ΕΦΑΡΜΟΣΜΕΝΗ ΣΥΝΔΥΑΣΤΙΚΗ

ΕΛΕΥΘΕΡΙΑ ΕΤΖΑΝΙΔΗ – Π22043

ΑΓΓΕΛΟΣ ΙΩΑΝΝΙΔΗΣ – Π22052

KNAPSACK

**Class Node**

A screen shot of a computer program

Description automatically generated

Η κλάση node αντιπροσωπεύει τους κόμβους του δέντρου. Κάθε κόμβος/ αντικείμενο της κλάσης θα έχει 5 ιδιότητες: 1) αξία 2) βάρος 2) δεξί παιδί 3) αριστερό παιδί 4) γονιό 5) την λίστα με τα αντικείμενα που θέλουμε να μπούν στο knapsack 5) την λίστα με τα αντικείμενα που βρίσκονται μέσα στο knapsack. Θα παίρνει ως παραμέτρους την αξία, το βάρος, την λίστα με τα αντικείμενα που δεν έχουν μπεί ακόμα, και τον γονιό

**Class object**

A computer screen shot of a code

Description automatically generated

Η κλάση object αντιπροσωπεύει τα αντικείμενα που θέλουμε να βάλουμε στο knapsack. Κάθε αντικείμενο της κλάσης θα έχει 4 ιδιότητες: 1) όνομα 2) αξία του αντικειμένου 3) βάρος του αντικειμένου 4) το ratio αξία/βάρος που θα μας βοηθήσει να αποφασίσουμε ποια αντικείμενα είναι πιο πολύτιμα.

**Συνάρτηση ratio(A)**

A blue background with yellow and blue text

Description automatically generated

H συνάρτηση ratio παίρνει ως παράμετρο ένα αντικείμενο (A) και μας επιστρέφει το ratio (r) του αντικειμένου.

**Συνάρτηση Knapsack(obj\_list, currentNode, root, C, m\_n)**

A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A blue and white text on a black background

Description automatically generated

Η συνάρτηση knapsack εκτελεί τον αλγόριθμο backtracking για την επίλυση του knapsack και θα παίρνει ως παραμέτρους:

1. τη λίστα με τα αντικείμενα που θέλουμε να βάλουμε στο knapsack (obj\_list)
2. σε ποιο κόμβο του δέντρου βρίσκεται ο αλγόριθμος (currentNode)
3. τη ρίζα του δέντρου
4. τη δεδομένη χωρητικότητα του knapsack σε βάρος (C)
5. τον βέλτιστο κόμβο

Πρώτα ελέγχει αν βρίσκεται στη ρίζα και ότι υπάρχει αριστερό παιδί για να τερματίσει το knapsack, επειδή τότε δεν χρειάζεται να συνεχίσουμε δεξιά της ρίζας. Αν όχι τότε συνεχίζει.

Μετά ελέγχει ότι υπάρχουν αντικείμενα για να μπουν στο knapsack. Αν υπάρχουν τότε συνεχίζει και ελέγχει ότι υπάρχει χώρος στο knapsack στον κόμβο που βρίσκεται ο αλγόριθμος.

Έπειτα ελέγχει αν ο κόμβος έχει αριστερό παιδί. Εάν δεν έχει, δημιουργεί αριστερό παιδί για αυτόν τον κόμβο προσθέτοντας το v και w του αντικειμένου που θα μπει στο knapsack. Αλλιώς δημιουργεί δεξί παιδί για τον κόμβο με τα ίδια v, w, δηλαδή δεν προσθέτει αντικείμενο.

Εφόσον δεν έχει υπερβεί το C, κάνει update το m\_v αν βρεθεί μεγαλύτερο .

Στην περίπτωση που έχει γεμίσει το knapsack, με μια while loop κάνει backtrack στο δέντρο μέχρι να βρει κόμβο που δεν έχει δεξί παιδί για να ξανακαλέσει το knapsack.

Στην περίπτωση που δεν υπάρχουν άλλα αντικείμενα για να μπουν στο knapsack με μια while loop κάνει backtrack στο δέντρο μέχρι να βρει κόμβο που να μην έχει δεξί παιδί για να ξανακαλέσει το knapsack.

Ξανακαλεί το knapsack κάθε φορά με ενημερωμένη λίστα, μέχρι ο αλγόριθμος με backtracking να φτάσει ξανά στη ρίζα και να έχει αριστερό παιδί ώστε να τελειώσει το knapsack.

**Συνάρτηση main**

A computer screen shot of a code

Description automatically generated

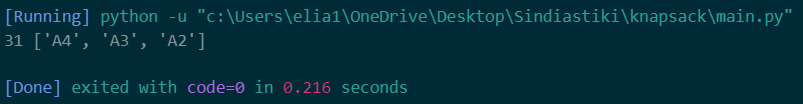
Στην main αρχικοποιούμε τα 4 αντικείμενα μας με τις τιμές τους χρησιμοποιώντας την κλάση object. Επίσης αρχικοποιούμε την λίστα με τα αντικείμενα που θέλουμε να βάλουμε στο knapsack.

Ταξινομούμε την λίστα από το πιο πολύτιμο αντικείμενο στο λιγότερο πολύτιμο αντικείμενο.

Αρχικοποιούμε την ρίζα, το currentNode, το C, το global max\_node.

Καλούμε την συνάρτηση knapsack που θα αποθηκευτεί στο max\_node που θα είναι και το αποτέλεσμα μας.

