# Αλγόριθμοι και Πολυπλοκότητα

## 3η σειρά γραπτών και προγραμματιστικών ασκήσεων

*CoReLab*

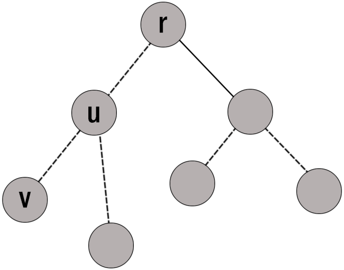
ΣΗΜΜΥ ΕΜΠ

Ιανουάριος 2017

# Outline

# Άσκηση 1α: Εφαρμογές *BFS* – *DFS*

**ΙΔΕΑ**: Το $T$ δέντρο άρα υπάρχει μοναδικό μονοπάτι από τον $r$ στον $v$ που περνάει από τον $u$.



# Άσκηση 1α: Εφαρμογές *BFS* – *DFS*

Στον αλγόριθμο *DFS* κάθε χρονική στιγμή έχουμε τριών ειδών κορυφές:

- Ανεξερεύνητη
- Υπό Εξέταση
- Εξερευνημένη

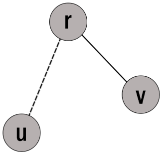
- Όταν $u$ ανεξερεύνητη $\Rightarrow$ $v$ ανεξερεύνητη
- Όταν $u$ υπό εξέταση $\Rightarrow$ $v$ υπό εξέταση ή εξερευνημένη
- Όταν $u$ εξερευνημένη $\Rightarrow$ $v$ εξερευνημένη

Έστω $s(u) \rightarrow$ η χρονική στιγμή που ο κόμβος $u$ έγινε υπό εξέταση.
Έστω $d(u) \rightarrow$ η χρονική στιγμή που ο κόμβος $u$ έγινε εξερευνημένος.

Αν $u$ πρόγονος του $v \Rightarrow s(u) < s(v)$ και $d(u) > d(v)$.

# Άσκηση 1α: Εφαρμογές *BFS* – *DFS*

Αντίστροφα: Αν ο $u$ δεν είναι πρόγονος του $v$.




Όταν $u$ υπό εξέταση $\Rightarrow$ $v$ ανεξερεύνητη.
Επομένως, $d(u) < s(v)$.

$u$ πρόγονος του $v \Leftrightarrow s(u) < s(v)$ και $d(u) > d(v)$.

- Προεπεξεργασία: Κάνουμε *DFS* και κρατάμε τα $s(u), d(u)$ για κάθε κόμβο $u$. Πολυπλοκότητα: $\mathcal{O}(n)$.
- Ερώτημα: Εξετάζουμε αν $s(u) < s(v)$ και $d(u) > d(v)$. Πολυπλοκότητα: $\mathcal{O}(1)$.

# Άσκηση 1β: Εφαρμογές *BFS* – *DFS*

Έστω $P_1, P_2$ δύο διαφορετικά μονοπάτια (όχι ξένα μεταξύ τους) από τον $s$ στον $t$ τ.ω. $|P_1| \rightarrow$ άρτιος και $|P_2| \rightarrow$ περιττός.

Το $G$ **Ισχυρά Συνεκτικό** $\Rightarrow$ υπάρχει μονοπάτι $P$ από τον $s$ στον $t$.
Χ.β.τ.γ. $|P|$ άρτιος $\Rightarrow$ υπάρχει κλειστή διαδρομή $(PP_2)$ που ξεκινάει από τον $s$ και καταλήγει στον $s$ και $|PP_2|$ περιττός.

<u>ΠΡΟΣΟΧΗ:</u> $(PP_2)$ κλειστή διαδρομή όχι απαραίτητα απλός κατευθυνόμενος κύκλος.

Η $(PP_2)$ αποτελείται από πολλούς απλούς κατευθυνόμενους κύκλους $C_1, C_2, \cdots, C_k$.
Αν κάθε $|C_i|$ άρτιος $\Rightarrow$ $(PP_2)$ άρτιος, ΑΤΟΠΟ.

# Άσκηση 1β: Εφαρμογές *BFS* – *DFS*

Αλγοριθμική Ιδέα: Αν βρούμε ότι από κόμβο $s$ υπάρχει περιττό και άρτιο μονοπάτι προς κόμβο $t \Rightarrow$ περιττός κύκλος.

Αποδοτική Υλοποίηση Με *BFS*:

- Κάνουμε *BFS* από τυχαίο κόμβο $s$.
- Κάθε κόμβος $u$, όταν εξερευνείται, παίρνει ένα χρώμα, **ΑΣΠΡΟ** ή **ΜΑΥΡΟ**.
- Κατά την εξέταση κάθε κόμβου $u$ εξευρευνούνται τα παιδιά του και:
  - αν δεν έχουν χρώμα, παίρνουν το αντίθετο από τον $u$.
  - αν κάποιο παιδί $v$ έχει ήδη χρώμα ίδιο με αυτό του $u$, απαντάμε **ΝΑΙ**.

# Άσκηση 1β: Εφαρμογές *BFS* – *DFS*

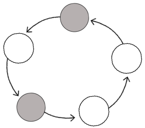
Ορθότητα: ($\Rightarrow$)

- Οι κόμβοι που 'βάφονται' με **ΑΣΠΡΟ** έχουν μονοπάτι άρτιου μήκους από τον $s$.
- Οι κόμβοι που 'βάφονται' με **ΜΑΥΡΟ** έχουν μονοπάτι περιττού μήκους από τον $s$.
- Όλοι οι κόμβοι θα 'βαφτούν' με κάποιο χρώμα γιατί το $G$ **Ισχυρά Συνεκτικό**.
- Αν βρεθεί κόμβος που μπορεί να 'βαφτεί' είτε με **ΑΣΠΡΟ** είτε με **ΜΑΥΡΟ** $\Rightarrow$ υπάρχει κύκλος περιττού μήκους.

# Άσκηση 1β: Εφαρμογές *BFS* – *DFS*

Ορθότητα: ($\Leftarrow$)

- Έστω ότι υπάρχει κύκλος $C$ περιττού μήκους.



- $G$ **Ισχυρά Συνεκτικό** $\Rightarrow$ Όλοι οι κόμβοι του $C$ παίρνουν κάποιο χρώμα.
- Όπως και να χρωματιστεί ο $C$, υπάρχει πάντα ακμή $(u, v)$ με ίδιο χρώμα στα άκρα της.

Πολυπλοκότητα: $\mathcal{O}(|V| + |E|)$.

