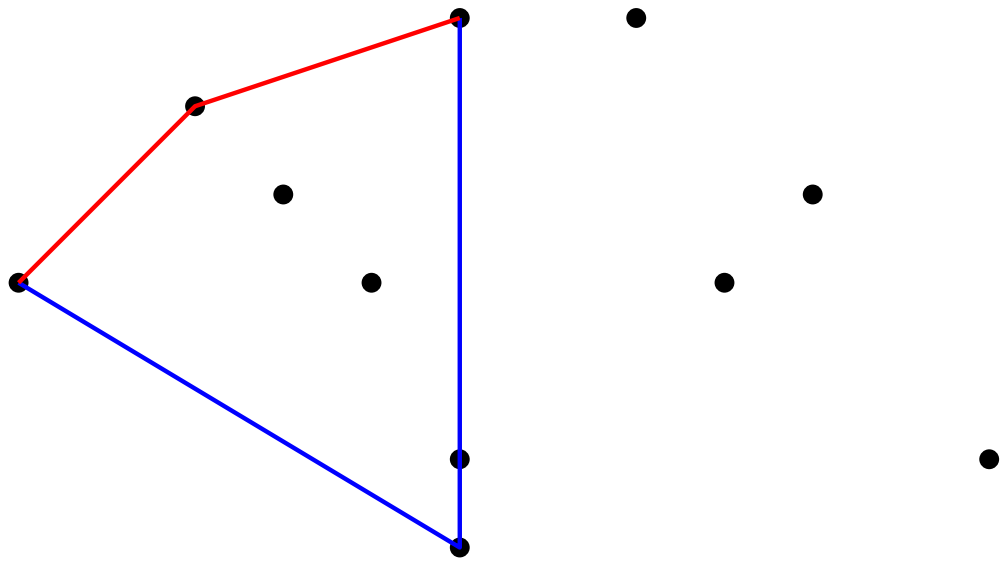
Αλγόριθμος του Andrew

Παράδειγμα



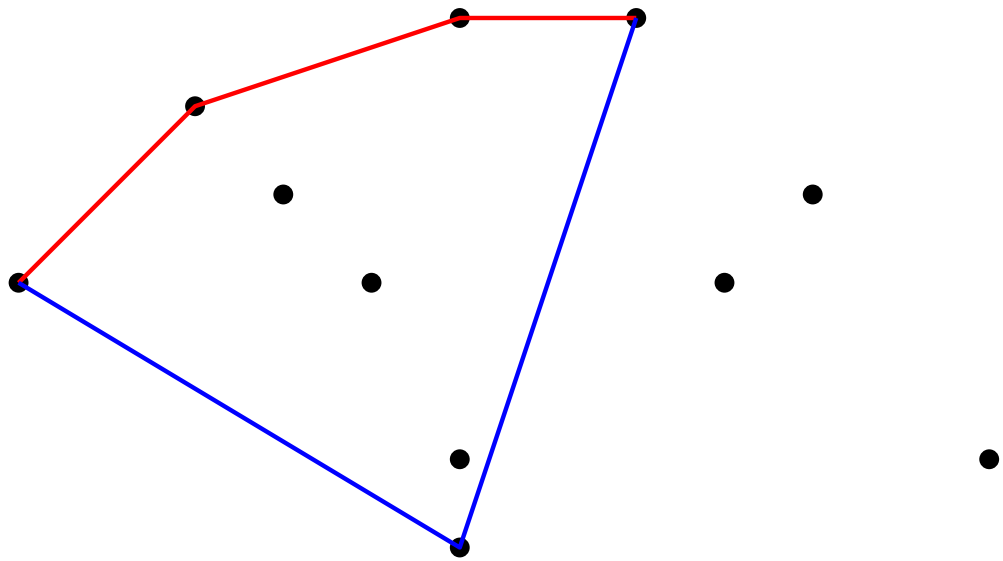
Αλγόριθμος του Andrew

Παράδειγμα



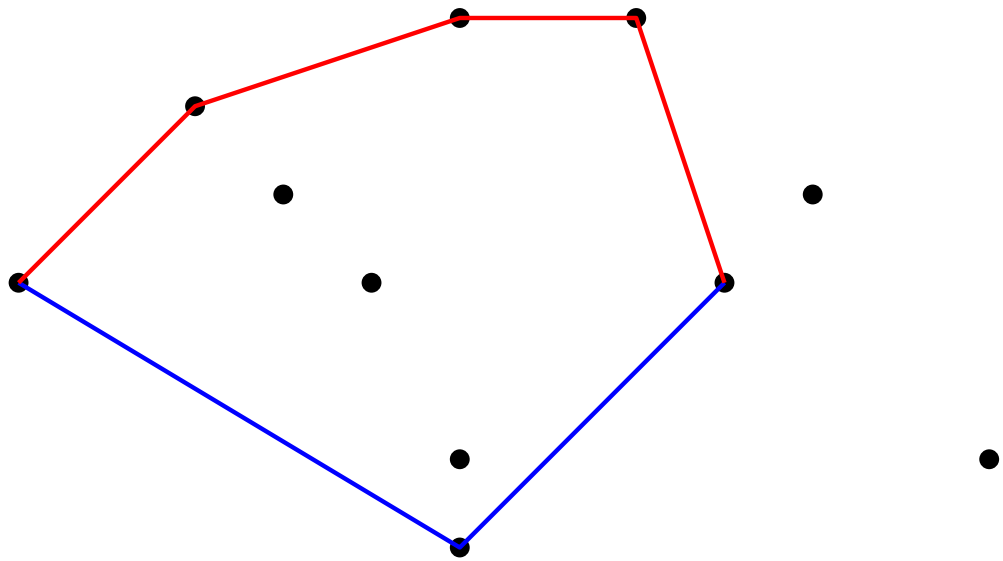
Αλγόριθμος του Andrew

Παράδειγμα



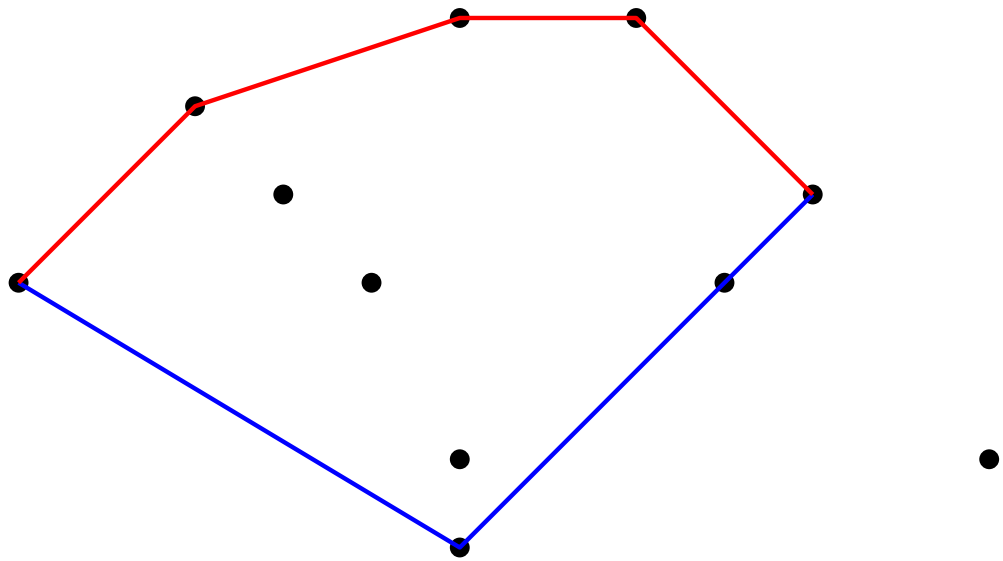
Αλγόριθμος του Andrew

Παράδειγμα



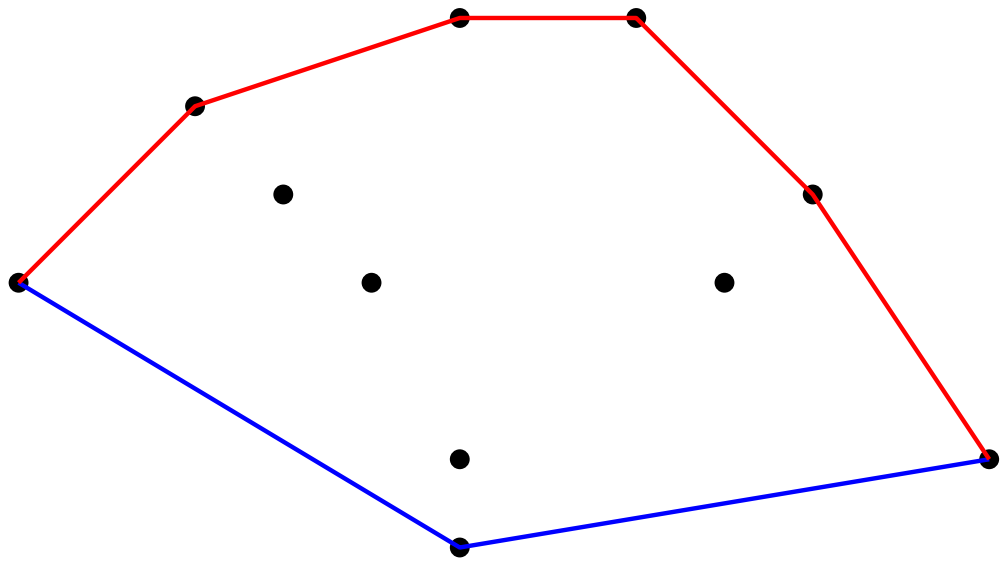
Αλγόριθμος του Andrew

Παράδειγμα



Αλγόριθμος του Andrew

Παράδειγμα



Αλγόριθμος του Andrew για ΚΠ2

Υλοποίηση στην Python

```
def andrew(points, return_hull=True):
  upper = []
  lower = []
