**Πολυδιάστατες Δομές**

**Αναφορά-Παραδοχές**

Convex Hull

**Γενική επεξήγηση αλγορίθμου:**

Ο αλγόριθμος που επιλέχθηκε για την εύρεση του convex hull με δισδιάστατα σημεία είναι ο Jarvis Μarch. Ο αλγόριθμος αυτός επιλέγει το σημείο με το μικρότερο x(starting point = sp), δηλαδή το αριστερότερο σημείο στο δισδιάστατο χώρο που είναι σίγουρο ότι θα περιέχεται στο convex hull. Στην συνέχεια παίρνει το επόμενο σημείο(next\_p) -από τον πίνακα που περιέχει όλα τα σημεία για τα οποία γίνεται έλεγχος- ως υποψήφιο σημείο για το convex hull και ελέγχει εάν τα διανύσματα (j, sp) και (j, next\_p) (όπου j οποιοδήποτε άλλο σημείο στον χώρο) δημιουργούν counterclockwise ή clockwise στροφή. Αν δημιουργούν counterclockwise στροφή τότε αντικαθιστά το next\_p με το j διότι το j βρίσκεται πιο αριστερά από το next\_p συνεπώς έχει μεγαλύτερη πιθανότητα να περιέχεται στο convex hull. Αν δημιουργούν clockwise στροφή τότε απλά αλλάζει το j και ελέγχει ξανά. Επαναλαμβάνει τα προηγούμενα βήματα μέχρι να φτάσει στο αρχικό σημείο(sp) όπου και έχει βρει όλα τα σημεία που περιλαμβάνονται στο convex hull.

**Επεξήγηση κώδικα:**

Ορίζονται δύο λίστες οι points\_2d και hull που η 1η θα χρησιμοποιηθεί για να αποθηκευτούν τα σημεία του dataset και η 2η για να αποθηκευτούν τα σημεία που αποτελούν τo convex hull(γραμμές 4,5).

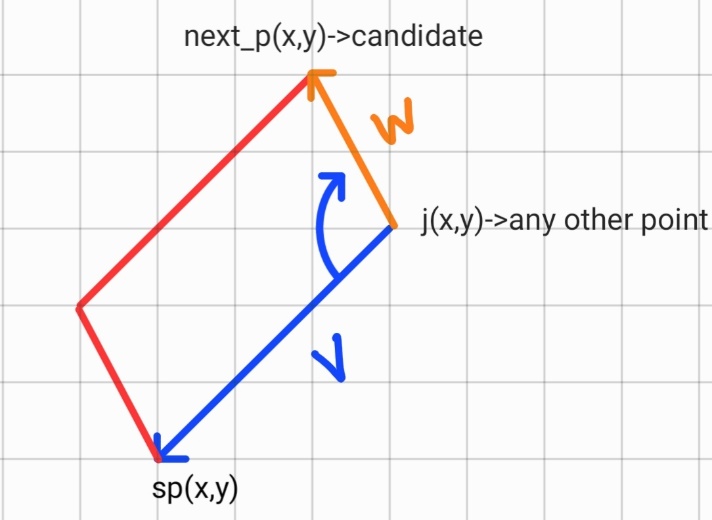
**Ccw function:** Παραπάνω αναφέρθηκε πως αλγόριθμος ελέγχει εάν τα διανύσματα V = (j, sp) = (spx – jx, spy - jy) και W = (j, next\_p) = (next\_px – jx, next\_py - jy) δημιουργούν counterclockwise ή clockwise στροφή, το οποίο ελέγχεται με την βοήθεια της ccw συνάρτησης. Η συνάρτηση παίρνει ως είσοδο τα τρία σημεία j, sp, next\_p, υπολογίζει το εξωτερικό τους γινόμενο(γραμμή 9) και αν είναι θετικό(counterclockwise) επιστρέφει 1, αν είναι αρνητικό(clockwise) επιστρέφει -1 και αν είναι 0(συγγραμμικά) τότε επιστρέφει 0.

Το εξωτερικό γινόμενο των διανυσμάτων V x W θα έχει ως αποτέλεσμα το εμβαδό του παραλληλογράμμου που φαίνεται στην παρακάτω εικόνα, το

