*Αν. Καθηγητής Π. Λουρίδας*

*Τμήμα Διοικητικής Επιστήμης και Τεχνολογίας*

*Οικονομικό Πανεπιστήμιο Αθηνών*

# Χορδές και Διαστήματα

Το Oulipo (Ouvroir de Littérature Potentielle, Εργαστήριο Δυνητικής Λογοτεχνίας) είναι μια ομάδα κυρίως γαλλόφωνων συγγραφέων, η οποία ιδρύθηκε το 1960 από τους Raymond Queneau and François Le Lionnais, τα μέλη της οποίας διερευνούν τη δημιουργία λογοτεχνίας μέσω χρήσης περιορισμών στη διαδικασία της γραφής. Ένα παράδειγμα τέτοιου περιορισμού είναι το *λειπόγραμμα*, δηλαδή η συγγραφή κειμένου χωρίς τη χρήση συγκεκριμένων γραμμάτων. Το πιο διάσημο λειπόγραμμα είναι το βιβλίο του Raymond Queneau *La Disparition* (Η Εξαφάνιση), όπου λείπει το γράμμα e από το σύνολο του βιβλίου (και αυτή είναι και η εν λόγω εξαφάνιση του τίτλου).

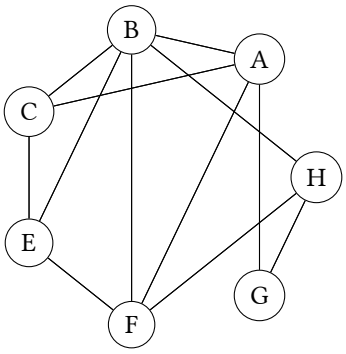
Πριν προχωρήσουμε, να σημειώσουμε ότι μέλη του Oulipo έχουν γράψει σημαντικά λογοτεχνικά έργα, και όχι μόνο λογοτεχνικά ευφειολογήματα. Μπορούμε να σταχυολογίσουμε (σε χρονολογική σειρά έκδοσης) τις *Οι Αόρατες Πόλεις* (Le Città Invisibili, 1972), του Italo Calvino, το *Ζωή, Οδηγίες Χρήσης* (La Vie Mode d'Emploi, 1978), του Georges Perec, και την *Η Ανωμαλία* (L' Anomalie, 2020) του Hervé Le Tellier. Αν θέλετε να καθαρίσετε το μυαλό της κατά τη διάρκεια εκπόνησης της εργασίας, δεν είναι κακή ιδέα να στραφείτε σε αυτά.

Ένα μέλος του Oulipo ήταν ο συγγραφέας και μαθηματικός Claude Berge, ο οποίος έγραψε ένα αφήγημα, Qui a tué le Duc de Densmore? (Ποιος σκότωσε τον Δούκα του Densmore;), μια αστυνομική ιστορία η λύση της οποίας βασίζεται στη θεωρία γράφων. Σύμφωνα με την ιστορία, ο Δούκας του Densmore δολοφονήθηκε, όταν εξεράγη μια βόμβα στο κάστρο του. Στην προσπάθεια για τη διελεύκανση της υπόθεσης, προέκυψε ότι είχε γίνει γνωστό πως ο δούκας στη διαθήκη του θα παραμελούσε μία από τις επτά γυναίκες του, χωρίς να ξέρουμε ποια. Έτσι οι υποψίες στράφηκαν στις πρώην συζύγους του. Προέκυψε ότι στο διάστημα πριν από την έκρηξη, τον επισκέφτηκαν και οι επτά· όχι όλες μαζί, πλην όμως κάποιες συνέπεσαν ταυτόχρονα στο κάστρο:

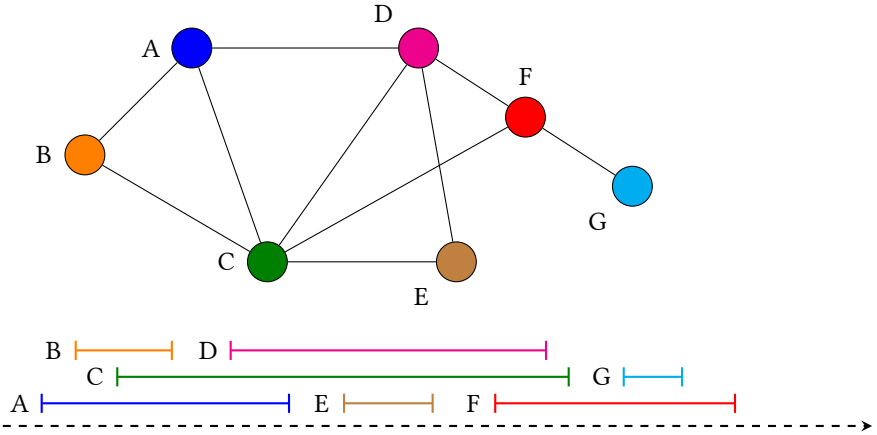
- Η Ann συνάντησε τις Betty, Charlotte, Felicia, Georgia.
- Η Betty συνάντησε τις Ann, Charlotte, Edith, Felicia, Helen.
- Η Charlotte συνάντησε τις Ann, Betty, Edith.
- Η Edith συνάντησε τις Betty, Charlotte, Felicia.
- Η Felicia συνάντησε τις Ann, Betty, Edith, Helen.
- Η Georgia συνάντησε τις Ann, Helen.
- Η Helen συνάντησε τις Betty, Felicia, Georgia.

