

01_ΓΡΑΦΗΜΑΤΑ / 01_graphsintro

Ορισμοί

Γράφημα

Γράφημα: μία διμελής σχέση μεταξύ των στοιχείων ενός συνόλου.

Ορισμός 1 'Ένα γράφημα $G(V, E)$ αποτελείται από

- ένα σύνολο κορυφών (κόμβων) $V(G)$ και
- ένα σύνολο ακμών

$$E(G) \subseteq \{\{u, v\} : u, v \in V(G)\}.$$

Τάξη γραφήματος: $n = |V(G)|$

Μέγεθος γραφήματος: $m = |E(G)|$

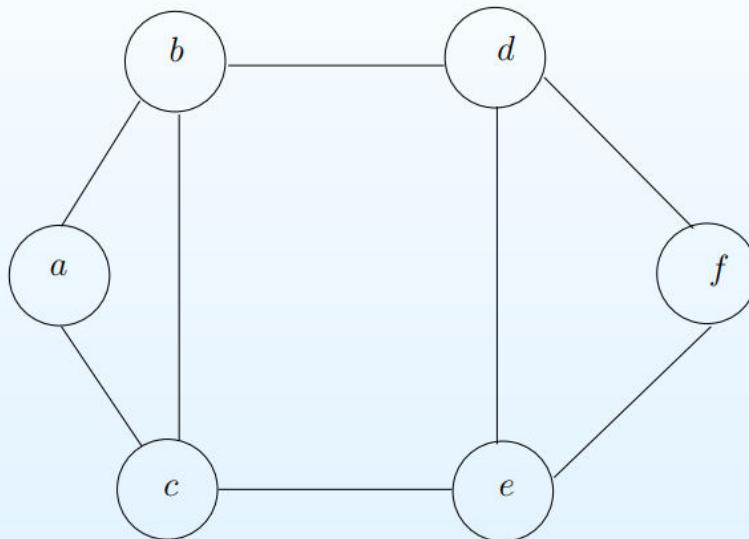
Av $n = 0 \Rightarrow$ κενό γράφημα

Av $n > m = 0 \Rightarrow$ γράφημα χωρίας ακμές

Παράδειγμα

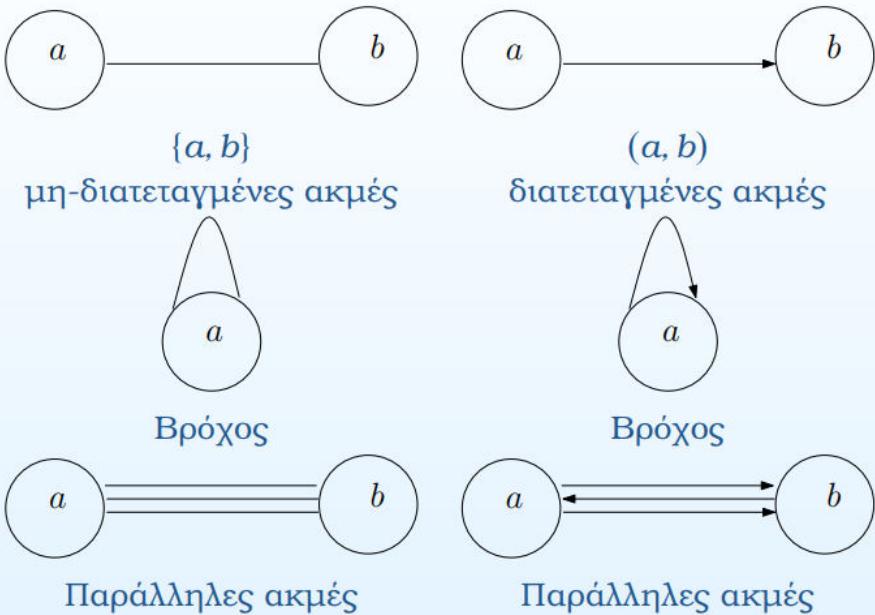
$$V(G) = \{a, b, c, d, e, f\}$$

$$E(G) = \{\{a, b\}, \{a, c\}, \{b, c\}, \{b, d\}, \{c, e\}, \{d, e\}, \{d, f\}, \{e, f\}\}$$



Σχήμα 1: Μη κατευθυνόμενο γράφημα με $n = 6, m = 8$.

Ορολογία



Απλό γράφημα: δεν περιέχει βρόχους ή παράλληλες ακμές

Βαθμός Κορυφής

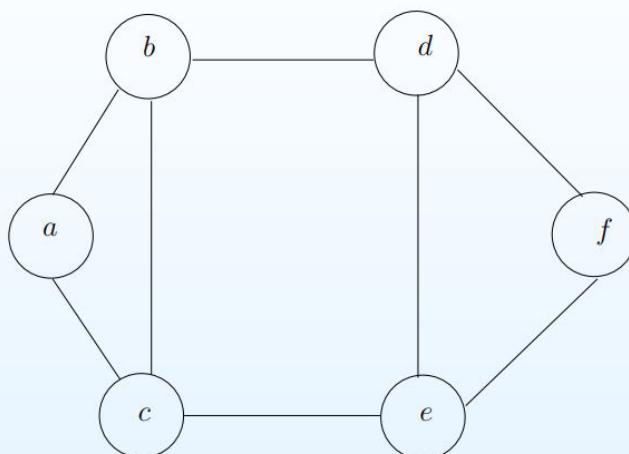
Μη-κατευθυνόμενο γράφημα:

$$N(v) = \{u \in V(G) : \{v, u\} \in E(G)\}, \\ d(v) = |N(v)|.$$

Κατευθυνόμενο γράφημα:

$$N^+(v) = \{u \in V(G) : (v, u) \in E(G)\}, d^+(v) = |N^+(v)| \\ N^-(v) = \{u \in V(G) : (u, v) \in E(G)\}, d^-(v) = |N^-(v)|$$

Παράδειγμα

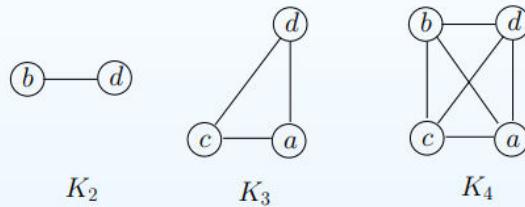


$$d(a) = 2, d(b) = 3, d(c) = 3, d(d) = 3, d(e) = 3, d(f) = 2.$$

Λίμνα Χειραψίας: $\sum_{v \in V(G)} d(v) = 2m$.

Πλήρες Γράφημα

Συμβολίζεται με K_n : απλό γράφημα με ακμές ανάμεσα σε όλους τους κόμβους



Σχήμα 2: Πλήρη γραφήματα με 2,3,4 κορυφές, αντίστοιχα

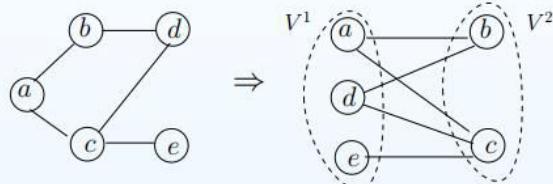
Αριθμός ακμών K_n : $\frac{n(n-1)}{2}$.

Για κάθε απλό γράφημα ισχύει

$$0 \leq m \leq \frac{n(n - 1)}{2}$$

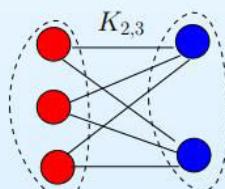
Διμερή Γραφήματα

Ένα γράφημα $G(V, E)$ ονομάζεται διμερές αν υπάρχει διαμερισμός του συνόλου των κορυφών σε σύνολα V^1, V^2 έτσι ώστε για κάθε ακμή $\{v, u\} \in E$, $v \in V^1$, $u \in V^2$.



Σχήμα 3: Διμερές γράφημα

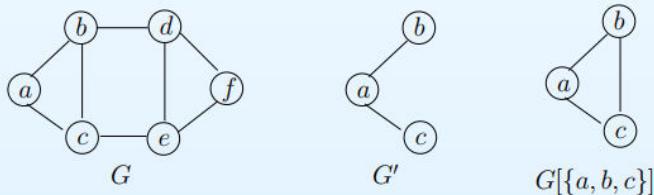
Ένα διμερές γράφημα με $|V^1| = n_1$, $|V^2| = n_2$ που έχει $n_1 * n_2$ ακμές ονομάζεται πλήρες διμερές και συμβολίζεται με K_{n_1, n_2} .



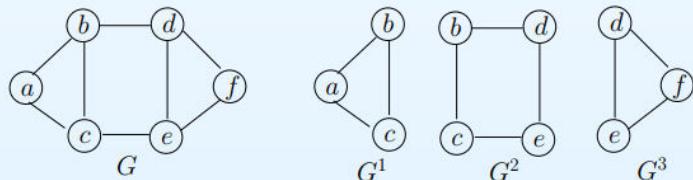
Υπογράφημα

Ένα υπογράφημα ενός γραφήματος $G(V, E)$ είναι ένα γράφημα $G'(V', E')$ με την ιδιότητα $V' \subseteq V$, $E' \subseteq E$. Συμβολίζουμε $G' \subseteq G$.

Ένα υπογράφημα $G' \subseteq G$ καλείται *μεγιστοτικό* αν δεν υπάρχει άλλο υπογράφημα $H \subseteq G$ τέτοιο ώστε $G' \subset H$. Ένα επαγόμενο υπογράφημα $G'(V', E')$ του G περιέχει κάθε ακμή ανάμεσα στους κόμβους του V' που υπάρχει στο G . Είναι δηλαδή ένα μεγιστοτικό υπογράφημα του G ως προς V' . Το συμβολίζουμε ως $G[V']$.

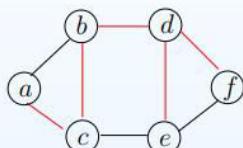


Σχήμα 5: Γράφημα G , Υπογράφημα G' , Επαγόμενο υπογράφημα $G[\{a, b, c\}]$



Σχήμα 5: Γράφημα G και όλα τα μεγιστοτικά υπογραφήματα με βαθμό κόμβου 2

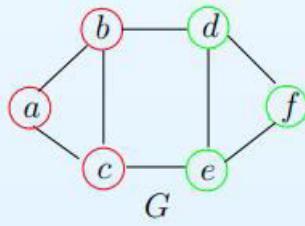
Ένα γεννητορικό υπογράφημα $G'(V', E')$ του $G(V, E)$ έχει $V = V'$ και $E' \subseteq E$. Άρα το G' είναι μεγιστοτικό ως προς το σύνολο E' .



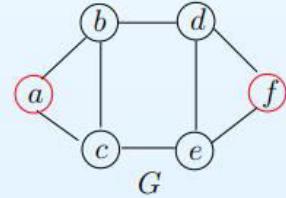
Σχήμα 6: Ένα γεννητορικό υπογράφημα του G (κόκκινο σύνολο ακμών).

clique: Υποσύνολο κόμβων $Q \subseteq V$ με ακμές ανάμεσα σε όλους τους κόμβους. Στο προηγούμενο σχήμα τα $\{a, b, c\}$, $\{d, e, f\}$ είναι **cliques** μεγέθους 3.

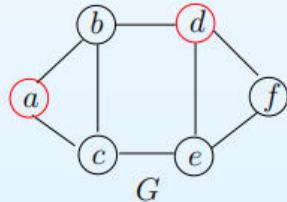
Ανεξάρτητο σύνολο: Υποσύνολο κόμβων $Q \subseteq V$ με $E(G[Q]) = \emptyset$. Δηλαδή, δεν υπάρχει ακμή ανάμεσα στους κόμβους του Q . Στο γράφημα του προηγούμενου σχήματος τα σύνολα $\{a, f\}$, $\{a, d\}$, $\{a, e\}$ είναι τα ανεξάρτητα σύνολα που συμμετέχει η κορυφή a .



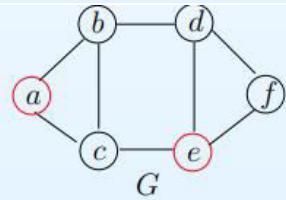
Σχήμα 7: cliques μεγέθους 3.



Σχήμα 7: Ανεξάρτητο σύνολο μεγέθους 2.



Σχήμα 7: Ανεξάρτητο σύνολο μεγέθους 2.



Σχήμα 7: Ανεξάρτητο σύνολο μεγέθους 2.

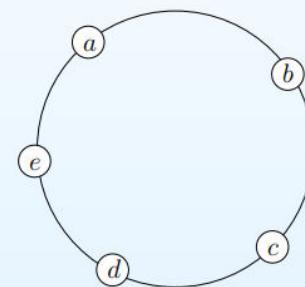
Διαδρομή και Μονοπάτια

Διαδρομή: Μία ακολουθία κορυφών $W = \langle v_0, v_1, \dots, v_k \rangle$ με $\{v_i, v_{i+1}\} \in E(G)$, $i = 0, \dots, k - 1$.

Μονοκονδυλιά: Διαδρομή χωρίς επαναλαμβανόμενη ακμή

Μονοπάτι: Διαδρομή χωρίς επαναλαμβανόμενη κορυφή

Κύκλος: Μονοπάτι όπου επαναλαμβάνεται μόνο η τερματική κορυφή.

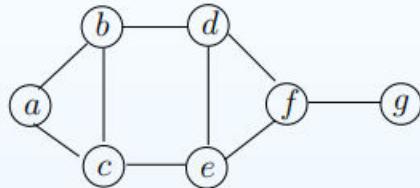


Σχήμα 8: Κύκλος C_5

Ένα γράφημα που δεν περιέχει κύκλο ονομάζεται άκυκλο

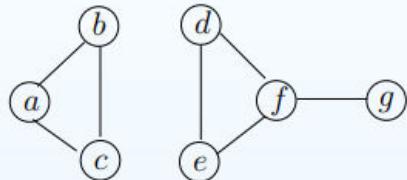
Συνδεδεμένο Γράφημα

Ένα γράφημα ονομάζεται συνδεδεμένο (ή συνεκτικό) αν υπάρχει μονοπάτι που συνδέει κάθε ζευγάρι κορυφών.



Σχήμα 9: Συνεκτικό γράφημα

Ένα γράφημα που δεν είναι συνδεδεμένο αποτελείται από γραφικές συνιστώσες.

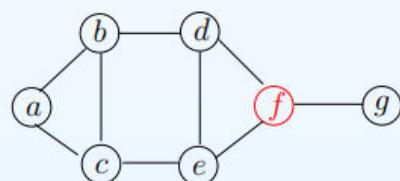


Σχήμα 9: Μη-συνεκτικό γράφημα με δύο συνιστώσες

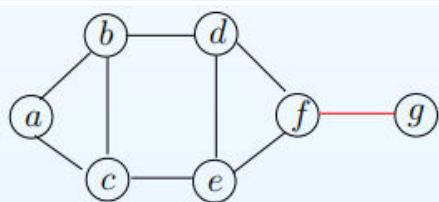
Τομές

Μία κορυφή ονομάζεται σημείο κοπής αν η αφαίρεση της (μαζί με τις προσπίπουσες ακμές) αποσυνδέει το γράφημα σε περισσότερες συνιστώσες.

Αντίστοιχα η ακμή ονομάζεται γέφυρα αν η αφαίρεση της αποσυνδέει το γράφημα σε περισσότερες συνιστώσες.



Σχήμα 10: Σημείο κοπής f



Σχήμα 10: Γέφυρα $\{f, g\}$

Παρατηρήσεις

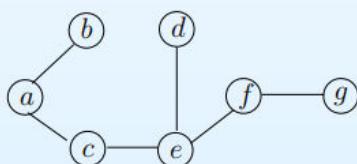
- Ένα συνεκτικό γράφημα έχει μία γραφική συνιστώσα.
- Αν σε κάποιο γράφημα υπάρχει κορυφή v με $d(v) = n - 1$ τότε το γράφημα είναι συνδεδεμένο.
- Αν από ένα γράφημα αφαιρέσουμε μια γέφυρα τότε αυξάνεται ο αριθμός των γραφικών συνιστωσών κατά ένα.
- Αν από ένα γράφημα αφαιρέσουμε το σημείο κοπής v τότε αυξάνεται ο αριθμός των γραφικών συνιστωσών το πολύ κατά $d(v) - 1$.
- Για οποιοδήποτε γράφημα με k συνιστώσες ισχύει $n \leq k + m$.
- Για κάθε συνεκτικό γράφημα ισχύει ότι ο αριθμός των ακμών πρέπει να είναι τουλάχιστον όσο ο αριθμός των κορυφών μείον ένα: $n - 1 \leq m$.
- Αν ένα συνεκτικό γράφημα έχει ΑΚΡΙΒΩΣ $n - 1$ ακμές, δηλαδή αν, $m = n - 1$ τότε λέγεται δένδρο. Ισοδύναμα, ένα δένδρο ορίζεται ένα γράφημα το οποίο είναι μεγιστοτικά άκυκλο και ελαχιστοτικά συνδεδεμένο.

Δένδρα

Ορισμοί

Έστω γράφημα $T(V, E)$. Τα παρακάτω είναι ισοδύναμα.

- Το γράφημα T είναι δένδρο.
- Στο T , μεταξύ κάθε ζεύγους κορυφών v, u με $v \neq u$ υπάρχει ένα μοναδικό μονοπάτι από την v στην u .
- Το T είναι ένας συνδεδεμένο ακυκλικό γράφημα.
- Το T είναι συνδεδεμένο γράφημα και έχει $n - 1$ ακμές.
- Το T είναι ακυκλικό γράφημα και έχει $n - 1$ ακμές.
- Το T είναι συνδεδεμένο γράφημα και κάθε ακμή είναι γέφυρα.
- Το T είναι ακυκλικό γράφημα και η προσθήκη ακμής δημιουργεί κύκλο.



Σχήμα 11: Γράφημα δένδρο.

Παρατηρήσεις

- Οι κορυφές με βαθμό ένα σε κάθε δένδρο ονομάζονται φύλλα.
- Οι υπόλοιπες κορυφές ονομάζονται εσωτερικές.

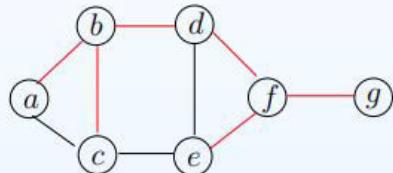
Ειδικά δένδρα.



Σχήμα 12: Γράφημα μονοπάτι (P_n) και γράφημα αστέρι (S_n).

Γεννητορικό Δένδρο

Έστω γράφημα $G(V, E)$ και υπογράφημα του $T(V, E')$, με $E' \subseteq E$ τέτοιο ώστε T είναι δένδρο. Το T ονομάζεται γεννητορικό ή συνεκτικό δένδρο του G .



Σχήμα 13: Κόκκινες ακμές σχηματίζουν ένα γεννητορικό δένδρο του G .

Ρίζα Δένδρου

Ρίζα: Μία διακεκριμένη κορυφή του δένδρου η οποία δεν είναι φύλλο.

Υπάρχει μοναδικό μονοπάτι από την ρίζα σε κάθε φύλλο.

Έστω $v_0, v_1, \dots, v_i, v_{i+1}, \dots, v_n$ ένα τέτοιο μονοπάτι. Η κορυφή v_i ονομάζεται γονέας (ή πατέρας) της v_{i+1} και η v_{i+1} παιδί της v_i .

- **Βάθος Κορυφής:** αριθμός των ακμών στο μονοπάτι μέχρι τη ρίζα.
- **Υψος δένδρου:** το μεγαλύτερο βάθος κάποιας κορυφής.
- **Επίπεδο:** όλες οι κορυφές με το ίδιο βάθος βρίσκονται στο ίδιο επίπεδο.
- Η ρίζα του δένδρου βρίσκεται σε επίπεδο μηδέν.

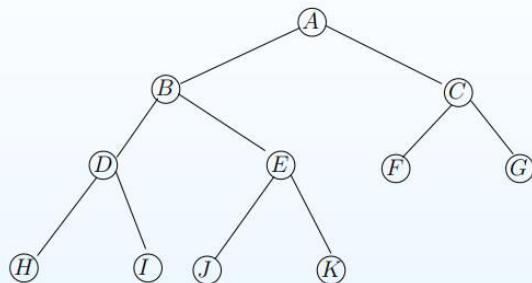
Δυαδικά Δένδρα

- Κάθε κόμβος έχει το πολύ δύο παιδιά
- Διακρίνουμε μεταξύ αριστερού και δεξιού παιδιού (υποδένδρου).
- Αν σε κάθε επίπεδο υπάρχουν όλοι οι κόμβοι τότε το δένδρο λέγεται πλήρες.
- Το επίπεδο d έχει το πολύ 2^d κορυφές
- Αν h το ύψος του δένδρου, τότε

$$h + 1 \leq n \leq 2^{h+1} - 1 \Rightarrow \lg(n + 1) \leq h \leq n - 1.$$

- Αν $n_i, i = 0, 1, 2$ είναι το πλήθος των κορυφών με i παιδιά τότε $n_0 = n_2 + 1$
- Ένα πλήρες δυαδικό δένδρο έχει συνολικά $2^{h+1} - 1$ κορυφές, 2^h φύλλα και $2^h - 1$ εσωτερικές κορυφές.

Διάσχιση Δένδρου

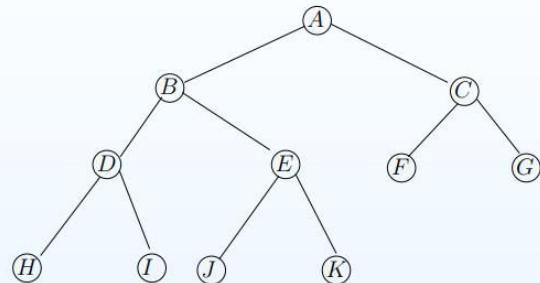


Σχήμα 14: Δυαδικό δένδρο.

Ένδο-διατεταγμένη (inorder) διέλευση:

- Αναδρομική διέλευση αριστερού υποδένδρου.
- Επεξεργασία ρίζας.
- Αναδρομική διέλευση δεξιού υποδένδρου.

HDIBJEKAFCG

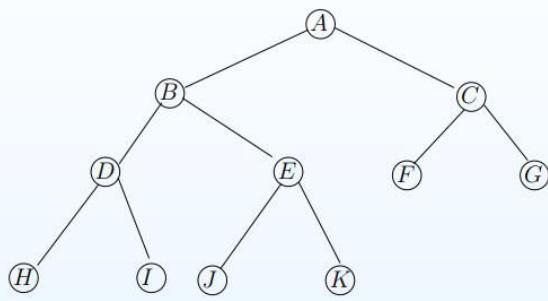


Σχήμα 14: Δυαδικό δένδρο.

Προ-διατεταγμένη (preorder) διέλευση:

- Επεξεργασία ρίζας.
- Αναδρομική διέλευση αριστερού υποδένδρου.
- Αναδρομική διέλευση δεξιού υποδένδρου.

ABDHIEJKCFG



Σχήμα 14: Δυαδικό δένδρο.

Μετα-διατεταγμένη (postorder) διέλευση :

- Αναδρομική διέλευση αριστερού υποδένδρου.
- Αναδρομική διέλευση δεξιού υποδένδρου.
- Επεξεργασία ρίζας.