ΕΚΤΕΛΕΣΗ:



Το max\_node ήταν το 31 και μέσα στο knapsack μπήκαν τα αντικείμενα Α4, Α3, Α2

­­­

ΨΕΥΔΟΚΩΔΙΚΑΣ:

ΚΛΑΣΗ Κόμβος

ΚΟΝΣΤΡΑΚΤΟΡΑΣ (όγκος, βάρος, λίστα, γονιός = Τιποτα)

v = όγκος

w = βάρος

right = Τιποτα

left = Τιποτα

parent = γονιός

node\_object\_list = λίστα

node\_objects\_saved = κενή λίστα

ΚΛΑΣΗ Αντικείμενο

ΚΟΝΣΤΡΑΚΤΟΡΑΣ \_\_init\_\_(όνομα, αξία, βάρος)

name = όνομα

v = αξία

w = βάρος

r = αξία / βάρος

ΣΥΝΑΡΤΗΣΗ ratio(Αντικείμενο)

ΕΠΕΣΤΡΕΨΕ r του Αντικείμενου

ΣΥΝΑΡΤΗΣΗ σακίδιο(λίστα\_αντικειμένων, τρέχων\_Κόμβος, ρίζα, Χωρητικότητα, μέγιστος\_κόμβος)

ΑΝ τρέχων\_Κόμβος.left != τίποτα ΚΑΙ τρέχων\_Κόμβος == ρίζα ΤΟΤΕ

ΕΠΕΣΤΡΕΨΕ μέγιστος\_κόμβος

ΑΝ λίστα\_αντικειμένων ΔΕΝ ΕΙΝΑΙ άδεια ΤΟΤΕ

ΑΝ τρέχων\_Κόμβος.w < Χωρητικότητα ΤΟΤΕ

ΑΝ τρέχων\_Κόμβος.left == τίποτα ΤΟΤΕ

νέος\_Κόμβος = Νέος Κόμβος(λίστα\_αντικειμένων[0].v + τρέχων\_Κόμβος.v, λίστα\_αντικειμένων[0].w + τρέχων\_Κόμβος.w, λίστα\_αντικειμένων[1:], τρέχων\_Κόμβος)

νέος\_Κόμβος.node\_objects\_saved = Αντίγραφο του τρέχων\_Κόμβος.node\_objects\_saved

ΠΡΟΣΘΕΣΕ λίστα\_αντικειμένων[0] στο νέος\_Κόμβος.node\_objects\_saved

τρέχων\_Κόμβος.left = νέος\_Κόμβος

τρέχων\_Κόμβος = νέος\_Κόμβος

ΑΛΛΙΩΣ

νέος\_Κόμβος = Νέος Κόμβος(τρέχων\_Κόμβος.v, τρέχων\_Κόμβος.w, λίστα\_αντικειμένων[1:], τρέχων\_Κόμβος)

νέος\_Κόμβος.node\_objects\_saved = Αντίγραφο του τρέχων\_Κόμβος.node\_objects\_saved

τρέχων\_Κόμβος.right = νέος\_Κόμβος

τρέχων\_Κόμβος = νέος\_Κόμβος

ΑΝ τρέχων\_Κόμβος.v > μέγιστος\_κόμβος.v ΚΑΙ τρέχων\_Κόμβος.w < Χωρητικότητα ΤΟΤΕ

μέγιστος\_κόμβος = τρέχων\_Κόμβος

ΑΛΛΙΩΣ

τρέχων\_Κόμβος = τρέχων\_Κόμβος.parent

ΟΣΟ τρέχων\_Κόμβος.right != τίποτα ΚΑΙ τρέχων\_Κόμβος.left != τίποτα

τρέχων\_Κόμβος = τρέχων\_Κόμβος.parent

ΑΛΛΙΩΣ

τρέχων\_Κόμβος = τρέχων\_Κόμβος.parent

ΟΣΟ τρέχων\_Κόμβος.right != τίποτα ΚΑΙ τρέχων\_Κόμβος.left != τίποτα

τρέχων\_Κόμβος = τρέχων\_Κόμβος.parent

αποτελέσματα = σακίδιο(λίστα του τρέχων\_Κόμβος.node\_object\_list, τρέχων\_Κόμβος, ρίζα, Χωρητικότητα, μέγιστος\_κόμβος)

ΕΠΕΣΤΡΕΨΕ αποτελέσματα

ΔΙΑΔΙΚΑΣΙΑ main

Α1 = Νέο Αντικείμενο("Α1", 8, 2)

Α2 = Νέο Αντικείμενο("Α2", 12, 6)

Α3 = Νέο Αντικείμενο("Α3", 9, 3)

Α4 = Νέο Αντικείμενο("Α4", 10, 2)

λίστα\_αντικειμένων = [Α1, Α2, Α3, Α4]

ταξινόμησε λίστα\_αντικειμένων με βάση το ratio, κατά φθίνουσα σειρά

ρίζα = Νέος Κόμβος(0, 0, λίστα\_αντικειμένων)

τρέχων\_Κόμβος = ρίζα

Χωρητικότητα = 12

μέγιστος\_κόμβος = ρίζα

μέγιστος\_κόμβος = σακίδιο(λίστα\_αντικειμένων, τρέχων\_Κόμβος, ρίζα, Χωρητικότητα, μέγιστος\_κόμβος)

ΓΡΑΨΕ μέγιστος\_κόμβος.v, [όνομα για κάθε αντικείμενο σε μέγιστος\_κόμβος.node\_objects\_saved]

NQUEENS

A screen shot of a computer

Description automatically generated

Εισαγωγή βιβλιοθήκης numpy για αριθμητικούς υπολογισμούς.

Ορισμός παγκόσμιας μεταβλητής Ν που έπειτα παίρνει τιμή 8 και αντιπροσωπεύει το μέγεθος της σκακιέρας.

**Συνάρτηση printBoard(board)**

A screen shot of a computer program

Description automatically generated

Η συνάρτηση αυτή χρησιμοποιείται για να εμφανίζει την σκακιέρα στην οθόνη με εμφωλευμένη for loop. Τα 1 εμφανίζονται ως Q και τα 0 ως «.».

Παίρνει ως παράμετρο την σκακιέρα

**Συνάρτηση check(board, row, col)**

A screen shot of a computer code

Description automatically generated

Αυτή η συνάρτηση ελέγχει αν μπορεί να τοποθετηθεί queen σε ένα συγκεκριμένο σημείο του πίνακα. Παίρνει ως παραμέτρους την σκακιέρα, την σειρά στην οποία βρίσκεται ο αλγόριθμος, και το col, το οποίο αντιπροσωπεύει σε ποια στήλη βρίσκεται εκείνη τη στιγμή ο αλγόριθμος, αλλά και πόσα queens έχουν τοποθετηθεί.

