ePublishing

ΕΘΝΙΚΟ ΚΕΝΤΡΟ ΤΕΚΜΗΡΙΩΣΗΣ
EKT
NATIONAL DOCUMENTATION CENTRE

**Open Schools Journal for Open Science**

Vol 2, No 1 (2019)

Special Issue Articles from the 1st Greek Student Conference on Research and Science

# Ταιριάσματα σε διμερή γραφήματα

*Α. Αρχοντάκη, Μ. Διονυσίδου, Χ. Κατσούλης, Μ. Μανιάτης, Α. Μίτσης, Χ. Μωραΐτης, Α. Ναυπλιώτου, Γ. Νιάρχου, Δ. Ξαγοράρη*

doi: 10.12681/osj.19334

**To cite this article:**

Αρχοντάκη Α., Διονυσίδου Μ., Κατσούλης Χ., Μανιάτης Μ., Μίτσης Α., Μωραΐτης Χ., Ναυπλιώτου Α., Νιάρχου Γ., & Ξαγοράρη Δ. (2019). Ταιριάσματα σε διμερή γραφήματα. *Open Schools Journal for Open Science*, *2*(1), 34–65. https://doi.org/10.12681/osj.19334

https://epublishing.ekt.gr | **e-Publisher:** EKT | **Downloaded at:** 17/08/2025 03:50:01

https://epublishing.ekt.gr | **e-Publisher:** EKT | **Downloaded at:** 17/08/2025 03:50:01

Open Schools Journal for Open Science

Volume 1

# Ταιριάσματα σε διμερή γραφήματα

*Μ. Διονυσίδου[1], Χ. Κατσούλης[1], Μ. Μανιάτης[1], Α. Μήτσης[1], Χ. Μωραΐτης[1], Α. Ναυπλιώτου[1], Γ. Νιάρχου[1], Δ. Ξαγοράρη[1], Β. Ξυμιαλή[1], Π. Πανίδου[1], Α. Παπά[1], Ε. Παπαγρηγορίου, Χ[1]. Παπαδάκης[1], Σ. Παπέλης[1], Α. Παπούλης[1], Α. Περιστέρη[1], Ε. Πουρνάρα[1], Β. Ρεμαντάς[1], Τ. Σαΐτη[1], Α. Σακελλαρίδης[1], Γ. Σπυρόπουλος[1], Ι. Στέκερ[1], Δ. Στεφάνου[1], Μ. Στρατιδάκης[1], Ο. Τζατζάνης[1], Η. Τσαβαρής[1], Ν. Φαράκλας[1], Αρχοντάκη Αθανασία[1], Παχούλη Αγνή[1], Τριπολιτάκη Μαρίνα[1]*

*[1]Πρότυπο Γυμνάσιο της Ιωνιδείου Σχολής Πειραιά (ΠΓΙΣΠ), Πειραιάς, Ελλάδα*

## ΠΕΡΙΛΗΨΗ

Ας υποθέσουμε ότι θέλουμε να παντρέψουμε πέντε κορίτσια με πέντε αγόρια. Ας υποθέσουμε επίσης ότι ζούμε σε μία μητριαρχική κοινωνία όπου ο λόγος των γυναικών υπερέχει εκείνου των αντρών, οπότε καταγράφουμε τις επιθυμίες μόνο των κοριτσιών. Κάθε αγόρι θα δεχτεί να παντρευτεί όποιο κορίτσι το επιλέξει. Ενδιαφερόμαστε όμως για ένα πλήρες ταίριασμα, δηλαδή, με άλλα λόγια, να παντρέψουμε κορίτσια και αγόρια κατά τις επιθυμίες των κοριτσιών και να μη μείνει κορίτσι (αλλά ούτε και αγόρι) ανύπαντρο. Το πρόβλημα αυτό είναι γνωστό στη βιβλιογραφία ως πρόβλημα του γάμου [1].

Η ύπαρξη ή μη λύσης στο παραπάνω πρόβλημα με τα ως άνω δεδομένα διαπιστώνεται εύκολα. Τι γίνεται όμως, αν τα δεδομένα αυξηθούν, αν ,για παράδειγμα, τα κορίτσια γίνουν 100;

Προβλήματα τέτοιας φύσεως απασχόλησαν τον Phillip Hall, ο οποίος έδωσε θεωρητική απάντηση το 1935 [1] (απέδειξε το πρόβλημα με χρήση μαθηματικής επαγωγής), ενώ οι αλγοριθμικές λύσεις ήρθαν πολύ αργότερα [4] και πλέον υπάρχουν πολλές μεταγραφές σε γλώσσες προγραμματισμού [5].

Στην εργασία αυτή παρουσιάζουμε τις ικανές και αναγκαίες συνθήκες ύπαρξης πλήρους ταιριάσματος σε διμερή γραφήματα μέσα από παραδείγματα και κατασκευάζουμε αλγοριθμικά λύσεις.. Στη συνέχεια, περιγράφουμε τον κώδικα που υλοποιήσαμε στο προγραμματιστικό περιβάλλον του Scratch1, για τις περιπτώσεις όπου το initial matching (που δεν εισάγεται από τον χρήστη, αλλά υπολογίζεται

δυναμικά) αφήνει αταίριαστο το πολύ ένα ζευγάρι κορυφών. Θα πρέπει να σημειωθεί ότι στο Scratch δεν μπορούν να οριστούν δυσδιάστατοι πίνακες και pointers, θέτοντας όρια εκ των προτέρων στη συγγραφή κώδικα και, γι' αυτόν ακριβώς το λόγο η περίπτωσή μας γίνεται πιο πολύπλοκη. Παραμένει, ωστόσο, ένα αξιόλογο εργαλείο εισαγωγής των μαθητών στον κόσμο των αλγόριθμων και του προγραμματισμού.

ΛΕΞΕΙΣ-ΚΛΕΙΔΙΑ:
Αλγόριθμοι; γραφήματα; θεώρημα Hall

## ΕΙΣΑΓΩΓΗ

Πότε λύνεται το πρόβλημα του γάμου; Αν λύνεται, μπορεί να κατασκευαστεί μία λύση; Μετά τον Hall [1], γνωρίζουμε ότι για να παντρευτούν όλα τα κορίτσια κατά τις επιθυμίες τους, πρέπει και αρκεί, οι επιθυμίες οποιουδήποτε υποσυνόλου κοριτσιών να είναι τουλάχιστον όσα και τα κορίτσια. Με άλλα λόγια, πρέπει και αρκεί:

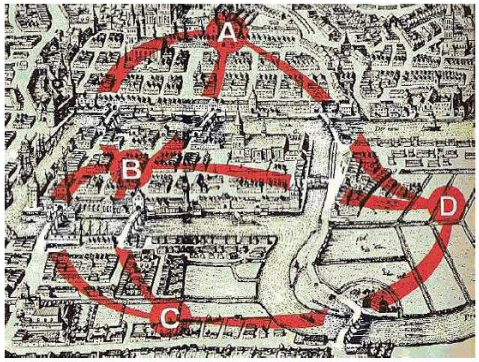
- ένα οποιαδήποτε κορίτσια να επιθυμεί τουλάχιστον ένα αγόρι
- δύο οποιαδήποτε κορίτσια να επιθυμούν τουλάχιστον δύο αγόρια
- τρία οποιαδήποτε κορίτσια να επιθυμούν τουλάχιστον τρία αγόρια

κτλ.

Ξεκινάμε με μία μικρή εισαγωγή στη θεωρία των γραφημάτων, τη μαθηματική θεωρία πίσω από τις αναζητήσεις αυτού του άρθρου ([2], [3], [4]). Αρχικά, παρουσιάζουμε το πρόβλημα του οποίου η προσπάθεια επίλυσης θεωρείται από τους ιστορικούς ότι έγινε η αιτία της γέννησης αυτής της θεωρίας.

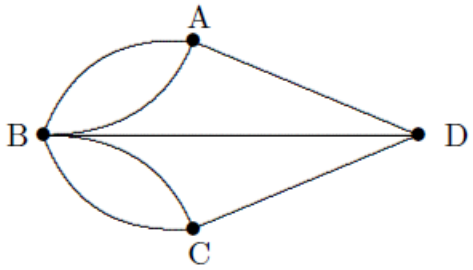
Οι 7 γέφυρες του Koenigsberg [2]. Στα μέσα του 18ου αιώνα υπήρχε στην τότε Πρωσία μια πόλη που ονομαζόταν Koenigsberg (το σημερινό Kaliningrad της Ρωσίας), την οποία διατρέχει το ποτάμι Pregel. Σε αυτό το ποτάμι υπάρχουν δύο μικρά νησιά τα οποία αποτελούν μέρος της πόλης και συνολικά 7 γέφυρες που συνδέουν τα ηπειρωτικά και νησιωτικά μέρη μεταξύ τους. Λέγεται ότι οι άνθρωποι στην πόλη διασκέδαζαν με το παρακάτω πρόβλημα: «Ξεκινώντας από οποιοδήποτε σημείο A,B,C ή D όπως στο σχήμα 1, είναι δυνατόν να βρεθεί διαδρομή που να περνά από κάθε μία από τις επτά γέφυρες ακριβώς μία φορά και να επιστρέφει στο σημείο που ξεκίνησε»; Κανείς δεν μπορούσε να βρει μία τέτοια διαδρομή, κανείς όμως δεν

μπορούσε και να αποδείξει ότι δεν υπάρχει.




Σχήμα 1.

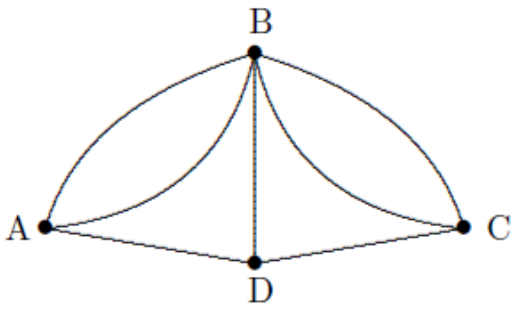
Το 1736, ο Leonhard Euler (1707-1783), επισκέφτηκε την πόλη, βρήκε το πρόβλημα ενδιαφέρον και συνειδητοποίησε ότι η φύση του είναι πολύ διαφορετική από αυτή των προβλημάτων της παραδοσιακής γεωμετρίας. Μελέτησε το πρόβλημα και το γενίκευσε, ανεξάρτητα από τον αριθμό των νησιών, των όχθεων και τον αριθμό των γεφυρών που τα συνδέουν και έλυσε το γενικευμένο πρόβλημα. Το εύρημά του, περιέχεται σε ένα άρθρο με τίτλο «Η λύση ενός προβλήματος που σχετίζεται με την γεωμετρία της θέσης» και δημοσιεύθηκε το 1736 [6]. Συμπέρανε ότι η λύση του προβλήματος είναι αδύνατη, για πρώτη φορά, με μαθηματικούς συλλογισμούς. Χρησιμοποίησε κορυφές (vertices) για να αναπαραστήσει νησιά και όχθες και ακμές (edges) για αναπαραστήσει τις γέφυρες. Η δύσκολη στην ανάγνωση γκραβούρα και το παρακάτω γράφημα (Σχήμα 2) μεταφέρουν ακριβώς την ίδια πληροφορία, μόνο που το γράφημα είναι πολύ πιο ευανάγνωστο.




Σχήμα 2.

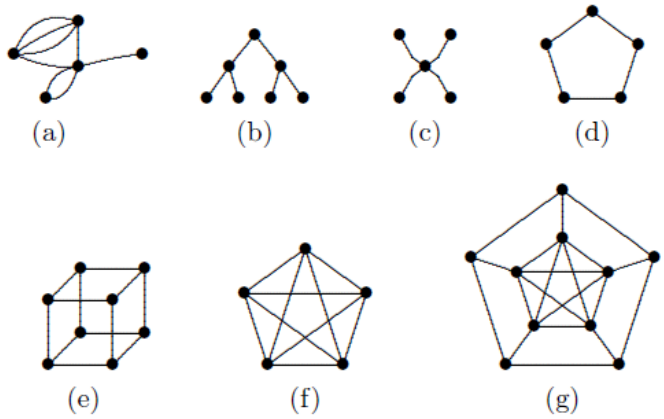
## ΓΡΑΦΗΜΑΤΑ

Το γράφημα του σχήματος 2 είναι σήμερα γνωστό ως πολυγράφημα (multigraph). Γενικά, ένα πολυγράφημα είναι ένα σύνολο κορυφών οι οποίες συνδέονται με μία ακμή ή έναν αριθμό ακμών. Για παράδειγμα, στο πολυγράφημα του σχήματος 2, οι κορυφές A και C δεν ενώνονται, οι κορυφές B και D συνδέονται από μία ακμή και οι κορυφές B και C ενώνονται από 2 ακμές. Οι θέσεις των κορυφών και τα μήκη των ακμών είναι αδιάφορα. Ενδιαφερόμαστε μόνο για το αν συνδέονται μεταξύ τους δύο κορυφές, και αν ναι, πόσες φορές. Έτσι, ο χάρτης του σχήματος 1 και τα γράφημα των σχημάτων 2 και 3, είναι ισοδύναμα μεταξύ τους (ισομορφικά).