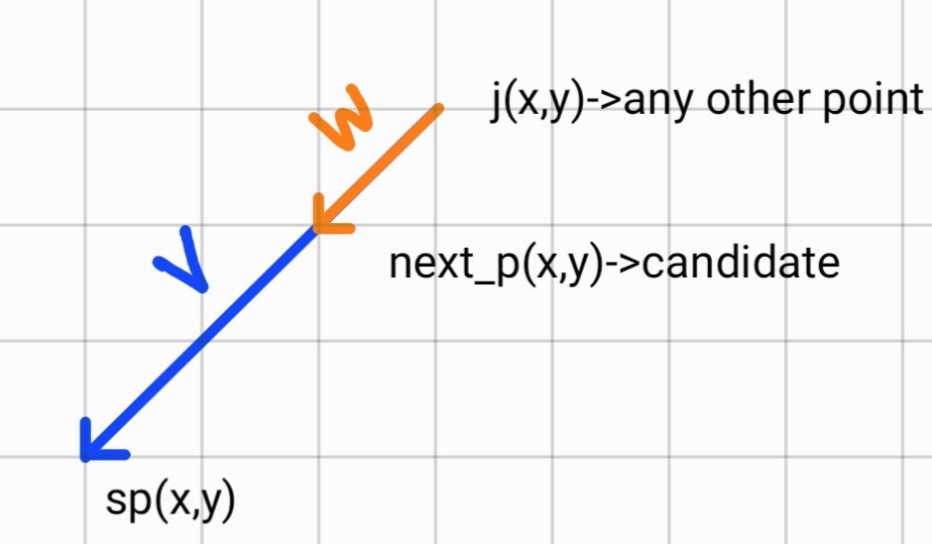


οποίο δίνεται από τον τύπο det(). Όμως επειδή τα σημεία βρίσκονται στον δισδιάστατο χώρο, δεν έχουν 3η διάσταση συνεπώς ο τύπος γίνεται det() το οποίο ισούται με Vx \* Wy – Vy \* Wx = (spx – jx) \* (next\_py - jy) – (spy - jy) \* (next\_px – jx). Το εμβαδόν του παραλληλογράμμου είναι θετικό για οποιεσδήποτε τιμές και συνεπώς η στροφή που δημιουργούν τα 2 διανύσματα θα είναι counterclockwise όπως φαίνεται και στην εικόνα.

Αντιθέτως αν στο εξωτερικό γινόμενο των διανυσμάτων (-)W \* V που φαίνεται στην παρακάτω εικόνα εφαρμοστεί ο ίδιος τύπος με πριν το αποτέλεσμα θα είναι αρνητικό, άρα το εμβαδόν του παραλληλογράμμου θα είναι αρνητικό και συνεπώς η στροφή θα είναι αντίθετη από πριν, δηλαδή clockwise.



Τέλος αν τα σημεία είναι συγγραμμικά τότε δεν υπάρχει γωνία μεταξύ τους και συνεπώς δεν δημιουργείται κάποιου είδους στροφή. Ο τρόπος που θα αντιμετωπιστεί αυτό το θέμα θα αναλυθεί στην περιγραφή του αλγορίθμου([Jarvis march](#other))



**starting\_point function:**

Αυτή η συνάρτηση χρησιμοποιείται για να πάρει τα δεδομένα από το txt file που περιέχει το dataset καθώς και να βρει και να επιστρέψει το αρχικό σημείο(sp). Αρχικά ανοίγει τον φάκελο που βρίσκεται το dataset(γραμμή 46), διαβάζει τον φάκελο γραμμή προς γραμμή και τα στοιχεία που διαβάζει τα χωρίζει με βάση το whitespace(γραμμή 47). Στην συνέχεια ελέγχει εάν το πλήθος των στοιχείων αυτών είναι πολλαπλάσιο του 2(\*)(γραμμή 48) και αν είναι, μπαίνει στο if statement και εκτελείται μία for loop όπου σε κάθε επανάληψη παίρνει δύο από τα στοιχεία στο txt\_split και τα βάζει ως ένα στοιχείο στην points\_2d(γραμμές 49,50). Συνεπώς έτσι δημιουργούνται τα δισδιάστατα σημεία που θα χρησιμοποιήσει ο αλγόριθμος για να βρει το convex hull. \*(διότι αν είναι μονός αριθμός τότε στην τελευταία επανάληψη η for loop θα προσπαθήσει να πάρει το επόμενο στοιχείο από το τελευταίο που δεν υφίσταται και συνεπώς θα υπάρξει το error: list index out of range).

Όσον αφορά στο αρχικό σημείο, έχει οριστεί η μεταβλητή starting\_min η οποία εξαρχής έχει μία τυχαία τιμή. Με ένα for loop(γραμμές 51-54) συλλέγονται όλες οι x συντεταγμένες των δισδιάστατων σημείων που δημιουργήθηκαν και συγκρίνονται με την starting\_min. Αν κάποια συντεταγμένη x είναι μικρότερη σε τιμή από την starting\_min τότε αναθέτουμε την τιμή της ως καινούργια starting\_min. Επαναλαμβάνουμε αυτήν την διαδικασία μέχρι η starting\_min να έχει ως τιμή την συντεταγμένη x του σημείου που βρίσκεται πιο αριστερά στον δισδιάστατο χώρο. Όμως επειδή συγκρίνονται x συντεταγμένες ενώ χρειάζεται να βρεθεί σημείο, αποθηκεύεται ο δείκτης του σημείου με την μικρότερη x συντεταγμένη, ο οποίος χρησιμοποιείται για να βρεθεί το σημείο. Στην συνέχεια αποθηκεύετε στην μεταβλητή st\_p την οποία και επιστρέφει η συνάρτηση.