  for p in sorted(points):
    while len(upper)>1 and ccwon(upper[-2], upper[-1], p):
      upper.pop()
    while len(lower)>1 and cwon(lower[-2], lower[-1], p):
      lower.pop()
    upper.append(point)
    lower.append(point)
  if return_hull:
    return lower[:-1]+ [x for x in reversed(upper[1:])]
  else:
    return upper, lower
```

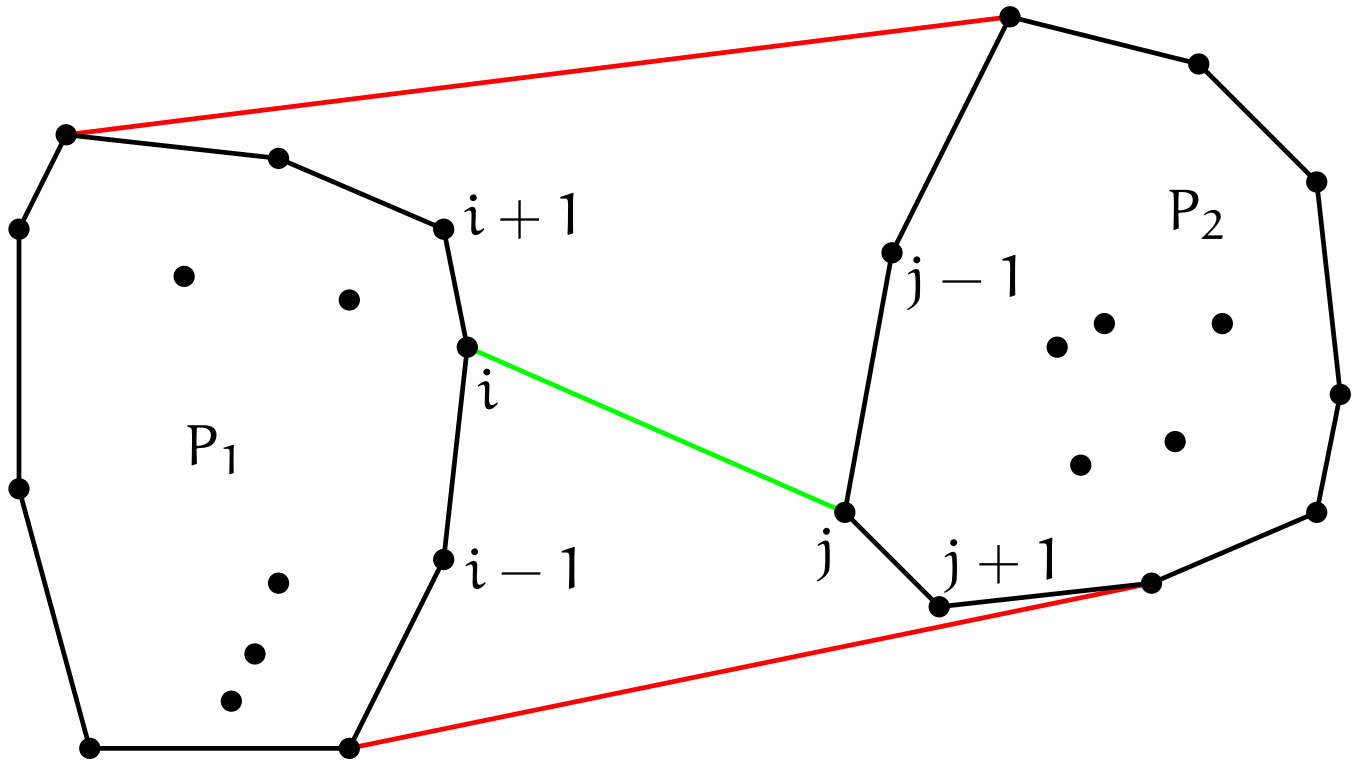
- Επιστρέφει το κυρτό περίβλημα σαν μια `ccw` λίστα σημείων.
- Αν η παράμετρος `return_hull` είναι `False` επιστρέφει δύο λίστες για το αΠ και το κΠ σε `cw` και `ccw` σειρά αντίτοιχα.