Σχήμα 3.

Περισσότερα παραδείγματα πολυγραφημάτων δίνονται παρακάτω:



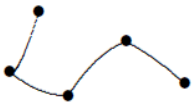

Σχήμα 4.

Από τα πολυγραφήματα του σχήματος 4, στο (a) υπάρχουν ζευγάρια κορυφών που συνδέονται με τουλάχιστον 2 ακμές. Στο καθένα από τα πολυγραφήματα (b) – (g), κάθε δύο κορυφές συνδέονται απ' το πολύ μία ακμή: τέτοια πολυγραφήματα ονομάζονται απλά γραφήματα ή για συντομία, γραφήματα. Έτσι κάθε γράφημα είναι ένα πολυγράφημα, αλλά το αντίστροφο δεν ισχύει.

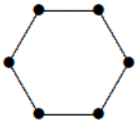
Έστω G πολυγράφημα, όπου V το σύνολο των κορυφών και E το σύνολο των ακμών του. Γράφουμε

G = (V,E).

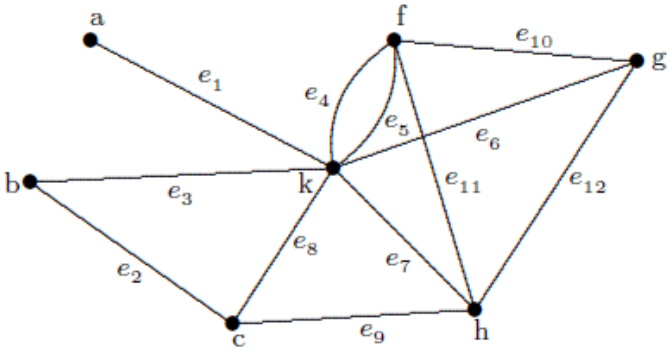
- Δύο κορυφές του G λέγονται γειτονικές αν συνδέονται με μία ακμή του G.
- Η γειτονιά μιας κορυφής είναι το σύνολο όλων των γειτονικών κορυφών της.
- Για κάθε S υποσύνολο του V ορίζουμε ως N(S) το υποσύνολο του G που αποτελείται από τις κορυφές του G που γειτονεύουν με κορυφές από το S. Γράφουμε |N(G)| για να δηλώσουμε το πλήθος των στοιχείων του N(G).
- Διαδρομή (walk) είναι κάθε ακολουθία κορυφών και ακμών όπου κάθε κορυφή γειτνιάζει με μία προηγούμενη και με μία επόμενη.
- Μονοπάτι (path) είναι κάθε διαδρομή που δεν επαναλαμβάνει τις ίδιες κορυφές και



- Κύκλος (cycle) είναι κάθε μονοπάτι που αρχίζει και τελειώνει στην ίδια κορυφή.



Για παράδειγμα, έστω το πολυγράφημα G του σχήματος 5. Έχει 7 κορυφές και 12 ακμές και V={a,b,c,f,g,h,k}, E={e1,e2,…,e12}.




Σχήμα 5.

Οι κορυφές a και k είναι γειτονικές, όμως η a και η b όχι. Η κορυφή a ανήκει (προσπίπτει) στην ακμή e1 αλλά όχι στην e3. Οι κορυφές a και k ενώνονται με την ακμή e1. Μπορούμε να γράψουμε: e1 = a-k, και να ονομάσουμε a και k τα δύο άκρα της e1.

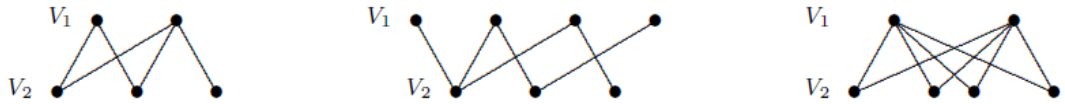
Στο πρόβλημα του γάμου, τα κορίτσια και αγόρια συνθέτουν τις κορυφές ενός γραφήματος, ενώ οι επιθυμίες των κοριτσιών αποτελούν τις ακμές του.

## ΔΙΜΕΡΗ ΓΡΑΦΗΜΑΤΑ

Σε αυτήν την παράγραφο, δίνουμε τον ορισμό των διμερών γραφημάτων, ο οποίος έχει κεντρικό ρόλο στα προβλήματα με τα οποία ασχολούμαστε. Ένα γράφημα G λέγεται διμερές (bipartite) αν το σύνολο των κορυφών του είναι η ένωση δύο μη κενών ξένων μεταξύ τους συνόλων κορυφών V1 και V2, τέτοιων που, κάθε ακμή του G να έχει το ένα άκρο της στο V1 και

το άλλο στο V2.

Τα παρακάτω γραφήματα είναι όλα διμερή.



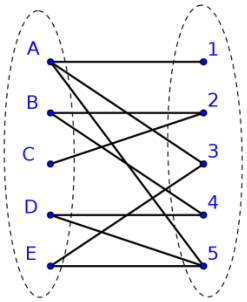

Στο πρόβλημα του γάμου τα κορίτσια και τα αγόρια αποτελούν κορυφές ενός γραφήματος, με V1 το σύνολο των κοριτσιών και V2 το σύνολο των αγοριών.

Ας δούμε ένα απλό παράδειγμα το οποίο κινείται πάνω στη λογική του προβλήματος του γάμου, προκειμένου να μας βοηθήσει να κατανοήσουμε την έννοια του πλήρους ταιριάσματος αλλά και πώς μπορούμε να φτάσουμε σε αυτήν.

**Το πρόβλημα των επιλογών του κηπουρού.** Ένας κηπουρός αποφάσισε να φυτέψει διάφορα καλλωπιστικά φυτά στις γλάστρες που ήταν άδειες στον κήπο που φρόντιζε. Οι άδειες γλάστρες ήταν διαφορετικού μεγέθους μεταξύ τους και τα φυτά που ήθελε να φυτέψει δε ταίριαζαν σε όλες τις γλάστρες. Για να μπορέσει να καταλήξει, έφτιαξε την παρακάτω λίστα, που περιγράφει ποια φυτά μπορούν να μπουν σε κάθε γλάστρα. Είναι εφικτό να φυτευτεί ακριβώς ένα τέτοιο φυτό σε κάθε γλάστρα, έτσι ώστε να έχει τελικά ο κηπουρός διαφορετικό φυτό σε κάθε γλάστρα;

| **POTS** | A | B | C | D | E |
|---|---|---|---|---|---|
| **FLOWERS** | 1,3,5 | 2,4 | 2 | 4,5 | 3,5 |

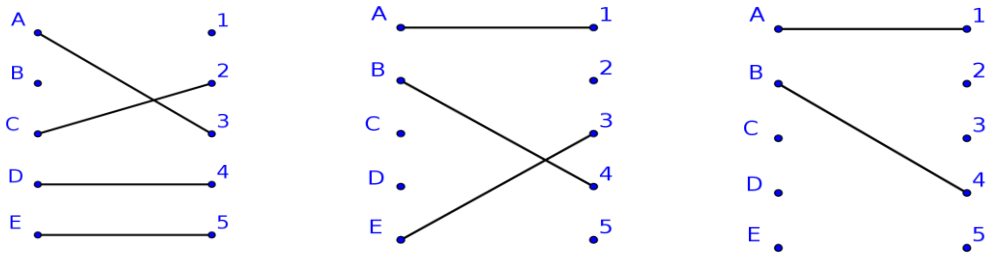
Είναι φανερό ότι ο παραπάνω πίνακας αντιστοιχεί στο διμερές γράφημα του σχήματος 6, όπου V1={A,B,C,D,E}, V2={1,2,3,4,5} και οι ακμές συνδέουν γλάστρες με φυτά που μπορούν να δεχτούν



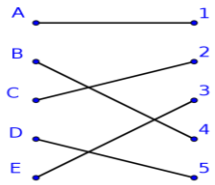
Σχήμα 6.

**Ταιριάσματα και πλήρη ταιριάσματα σε διμερή γραφήματα.** Συνεχίζουμε με δύο πολύ χρήσιμες έννοιες:

- Ταίριασμα (matching) σε διμερές γράφημα G = (V1 ∪ V2, E) είναι μία ένα προς ένα αντιστοίχηση κάποιων ή όλων των κορυφών του V1 με εκείνες του V2. Ταιριάσματα που προκύπτουν από το γράφημα του σχήματος 6 είναι μεταξύ άλλων τα ακόλουθα:




- Πλήρες ταίριασμα (complete matching) είναι κάθε ταίριασμα όπου περιλαμβάνει όλες τις κορυφές του γραφήματος. Πλήρες ταιριάσματα του γραφήματος του σχήματος 6 είναι το παρακάτω. Αυτό, δεν είναι παρά μία λύση στο πρόβλημα των επιλογών του κηπουρού:




Σχήμα 7.

Είμαστε πλέον σε θέση να διατυπώσουμε το θεώρημα του Hall [1] στη γλώσσα των μαθηματικών:

## ΤΟ ΘΕΩΡΗΜΑ ΤΟΥ HALL

Έστω G = (V1 ∪ V2, E) διμερές γράφημα, με |V1|=|V2|. Στο G υπάρχει πλήρες ταίριασμα, αν και μόνο αν, για κάθε υποσύνολο S του V1, είναι |S| ≤ |N(S)|, όπου N(S) οι γειτονικές κορυφές του S.

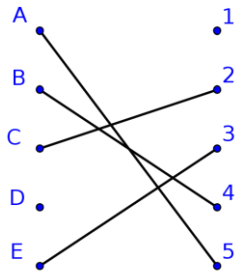
Πρέπει να επισημάνουμε ότι ο έλεγχος πλήρωσης των συνθηκών του θεωρήματος του

Hall δεν είναι απλή υπόθεση, αφού για ένα σύνολο μεγέθους n απαιτούνται 2^n -1 υπολογισμοί. Όταν ικανοποιούνται δεδομένες συνθήκες, το θεώρημα εξασφαλίζει την ύπαρξη λύσης, αλλά δεν την υπολογίζει. Στην επόμενη παράγραφο, επιγραμματικά, μέσα από παράδειγμα, παρουσιάζεται ο αλγόριθμος που οδηγεί στην κατασκευή λύσης [4].

## Ο ΑΛΓΟΡΙΘΜΟΣ ΠΛΗΡΟΥΣ ΤΑΙΡΙΑΣΜΑΤΟΣ ΣΕ ΔΙΜΕΡΗ ΓΡΑΦΗΜΑΤΑ

Πραγματοποιώντας 2^5-1 ελέγχους παρατηρούμε ότι οι συνθήκες του θεωρήματος του Hall ικανοποιούνται στην περίπτωση των επιλογών του κηπουρού. Άρα υπάρχει λύση και για να τη βρούμε εκτελούμε τα ακόλουθα βήματα :

**Βήμα 1.** Ξεκινάμε με ένα αρχικό ταίριασμα: (δηλαδή με μία πρώτη επιλογή. Κριτήρια σύμφωνα με τα οποία κατασκευάζουμε ένα πρώτο ταίριασμα θα τεθούν και θα αναλυθούν στην παράγραφο παρουσίασης του κώδικα).




Σχήμα 8.

Το αρχικό ταίριασμα του σχήματος 8, είναι ισοδύναμο με την ακολουθία ακμών: A-5, B-4, C-2, E-3.

**Βήμα 2.** Η κορυφή D δεν έχει ταίρι. Σύμφωνα όμως με το γράφημα 6, στην D θα μπορούμε να φυτέψουμε είτε το λουλούδι 4 είτε το λουλούδι 5. Στο αρχικό ταίριασμα, αντιστοιχίσαμε το λουλούδι 4 στη γλάστρα B, δηλαδή θεωρήσαμε την ακμή B-4. Διαγράφουμε αυτήν την ακμή, συμβολικά γράφουμε B=4 και δίνουμε το 4 στην D, δηλαδή σχηματίζουμε την ακμή D-4. Η B τώρα μένει ελεύθερη και την αντιστοιχούμε στο 2, αφαιρώντας παράλληλα το λουλούδι 2 από τη γλάστρα C. Μέχρι τώρα, έχουμε κάνει τις εξής κινήσεις:

D-4=B-2=C

**Βήμα 3.** Όμως, στην C δεν μπορούμε φυτέψουμε κάτι διαφορετικό από το 2. Οπότε, θα πρέπει να επαναλαμβάνουμε την όλη διαδικασία από την αρχή και να αντιστοιχήσουμε στην γλάστρα

D την άλλη της επιλογή, δηλαδή το λουλούδι 5. Επομένως το 5 φεύγει από την A και δίνουμε στην A την πρώτη της επιλογή, δηλαδή το λουλούδι 1. Δηλαδή:

D-5=A-1

Η παραπάνω ακολουθία αποτελεί ένα εναλλακτικό μονοπάτι (alternating path) για το γράφημα G.