**Jarvis\_march function:**

Αρχικά ορίζονται οι μεταβλητές, k που θα χρησιμοποιηθεί ως δείκτης για τις επαναλήψεις της while loop, η sp\_index που περιέχει τον δείκτη του sp, η next\_p\_index που περιέχει τον δείκτη του επόμενου στοιχείου(next\_p) σε κάθε επανάληψη. Στην συνέχεια προστίθεται το sp στην hull αφού είναι σίγουρο πως περιέχεται στο convex hull και έπειτα εκτελείται η while loop.

Μέσα σε αυτήν υπάρχουν αρχικά δύο έλεγχοι. Το 1ο if statement(γραμμή 24) ελέγχει εάν η while loop βρίσκεται στην 1η επανάληψη και αν ναι, θέτει ως υποψήφιο σημείο(γραμμή 25) για το convex hull το επόμενο σημείο από το sp στην points\_2d. Το 2ο if statement ελέγχει εάν η while loop βρίσκεται σε οποιαδήποτε άλλη επανάληψη (γραμμή 26) πέρα από την 1η και αν ναι, θέτει ως υποψήφιο σημείο(γραμμή 27) το επόμενο από το next\_p στην Points\_2d. Έπειτα μία for loop διαπερνάει όλα τα σημεία στην points\_2d και ελέγχει για κάθε σημείο(έστω j) αν είναι ίδιο με το sp(γραμμή 30) και αν ναι τότε επαναφέρει την λούπα στην επόμενη επανάληψη με το continue. Αν το j δεν είναι ίδιο με το sp τότε ελέγχει με την βοήθεια της [ccw()](#ccw) (γραμμή 33) αν το εξωτερικό γινόμενο των σημείων sp, j και next\_p είναι ίσο με 1([counterclockwise](#cclock)). Αν είναι, αυτό σημαίνει ότι το σημείο j είναι πιο εξωτερικό σημείο και βρίσκεται πιο αριστερά σε σχέση με το next\_p και συνεπώς έχει μεγαλύτερη πιθανότητα να βρίσκεται στο convex hull, άρα θέτει το next\_p ίσο με το σημείο j και προχωράει στην επόμενη επανάληψη. Αν ωστόσο το αποτέλεσμα της ccw είναι 0(δηλαδή τα σημεία είναι συγγραμμικά) δεν είναι εφικτό να γίνει κάποια ενέργεια με βάση τα κριτήρια που χρησιμοποιήθηκαν μέχρι τώρα. Για να αντιμετωπιστεί αυτό χρησιμοποιείται και ένα τρίτο κριτήριο στο οποίο συγκρίνεται η ευκλείδεια απόσταση του j από το sp και του next\_p από το sp. Αν το j απέχει μεγαλύτερη απόσταση αποτι το next\_p από το sp, αυτό σημαίνει ότι το j είναι πιο εξωτερικό σημείο και συνεπώς έχει μεγαλύτερη πιθανότητα να βρίσκεται στο convex hull, άρα όπως και προηγουμένως θέτει το next\_p ίσο με j και προχωράει στην επόμενη επανάληψη. Όταν η for loop τελειώσει αποθηκεύεται ο δείκτης του next\_p στην next\_p\_index(γραμμή 35) η οποία χρησιμοποιείται για να βρεθεί το επόμενο starting point στην points\_2d(γραμμή 36), που θα χρησιμοποιηθεί στην επόμενη επανάληψη της while loop.

Στο τέλος της while loop ελέγχεται εάν το επόμενο starting point είναι ίδιο με το αρχικό starting point(γραμμές 37,38) και σε περίπτωση που είναι, αυτό σημαίνει ότι έχουν βρεθεί όλα τα σημεία του convex hull και συνεπώς χρησιμοποιείται το brake για να σταματήσει η λούπα. Αντιθέτως αν δεν είναι ίδιο με το αρχικό starting point τότε προστίθεται αυτό το σημείο στο convex hull(γραμμή 39) και η while loop προχωράει στην επόμενη επανάληψη.

**Παραδοχές**

Το dataset των δισδιάστατων σημείων που χρησιμοποιήθηκε παράγεται από το πρόγραμμα dataset\_generator το οποίο τυπώνει σε μία γραμμή αριθμούς κινητής υποδιαστολής με μέγιστο τα 3 ψηφία μετά την υποδιαστολή χωρισμένους με whitespace.

Πηγές:

<https://www.youtube.com/watch?v=B2AJoQSZf4M&t=300s&ab_channel=StableSort>

<https://www.youtube.com/watch?v=YbZmAqGUkqc&ab_channel=TheOrganicChemistryTutor>

<https://algorithmtutor.com/Computational-Geometry/Convex-Hull-Algorithms-Jarvis-s-March/>