Για να επιστρέψει true, δηλαδή για να μπορέσει να τοποθετηθεί queen θα πρέπει η check να ελέγξει ότι δεν υπάρχει άλλο queen στην σειρά, στην πάνω διαγώνιο και στην κάτω διαγώνιο. Το col περιορίζει την check να ελέγχει για queens μόνο αριστερά από την στήλη που βρισκόμαστε, γιατί μόνο από τα αριστερά θα υπάρχουν queens.

**Συνάρτηση step(board, col)**

A screenshot of a computer program

Description automatically generated

Αυτή η συνάρτηση τοποθετεί queens. Παίρνει ως παραμέτρους την σκακιέρα και την στήλη στην οποία βρίσκεται ο αλγόριθμος.

Για να τοποθετήσει queen πρέπει να ελέγξει πρώτα ότι δεν έχουν τοποθετηθεί όλα τα queens. Αν έχουν, επιστρέφει true.

Μετά, σε μια for loop τσεκάρει ποιο είναι η πρώτη θέση της στήλης που μπορεί να τοποθετηθεί queen και την τοποθετεί. Ξανακαλεί τον εαυτό της για την επόμενη στήλη μέχρι να τοποθετήσει queen και στην τελευταία στήλη. Αν αποτύχει να τοποθετήσει στην τελευταία στήλη, ακυρώνει το queen που είχε βάλει στην αρχή και το τοποθετεί σε μια άλλη θέση, μέχρις ότου να βρει μια θέση στην current στήλη, η οποία θα της επιτρέψει να τοποθετήσει queen και στην τελευταία στήλη. Αφαιρεί τα queens που είχε τοποθετήσει βάζοντας 0.

Αν δεν βρει καμία λύση επιστρέφει false.

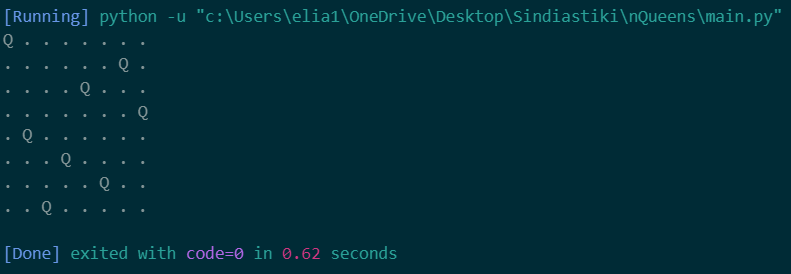
**Συνάρτηση nQueens()**

A computer screen shot of text

Description automatically generated

Αυτή η συνάρτηση πρώτα γεμίζει την σκακιέρα με μηδενικά. Μετά καλεί την step σε ένα if statement. Αν είναι true εμφανίζει το αποτέλεσμα και επιστρέφει true. Αλλιώς εμφανίζει ότι δεν υπάρχει αποτέλεσμα και επιστρέφει false.

ΕΚΤΕΛΕΣΗ



ΨΕΥΔΟΚΩΔΙΚΑΣ

Ορισμός N = 8

Συνάρτηση printBoard(board):

Για κάθε i από 0 μέχρι N-1:

Για κάθε j από 0 μέχρι N-1:

Αν board[i][j] == 1:

Εκτύπωσε "Q"

Αλλιώς:

Εκτύπωσε "."

Εκτύπωσε νέα γραμμή

Συνάρτηση check(board, row, col):

Για κάθε i από 0 μέχρι col-1:

Αν board[row][i] == 1:

Επέστρεψε False

Για κάθε (i, j) από τη zip(range(row, -1, -1), range(col, -1, -1)):

Αν board[i][j] == 1:

Επέστρεψε False

Για κάθε (i, j) από τη zip(range(row, N, 1), range(col, -1, -1)):

Αν board[i][j] == 1:

Επέστρεψε False

Επέστρεψε True

Συνάρτηση step(board, col):

Αν col >= N:

Επέστρεψε True

Για κάθε i από 0 μέχρι N-1:

Αν check(board, i, col):

Τοποθέτησε βασίλισσας στο board[i][col] = 1

step(board, col + 1) και έλεγχος επιτυχίας:

Αν επιτυχής:

Επέστρεψε True

Αλλιώς:

Αφαίρεσε βασίλισσας από το board[i][col] = 0

Επέστρεψε False

Επέστρεψε False

Συνάρτηση nQueens():

Δημιούργησε board με μηδενικά (διάστασης N x N)

Αν step(board, 0) είναι True:

printBoard(board)

Επέστρεψε True

Αλλιώς:

Εκτύπωσε "Solution does not exist"

Επέστρεψε False

Αν name == 'main':

nQueens()

KNIGHT TOUR

**Συνάρτηση print\_board(board)**

A screen shot of a computer program

Description automatically generated

Η συνάρτηση αυτή εκτυπώνει την σκακιέρα. Παίρνει ως παράμετρο την σκακιέρα. Όπου βρίσκει 1 βάζει Κ, όπου 2 βάζει Χ και όπου τίποτα βάζει «.».

**Συνάρτηση check(x, y, board)**

A blue background with yellow and blue text

Description automatically generated

Η συνάρτηση αυτή ελέγχει αν μπορεί να μετακινηθεί ο ιππότης στην θέση x,y που της δίνεται. Παίρνει ως παραμέτρους το x, το y, και την σκακιέρα.

Αν το x και το y είναι μεγαλύτερα ή ίσα του 0 και μικρότερα του n και η θέση x, y είναι άδεια επιστρέφει true.

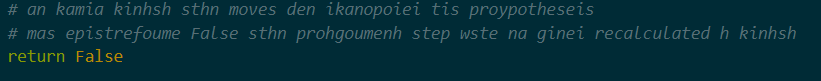
**Συνάρτηση step(board, pos, past\_moves)**

A screenshot of a computer program

Description automatically generated

A computer screen shot of text

Description automatically generated



Αυτή η συνάρτηση κουνάει τον ιππότη. Παίρνει ως παραμέτρους την σκακιέρα και την τωρινή θέση του ιππότη και την λίστα με τις προηγούμενες κινήσεις.