**Βήμα 4.** Συνδυάζουμε τα αποτελέσματα του προηγούμενου βήματος με τις ακμές του αρχικού ταιριάσματος που δεν έχουμε διαγράψει μέχρι τώρα και σχηματίζουμε την ακολουθία ακμών

A-1, B-4, C-2, C-5, E-3

**Βήμα 5.** Διαπιστώνουμε ότι η παραπάνω ακολουθία δεν είναι παρά ένα πλήρες ταίριασμα και το πρόβλημα των επιλογών του κηπουρού έχει μόλις λυθεί.

**Ο αλγόριθμος επιγραμματικά.** Ο αλγόριθμος που παρουσιάσαμε συνοψίζεται στα ακόλουθα:

- ➔ **Βήμα 1**. Ξεκίνα από ένα οποιοδήποτε αρχικό ταίριασμα
- ➔ **Βήμα 2.** Ψάξε ένα εναλλακτικό μονοπάτι
- ➔ **Βήμα3.** Χρησιμοποίησε το εναλλακτικό μονοπάτι που μόλις βρήκες για να βελτιώσεις το αρχικό ταίριασμα. Αν δεν μπορείς να βρεις εναλλακτικό μονοπάτι, σταμάτα.
- ➔ **Βήμα 4.** Συνδύασε τα ταιριάσματα που δημιούργησες από το προηγούμενο βήμα με αυτά που είχες στο αρχικό σου ταίριασμα. Βάλε μαζί τις ακμές του αρχικού ταιριάσματος που δεν έχεις πειράξει μέχρι τώρα.
- ➔ **Βήμα 5**. Αν βρήκες πλήρες ταίριασμα σταμάτα, διαφορετικά ξαναγύρνα στο βήμα 2.

## Υλοποίηση σε περιβάλλον Scratch της αναζήτησης πλήρους ταιριάσματος σε διμερές γράφημα (Complete Matching)

Για την παρουσίαση του αλγόριθμου αναζήτησης πλήρους ταιριάσματος σε διμερές γράφημα, θα αναλυθούν τα επιμέρους βήματα επίλυσης του προβλήματος μέσω του παρακάτω παραδείγματος.

- **Το πρόβλημα**

Στοχεύοντας σε αποδοτικότερη λειτουργία της εταιρείας, ο Διευθυντής αποφάσισε να

αναθέσει συγκεκριμένη εργασία/εργασίες σε κάθε υπάλληλο. Για δική του διευκόλυνση, έφτιαξε την παρακάτω λίστα που περιγράφει τις εργασίες μπορεί να διεκπεραιώσει κάθε υπάλληλος της εταιρείας. Είναι εφικτό το ταίριασμα ζευγαριών, έτσι ώστε κάθε υπάλληλος να αναλάβει μια διαφορετική εργασία;

| **PERSON** | **TASKS** |
|---|---|
| A | 5, 1, 3 |
| B | 2, 4 |
| C | 2 |
| D | 3, 1 |
| E | 3, 5 |

**Η λύση**

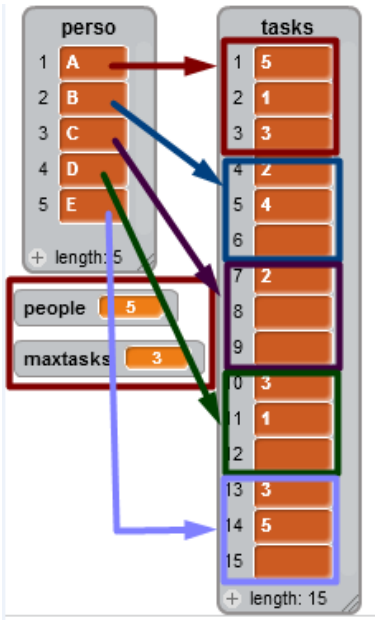
Το πρόγραμμα θα πρέπει να υλοποιεί τα εξής:

1. Αναζήτηση ατόμων με μία μόνο επιλογή και υποχρεωτικό ταίριασμα με την επιλογή αυτή.
2. Αναζήτηση επιλογής που ταιριάζει σε ένα μόνο άτομο. Εφόσον δεν συμπεριλαμβάνεται στις επιλογές των υπόλοιπων ατόμων, υποχρεωτικό ταίριασμα της επιλογής με το συγκεκριμένο άτομο.
3. Αρχικό ταίριασμα κάθε ατόμου με την πρώτη του επιλογή. Σε περίπτωση που η πρώτη επιλογή του δεν είναι διαθέσιμη, παραμένει αζευγάρωτος.
4. Αναζήτηση διαθέσιμης επιλογής για το αζευγάρωτο άτομο από το σύνολο των επιλογών του.
5. Αν δεν υπάρχει διαθέσιμη επιλογή για το αζευγάρωτο άτομο, αλλά μπορεί να αποδεσμευτεί μια επιλογή από κάποιο άλλο άτομο το οποίο έχει και άλλες δυνατές επιλογές, γίνεται αντιστοίχιση για το αζευγάρωτο άτομο.
6. Τα προηγούμενα βήματα επαναλαμβάνονται μέχρι να προκύψει πλήρες ταίριασμα ή να έχουν εξαντληθεί όλοι οι πιθανοί συνδυασμοί ταιριασμάτων (εναλλακτικά μονοπάτια).

**Ροή λειτουργιών προγράμματος και Ανάλυση των λειτουργιών**

1. Με το πάτημα της πράσινης σημαίας καλείται η διαδικασία **Data Entry** όπου:
   - αρχικοποιούνται οι τιμές των μεταβλητών
   - διαγράφονται τα περιεχόμενα από όλες τις λίστες
   - ο χρήστης, μέσω σχετικών ερωτήσεων, εισάγει:
     - σε δύο ξεχωριστές λίστες τα υποσύνολα κορυφών PERSON και TASKS (δημιουργούνται οι λίστες **perso** και **tasks**)

- το πλήθος των κορυφών για τις οποίες αναζητείται ταίριασμα (μεταβλητή **people**)
- το μέγιστο δυνατό πλήθος επιλογών ανά κορυφή που αναζητά ταίριασμα (μεταβλητή **maxtasks**)




Στο παράδειγμα, τα ονόματα των υπαλλήλων καταχωρούνται στη λίστα **perso** και οι επιλογές τους στη λίστα **tasks**.

Σε περίπτωση που κάποιος υπάλληλος έχει επιλογές λιγότερες από το μέγιστο δυνατό πλήθος (**maxtasks**), το αντίστοιχο στοιχείο της λίστας συμπληρώνεται με κενό χαρακτήρα.

Για παράδειγμα, ο υπάλληλος Β έχει δύο επιλογές (με μέγιστο πλήθος τις 3 επιλογές) οπότε η 3η επιλογή του συμπληρώνεται με κενό χαρακτήρα (στοιχείο 6 της λίστας tasks).

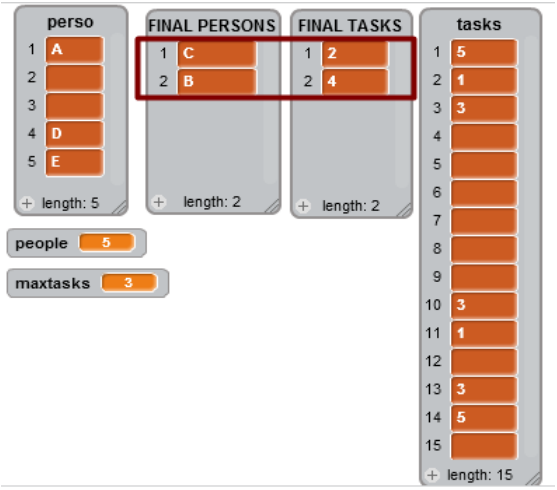
2. Στην επόμενη διαδικασία **Searching_elements_with_only_one_choice** που καλείται:

- Γίνεται αναζήτηση των κορυφών για τις οποίες υπάρχει μία μόνο επιλογή και τις ταιριάζει αυτόματα με αυτή.
  - Για να εξακριβωθεί αν μία κορυφή έχει μόνο μία επιλογή, γίνεται καταμέτρηση των κενών επιλογών της κορυφής (μεταβλητή **2keno**).
  - Εάν η τιμή της μεταβλητής **2keno** είναι ίση με τη διαφορά του συνόλου των δυνατών επιλογών (μεταβλητή maxtasks) - 1, δηλαδή <u>όλες οι επιλογές εκτός από μία είναι κενά</u>, τότε:
    - Γίνεται έλεγχος διαθεσιμότητας της συγκεκριμένης επιλογής.
      - Εάν η επιλογή είναι διαθέσιμη, η κορυφή και η αντίστοιχη επιλογή της προσθέτονται στις οριστικές λίστες **FINAL PERSONS** και **FINAL TASKS** (ταίριασμα). Επιπλέον, αντικαθίσταται η συγκεκριμένη κορυφή στη λίστα perso από κενό και τα στοιχεία της λίστας TASKS

που είναι ίσα με την επιλογή που προστέθηκε στη λίστα FINAL TASKS αντικαθίστανται με κενό. Η επιλογή αυτή δεν είναι πλέον διαθέσιμη για τους υπόλοιπους υποψήφιους.

- Εάν η επιλογή δεν είναι διαθέσιμη, τότε υπάρχουν κορυφές που διεκδικούν την ίδια μοναδική επιλογή. Αυτό συνεπάγεται αδυναμία πλήρους ταιριάσματος και το πρόγραμμα ολοκληρώνεται.

- Σε αντίθετη περίπτωση, εάν δηλαδή υπάρχουν περισσότερες από μία επιλογές για τη συγκεκριμένη κορυφή, η τιμή της μεταβλητής **2keno** μηδενίζεται και η διαδικασία επαναλαμβάνεται για την επόμενη κορυφή, μέχρι να ελεγχθούν όλες οι κορυφές.




Σύμφωνα με την αρχική καταγραφή επιλογών ανά υπάλληλο, ο υπάλληλος **C** είχε μία μόνο επιλογή (την εργασία **2**), η οποία του δόθηκε.

Το όνομά του και η επιλογή που του αναλογεί εκχωρούνται στις τελικές λίστες **FINAL PERSONS** και **FINAL TASKS**.

Διαγράφονται οι συγκεκριμένες τιμές από τη λίστα **perso** (τιμή C) και την λίστα **tasks** (τιμή 2).

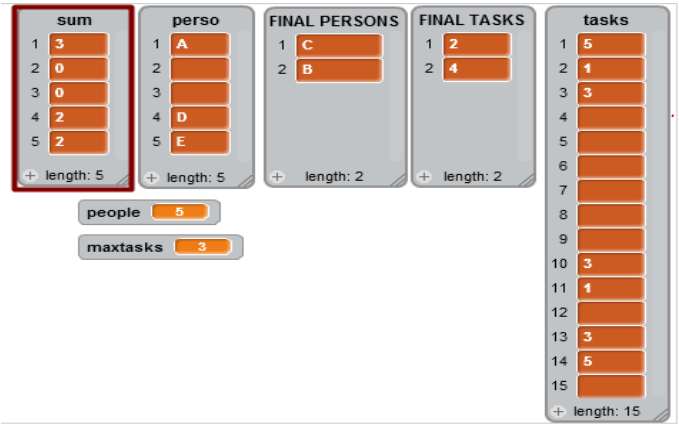
Διαγράφοντας τη δεσμευμένη εργασία 2 από τη λίστα των υπόλοιπων υπαλλήλων, προκύπτει μοναδική εργασία και για τον υπάλληλο B (εργασία 4).

Ακολουθείται η ίδια διαδικασία για τον υπάλληλο **B** και την εργασία **4**.

3. Ακολουθεί έλεγχος για πλήρες ταίριασμα μέσω της διαδικασίας **Check_for_complete_matching**, καθώς στο προηγούμενο βήμα (2) ενδέχεται όλες οι κορυφές να βρήκαν ταίρι. Σε περίπτωση πλήρους ταιριάσματος εμφανίζεται σχετικό μήνυμα και τα τελικά ζευγάρια καταχωρούνται στις λίστες **FINAL PERSONS** και **FINAL TASKS**. Σε αντίθετη περίπτωση,

συνεχίζεται η αναζήτηση.

4. Στη συνέχεια, καλείται η διαδικασία **Calculate_sum_of_choices_per_element** για τον υπολογισμό του συνολικού πλήθους επιλογών ανά κορυφή και το άθροισμα αποθηκεύεται σε αντίστοιχα στοιχεία της λίστας **sum**.