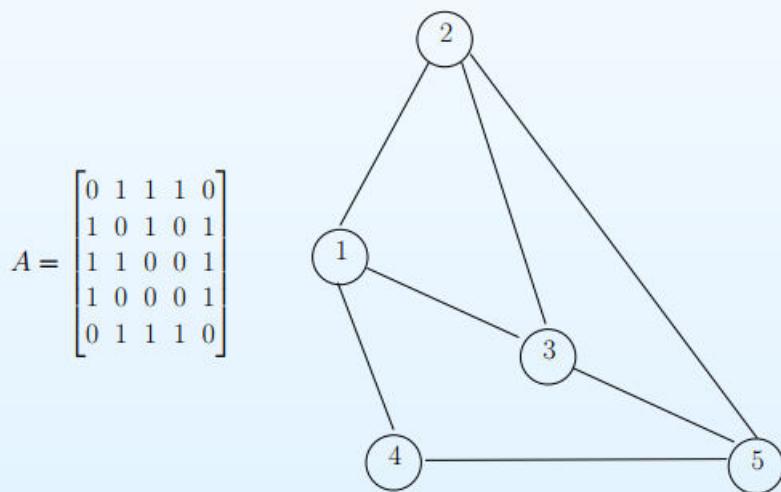
HIDJKEBFGCA

02_ΜΗ-ΚΑΤΕΥΘΥΝΟΜΕΝΑ ΓΡΑΦΗΜΑΤΑ / 02_graphreprtransv

Αναπαράσταση

Πίνακας

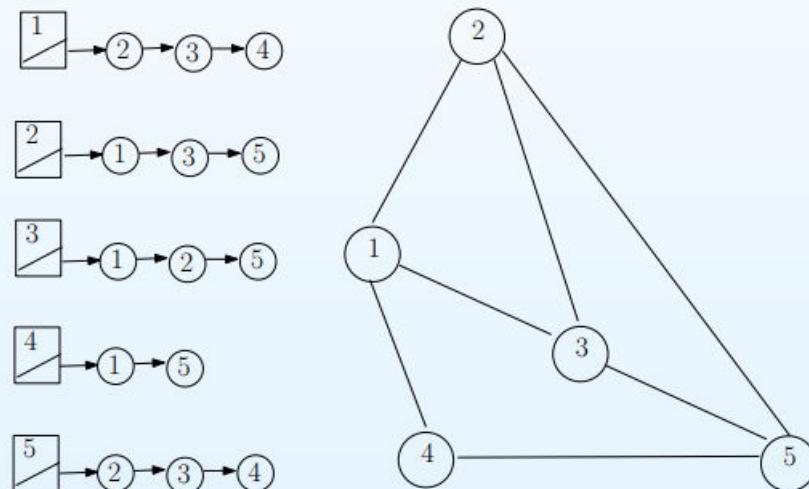
Ο πρώτος τρόπος αναπαράστασης είναι μέσω του **πίνακα γειτνίασης** ή **μητρώο σύνδεσης**. Ο πίνακας έχει n γραμμές και n στήλες - αντιστοιχούν στις κορυφές του γραφήματος· στην γραμμή v , στήλη u υπάρχει η τιμή 1 αν $\{v, u\} \in E$ και 0 διαφορετικά.



Σχήμα 1: Πίνακας Γειτνίασης

Λίστα

Εναλλακτικά μπορούμε να χρησιμοποιήσουμε απεικόνιση μέσω λιστών: για κάθε κόμβο υπάρχει μία λίστα όπου εμφανίζονται οι γείτονες του κόμβου αυτού με τυχαία σειρά (συνήθως χρησιμοποιούμε λεξικογραφική σειρά)



Σχήμα 2: Λίστα Γειτνίασης

Διάσχιση

Διάσχιση

Πρόβλημα 1 Επίσκεψη των κορυφών του γραφήματος $G(V, E)$ με συστηματικό τρόπο.

Η επίλυση του παραπάνω προβλήματος ισοδυναμεί με τον “ύπολογισμό” γεννητορικού δένδρου του $G(V, E)$. Υπάρχουν δύο στρατηγικές:

- Διάσχιση κατά Βάθος (Depth-First Search)
- Διάσχιση κατά Πλάτος (Breadth-First Search)

Οι στρατηγικές αυτές αποτελούν γενικεύσεις των αλγορίθμικών διαδικασιών διάσχισης δένδρου που είδαμε σε προηγούμενο μάθημα.

Ο αλγόριθμος βασίζεται στην παρακάτω αναδρομική διαδικασία η οποία δέχεται σαν είσοδο μία κορυφή v και επισκέπτεται αναδρομικά το αριστερό και μετά το δεξιό υποδένδρο. Σε κάθε φάση ο μονοδ. πίνακας $visited$ έχει στη θέση v τιμή $true$ αν η κορυφή v έχει ‘άνακαλυφθεί.’

explore($v, visited$)

$visited[v] \leftarrow true;$

for $u \in N(v)$

if not $visited[u]$ **then** $\text{explore}(u, visited);$

Η διαδικασία αυτή καλείται από το “κύριο” πρόγραμμα:

main()

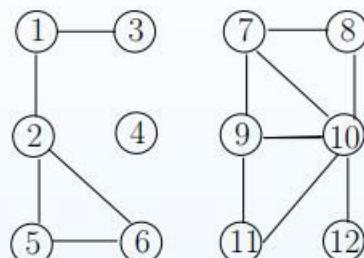
for $v \in V$

$visited[v] \leftarrow false;$

for $v \in V$

if not $visited[v]$ **then** $\text{explore}(v, visited);$

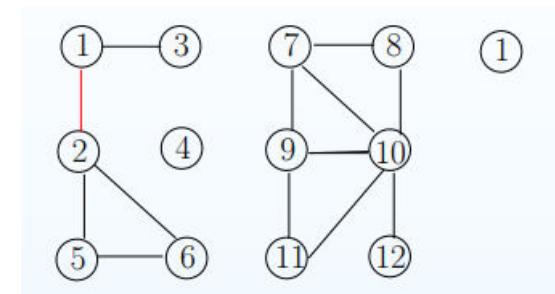
Παράδειγμα



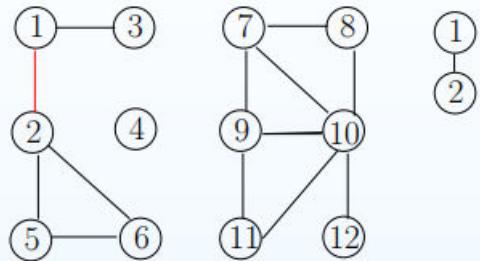
Σχήμα 3: Γράφημα

Θα εκτελέσουμε ΔκΒ στο παραπάνω γράφημα θεωρώντας ότι οι κόμβοι στη λίστα γειτνίασης εμφανίζονται με λεξικογραφική σειρά.

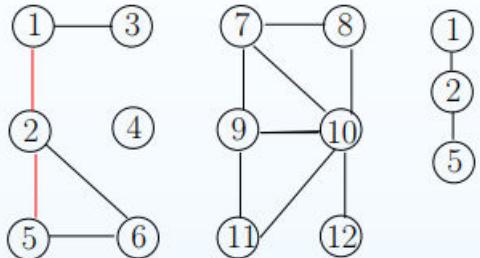
Στα παρακάτω σχήματα απεικονίζεται το χτίσιμο του γεννητορικού δάσους - οι ακμές οι οποίες οδηγούν σε κορυφές που έχει ο αλγόριθμος ήδη επισκεφτεί εμφανίζονται διακεκομμένες.



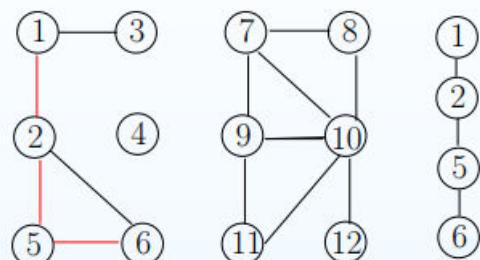
Σχήμα 4: Γράφημα - ΔκΒ



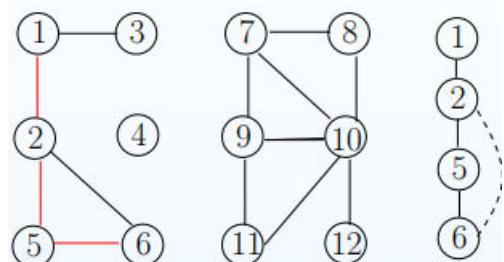
Σχήμα 4: Γράφημα - ΔκΒ



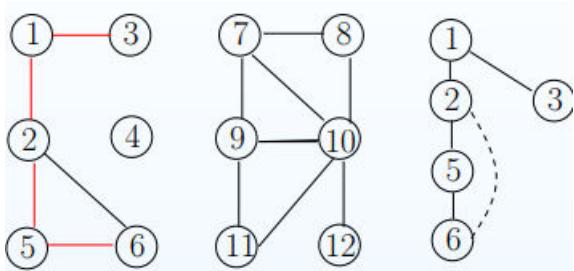
Σχήμα 4: Γράφημα - ΔκΒ



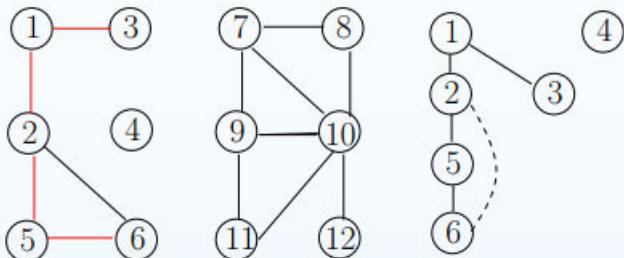
Σχήμα 4: Γράφημα - ΔκΒ



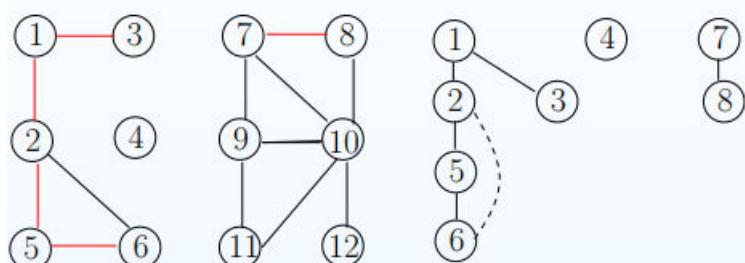
Σχήμα 4: Γράφημα - ΔκΒ



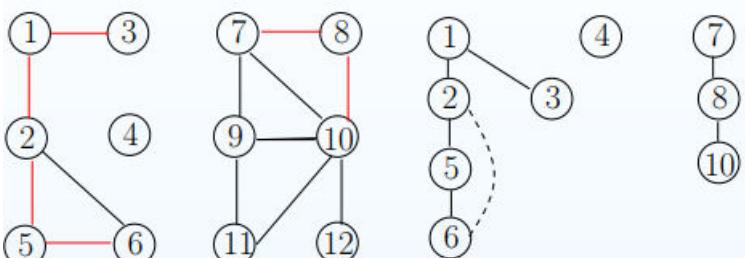
Σχήμα 4: Γράφημα - ΔκΒ



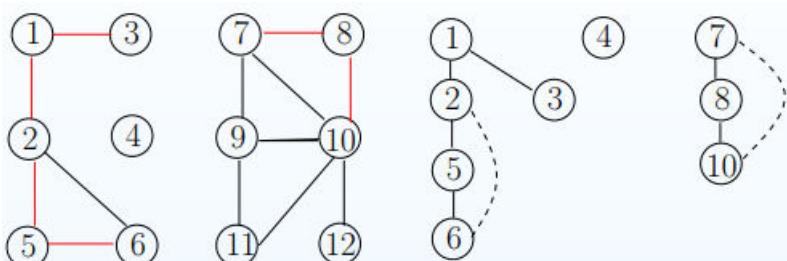
Σχήμα 4: Γράφημα - ΔκΒ



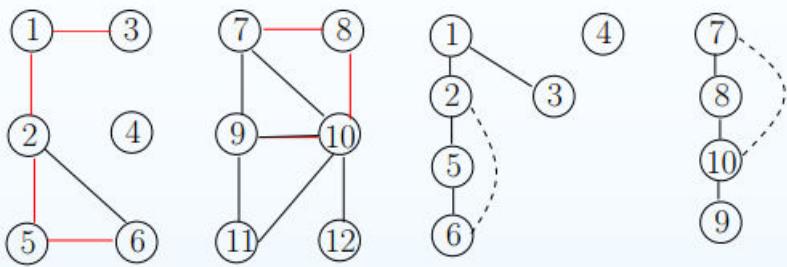
Σχήμα 4: Γράφημα - ΔκΒ



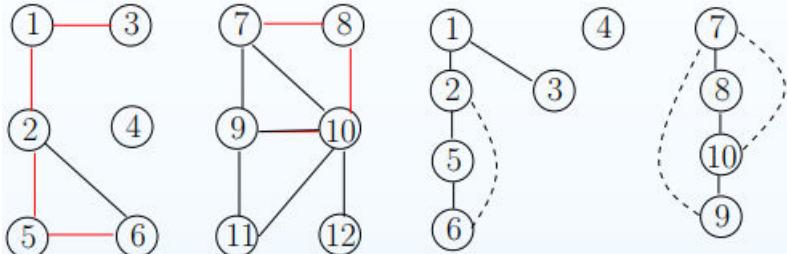
Σχήμα 4: Γράφημα - ΔκΒ



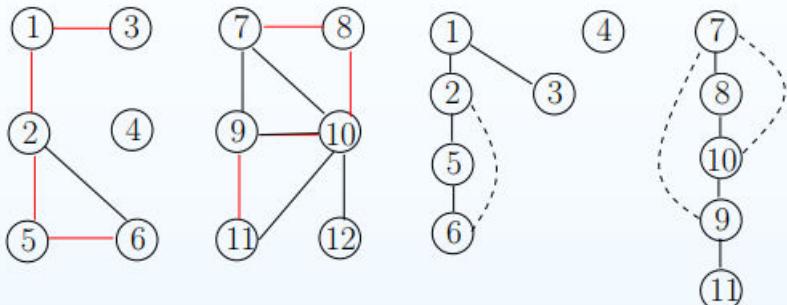
Σχήμα 4: Γράφημα - ΔκΒ



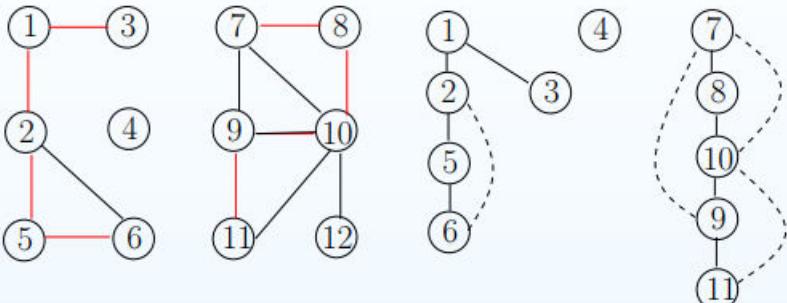
Σχήμα 4: Γράφημα - ΔκΒ



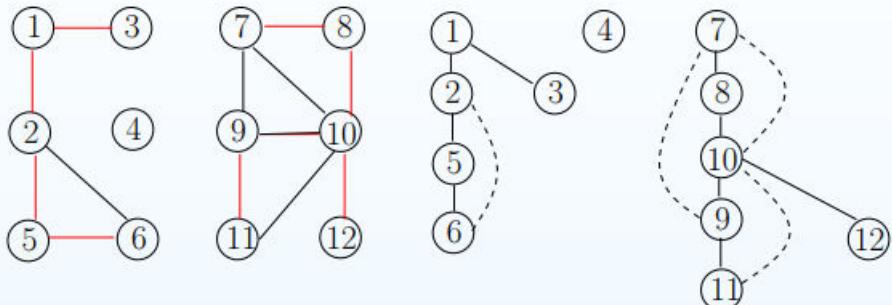
Σχήμα 4: Γράφημα - ΔκΒ



Σχήμα 4: Γοάφημα - ΔκΒ



Σχήμα 4: Γράφημα - ΔκΒ



Σχήμα 4: Γράφημα - ΔκΒ

ΔκΒ με στοίβα

Είναι προφανές ότι η ΔκΒ στην πραγματικότητα υλοποιεί έναν μηχανισμό στοίβας: όταν επισκέπτεται έναν κόμβο αποθηκεύει τους γείτονες σε μία ουρά Q με ιεραρχία LIFO. Μπορούμε να εξαλείψουμε την αναδρομή χρησιμοποιώντας την Q . Η διαδικασία *explore* γίνεται:

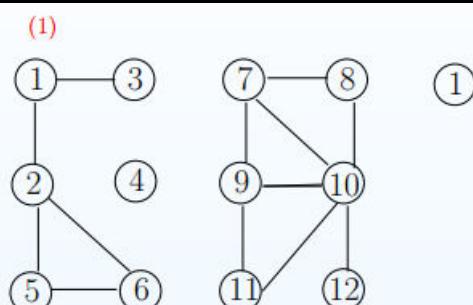
```
explore(clock,  $Q$ , visited, preorder)
while  $Q \neq \emptyset$ 
     $v \leftarrow \text{pop}(Q)$ ;
    if not visited[ $v$ ] then
        visited[ $v$ ]  $\leftarrow \text{true}$ ;
        preorder[ $v$ ]  $\leftarrow ++\text{clock}$ ;
        for ( $u \in N(v)$ ) and (not visited[ $u$ ]) push( $u$ ,  $Q$ );
```

Η μεταβλητή $clock$ χρησιμοποιείται για να μπορούμε να καταγράψουμε τη σειρά επίσκεψης. Η καταγραφή γίνεται στον μονοδιάστατο πίνακα *preorder*.

Το “κύριο” πρόγραμμα γίνεται:

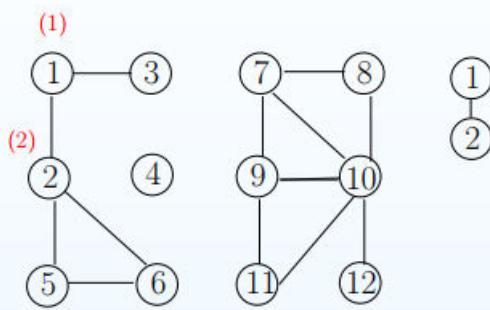
```
main()
    create  $Q$ ;
     $clock \leftarrow 0$ ;
    for  $v \in V$ 
        visited[ $v$ ]  $\leftarrow \text{false}$ ; preorder[ $v$ ]  $\leftarrow 0$ ;
    for  $v \in V$ 
        if not visited[ $v$ ] then
            push( $v$ ,  $Q$ );
            explore( $clock$ ,  $Q$ , visited, preorder);
```