# Άσκηση 2: Συνάρτηση Κόστους σε Κατευθ. Γραφήματα

Έστω κατευθυνόμενο γράφημα όπου κάθε κορυφή έχει μια ακέραια τιμή $p_u > 0$.
Ορίζουμε ως **κόστος** κάθε κορυφής:

$$\mathbf{c(u)} = \text{τιμή της φθηνότερης κορυφής που είναι προσπελάσιμη από τη } u \text{ (συμπεριλαμβανομένης της } u\text{)}$$

**Είσοδος:** Κατευθυν. γράφημα $G(V, E)$ με τιμές $p_u$, $\forall u \in V$
**Ζητούμενο:** Αλγόριθμος **γραμμικού χρόνου** που υπολογίζει το $c(u)$ για κάθε κορυφή $u$

**(α)** DAG

**(β)** Γενίκευση για κάθε κατευθυνόμενο γράφημα $G$

# Άσκηση 2: Συνάρτηση Κόστους σε Κατευθ. Γραφήματα

Έστω μια κορυφή $u$.
Τότε όλες οι κορυφές που είναι προσπελάσιμες από τη $u$ είναι οι εξής:

- η ίδια η κορυφή $u$
- οι κορυφές με τις οποίες συνδέεται με ακμή $u \to v_i$: $v_1, \ldots, v_d$
- οι κορυφές που είναι προσπελάσιμες από τις $v_1, \ldots, v_d$

Συνοπτικά γράφουμε:

$$c(u) = min\{p_u, min_{v:(u,v)\in E}\{c(v)\}\}$$

# Άσκηση 2: Συνάρτηση Κόστους σε Κατευθ. Γραφήματα

## (α) Αλγόριθμος για DAG

Θέλουμε να βρούμε ένα τρόπο

- να υπολογίσουμε πρώτα όλες τις τιμές $c(v)$, $\forall v : (u, v) \in e$ και
- μετά να κρατήσουμε τη μικρότερη τιμή $min\{p_u, min_{v:(u,v)\in E}\{c(v)\}\}$

Θα θέλαμε να χειριστούμε τις κορυφές με μια *συγκεκριμένη σειρά*.

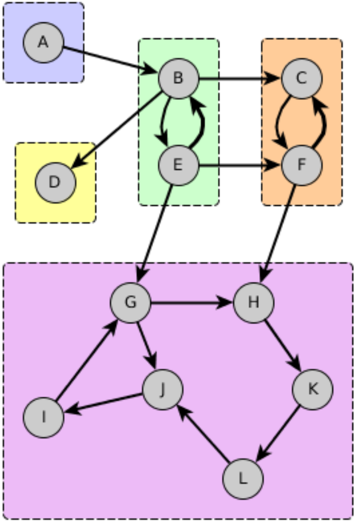
$$\Downarrow$$

Επειδή το γράφημα είναι **DAG** , μπορούμε να χρησιμοποιήσουμε **τοπολογική διάταξη** και έπειτα να εξετάσουμε τις κορυφές στην αντίστροφη σειρά.

### Αλγόριθμος

1. Τοπολογική διάταξη των κορυφών στο $G$
2. Εξετάζοντας τις κορυφές σε αντίστροφη σειρά από την τοπολογική διάταξη:
   Για κάθε κορυφή $u \in V$ υπολογίζουμε
   $$c(u) = min\{p_u, min_{v:(u,v)\in E}\{c(v)\}\}$$

*Πολυπλοκότητα*: Και τα δύο βήματα απαιτούν χρόνο $O(|V| + |E|)$, άρα γραμμικός αλγόριθμος

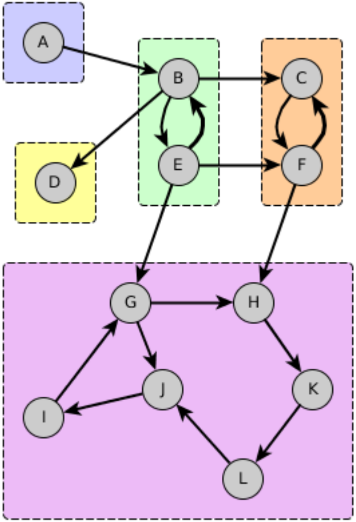


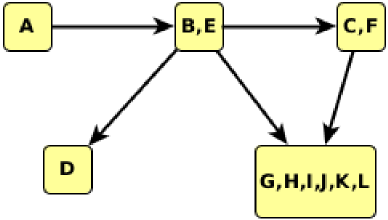

**Ισχυρά συνεκτική συνιστώσα (ΙΣΣ)** ενός κατευθυνόμενου γραφήματος $G(V, E) =$ ένα μέγιστο σύνολο κορυφών $U \subseteq V$, τ.ώ. $u, v \in U$ να υπάρχει διαδρομή $u \Rightarrow v$ και $v \Rightarrow u$.

Θα βασιστούμε στο γεγονός ότι

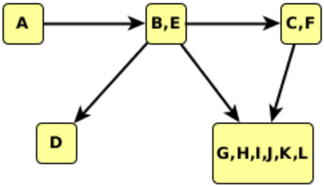
> Οι **ισχυρά συνεκτικές συνιστώσες** ενός κατευθυνόμενου γραφήματος μπορούν να υπολογιστούν σε **γραμμικό χρόνο**.

## (β) Γενίκευση για κάθε κατευθυνόμενο γράφημα $G$



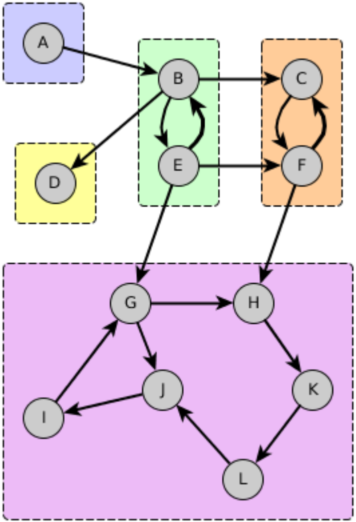

Το γράφημα $G'$ που προκύπτει αν 'συρρικνώσουμε' κάθε ΙΣΣ σε μια κορυφή είναι DAG.