Με βάση τα παραπάνω, μπορούμε να βρούμε την ένοχο. Πώς;

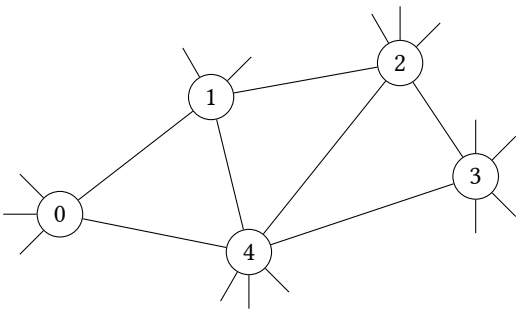
Κατασκευάζουμε ένα γράφο, όπου οι κορυφές του αντιστοιχούν στις επτά γυναίκες, ενώ οι ακμές αντιστοιχούν στις συναντήσεις τους:



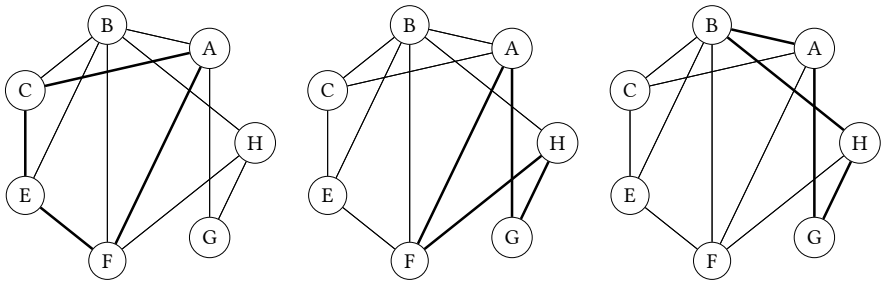

Μπορούμε τώρα να παρατηρήσουμε ότι στον γράφο αυτό οι κορυφές αντιστοιχούν στα χρονικά διαστήματα που οι σύζυγοι επισκέφτηκαν τον Δούκα, οπότε δύο κορυφές συνδέονται αν δύο χρονικά διαστήματα αλλοκαλύπτονται (έστω μερικώς), ώστε οι δύο σύζυγοι να συναντηθούν. Γενικότερα, ένας μη κατευθυνόμενος γράφος που προκύπτει από ένα σύνολο διαστημάτων στην ευθεία των πραγματικών αριθμών, με μία κορυφή για κάθε διάστημα και μια ακμή μεταξύ δύο ακμών που αντιστοιχούν σε τεμνόμενα διαστήματα, ονομάζεται *γράφος διαστημάτων* (interval graph). Η παρακάτω εικόνα (από τη Wikipedia) δείχνει έναν γράφο διαστημάτων, μαζί με τα διαστήματα από τα οποία προκύπτει.



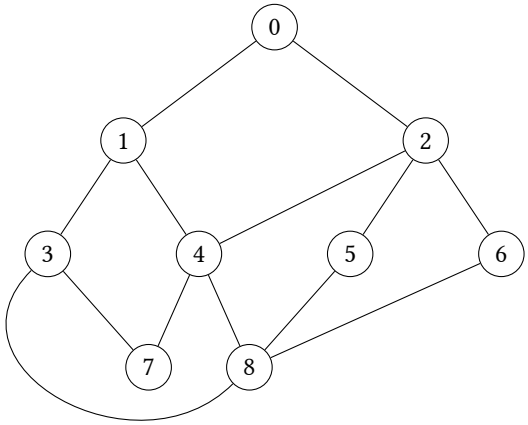

Ένας γράφος διαστημάτων πληροί κάποιες ιδιότητες όσον αφορά τη δομή του. Αν σε έναν γράφο όλοι οι κύκλοι που απαρτίζονται από τέσσερεις ή περισσότερους συνδέσμους έχουν μια *χορδή* (chord), δηλαδή έναν σύνδεσμο που συνδέει δύο κορυφές του κύκλου χωρίς να είναι μέρος του κύκλου, ο γράφος ονομάζεται *χορδικός* (chordal). Στην παρακάτω εικόνα βλέπετε έναν μέρος ενός γράφου, όπου ο κύκλος 01234 έχει δύο χορδές, ο κύκλος 1234 έχει μία χορδή, και ο κύκλος 0124 έχει μία χορδή.




Αποδεικνύεται ότι ένας γράφος διαστημάτων είναι χορδικός γράφος. Αυτό μας δίνει τη λύση στο ποια είναι ένοχη για τη δολοφονία του Δούκα. Αν προσέξουμε, στο γράφο υπάρχουν τρεις κύκλοι, οι ACEF, AFHG, ABHG, οι οποίοι δεν έχουν χορδές. Άρα ο γράφος δεν είναι χορδικός, και επομένως δεν μπορεί να είναι γράφος διαστημάτων. Κάποια έχει πει ψέμματα σχετικά με τις συναντήσεις της. Στους τρεις αυτούς κύκλους, η μόνη κοινή σύζυγος είναι η Ann, επομένως αυτή έχει πει ψέμματα και είναι η ένοχος για τη δολοφονία του δούκα.