Παράδειγμα



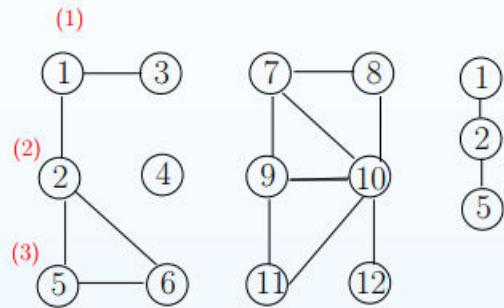
$Q : 2 - 3$

Σχήμα 5: Γράφημα - ΔκΒ



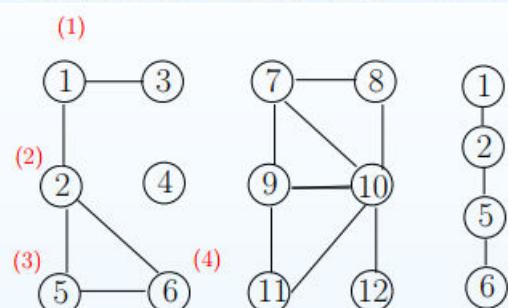
$$Q : 5 - 6 - 3$$

Σχήμα 5: Γράφημα - ΔκΒ



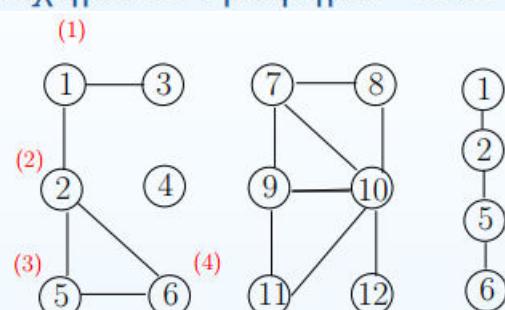
$$Q : 6 - 6 - 3$$

Σχήμα 5: Γράφημα - ΔκΒ



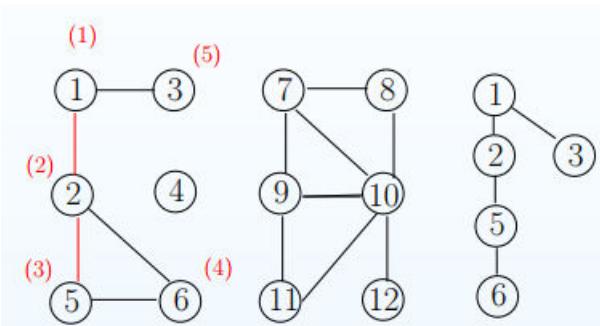
$$Q : 6 - 3$$

Σχήμα 5: Γράφημα - ΔκΒ



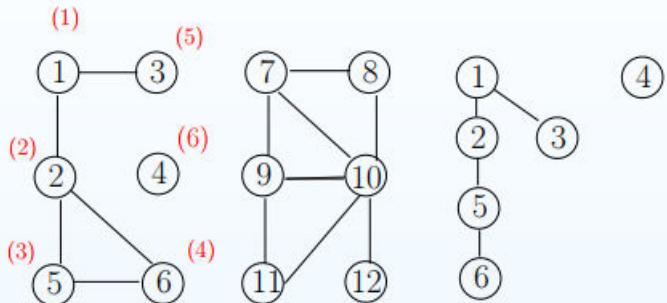
$$Q : 3$$

Σχήμα 5: Γράφημα - ΔκΒ



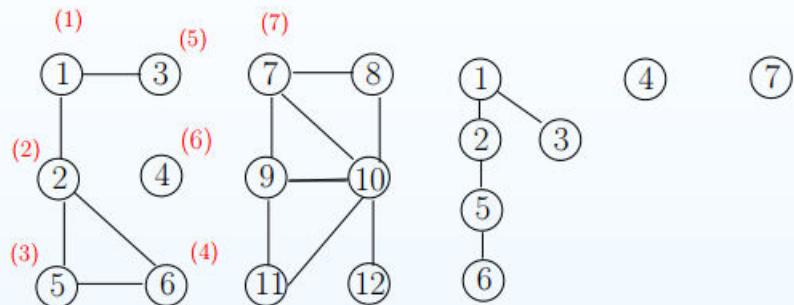
$Q :$

Σχήμα 5: Γράφημα - ΔκΒ



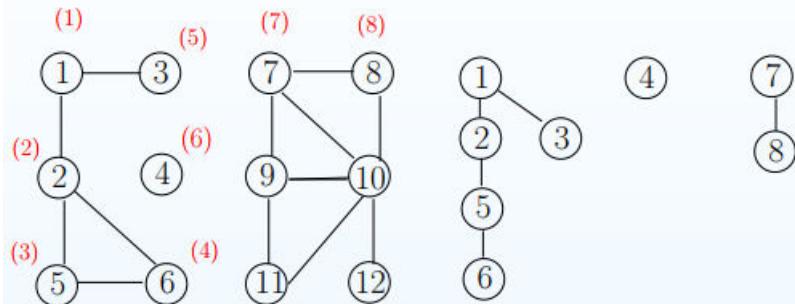
$Q :$

Σχήμα 5: Γράφημα - ΔκΒ



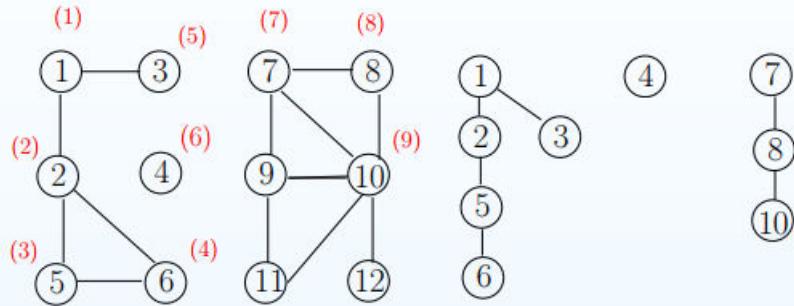
$Q : 8 - 9 - 10$

Σχήμα 5: Γράφημα - ΔκΒ



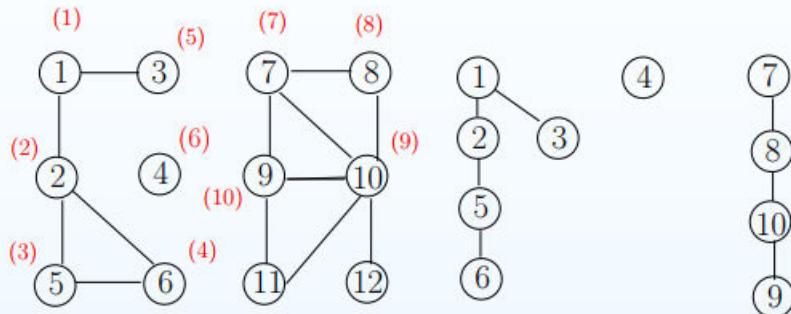
$Q : 10 - 9 - 10$

Σχήμα 5: Γράφημα - ΔκΒ



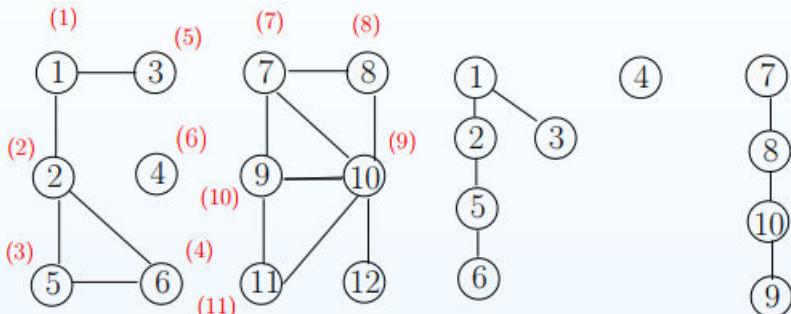
$$Q : 9 - 11 - 12 - 9 - 10$$

Σχήμα 5: Γράφημα - ΔκΒ



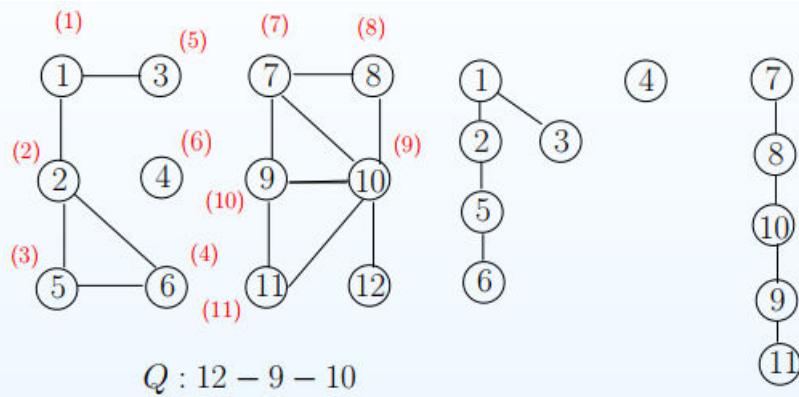
$$Q : 11 - 11 - 12 - 9 - 10$$

Σχήμα 5: Γράφημα - ΔκΒ

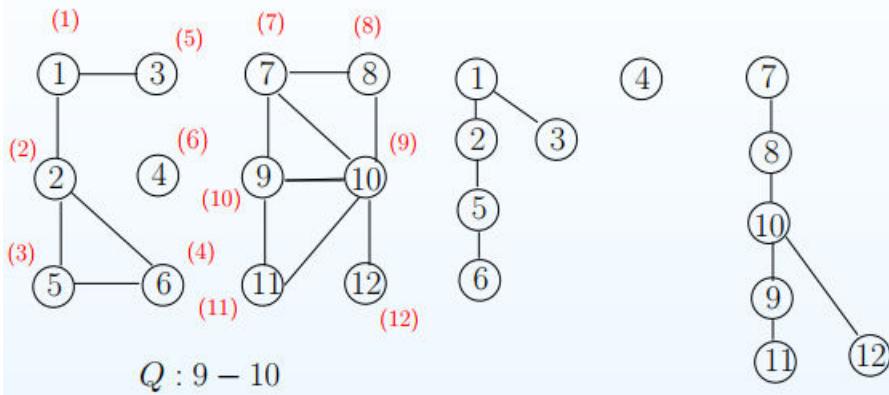


$$Q : 11 - 12 - 9 - 10$$

Σχήμα 5: Γράφημα - ΔκΒ



Σχήμα 5: Γράφημα - ΔκΒ



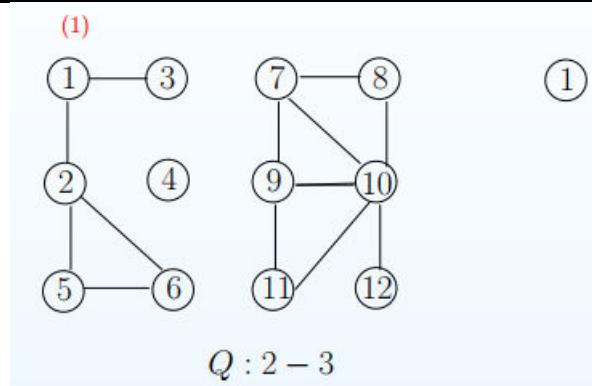
Σχήμα 5: Γράφημα - ΔκΒ

ΔκΠ

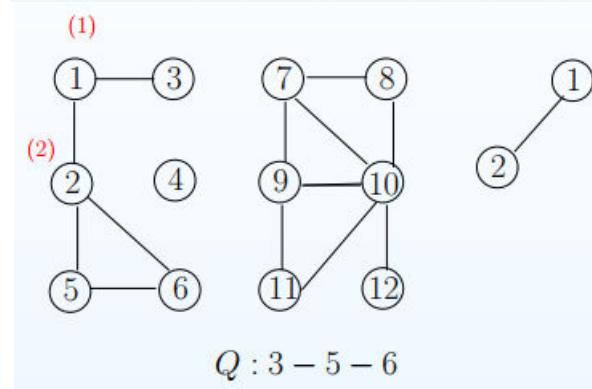
Η Διάσχιση κατά Πλάτος (ΔκΠ) πρώτα επισκέπτεται όλους τους κόμβους που βρίσκονται στο ίδιο επίπεδο και μετά προχωρά στους επόμενους κόμβους. Προκειμένου να υλοποιηθεί αυτή η στρατηγική η μόνη αλλαγή που χρειάζεται να γίνει είναι στην επεξεργασία της ουράς Q . Αντί για LIFO εφαρμόζουμε FIFO. Έτσι όταν προσθέτουμε τους γείτονες ενός κόμβου τους βάζουμε στο τέλος της Q (και όχι στην αρχή.) Η διαδικασία εξερεύνησης γίνεται:

```
explore(clock, Q, visited, preorder)
while  $Q \neq \emptyset$ 
   $v \leftarrow \text{pop}(Q);$ 
  if not visited[ $v$ ] then
    visited[ $v$ ]  $\leftarrow \text{true};$ 
    preorder[ $v$ ]  $\leftarrow ++\text{clock};$ 
    for ( $u \in N(v)$ ) and (not visited[ $u$ ]) append( $u$ ,  $Q$ );
```

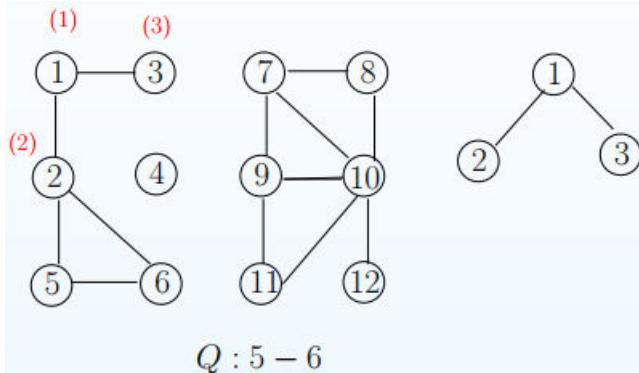
Παράδειγμα



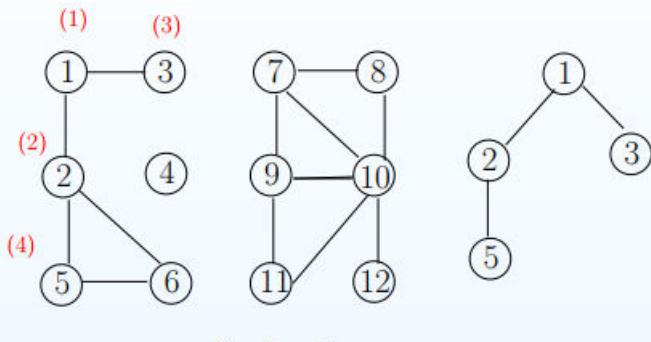
Σχήμα 6: Γράφημα - ΔκΠ



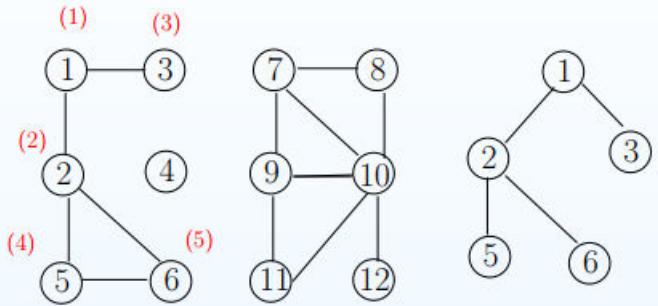
Σχήμα 6: Γράφημα - ΔκΠ



Σχήμα 6: Γράφημα - ΔκΠ

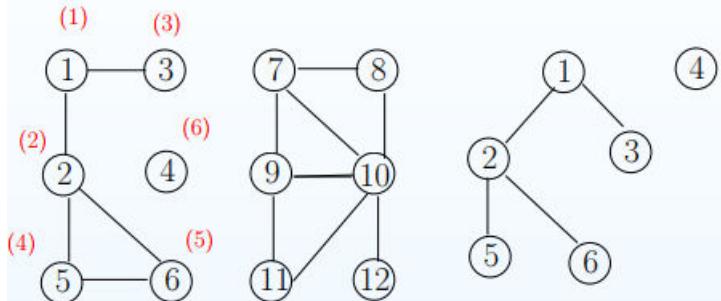


Σχήμα 6: Γράφημα - ΔκΠ



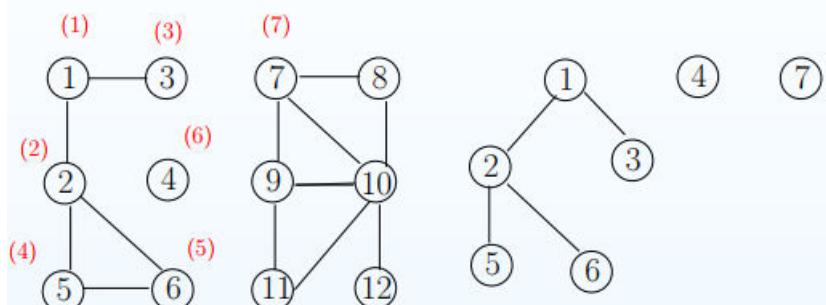
$Q : 6$

Σχήμα 6: Γράφημα - ΔκΠ



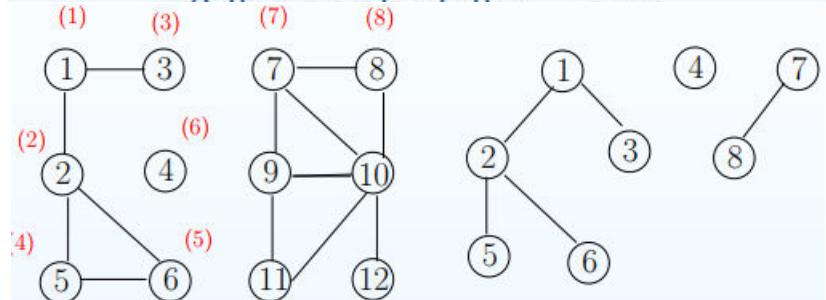
$Q :$

Σχήμα 6: Γράφημα - ΔκΠ



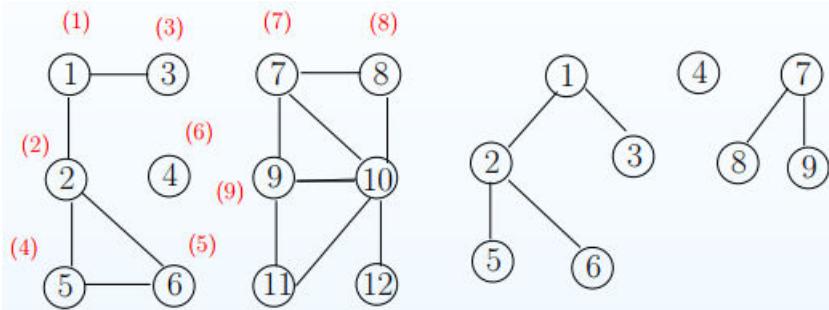
$Q : 8 - 9 - 10$

Σχήμα 6: Γράφημα - ΔκΠ



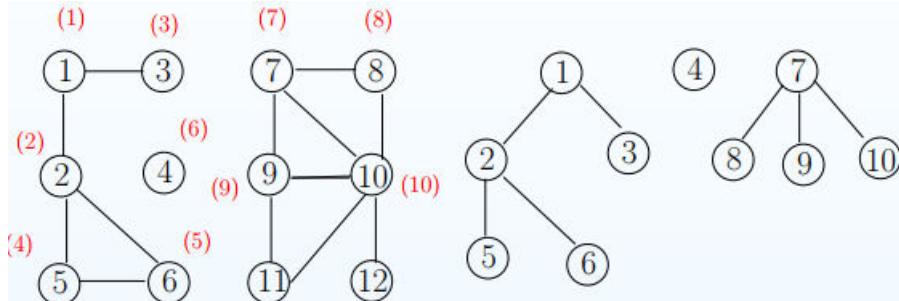
$Q : 9 - 10 - 10$

Σχήμα 6: Γράφημα - ΔκΠ



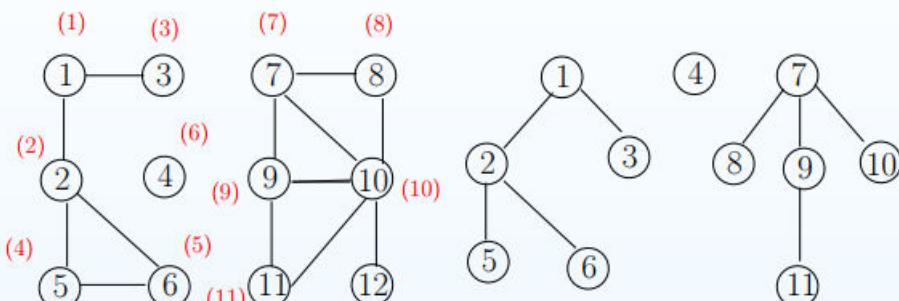
$$Q : 10 - 10 - 10 - 11$$

Σχήμα 6: Γράφημα - ΔκΠ



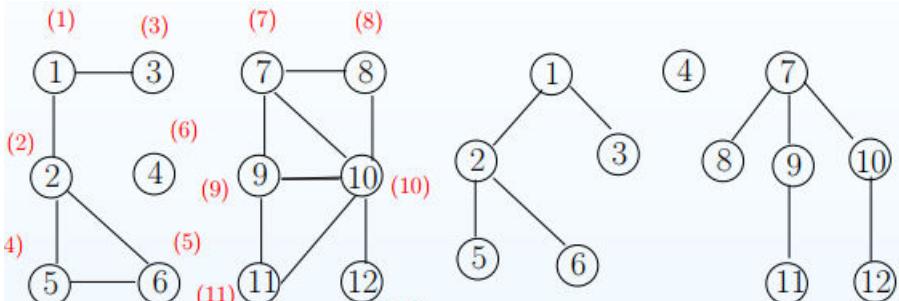
$$Q : 10 - 10 - 11 - 11 - 12$$

Σχήμα 6: Γράφημα - ΔκΠ



$$Q : 11 - 12$$

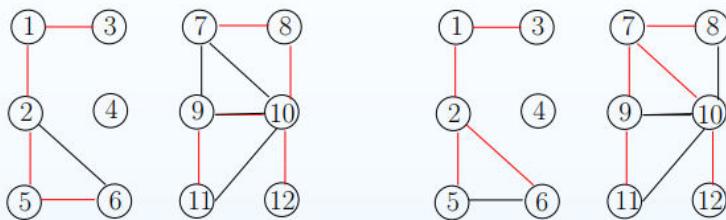
Σχήμα 6: Γράφημα - ΔκΠ



$$Q :$$

Σχήμα 6: Γράφημα - ΔκΠ

Παρατηρήσεις



Σχήμα 7: Γεννητορικά δένδρα ΔκΒ και ΔκΠ (κόκκινες ακμές)

Στο Σχήμα 7 απεικονίζονται τα δένδρα που σχημάτισαν οι δυο στρατηγικές διάσχισης. Από κατασκευής το δένδρο της ΔκΠ έχει την ιδιότητα ότι κάθε μονοπάτι που συνδέει τη ρίζα v με ένα φύλλο w είναι το συντομότερο ως προς τον αριθμό των ακμών. Για παράδειγμα το μονοπάτι ανάμεσα στις κορυφές 7, 11 με το μικρότερο αριθμό ακμών είναι το 7-9-11. Το αντίστοιχο μονοπάτι που υπολόγισε η ΔκΒ είναι το 7-8-10-9-11.

Πολυπλοκότητα

Αμφότερες οι στρατηγικές διάσχισης επισκέππονται τις κορυφές του γραφήματος δια μέσου των ακμών.

Παρατηρούμε ότι κάθε φορά που ‘άνακαλύπτεται’ μία ακμή εκτελείται σταθερός αριθμόν στοιχειωδών πράξεων. Επίσης η κάθε ακμή ‘άνακαλύπτεται’ δύο φορές (μία φορά για κάθε προσπίπουσα κορυφή). Άρα μπορούμε να θεωρήσουμε ότι για κάθε ακμή εκτελούνται το πολύ c στοιχειώδεις πράξεις. Άρα αν $T(m)$ είναι ο αριθμός των στοιχειωδών πράξεων της διάσχισης ενός γραφήματος με m ακμές, έχουμε

$$T(m) \leq B(m), \text{ όπου } B(m) = B(m - 1) + c.$$

Σε κλειστή μορφή έχουμε $B(m) = c \cdot m$ και άρα

$$T(m) = O(n + m)$$

εφόσον στην $T(m)$ συνυπολογίσουμε και το κόστος αρχικοποίησης των δομών που χρησιμοποιεί η διάσχιση.

Χρησιμότητα

Η ΔκΒ είναι ιδιαίτερα χρήσιμη γιατί, εκτός από την ανάδειξη ενός γεννητορικού δένδρου, μπορεί να “διαγνώσει” πολλά χαρακτηριστικά ενός γραφήματος $G(V, E)$. Ενδεικτικά,

- Υπάρχει μονοπάτι που να συνδέει δύο κορυφές v, u ;
- Είναι το γράφημα άκυκλο;
- Είναι το γράφημα συνδεδεμένο;

Θα εμπλουτίσουμε τον αλγόριθμο της ΔκΒ ώστε να καταγράφονται πληροφορίες που θα βοηθούν στην αποκάλυψη (και άλλων) χαρακτηριστικών ενός μη-κατευθυνόμενου γραφήματος.

Διάσχιση

Θεωρούμε την αναδρομική έκδοση του αλγόριθμου όπως αυτός αποτυπώνεται από την παρακάτω διαδικασία

```
explore( $v, visited$ )
 $visited[v] \leftarrow true$ ;
for  $u \in N(v)$ 
    if not  $visited[u]$  then  $\text{explore}(u, visited)$ ;
```

Έχουμε ήδη δει ότι η παραπάνω διαδικασία μπορεί να εμπλουτίσει με τη μεταβλητή $clock$ και τον πίνακα $preorder[]$. Ο μηχανισμός αυτός καταγράφει την πρώτη φορά που συναντάμε κάθε κορυφή κατά την διάρκεια της διάσχισης. Επίσης θα χρησιμοποιήσουμε τον πίνακα $parent[]$ ο οποίος για κάθε κορυφή u θα καταγράφει τον άμεσο πρόγονο της στο δένδρο που δημιουργεί η ΑκΒ. Έτσι η ακμή $\{v, u\}$ θα ανήκει στο δένδρο ανν $v = parent[u]$. Η ρίζα του δένδρου r θα έχει $parent[r] = 0$.

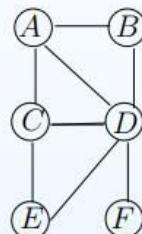
ΔκΒ

```
explore(clock, v, visited, preorder, parent)
    visited[v] ← true;
    preorder[v] ← ++clock;
    for u ∈ N(v)
        if not visited[u] then
            parent[u] ← v;
            explore(clock, u, visited, preorder, parent);

main()
    clock ← 0;
    for v ∈ V
        visited[v] ← false;
        preorder[v] ← 0; parent[v] ← 0;
    for v ∈ V
        if not visited[v] then
            explore(clock, v, visited, preorder, parent);
```

Παράδειγμα

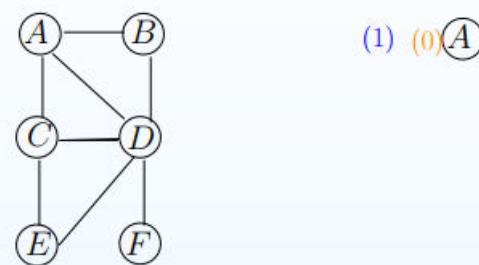
Θα εφαρμόσουμε τον αλγόριθμο στο γράφημα του Σχήματος 8.



Σχήμα 8: Γράφημα - ΔκΒ

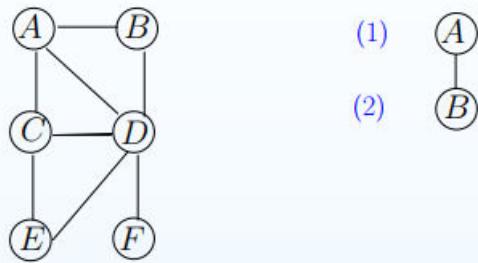
Οι τιμές του πίνακα preorder απεικονίζονται με μπλέ χρώμα.

Παράδειγμα – Εκτέλεση



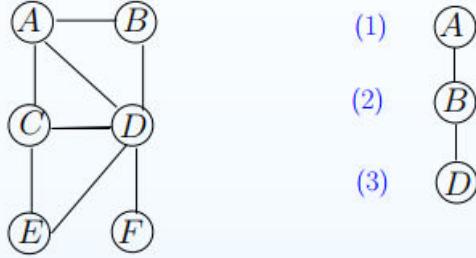
parent = (0, 0, 0, 0, 0, 0)

Σχήμα 9: Γράφημα - ΔκΒ



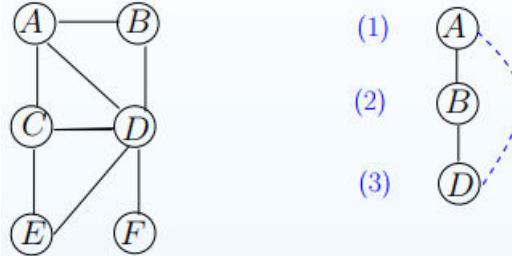
parent= (0, A, 0, 0, 0, 0)

Σχήμα 9: Γράφημα - ΔκΒ



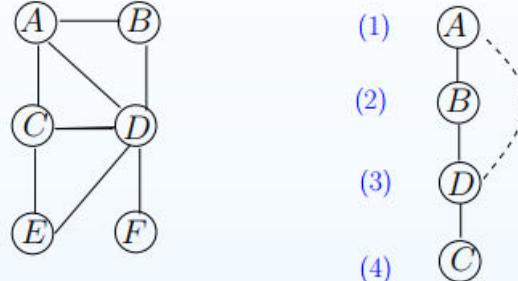
parent= (0, A, 0, B, 0, 0)

Σχήμα 9: Γράφημα - ΔκΒ



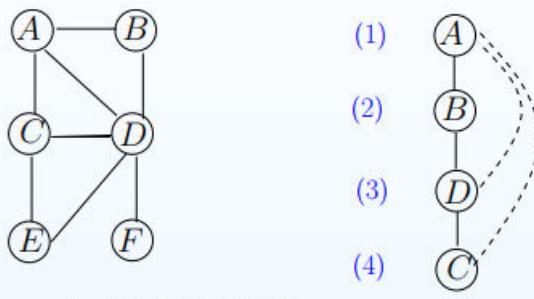
parent= (0, A, 0, B, 0, 0)

Σχήμα 9: Γράφημα - ΔκΒ



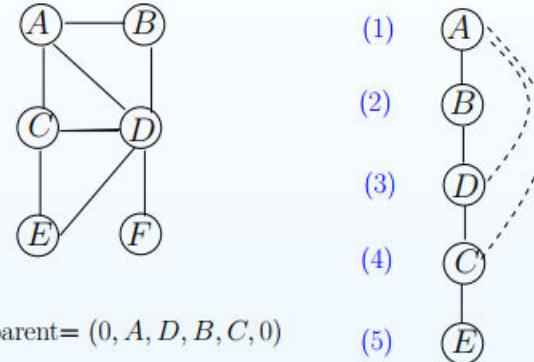
parent= (0, A, D, B, 0, 0)

Σχήμα 9: Γράφημα - ΔκΒ



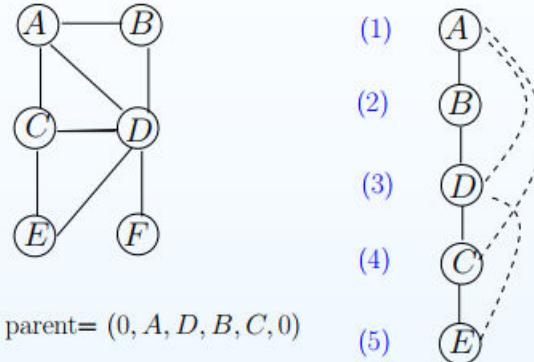
parent= (0, A, D, B, 0, 0)

Σχήμα 9: Γράφημα - ΔκΒ



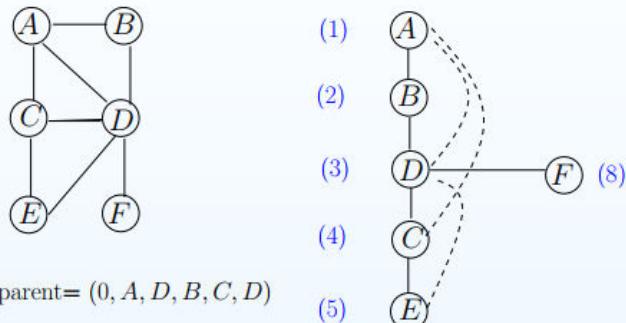
parent= (0, A, D, B, C, 0)

Σχήμα 9: Γράφημα - ΔκΒ



parent= (0, A, D, B, C, 0)

Σχήμα 9: Γράφημα - ΔκΒ



parent= (0, A, D, B, C, D)

Σχήμα 9: Γράφημα - ΔκΒ

Έστω T το δένδρο της ΔκΒ. Για κάθε ακμή (v, u) (κορυφές εμφανίζονται στο ζευγάρι με τη σειρά που η ακμή προστίθεται στο δένδρο) $parent[u] = v$, αν η ακμή ανήκει στο T .