Αλγόριθμος Διαίρει και Βασίλευε για ΚΠ2

- Υπολογίζουμε το σημείο με την κεντρική τετμημένη και χωρίζουμε το αρχικό σημειοσύνολο στα δύο αντίστοιχα υποσύνολα.
- Εφαρμόζουμε τον αλγόριθμο για τα ΚΠ των δύο υποσυνόλων.
- Συνθέτουμε το συνολικό ΚΠ από τα δύο κυρτά περιβλήματα.
- Σύνθεση: εντοπισμός ακμών (*γέφυρες*) του συνολικού ΚΠ που $\notin$ σε κανένα από τα δύο ΚΠ των υποπροβλημάτων.
- Οι γέφυρες ενώνουν μια κορυφή του αριστερού ΚΠ με μια κορυφή του δεξιού ΚΠ.
- Η *πάνω γέφυρα*, από τα αριστερά προς τα δεξιά, αφήνει στα δεξιά τις υπόλοιπες κορυφές των ΚΠ των υποπροβλημάτων.
- Η *κάτω γέφυρα*, με την ίδια φορά, αφήνει στα αριστερά τις υπόλοιπες κορυφές των ΚΠ των υποπροβλημάτων.

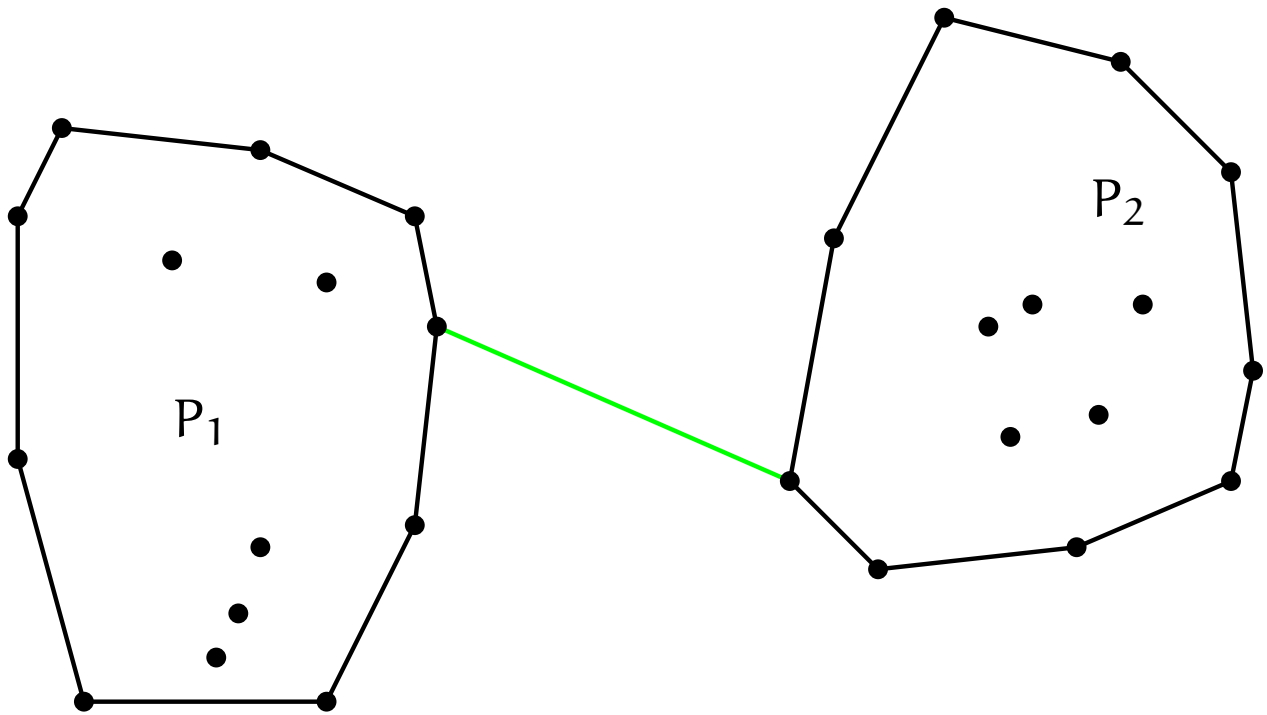
Αλγόριθμος Διαίρει και Βασίλευε για ΚΠ2

Βήμα σύνθεσης των υποπροβλημάτων



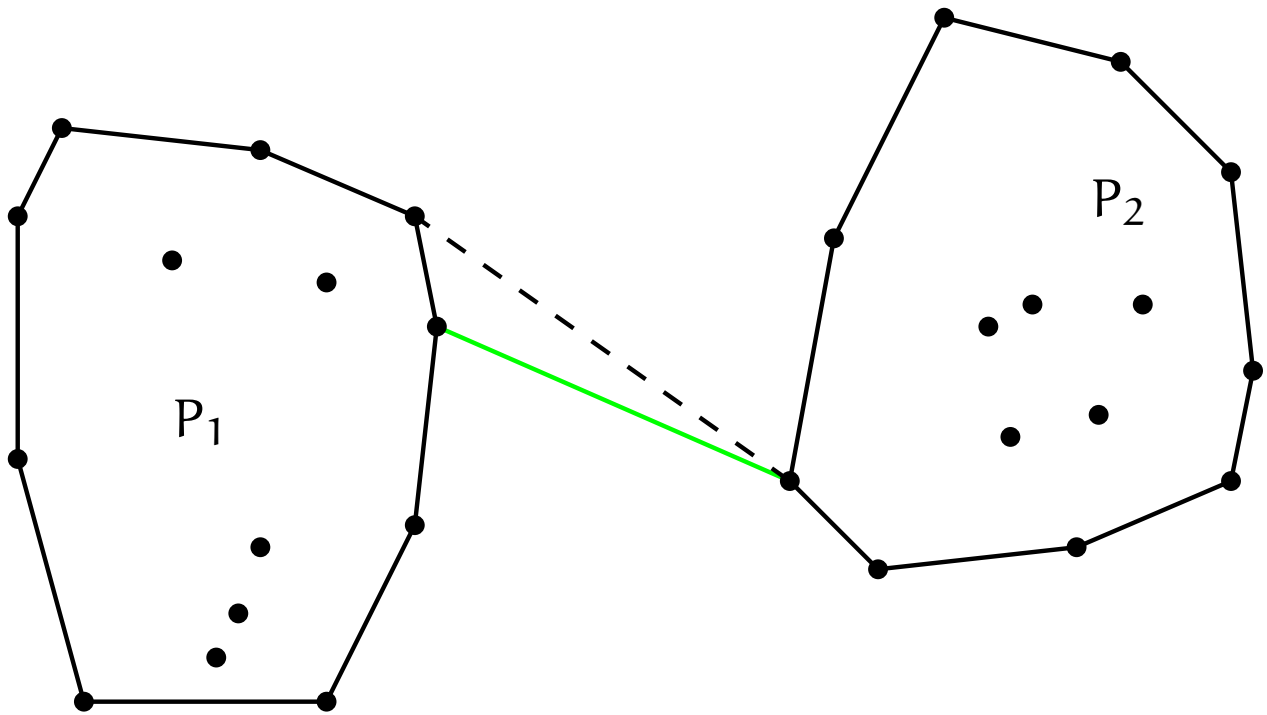