Γενικότερα τώρα, εμείς δεν θέλουμε να βρίσκουμε αν ένας γράφος είναι χονδρικός με το μάτι: χρειαζόμαστε έναν αλγόριθμο. Για να δούμε τον αλγόριθμο αυτό, θα πρέπει να θυμηθούμε πως εξερευνούμε έναν γράφο: έχουμε δύο βασικές μεθόδους, την κατά βάθος αναζήτηση (Depth-First Search, DFS) και την κατά πλάτος αναζήτηση (Breadth-First Search, BFS). Αν έχουμε τον παρακάτω γράφο, στην κατά πλάτος αναζήτηση ξεκινώντας από τον κόμβο 0 θα επισκεφτούμε τους κόμβους ανά στρώματα: πρώτα τους κόμβους που απέχουν ένα σύνδεσμο από τον κόμβο 0 (οι κόμβοι 1, 2) μετά τους κόμβους που απέχουν δύο συνδέσμους από τον κόμβο 0 (οι κόμβοι 3, 4, 5,6), και τέλος τους κόμβους που απέχουν τρεις συνδέσμους από τον κόμβο 0 (οι κόμβοι 7, 8).



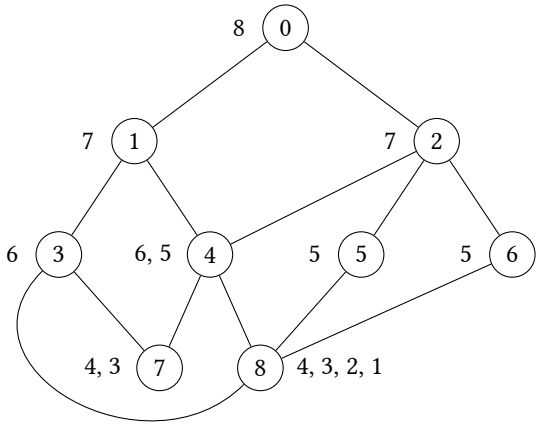

Στην περίπτωση που έχουμε ελευθερία στην επιλογή του επόμενου κόμβου που επισκεφτόμαστε στην κατά πλάτος αναζήτηση, μπορούμε να χρησιμοποιήσουμε τον εξής κανόνα: θα επιλέγουμε τον κόμβο του οποίου έχουμε επισκεφτεί περισσότερους κόμβους που οδηγούν σε αυτόν. Έτσι, στο παράδειγμά μας, όταν εξαντλήσουμε το δεύτερο στρώμα, θα ξεκινήσουμε το τρίτο στρώμα από τον κόμβο 4, γιατί θα έχουμε επισκεφτεί δύο κόμβους (1, 2) που οδηγούν στον κόμβο 4, ενώ θα έχουμε επισκεφτεί έναν κόμβο που οδηγεί σε κάθε ένα από τους άλλους κόμβους του στρώματος. Ομοίως θα εργαστούμε για το τρίτο στρώμα οπότε η σειρά με την οποία θα επισκεφτούμε τους κόμβους θα είναι 0, 1, 2, 4, 3, 5, 6, 8, 7.

Αυτή η τροποποίηση της κατά πλάτος αναζήτησης ονομάζεται *Λεξικογραφική Κατά Πλάτος Αναζήτηση* (Lexicographic BFS, LexBFS). Η ονομασία προκύπτει από το γεγονός ότι ο κανόνας που περιγράψαμε αντιστοιχεί στην επιλογή του μεγαλύτερου λεξικογραφικά κόμβου, αν σε κάθε κόμβο σημειώνουμε μια ετικέτα που προκύπτει από τους κόμβους που έχουμε επισκεφτεί και οδηγούν σε αυτόν. Η λεξικογραφική κατά πλάτος αναζήτηση μπορεί να υλοποιηθεί ως εξής:

```
LexBFSLabelling(G, s) → lexbfs
    Input: G = (V, E), ένας γράφος
           s, ο κόμβος εκκίνησης
    Data: label, πίνακας μεγέθους |V|
    Output: lexbfs, πίνακας μεγέθους |V|
1   foreach u in V do
2       label[u] ← ∅
3       lexbfs[u] ← −1
4   label[s] ← [|V| − 1]
5   for i = |V| − 1 to 0 do
6       u ← PickLargestLabelledVertex(V)
7       lexbfs[u] ← |V| − i
8       foreach v in AdjacencyList(u) do
9           if lexbfs[v] = −1 then
10              Append(label[v], i − 1)
11  return lexbfs
```

Στον αλγόριθμο αυτό, η συνάρτηση `PickLargestLabelledVertex` επιλέγει το κόμβο με τη μεγαλύτερη λεξικογραφικά ετικέτα, από τους κόμβους που δεν έχουμε επισκεφτεί ακόμα. Κάθε φορά που επισκεφτόμαστε έναν κόμβο, προσάπτουμε στην ετικέτα των γειτόνων του που δεν έχουμε επισκεφτεί ένα φθίνοντα αριθμό. Αρχικά οι ετικέτες είναι κενές, εκτός από τον κόμβο εκκίνησης που η ετικέτα του είναι $|V| - 1$.

Αν εφαρμόσουμε τον αλγόριθμο αυτόν στον προηγούμενο γράφο, οι κόμβοι θα πάρουν τις ετικέττες που εμφανίζονται στα δεξιά τους, και οι οποίες μας δίνουν τη λεξικογραφική κατά πλάτος διάταξη που είδαμε.