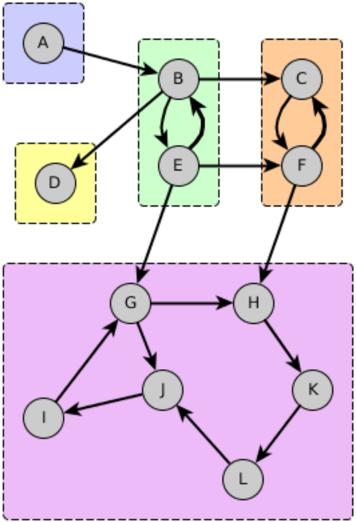

- Αν βρούμε τη *τοπολογική διάταξη του 'μεταγραφήματος'* αυτού, ουσιαστικά βάζουμε τις ΙΣΣ σε μια φθίνουσα σειρά χρόνων αναχώρησης του DFS
- Οι $\{G, H, I, J, K, L\}$, $\{D\}$ είναι **τερματικές** ισχυρά συνεκτικές συνιστώσες (= ισχυρά συνεκτική συνιστώσα χωρίς εξερχόμενες ακμές στο μεταγράφημα).
- Στην τοπολογική διάταξη του μεταγραφήματος, μια τερματική ΙΣΣ είναι η μετακορυφή με το *χαμηλότερο χρόνο αναχώρησης*, δηλ. η τελευταία στη διάταξη.

# Άσκηση 2: Συνάρτηση Κόστους σε Κατευθ. Γραφήματα

## (β) Γενίκευση για κάθε κατευθυνόμενο γράφημα $G$




- Αν ξεκινήσουμε τον DFS από έναν κόμβο που ανήκει σε μια τερματική ΙΣΣ, τότε θα εξερευνήσουμε όλη την ΙΣΣ αυτή και θα σταματήσουμε.
- **Ιδέα αλγόριθμου**:
  1. Ξεκινάμε από έναν κόμβο που ανήκει σε μια τερματική ΙΣΣ.
  2. Βρίσκουμε μέσω DFS όλους τους κόμβους σε αυτήν την ΙΣΣ.
  3. Αφαιρούμε την ΙΣΣ αυτή και επαναλαμβάνουμε.
- *Πώς βρίσκουμε μια τερματική ΙΣΣ;*




- Αν ξεκινήσουμε τον DFS από έναν κόμβο που ανήκει σε μια τερματική ΙΣΣ, τότε θα εξερευνήσουμε όλη την ΙΣΣ αυτή και θα σταματήσουμε.
- **Ιδέα αλγόριθμου**:
  1. Ξεκινάμε από έναν κόμβο που ανήκει σε μια τερματική ΙΣΣ.
  2. Βρίσκουμε μέσω DFS όλους τους κόμβους σε αυτήν την ΙΣΣ.
  3. Αφαιρούμε την ΙΣΣ αυτή και επαναλαμβάνουμε.
- *Πώς βρίσκουμε μια τερματική ΙΣΣ;*

Στο αντίστροφο γράφημα $G^R$ μια τερματική ΙΣΣ γίνεται μια αρχική ΙΣΣ, δηλαδή μια ΙΣΣ χωρίς εισερχόμενες ακμές!

### Αλγόριθμος για ισχυρά συνεκτικές συνιστώσες

1. DFS($G$) για τον υπολογισμό των χρόνων αναχώρησης $f[u]$ για κάθε $u \in V$
2. υπολογισμός **αντίστροφου γραφήματος** $G^R$ (δηλ. με ανεστραμμένες τις ακμές)
3. DFS($G^R$) $\rightarrow$ στο κύριο βρόχο for της DFS_Init εξετάζουμε τις κορυφές *σε φθίνουσα σειρά των χρόνων* $f[u]$ από το βήμα 1
4. οι κορυφές κάθε δένδρου από το βήμα 3 συναποτελούν μια συνεκτική συνιστώσα

*Πολυπλοκότητα*: $O(|V| + |E|)$

*Πώς ο αλγόριθμος για τις ισχυρά συνεκτικές συνιστώσες μας βοηθά στον υπολογισμό του $c(u)$;*

- Σε μια ισχυρά συνεκτική συνιστώσα $U \subseteq V$, για κάθε $u, v \in U$ υπάρχει διαδρομή $u \Rightarrow v$ και $v \Rightarrow u$. Άρα $c(u) = c(v)$**,** $\forall u, v \in U$**.**
- Συνεπώς, αρκεί να υπολογίσουμε τη μικρότερη τιμή $p(u)$ κάθε ΙΣΣ και να τρέξουμε τον αλγόριθμο του ερωτήματος (α) στο μεταγράφημα των ΙΣΣ του $G$.

### Αλγόριθμος

1. Βρίσκουμε τις συνεκτικές συνιστώσες του γραφήματος.
2. Για κάθε ΙΣΣ $U$, υπολογίζουμε το $p'(U) = min_{u \in U} p(u)$.
3. Στο μεταγράφημα με κορυφές τις ΙΣΣ του $G$, τρέχουμε τον αλγόριθμο του ερωτήματος (α). Βρίσκουμε τις τιμές $c'(U)$.
4. Για κάθε ΙΣΣ, για κάθε κορυφή $u \in U$ θέτουμε $c(u) = c'(U)$.

Κάθε βήμα απαιτεί χρόνο $O(|V| + |E|) \Rightarrow$ γραμμικός αλγόριθμος.

# Άσκηση 3: Ανάλυση Ασφάλειας

**Είσοδος:**

- Εταιρικό δίκτυο με $n$ υπολογιστές $C_1, .., C_n$
- Τριάδες της μορφής $(C_i, C_j, t)$ σε αύξουσα σειρά
  $\Rightarrow$ οι $C_i$ και $C_j$ θα επικοινωνήσουν τη χρονική στιγμή $t$
- $C_1$ μολύνεται τη χρονική στιγμή $t_1 = 0$

**Ζητούμενο:** Αλγόριθμος *γραμμικού χρόνου* που βρίσκει για κάθε υπολογιστή $C_j$ τη χρονική στιγμή $t_j$ που μολύνθηκε

**Σημείωση:** Ένας υπολογιστής μολύνεται όταν επικοινωνεί τη στιγμή $t$ με κάποιον υπολογιστή που μολύνθηκε κάποια στιγμή $t' \leq t$.