Καλεί την παγκόσμια μεταβλητή s και την αυξάνει κατά 1.

Ελέγχει αν έχουν πραγματοποιηθεί όλες οι κινήσεις για να εμφανίσει το αποτέλεσμα και να επιστρέψει true.

Αν δεν έχουν, συνεχίζει. Προϋπολογίζει τις 8 κινήσεις που μπορεί να κάνει ο ιππότης από την θέση που βρίσκεται και τις αποθηκεύει.

Σε ένα for loop εξετάζει κάθε κίνηση με την check και τοποθετεί τον ιππότη. Προχωράει στις επόμενες κινήσεις καλώντας τον εαυτό της recursively σε ένα if statement, μέχρι να φτάσει στην τελευταία κίνηση. Αν αποτύχει, γυρνάει πίσω και αφαιρεί τις κινήσεις που έκανε.

Αν καμία από τις 8 κινήσεις που είχε αποθηκεύσει δεν επιτρέπει τον αλγόριθμο να συνεχίσει, επιστρέφει false στην προηγούμενη step έτσι ώστε να αφαιρεθεί η προηγούμενη κίνηση που έγινε και να υπολογιστεί η επόμενη κίνηση

Αν εξετάσει όλες τις πιθανές περιπτώσεις και δεν βρει λύση, επιστρέφει false, καθώς δεν υπάρχει λύση.

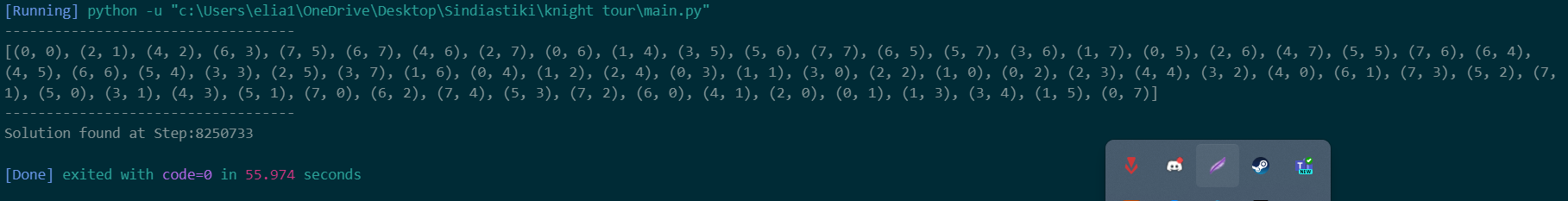
**Συνάρτηση main()**

A screenshot of a computer program

Description automatically generated

Σε αυτή τη συνάρτηση αρχικοποιείται η παγκόσμια μεταβλητή s με 0 και η n με 8 που είναι το μέγεθος της σκακιέρας. Γεμίζει η σκακιέρα με 0 και αρχικοποιουνται τα i, j με 0. Ορίζεται η πρώτη κίνηση και προσθέτεται στην λίστα κινήσεων. Καλείται η συνάρτηση step.

ΕΚΤΕΛΕΣΗ



Εκτυπώνεται η λίστα με κάθε βήμα του ιππότη και πόσα βήματα χρειάστηκαν

ΨΕΥΔΟΚΩΔΙΚΑΣ

Συνάρτηση print\_board(board):

Για κάθε γραμμή στη board:

Για κάθε στήλη στη γραμμή:

Αν col == 1:

Εκτύπωσε "K" (αρχική θέση)

Αλλιώς αν col == 2:

Εκτύπωσε "X" (προηγούμενη κίνηση)

Αλλιώς:

Εκτύπωσε "." (κενό κελί)

Εκτύπωσε νέα γραμμή

Συνάρτηση check(x, y, board):

Επίστρεψε x >= 0 και y >= 0 και x < n και y < n και board[x][y] == 0

Συνάρτηση step(board, pos, past\_moves):

Διεθνής μεταβλητή s

Αύξησε s κατά 1

Αν το μήκος των past\_moves >= n^2:

Εκτύπωσε past\_moves

Εκτύπωσε "Solution found at Step:{}".format(s)

Επίστρεψε True

Ορισμός move\_x και move\_y με τις κινήσεις του ίππου

Ζεύξη των κινήσεων move\_x και move\_y σε move

Για κάθε i από 0 έως 7:

Υπολογισμός νέων συντεταγμένων new\_x και new\_y

Αν check(new\_x, new\_y, board):

Ορισμός board[new\_x][new\_y] = 2

Πρόσθεσε (new\_x, new\_y) στις past\_moves

Αν step(board, (new\_x, new\_y), past\_moves):

Επίστρεψε True

Ορισμός board[new\_x][new\_y] = 0

Αφαίρεσε (new\_x, new\_y) από τις past\_moves

Επίστρεψε False

Κύρια Συνάρτηση main():

Διεθνής μεταβλητή s, n

Ορισμός s = 0

Ορισμός n = 8

Ορισμός board = Πίνακα n\*n με μηδενικά

Ορισμός i,j = 0

Ορισμός board[i][j] = 1

Ορισμός pos = (i, j)

Δημιουργία λίστας past\_moves και προσθήκη της αρχικής θέσης

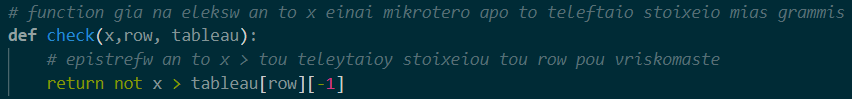
step(board, pos, past\_moves)

Αν \_\_name\_\_ == "\_\_main\_\_":

main()

TABLEAU

**Συνάρτηση check(x, row tableau)**



Αυτή η συνάρτηση επιστρέφει true αν το x που θέλουμε να προσθέσουμε είναι μικρότερο από το τελευταίο στοιχείο/ το μεγαλύτερο στοιχείο της γραμμής που ελέγχει. Παίρνει ως παραμέτρους το στοιχείο x, την σειρά και το ταμπλό.