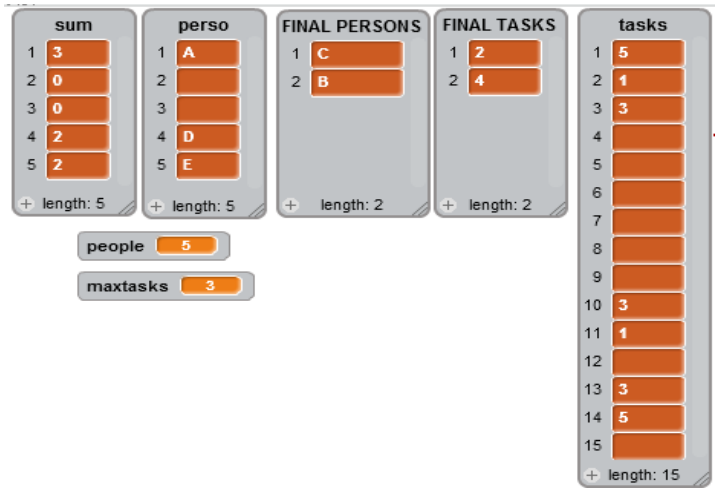



Στο παράδειγμα, ο 4ος υπάλληλος **D** έχει συνολικά 2 επιλογές (στοιχεία 10 και 11 της λίστας **tasks**).

Συμπληρώνεται το πλήθος επιλογών του υπάλληλου **D** στο αντίστοιχο 4ο στοιχείο της λίστας **sum**.

5. Η διαδικασία **Match_choices_which_appear_only_once** αναζητά επιλογές που αντιστοιχούν σε μία μόνο κορυφή, επιλογές δηλαδή που εμφανίζονται μία μόνο φορά στη λίστα **tasks**. Εάν παρουσιαστεί τέτοια περίπτωση, γίνεται υποχρεωτικό ταίριασμα κορυφής με αυτή την επιλογή.

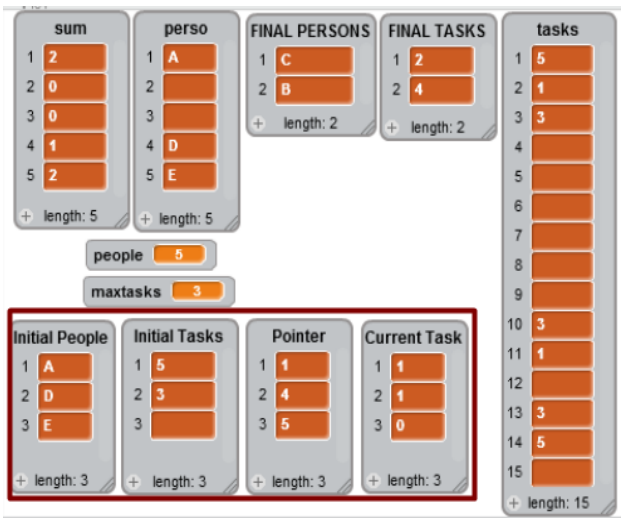
- Η κορυφή και η αντίστοιχη επιλογή της προσθέτονται στις οριστικές λίστες **FINAL PERSONS** και **FINAL TASKS** (ταίριασμα). Επιπλέον, αντικαθιστάται η συγκεκριμένη κορυφή στη λίστα perso από κενό και τα στοιχεία της λίστας TASKS που είναι ίσα με την επιλογή που προστέθηκε στη λίστα FINAL TASKS αντικαθίστανται με κενό. Η επιλογή αυτή δεν είναι πλέον διαθέσιμη για τους υπόλοιπους υποψήφιους.




Στο παράδειγμα δεν προκύπτει τέτοια περίπτωση, καθώς κάθε αδέσμευτη επιλογή (1, 3 και 5) της λίστας **tasks** συμπεριλαμβάνεται στις επιλογές περισσότερων του ενός ατόμων.

6. Εφόσον δεν έχει προκύψει πλήρες ταίριασμα, καλείται η διαδικασία **Init_match** που πραγματοποιεί το αρχικό ταίριασμα (initial matching). Ταιριάζει δηλαδή τις αδέσμευτες κορυφές με την πρώτη τους επιλογή, εάν αυτή είναι διαθέσιμη.

- Προσθέτει τα στοιχεία της λίστας **perso** που δεν είναι κενά στη λίστα **Initial People**
- Ξεκινώντας από το πρώτο άτομο, ελέγχει εάν η πρώτη επιλογή του υπάρχει ήδη στη λίστα **Initial Tasks** (αρχικά είναι κενή).
    - Εάν η συγκεκριμένη επιλογή δεν υπάρχει ήδη στη λίστα **Initial Tasks**:
        - την εισάγει στη λίστα **Initial Tasks**,
        - μειώνει το αντίστοιχο στοιχείο της λίστας **sum** κατά 1, αφού πλέον το άτομο έχει μία λιγότερη διαθέσιμη επιλογή
        - εισάγει τον αριθμό **1** στο αντίστοιχο στοιχείο της λίστας **Current Task**, υποδεικνύοντας ότι το άτομο έχει αντιστοιχηθεί αυτή τη στιγμή με την πρώτη του επιλογή.
        - στο αντίστοιχο στοιχείο της λίστας **Pointer** εισάγεται ο αύξοντας αριθμός του ατόμου
    - Εάν η συγκεκριμένη επιλογή υπάρχει ήδη στη λίστα **Initial Tasks**:
        - εισάγει στη λίστα **Initial Tasks** ένα κενό χαρακτήρα
        - εισάγει στο αντίστοιχο στοιχείο της λίστας **Current Task** τον αριθμό **0**, υποδεικνύοντας ότι το άτομο δεν έχει αντιστοιχηθεί ακόμα με την πρώτη του επιλογή.
        - στο αντίστοιχο στοιχείο της λίστας **Pointer** εισάγεται ο αύξοντας αριθμός του ατόμου



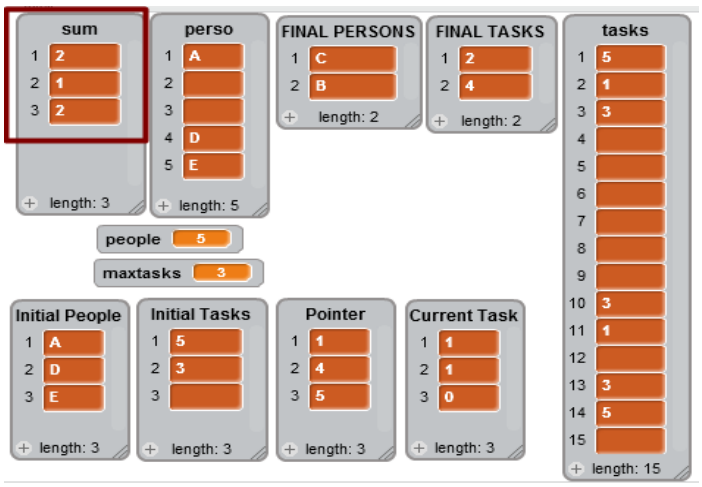

Για παράδειγμα, ο υπάλληλος **A** έχει αντιστοιχηθεί με την πρώτη του επιλογή (5) ενώ παράλληλα το στοιχείο 1 της λίστας **sum** μειώθηκε κατά 1 και της λίστας **Current Task** έγινε 1. Καθώς ο υπάλληλος **A** ήταν ο πρώτος υπάλληλος σε σειρά, καταχωρήθηκε στο αντίστοιχο στοιχείο της λίστας **Pointer** ο αύξοντας αριθμός 1.

Όμοια και για τον υπάλληλο **D**.

Αντίθετα, ο υπάλληλος **E** δεν μπορεί να αντιστοιχηθεί με την πρώτη του επιλογή (3) επειδή την έχουμε ήδη ταιριάξει με τον υπάλληλο **D**. Το αντίστοιχο στοιχείο της λίστας Initial Tasks για τον υπάλληλο **E** παραμένει κενό, ενώ το στοιχείο Current Task του ορίζεται σε 0.

7. Ακολουθεί η κλήση της διαδικασίας **Discard_sum_equal_to_zero**, η οποία εξασφαλίζει τη διαγραφή των μηδενικών στοιχείων από τη λίστα **sum**. Πρόκειται για στοιχεία που αντιστοιχούν σε άτομα που έχουμε ήδη ταιριάξει οριστικά και βρίσκονται πλέον στη λίστα **Final Persons**.

- Ελέγχονται διαδοχικά όλα τα στοιχεία της λίστας **sum** και διαγράφονται όσα έχουν μηδενική τιμή.

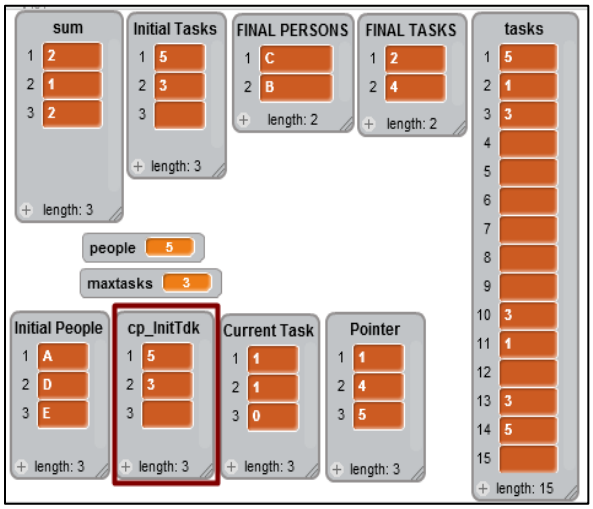



Στο προηγούμενο βήμα, τα στοιχεία 2 και 3 της λίστας **sum** είχαν μηδενική τιμή καθώς αντιστοιχούσαν στα άτομα **B** και **C** που έχουμε ταιριάξει οριστικά.

Για τον λόγο αυτό τα στοιχεία αυτά διαγράφηκαν εντελώς από τη λίστα και παρέμεινε μόνο το συνολικό πλήθος επιλογών των αταίριαστων υπαλλήλων (A, D και E).

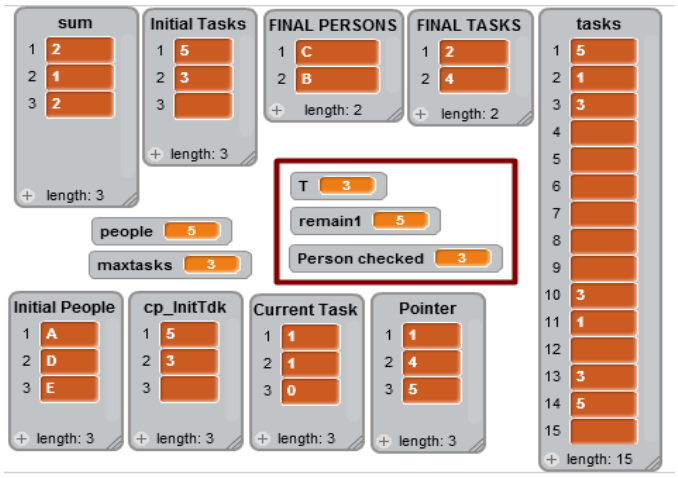
8. Μετά τον έλεγχο γα πλήρες ταίριασμα - κι εφόσον δεν έχει προκύψει ακόμα - καλείται η διαδικασία **Complete_Match** στην οποία:

- Δημιουργείται η λίστα **cp_initTdk** που αποτελεί αντίγραφο του αρχικού ταιριάσματος, προσθέτουμε δηλαδή όλα τα στοιχεία της λίστας Intial Tasks. Στη νέα λίστα **cp_intiTdk** θα γίνονται τυχόν αντικαταστάσεις επιλογών ανά άτομο.




- Ακολουθεί ο εντοπισμός του αζευγάρωτου υπάλληλου στη λίστα **cp_InitTdk** με τη βοήθεια της διαδικασίας **Search_for_unmatched_person**:
    - Ελέγχονται τα στοιχεία της λίστας **cp_InitTdk** μέχρι να εντοπιστεί κενό στοιχείο, άρα πρόκειται για τον υπάλληλο που - προς το παρόν - δεν έχει ταίρι.

- Καταχωρείται ο αύξοντας αριθμός του κενού στοιχείου στη μεταβλητή **T,** στη μεταβλητή **Person checked** και ορίζεται η μεταβλητή **remain1** στο στοιχείο T της λίστας **pointer** (ώστε να ξέρουμε σε ποιά θέση της λίστας **perso** βρίσκεται ο αζευγάρωτος υπάλληλος κι επομένως ποιές επιλογές της λίστας **tasks** του αντιστοιχούν).