Ιδιότητες

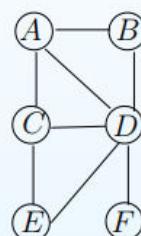
Μετά την εκτέλεση της ΔκΒ, οι ακμές του G που δεν ανήκουν στο σύνολο του παραχθέντος γεννητορικού δένδρου ονομάζονται οπισθοακμές. Αυτές απεικονίζονται στο Σχήμα 9 σαν διακεκομένες.

Mία οπισθοακμή πιστοποιεί την ύπαρξη κύκλου.

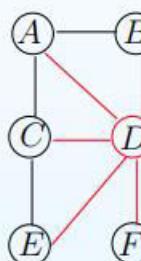
Εφόσον μπορούμε μέσω του μονοδιάστατου πίνακα parent να διαπιστώσουμε αν μία ακμή ανήκει στο δένδρο αμέσως γνωρίζουμε ότι αν δεν ανήκει αποτελεί οπισθοακμή και άρα στο γράφημα υπάρχει κύκλος.

Σημείο Κοπής

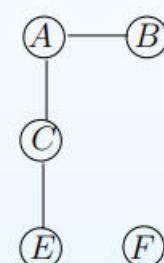
Μία από τις πολλές εφαρμογές της ΔκΒ είναι η εύρεση των σημείων κοπής, δηλαδή η εύρεση των κορυφών των οποίων η αφαίρεση αποσυνθέτει το γράφημα σε γραφικές συνιστώσες.



Σχήμα 10: Σημείο Κοπής



Σχήμα 10: Σημείο Κοπής



Σχήμα 10: Σημείο Κοπής

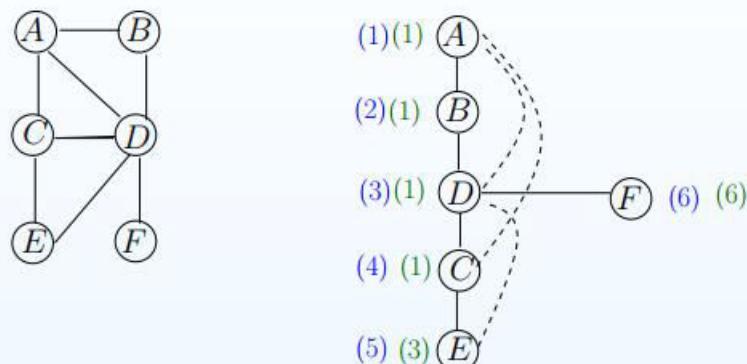
Θεωρούμε ότι έχει εκτελεστεί η ΔκΒ και σε κάθε κορυφή v έχει αποδοθεί τιμή $preorder[v]$.

Στη συνέχεια, για κάθε κορυφή $v \in V$, πρέπει να υπολογιστεί μία τιμή $low[v]$, η οποία να είναι η μικρότερη τιμή $preorder[u]$ στην οποία μπορεί να φτάσει ένα μονοπάτι που να ξεκινάει από την v ή κάποιο απόγονό της στο δένδρο της ΔκΒ και να χρησιμοποιεί ακριβώς μία οπισθοακμή.

Μία κορυφή $v \in V$ θα είναι σημείο κοπής αν είτε

- είναι ρίζα του δένδρου ΔκΒ και έχει πάνω από ένα κόμβο παιδί, ή,
- έχει παιδί u στο δένδρο ΔκΒ (δηλαδή γειτονική κορυφή στο δένδρο της ΔκΒ) με τιμή $low[u] \geq preorder[v]$.

Παράδειγμα



Σχήμα 11: Τιμές *low*

Στο Σχήμα 11 απεικονίζεται ένα γράφημα και το αντίστοιχο δένδρο ΔκΒ με τις τιμές *preorder* (μπλέ) και *low* (πράσινο). Παρατηρείστε ότι η κορυφή D έχει παιδί τον κόμβο F και ισχύει $low[F] > preorder[D]$. Επομένως ο D είναι σημείο κοπής.

Θεωρούμε ότι έχει ήδη εκτελεστεί ΔΚΒ και οι πίνακες *preorder*, *parent* έχουν ήδη πάρει τιμές.

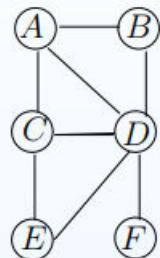
```

comp_low(v, visited, preorder, parent, low, cut_node)
    visited[v]  $\leftarrow$  true;
    low[v]  $\leftarrow$  preorder[v];
    for u  $\in$  N(v)
        if not visited[u] then
            comp_low(u, visited, preorder, parent, low, cut_node)
            if low[v]  $>$  low[u] then
                low[v]  $\leftarrow$  low[u];
            if low[u]  $\geq$  preorder[v] then
                cut_node[v]  $\leftarrow$  true;
            else /* visited[u] = true */
                if u  $\neq$  parent[v] then /*(v, u) οπισθοακμή*/
                    if low[v]  $>$  preorder[u] then
                        low[v]  $\leftarrow$  preorder[u];

main()
    clock  $\leftarrow$  0;
    for v  $\in$  V
        visited[v]  $\leftarrow$  cut_node[v]  $\leftarrow$  false;
        preorder[v]  $\leftarrow$  parent[v]  $\leftarrow$  0;
    for v  $\in$  V
        if not visited[v] then
            explore(clock, v, visited, preorder, parent);
    for v  $\in$  V visited[v]  $\leftarrow$  false;
    for v  $\in$  V
        if not visited[v] then
            comp_low(v, visited, preorder, parent, low, cut_node);
    for v  $\in$  V
        if cut_node[v] then
            if parent[v]  $\neq$  0 then
                print(v);
            else /*κορυφή v είναι ρίζα της ΔΚΒ*/
                num_root_children  $\leftarrow$  0;
                for u  $\in$  N(v)
                    if parent[u] = v then num_root_children  $\leftarrow$  num_root_children + +;
                    if num_root_children  $\geq$  2 then
                        print(v);
```

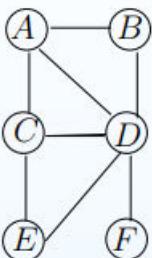
Παρατήρηση: αν υπάρχει οπισθοακμή προς τη ρίζα του δένδρου της ΔκΒ τότε αυτή χαρακτηρίζεται (πιθανώς) λανθασμένα από τον αλγόριθμο σαν σημείο κοπής. Η ρίζα είναι σημείο κοπής μόνο αν έχει τουλάχιστον δυο παιδιά στο δένδρο της ΔκΒ. Αυτό ακριβώς ελέγχεται στον παραπάνω κώδικα.

Υπολογισμός low - Παράδειγμα



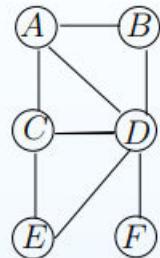
- | | | |
|--------|-----|---------|
| (1)(1) | (A) | |
| (2) | (B) | |
| (3) | (D) | (F) (6) |
| (4) | (C) | |
| (5) | (E) | |

Σχήμα 12: Υπολογισμός *low*



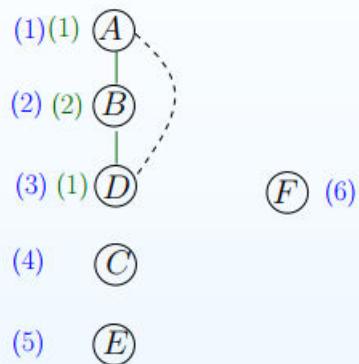
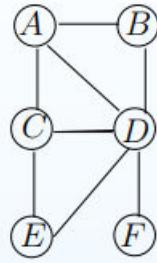
- | | | |
|---------|-----|---------|
| (1)(1) | (A) | |
| (2) (2) | (B) | |
| (3) | (D) | (F) (6) |
| (4) | (C) | |
| (5) | (E) | |

Σχήμα 12: Υπολογισμός *low*

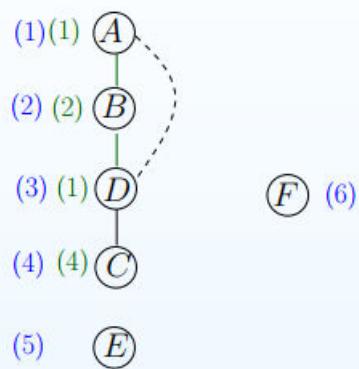
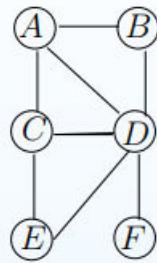


- | | | |
|---------|-----|---------|
| (1)(1) | (A) | |
| (2) (2) | (B) | |
| (3) (3) | (D) | (F) (6) |
| (4) | (C) | |
| (5) | (E) | |

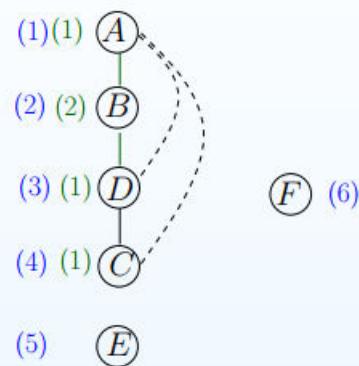
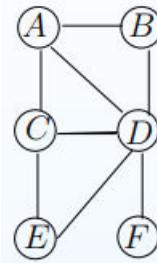
Σχήμα 12: Υπολογισμός *low*



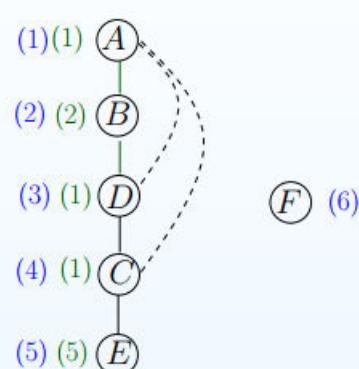
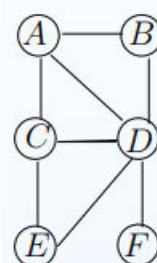
Σχήμα 12: Υπολογισμός *low*



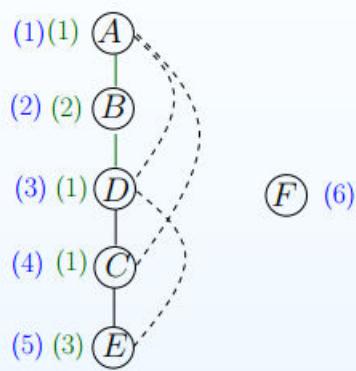
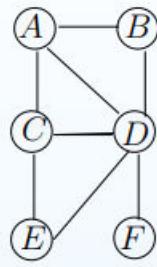
Σχήμα 12: Υπολογισμός *low*



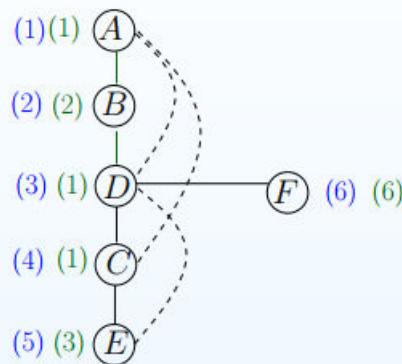
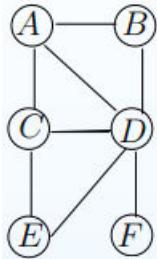
Σχήμα 12: Υπολογισμός *low*



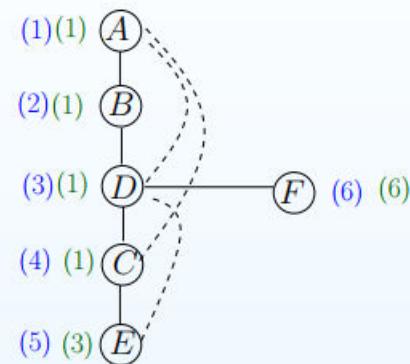
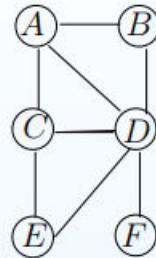
Σχήμα 12: Υπολογισμός *low*



Σχήμα 12: Υπολογισμός *low*



Σχήμα 12: Υπολογισμός *low*



Σχήμα 12: Υπολογισμός *low*

Επειδή

$$\text{low}[F] = 6 \geq \text{preorder}[D] = 3$$

η κορυφή D αναγνωρίζεται σαν σημείο κοπής. Πράγματι η διαγραφή της από το γράφημα δημιουργεί μία γραφική συνιστώσα που περιέχει την κορυφή F και μία άλλη με όλες τις υπόλοιπες κορυφές.

Γέφυρες

Με τον υπολογισμό του πίνακα low μπορούμε να υπολογίσουμε και τις γέφυρες του γραφήματος.

Μία ακμή (v, u) ($v = parent[u]$) είναι γέφυρα ANN

$$low[u] = preorder[u]$$

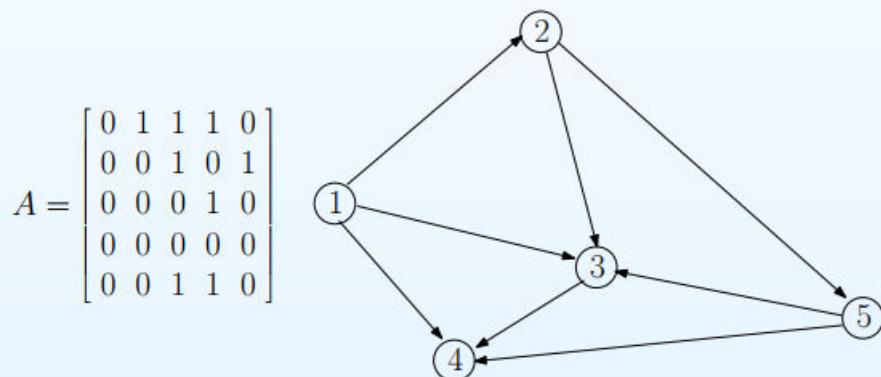
(γιατί ;)

03_ΚΑΤΕΥΘΥΝΟΜΕΝΑ ΓΡΑΦΗΜΑΤΑ / 03_graphtransdir

Αναπαράσταση

Πίνακας

Κατά τρόπο αντίστοιχο των μη-κατευθυνομένων γραφημάτων ο πίνακας γειτνίασης έχει n γραμμές και n στήλες - αντιστοιχούν στις κορυφές του γραφήματος. στην γραμμή v , στήλη u υπάρχει η τιμή 1 αν η κατευθυνόμενη ακμή $(v, u) \in E$ και 0 διαφορετικά.

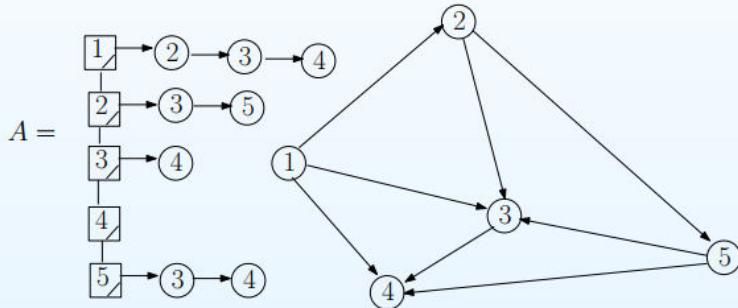


Σχήμα 1: Πίνακας Γειτνίασης

Παρατηρείστε ότι ο πίνακας γειτνίασης δεν είναι συμμετρικός ως προς την κύρια διαγώνιο.

Λίστα

Η απεικόνιση μέσω λίστας γειτνίασης επίσης δεν παρουσιάζει συμμετρία: στη λίστα κάθε κορυφής v εμφανίζονται (συνήθως σε λεξικογραφική σειρά) οι κορυφές $N^+(v)$.



Σχήμα 2: Λίστα Γειτνίασης

Διάσχιση

Διάσχιση

Το πρόβλημα της διάσχισης είναι το ίδιο όπως και στην περίπτωση των μη-κατευθυνόμενων γραφημάτων.

Πρόβλημα 1 Επίσκεψη των κορυφών του γραφήματος $G(V, E)$ με συστηματικό τρόπο.

Στην περίπτωση των κατευθυνόμενων γραφημάτων η επίσκεψη στις κορυφές γίνεται κατά τη φορά των ακμών. Δηλαδή, από μία κορυφή v επισκεπτόμαστε τις κορυφές του συνόλου $N^+(v)$ (και όχι τις κορυφές του συνόλου $N^-(v)$).

ΔκΒ

Ο αλγόριθμος λειτουργεί όπως και στην περίπτωση των μη-κατευθυνόμενων γραφημάτων. Η διαφορά βρίσκεται στην πληροφορία που καταγράφεται κατά τη διάρκεια της διάσχισης. Συγκεκριμένα, στην περίπτωση των μη-κατευθυνόμενων γραφημάτων καταγραφόταν η πρώτη φορά επίσκεψης και ο άμεσος πρόγονος μίας κορυφής στους πίνακες *preorder* και *parent*, αντίστοιχα. Στην περίπτωση των κατευθυνόμενων γραφημάτων ο πίνακας *preorder* διατηρείται ενώ ο πίνακας *parent* αντικαθίσταται από τον πίνακα *postorder*. Στον πίνακα αυτό καταγράφεται για κάθε κορυφή v η τελευταία φορά που συναντάται από τη διάσχιση. Έτσι ο χρόνος (αριθμός των βημάτων) που μία κορυφή v βρίσκεται στη στοίβα (της ΔκΒ) είναι

$$\text{postorder}[v] - \text{preorder}[v] + 1$$

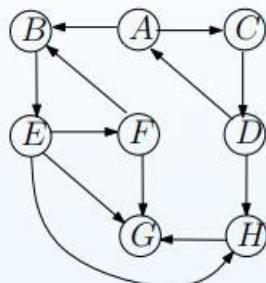
Αλγόριθμος ΔκΒ

```
explore(clock, v, visited, preorder, postorder)
    visited[v] ← true;
    preorder[v] ← ++clock;
    for u ∈ N+(v)
        if not visited[u] then
            explore(clock, u, visited, preorder, postorder);
            postorder[v] ← ++clock;

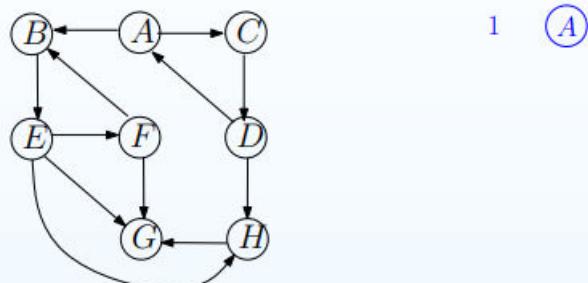
main()
    clock ← 0;
    for v ∈ V
        visited[v] ← false;
        preorder[v] ← 0; postorder[v] ← 0;
    for v ∈ V
        if not visited[v] then
            explore(clock, v, visited, preorder, postorder);
```

Παράδειγμα

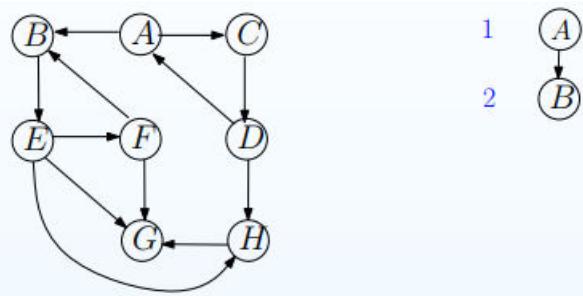
Να πραγματοποιηθεί ΔκΒ στο γράφημα του Σχήματος 3.



Σχήμα 3: Γράφημα - ΔκΒ

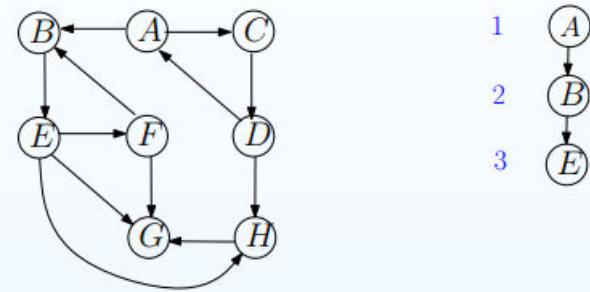


Σχήμα 4: Γράφημα - ΔκΒ



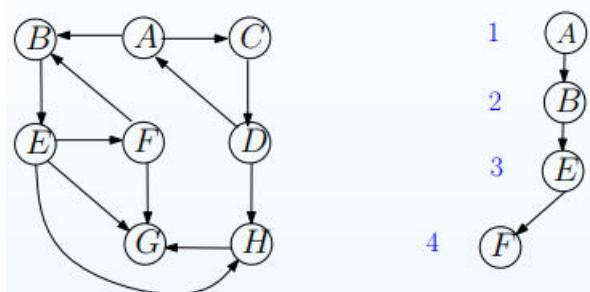
1 A
2 B

Σχήμα 4: Γράφημα - ΔκΒ



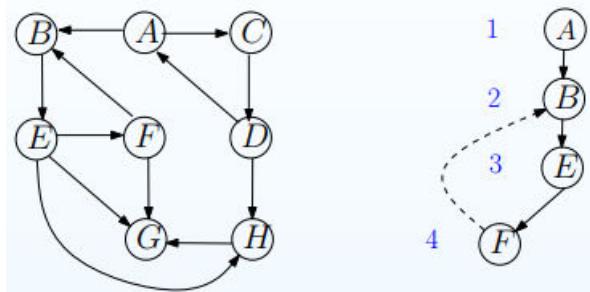
1 A
2 B
3 E

Σχήμα 4: Γράφημα - ΔκΒ



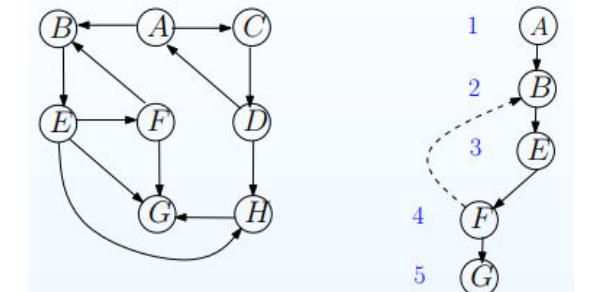
1 A
2 B
3 E
4 F

Σχήμα 4: Γράφημα - ΔκΒ



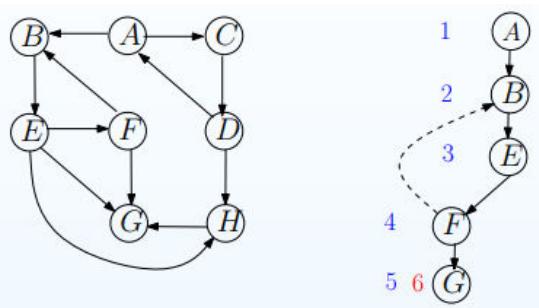
1 A
2 B
3 E
4 F

Σχήμα 4: Γράφημα - ΔκΒ

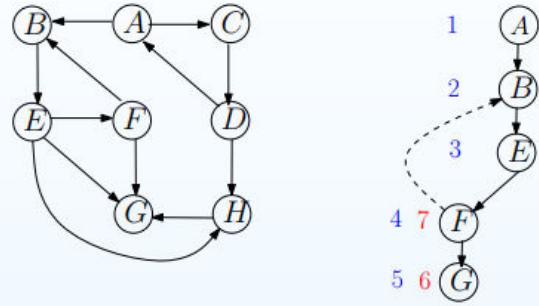


1 A
2 B
3 E
4 F
5 G

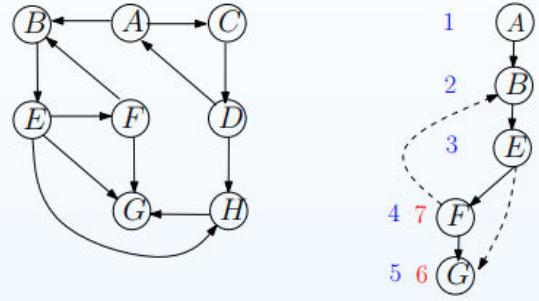
Σχήμα 4: Γράφημα - ΔκΒ



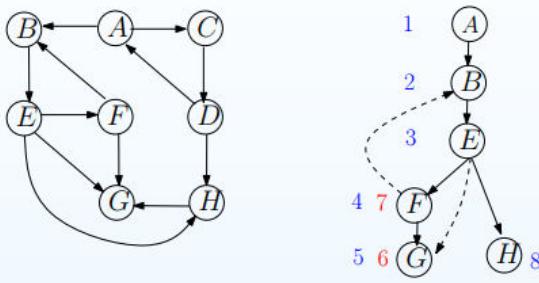
Σχήμα 4: Γράφημα - ΔκΒ



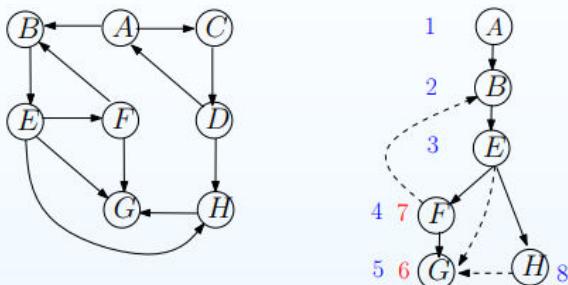
Σχήμα 4: Γράφημα - ΔκΒ



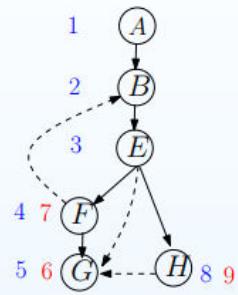
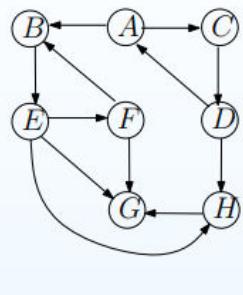
Σχήμα 4: Γράφημα - ΔκΒ