**Συνάρτηση add(x, row, tableau)**



Η συνάρτηση αυτή εισάγει στοιχεία στο ταμπλό. Παίρνει ως παραμέτρους το χ που θα προστεθεί, την σειρά που βρίσκεται και το ταμπλό. Πρώτα ελέγχει αν η σειρά που του έχει δοθεί, υπάρχει στο ταμπλό για να συνεχίσει.

Καλεί την check σε ένα if statement. Αν true, σημαίνει ότι το χ θα μπει ενδιάμεσα της σειράς. Σε μια επανάληψη, το πρώτο μεγαλύτερο στοιχείο από το χ που θα βρει, θα το αντικαταστήσει με το χ και επιστρέψει αυτό το στοιχείο. Αν δεν βρει, επιστρέφει το χ που θα πρέπει να μπει σε επόμενη σειρά.

Αν το check βγει false τότε το χ θα προστεθεί στο τέλος της σειράς.

Aν έχουμε τελειώσει με όλες τις σειρές, το χ θα μπει στο τέλος του ταμπλό και θα επιστραφεί 0.

**Συνάρτηση bumping(x, tableau)**

A computer screen shot of a computer code

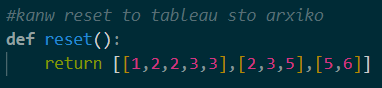
Description automatically generated

Αυτή η συνάρτηση εκτελεί τον αλγόριθμο bumping για την προσθήκη ενός στοιχείου στο ταμπλό. Παίρνει ως παραμέτρους το χ και το ταμπλό.

To counter είναι 0 επειδή αρχίζουμε από την πρώτη σειρά.

Σε ένα while loop, όσο το χ δεν είναι μηδέν, καλεί την add, παίρνει πίσω ένα χ, αυξάνει το counter, μέχρι η add να επιστρέψει 0 στο χ.

**Συνάρτηση reset()**



Επιστρέφει το αρχικό ταμπλό.

**Συνάρτηση revBumping(endTableau, tableau)**

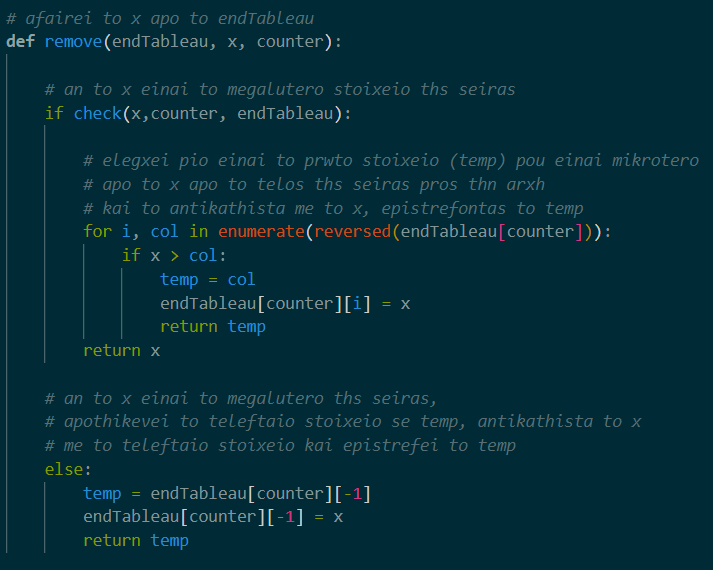


Εκτελεί τον reverse bumping αλγόριθμο. Παίρνει ως παραμέτρους το τελικό ταμπλό και το αρχικό ταμπλό.

Πρώτα ελέγχει αν τα δυο ταμπλό έχουν τον ίδιο αριθμό σειρών, γιατί αλλιώς ξέρουμε ότι το στοιχείο που ψάχνουμε είναι το τελευταίο του τελικού ταμπλό. Στην περίπτωση που έχουν ίδιο αριθμό σειρών, συγκρίνει κάθε σειρά των 2 ταμπλό μεταξύ τους, μέχρι να βρει πού προστέθηκε το στοιχείο. Όταν βρίσκει την σειρά, αποθηκεύει το νούμερο της και στο start αποθηκεύει το τελευταίο στοιχείο της σειράς.

Αποθηκεύει σε μια μεταβλητή χ το στοιχείο που θα αφαιρέσει. Σε μια while loop, όσο τα ταμπλό έχουν διαφορές σε κάθε σειρά, αφαιρεί το χ. Όταν τελειώσει επιστρέφει το χ.

**Συνάρτηση remove(endTableau, x, counter)**



Αυτή η συνάρτηση αφαιρεί το χ από το ταμπλό. Παίρνει ως παραμέτρους το τελικό ταμπλό, το χ και το counter που μας δείχνει την σειρά.

Ελέγχει αν το χ δεν είναι το μεγαλύτερο στοιχείο της γραμμής και με μια βρίσκει το πρώτο μικρότερο στοιχείο του χ για να το επιστρεψει, αντικαθιστώντας το χ με αυτό το στοιχείο. Αν δεν βρει, επιστρέφει το χ.

Αν το χ είναι το μεγαλύτερο στοιχείο της σειράς, επιστρέφει το τελευταιο στοιχείο της σειράς και το αντικαθιστά με το χ.

**Συνάρτηση print\_tableau(T)**

A screen shot of a computer code

Description automatically generated

Η συνάρτηση print\_tableau χρησιμοποιέιται για την εκτύπωση κάθε row ενός tableau Τ.