Στο παράδειγμα, το τρίτο στοιχείο της λίστας **cp_InitTdk** είναι αυτό που δεν έχει ταίρι, οπότε εκχωρείται η τιμή 3 στην μεταβλητή **T**.

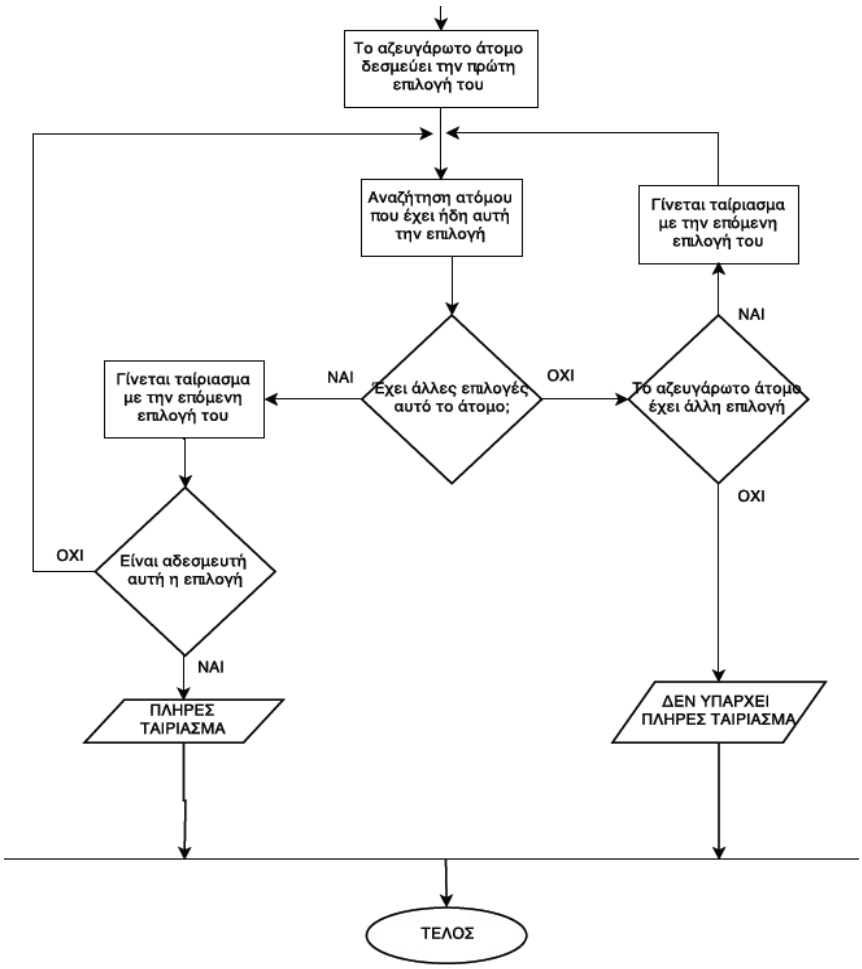
Την ίδια τιμή παίρνει και η μεταβλητή **Person checked**, ενώ η μεταβλητή **remain1** αποθηκεύει την αρχική θέση του αζευγάρωτου υπάλληλου στην λίστα **perso** (ο υπάλληλος Ε ήταν 5ος στη σειρά).

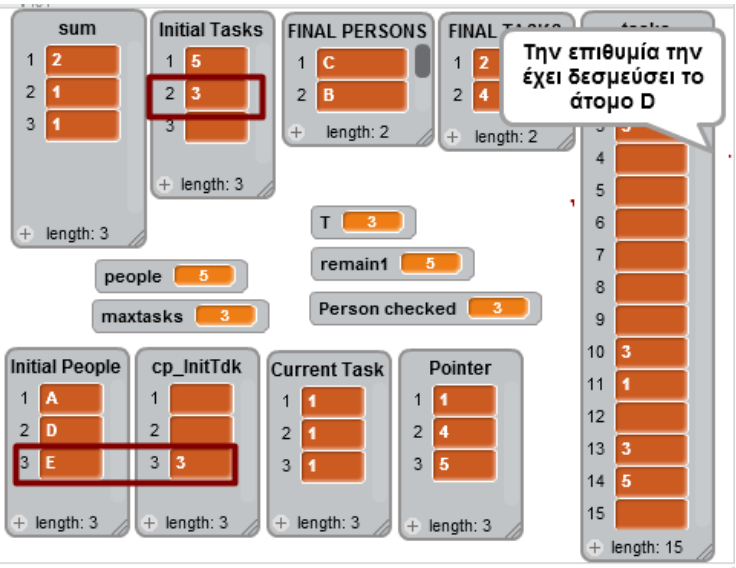
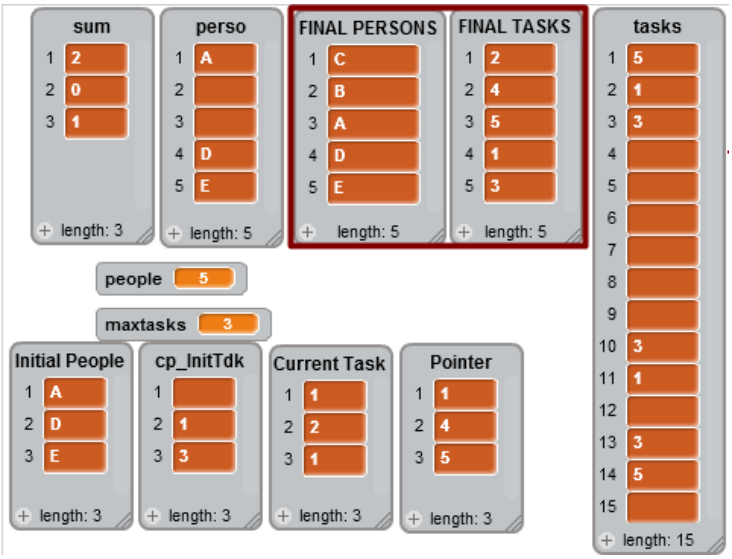
- Αρχικά, πραγματοποιείται αναζήτηση διαθέσιμης επιλογής για τον αζευγάρωτο υπάλληλο από το σύνολο των επιλογών του. Ελέγχονται με τη βοήθεια των μεταβλητών **remain1** και **Person checked** όλες οι διαθέσιμες επιλογές του αζευγάρωτου υπάλληλου.
  - Εάν υπάρχει διαθέσιμη επιλογή, που δεν περιέχεται δηλαδή ήδη στη λίστα **cp_InitTdk**, τότε την προσθέτουμε στην αντίστοιχη θέση της λίστας **cp_InitTdk** και προκύπτει πλήρες ταίριασμα.
  - Εάν δεν υπάρχει διαθέσιμη επιλογή για το αζευγάρωτο άτομο, καλείται η διαδικασία **Searching_for_alternating_path** για την αναζήτηση εναλλακτικού μονοπατιού (alternating path).

Σύμφωνα με διαδικασία **Searching_for_alternating_path,** αν μπορεί να αποδεσμευτεί η πρώτη αρχικά επιλογή του αζευγάρωτου υπάλληλου από κάποιον άλλο υπάλληλο που έχει κι άλλες δυνατές επιλογές, γίνεται ταίριασμα με την επόμενη δυνατή επιλογή του. Το παραπάνω βήμα επαναλαμβάνεται μέχρι να προκύψει πλήρες ταίριασμα ή να έχουν εξαντληθεί όλοι οι πιθανοί συνδυασμοί (εναλλακτικά μονοπάτια).

Η διαδικασία **Searching_for_alternating_path** μπορεί να γίνει πιο κατανοητή βλέποντας το

τμήμα του λογικού διαγράμματος που της αντιστοιχεί:



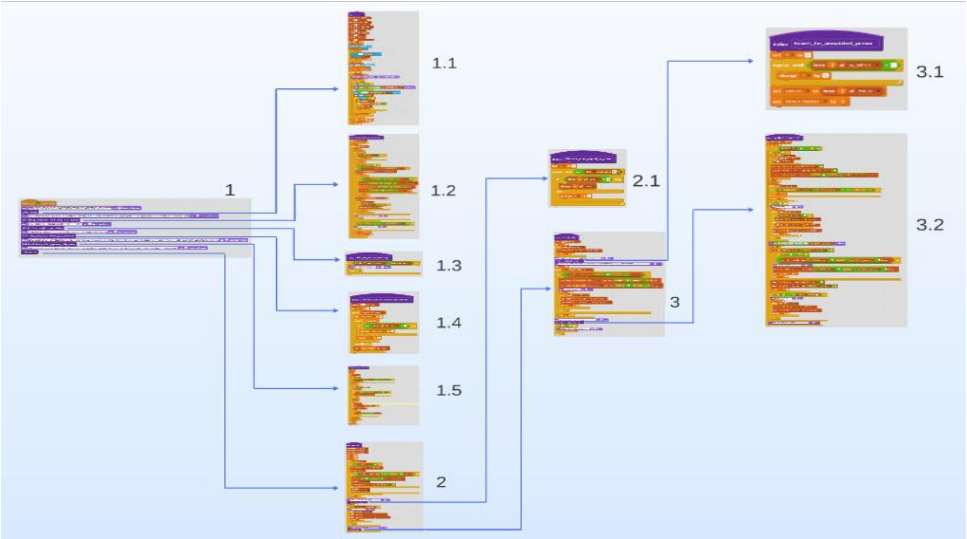
| | |
|---|---|
|  | Στο παράδειγμα, ο αζευγάρωτος υπάλληλος **E** δεσμεύει την πρώτη επιλογή του **3** και την προσθέτει στη λίστα **cp_InitTdk**.<br><br>Η επιλογή 3 δεν είναι διαθέσιμη, την έχει ήδη ο υπάλληλος **D** (λίστα **Initial Tasks**).<br><br>Η επόμενη επιλογή **1** του υπάλληλου **D** είναι διαθέσιμη, οπότε γίνεται ταίριασμα του D με την επόμενη επιλογή του. |
|  | Στη λίστα **cp_InitTdk** έχουν δεσμευθεί οι επιλογές **1** και **3** για τους υπάλληλους **D** και **E** αντίστοιχα.<br><br>Εφόσον στις επιλογές του υπάλληλου **A** συμπεριλαμβάνεται η αδέσμευτη επιλογή **5**, τη δεσμεύει και καταλήγουμε σε πλήρες ταίριασμα. |

**Λογικό Διάγραμμα**

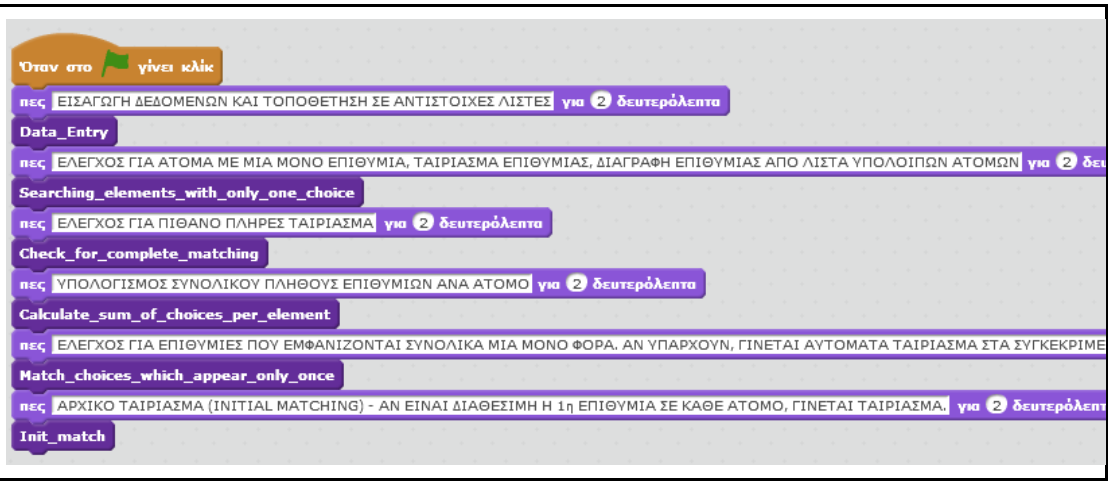
Το λογικό διάγραμμα που απεικονίζει όλα τα βήματα του παραπάνω αλγόριθμου είναι προσβάσιμο στον ακόλουθο σύνδεσμο:
https://drive.google.com/file/d/0B_jQOhJLJdkWTFdkTjFBOWxPZ1k/view?usp=sharing