Αλγόριθμος Διαίρει και Βασίλευε για ΚΠ2

Παράδειγμα



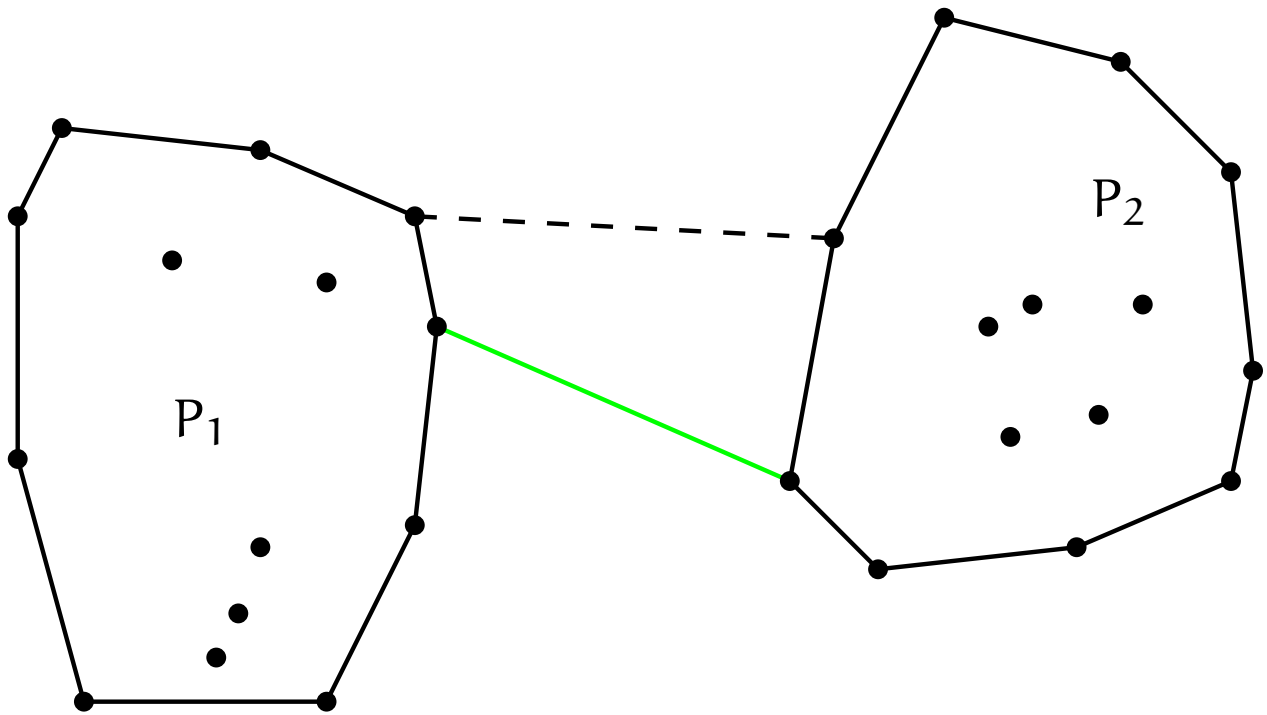
Αλγόριθμος Διαίρει και Βασίλευε για ΚΠ2

Παράδειγμα



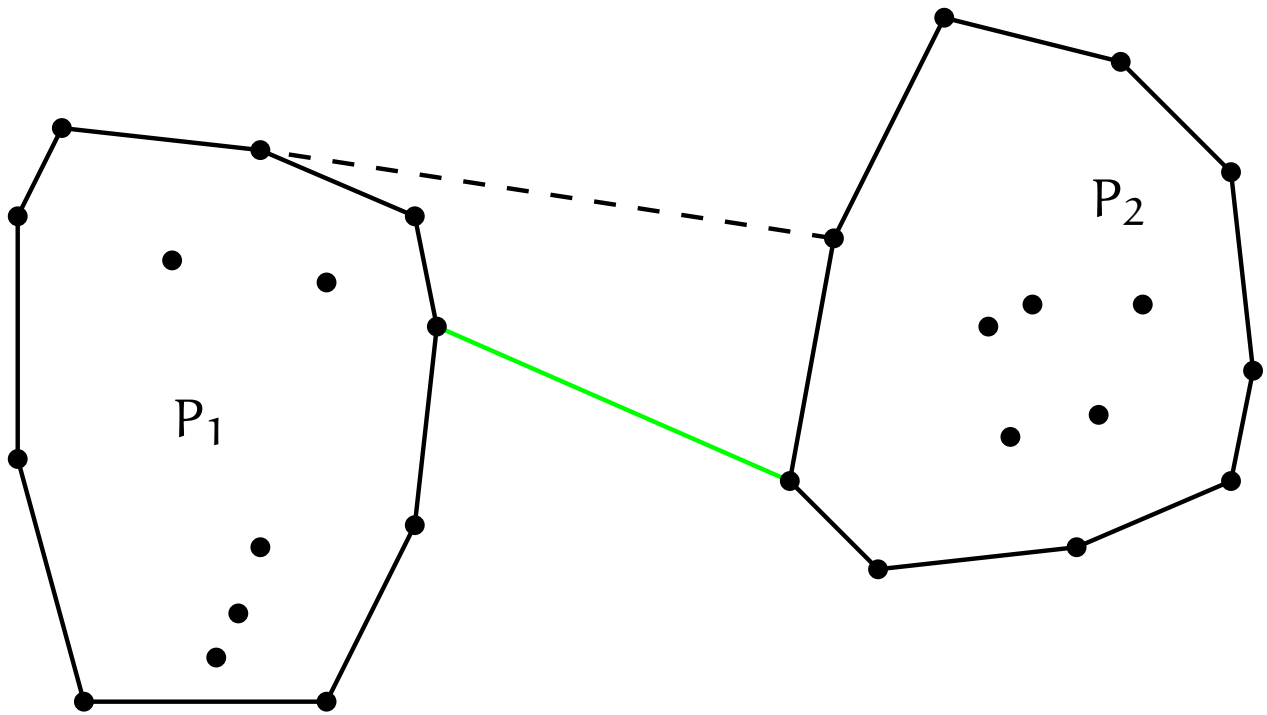
Αλγόριθμος Διαίρει και Βασίλευε για ΚΠ2

Παράδειγμα



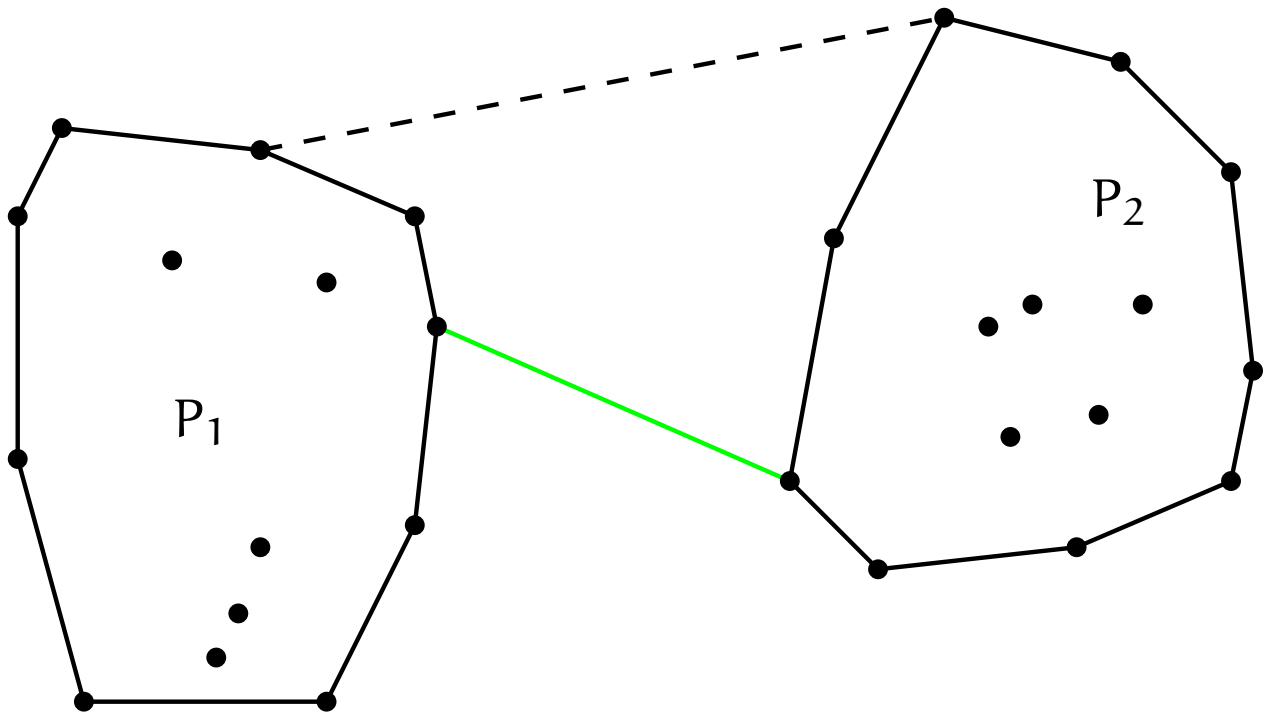
Αλγόριθμος Διαίρει και Βασίλευε για ΚΠ2

Παράδειγμα



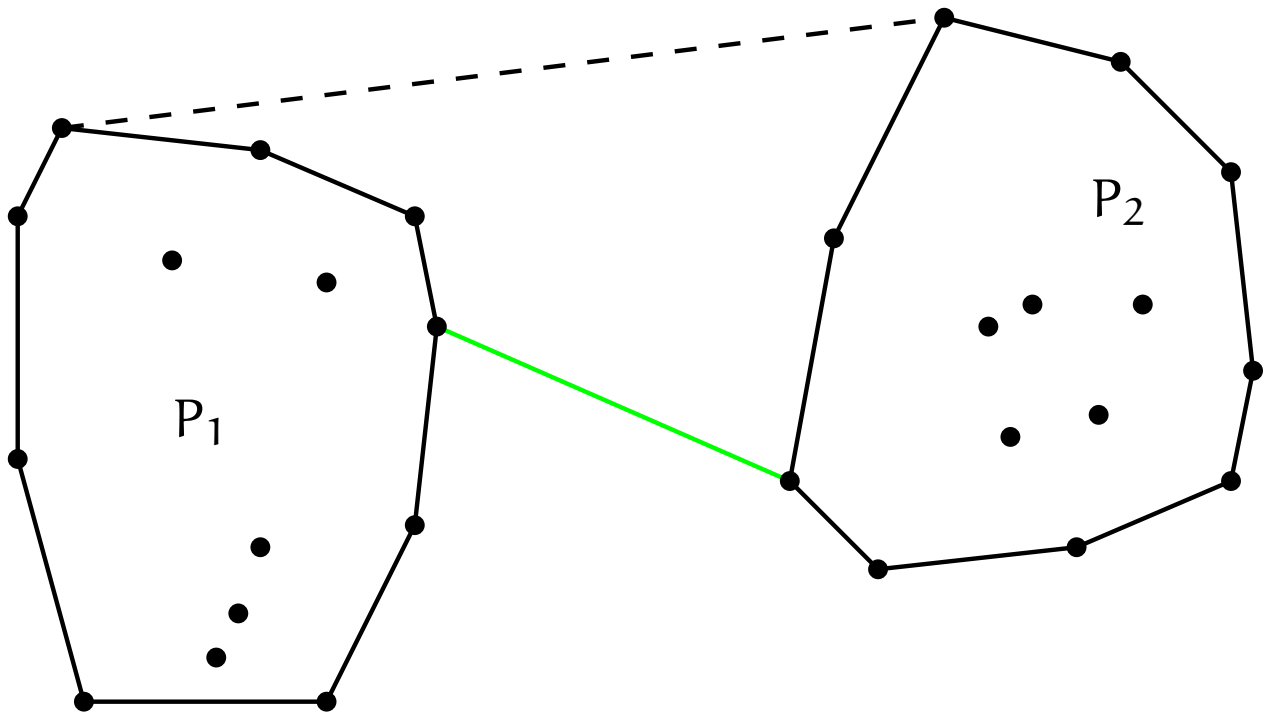
Αλγόριθμος Διαίρει και Βασίλευε για ΚΠ2