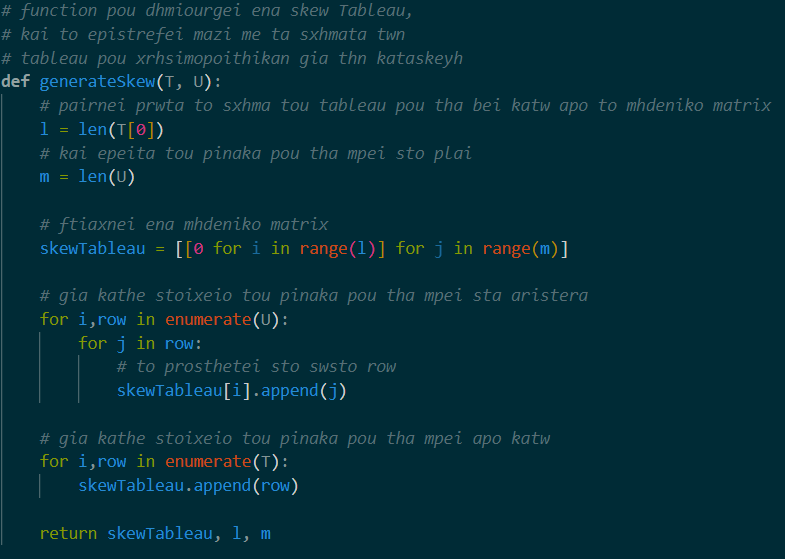
**Συνάρτηση prod(T,U)**

A screen shot of a computer

Description automatically generated

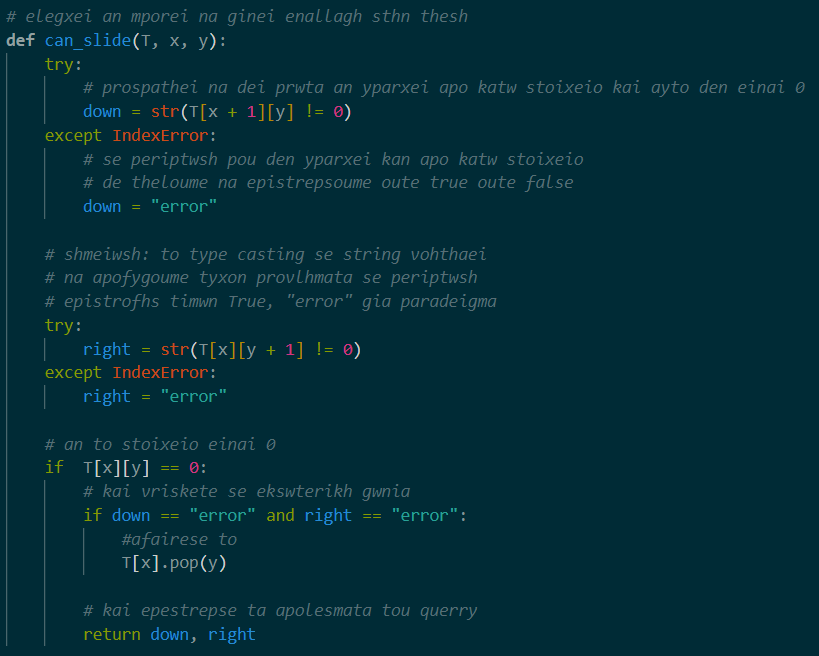
Η συνάρτηση prod(T, U) υπολογίζει το γινόμενο δύο tableau T και U. Συγκεκριμένα, προσθέτει τα στοιχεία του U στο T με τη σειρά από κάτω προς τα πάνω και από αριστερά προς τα δεξιά, χρησιμοποιώντας τον αλγόριθμο bumping για καθε στοιχειο που θα προστεθεί.

**Συνάρτηση generateSkew(T,U)**



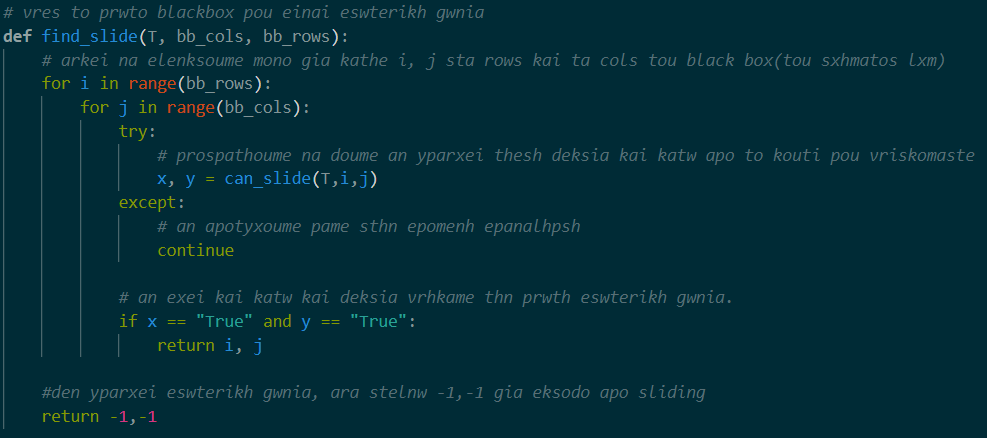
Η συνάρτηση generateSkew παίρνει ως ορίσματα δύο tableau, τον T και τον U, και υπολογίζει το σχήμα λxμ, όπου λ είναι το πλήθος των στοιχείων της μεγαλύτερης σειράς (πρώτης) του πίνακα Τ και μ το πλήθος των στηλών του U. Έπειτα, δημιουργεί έναν μηδενικό πίνακα λxμ, προσθέτει τα στοιχεία που θα προστεθούν (Πίνακας U) στο πλάι, και τον Πίνακα Τ από κάτω. Τέλος, επιστρέφει τον skew πίνακα, το λ, και το μ.