## Διάγραμμα Κώδικα

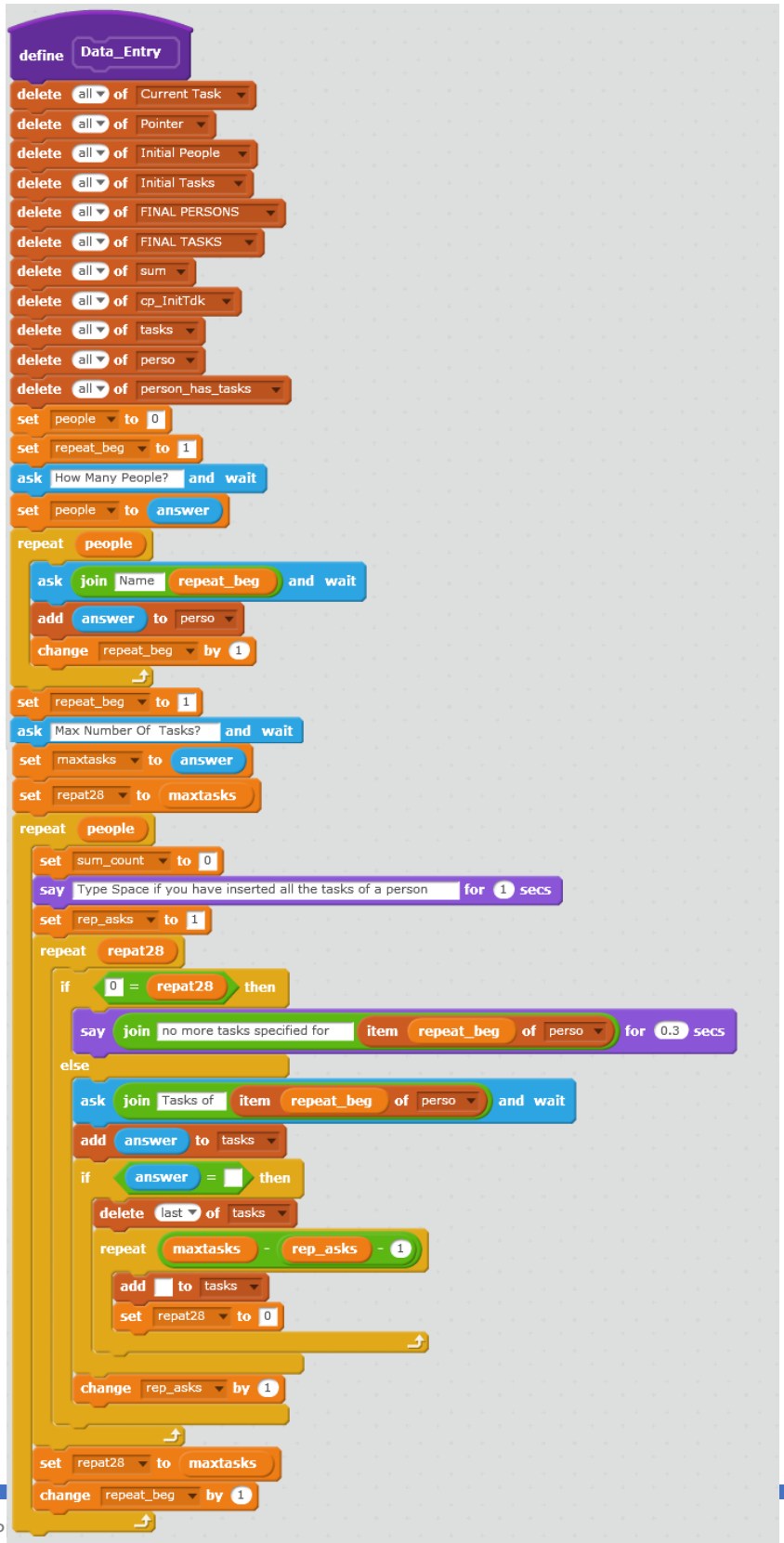
Για την κατανόηση των λειτουργιών προτείνεται να επισκεφθείτε τον παρακάτω σύνδεσμο (http://prezi.com/oegcwiqeihoe/?utm_campaign=share&utm_medium=copy), όπου παρουσιάζονται τα διάφορα τμήματα του κώδικα και ο τρόπος που συνδέονται μεταξύ τους.



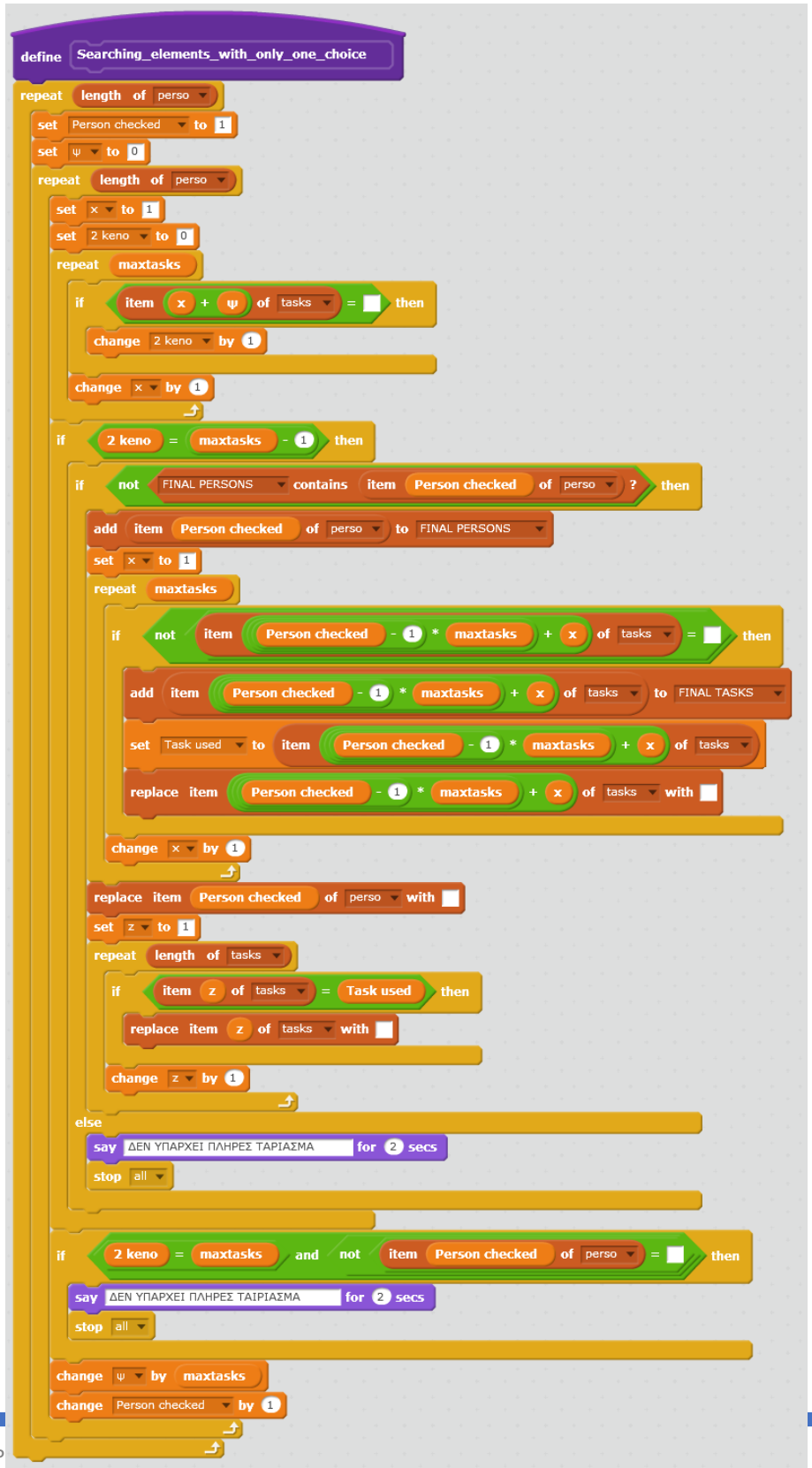
## Ο κώδικας σε Scratch

| | |
|---|---|
| **1** |  |

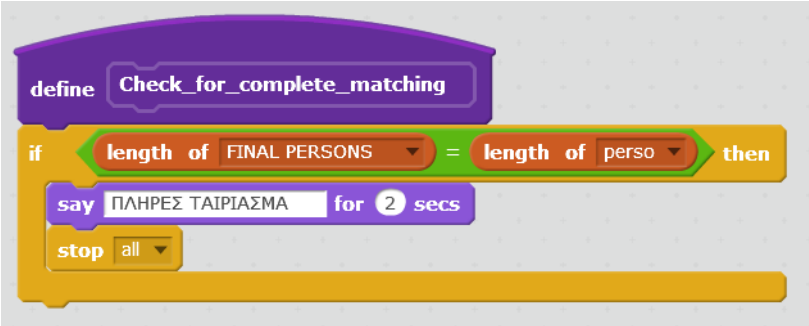
**1.1**


define Data_Entry
delete all of Current Task
delete all of Pointer
delete all of Initial People
delete all of Initial Tasks
delete all of FINAL PERSONS
delete all of FINAL TASKS
delete all of sum
delete all of cp_InitTdk
delete all of tasks
delete all of perso
delete all of person_has_tasks
set people to 0
set repeat_beg to 1
ask How Many People? and wait
set people to answer
repeat people
ask join Name repeat_beg and wait
add answer to perso
change repeat_beg by 1
set repeat_beg to 1
ask Max Number Of Tasks? and wait
set maxtasks to answer
set repat28 to maxtasks
repeat people
set sum_count to 0
say Type Space if you have inserted all the tasks of a person for 1 secs
set rep_asks to 1
repeat repat28
if 0 = repat28 then
say join no more tasks specified for item repeat_beg of perso for 0.3 secs
else
ask join Tasks of item repeat_beg of perso and wait
add answer to tasks
if answer = then
delete last of tasks
repeat maxtasks - rep_asks - 1
add to tasks
set repat28 to 0
change rep_asks by 1
set repat28 to maxtasks
change repeat_beg by 1

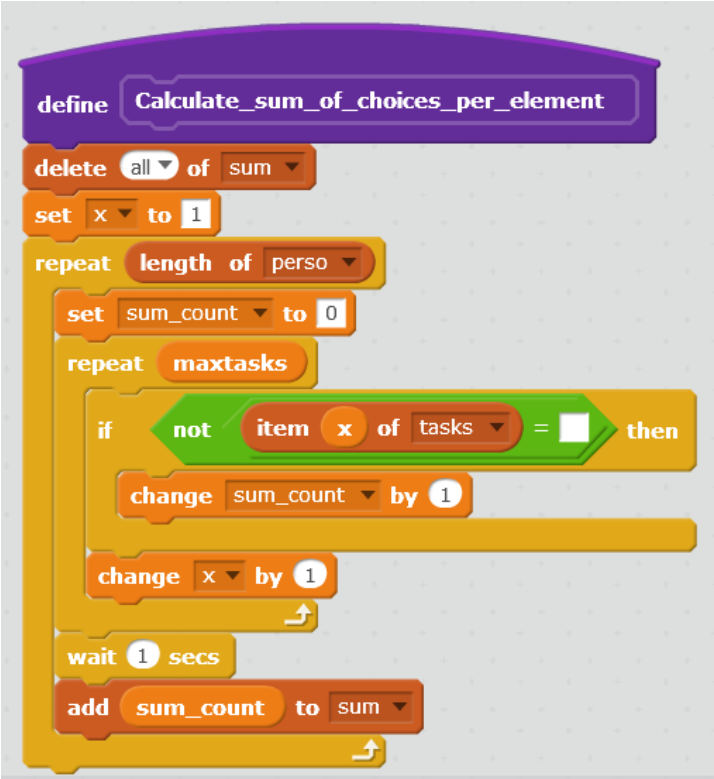
**1.2**


define Searching_elements_with_only_one_choice
repeat length of perso
set Person checked to 1
set ψ to 0
repeat length of perso
set x to 1
set 2 keno to 0
repeat maxtasks
if item x + ψ of tasks = then
change 2 keno by 1
change x by 1
if 2 keno = maxtasks - 1 then
if not FINAL PERSONS contains item Person checked of perso ? then
add item Person checked of perso to FINAL PERSONS
set x to 1
repeat maxtasks
if not item Person checked - 1 * maxtasks + x of tasks = then
add item Person checked - 1 * maxtasks + x of tasks to FINAL TASKS
set Task used to item Person checked - 1 * maxtasks + x of tasks
replace item Person checked - 1 * maxtasks + x of tasks with
change x by 1
replace item Person checked of perso with
set z to 1
repeat length of tasks
if item z of tasks = Task used then
replace item z of tasks with
change z by 1
else
say ΔΕΝ ΥΠΑΡΧΕΙ ΠΛΗΡΕΣ ΤΑΡΙΑΣΜΑ for 2 secs
stop all
if 2 keno = maxtasks and not item Person checked of perso = then
say ΔΕΝ ΥΠΑΡΧΕΙ ΠΛΗΡΕΣ ΤΑΙΡΙΑΣΜΑ for 2 secs
stop all
change ψ by maxtasks
change Person checked by 1