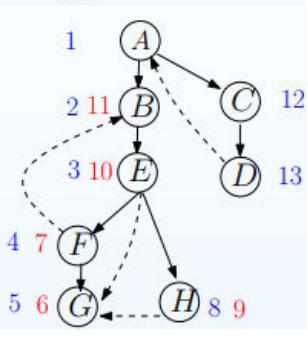
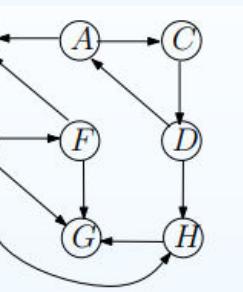
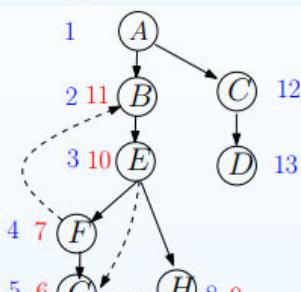
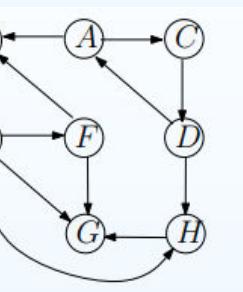
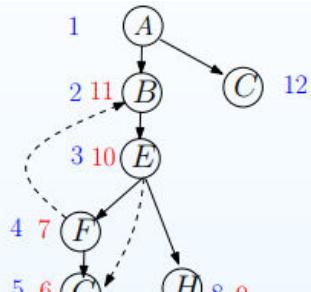
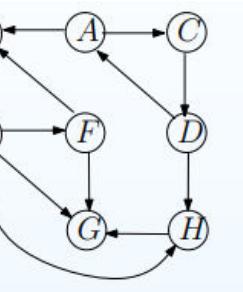
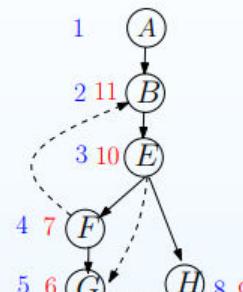
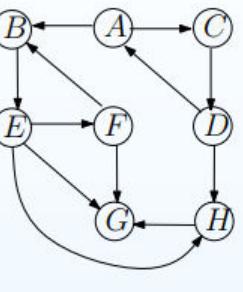
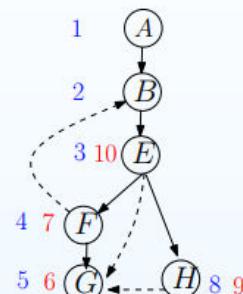
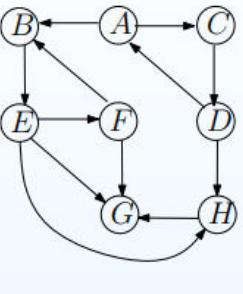
Σχήμα 4: Γράφημα - ΔκΒ

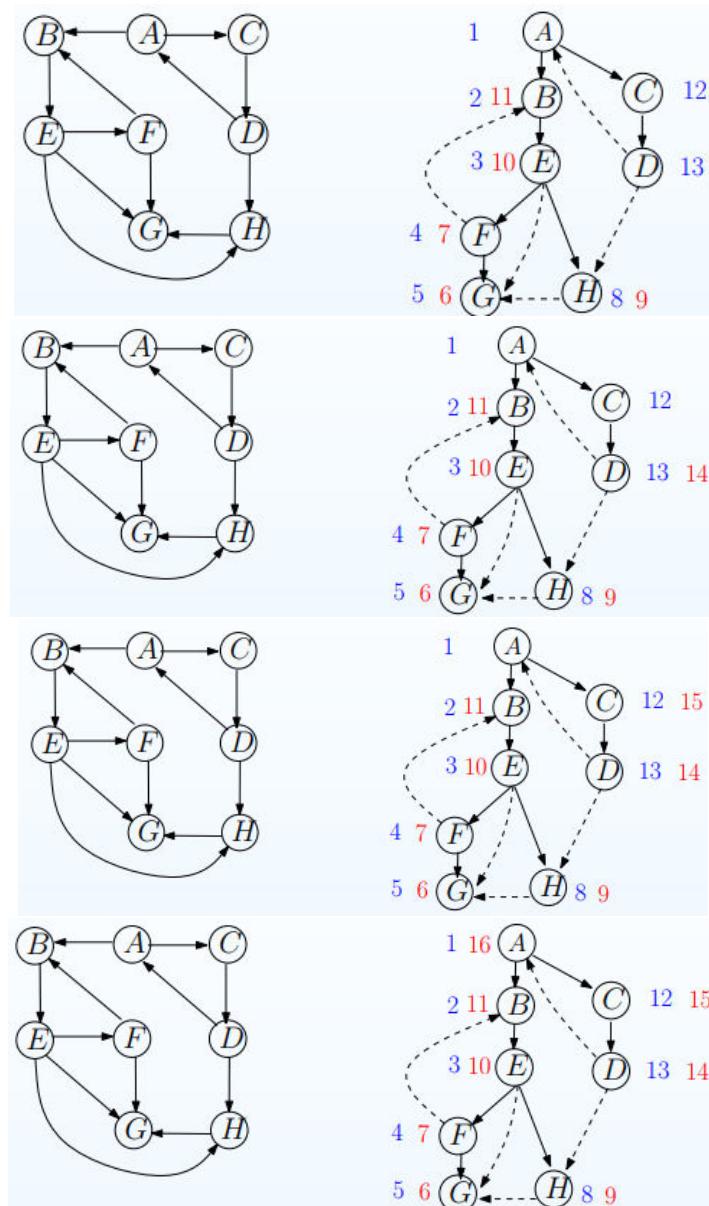


Σχήμα 4: Γράφημα - ΔκΒ

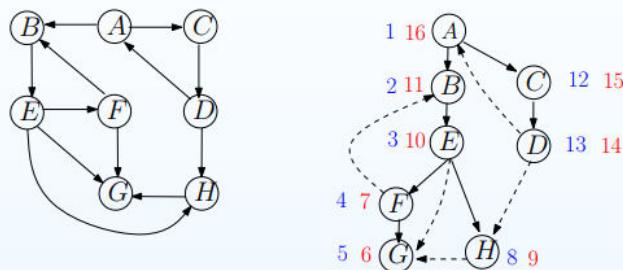


Σχήμα 4: Γράφημα - ΔΚΒ





ΔκΒ - Ακμές



Σχήμα 5: Γράφημα - ΔκΒ

Ακμή (v, u) είναι

- εμπροσθοακμή αν $pre[v] < pre[u] < post[u] < post[v]$,
- οπισθοακμή αν $pre[u] < pre[v] < post[v] < post[u]$,
- εγκάρσια αν $[pre[v], post[v]] \cap [pre[u], post[u]] = \emptyset$.

Οι δενδρικές ακμές είναι εμπροσθοακμές αλλά δεν τισχύει ότι κάθε εμπροσθοακμή είναι δενδρική ακμή (δες ακμή (E, G) στο Σχήμα 5)

Παρατηρήσεις

- Η ύπαρξη οπισθοακμής υποδηλώνει την ύπαρξη κατευθυνόμενου κύκλου
- Η ΔκΒ μπορεί να εντοπίσει αν υπάρχει κατευθυνόμενο μονοπάτι που να συνδέει δύο διθείσες κορυφές καθώς και κύκλο.
- Αν το γράφημα δεν έχει (κατευθυνόμενο) κύκλο τότε ονομάζεται άκυκλο κατευθυνόμενο γράφημα (ΑΚΓ).

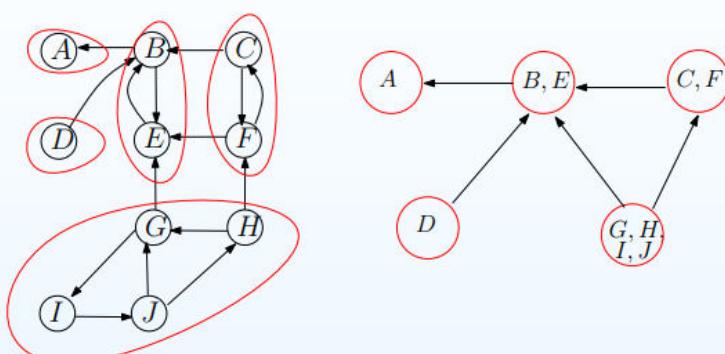
ΙΣΓΣ

Συνδεσιμότητα

Η έννοια της συνδεσιμότητας στα κατευθυνόμενα γραφήματα είναι διαφορετική από την αντίστοιχη έννοια στα μη-κατευθυνόμενα γραφήματα. Συγκεκριμένα, ένα κατευθυνόμενο γράφημα αποτελείται (μπορεί να αποσυνδεθεί σε) *Ισχυρά Συνδεδεμένες Γραφικές Συνιστώσες* (ΙΣΓΣ).

Ορισμός 2 *Mία ΙΣΓΣ είναι ένα μεγιστοτικό υπογράφημα ενός κατευθυνόμενου γραφήματος με την ιδιότητα ότι για κάθε δύο κορυφές v , u που ανήκουν σε αυτή ισχύει ότι υπάρχει κατευθυνόμενο μονοπάτι και από v στη u αλλά και αντίστροφα.*

Παράδειγμα



Σχήμα 6: ΙΣΓΣ με κόκκινο

Στο Σχήμα 6 απεικονίζεται ένα κατευθυνόμενο γράφημα και οι ΙΣΓΣ. Αν συρρικνώσουμε κάθε ΙΣΓΣ σε μία κορυφή τότε παράγεται ένα ΑΚΓ (δεξιό γράφημα του Σχήματος 6). Οι κορυφές στο ΑΚΓ που έχουν μόνο εξερχόμενες ακμές ονομάζονται *αφετηριακές ΙΣΓΣ* ενώ αυτές που έχουν μόνο εισερχόμενες ακμές *τερματικές ΙΣΓΣ*.

Ιδιότητες

Ιδιότητα 1 Αν εκτελέσουμε ΔκΒ σε μία ΙΣΓΣ εκκινώντας από μία κορυφή της ν θα μπορέσουμε να προσπελάσουμε όλους τις κορυφές της συνιστώσας αυτής.

Ιδιότητα 2 Αν εκτελέσουμε ΔκΒ σε όλο το γράφημα, η κορυφή με την μεγαλύτερη τιμή *postorder* βρίσκεται σε μία αφετηριακή ΙΣΓΣ

Ιδιότητα 3 Αν C και C' δύο ΙΣΓΣ και υπάρχει ακμή από κάποια κορυφή της πρώτης προς τη δεύτερη, τότε ο μεγαλύτερος αριθμός *postorder* στη C είναι μεγαλύτερος από τον μεγαλύτερο αριθμό *postorder* στη C' .

Η τελευταία ιδιότητα μας λέει ότι μπορούμε να κατατάξουμε τις ΙΣΓΣ σύμφωνα με το μεγαλύτερο αριθμό *postorder* που εμφανίζεται σε κάποια από τις κορυφές τους. Επίσης σύμφωνα με την Ιδιότητα 2 μπορούμε να ξεκινήσουμε από μία αφετηριακή ΙΣΓΣ. Θα εκτελέσουμε ΔκΒ σε αυτή και αφού προσπελάσουμε όλες τις κορυφές της (σύμφωνα με την Ιδιότητα 1) μπορούμε να “διαγράψουμε” τις κορυφές της και να επαναλάβουμε τη διαδικασία αυτή.

Πρόβλημα: Πως όμως θα εντοπίσουμε μία αφετηριακή ΙΣΓΣ;

Ιδέα: Θα χρησιμοποιήσουμε το γράφημα G^R . Το γράφημα αυτό παράγεται από το G αν αντιστρέψουμε τη φορά των ακμών. Παρατηρήστε ότι το G^R έχει τις ίδιες ΙΣΓΣ όπως και το G μόνο που οι αφετηριακές και τερματικές ΙΣΓΣ είναι σε αντίστροφους ρόλους.

Αλγόριθμος

1. Εκτελούμε ΔκΒ στο G υπολογίζοντας *postorder*.
2. Παράγουμε τον γράφημα G^R
3. Επιλέγουμε την κορυφή v με το μεγαλύτερο αριθμό *postorder* και εκτελούμε ΔκΒ στο G^R εκκινώντας από αυτή. Οι κορυφές που ανακαλύπτονται από τη ΔκΒ ανήκουν στην ίδια ΙΣΓΣ.
4. Διαγράφουμε την ΙΣΓΣ.
5. Αν υπάρχουν κορυφές που δεν τις έχει επισκεφθεί η ΔκΒ επιστρέφουμε στο Βήμα 3.

Ο Αλγόριθμος που παρουσιάζουμε στη συνέχεια προϋποθέτει ότι έχει πάρει τιμές ο πίνακας *postorder*. Δηλαδή έχει προηγηθεί στο αρχικό γράφημα η ΔκΒ.

Ο Αλγόριθμος χρησιμοποιεί το σύνολο Q στον οποίο αποθηκεύει τις ΙΣΓΣ.

scc(*postorder*)

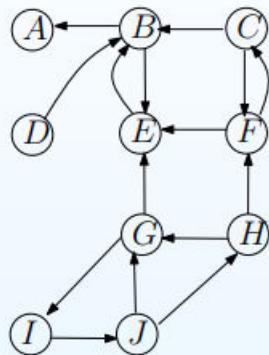
```
for  $v \in V$   $visited[v] \leftarrow false$ ;
while  $\exists u \in V$  such that  $visited[u] = false$ 
     $v \leftarrow \text{argmax}\{\text{postorder}[u] : u \in V, visited[u] = false\}$ ;
     $Q \leftarrow \emptyset$ ;
    explore( $v, visited, Q$ );
    print  $Q$ ;
```

όπου

```
explore( $v, visited, Q$ );
 $visited[v] \leftarrow true$ ;
 $Q \leftarrow Q \cup \{v\}$ ;
for  $u \in N^-(v)$ 
    if not  $visited[u]$  then
        explore( $v, visited, Q$ );
```

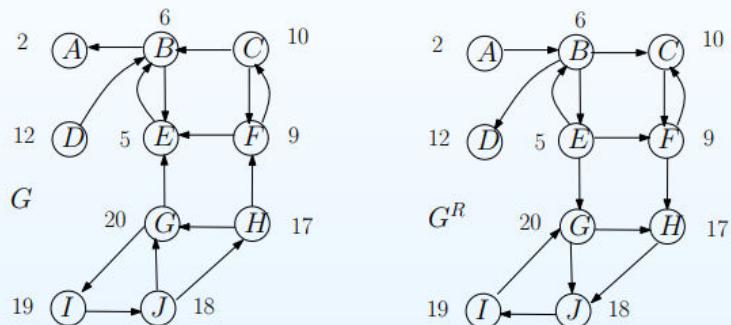
Παράδειγμα

Να βρεθούν οι ΙΣΓΣ στο γράφημα του Σχήματος 7



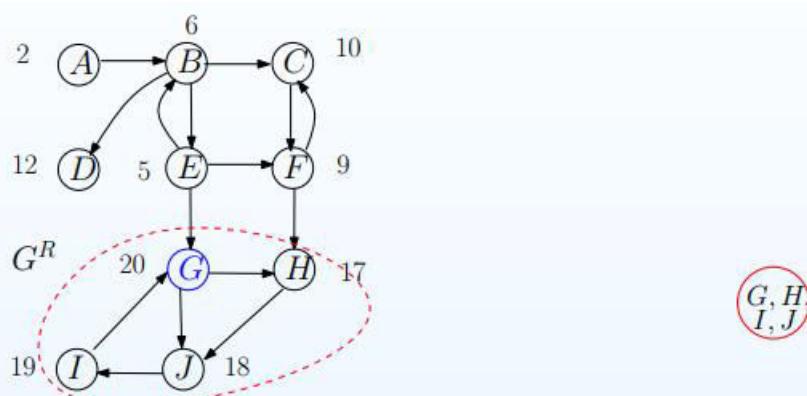
Σχήμα 7: Κατευθυνόμενο Γράφημα

Εκτελούμε ΔκΒ προκειμένου να υπολογιστούν τα στοιχεία του πίνακα *postorder*. Οι τιμές του πίνακα σημειώνονται δίπλα σε κάθε κορυφή στο Σχήμα 8.

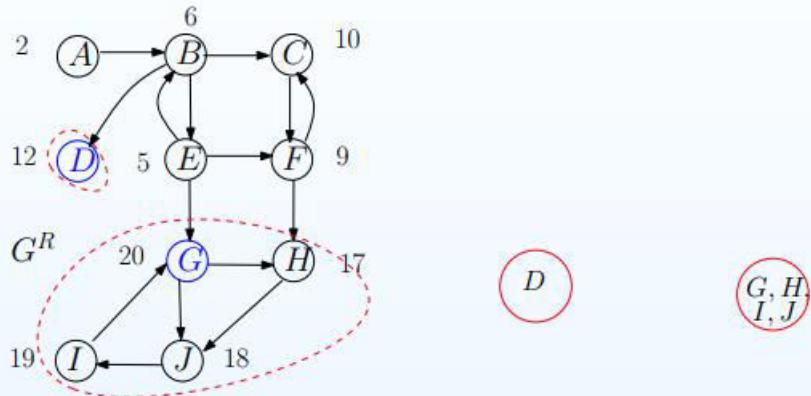


Σχήμα 8: Γραφήματα G και G^R

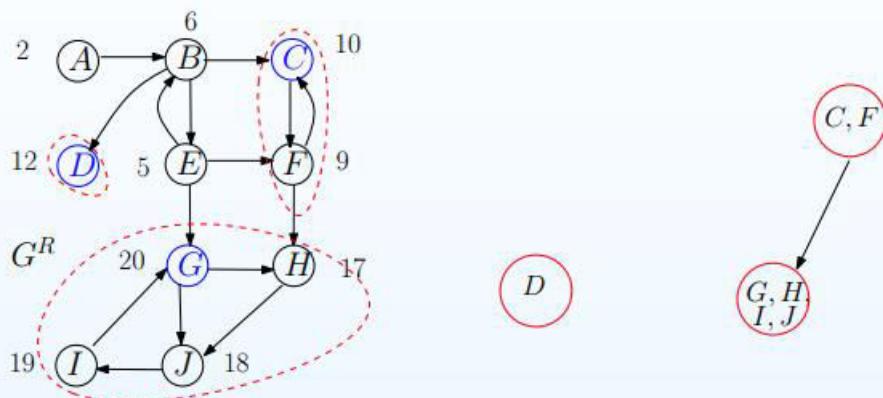
Στη συνέχεια κατασκευάζουμε το γράφημα G^R με αντιστροφή της φοράς των ακμών (Σχήμα 8). Ο Αλγόριθμος συνεχίζει με την ανάδειξη των ΙΣΓΣ ξεκινώντας από την κορυφή σε κάθε συνιστώσα με το μεγαλύτερο αριθμό *postorder*.



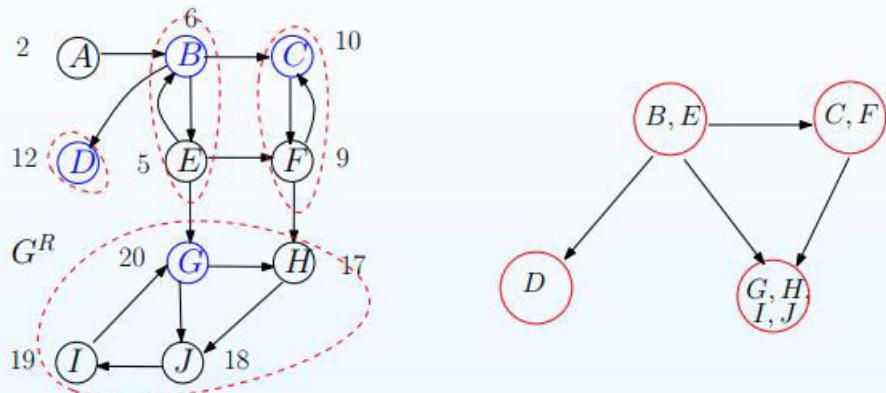
Σχήμα 9: Υπολογισμός ΙΣΓΣ - κορυφή με μεγαλύτερο συντελεστή *postorder* σε μπλε.



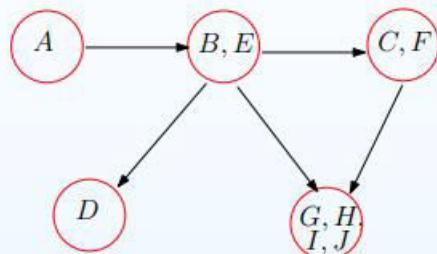
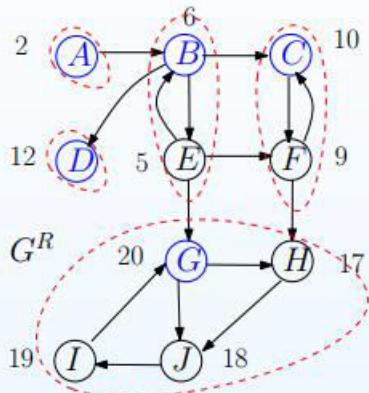
Σχήμα 9: Υπολογισμός ΙΣΓΣ - κορυφή με μεγαλύτερο συντελεστή *postorder* σε μπλε.



Σχήμα 9: Υπολογισμός ΙΣΓΣ - κορυφή με μεγαλύτερο συντελεστή *postorder* σε μπλε.



Σχήμα 9: Υπολογισμός ΙΣΓΣ - κορυφή με μεγαλύτερο συντελεστή *postorder* σε μπλε.



Σχήμα 9: Υπολογισμός ΙΣΓΣ - κορυφή με μεγαλύτερο συντελεστή *postorder* σε μπλε.

04_ΣΥΝΤΟΜΟΤΕΡΑ ΜΟΝΟΠΑΤΙΑ / 05_shpaths

Ορισμοί

Διαδρομή και Μονοπάτια

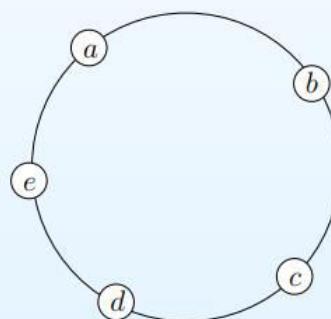
Έστω ένα μη-κατευθυνόμενο γράφημα $G(V, E)$.