**Συνάρτηση can\_slide(T, x, y)**



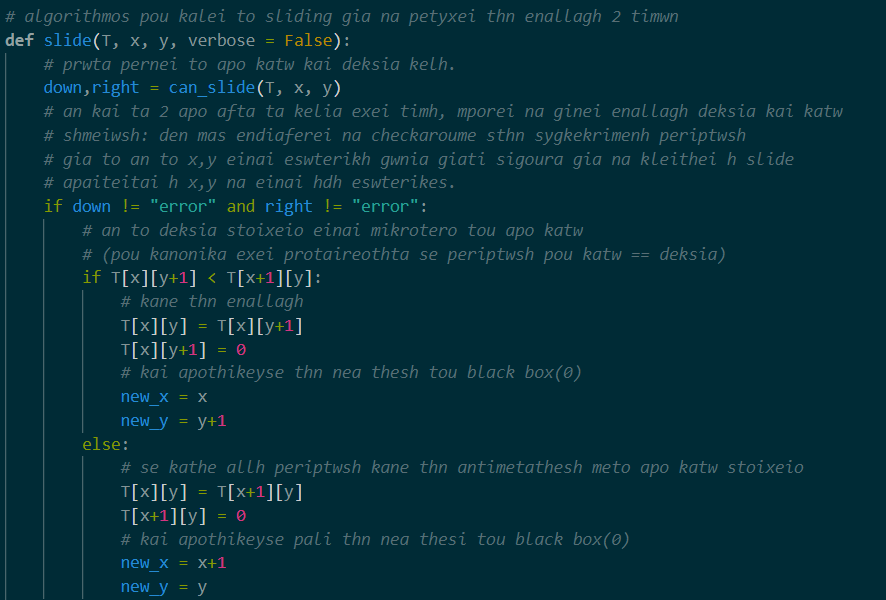
Η συνάρτηση can\_slide(T, x, y) ελέγχει αν μπορεί το τρέχον κελί να αλλάξει θέση, ελέγχοντας αν υπάρχει κελί από κάτω και δεξιά που η τιμή του να μην είναι 0. Συγκεκριμένα, πρώτα προσπαθεί να αποθηκεύσει την ύπαρξη κελιού από κάτω και από δεξιά, αποθηκεύοντας "true" αν υπάρχει, "false" αν υπάρχει και είναι 0, και "error" αν δεν υπάρχει κελί. Στη συνέχεια, ελέγχει αν το στοιχείο που βρισκόμαστε τώρα είναι μαύρο κουτί (0), που σημαίνει ότι μπορεί να εναλλαχθεί. Μετά, ελέγχει αν το κελί βρίσκεται σε εξωτερική γωνία που, στην συγκεκριμένη περίπτωση, το αφαιρεί, και έπειτα επιστρέφει τα αποτελέσματα.

**Συνάρτηση find\_slide(T, bb\_cols, bb\_rows)**



Η συνάρτηση find\_slide(T, bb\_cols, bb\_rows) αναζητάει το πρώτο μηδενικό σε εσωτερική γωνία, με τη χρήση της can\_slide, όπου τα αποτελέσματα της θα πρέπει να είναι True και True.

**Συνάρτηση slide(T, x, y, verbose = False)**



A computer screen shot of text

Description automatically generated

Η συνάρτηση slide(T, x, y, verbose = False) πραγματοποιεί την εναλλαγή του στοιχείου T[x][y], ανάλογα με τα αποτελέσματα της can\_slide. Συγκεκριμένα, στην περίπτωση που υπάρχουν και κάτω και δεξιά κελιά (και αυτά δεν είναι 0), αν το δεξί κελί είναι μικρότερο, πραγματοποιείται εναλλαγή με αυτό, αλλιώς γίνεται με το από κάτω. Ενώ, αν υπάρχει μόνο το από κάτω ή μόνο το δεξιά κελί, γίνεται η αντιμετάθεση αντίστοιχα. Σε όλες τις περιπτώσεις, αποθηκεύεται η νέα θέση του μηδενικού που μόλις αλλάξαμε θέση και η slide ξανακαλείται μέχρι αυτό να διωχθεί εντελώς από το tableau και να επιστρέψει true. Αν έχει δοθεί σαν παράμετρος στο verbose η τιμή True, σε κάθε αντιμετάθεση εκτυπώνεται το tableau, καθώς και το στοιχείο που αλλάξαμε.

**Συνάρτηση sliding(T, cols, rows, verbose = False)**

A computer screen shot of a computer code

Description automatically generated

Η συνάρτηση sliding(T, cols, rows, verbose = False), βρίσκει τις συντεταγμένες του πρώτου μαύρου κουτιού, και εφόσον αυτά δεν έχει την τιμή -1 (που ειναι η τιμή τερματισμού), καλεί την slide, εναλλάσοντάς το μεχρι αυτο να εξαφανιστέι, και η slide να επιστρέψει true, και μετά καλεί ξανά την slide για να συνεχιστεί η αντιμετάθεση.

**Συνάρτηση main()**



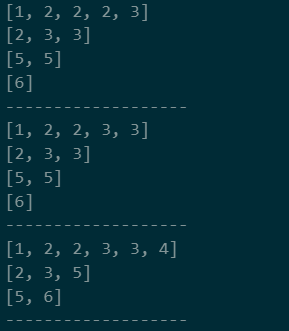
A screenshot of a computer program

Description automatically generated

Στην main, αρχικοποιούμε το tableau που μας δίνεται από την εκφώνηση της εργασίας καθώς και τις τιμές x, y, z και έπειτα κάνουμε add τις τιμές με τη χρήση bumping. Αποθηκεύουμε τον καινούργιο πίνακα και μετά κάνουμε reset το αρχικό tableau για να προσθέσουμε κάθε στοιχείο διαδοχικά. Έπειτα, καλούμε στους καινούργιους τελικούς πίνακες την reverse bumping για να βγάλουμε από αυτούς τις τιμές x, y, z που προσθέσαμε αντίστοιχα. Μετά, αρχικοποιούμε 2 καινούργια tableau T και U και καλούμε αρχικά την prod() πάνω τους και εκτυπώνουμε το product και στη συνέχεια φτιάχνουμε ένα skew tableau από τα T και U και καλούμε την sliding. Καθαρίζουμε τις περιττές κενές γραμμές από το αποτέλεσμα και εκτυπώνουμε και το αποτέλεσμα της sliding. Τέλος, παρατηρούμε ότι και για το τρίτο υποερώτημα (prod) και για το τέταρτο (sliding) ο τελικός πίνακας που προκύπτει είναι ο ίδιος.