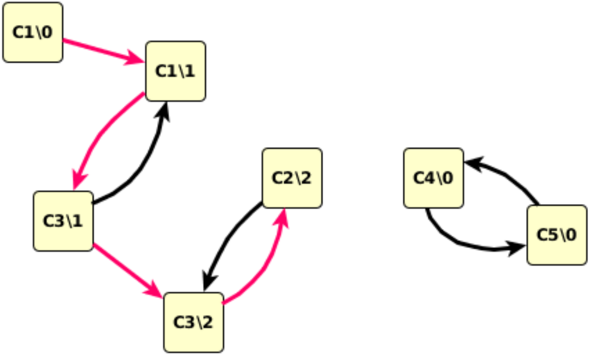
# Άσκηση 3: Ανάλυση Ασφάλειας

Η λύση βασίζεται στην κατασκευή του εξής **κατευθυνόμενου γράφου**:

- Διαβάζουμε τις τριάδες σε **αύξουσα σειρά χρόνου**.
- Για κάθε τριάδα $(C_i, C_j, t)$, δημιουργούμε έναν **κόμβο** $[C_i/t]$ και έναν $[C_j/t]$.
- Προσθέτουμε μια **κατευθυνόμενη ακμή** $[C_i/t] \rightarrow [C_j/t]$ και μια κατευθυνόμενη ακμή $[C_i/t] \leftarrow [C_j/t]$.
- **Ελέγχουμε** αν είναι η πρώτη φορά που συναντάμε στις τριάδες τον υπολογιστή $C_i$.
  Αν όχι, βρίσκουμε τη τελευταία χρονική στιγμή $t' \leq t$ που συναντήσαμε τον $C_i$ σε κάποια τριάδα. Έτσι, προσθέτουμε μια κατευθυνόμενη ακμή $[C_i/t'] \rightarrow [C_i/t]$.
  Όμοια, για τον $C_j$.
  *Υλοποίηση αυτού του βήματος*: Με ένα πίνακα $n$ θέσεων. Στη θέση $i$ για τον $C_i$ αποθηκεύουμε την πιο πρόσφατη επικοινωνία που είχε (δηλ. κάποιον κόμβο $[C_j/t']$).

# Άσκηση 3: Ανάλυση Ασφάλειας

Έχοντας κατασκευάσει αυτό το γράφο, το αρχικό πρόβλημα μεταφράζεται στο πρόβλημα εύρεσης όλων των κόμβων που ανήκουν στην **ίδια συνεκτική συνιστώσα με τον κόμβο** $[C_1/0]$.




Πχ. στο παραπάνω παράδειγμα, οι $C_4$, $C_5$ δεν μπορούν να μολυνθούν

# Άσκηση 3: Ανάλυση Ασφάλειας

Διατηρούμε έναν ακόμα **πίνακα** $results[1..n]$ θέσεων, όπου στην $i$-οστή θέση θα αποθηκεύουμε τη χρονική στιγμή που προσβλήθηκε ο $C_i$.

1. **Αρχικοποιούμε** τις θέσεις του πίνακα $results[1..n]$ στο NULL (μη μολυσμένοι υπολογιστές).
2. Ξεκινώντας από τον κόμβο $[C_1/0]$, εκτελούμε τον **BFS/DFS** στον κατευθυνόμενο γράφο που κατασκευάσαμε.
3. Κάθε φορά που συναντάμε ένα νέο κόμβο $[C_i, t]$ και ο $C_i$ είναι **μη μολυσμένος** (δηλαδή $results[i] \neq NULL$), θέτουμε $results[i] = t$. Αλλιώς θέτουμε $results[i] = min(t, results[i])$.

Βασιζόμαστε στο εξής:

> Υπάρχει ένα κατευθυνόμενο μονοπάτι από το κόμβο $[C_1/0]$ στον $[C^*/t]$, ανν όντως ο $C^*$ έχει μολυνθεί μέχρι και τη στιγμή $t$.

- Από την κατασκευή του γράφου, ακμή υπάρχει μόνο ανάμεσα σε δύο υπολογιστές που επικοινωνούν την ίδια στιγμή $t'$ ή ανάμεσα σε δύο κόμβους $[C_i/t_1] \rightarrow [C_i/t_2]$, $t_1 \leq t_2$.
  Έτσι, κατά την εκτέλεση του *BFS*, ο ιός βρίσκει μόνο έγκυρα κατευθυνόμενα μονοπάτια μέχρι τον υπολογιστή $[C^*/t]$.
- *Αντίστροφα*, αν ο $C^*$ μολύνεται τη στιγμή $t' \leq t$, θα υπάρχει μια ακολουθία επικοινωνίας ανάμεσα σε υπολογιστές που οδήγησε στη μόλυνση του $C^*$. Αντίστοιχα, στο γράφο θα υπάρχει εκ κατασκευής κάποιος κόμβος $[C^*/t']$, οι αντίστοιχοι ενδιάμεσοι κόμβοι και ακμές και άρα ένα μονοπάτι $[C_1/0] \rightarrow [C^*/t']$.

**Χρονική Πολυπλοκότητα** $O(m)$

- Μέγεθος κατευθυνόμενου γράφου: $|V| = O(m)$ και $|E| = O(m)$
  - Για κάθε τριάδα προσθέτω το πολύ 2 κόμβους και το πολύ 4 ακμές.
- Κατασκευή γράφου και πίνακα ελέγχου: $O(m)$
- Εκτέλεση BFS σε γραμμικό χρόνο ως προς το γράφο: $O(m)$.

# Άσκηση 4α: Το Σύνολο των Συνδετικών Δέντρων

## 4α)

- Έστω $T_1, T_2$ δύο διαφορετικά συνδετικά δέντρα και $e \in T_2 \backslash T_1$ $(T_2 \backslash T_1 \neq \varnothing)$.
- $T_1 \cup \{e\}$ περιέχει κύκλο $C$. Υπάρχει ακμή $e' \in C, e' \notin T_2$(Αλλίως $T_2$ περιέχει τον $C$)
- $(T_1 \backslash \{e\}) \cup \{e'\}$ είναι άκυκλο με $n-1$ ακμές $\Longrightarrow$ Συνδετικό δέντρο.

## Εύρεση $e'$

Έστω $T_1$, $T_2$ και $e = \{u, v\}$.