Το πρόβλημα με τον αλγόριθμο `LexBFSLabelling` είναι η εύρεση κάθε φορά του μεγαλύτερου λεξικογραφικά κόμβου. Αν κάθε φορά αναζητούμε στους κόμβους να τον βρούμε, τότε η αναζήτηση δεν δουλεύει σε γραμμικό χρόνο, όπως δουλεύει η

απλή κατά πλάτος αναζήτηση. Για το λόγο αυτό, δεν χρησιμοποιούμε τον αλγόριθμο `LexBFSLabelling`, αλλά εργαζόμαστε διαφορετικά, βασιζόμενοι σε διαδοχικές διαμερίσεις του συνόλου των κόμβων του γράφου. Στην αρχή βάζουμε όλους τους κόμβους σε ένα σύνολο. Παίρνουμε τον κόμβο εκκίνησης, και μετά διαμερίζουμε το σύνολο σε δύο, το πρώτο με τους γείτονες του κόμβου εκκίνησης (αν έχει), και το δεύτερο με τους υπόλοιπους (αν υπάρχουν). Συνεχίζουμε με τον ίδιο τρόπο για τους υπόλοιπους κόμβους: παίρνουμε τον πρώτο κόμβο από το πρώτο σύνολο και στη συνέχεια ελέγχουμε τους γείτονες και διαμερίζουμε τα υπόλοιπα σύνολα σε γείτονες και μη γείτονες (αν ένα σύνολο να είναι μόνο γείτονες ή μη γείτονες, τότε η διαμέριση είναι το ίδιο σύνολο). Αυτό μπορεί να περιγραφεί ως εξής:

- Φτιάχνουμε μία λίστα $\Sigma$ η οποία περιέχει ένα σύνολο με όλους τους κόμβους του γράφου.
- Όσο η λίστα $\Sigma$ δεν είναι άδεια:
  - Αφαιρούμε έναν κόμβο $u$ από το πρώτο σύνολο της $\Sigma$. Προσθέτουμε τον $u$ στη λεξικογραφική κατά πλάτος διάταξη, που είναι η σειρά με την οποία επισκεφτόμαστε τους κόμβους.
  - Αν το πρώτο σύνολο της $\Sigma$ αδειάσει, το αφαιρούμε από την $\Sigma$.
  - Για κάθε έναν γείτονα $v$ του $u$ που δεν έχουμε επισκεφτεί ακόμα, έστω ότι $\Sigma_v$ είναι το σύνολο στο οποίο ανήκει ο $v$:
    - Αφαιρούμε τον $v$ από το $\Sigma_v$.
    - Τον προσθέτουμε σε ένα σύνολο $\Sigma'_v$, στο οποίο συλλέγουμε όλους τους γείτονες του $u$ που ανήκουν στο $\Sigma_v$.
  - Εισάγουμε το $\Sigma'_v$ πριν από το $\Sigma_v$ στη $\Sigma$.
  - Αν το $\Sigma_v$ άδειασε, το αφαιρούμε από τη $\Sigma$.
- Επιστρέφουμε τη λεξικογραφική κατά πλάτος σειρά με την οποία επισκεφτήκαμε τους κόμβους.

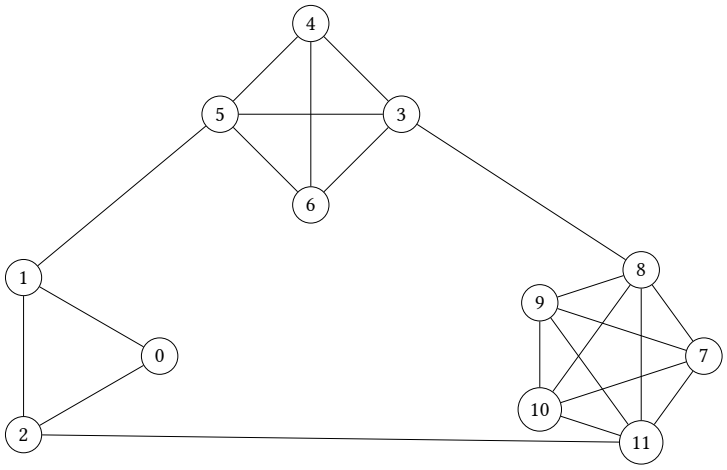
Αν ακολουθήσουμε τα παραπάνω βήματα, τότε ξεκινώντας από το αρχικό σύνολο με όλους τους κόμβους θα προχωρήσουμε όπως φαίνεται παρακάτω, όπου σε κάθε γραμμή έχουμε την μέχρι τώρα σειρά των κόμβων που έχουμε επισκεφτεί και τη λίστα $\Sigma$ με τις διαμερίσεις. Μπορούμε να παρατηρήσουμε ότι το αποτέλεσμα είναι το ίδιο με αυτό του μηχανισμού με τις ετικέτες, αφού κάθε φορά που επισκεφτόμαστε έναν κόμβο οι γείτονές του μετακινούνται πιο μπροστά στη λίστα που περιέχει τους κόμβους που δεν έχουμε επισκεφτεί ακόμα. Ταυτόχρονα, κάθε διαμέριση εξασφαλίζει ότι πρώτα θα επισκεφτούμε τους γείτονες του κόμβου που βρισκόμαστε και αργότερα θα επισκεφτούμε τους κόμβους με τους οποίους δεν γειτονεύει.

```
() [{0, 1, 2, 3, 4, 5, 6, 7, 8}]
(0) [{1, 2}, {3, 4, 5, 6, 7, 8}]
(0, 1) [{2}, {3, 4}, {5, 6, 7, 8}]
(0, 1, 2) [{4}, {3}, {5, 6}, {7, 8}]
(0, 1, 2, 4) [{3}, {5, 6}, [7, 8}]
```

```
(0, 1, 2, 4, 3) [{5, 6}, {7, 8}]
(0, 1, 2, 4, 3, 5) [{6}, {8}, {7}]
(0, 1, 2, 4, 3, 5, 6) [{8}, {7}]
(0, 1, 2, 4, 3, 5, 6, 8) [{7}]
(0, 1, 2, 4, 3, 5, 6, 8, 7) []
```