Διαδρομή: Μία ακολουθία κορυφών $W = \langle v_0, v_1, \dots, v_k \rangle$ με $\{v_i, v_{i+1}\} \in E(G)$, $i = 0, \dots, k - 1$.

Μονοκονδυλιά: Διαδρομή χωρίς επαναλαμβανόμενη ακμή

Μονοπάτι: Διαδρομή χωρίς επαναλαμβανόμενη κορυφή

Κύκλος: Μονοπάτι όπου επαναλαμβάνεται μόνο η τερματική κορυφή.



Σχήμα 1: Κύκλος C_5

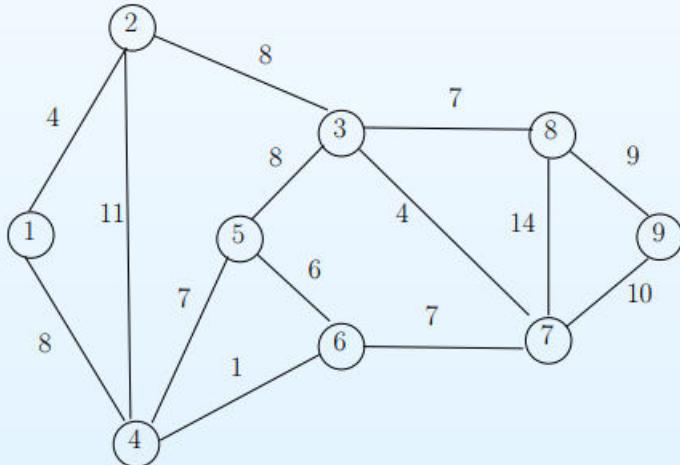
Ένα γράφημα που δεν περιέχει κύκλο ονομάζεται **άκυκλο**

Εμβαρές Γράφημα

Ένα γράφημα $G(V, E)$ ονομάζεται εμβαρές ή βεβαρημένο αν υπάρχει συνάρτηση

$$w : E \rightarrow \mathbb{R}.$$

Οι συντελεστές ω των ακμών αναφέρονται και σαν μήκη (αποστάσεις) των ακμών.



Σχήμα 2: Εμβαρές Γράφημα

Συντομότερο Μονοπάτι

- **Πρόβλημα:** Δεδομένου ενός εμβαρούς γραφήματος $G(V, E, w)$ και δύο κορυφών του $s, t \in V$, θέλουμε να βρούμε το συντομότερο μονοπάτι που συνδέει τις δύο κορυφές.
- **Συντομότερο:** το άθροισμα των συντελεστών των ακμών που συμμετέχουν σε αυτό να ελαχιστοποιείται.

Μη Αρνητικότητα

Αν θεωρήσουμε ότι οι συντελεστές των ακμών είναι μη-αρνητικοί αριθμοί ($w(e) \geq 0, e \in E$), το συντομότερο μονοπάτι ανάμεσα σε κάθε ζευγάρι κορυφών συμπίπτει με τη συντομότερη μονοκονδυλιά και τελικά με τη συντομότερη διαδρομή ανάμεσα σε αυτές τις κορυφές. Αυτό είναι προφανές αφού η επανάληψη μίας κορυφής (ή/και ακμής) σε μία σειρά κορυφών υποδηλώνει την ύπαρξη κύκλου και (λόγω της μη-αρνητικότητας των συντελεστών w) όλοι οι κύκλοι έχουν μη-αρνητικό μήκος και επομένως η διάσχιση τους δεν μειώνει την απόσταση ανάμεσα στις δύο κορυφές.

Βέλτιστη Υποδομή

Έστω ότι το συντομότερο μονοπάτι από την κορυφή u_0 στην κορυφή u_k είναι

$$P_{u_0, u_k} = u_0 - u_1 - u_2 - \cdots - u_k.$$

Τότε το συντομότερο μονοπάτι P_{u_i, u_j} , με $i < j$ και $i, j \in \{0, \dots, k\}$, εμπεριέχεται στο P_{u_0, u_k} .

Μοναδική Πηγή

Πρόβλημα

Θα ασχοληθούμε με την εύρεση συντομότερων μονοπατιών από μία δεδομένη κορυφή s προς κάθε άλλη κορυφή ενός γραφήματος.

ΜΠΣΔ Δεδομένου ενός εμβαρούς γράφημα $G(V, E, w)$, $w \geq 0$ και μίας κορυφής του $s \in V$, να βρεθούν τα ελάχιστα μονοπάτια από το s προς κάθε άλλη κορυφή του γραφήματος

Στην περίπτωση που το γράφημα δεν είναι εμβαρές (ισοδύναμο με την περίπτωση όπου $w(e) = 1$, για κάθε $e \in E$), το πρόβλημα επιλύεται με τη διάσχιση κατά πλάτος (ΔκΠ) εκκινώντας από την κορυφή s .

Ο αλγόριθμος του Dijkstra επιλύει το πρόβλημα ΜΠΣΔ στη γενική περίπτωση.

Ιδέα

Έστω d_v η μεταβλητή που θα περιέχει το μήκος του συντομότερου μονοπατιού που συνδέει την αφετηρία (κορυφή s) με την κορυφή v στο τέλος της εκτέλεσης του Αλγόριθμου. Ο Αλγόριθμος, κατά τη διάρκεια της εκτέλεσης, επανυπολογίζει την τιμή της μεταβλητής d_v , για κάθε κορυφή v , μέχρι η τιμή της να είναι ίση με το μήκος που αναφέραμε παραπάνω.

Βασική Ιδέα

Σε κάθε στιγμή, ο Αλγόριθμος διαχωρίζει τις κορυφές σε δύο ομάδες: στην ομάδα των μόνιμων κορυφών (σύνολο S) και στην ομάδα των μη-μόνιμων κορυφών. Στην πρώτη ομάδα ανήκουν οι κορυφές των οποίων η τιμή της μεταβλητής d δεν θα αλλάξει μέχρι το τέλος του αλγόριθμου. Δηλαδή, για κάθε μόνιμη κορυφή έχει υπολογιστεί το συντομότερο μονοπάτι από την αφετηρία. Στη δεύτερη ομάδα ανήκουν οι υπόλοιπες κορυφές.

Λειτουργία

Ο Αλγόριθμος λειτουργεί επαναληπτικά. Η κάθε επανάληψη χωρίζεται σε δύο φάσεις.

Φάση I

Από τις μη-μόνιμες κορυφές επιλέγεται αυτή που έχει το μικρότερο d και γίνεται μόνιμη. Έστω ότι αυτή είναι η κορυφή v^* . Δηλαδή,

$$v^* = \operatorname{argmin}\{d_v : v \in V \setminus S\}. \quad (1)$$

Στη συνέχεια η κορυφή v^* προστίθεται στο σύνολο S .

Φάση II

Για κάθε ΓΕΙΤΟΝΙΚΗ ΚΟΡΥΦΗ u της v^* η οποία ΑΝΗΚΕΙ ΣΤΟ ΣΥΝΟΛΟ $V \setminus S$ επανυπολογίζεται η τιμή της μεταβλητής d_u από τον τύπο

$$d_u = \min\{d_u, d_{v^*} + w(v^*, u)\}, \quad (2)$$

όπου $w(v^*, u)$ είναι το βάρος (μήκος) της ακμής (v^*, u) .

Η (2) υποδηλώνει ότι η τιμή d_u θα αλλάξει αν το μονοπάτι που συνδέει την s με τη u και ΠΕΡΝΑΕΙ ΑΠΟ ΤΗΝ κορυφή v^* είναι συντομότερο από το μονοπάτι που μέχρι πρότινος συνέδεε την s με τη u (προηγούμενη τιμή της d_u .)

Ο Αλγόριθμος εκτελεί $|V| - 1$ επαναλήψεις. Στην Φάση I της πρώτης επανάληψης γίνεται μόνιμη η αφετηρία (κορυφή s). Αυτό είναι το αποτέλεσμα της αρχικοποίησης του Αλγόριθμου κατά την οποία τίθεται

$$d_s \leftarrow 0, \quad d_v \leftarrow \infty, \forall v \in V \setminus \{s\}, \quad S \leftarrow \emptyset.$$

Αλγόριθμος

Require: $w : E \rightarrow \mathbb{R}_+$

1: void **Dijkstra**($G(V, E, w), s$)

2: **for all** $v \in V$ **do**

3: $d_v \leftarrow \infty; p_v \leftarrow \text{NULL};$

4: **end for**

5: $S \leftarrow \emptyset; d_s \leftarrow 0;$

6: **for** $i \leftarrow 1; i \leq |V| - 1; i++$ **do**

7: $v^* \leftarrow \operatorname{argmin}\{d_v : v \in V \setminus S\};$

8: $S \leftarrow S \cup \{v^*\};$

9: **for all** $u \in V \setminus S$ **and** $u \in N(v^*)$ **do**

10: **if** $d_u > d_{v^*} + w(v^*, u)$ **then**

11: $d_u \leftarrow d_{v^*} + w(v^*, u);$

12: $p_u \leftarrow v^*;$

13: **end if**

14: **end for**

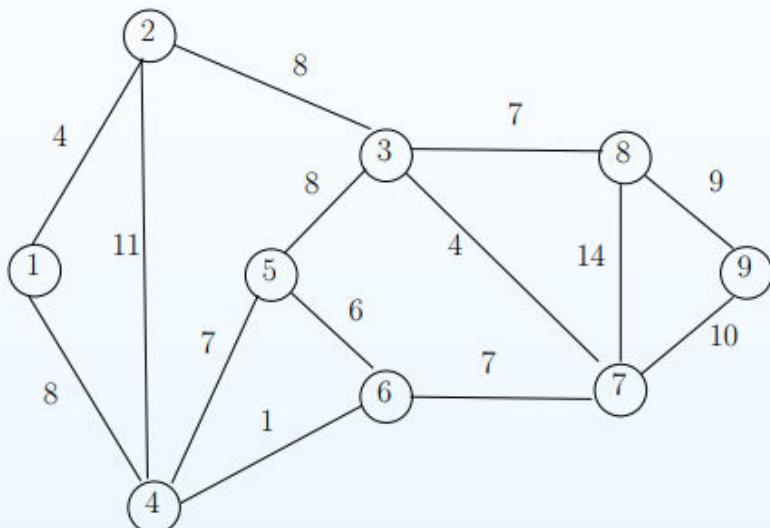
15: **end for**

Παρατηρήσεις

- Οι μεταβλητές p_v αποθηκεύουν την προτελευταία κορυφή στο συντομότερο μονοπάτι από την κορυφή s στην v . Με αυτό τον τρόπο μπορούμε να ανακτήσουμε όλο το μονοπάτι από την s στην v (όχι μόνο την απόσταση του που δίνεται από την τιμή της d_v).
- Οι μεταβλητές d_v και p_v μπορούν να υλοποιηθούν σαν μονοδιάστατοι πίνακες με πλήθος στοιχείων μεγαλύτερο-ίσο με τον πληθάριθμο του αριθμού των κορυφών.

Παράδειγμα

Να βρεθούν τα συντομότερα μονοπάτια από την κορυφή 1 προς τις υπόλοιπες κορυφές στο γράφημα του Σχήματος 2.



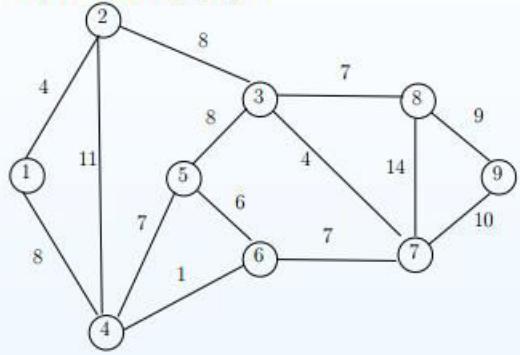
Αρχικοποίηση

$$d = [0, \infty, \infty, \infty, \infty, \infty, \infty, \infty, \infty]$$

$$p = [\quad, \quad, \quad, \quad, \quad, \quad, \quad, \quad, \quad]$$

$$S = \emptyset$$

Επανάληψη 1



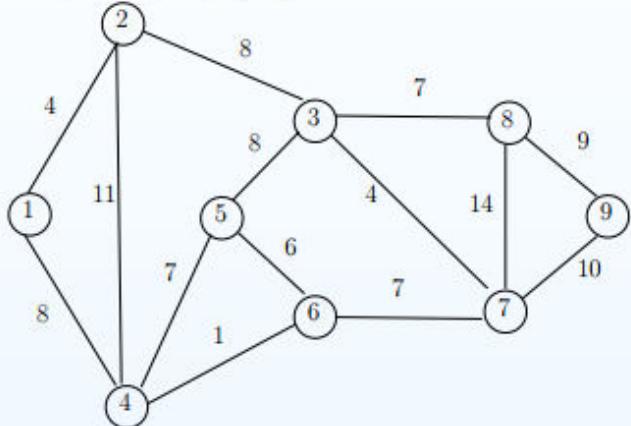
ΦΑΣΗ I

$$\begin{aligned}v^* &\leftarrow \operatorname{argmin}\{d_v : v \in \{1, \dots, 9\}\} \\&= \operatorname{argmin}\{0, \infty, \infty, \infty, \infty, \infty, \infty, \infty, \infty\} = 1, \\S &\leftarrow S \cup \{1\} = \emptyset \cup \{1\} = \{1\}\end{aligned}$$

ΦΑΣΗ II

$$\begin{aligned}d_2 &\leftarrow \min\{d_2, d_1 + w(1, 2)\} = \min\{\infty, 0 + 4\} = 4, \quad p_2 \leftarrow 1, \\d_4 &\leftarrow \min\{d_4, d_1 + w(1, 4)\} = \min\{\infty, 0 + 8\} = 8, \quad p_4 \leftarrow 1.\end{aligned}$$

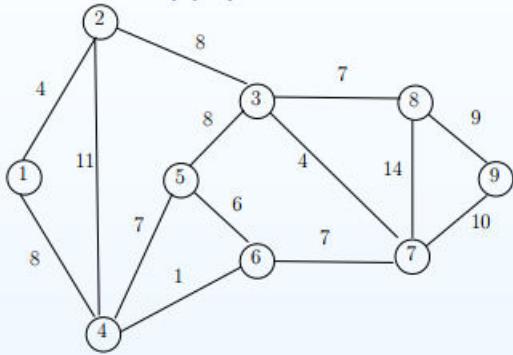
Επανάληψη 1



Έξοδος Επανάληψης

$$\begin{aligned}d &= [\textcolor{red}{0}, 4, \infty, 8, \infty, \infty, \infty, \infty, \infty] \\p &= [\quad, 1, \quad, 1, \quad, \quad, \quad, \quad, \quad] \\S &= \{1\}\end{aligned}$$

Επανάληψη 2



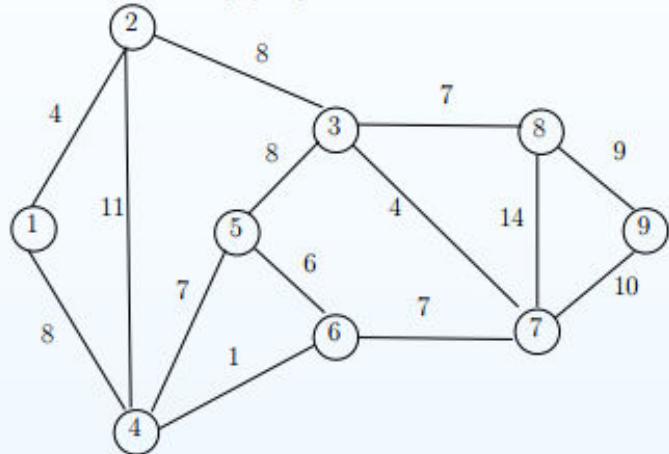
ΦΑΣΗ I

$$\begin{aligned}v^* &\leftarrow \operatorname{argmin}\{d_v : v \in \{2, \dots, 9\}\} \\&= \operatorname{argmin}\{4, \infty, 8, \infty, \infty, \infty, \infty\} = 2, \\S &\leftarrow S \cup \{1\} = \{1\} \cup \{2\} = \{1, 2\}\end{aligned}$$

ΦΑΣΗ II

$$\begin{aligned}d_3 &\leftarrow \min\{d_3, d_2 + w(2, 3)\} = \min\{\infty, 4 + 8\} = 12, \quad p_3 \leftarrow 2, \\d_4 &\leftarrow \min\{d_4, d_2 + w(2, 4)\} = \min\{8, 4 + 11\} = 8.\end{aligned}$$

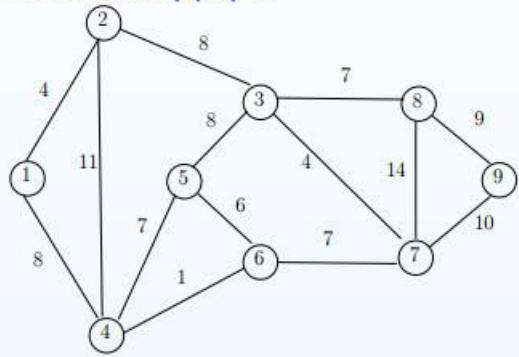
Επανάληψη 2



Έξοδος Επανάληψης

$$\begin{aligned}d &= [0, 4, 12, 8, \infty, \infty, \infty, \infty, \infty] \\p &= [, 1, 2, 1, , , , ,] \\S &= \{1, 2\}\end{aligned}$$

Επανάληψη 3



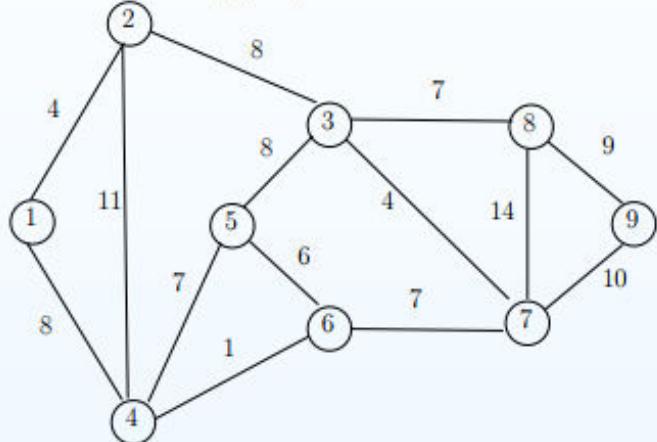
ΦΑΣΗ I

$$\begin{aligned}v^* &\leftarrow \operatorname{argmin}\{d_v : v \in \{3, \dots, 9\}\} \\&= \operatorname{argmin}\{12, 8, \infty, \infty, \infty, \infty\} = 4, \\S &\leftarrow S \cup \{4\} = \{1, 2\} \cup \{4\} = \{1, 2, 4\}\end{aligned}$$

ΦΑΣΗ II

$$\begin{aligned}d_5 &\leftarrow \min\{d_5, d_4 + w(4, 5)\} = \min\{\infty, 8 + 7\} = 15, \quad p_5 \leftarrow 4, \\d_6 &\leftarrow \min\{d_6, d_4 + w(4, 6)\} = \min\{\infty, 8 + 1\} = 9, \quad p_6 \leftarrow 4.\end{aligned}$$

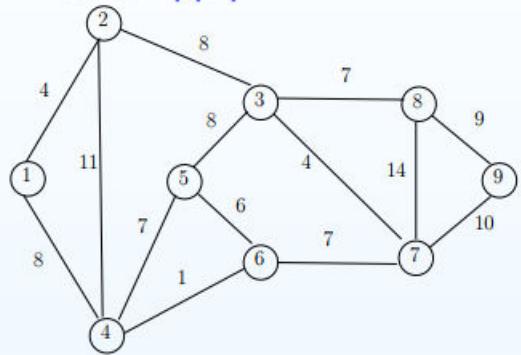
Επανάληψη 3



Έξοδος Επανάληψης

$$\begin{aligned}d &= [\textcolor{red}{0}, \textcolor{red}{4}, 12, \textcolor{red}{8}, 15, 9, \infty, \infty, \infty] \\p &= [\quad, 1, 2, 1, 4, 4, \quad, \quad, \quad] \\S &= \{1, 2, 4\}\end{aligned}$$

Επανάληψη 4



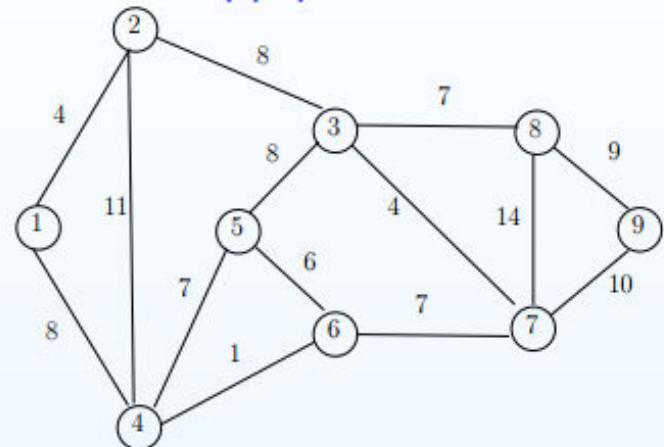
ΦΑΣΗ I

$$\begin{aligned}v^* &\leftarrow \operatorname{argmin}\{d_v : v \in \{5, \dots, 9\} \cup \{3\}\} \\&= \operatorname{argmin}\{15, 9, \infty, \infty, \infty, 12\} = 6, \\S &\leftarrow S \cup \{6\} = \{1, 2, 4\} \cup \{6\} = \{1, 2, 4, 6\}\end{aligned}$$

ΦΑΣΗ II

$$\begin{aligned}d_5 &\leftarrow \min\{d_5, d_6 + w(6, 5)\} = \min\{15, 9 + 6\} = 15, \\d_7 &\leftarrow \min\{d_7, d_6 + w(6, 7)\} = \min\{\infty, 9 + 7\} = 16, \quad p_7 \leftarrow 6.\end{aligned}$$

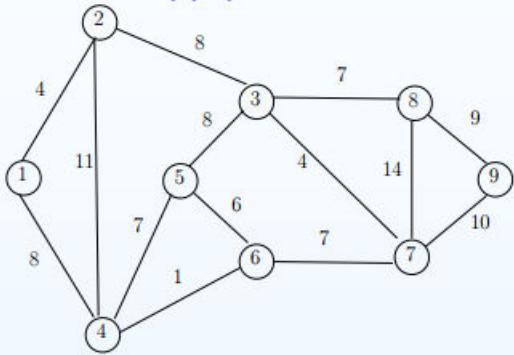
Επανάληψη 4



Έξοδος Επανάληψης

$$\begin{aligned}d &= [0, 4, 12, 8, 15, 9, 16, \infty, \infty] \\p &= [, 1, 2, 1, 4, 4, 6, ,] \\S &= \{1, 2, 4, 6\}\end{aligned}$$

Επανάληψη 5



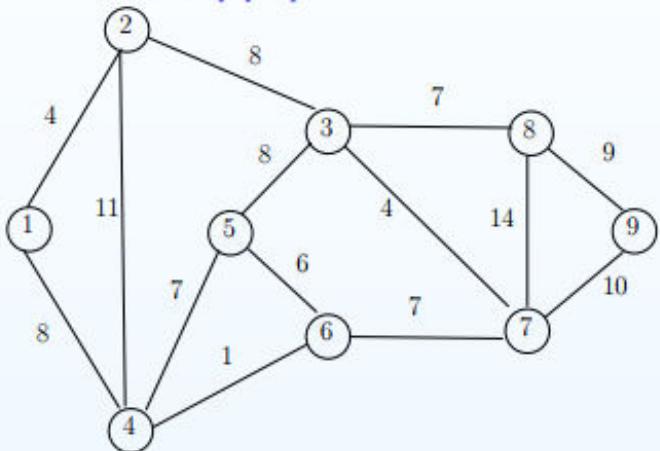
ΦΑΣΗ I

$$\begin{aligned}
 v^* &\leftarrow \operatorname{argmin}\{d_v : v \in \{3, 5, 7, 8, 9\}\} \\
 &= \operatorname{argmin}\{12, 15, 16, \infty, \infty\} = 3, \\
 S &\leftarrow S \cup \{3\} = \{1, 2, 4, 6\} \cup \{3\} = \{1, 2, 3, 4, 6\}
 \end{aligned}$$