- Σε $O(|V|)$ κάνουμε Διάσχιση κατά Βάθος από το $u$ στο $v$ στο δέντρο $T_1$ και βρίσκουμε μονοπάτι $P$ που τους συνδέει.
- Για κάθε $e \in P$ ελεγχούμε σε $O(1)$ αν ανήκει στο $T_2$ και μόλις βρούμε $e \in P, e \notin T_2$ την αφαιρουμε. Συνολικά, $O(|V|)$

# Άσκηση 4β: Το Σύνολο των Συνδετικών Δέντρων

## Ιδιότητα

Έστω $T_1, T_2 \in H$ και $d_H(T_1, T_2)$ το μήκος του συντομότερο μονοπατιού μεταξύ $T_1, T_2$ στο $H$. Τότε $|T_1 \setminus T_2| = k$ ανν $d_H(T_1, T_2) = k$

## Απόδειξη

Θα χρησιμοποιήσουμε Επαγωγή στο μήκος του μονοπατιού:

- Επαγωγική Βάση: Από ορισμό του $H$, $|T_1 \setminus T_2| = 1$ ανν $d_H(T_1, T_2) = 1$.
- Επαγωγική Υπόθεση: $|T_1 \setminus T_2| = k$ ανν $d_H(T_1, T_2) = k$.
- Επαγωγική Βήμα: Θ.δ.ο. $|T_1 \setminus T_2| = k + 1$ ανν $d_H(T_1, T_2) = k + 1$

# Άσκηση 4: Το Σύνολο των Συνδετικών Δέντρων

## Απόδειξη Επαγωγικού Βήματος

- $\Longrightarrow$ Έστω $e \in T_1 / T_2$. Λόγω (α), υπαρχει $e' \in T_2 / T_1$ τ.ω. $T_1' = (T_1 \setminus \{e\}) \cup \{e'\} \in H$. Αλλά, $|T_1' \setminus T_2| = k \Longrightarrow_{\text{E.Υ}} d_H(T_1, T_2) \leq k+1$. Από Ε.Υ. αν $d_H(T_1, T_2) = k$ τότε $d_H(T_1, T_2) = k$. Άτοπο($|T_1 \setminus T_2| = k+1$) Άρα, $d_H(T_1, T_2) \geq k+1 \Longrightarrow d_H(T_1, T_2) = k+1$.
- $\Longleftarrow$ Εφόσον $d_H(T_1, T_2) = k+1$ υπάρχει $T_1' \in H$ τ.ω. $d_H(T_1, T_1') = 1$,$d_H(T_1', T_2) = k$. Από Ε.Υ. $|T_1' \setminus T_2| = k$. Άρα, $|T_1 \setminus T_2| = k-1$ ή $|T_1 \setminus T_2| = k+1$. Αν $|T_1 \setminus T_2| = k-1 \Longrightarrow_{\text{E.Ψ.}} d_H(T_1, T_2) = k-1$ Άτοπο.

# Άσκηση 4: Το Σύνολο των Συνδετικών Δέντρων

## Υπολογισμός Συντομότερου Μονοπατιού

- Υπολογίζουμε το σύνολο ακμών $T_2 \setminus T_1$
- Για κάθε $e \in T_2 \setminus T_1$ προσθέτουμε την $e$ στο υπάρχον δέντρο.

## Ορθότητα

Αν $T_2 \setminus T_1 = k$ τότε σε $k$ βήματα έχουμε μετατρέψει το $T_1$ σε $T_2$. Άρα, έχουμε βρει μονοπάτι στο $H$ μήκους $k \implies$ Το συντομότερο μονοπάτι.

## Πολυπλοκότητα

Σε $O(|V|)$ αποθηκεύουμε το $T_1$ ως *rooted* δέντρο(κάθε κόμβος δείχνει στον πατέρα του). Για κάθε ακμή $e \in T_2$ έλεγχουμε σε $O(1)$ αν $e \in T_1$. Άρα, $O(|V|)$. Κάνουμε $k$ ανανεώσεις ακμών σε $O(|V|)$ βήματα. Σύνολο, $O(k \cdot |V|)$.

# Άσκηση 4γ: Το Σύνολο των Συνδετικών Δέντρων

## Αλγόριθμος

- Αν $|E_1| < k$, προφανώς δεν υπάρχει το ζητούμενο ΣΔ.
- Θέτουμε στις ακμές του συνόλου $E_1$ βάρος 1, και στις ακμές του συνόλου $E_2$ βάρος 0.
- Εκτελούμε τον αλγόριθμο *Kruskal* στο γράφημα που προκύπτει, οπότε λαμβάνουμε ένα ΕΣΔ $T_1$.
- Αν $w(T_1) > k$, τότε δεν υπάρχει το ζητούμενο ΕΣΔ.
- Έστω $k_1$ το πλήθος των ακμών του $T_1 \cap E_1$. .
- Θεωρούμε το γράφημα $G'$ που περιέχει μόνο αυτές τις $k_1$ ακμές.
- Θέτουμε στο $G'$, $w(E_1 \setminus (T_1 \cap E_1)) = 0$ και $w(E_2) = 1$.
- Εκτελούμε τον αλγόριθμο *Kruskal* στο $G'$, προσθέτοντας ακμές ώσπου να έχουμε συνολικά $k$ ακμές του συνόλου $E_1$.
- Η εκτέλεση του *Kruskal* συνεχίζεται με τις ακμές του συνόλου $E_2$, μέχρι το $G'$ να γίνει συνεκτικό.
- Ονομάζουμε το γράφημα που προκύπτει $G_1$ και το επιστρέφουμε ως ΣΔ του $G$.

# Άσκηση 4γ: Το Σύνολο των Συνδετικών Δέντρων

## Ορθότητα

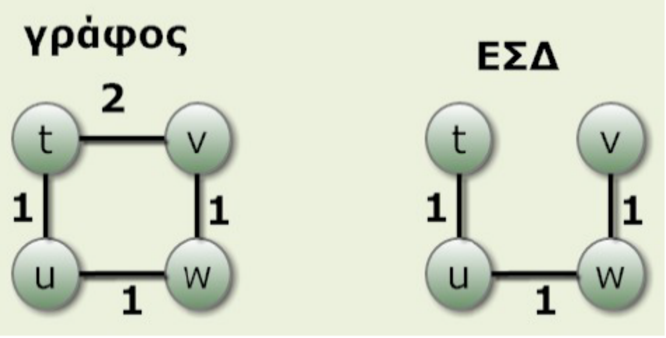
- Έστω το $G'' = G_1 \cup E_2$. Το $G''$ έχει ως ΣΔ το $T_1$ επομένως είναι συνεκτικό. Η εκτέλεση του αλγορίθμου *Kruskal* στο $G''$ θα επιστρέψει ως ΕΣΔ το $G_1$, επομένως το $G'$ κάποια στιγμή θα γίνει συνεκτικό.
- Προφανώς, στο $G_1$ υπάρχουν ακριβώς $k$ ακμές του $E_1$, από τον ορισμό του αλγορίθμου.

## Πολυπλοκότητα

$\mathcal{O}(m * logm)$ λόγω του αλγορίθμου *Kruskal*.