Έχοντας στη διάθεση μας τη λεξικογραφική κατά πλάτος αναζήτηση, μπορούμε να τη χρησιμοποιήσουμε για να αποφανθούμε αν ένας γράφος είναι χορδικός. Πριν το κάνουμε αυτό, θα δούμε δύο επιπλέον έννοιες στους γράφους.

Μία *κλίκα* (clique) είναι ένα υποσύνολο των κόμβων ενός γράφου που αποτελούν έναν πλήρη γράφο. Στην παρακάτω εικόνα έχουμε έναν γράφο με τρεις κλίκες, με τρεις, τέσσερεις, και πέντε κόμβους.




Μια *διάταξη τέλειας εξάλειψης* (perfect elimination ordering) είναι μια διάταξη των κόμβων του γράφου τέτοια ώστε για κάθε κόμβο $u$ στη διάταξη, ο γράφος που προκύπτει από τους γείτονες του $u$ που ακολουθούν στη διάταξη είναι κλίκα.

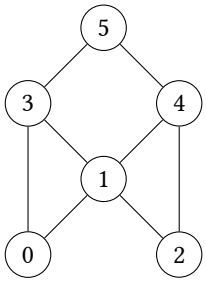
Ένας γράφος είναι χορδικός αν οι κόμβοι του μπορούν μπουν σε μία διάταξη τέλειας εξάλειψης. Αποδεικνύεται ότι η ανίστροφη μιας λεξικογραφικής κατά πλάτος διάταξης είναι μια διάταξη τέλειας εξάλειψης. Επομένως, για να αποφανθούμε αν ένας γράφος είναι χορδικός, αρκεί να βρούμε μια λεξικογραφική κατά πλάτος διάταξη των κόμβων του, να την αντιστρέψουμε, και να ελέγξουμε αν αυτή είναι διάταξη τέλειας εξάλειψης:

- Για κάθε κόμβο $u$ σε αντίστροφη λεξικογραφική κατά πλάτος διάταξη:
  - Έστω $v$ ο πρώτος γείτονας του $u$ στην αντίστροφη λεξικογραφική κατά πλάτος διάταξη, αν υπάρχει:
    - Έστω $RN(u)$ οι γείτονες του $u$ που έπονται του $u$ στην αντίστροφη λεξικογραφική κατά πλάτος διάταξη, και $RN(v)$ αντίστοιχα οι γείτονες του $v$ που έπονται του $v$ στην αντίστροφη λεξικογραφική κατά πλάτος διάταξη.

    - Αν $RN(u) \setminus \{v\} \not\subseteq RN(v)$, τότε ο γράφος δεν είναι χορδικός, και σταματάμε.
- Αφού δεν προέκυψε το αντίθετο, ο γράφος είναι χορδικός.

Έχουμε λοιπόν στη διάθεσή μας τον τρόπο με τον οποίο μπορούμε να διαπιστώσουμε αν ένας γράφος είναι χορδικός, που είναι απαίτηση προκειμένου να διαπιστώσουμε αν ο γράφος είναι γράφος διαστημάτων. Πλην όμως, η απαίτηση δεν είναι επαρκής. Χρειάζεται κάτι επιπλέον.

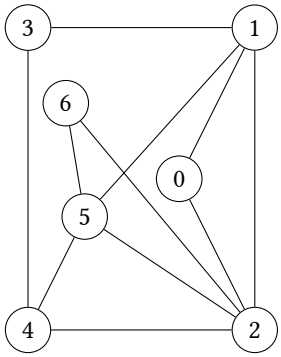
Ένα σύνολο κόμβων ενός γράφου είναι ένα *ανεξάρτητο σύνολο κόμβων* (independent vertex set) αν κανένας από τους κόμβους του συνόλου δεν γειτονεύει με άλλον κόμβο του συνόλου. Μια *αστεροειδής τριάδα* (asteroidal triple) είναι ένα ανεξάρτητο σύνολο τριών κόμβων όπου κάθε δύο από αυτούς συνδέονται με ένα μονοπάτι που δεν περιλαμβάνει τους γείτονες του τρίτου. Στον παρακάτω γράφο, οι κόμβοι 0, 2, 5 είναι ανεξάρτητοι και ανά δύο συνδέονται με μονοπάτια που αποφεύγουν τη γειτονιά του τρίτου: $0-3-5$, $0-1-2$, $2-4-5$.




