Πολυδιάστατες Δομές Δεδομένων

Ερώτημα:

4)**Line Segment Intersection:** Υλοποίηση αλγορίθμων εύρεσης τομών μεταξύ ευθυγράμμων τμημάτων στο επίπεδο που προκύπτουν από τα «σύνορα» πολυγωνικών περιοχών π.χ. σε εφαρμογές υπέρθεσης χαρτών στα GIS κ.τ.λ..

Υλοποίηση:

Η υλοποίηση του αλγορίθμου έγινε σε γλώσσα python και χρησιμοποιήθηκαν οι εξής βιβλιοθήκες:

pandas, matplotlib , geopandas, shapely, nympy,time. Για την δοκιμή χρησιμοποίησα δεδομένα από τον σύνδεσμο https://openflights.org/data.html. Τα δεδομένα απεικονίζουν πτήσεις που έχουν γίνει με σημεία έναρξης και τερματισμού, τα οποία και αξιοποιώ στον αλγόριθμο.

Γενική Περιγραφή:

Αρχικά διαχειριστηκα τα δεδομένα ως csv αρχείο μέσω του pandas.read_csv(). Για τις ευθείες χρησιμοποίησα την κλάση Linestring και Point από την βιβλιοθήκη geometry της shapely. Στην συνέχεια εφτιαξα στο dataframe νέα columns με points για τα σημεία έναρξης και τερματισμού των ευθειών. Ακόμη υλοποίησα τρεις συναρτήσεις για την εύρεση τομής. Τέλος για κάθε γραμμή του dataframe έτρεξα την συνάρτηση doIntersect όπου συνδιάζει τις άλλες 2 και με την βοήθεια του intersection, μίας λειτουργίας του geometry βρήκα τα σημεία που τέμνονται οι ευθείες και τα εκτύπωσα.

Περιγραφή Συναρτήσεων:

1η onSegment

```
def onSegment(p, q, r):
    if ( (q.x <= max(p.x, r.x)) and (q.x >= min(p.x, r.x)) and
           (q.y <= max(p.y, r.y)) and (q.y >= min(p.y, r.y))):
        return True
    return False
```

Αυτή η συνάρτηση δέχεται ως όρισμα 3 σημεία και ελέγχει αν το σημείο q βρίσκεται πάνω στην ευθεία που σχηματίζουν τα άλλα δύο, δηλαδή ελέχγει αν υπάρχει τομή. Αν ναι επιστρέφει true, αν όχι επιστρέφει false.

2η orientation

```
def orientation(p, q, r):
    val = (float(q.y - p.y) * (r.x - q.x)) - (float(q.x - p.x) * (r.y - q.y))
    if (val > 0):

        # Clockwise orientation
        return 1
    elif (val < 0):

        # Counterclockwise orientation
        return 2
    else:

        # Collinear orientation
        return 0
```

Συμπληρωματικά με την προηγούμενη συνάρτηση, αυτή βρίσκει την φορά των ευθειών. Αν είναι με την φορά του ρολογιού επιστρέφει 1, αν είναι αντίθετα επιστρέφει 2 και αν είναι συνευθειακά επιστρέφει 0.

3η doIntersect

```
def doIntersect(p1,q1,p2,q2):

    # Find the 4 orientations required for
    # the general and special cases
    o1 = orientation(p1, q1, p2)
    o2 = orientation(p1, q1, q2)
    o3 = orientation(p2, q2, p1)
    o4 = orientation(p2, q2, q1)

    # General case
    if ((o1 != o2) and (o3 != o4)):
        return True

    # Special Cases

    # p1 , q1 and p2 are collinear and p2 lies on segment p1q1
    if ((o1 == 0) and onSegment(p1, p2, q1)):
        return True

    # p1 , q1 and q2 are collinear and q2 lies on segment p1q1
    if ((o2 == 0) and onSegment(p1, q2, q1)):
        return True

    # p2 , q2 and p1 are collinear and p1 lies on segment p2q2
    if ((o3 == 0) and onSegment(p2, p1, q2)):
        return True

    # p2 , q2 and q1 are collinear and q1 lies on segment p2q2
    if ((o4 == 0) and onSegment(p2, q1, q2)):
        return True

    # If none of the cases
    return False
```

Η συνάρτηση συνδιάζει τις προηγούμενες δύο. Παίρνει την περίπτωση τομής αλλά διαφορετικής φοράς και επιστρέφει αληθές και μετά ελέγχω διαφορετικές περιπτώσεις ανάλογα με τους συνδιασμούς που μπορούν να προκύψουν για σημεία 2 ευθειών

Εφαρμογή:

Στο τέλος εφαρμόζω την συνάρτηση doIntersect με δύο δομές επανάληψης και να η κάθε ευθεία με την επόμενη της έχουν τομή τότε βρίσκω το σημείο τομής με την intersection της shapely. Για την απεικόνηση έφτιαξα δύο πίνακες, για στοιχεία x και y. Αφού τελειώσει αυτή η δομή με την matplotlib εμφανίζω τα δεδομένα σε ένα γράφημα

```
x= []
y= []
for index, row1 in flights.iterrows():
    for j, row2 in flights.iterrows():
        if doIntersect(row1['start_points'],row1['end_points'] , row2['start_points'], row2['end_points']):
                print("Yes")
                res = row1['geometry'].intersection(row2['geometry'])
                x.append(res.x)
                y.append(res.y)
                # Add first scatter trace with medium sized markers
        else:
                print("No intersection")

ax.scatter(x,y)
plt.show()
```

Πολυπλοκότητα:

Η πολυπλοκότητα του αλγορίθμου προκύπτει από τις επαναλύψεις και στην συγκεκριμένη περίπτωση η μεγαλύτερη καθυστέρηση προκύτει από τις δύο for που τρέχουν για κάθε ένα σημείο αυξάνοντας την πολυπλοκότητα σε O(n^2). Επομένως ο αλγόριθμος δεν είναι ο βέλτιστος για αυτή την διαδικασία

```
Execution time: 19.283034324645996 seconds
```