ΕΚΤΕΛΕΣΗ

Α)



Β)



Γ)

A screenshot of a computer

Description automatically generated

Δ)

A screenshot of a computer

Description automatically generated

Ε) Παρατηρούμε ότι και για το τρίτο υποερώτημα (prod) και για το τέταρτο (sliding) ο τελικός πίνακας που προκύπτει είναι ο ίδιος.

ΨΕΥΔΟΚΩΔΙΚΑΣ

Συναρτηση check(x, row, tableau):

Επέστρεψε (όχι x > τελευταίο στοιχείο της γραμμής row)

Συναρτηση add(x, row, tableau):

Αν row <= μήκος tableau - 1:

Αν check(x, row, tableau):

Για i, col σε enumerate(tableau[row]):

Αν x < col:

Αντικατεστησε col με x

Επεστρεψε col

Επεστρεψε x

Αλλιώς:

Πρόσθεσε x στο τέλος της γραμμής

Επεστρεψε 0

Αλλιώς:

Πρόσθεσε νέα γραμμή και βαλε x

Επεστρεψε 0

Συνάρτηση bumping(x, tableau):

counter = 0

Όσο x != 0:

x = add(x, counter, tableau)

counter += 1

Συνάρτηση reset():

Επεστρεψε [[1, 2, 2, 3, 3], [2, 3, 5], [5, 6]]

Συνάρτηση revBumping(endTableau, tableau):

Αν μήκος endTableau == μήκος tableau:

Για i, row σε enumerate(tableau):

Αν μήκος endTableau[i] != μήκος tableau[i]:

counter = i

start = [i,-1]

Αλλιώς:

counter = 3

start = [-1,-1]

x = endTableau[start[0]].pop(start[1])

Όσο counter >= 1:

Αν endTableau[counter-1] != tableau[counter-1]:

x = remove(endTableau, x, counter-1)

counter -= 1

Επεστρεψε x

Συνάρτηση remove(endTableau, x, counter):

Αν check(x, counter, endTableau):

Για i, col σε enumerate(αντίστροφο endTableau[counter]):

Αν x > col:

Αντικατέστησε col με x

Επέστρεψε col

Επιστροφή x

Αλλιώς:

temp = τελευταίο στοιχείο της γραμμής

Αντικατέστησε τελευταίου στοιχείου με x

Επέστρεψε temp

Συνάρτηση print\_tableau(T):

Για κάθε γραμμή του T:

Εκτύπωσε γραμμή

Συνάρτηση prod(T, U):

Για κάθε γραμμή του U από το τέλος προς την αρχή:

Για κάθε στοιχείο της γραμμής:

bumping(στοιχείο, T)

Συνάρτηση generateSkew(T, U):

l = μήκος πρώτης γραμμής του T

m = μήκος του U

Δημιουργία μηδενικού πίνακα skewTableau με διαστάσεις m x l

Για κάθε γραμμή του U:

Πρόσθεσε στοιχείων στη γραμμή του skewTableau

Για κάθε γραμμή του T:

Πρόσθεσε γραμμής στον skewTableau

Επεστρεψε skewTableau, l, m

Συνάρτηση can\_slide(T, x, y):

Προσπάθεια εύρεσης στοιχείου κάτω

Προσπάθεια εύρεσης στοιχείου δεξιά

Αν T[x][y] == 0 και κάτω και δεξιά είναι "error":

Αφαίρεσε κελιού

Επέστρεψε αποτελεσμάτων

Συνάρτηση find\_slide(T, bb\_cols, bb\_rows):

Για κάθε i σε bb\_rows:

Για κάθε j σε bb\_cols:

Αν can\_slide(T, i, j):

Επέστρεψε i,j

Επέστρεψε -1, -1

Συνάρτηση slide(T, x, y, verbose=False):

Ανάθεσε θέσεων κάτω και δεξιά

Αν κάτω και δεξιά δεν είναι "error":

Αν T[x][y+1] < T[x+1][y]:

Αντικατέστησε στοιχείου με δεξιά

Ανάθεσε νέας θέσης black box

Αλλιώς:

Αντικατέστησε στοιχείου με κάτω

Ανάθεσε νέας θέσης black box

Αν κάτω δεν είναι "error" και δεξιά είναι "error":

Αντικατέστησε στοιχείου με κάτω

Ανάθεσε νέας θέσης black box

Αν κάτω είναι "error" και δεξιά δεν είναι "error":

Αντικατέστησε στοιχείου με δεξιά

Ανάθεσε νέας θέσης black box

Αν κάτω και δεξιά είναι "error":

Επέστρεψε True

Αν verbose:

Εκτύπωσε "swapping: " και tableau

Κλήση slide με νέα θέση black box

Επέστρεψε True

Συνάρτηση sliding(T, cols, rows, verbose=False):

x,y = find\_slide(T, cols, rows)

Αν x, y == -1:

Επέστρεψε

Αν slide(T, x, y, verbose) είναι επιτυχές:

Κλήση sliding

Συνάρτηση main():

Αρχικοποίησε tableau και x, y, z

bumping(x, tableau)

Εκτύπωσε tableau

Αντέγραψε σε endTableau1

reset tableau

bumping(y, tableau)

Εκτύπωση tableau

Αντέγραψε σε endTableau2

reset tableau

bumping(z, tableau)

Εκτύπωση tableau

Αντέγραψε σε endTableau3

reset tableau

Εκτύπωση revBumping(endTableau1, tableau)

Εκτύπωση revBumping(endTableau2, tableau)

Εκτύπωση revBumping(endTableau3, tableau)

Αρχικοποίηση T και U

Αντέγραψε Τ σε t\_prod

prod(t\_prod, U)

Εκτύπωσε "product of T\*U:"

Εκτύπωσε t\_prod

Δημιουργία skewTableau και σχήματα l, m

sliding(skewTableau, l, m)

Εκτύπωσε "Rect(T\*U):"

Αφαίρεσε κενών γραμμών από skewTableau

Εκτύπωσε skewTableau

Κλήση main()