Ένας γράφος είναι χωρίς αστεροειδείς τριάδες (asteroidal triple-free, AT-free) αν δεν περιέχει καμμία αστεροειδή τριάδα. Τώρα είμαστε σε θέση να μπορούμε να αποφανθούμε αν ένας γράφος είναι γράφος διαστημάτων: *ένας γράφος είναι γράφος διαστημάτων αν και μόνο αν είναι χορδικός και δεν έχει αστεροειδείς τριάδες.* Πρέπει λοιπόν να βρούμε έναν τρόπο με τον οποίο να αναγνωρίζουμε αν ένας γράφος $G = (V, E)$ δεν έχει αστεροειδείς τριάδες. Ας συμβολίσουμε με $C^u(v)$ τη συνιστώσα (connected component) του γράφου $G \setminus N(u)$ που περιέχει τον κόμβο $v$, όπου $N(u)$ οι γείτονες του $u$. Τότε, μία ανεξάρτητη τριάδα κόμβων $u$, $v$, $w$ του γράφου $G$ είναι μία αστεροειδής τριάδα αν και μόνο αν $C^u(v) = C^u(w)$, $C^v(u) = C^v(w)$, και $C^w(v) = C^w(u)$.

- Βρίσκουμε, χρησιμοποιώντας την απλή κατά πλάτος αναζήτηση, τις συνιστώσες που προκύπτουν από τον γράφο $G$ αφού αφαιρέσουμε κάθε έναν κόμβο $u$ και τους γείτονες του από τον $G$, δηλαδή τις συνιστώσες του $G \setminus N(u)$ για κάθε $u$.
- Κατασκευάζουμε έναν πίνακα $C$ διαστάσεων $|V| \times |V|$, όπου στο κελί $u, v$ του $C$ βάζουμε 0 αν οι κόμβοι $u$ και $v$ είναι γείτονες, διαφορετικά βάζουμε τη συνιστώσα του $G \setminus N(u)$ στην οποία ανήκει ο κόμβος $v$.
- Για να μην έχει ο γράφος αστεροειδείς τριάδες, θα πρέπει για καμμία ανεξάρτητη τριάδα $u, v, w$ να μην ισχύει $C[u, v] = C[u, w]$ και $C[v, u] = C[v, w]$ και $C[w, u] = C[w, v]$.

Αν έχουμε τον παρακάτω γράφο:




τότε ο πίνακας $C$, εφόσον ονομάσουμε τις συνιστώσες με διαδοχικά γράμματα του αλφαβήτου, θα είναι ο:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | *a* | *a* | *a* | *a* |
| 1 | 0 | 0 | 0 | 0 | *b* | 0 | *c* |
| 2 | 0 | 0 | 0 | *d* | 0 | 0 | 0 |
| 3 | *e* | 0 | *e* | 0 | 0 | *e* | *e* |
| 4 | *f* | *f* | 0 | 0 | 0 | 0 | *g* |
| 5 | *h* | 0 | 0 | *i* | 0 | 0 | 0 |
| 6 | *j* | *j* | 0 | *j* | *j* | 0 | 0 |

Προσέξτε ότι μια συνιστώσα εμφανίζεται ξεχωριστά για κάθε κόμβο από τον οποίο προκύπτει. Έτσι, η συνιστώσα $d$ που προκύπτει από τον γράφο $G \setminus N(2)$ είναι ίση με τη συνιστώσα $i$ που προκύπτει από το γράφο $G \setminus N(5)$.

Αντικείμενο της εργασίας είναι για έναν γράφο που σας δίνεται να μπορείτε να εκτελέσετε λεξικογραφική κατά πλάτος αναζήτηση, να αποφανθείτε αν είναι χορδικός, και αν είναι γράφος διαστημάτων.

Προσοχή! Έχουν αναπτυχθεί πιο αποτελεσματικοί αλγόριθμοι τόσο για τον εντοπισμό αστερεοειδών τριάδων όσο και για την αναγνώριση γράφων διαστημάτων. Εσείς όμως θα πρέπει να υλοποιήσετε τον αλγόριθμο που περιγράφεται στην παρούσα εκφώνηση.

## Απαιτήσεις Προγράμματος

Κάθε φοιτητής θα εργαστεί σε αποθετήριο στο GitHub. Για να αξιολογηθεί μια εργασία θα πρέπει να πληροί τις παρακάτω προϋποθέσεις:

- Για την υποβολή της εργασίας θα χρησιμοποιηθεί το ιδιωτικό αποθετήριο του φοιτητή που δημιουργήθηκε για τις ανάγκες του μαθήματος και του έχει αποδοθεί. Το αποθετήριο αυτό έχει όνομα του τύπου `username-algo-assignments`, όπου `username` είναι το όνομα του φοιτητή στο GitHub. Για παράδειγμα, το σχετικό αποθετήριο του διδάσκοντα θα

ονομαζόταν `louridas-algo-assignments` και θα ήταν προσβάσιμο στο https://github.com/dmst-algorithms-course/louridas-algo-assignments. *Τυχόν άλλα αποθετήρια απλώς θα αγνοηθούν.*