# Άσκηση 5α: Μοναδικότητα Ελάχιστου Συνδετικού Δέντρου

## 5α)



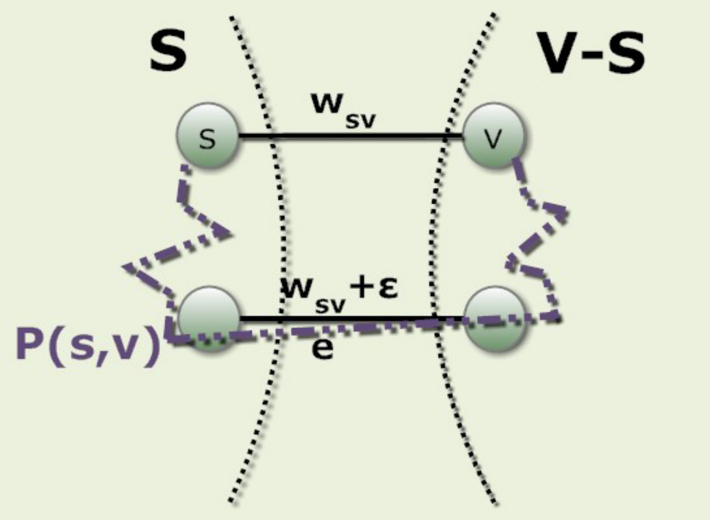
# Άσκηση 5: Μοναδικότητα Ελάχιστου Συνδετικού Δέντρου

## 5β)

- Έστω $T_1 \neq T_2$ 2 ΕΣΔ του $G(V, E, w)$
- Αφού $T_1 \neq T_2 \Rightarrow \exists \{s, v\} \in E : \{s, v\} \in T_1 \wedge \{s, v\} \notin T_2$
- $\{s, v\} \Rightarrow$ τομή $(S, V - S)$
- Όμως (από υπόθεση) για κάθε τομή $C \Rightarrow | \, argmin_e\{\delta(C)\} \, | = 1 \Rightarrow \{s, v\} = argmin_{e \in \delta(S, V-S)}\{w(e)\} \in$ σε κάποιο ΕΣΔ (έστω το $T_1$)
- $\{s, v\} \notin T_2 \wedge T_2$ ΕΣΔ $\Rightarrow$ υπάρχει μονοπάτι $P(s, v) \in T_2$ $\Rightarrow \exists e \in \delta(S, V - S) : e \in P(s, v) \in T_2 \Rightarrow P(s, v) \cup \{s, v\}$ κύκλος $\Rightarrow (T_2 - \{e\}) \cup \{s, v\} = T'$ συνδετικό δέντρο με $w(T') < w(T)$ ΑΤΟΠΟ

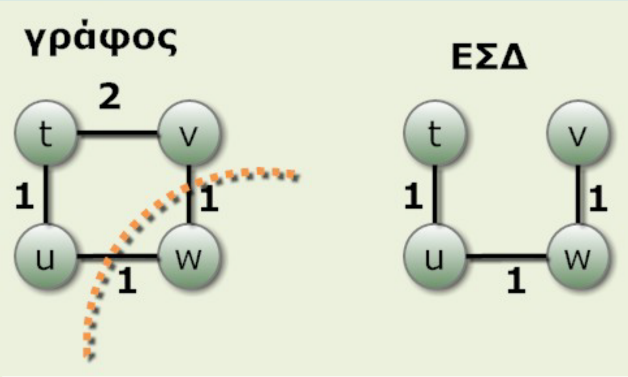
# Άσκηση 5: Μοναδικότητα Ελάχιστου Συνδετικού Δέντρου

## 5β)



# Άσκηση 5: Μοναδικότητα Ελάχιστου Συνδετικού Δέντρου

## 5β) Αντιπαράδειγμα :



# Άσκηση 5: Μοναδικότητα Ελάχιστου Συνδετικού Δέντρου

## 5γ)

- Αν κύκλος $(e, T)$, ο κύκλος προκύπτει στο $T \cup \{e\}$, όπου $T$ συνδετικό δέντρο
- Αν σύνολο $K_e^{max} = argmax_{e \in (e,T)}\{w(e)\}$
- $\exists! T$ ΕΣΔ $\Leftrightarrow \forall e \notin T$ συνδετικό δέντρο $e \in K_e^{max} \wedge \mid K_e^{max} \mid = 1$

# Άσκηση 5: Μοναδικότητα Ελάχιστου Συνδετικού Δέντρου

## 5γ)

$$\exists! T \text{ ΕΣΔ} \Leftrightarrow \forall e \notin T \text{ ΣΔ } e \in K_e^{max} \wedge \mid K_e^{max} \mid = 1$$

## Απόδειξη $\Rightarrow$

Αν $\exists e \notin T : e \notin K_e^{max} \Rightarrow T \not\text{ΕΣΔ}$
Αν $\exists e \in T (\Rightarrow e \in K_e^{max}) : \exists e' (\neq e) \in K_e^{max} \Rightarrow (T - \{e\}) \cup \{e'\}$ επίσης ΕΣΔ

## Απόδειξη $\Leftarrow$

Έστω $T' \neq T$ ένα ΕΣΔ $\Rightarrow \exists (s, v) \in T' \wedge \notin T$
Έστω $P(v, s) \in T$ και $(S, V \setminus S)$ τομή της $(s, v)$
$\exists e : e \in \delta(S, V \setminus) \wedge e \in P(s, v) \wedge e \notin T' \Rightarrow w(s, v) > e', \forall e' \in$ κύκλος$(T, (s, v)) \Rightarrow w((T' - \{(s, v)\}) \cup \{e\}) < w(T')$

## 5δ)

- Υπολογίζουμε ένα ΕΣΔ $T$
- $\forall v \in T$ τρέχουμε $DFS(v)$ το οποίο
  - επιστρέφει έναν πίνακα $wMaxEdge[v][]$, όπου $wMaxEdge[v][u]$ το βάρος της πιο βαριάς ακμής στο $P(v,u)$ στο δέντρο $T$
  - κατά την **επίσκεψη** της ακμής $(a,b)$ ισχύει $wMaxEdge[v][b] = max\{wMaxEdge[v][a], w(a,b)\}$
  - $wMaxEdge[v][]$ αποθηκεύει τα αποτελέσματα του $DFS$
- $\forall (v,u) \notin T$
  - $if(w(v,u) < wMaxEdge[y][u]) \;\; return \; false;$
- $return \; true;$