ΦΑΣΗ II

$$\begin{aligned}
 d_5 &\leftarrow \min\{d_5, d_3 + w(3, 5)\} = \min\{15, 12 + 8\} = 15, \\
 d_7 &\leftarrow \min\{d_7, d_3 + w(7, 3)\} = \min\{16, 12 + 4\} = 16, \\
 \text{iwa.gr } d_8 &\leftarrow \min\{d_8, d_3 + w(3, 8)\} = \min\{\infty, 12 + 7\} = 19 \quad p_8 \xleftarrow{14} 3.
 \end{aligned}$$

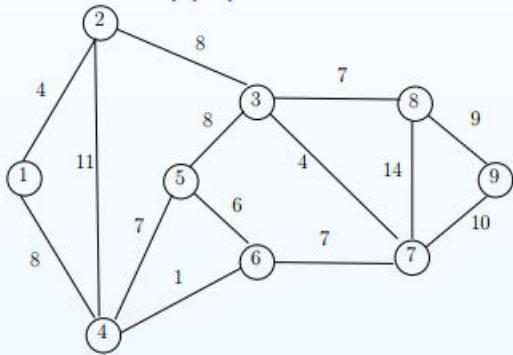
Επανάληψη 5



Έξοδος Επανάληψης

$$\begin{aligned}
 d &= [0, 4, 12, 8, 15, 9, 16, 19, \infty] \\
 p &= [, 1, 2, 1, 4, 4, 6, 3,] \\
 S &= \{1, 2, 4, 6\}
 \end{aligned}$$

Επανάληψη 6

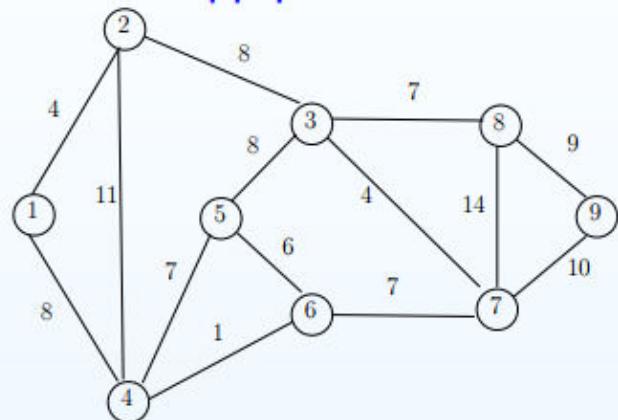


ΦΑΣΗ Ι

$$\begin{aligned}v^* &\leftarrow \operatorname{argmin}\{d_v : v \in \{5, 7, 8, 9\}\} \\&= \operatorname{argmin}\{15, 16, 19, \infty\} = 5, \\S &\leftarrow S \cup \{5\} = \{1, 2, 3, 4, 6\} \cup \{5\} = \{1, 2, 3, 4, 5, 6\}\end{aligned}$$

ΦΑΣΗ ΙΙ

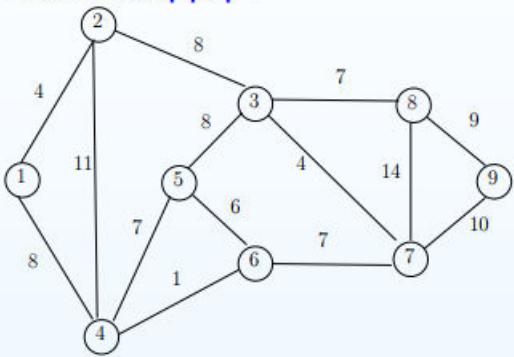
Επανάληψη 6



Έξοδος Επανάληψης

$$\begin{aligned}d &= [0, 4, 12, 8, 15, 9, 16, 19, \infty] \\p &= [, 1, 2, 1, 4, 4, 6, 3,] \\S &= \{1, 2, 3, 4, 5, 6\}\end{aligned}$$

Επανάληψη 7



ΦΑΣΗ Ι

$$v^* \leftarrow \operatorname{argmin}\{d_v : v \in \{7, 8, 9\}\}$$

$$= \operatorname{argmin}\{16, 19, \infty\} = 7,$$

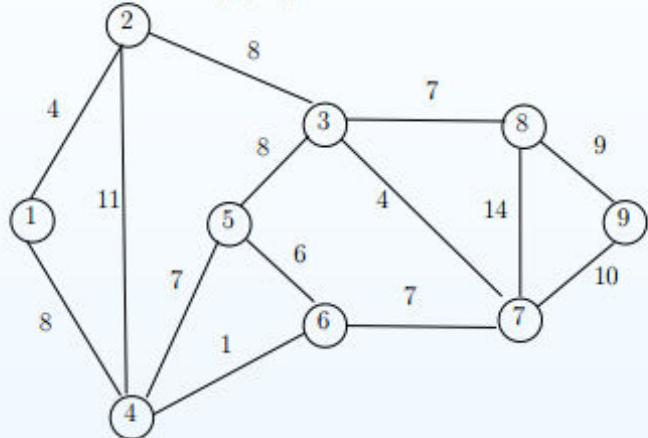
$$S \leftarrow S \cup \{7\} = \{1, 2, 3, 4, 5, 6\} \cup \{7\} = \{1, 2, 3, 4, 5, 6, 7\}$$

ΦΑΣΗ II

$$d_8 \leftarrow \min\{d_8, d_7 + w(7, 8)\} = \min\{19, 16 + 14\} = 19$$

$$d_9 \leftarrow \min\{d_9, d_7 + w(7, 9)\} = \min\{\infty, 16 + 10\} = 26 \quad p_9 \leftarrow 7.$$

Επανάληψη 7



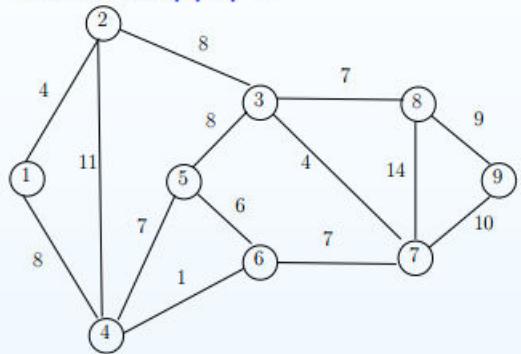
Έξοδος Επανάληψης

$$d = [0, 4, 12, 8, 15, 9, 16, 19, 26]$$

$$p = [, 1, 2, 1, 4, 4, 6, 3, 7]$$

$$S = \{1, 2, 3, 4, 5, 6, 7\}$$

Επανάληψη 8



ΦΑΣΗ I

$$v^* \leftarrow \operatorname{argmin}\{d_v : v \in \{8, 9\}\}$$

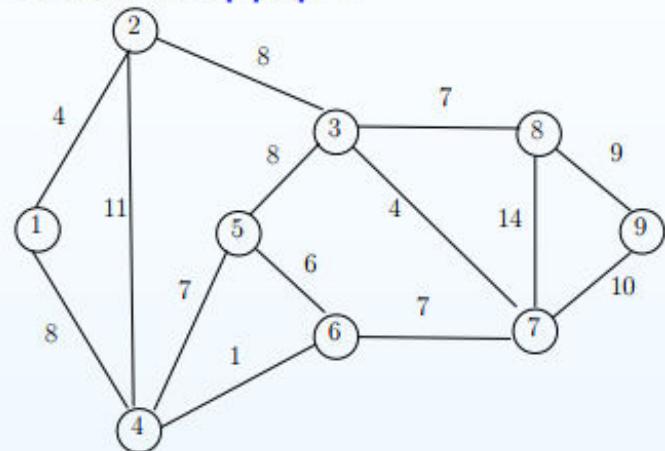
$$= \operatorname{argmin}\{19, 26\} = 8,$$

$$S \leftarrow S \cup \{8\} = \{1, 2, 3, 4, 5, 6, 7\} \cup \{8\} = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

ΦΑΣΗ II

$$d_9 \leftarrow \min\{d_9, d_8 + w(8, 9)\} = \min\{26, 19 + 9\} = 26.$$

Επανάληψη 8



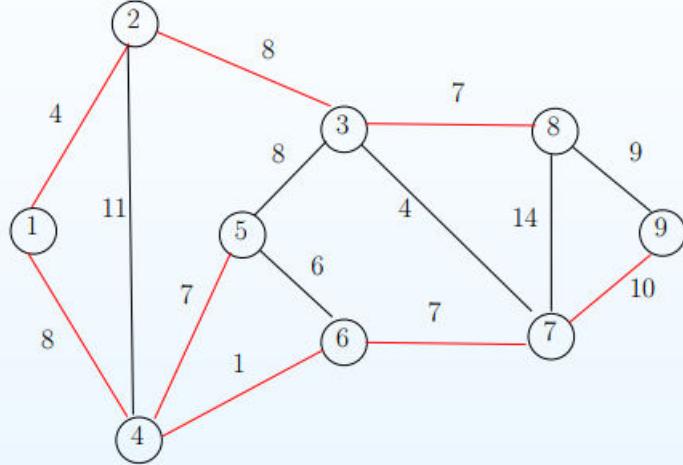
Έξοδος Επανάληψης

$$d = [0, 4, 12, 8, 15, 9, 16, 19, 26]$$

$$p = [, 1, 2, 1, 4, 4, 6, 3, 7]$$

$$S = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

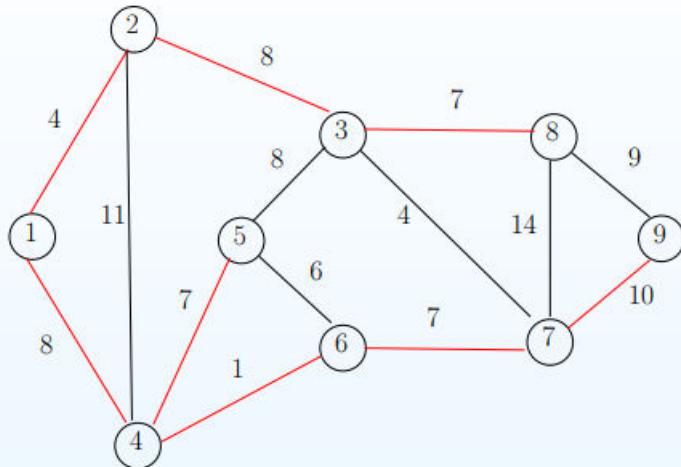
Λύση



$$d = [0, 4, 12, 8, 15, 9, 16, 19, 26]$$

$$p = [\quad, 1, 2, 1, 4, 4, 6, 3, 7]$$

Παρατηρήσεις



Ο Αλγόριθμος του Dijkstra υπολογίζει το συντομότερο μονοπάτι από την αφετηρία προς κάθε άλλο κόμβο.

Επομένως, παράγονται $n - 1$ μονοπάτια τα οποία συνιστούν ένα δένδρο συντομότερων μονοπατιών (ΔΣΜ) (λόγω της ιδιότητας της βέλτιστης υποδομής). Η πληροφορία αυτή βρίσκεται αποθηκευμένη στον πίνακα p . Στο συγκεκριμένο παράδειγμα έχουμε

$$p = [\quad, 1, 2, 1, 4, 4, 6, 3, 7].$$

Ορθότητα

Πρόταση 1 Ο Αλγόριθμος του Dijkstra υπολογίζει σωστά το ελάχιστο μονοπάτι από την αφετηρία (κορυφή s) προς κάθε άλλη κορυφή του γραφήματος

Απόδειξη Θα συμβολίσουμε με $d(s, v)$ το μήκος του συντομότερου μονοπατιού από την κορυφή s ως την κορυφή v , για κάθε $v \in V$. Για να δείξουμε το ζητούμενο, θα χρησιμοποιήσουμε επαγωγή.

Βάση Επαγωγής: $|S| = 0$, $d_s = 0 = d(s, s)$, $S \leftarrow \{s\}$.

Επαγωγικό Βήμα: Στην αρχή μίας τυχαίας επανάληψης, θα υποθέσουμε ότι για κάθε κορυφή $v \in S$ ισχύει ότι $d(s, v) = d_v$. Θα δείξουμε ότι $d(s, u) = d_u$, όπου u η κορυφή που επιλέγει ο Αλγόριθμος να κάνει μόνιμη στην παρούσα επανάληψη, ήτοι

$$u = v^* = \operatorname{argmin}\{d_v : v \in V \setminus S\}. \quad (3)$$

Έστω ότι, αντίθετα με το ζητούμενο,

$$d(s, u) < d_u. \quad (4)$$

Επομένως, υπάρχει συντομότερο μονοπάτι P_{su} από την κορυφή s στη u από αυτό που έχει ανακαλύψει ο Αλγόριθμος.

Έστω x η τελευταία κορυφή του P_{su} που ανήκει στο S και y η αμέσως επόμενη κορυφή και επομένως $y \in V \setminus S$.

Από την επαγωγική υπόθεση $d(s, x) = d_x$ και επειδή οι συντελεστές των ακμών είναι μη-αρνητικοί αριθμοί θα πρέπει

$$d_x + w(x, y) \leq d(s, u). \quad (5)$$

Όμως θα πρέπει

$$d_y \leq d_x + w(x, y) \quad (6)$$

αφού στην (όποια προηγούμενη) επανάληψη που η κορυφή x έγινε μόνιμη το d_y υπολογίστηκε από τον τύπο

$$d_y = \min\{d_y, d_x + w(x, y)\}$$

Από τις (4), (5), (6), έχουμε $d_y < d_u$. Αυτό όμως αποτελεί αντίφαση στην επιλογή του u από τον τύπο (3).

Πολυπλοκότητα

Έστω $[V] = n$.

Εξετάζουμε την k -ιοστή επανάληψη.

ΦΑΣΗ Ι

$|S| = k - 1$ και επομένως $|V \setminus S| = n - k + 1$. Ο Αλγόριθμος αναζητάει ανάμεσα σε $n - k + 1$ στοιχεία να βρει το μικρότερο. Άρα ο αριθμός των συγκρίσεων προκειμένου να βρεθεί το v^* είναι $a(k) = n - k$.

ΦΑΣΗ ΙΙ

Εφόσον $S \leftarrow S \cup \{v^*\}$ τώρα $|S| = k$ και επομένως $|V \setminus S| = n - k$. Ο Αλγόριθμος εκτελεί προσθέσεις και συγκρίσεις στη φάση αυτή. Για κάθε κορυφή $v \in N(v^*)$ εκτελείται μία πρόσθεση και μία σύγκριση. Ο αριθμός των συγκρίσεων $b(k)$ είναι ίσος με τον αριθμό των προσθέσεων $c(k)$ και φράζεται από τα επάνω από την ποσότητα $n - k$ - το φράγμα επιτυγχάνεται όταν $N(v^*) = V \setminus S$.

Ο αριθμός των στοιχειωδών πράξεων στην επανάληψη k ικανοποιεί τη σχέση

$$a(k) + b(k) + c(k) \leq 3(n - k). \quad (7)$$

Αθροίζοντας για όλες τις τιμές του k παίρνουμε το συνολικό αριθμό των στοιχειωδών πράξεων, ήτοι

$$\begin{aligned} T(n) &\leq \sum_{k=1}^{n-1} 3(n - k) = 3 \left(\sum_{k=1}^{n-1} n - \sum_{k=1}^{n-1} k \right) \\ &= 3 \left(n(n - 1) - \frac{n(n - 1)}{2} \right) = \frac{3}{2} n(n - 1) \Rightarrow \\ T(n) &= O(n^2). \end{aligned}$$

Κατευθυνόμενα Γραφήματα

Αρνητικά βάρη

Ο Αλγόριθμος του Dijkstra μπορεί να εφαρμοστεί και σε κατευθυνόμενα γραφήματα αρκεί τα βάρη των ακμών να είναι μη-αρνητικοί αριθμοί. Στην περίπτωση αυτή υπολογίζονται το συντομότερα κατευθυνόμενα μονοπάτια από την αφετηρία προς όλες τις άλλες ακμές.

Αν υπάρχουν αρνητικά βάρη σε κάποιες από τις ακμές ΣΤΗΝ ΠΕΡΙΠΤΩΣΗ ΤΩΝ ΚΑΤΕΥΘΥΝΟΜΕΝΩΝ ΓΡΑΦΗΜΑΤΩΝ χρησιμοποιείται ο Αλγόριθμος των Bellman-Ford.

Σημειώνουμε ότι αν στο γράφημα υπάρχουν αρνητικά βάρη στις ακμές, τότε μπορεί να υπάρχουν και αρνητικοί κύκλοι. Σε αυτή την περίπτωση το πρόβλημα του συντομότερου μονοπατιού δεν είναι καλά ορισμένο: όσες περισσότερες φορές διασχίσουμε τον κύκλο μικραίνει η απόσταση.

Ο Αλγόριθμος των Bellman-Ford εντοπίζει και την ύπαρξη αρνητικού κύκλου.

Ιδέα

Για τη συνέχεια η αναφορά σε μονοπάτια ή κύκλους υπονοεί την έννοια ‘κατευθυνόμενα’ αφού όλα τα γραφήματα είναι κατευθυνόμενα.

- $d(s, v)$: το μήκος του συντομότερου μονοπατιού από την αφετηρία s μέχρι την κορυφή v .
- d_v^k : το μήκος του συντομότερου μονοπατιού από την αφετηρία s μέχρι την κορυφή v το οποίο αποτελείται από ΤΟ ΠΟΛΥ k ακμές.
- $d_v^k \leq d_v^{k-1}$ και $d_v^{|V|-1} = d(s, v)$.

Μπορούμε να υπολογίσουμε το d_v^k αν γνωρίζουμε τα d_u^{k-1} για κάθε κορυφή $u \in N^-(v) \cup \{v\}$ από τον ακόλουθο τύπο

$$d_v^k = \min\{d_v^{k-1}, \min\{d_u^{k-1} + w(u, v) : u \in N^-(v)\}\} \quad (8)$$

Ο επόμενος αλγόριθμος βασίζεται σε αυτές τις ιδέες.

Αλγόριθμος

```
1: void Bellman-Ford( $G(V, E, w)$ ,  $s$ )
2: for all  $v \in V$  do
3:    $d_v^0 \leftarrow \infty$ ;  $p_v^0 \leftarrow \text{NULL}$ ;
4: end for
5:  $d_s^0 \leftarrow 0$ ;
6: for  $k \leftarrow 1$ ;  $k \leq |V| - 1$ ;  $k++$  do
7:   for all  $v \in V$  do
8:      $d_v^k \leftarrow d_v^{k-1}$ ;  $p_v^k \leftarrow p_v^{k-1}$ ;
9:   end for
10:  for  $(u, v) \in E$  do
11:    if  $d_v^k > d_u^{k-1} + w(u, v)$  then
12:       $d_v^k \leftarrow d_u^{k-1} + w(u, v)$ ;
13:       $p_v^k \leftarrow u$ ;
14:    end if
15:  end for
16: end for
    ///////////////////////////////////////////////////////////////////check for a negative cycle
for  $(u, v) \in E$  do
  if  $d_v^{|V|-1} > d_u^{|V|-1} + w(u, v)$  then
    Print "Negative Cycle"
  end if
end for
```

Παρατήρηση: Οι γραμμές 7-15 υλοποιούν την (8) για κάθε $v \in V \setminus \{s\}$. Οι γραμμές 18-22 ελέγχουν την ύπαρξη αρνητικού κύκλου.

Απόδειξη

Πρόταση 2 Αν το κατευθυνόμενο γράφημα $G(V, E)$ δεν περιέχει αρνητικό κύκλο ο αλγόριθμος των Bellman-Ford υπολογίζει την ελάχιστη απόσταση από την αφετηρία προς κάθε άλλη κορυφή.

Απόδειξη Όπως και στην προηγούμενη περίπτωση $d(s, v)$ συμβολίζει το μήκος του συντομότερου μονοπατιού από την κορυφή s ως την κορυφή v , για κάθε $v \in V$. Με τη χρήση επαγωγής, στην τιμή του k , θα δείξουμε ότι d_v^k θα περιέχει το μήκος του μικρότερο μονοπατιού από την κορυφή s στην κορυφή v με αριθμό ακμών $\leq k$.

Βάση Επαγωγής: Για $k = 0$, ο αλγόριθμος επιστρέφει το σωστό αποτέλεσμα αφού

$$d_s^0 = 0 = d(s, s), d_v^0 = \infty, \forall v \in V \setminus \{s\}.$$

Επαγωγικό Βήμα: Έστω ότι για κάθε κορυφή u , d_u^{k-1} είναι το μήκος του συντομότερου μονοπατιού από την s στη u , χρησιμοποιώντας το πολύ $k - 1$ ακμές (επαγωγική υπόθεση). Έστω P το συντομότερο μονοπάτι από την s στην v , με $\leq k$ ακμές και μήκος $w(P)$. Έστω u η προτελευταία κορυφή στο μονοπάτι αυτό. Λόγω της ιδιότητας της βέλτιστης υποδομής το μονοπάτι αυτό περιέχει το συντομότερο μονοπάτι από την s στη u , έστω Q . Το μονοπάτι Q περιέχει $\leq k - 1$ κορυφές και άρα $\leq k - 1$ ακμές. Έστω $w(Q)$ το μήκος του μονοπατιού αυτού. Από την επαγωγική υπόθεση

$$w(Q) = d_u^{k-1}. \quad (9)$$

Επίσης από κατασκευής ισχύει ότι

$$w(P) = w(Q) + w(u, v). \quad (10)$$

Στην επανάληψη k επανυπολογίζουμε την απόσταση d_v^k από την (8) η οποία γράφεται

$$d_v^k = \min\{d_v^{k-1}, d_u^{k-1} + w(u, v)\}. \quad (11)$$

Άρα $d_v^k \leq d_u^{k-1} + w(u, v)$ η οποία σε συνδυασμό με (9) και στη συνέχεια με (10) παράγει

$$d_v^k \leq w(Q) + w(u, v) \Rightarrow d_v^k \leq w(P) \quad (12)$$

Από την επαγωγική υπόθεση, το d_v^{k-1} είναι το μήκος του συντομότερου μονοπατιού από την s στη v , το οποίο περιέχει το πολύ $k - 1$ ακμές. Άρα το μήκος αυτό θα είναι τουλάχιστον ίσο με το $w(P)$ αφού το μονοπάτι P μπορεί να χρησιμοποιεί παραπάνω ακμές. Δηλαδή $w(P) \leq d_v^{k-1}$.

Από την τελευταία παρατήρηση και την (11), έχουμε

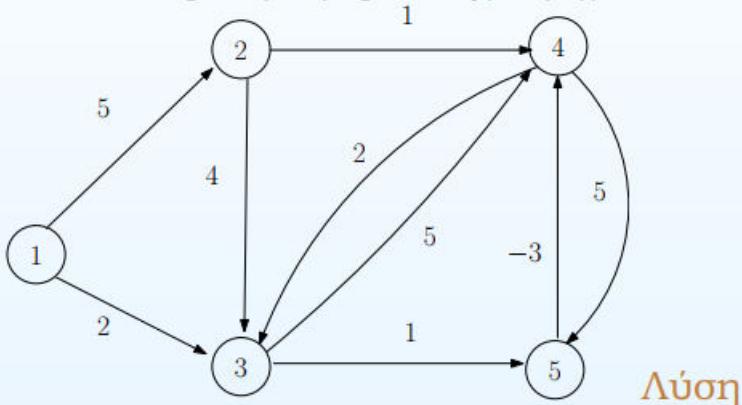
$$d_v^k = w(P).$$

Πολυπλοκότητα

Πρόταση 3 Ο αλγόριθμος Bellman-Ford εκτελεί $O(|V| \cdot |E|)$ βήματα.