- Μέσα στο αποθετήριο αυτό θα πρέπει να δημιουργηθεί ένας κατάλογος `assignment-2023-2`.
- Μέσα στον παραπάνω κατάλογο το πρόγραμμα θα πρέπει να αποθηκευτεί με το όνομα `interval_graphs.py`.
- Δεν επιτρέπεται η χρήση έτοιμων βιβλιοθηκών γράφων ή τυχόν έτοιμων υλοποιήσεων των αλγορίθμων, ή τμημάτων αυτών, εκτός αν αναφέρεται ρητά ότι επιτρέπεται.
- Επιτρέπεται η χρήση δομών δεδομένων της Python όπως στοίβες, λεξικά, σύνολα, κ.λπ.
- Επιτρέπεται η χρήση των παρακάτω βιβλιοθηκών ή τμημάτων τους όπως ορίζεται:
  - `sys.argv`
  - `argparse`
  - `deque`
- Το πρόγραμμα θα πρέπει να είναι γραμμένο σε Python 3.
- Υπάρχουν διάφοροι τρόποι να ελεχθεί αν ένας γράφος είναι γράφος διαστημάτων· εσείς θα πρέπει να υλοποιήσετε τον τρόπο που περιγράφεται εδώ. Υλοποιήσεις άλλων μεθόδων δεν θα γίνουν δεκτές.
- Ο αλγόριθμος για τη λεξικογραφική κατά πλάτος αναζήτηση απαιτεί να εισάγονται και να αφαιρούνται στοιχεία από τη λίστα των διαμερίσεων, σε οποιοδήποτε σημείο της. Οι λίστες της Python δεν υλοποιούνται μέσω συνδέσμων, αλλά μέσω πινάκων, που σημαίνει ότι η χρήση τους θα επιφέρει επιπλέον κόστος στον αλγόριθμο γιατί η εισαγωγή και η διαγραφή στοιχείων εν μέσω της λίστας δεν απαιτεί χρόνο $O(1)$ όπως θα θέλαμε (απαιτεί χρόνο $O(1)$ όταν εισάγουμε ή αφαιρούμε από το τέλος της λίστας). Ενώ μπορείτε να χρησιμοποιήσετε λίστες της Python ως πρώτη προσέγγιση, για να είναι ο αλγόριθμος αποτελεσματικός θα πρέπει να σκεφτείτε τι και πώς να χρησιμοποιήσετε για να δουλεύει ο αλγόριθμος ως υλοποιημένος με συνδεδεμένες λίστες.
- Σε σχέση με το παραπάνω σημείο, να έχετε πάντα υπόψη σας ότι «η πρόωρη βελτιστοποίηση είναι η ρίζα όλων των κακών» ή αγγλιστί "premature optimization is the root of all evil", Knuth (Δεκεμβρίου 1974).
- Η έξοδος του προγράμματος θα πρέπει να περιλαμβάνει μόνο ό,τι φαίνεται στα παραδείγματα. *Η φλυαρία δεν επιβραβεύεται.*

Το πρόγραμμα θα καλείται ως εξής (όπου `python` η κατάλληλη εντολή στο εκάστοτε σύστημα):

```
python interval_graphs.py task input_filename
```

Η σημασία των παραμέτρων είναι η εξής:

- Η παράμετρος `task` υποδεικνύει τι θέλουμε να εκτελέσει το πρόγραμμα και οι επιτρεπτές τιμές:
  - `lexbfs`: το πρόγραμμα θα εμφανίσει τη λεξικογραφική κατά πλάτος διάταξη των κόμβων του γράφου.
  - `chordal`: το πρόγραμμα θα εμφανίσει αν ο γράφος είναι χορδικός ή όχι.
  - `interval`: το πρόγραμμα θα εμφανίζει αν ο γράφος είναι γράφος διαστημάτων ή όχι.
- Η παράμετρος `input_filename` δίνει το όνομα του αρχείου που περιγράφει το γράφο. Το αρχείο αποτελείται από γραμμές της μορφής `x y` που δείχνουν ότι υπάρχει μη κατευθυνόμενος σύνδεσμος μεταξύ των κόμβων `x` και `y`.

## Παραδείγματα

### Παράδειγμα 1

Αν ο χρήστης του προγράμματος δώσει:

```
python interval_graphs.py lexbfs example_1.txt
```

τότε το πρόγραμμα θα διαβάσει το αρχείο example_1.txt. Τα περιεχόμενα του αρχείου περιγράφουν τον γράφο των εννέα κόμβων με τον οποίο περιγράψαμε την λεξικογραφική κατά πλάτος αναζήτηση. Στην έξοδο το πρόγραμμα θα εμφανίσει:

```
[0, 1, 2, 4, 3, 5, 6, 8, 7]
```

### Παράδειγμα 2

Αν ο χρήστης του προγράμματος δώσει:

```
python interval_graphs.py lexbfs example_2.txt
```

τότε το πρόγραμμα θα διαβάσει το αρχείο example_2.txt και θα εμφανίσει στην έξοδο:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

### Παράδειγμα 3

Αν ο χρήστης του προγράμματος δώσει:

```
python interval_graphs.py chordal example_2.txt
```

τότε το πρόγραμμα θα διαβάσει το αρχείο example_2.txt και θα εμφανίσει στην έξοδο:

```
True
```

**Παράδειγμα 4**

Αν ο χρήστης του προγράμματος δώσει:

```
python interval_graphs.py interval example_2.txt
```

τότε το πρόγραμμα θα διαβάσει το αρχείο example_2.txt και θα εμφανίσει στην έξοδο:

```
True
```

**Παράδειγμα 5**

Αν ο χρήστης του προγράμματος δώσει:

```
python interval_graphs.py lexbfs example_3.txt
```

τότε το πρόγραμμα θα διαβάσει το αρχείο example_3.txt και θα εμφανίσει στην έξοδο:

```
[0, 2, 6, 7, 3, 4, 5, 1]
```

Ο γράφος αυτός είναι γράφος διαστημάτων (και άρα και χορδικός).