# Άσκηση 5: Μοναδικότητα Ελάχιστου Συνδετικού Δέντρου

## 5δ)

- $\Theta(|E| log |E|)$ με *kruskal* για την εύρεση ΕΣΔ
- *DFS* σε δέντρο $\Theta(|V| + |V| - 1) = \Theta(|V|)$
  - $|V|$ εκτελέσεις $\Theta(|V|^2$
- $O(|E|)$ το τελικό στάδιο
- Σύνολο : $O(|V|^2 + |E| log |E|)$

# Άσκηση 5: Μοναδικότητα Ελάχιστου Συνδετικού Δέντρου

## 5δ) [*bonus*]

Θα τροποποιήσουμε τον *kruskal* ως εξής :

- Αντί για ακμές *μία*, *μία*, εξετάζουμε **συστάδες ακμών ίδιου βάρους** (σε αύξουσα σειρά)
- $\forall$ τέτοια συστάδα, όποια ακμή **σχηματίζει κύκλο** με το μέχρι τώρα δάσος (από ακμές μικρότερου βάρους) δεν ανήκει σε κανένα ΕΣΔ, άρα μπορεί να αγνοηθεί
  - *είναι οι πιο βαριές ακμές σε κύκλο με ΣΔ*
- Είσάγουμε όλες τις υπόλοιπες
  - Αν κύκολος$\neg\exists!$ ΕΣΔ
  - τουλάχιστον 2 εναλλακτικές για ΕΣΔ (οι πιο βαριές ακμές του κύκλου που μόλις βρήκαμε)

# 1η προγραμματιστική Άσκηση: Τηλεμεταφορές

Είσοδος: Κόστος καναλιού μεταφοράς μεταξύ πλανητών + κόστος εγκατάστασης σταθμού τηλεματαφοράς σε κάθε πλανήτη
Έξοδος: Δίκτυο ελαχίστου συνολικού κόστους

## Ιδέα

Το βέλτιστο Δίκτυο Μεταφορών είναι το έλαχιστο από:

- το βέλτιστο Δίκτυο Μεταφορών χωρίς σταθμό Τηλεμεταφοράς
- το βέλτιστο Δίκτυο Μεταφορών με τουλάχιστον δύο σταθμούς Τηλεμεταφοράς

### Μετατροπή σε Πρόβλημα ΕΣΔ

Δημιουργούμε γράφημα $G(V, E, w)$ ως εξής:

- Για κάθε πλανήτη $i$, δημιουργούμε κόμβο $i \in V$
- Για πλανήτες $i, j$ ,που είναι δυνατή η μεταφορά, δημιουργούμε ακμή $e = \{i, j\} \in E$ τ.ω. $w_e = A[i, j]$
- Δημιουργούμε υπερ-πλανήτη $s \in V$
- Για κάθε πλανήτη $i$, δημιουργούμε ακμή $e = \{i, s\} \in E$ τ.ω. $w_e = B[i]$

Μπορούμε να υπολογίσουμε το βέλτιστο Δίκτυο Μεταφορών με χρήση του ΕΣΔ$(G)$,ΕΣΔ$(G \setminus \{s\})$!

# 1η προγραμματιστική Άσκηση: Τηλεμεταφορές

**Υπολογισμός Βέλτιστου Δικτύου από ΕΣΔ($G$)**

Έστω $T_1 =$ΕΣΔ$(G), T_2 =$ΕΣΔ$(G \backslash \{s\})$ τότε υπολογίζουμε το Βέλτιστο Δίκτυ Μεταφορών ως εξής:

1. if $w(T_1) < w(T_2)$ then return $T_1$
2. else return $T_2$

Ο παραπάνω αλγόριθμος είναι βέλτιστος.

**Απόδειξη**

Έστω $OPT$ το βέλτιστο δίκτυο και $S$ το δίκτυο που επιστρέφει ο παραπάνω αλγόριθμος:

- Αν $OPT$ δεν περιέχει σταθμούς Τηλεμεταφοράς: τότε προφανώς $w(OPT) = w(T_2)$ γιατί χρησιμοποιούνται μόνο κανάλια μεταφοράς και άρα ακμές που δεν περιέχουν τον $s$. Αν $degree_{T_1}(s) = 1$ τότε $w(T_1) > w(T_2)$. Αν $degree_{T_1}(s) \geq 2$ τότε $w(T_1) \geq w(T_2)$ γιατί αλλίως το $OPT$ θα περιέχει σταθμούς Τηλεμεταφοράς. Άρα, $w(T_2) < W(T_1) \Longrightarrow S = T_2 = OPT$.
- Αν $OPT$ δεν περιέχει σταθμούς Τηλεμεταφοράς: Τότε $OPT \leq w(T_2)$ γιατί $T_2$ το βέλτιστο δίκτυο χωρίς χρήση σταθμών τηλεμεταφοράς. Επίσης, $OPT$ είναι συνδετικό δέντρο για το $G$ και άρα $w(T_1) = OPT \Longrightarrow w(T_1) \leq w(T_2) \Longrightarrow S = T_1 = OPT$.

**Συνολικά**

Υπολογισμός δύο ΕΣΔ, άρα $O(|E| log(|E|))$.

# 2η προγραμματιστική Άσκηση: Καλοκαιρινές Διακοπές

Μας ζητείται το *bottleneck* κόστος μεταξύ όλων των ζευγών $(i, j) \in Q$.

<u>ΠΑΡΑΤΗΡΗΣΗ:</u> Το *bottleneck* κόστος για το ζεύγος $(i, j)$ στο γράφημα $G$ ισούται με το *bottleneck* κόστος του $(i, j)$ στο ΕΣΔ $T$ του $G$.

<u>ΑΠΟΔΕΙΞΗ:</u> Το μονοπάτι που ελαχιστοποιεί το *bottleneck* κόστος του ζεύγους $(i, j)$ είναι το μονοπάτι που θα ενώνει τις κορυφές $(i, j)$ και στο $T$.

Μπορούμε να υπολογίσουμε το ΕΣΔ $T$ του $G$ σε χρόνο $\mathcal{O}(m * logm)$.

Αρκεί να λύσουμε το πρόβλημα στο $T$. Θα παρουσιάσουμε τρεις διαφορετικές λύσεις.

Λύση 1 (*Naive*): Για κάθε ζεύγος $(i, j) \in Q$ βρίσκουμε με *BFS* ή *DFS* το **μοναδικό** μονοπάτι από τον $i$ στον $j$, και επιστρέφουμε την βαρύτερη ακμή.
Πολυπλοκότητα: $\mathcal{O}(m * logm + |Q| * n)$.