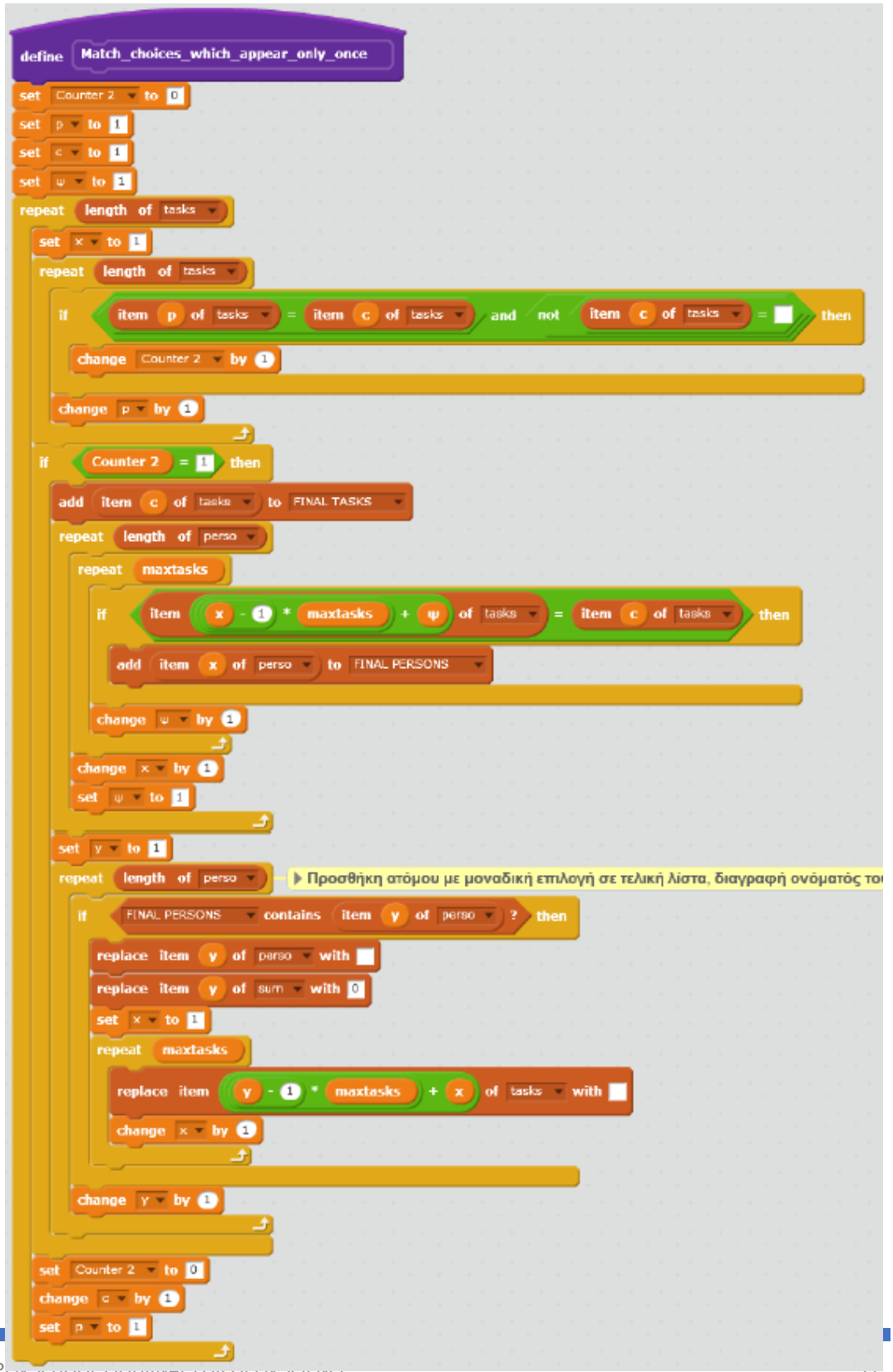
**1.3**


define Check_for_complete_matching
if length of FINAL PERSONS = length of perso then
say ΠΛΗΡΕΣ ΤΑΙΡΙΑΣΜΑ for 2 secs
stop all

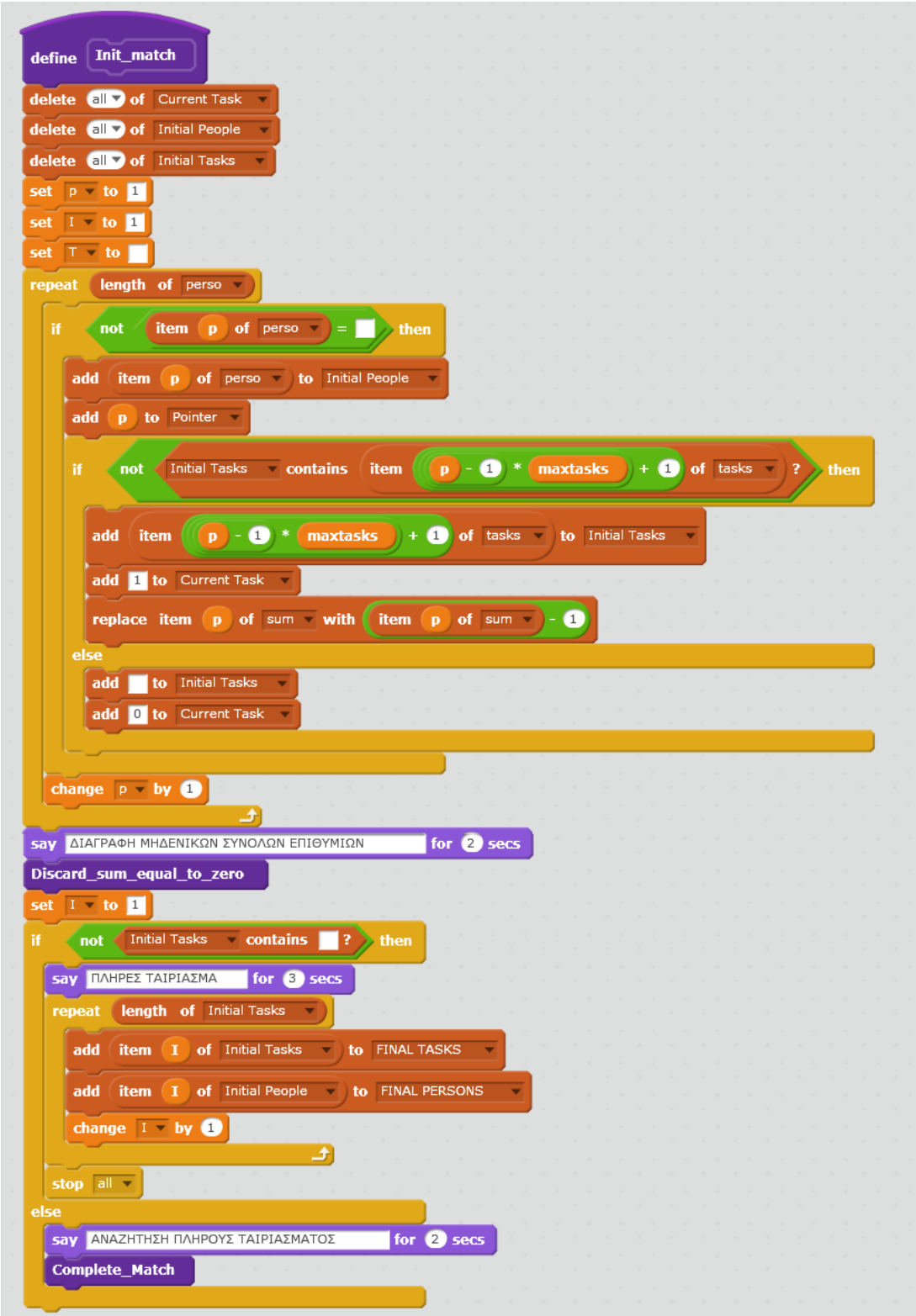
**1.4**

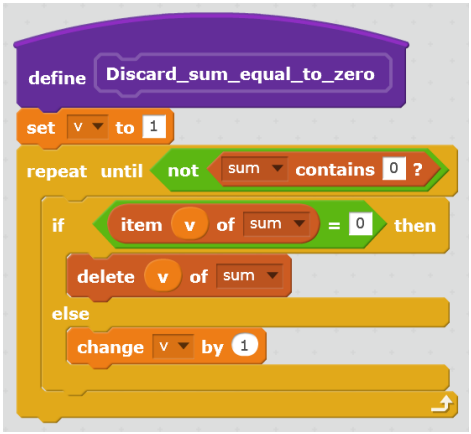

define Calculate_sum_of_choices_per_element
delete all of sum
set x to 1
repeat length of perso
set sum_count to 0
repeat maxtasks
if not item x of tasks = then
change sum_count by 1
change x by 1
wait 1 secs
add sum_count to sum

**1.5**


define Match_choices_which_appear_only_once
set Counter 2 to 0
set p to 1
set c to 1
set ψ to 1
repeat length of tasks
set x to 1
repeat length of tasks
if item p of tasks = item c of tasks and not item c of tasks = then
change Counter 2 by 1
change p by 1
if Counter 2 = 1 then
add item c of tasks to FINAL TASKS
repeat length of perso
repeat maxtasks
if item x - 1 * maxtasks + ψ of tasks = item c of tasks then
add item x of perso to FINAL PERSONS
change ψ by 1
change x by 1
set ψ to 1
set y to 1
repeat length of perso
Προσθήκη ατόμου με μοναδική επιλογή σε τελική λίστα, διαγραφή ονόματός του
if FINAL PERSONS contains item y of perso ? then
replace item y of perso with
replace item y of sum with 0
set x to 1
repeat maxtasks
replace item y - 1 * maxtasks + x of tasks with
change x by 1
change y by 1
set Counter 2 to 0
change c by 1
set p to 1