**Παράδειγμα 6**

Αν ο χρήστης του προγράμματος δώσει:

```
python interval_graphs.py interval example_4.txt
```

τότε το πρόγραμμα θα διαβάσει το αρχείο example_4.txt και θα εμφανίσει στην έξοδο:

```
True
```

**Παράδειγμα 7**

Αν ο χρήστης του προγράμματος δώσει:

```
python interval_graphs.py interval example_5.txt
```

τότε το πρόγραμμα θα διαβάσει το αρχείο example_5.txt και θα εμφανίσει στην έξοδο:

```
False
```

Επιπλέον, ο γράφος δεν είναι χορδικός.

**Παράδειγμα 8**

Αν ο χρήστης του προγράμματος δώσει:

```
python interval_graphs.py chordal example_6.txt
```

τότε το πρόγραμμα θα διαβάσει το αρχείο `example_6.txt` και θα εμφανίσει στην έξοδο:

```
True
```

**Παράδειγμα 9**

Αν ο χρήστης του προγράμματος δώσει:

```
python interval_graphs.py interval example_6.txt
```

τότε το πρόγραμμα θα διαβάσει το αρχείο `example_6.txt` και θα εμφανίσει στην έξοδο:

```
False
```

## Περισσότερες Πληροφορίες

Οι γράφοι διαστημάτων έχουν εφαρμογές στη βιολογία, στην κατανομή πόρων, στο σχεδιασμό διαδικασιών, γενικότερα στην επιχειρησιακή έρευνα, και στην επιστήμη των υπολογιστών. Η θεωρία τους αναπτύχθηκε από ερευνητές στην RAND Corporation, βλ. Cohen (1978). Η λεξικογραφική κατά πλάτος αναζήτηση, εκτός από τον εντοπισμό χορδικών γράφων, έχει και άλλες εφαρμογές, όπως τον χρωματισμό γράφων· πάντως ο εντοπισμός χορδικών γράφων ήταν το αρχικό κίνητρο, βλ. Rose, Tarjan και Lueker (1976). Μια επισκόπηση μπορείτε να δείτε στο Corneil 2005. Για την αναγνώριση γράφων χωρίς αστεροειδείς τριάδες, όπου μπορείτε να δείτε αποτελεσματικότερους αλγόριθμους από αυτόν που περιγράψαμε εδώ, βλ. Köhler (2004). Συγκεκριμένα, ο αλγόριθμος για την αναγνώριση γράφων χωρίς αστεροειδείς τριάδες έχει πολυπλοκότητα $O(n^3)$, όπου $n$ ο αριθμός των κόμβων, αφού ελέγχουμε κάθε τριάδα κόμβων. Γενικότερα, για να αποφανθούμε αν ένας γράφος είναι γράφος διαστημάτων ή όχι, μπορούμε να εργαστούμε διαφορετικά, με αλγόριθμους οι οποίοι έχουν πιο περίπλοκη λογική, αλλά πολύ καλύτερη πολυπλοκότητα. Ο πρώτος αλγόριθμος που αναγνωρίζει γράφους διαστημάτων σε γραμμικό χρόνο παρουσιάστηκε από τους Booth και Lueker (1976). Στη συνέχεια παρουσιάστηκαν και άλλοι βελτιωμένοι αλγόριθμοι, βλ. Hsu (1993), Habib κ.ά. (2000), Corneil, Olariu και Stewart (2010).

Για περισσότερα σχετικά με τις σχέσεις μεταξύ μαθηματικών και λογοτεχνίας, δείτε το εξαιρετικό βιβλίο της Sarah Hart (2023).

# Αναφορές


Booth, Kellogg S. και George S. Lueker. 1976. “Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms”. *Journal of Computer and System Sciences* 13 (3): 335–379.

Cohen, Joel E. 1978. *Food Webs and Niche Space. (MPB-11), Volume 11.* Princeton University Press.

Corneil, Derek G. 2005. “Lexicographic Breadth First Search—A Survey”. Στο *Graph-Theoretic Concepts in Computer Science,* επιμέλεια υπό Juraj Hromkovič, Manfred Nagl και Bernhard Westfechtel, 1–19. Berlin, Heidelberg: Springer Berlin Heidelberg.

Corneil, Derek G., Stephan Olariu και Lorna Stewart. 2010. “The LBFS Structure and Recognition of Interval Graphs”. *SIAM Journal on Discrete Mathematics* 23 (4): 1905–1953.

Habib, Michel, Ross McConnell, Christophe Paul και Laurent Viennot. 2000. “Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing”. *Theoretical Computer Science* 234 (1): 59–84. ISSN: 0304-3975.

Hart, Sarah. 2023. *Once Upon a Prime: The Wondrous Connections Between Mathematics and Literature.* London: Mudlark.

Hsu, Wen-Lian. 1993. “A simple test for interval graphs”. Στο *Graph-Theoretic Concepts in Computer Science,* επιμέλεια υπό Ernst W. Mayr, 11–16. Berlin, Heidelberg: Springer Berlin Heidelberg.

Knuth, Donald E. Δεκεμβρίου 1974. “Structured Programming with Go to Statements”. *ACM Computing Surveys* (New York, NY, USA) 6, αριθμός 4 (): 261–301.

Köhler, Ekkehard. 2004. “Recognizing graphs without asteroidal triples”. The 26th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2000), *Journal of Discrete Algorithms* 2 (4): 439–452.

Rose, Donald J., R. Endre Tarjan και George S. Lueker. 1976. “Algorithmic Aspects of Vertex Elimination on Graphs”. *SIAM Journal on Computing* 5 (2): 266–283.

Καλή Επιτυχία!