Στην διάσχιση του ΕΣΔ στην προηγούμενη λύση, κατά την διάρκεια του υπολογισμού του *bottleneck* κόστους για ένα ζεύγος $(i, j)$ αγνοούμε πολλή πληροφορία. Μπορούμε να εκμεταλλευτούμε την δενδρική δομή του ΕΣΔ ώστε να γλιτώσουμε περιττούς υπολογισμούς.

Λύση 2:

- Διαλέγουμε αυθαίρετα μία κορυφή $r$ ως ρίζα του $T$.
- Με διάσχιση *BFS* στο $T$ υπολογίζουμε για κάθε ζεύγος $(i, j) \in VxV$ τον Ελάχιστο Κοινό Πρόγονο (*Least Common Ancestor*) των $(i, j)$ σε συνολικό χρόνο και χώρο $\mathcal{O}(n^2)$, και τον αποθηκεύουμε σε ένα πίνακα $A$ μεγέθους $nxn$.
- Υπολογίζουμε για κάθε κορυφή $u$ το *bottleneck* κόστος του $u$ με όλους τους προγόνους του. Αυτό μπορεί να γίνει σε συνολικό χρόνο $\mathcal{O}(n^2)$, και το αποθηκεύουμε σε ένα πίνακα $B$ μεγέθους $nxn$.
- Για κάθε ζεύγος κορυφών $(i, j) \in Q$ βρίσκουμε τον Ελάχιστο Κοινό Πρόγονο των $(i, j)$, έστω $k$, ο οποίος βρίσκεται στην θέση $A[i][j]$.
- Το *bottleneck* κόστος του μονοπατιού μεταξύ των $(i, j)$ είναι το $max(B[i][k], B[k][j])$.

Πολυπλοκότητα: $\mathcal{O}(m * logm + n^2 + |Q|)$.

Στην προηγούμενη λύση πετύχαμε σταθερό χρόνο για κάθε ερώτημα $(i, j) \in Q$ αλλά χρειαστήκαμε προεπεξεργασία $\mathcal{O}(n^2)$. Μπορούμε να μειώσουμε τον χρόνο προεπεξεργασίας και να αυξήσουμε λίγο τον χρόνο που χρειαζόμαστε για κάθε ερώτημα, πετυχαίνοντας όμως έτσι συνολικά καλύτερη πολυπλοκότητα.

Λύση 3:

- Διαλέγουμε αυθαίρετα μία κορυφή $r$ ως ρίζα του $T$.
- Θεωρούμε έναν πίνακα $A$ μεγέθους $n * logn$ και στο στοιχείο $A[u][i]$ αποθηκεύουμε τον πρόγονο του $u$ που απέχει στο $T$ από τον $u$ απόσταση $2^i$. Παράλληλα αποθηκεύουμε σε έναν αντίστοιχο πίνακα $B$ το *bottleneck* κόστος μεταξύ του $u$ και του $A[u][i]$. Η διαδικασία τελειώνει όταν συναντήσουμε (ή περάσουμε) τον $r$. (Σκεφτείτε γιατί χρειαζόμαστε μόνο $logn$ στήλες).

# 2η προγραμματιστική Άσκηση: Καλοκαιρινές Διακοπές

Λύση 3: (Συνέχεια)

- Θεωρούμε ένα ερώτημα $(u, v)$. Ξεκινάμε από την γραμμή $u$ του πίνακα $A$.
  - Για κάθε στοιχείο $A[u][k]$ μπορούμε σε σταθερό χρόνο να υπολογίσουμε εάν είναι πρόγονος του $v$ (Άσκηση 1 - 3η Σειρά Γραπτών Ασκήσεων).
  - Διατρέχουμε την γραμμή $u$ ξεκινώντας από την τελευταία στήλη, που γνωρίζουμε ότι είναι το $r$, άρα είναι πρόγονος του $v$, και κινούμαστε προς τα αριστερά.
  - Όσο το $A[u][k]$ είναι πρόγονος του $v$, κινούμαστε κατά μία θέση προς τα αριστερά.
  - Μόλις το $A[u][k]$ δεν είναι πρόγονος του $v$ για πρώτη φορά, μεταφερόμαστε στην γραμμή $m = A[u][k]$ και συνεχίζουμε την προς τα αριστερά αναζήτηση από την στήλη $k$.
  - Τερματίζουμε μόλις φθάσουμε στην στήλη 0, επιστρέφοντας ως αποτέλεσμα, εάν βρισκόμαστε στην γραμμή $w$, το στοιχείο $B[w][0]$.

Ορθότητα:

- Έστω ότι κατά την διάρκεια της διάσχισης της γραμμής $u$, πήραμε το πρώτο 'ΟΧΙ' στο σημείο $k$. Αυτό σημαίνει ότι η κορυφή $u + 2^{k+1}$ είναι πρόγονος του $v$, και ότι η κορυφή $u + 2^k$ δεν είναι πρόγονος του $v$. Επομένως, το *bottleneck* κόστος του ζεύγους $(u, v)$, έστω $bl(u, v)$, είναι ίσο με $max(bl(u, u + 2^k), bl(u + 2^k, v))$. Όμως, το $bl(u, u + 2^k) = B[u][k]$.
- Γνωρίζουμε ότι το $u + 2^{k+1}$ που είναι η κορυφή $A[u + 2^k][k]$, είναι πρόγονος του $v$, άρα μπορούμε να συνεχίσουμε την αναζήτηση στην γραμμή $u + 2^k$ από την στήλη $k - 1$.
- Όταν φθάσουμε σε κάποιο στοιχείο $A[w][0]$ γνωρίζουμε πως ο $w$ είναι ο Ελάχιστος Κοινός Πρόγονος των $(u, v)$.

Πολυπλοκότητα: $\mathcal{O}(m * logm + |Q| * logn + n * logn)$.

- Υπολογισμός ΕΣΔ: $\mathcal{O}(m * logm)$.
- Υπολογισμός πινάκων $A, B$: $\mathcal{O}(n * logn)$.
- Για κάθε ερώτημα, διατρέχουμε συνολικά τις $logn$ στήλες των πινάκων $A, B$, επομένως έχουμε $\Theta(logn)$.

ΕΡΩΤΗΜΑ: Θα μπορούσαμε με παρόμοια ανάλυση να πετύχουμε καλύτερη συνολική πολυπλοκότητα ή όχι και γιατί;