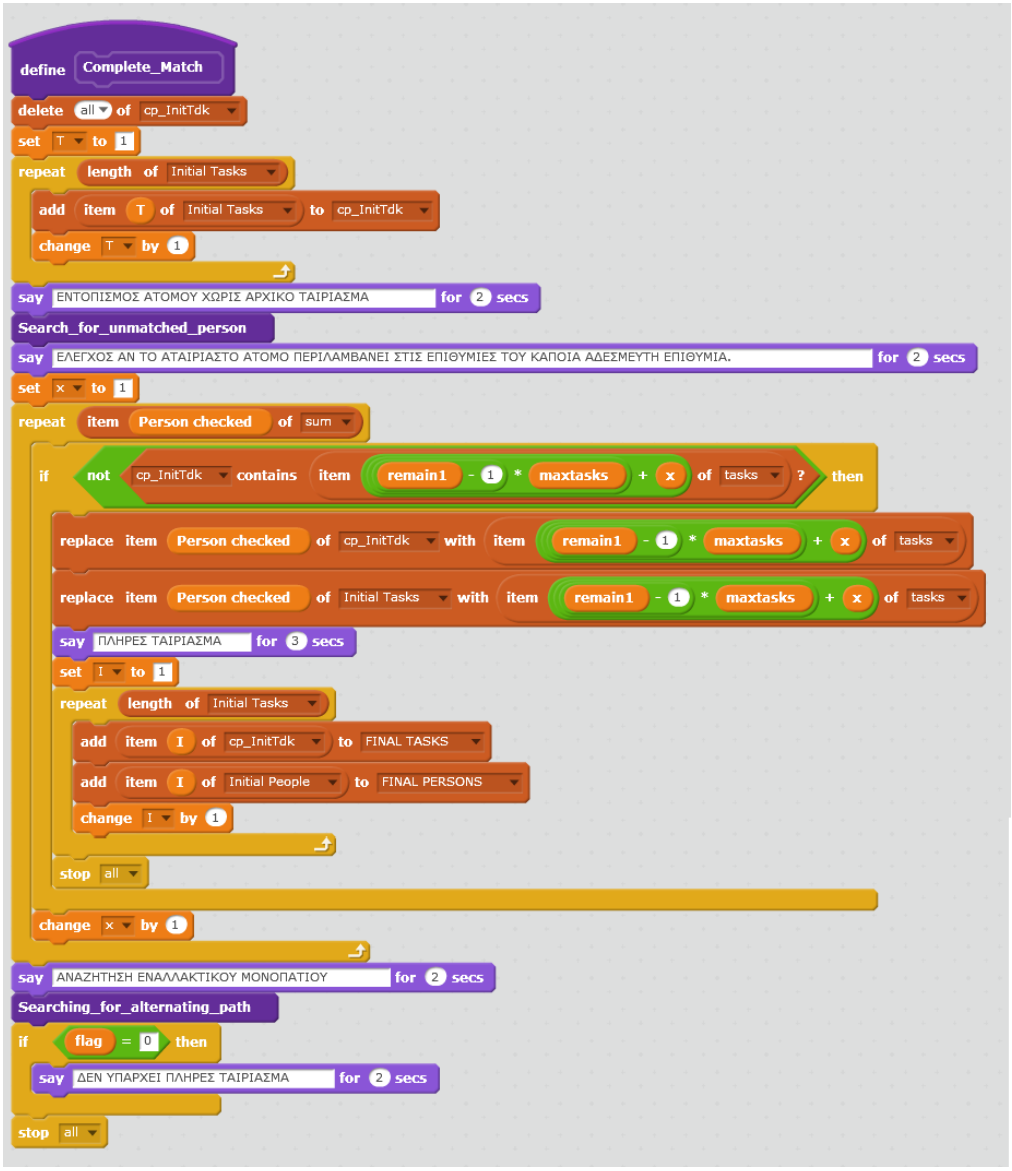
**2**


define Init_match
delete all of Current Task
delete all of Initial People
delete all of Initial Tasks
set p to 1
set I to 1
set T to
repeat length of perso
if not item p of perso = then
add item p of perso to Initial People
add p to Pointer
if not Initial Tasks contains item p - 1 * maxtasks + 1 of tasks ? then
add item p - 1 * maxtasks + 1 of tasks to Initial Tasks
add 1 to Current Task
replace item p of sum with item p of sum - 1
else
add to Initial Tasks
add 0 to Current Task
change p by 1
say ΔΙΑΓΡΑΦΗ ΜΗΔΕΝΙΚΩΝ ΣΥΝΟΛΩΝ ΕΠΙΘΥΜΙΩΝ for 2 secs
Discard_sum_equal_to_zero
set I to 1
if not Initial Tasks contains ? then
say ΠΛΗΡΕΣ ΤΑΙΡΙΑΣΜΑ for 3 secs
repeat length of Initial Tasks
add item I of Initial Tasks to FINAL TASKS
add item I of Initial People to FINAL PERSONS
change I by 1
stop all
else
say ΑΝΑΖΗΤΗΣΗ ΠΛΗΡΟΥΣ ΤΑΙΡΙΑΣΜΑΤΟΣ for 2 secs
Complete_Match

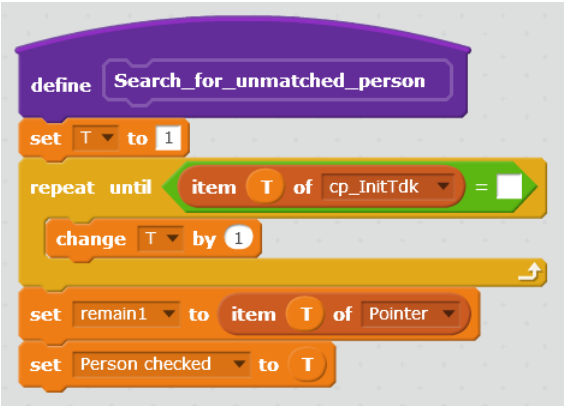
| | |
|---|---|
| **2.1** |  |

define Discard_sum_equal_to_zero
set v to 1
repeat until not sum contains 0 ?
if item v of sum = 0 then
delete v of sum
else
change v by 1

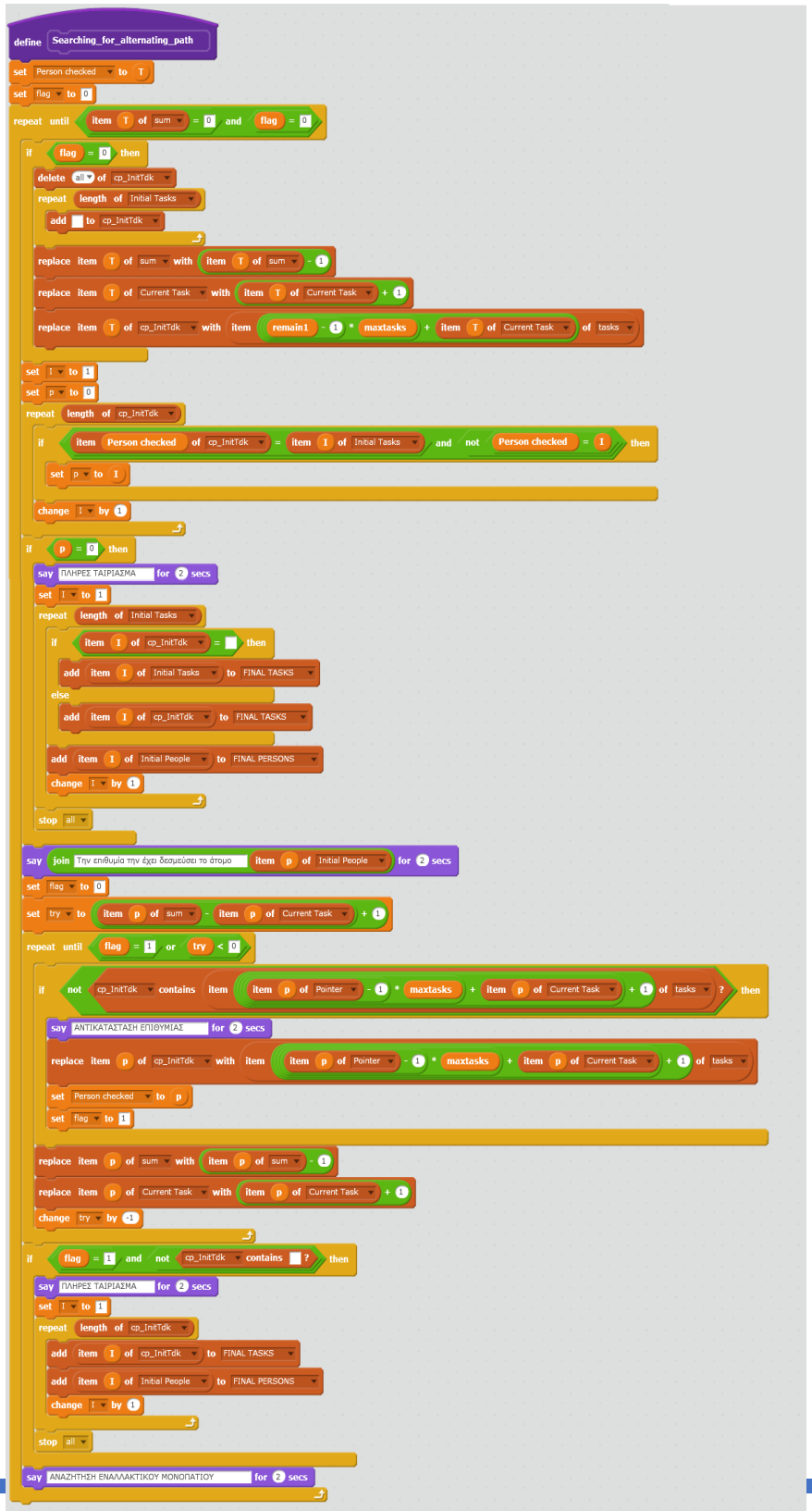
**3**


define Complete_Match
delete all of cp_InitTdk
set T to 1
repeat length of Initial Tasks
add item T of Initial Tasks to cp_InitTdk
change T by 1
say ΕΝΤΟΠΙΣΜΟΣ ΑΤΟΜΟΥ ΧΩΡΙΣ ΑΡΧΙΚΟ ΤΑΙΡΙΑΣΜΑ for 2 secs
Search_for_unmatched_person
say ΕΛΕΓΧΟΣ ΑΝ ΤΟ ΑΤΑΙΡΙΑΣΤΟ ΑΤΟΜΟ ΠΕΡΙΛΑΜΒΑΝΕΙ ΣΤΙΣ ΕΠΙΘΥΜΙΕΣ ΤΟΥ ΚΑΠΟΙΑ ΑΔΕΣΜΕΥΤΗ ΕΠΙΘΥΜΙΑ. for 2 secs
set x to 1
repeat item Person checked of sum
if not cp_InitTdk contains item remain1 - 1 * maxtasks + x of tasks ? then
replace item Person checked of cp_InitTdk with item remain1 - 1 * maxtasks + x of tasks
replace item Person checked of Initial Tasks with item remain1 - 1 * maxtasks + x of tasks
say ΠΛΗΡΕΣ ΤΑΙΡΙΑΣΜΑ for 3 secs
set I to 1
repeat length of Initial Tasks
add item I of cp_InitTdk to FINAL TASKS
add item I of Initial People to FINAL PERSONS
change I by 1
stop all
change x by 1
say ΑΝΑΖΗΤΗΣΗ ΕΝΑΛΛΑΚΤΙΚΟΥ ΜΟΝΟΠΑΤΙΟΥ for 2 secs
Searching_for_alternating_path
if flag = 0 then
say ΔΕΝ ΥΠΑΡΧΕΙ ΠΛΗΡΕΣ ΤΑΙΡΙΑΣΜΑ for 2 secs
stop all

**3.1**


define Search_for_unmatched_person
set T to 1
repeat until item T of cp_InitTdk =
change T by 1
set remain1 to item T of Pointer
set Person checked to T

**3.2**


define Searching_for_alternating_path
set Person checked to T
set flag to 0
repeat until item T of sum = 0 and flag = 0
if flag = 0 then
delete all of cp_InitTdk
repeat length of Initial Tasks
add to cp_InitTdk
replace item T of sum with item T of sum - 1
replace item T of Current Task with item T of Current Task + 1
replace item T of cp_InitTdk with item remain1 - 1 * maxtasks + item T of Current Task of tasks
set I to 1
set p to 0
repeat length of cp_InitTdk
if item Person checked of cp_InitTdk = item I of Initial Tasks and not Person checked = I then
set p to I
change I by 1
if p = 0 then
say ΠΛΗΡΕΣ ΤΑΙΡΙΑΣΜΑ for 2 secs
set I to 1
repeat length of Initial Tasks
if item I of cp_InitTdk = then
add item I of Initial Tasks to FINAL TASKS
else
add item I of cp_InitTdk to FINAL TASKS
add item I of Initial People to FINAL PERSONS
change I by 1
stop all
say join Την επιθυμία την έχει δεσμεύσει το άτομο item p of Initial People for 2 secs
set flag to 0
set try to item p of sum - item p of Current Task + 1
repeat until flag = 1 or try < 0
if not cp_InitTdk contains item item p of Pointer - 1 * maxtasks + item p of Current Task + 1 of tasks ? then
say ΑΝΤΙΚΑΤΑΣΤΑΣΗ ΕΠΙΘΥΜΙΑΣ for 2 secs
replace item p of cp_InitTdk with item item p of Pointer - 1 * maxtasks + item p of Current Task + 1 of tasks
set Person checked to p
set flag to 1
replace item p of sum with item p of sum - 1
replace item p of Current Task with item p of Current Task + 1
change try by -1
if flag = 1 and not cp_InitTdk contains ? then
say ΠΛΗΡΕΣ ΤΑΙΡΙΑΣΜΑ for 2 secs
set I to 1
repeat length of cp_InitTdk
add item I of cp_InitTdk to FINAL TASKS
add item I of Initial People to FINAL PERSONS
change I by 1
stop all
say ΑΝΑΖΗΤΗΣΗ ΕΝΑΛΛΑΚΤΙΚΟΥ ΜΟΝΟΠΑΤΙΟΥ for 2 secs

## ΕΥΧΑΡΙΣΤΙΕΣ

Κλείνοντας την παρούσα εργασία θα θέλαμε να ευχαριστήσουμε τις επιβλέπουσες καθηγήτριές μας, που με υπομονή και συνέπεια προσπάθησαν να μας εισάγουν στον κόσμο των Μαθηματικών και της Πληροφορικής, καθώς και τους γονείς μας, που μας στήριξαν σε αυτήν μας τη προσπάθεια.

## ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Phillip Hall, On Representatives of subsets, J. London Math, Soc., 10 (1): 26-30, 1935.

[2] Grade 6 Math Circles, Graph Theory, Centre for Education in Mathematics and Computing, University of Waterloo, Canada: http://www.cemc.uwaterloo.ca/events/mathcircles/2015-16/Fall/Junior6_Nov18.pdf

[3] K.M. Koh, F.M. Dong, E.G. Tay, Graphs and Their Applications , Mathematical Medley I Volume 33 No. 2, December 2006.

[4] Susie Jameson, Edexcel AS and A Level Modular Mathematics Decision Mathematics 1 D1 (Edexcel GCE Modular Maths), Pearson.

[5] Maximum matching algorithm σε python, c++, Java: http://www.geeksforgeeks.org/maximum-bipartite-matching/

[6] The Euler Archive, http://eulerarchive.maa.org//pages/E053.html