Παράδειγμα



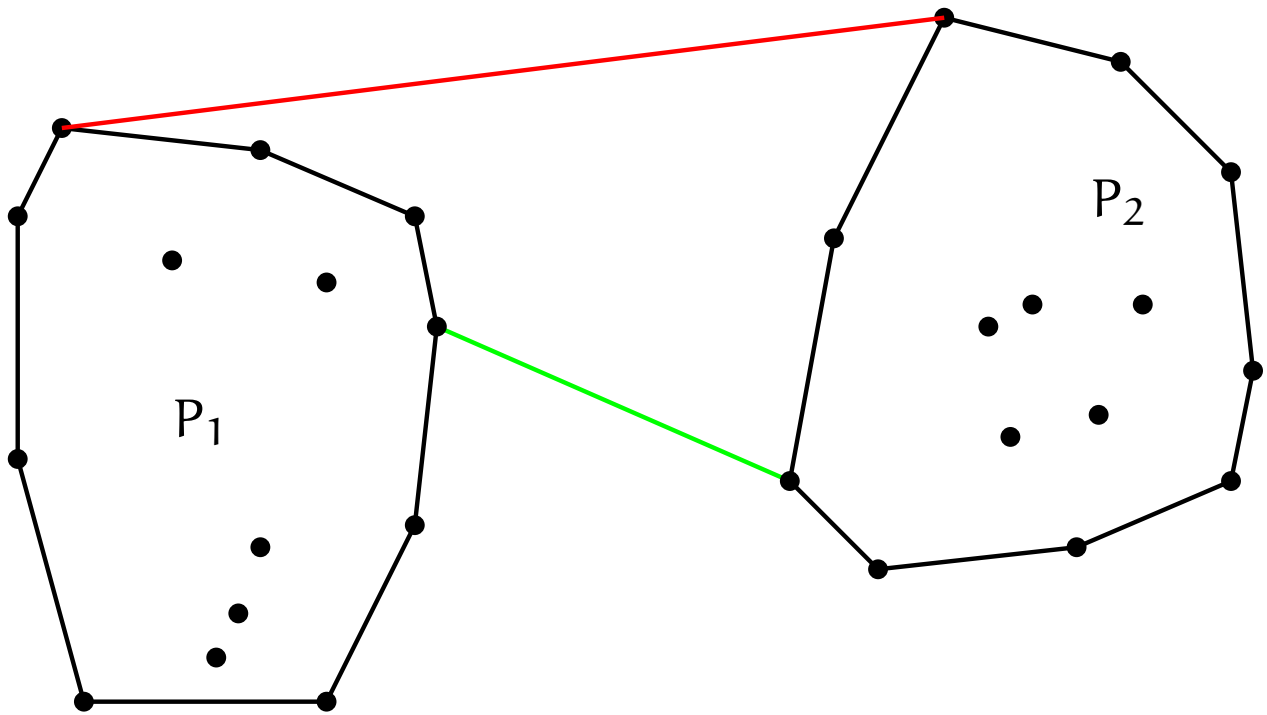
Αλγόριθμος Διαίρει και Βασίλευε για ΚΠ2

Παράδειγμα



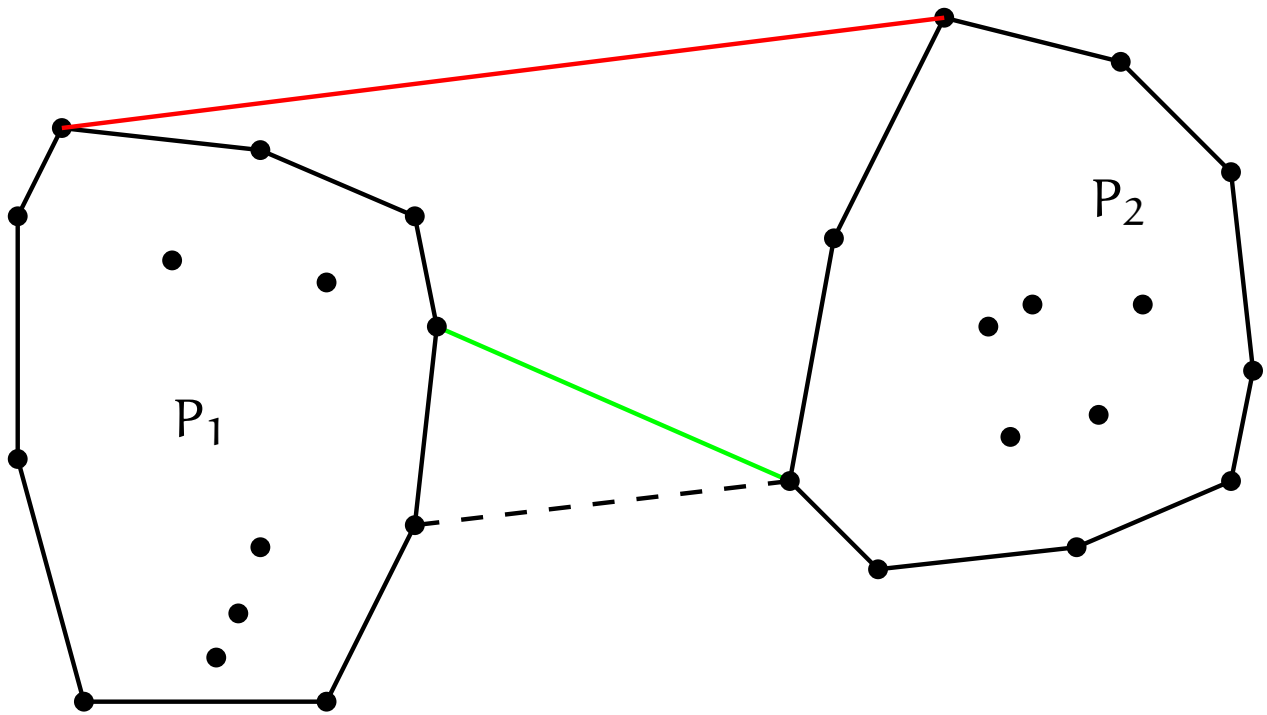
Αλγόριθμος Διαίρει και Βασίλευε για ΚΠ2

Παράδειγμα



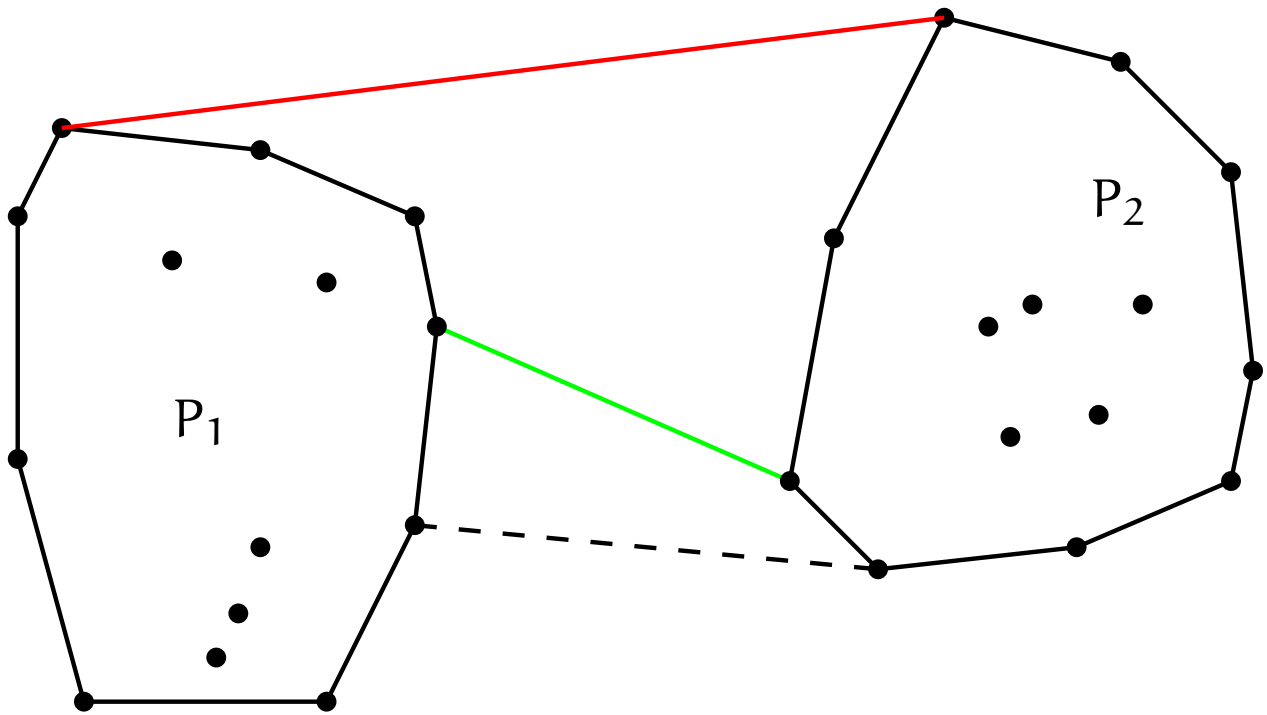
Αλγόριθμος Διαίρει και Βασίλευε για ΚΠ2

Παράδειγμα



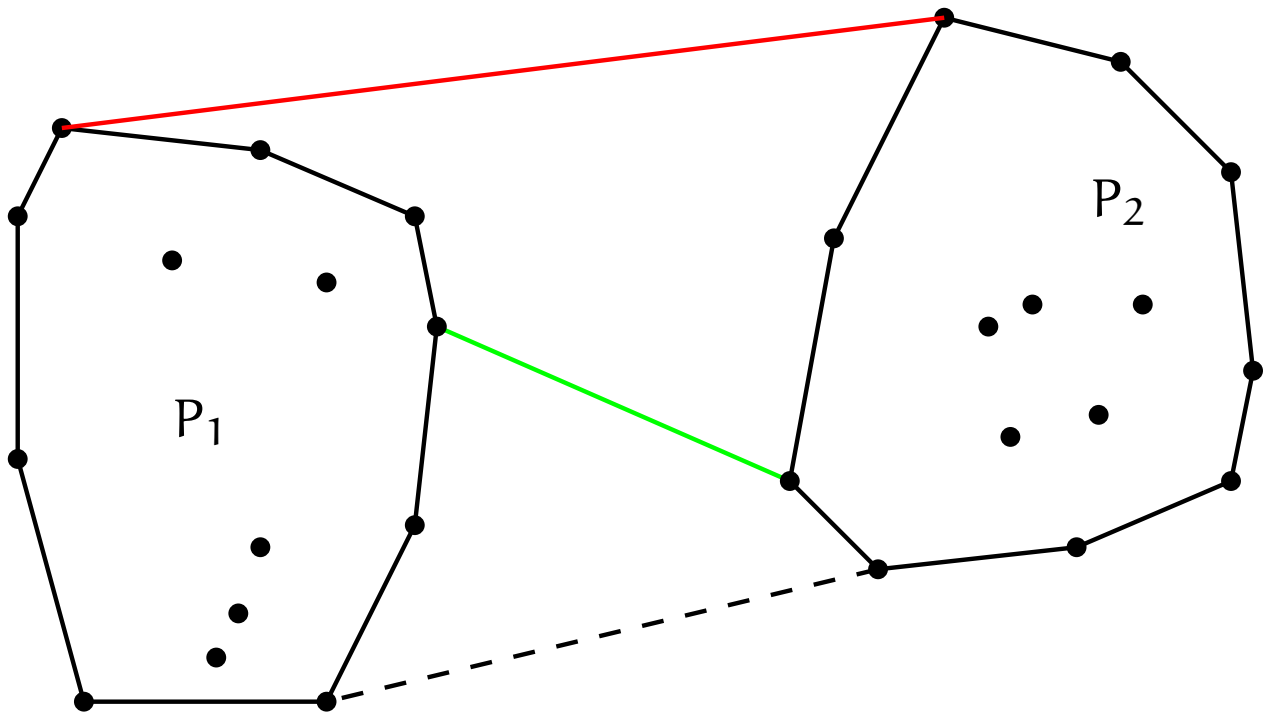
Αλγόριθμος Διαίρει και Βασίλευε για ΚΠ2

Παράδειγμα



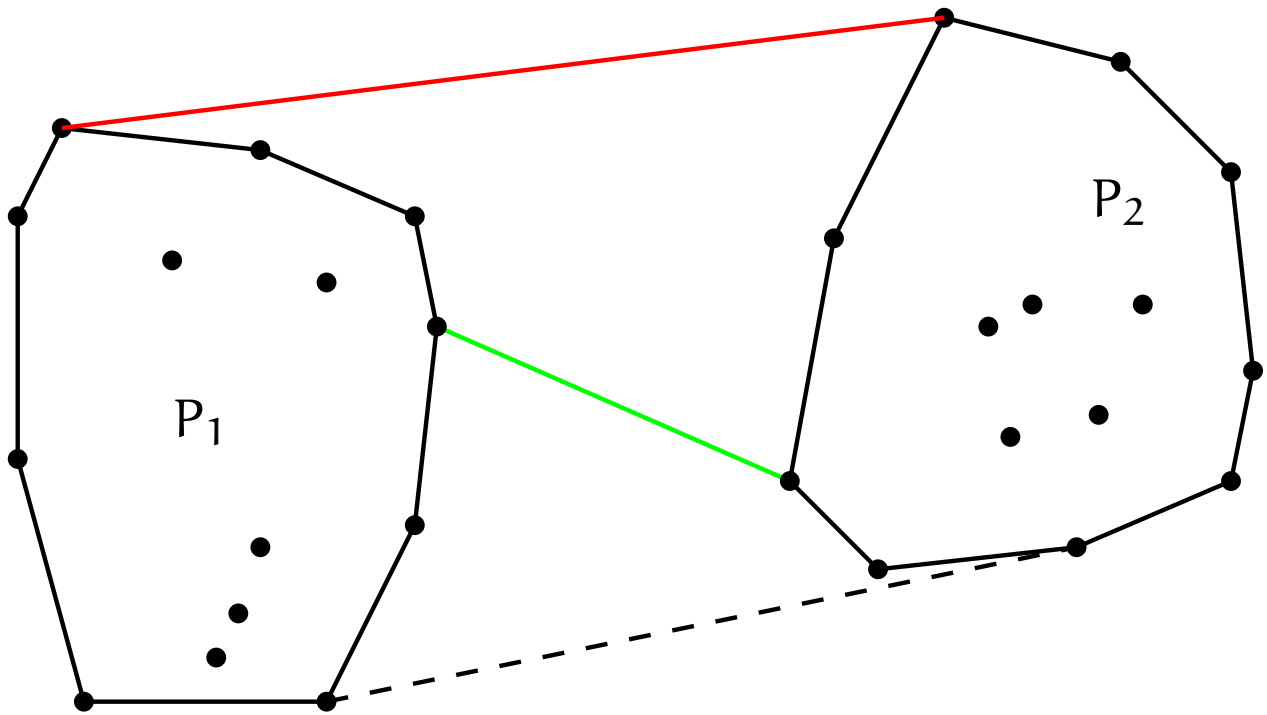
Αλγόριθμος Διαίρει και Βασίλευε για ΚΠ2

Παράδειγμα



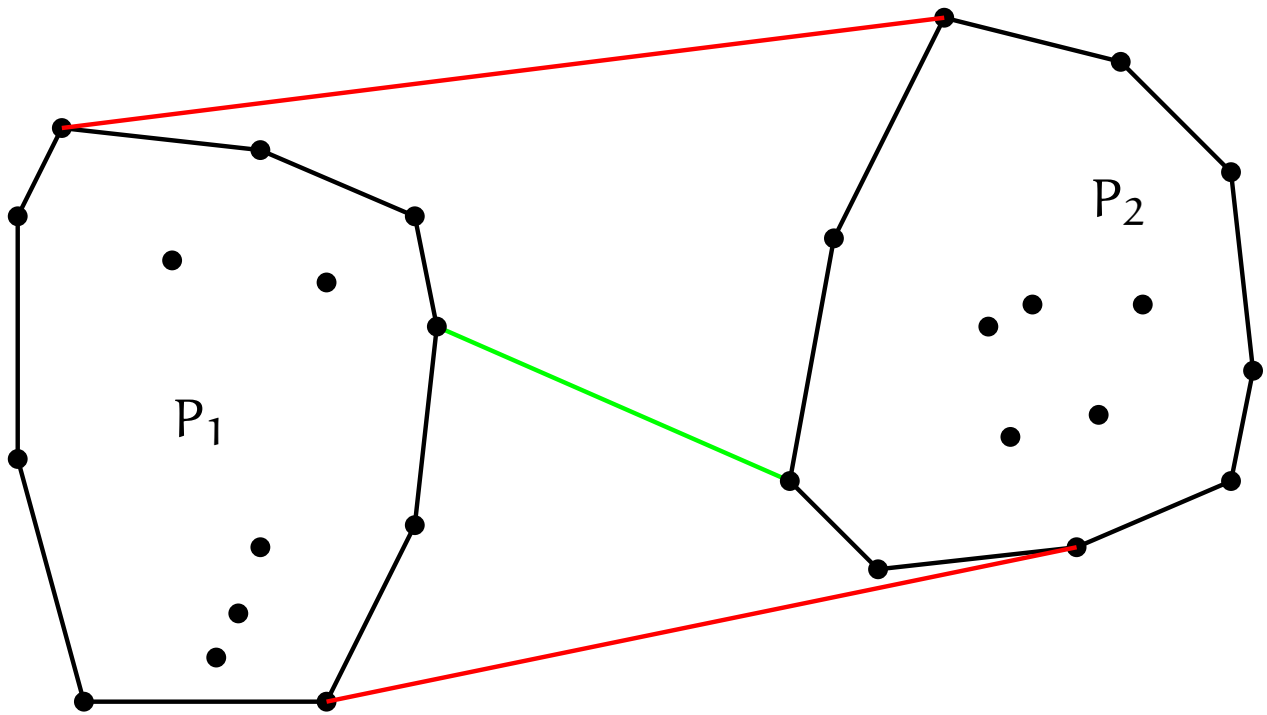
Αλγόριθμος Διαίρει και Βασίλευε για ΚΠ2

Παράδειγμα



Αλγόριθμος Διαίρει και Βασίλευε για ΚΠ2

Παράδειγμα



Αλγόριθμος Διαίρει και Βασίλευε για ΚΠ2

Υλοποίηση της σύνθεσης των υποπροβλημάτων στην Python

```
def find_upper_bridge(polygon1, polygon2):
  i = polygon1.index(max(polygon1.vertices))
  j = polygon2.index(min(polygon2.vertices))
  i_changed = j_changed = False
  while not i_changed and not j_changed:
    if not ccw(polygon1[i], polygon1[i+1], polygon2[j]):
      i = i+1
      i_changed = True
    else:
      i_changed = False
    if not cw(polygon2[j], polygon2[j-1], polygon1[i]):
      j = j-1
      j_changed = True
    else:
      j_changed = False
  return Segment2(polygon1[i], polygon2[j])
```