Παράδειγμα

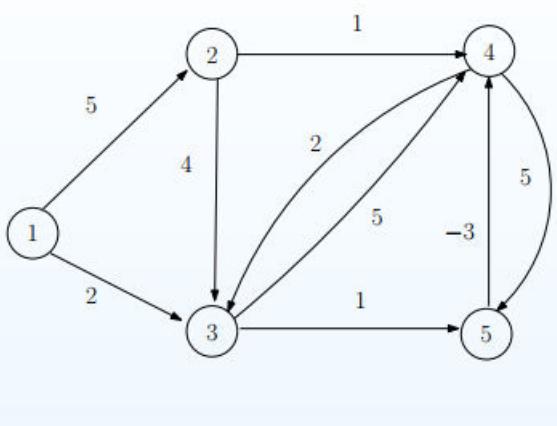
Με τη χρήση του αλγόριθμου των Bellman-Ford να βρεθούν τα συνομότερα μονοπάτια από την κορυφή 1 προς τις υπόλοιπες κορυφές στο γράφημα



Αρχικοποίηση

$$d = [0, \infty, \infty, \infty, \infty]$$

$$p = [, , , ,]$$



$$d^0 = [0, \infty, \infty, \infty, \infty]$$

$$p^0 = [, , , ,]$$

Επανάληψη $k = 1$

$$d_1^1 = d_1^0 = 0,$$

$$d_2^1 = \min\{d_2^0, d_1^0 + w(1, 2)\} = \min\{\infty, 0 + 5\} = 5, \quad p_2^1 = 1,$$

$$d_3^1 = \min\{d_3^0, d_1^0 + w(1, 3), d_2^0 + w(2, 3), d_4^0 + w(4, 3)\}$$

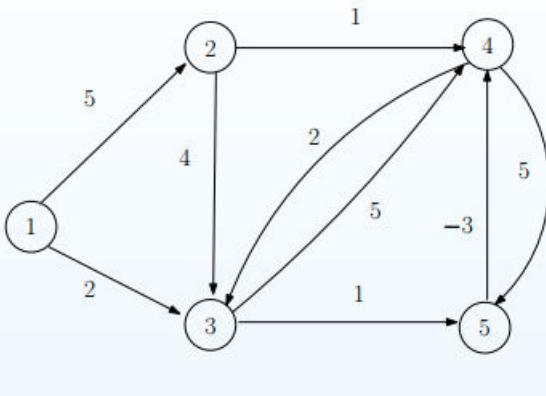
$$= \min\{\infty, 0 + 2, \infty + 4, \infty + 2\} = 2, \quad p_3^1 = 1,$$

$$d_4^1 = \min\{d_4^0, d_2^0 + w(2, 4), d_3^0 + w(3, 4), d_5^0 + w(5, 4)\}$$

$$= \min\{\infty, \infty + 1, \infty + 5, \infty - 3\} = \infty,$$

$$d_5^1 = \min\{d_5^0, d_3^0 + w(3, 5), d_4^0 + w(4, 5)\}$$

$$= \min\{\infty, \infty + 1, \infty + 5, \infty\} = \infty$$



$$d^1 = [0, 5, 2, \infty, \infty]$$

$$p^1 = [, 1, 1, ,]$$

Επανάληψη $k = 2$

$$d_1^2 = d_1^1 = 0,$$

$$d_2^2 = \min\{d_2^1, d_1^1 + w(1, 2)\} = \min\{5, 0 + 5\} = 5,$$

$$d_3^2 = \min\{d_3^1, d_1^1 + w(1, 3), d_2^1 + w(2, 3), d_4^1 + w(4, 3)\}$$

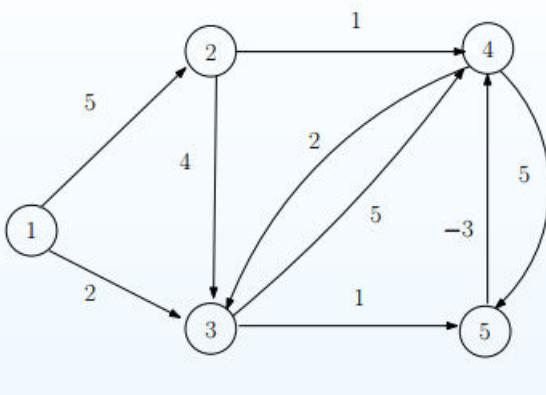
$$= \min\{2, 0 + 2, 5 + 4, \infty + 2\} = 2,$$

$$d_4^2 = \min\{d_4^1, d_2^1 + w(2, 4), d_3^1 + w(3, 4), d_5^1 + w(5, 4)\}$$

$$= \min\{\infty, 5 + 1, 2 + 5, \infty - 3\} = 6, \quad p_4^2 = 2,$$

$$d_5^2 = \min\{d_5^1, d_3^1 + w(3, 5), d_4^1 + w(4, 5)\}$$

a.gr = min{∞, 2 + 1, ∞ + 5, } = 3 p₅² = 3 2



$$d^2 = [0, 5, 2, 6, 3]$$

$$p^2 = [, 1, 1, 2, 3]$$

Επανάληψη $k = 3$

$$d_1^3 = d_1^2 = 0,$$

$$d_2^3 = \min\{d_2^2, d_1^2 + w(1, 2)\} = \min\{5, 0 + 5\} = 5,$$

$$d_3^3 = \min\{d_3^2, d_1^2 + w(1, 3), d_2^2 + w(2, 3), d_4^2 + w(4, 3)\}$$

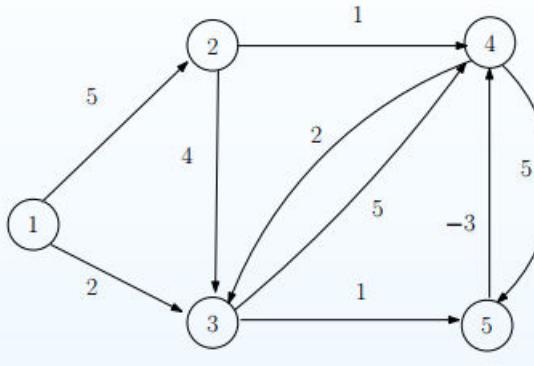
$$= \min\{2, 0 + 2, 5 + 4, 6 + 2\} = 2,$$

$$d_4^3 = \min\{d_4^2, d_2^2 + w(2, 4), d_3^2 + w(3, 4), d_5^2 + w(5, 4)\}$$

$$= \min\{6, 5 + 1, 2 + 5, 3 - 3\} = 0, \quad p_4^3 = 5,$$

$$d_5^3 = \min\{d_5^2, d_3^2 + w(3, 5), d_4^2 + w(4, 5)\}$$

wa.gr = min{3, 2 + 1, 6 + 5, } = 3 29



$$d^3 = [0, 5, 2, 0, 3]$$

$$p^3 = [, 1, 1, 5, 3]$$

Επανάληψη $k = 4$

$$d_1^4 = d_1^3 = 0,$$

$$d_2^4 = \min\{d_2^3, d_1^3 + w(1, 2)\} = \min\{5, 0 + 5\} = 5,$$

$$\begin{aligned} d_3^4 &= \min\{d_3^3, d_1^3 + w(1, 3), d_2^3 + w(2, 3), d_4^3 + w(4, 3)\} \\ &= \min\{2, 0 + 2, 5 + 4, 0 + 2\} = 2, \end{aligned}$$

$$\begin{aligned} d_4^4 &= \min\{d_4^3, d_2^3 + w(2, 4), d_3^3 + w(3, 4), d_5^3 + w(5, 4)\} \\ &= \min\{0, 5 + 1, 2 + 5, 3 - 3\} = 0, \end{aligned}$$

$$d_5^4 = \min\{d_5^3, d_3^3 + w(3, 5), d_4^3 + w(4, 5)\}$$

$$\underline{\underline{m}} = \min\{3, 2 + 1, 0 + 5, \} = 3 \quad \underline{\underline{m}}$$

Output

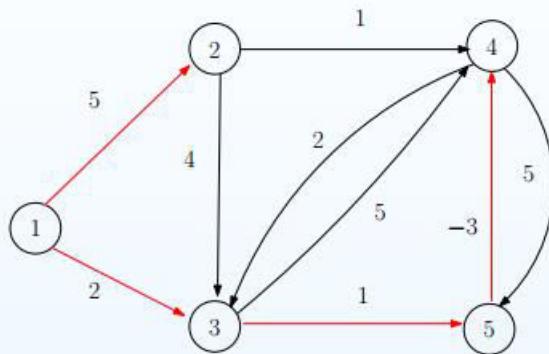
$$d = [0, 5, 2, 0, 3]$$

$$p = \begin{bmatrix} 1 & 1 \\ 1 & 1 & 2 & 3 \\ 1 & 1 & 5 & 3 \\ 1 & 1 & 5 & 3 \end{bmatrix}$$

Παρατήρηση: Στην έξοδο εμφανίζονται (υπό μορφή πίνακα p) όλα τα διανύσματα p^k για κάθε $k = 1, \dots, |V| - 1$.

Χρειαζόμαστε όλη την πληροφορία αυτή για να εντοπίσουμε τα συντομότερα μονοπάτια από την αφετηρία προς κάθε άλλη κορυφή. Για παράδειγμα, για να βρούμε τη συντομότερη διαδρομή από την κορυφή 1 στην 4 ξεκινώντας από την τελευταία κορυφή έχουμε

$$4 \leftarrow p_4^4 = 5 \leftarrow p_5^3 = 3 \leftarrow p_3^2 = 1 \leftarrow p_1^1 =$$



Σχήμα 3: Παράδειγμα Bellman-Ford : Δένδρο συντομότερων διαδρομών

Παρατήρηση

- Μπορούμε να τερματίσουμε τον αλγόριθμο αν σε δύο διαδοχικές επαναλήψεις δεν μεταβληθεί το διάνυσμα των συντομότερων αποστάσεων (d).
- Το πρόβλημα εύρεσης των συντομότερων μονοπατιών κοινής αφετηρίας σε ένα μη-κατευθυνόμενο γράφημα $G(V, E, w)$ με μη-αρνητικά βάρη στις ακμές μπορεί να λυθεί με τον αλγόριθμο των Bellman-Ford ως εξής: κατασκευάζουμε το κατευθυνόμενο γράφημα $G'(V, E', w)$, όπου για κάθε μη κατευθυνόμενη ακμή $\{v, u\} \in E$ εισάγουμε ακμές $(v, u) \in E'$ και $(u, v) \in E'$ με $w((v, u)) = w((u, v)) = w(\{v, u\})$.
- Δυστυχώς ο μετασχηματισμός αυτός παράγει ένα κατευθυνόμενο γράφημα με αρνητικό κύκλο αν το βάρος κάποιας ακμής είναι αρνητικός αριθμός. Στην περίπτωση αυτή το πρόβλημα μπορεί να επιλυθεί μέσω ενός 'τέλειου ταιριάσματος'.

Κατευθυνόμενα Άκυκλα Γράφηματα

ΚΑΓ

Το πρόβλημα ΜΠΣΔ απλοποιείται σημαντικά αν το γράφημα δεν έχει κύκλο -στην περίπτωση αυτή το γράφημα ονομάζεται Κατευθυνόμενο Άκυκλο Γράφημα (ΚΑΓ).

Βασική Ιδέα

Έστω ότι έχουμε αριθμίσει τις κορυφές του γραφήματος με τους αριθμούς $1, 2, 3, \dots, n$ κατά τρόπο ώστε να υπάρχει ακμή από μία κορυφή i σε μία κορυφή j ανν $i < j$ (στην παραπάνω αρίθμηση.) Χώρις βλάβη της γενικότητας θεωρούμε ότι η κορυφή πηγή (s) έχει αρίθμηση 1.

Τα μήκη των συντομότερων διαδρομών από την κορυφή 1 προς τις υπόλοιπες κορυφές δύνονται από τη σχέση

$$d_v = \begin{cases} 0, & v = 1, \\ \min\{d_j + w(j, v) : j < v, j \in N^-(v)\}, & v \in \{2, \dots, |V|\} \end{cases}$$

Αλγόριθμος

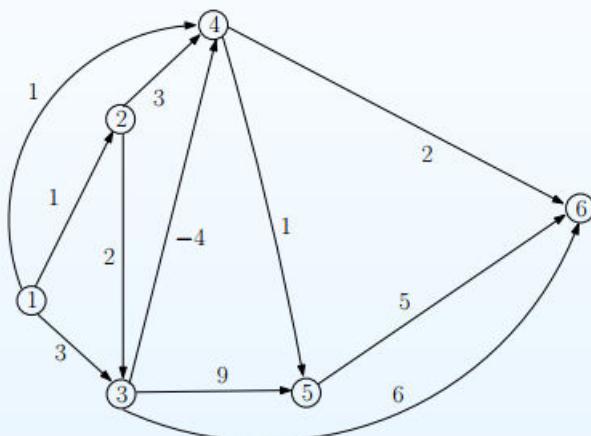
Ο Αλγόριθμος που ακολουθεί υπολογίζει τις συντομότερες διαδρομές σε ένα ΚΑΓ από την κορυφή 1.

```
1: void DAG( $G(V, E, w)$ )
2: for all  $v \leftarrow 1; v \leq |V|; v++ \text{ do}$ 
3:    $d_v \leftarrow \infty; p_v \leftarrow \text{NULL};$ 
4: end for
5:  $d_1 \leftarrow 0;$ 
6: for  $v \leftarrow 2; v \leq |V|; v++ \text{ do}$ 
7:    $d_v \leftarrow \min\{d_j + w(j, v) : j < v, j \in N^-(v)\};$ 
8:    $p_v \leftarrow \operatorname{argmin}\{d_j + w(j, v) : j < v, j \in N^-(v)\};$ 
9: end for
```

Πρόταση 4 Η επίλυση του προβληματος ΜΠΣΔ σε ΚΑΓ επιλύεται σε $O(|V|^2)$ βήματα.

Παράδειγμα

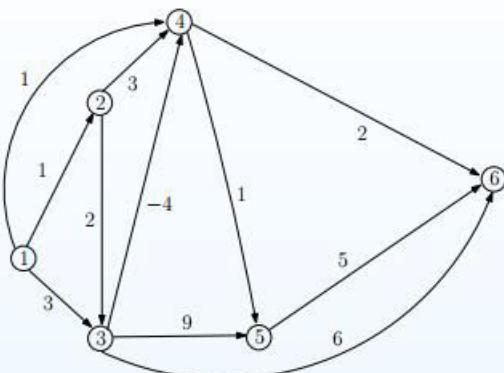
Να βρεθούν οι συντομότερες διαδρομές από την κορυφή 1 προς τις υπόλοιπες κορυφές στο γράφημα του Σχήματος 4.



Σχήμα 4: Βεβαρυμένο ΚΑΓ

Παρατήρηση

Μπορεί να υπάρχουν αρνητικά βάρη στις ακμές από τη στιγμή που το γράφημα δεν περιέχει κύκλους (άρα δεν περιέχει και αρνητικούς κύκλους)



Λύση

$$d_2 = \min\{d_1 + w(1, 2)\} = \min\{0 + 1\} = 1$$

$$d_3 = \min\{d_1 + w(1, 3), d_2 + w(2, 3)\} = \min\{0 + 3, 1 + 2\} = 3$$

$$\begin{aligned} d_4 &= \min\{d_1 + w(1, 4), d_2 + w(2, 4), d_3 + w(3, 4)\} \\ &= \min\{0 + 1, 1 + 3, 3 + (-4)\} = -1 \end{aligned}$$

$$d_5 = \min\{d_3 + w(3, 5), d_4 + w(4, 5)\} = \min\{3 + 9, -1 + 1\} = 0$$

$$d_6 = \min\{d_3 + w(3, 6), d_4 + w(4, 6), d_5 + w(5, 6)\}$$

Το συμπληρωματικό πρόβλημα

Μακρύτερα Μονοπάτια

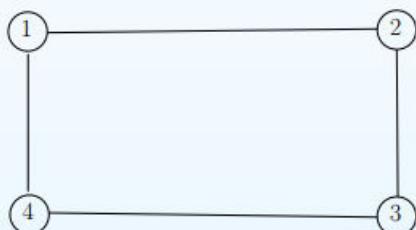
Το συμπληρωματικό του προβλήματος εύρεσης του συντομότερου μονοπατιού είναι το πρόβλημα εύρεσης του μακρύτερου (απλού) μονοπατιού:

Πρόβλημα: Δεδομένου ενός εμβαρούς γραφήματος $G(V, E, w)$ με μη-αρνητικούς συντελεστές ακμών και μίας κορυφής $s \in V$, θέλουμε να βρούμε το μακρύτερο μονοπάτι ανάμεσα στην s και κάθε άλλη κορυφή $v \in V \setminus \{s\}$.

Εκ' πρώτης όψεως το πρόβλημα αυτό φαίνεται εύκολο αφού είναι συμμετρικό του προβλήματος που έχουμε μελετήσει στα προηγούμενα. Παρόλα αυτά το πρόβλημα αυτό δύσκολο (ανήκει στην κατηγορία NP-hard) αφού αν μπορούσαμε να το επιλύσουμε θα μπορούσαμε να απαντήσουμε αν το γράφημα περιέχει μονοπάτι Hamilton: ένα μονοπάτι που επισκέπτεται όλες τις κορυφές ακριβώς μία φορά

Ιδιότητες

Ο βασικός λόγος που το πρόβλημα αυτό είναι δύσκολο φαίνεται να είναι ότι δεν υπάρχει η ιδιότητα της βέλτιστης υποδομής. Για παράδειγμα, θεωρούμε το παρακάτω γράφημα όπου κάθε ακμή έχει βάρος 1.



Το μακρύτερο μονοπάτι από το 1 στο 4 είναι το $P = 1 - 2 - 3 - 4$. Όμως το μακρύτερο μονοπάτι από το 2 στο 3 ΔΕΝ είναι αυτό που περιέχεται στο P αλλά το $2 - 1 - 4 - 3$.

Άκυκλα Γραφήματα

Το παραπάνω παράδειγμα μας προϊδεάζει ότι το πρόβλημα εύρεσης του μακρύτερου μονοπατιού ίσως είναι εύκολο για τα άκυκλα γραφήματα. Αυτό όντως ισχύει τόσο για κατευθυνόμενα όσο και για τα μη-κατευθυνόμενα γραφήματα. Στην περίπτωση αυτή με απλές μετατροπές στο γράφημα μπορούμε να χρησιμοποιήσουμε τον αλγόριθμο που είδαμε μέχρι τώρα και να υπολογίσουμε τα μακρύτερα μονοπάτια.

Συγκεκριμένα,

- αν το γράφημα είναι ΚΑΓ, μπορούμε να πολλαπλασιάσουμε κάθε βάρος ακμής με -1 και να επιλύσουμε ένα πρόβλημα ΜΠΣΔ στο τροποποιημένο γράφημα,
- αν το γράφημα δεν είναι κατευθυνόμενο και δεν έχει αρνητικούς κύκλους μπορούμε να το ανάγουμε σε ένα πρόβλημα εύρεσης τέλειου ταιριάσματος (δεν είναι στους σκοπούς του παρόντος μαθήματος)

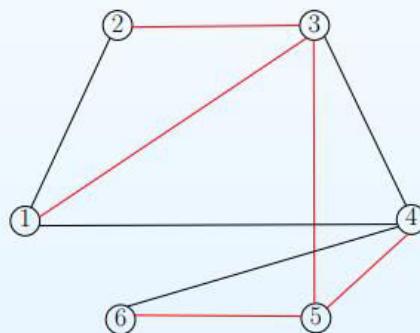
05_ΕΛΑΧΙΣΤΟ ΣΥΝΔΕΤΙΚΟ ΔΕΝΔΡΟ / 06_sptrees

Ορισμοί και Ιδιότητες

Ορισμοί

Έστω $G(V, E)$ ένα μη κατευθυνόμενο συνδεδεμένο γράφημα.

Ένα υπογράφημα του G , με σύνολο κορυφών το V και σύνολο ακμών $E' \subseteq E$, το οποίο είναι δένδρο ονομάζεται συνδετικό ή γενινητορικό δένδρο του G .



Σχήμα 1: Γράφημα $G(V, E)$ και ένα συνδετικό του δένδρο (κόκκινες ακμές)

Συμβάσεις

Για απλοποίηση της παρουσιάσης θα χρησιμοποιήσουμε τις εξής συμβάσεις σε σχέση με ένα μη-κατευθυνόμενο γράφημα $G(V, E)$ και ένα υπογράφημα T του G .

- Η ακμή e που συνδέει δυο κορυφές u, v συμβολίζεται με $(u, v) \in E(G)$ αντί του πιο ορθού $\{u, v\} \in E(G)$.
- Το υπογράφημα που προκύπτει από την αφαίρεση μίας ακμής $e \in E(T)$ συμβολίζεται με $T - e$.
- Το υπογράφημα που προκύπτει από την προσθήκη μίας ακμής $e' \in E(G) \setminus E(T)$ συμβολίζεται με $T + e'$.

Ιδιότητες

Θεώρημα 1 *Κάθε γράφημα περιέχει ένα συνδετικό δένδρο ανν το γράφημα είναι συνδεδεμένο.*

- Ένα γράφημα μπορεί να έχει πάνω του ενός συνδετικά δένδρα. Το πλήρες γράφημα K_n έχει n^{n-2} συνδετικά δένδρα.
- Εάν T είναι ένα δένδρο με k ακμές και G είναι απλό γράφημα με $\delta(G) \geq k$, τότε το T είναι ένας υπογράφος του G .

Λήμμα 2 *Έστω T ένα συνδετικό δένδρο ενός βεβαρημένου, μη-κατευθυνόμενου, συνδεδεμένου γραφήματος $G(V, E, w)$ και ακμή $(u, v) \in E(G) \setminus E(T)$. Το γράφημα $T + (u, v)$ περιέχει κύκλο.*

Απόδειξη Εφόσον το T είναι συνδετικό δένδρο του G υπάρχει ένα μονοπάτι με ακμές που ανήκουν στο $E(T)$ που συνδέει το u με το v . Επομένως η προσθήκη της ακμής (u, v) δημιουργεί κύκλο.

Πόρισμα 4 *Εάν T, T' είναι συνδετικά δένδρα ενός συνδεδεμένου γράφου G και $e \in E(T) \setminus E(T')$, τότε υπάρχει μια ακμή $e' \in E(T') \setminus E(T)$ έτσι ώστε $T - e + e'$ είναι συνδετικό δένδρο του G .*

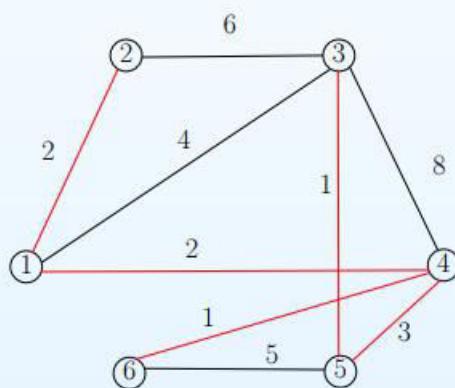
Λήμμα 3 Έστω $G(V, E, w)$, T , (u, v) όπως ορίστηκαν στο Λήμμα 2 και C ο κύκλος που εμπεριέχεται στο $T + (u, v)$. Επιπλέον έστω $(x, y) \in E(C)$. Το γράφημα $T' = T + (u, v) - (x, y)$ είναι συνδετικό δένδρο του G .

Απόδειξη Αφού $(u, v) \notin E(T)$ και $(x, y) \in E(C) \subseteq E(T) \cup \{(u, v)\}$,

$$|E(T')| = |E(T)| + 1 - 1 = |V(G)| - 1.$$

Βεβαρημένα Γραφήματα

Έστω $G(V, E, w)$ ένα βεβαρημένο γράφημα. Ένα ελάχιστο συνδετικό δέντρο -ΕΣΔ (minimum spanning tree - MST) του G είναι ένα συνδετικό δέντρο του G με ελάχιστο βάρος (άθροισμα των βαρών των ακμών του).



Σχήμα 2: Γράφημα $G(V, E, w)$ και το ελάχιστο συνδετικό του δένδρο (κόκκινες ακμές)

Το πρόβλημα

ΕΣΔ

ΕΣΔ Δεδομένου ενός βεβαρημένου συνδεδεμένου γραφήματος $G(V, E, w)$, να βρεθεί το συνδετικό δένδρο του G να ελαχιστοποιεί το συνολικό βάρος των ακμών που συμμετέχουν σε αυτό.

Δηλαδή αν \mathcal{T} είναι το σύνολο των συνεκτικών δένδρων του G , ζητάμε

$$\min\{w(T) : T \in \mathcal{T}\},$$

$$\text{όπου } w(T) = \sum\{w(e) : e \in E(T)\}.$$

Ιδέα μέθοδος απλησίας - σε κάθε βήμα επιλέγουμε να προσθέσουμε την επόμενη ελάχιστη ακμή που συνδέει δύο σύνολα κορυφών

Ιδιότητες

- Ένα συνδεδεμένο γράφημα G με διακριτά βάρη ακμών έχει ένα μοναδικό ελάχιστο συνδετικό δένδρο.
- Εάν το ελάχιστο βάρος ακμής είναι μοναδικό, τότε η ακμή αυτή περιέχεται σε όλα τα ελάχιστα συνδετικά δένδρα του G .
- Έστω συνδεδεμένο γράφημα G με διακριτά βάρη ακμών, κύκλος C και έστω e η ακμή μέγιστου βάρους του κύκλου. Τότε το ΕΣΔ δεν περιέχει την ακμή e .
- Έστω συνδεδεμένο γράφημα G με βάρη στις ακμές και έστω $U \subset V$. Εάν (u, v) είναι μια ακμή ελάχιστου βάρους έτσι ώστε $u \in U$ και $v \in V \setminus U$, τότε υπάρχει ένα ΕΣΔ το οποίο περιέχει την (u, v) .

Αλγόριθμος Prim

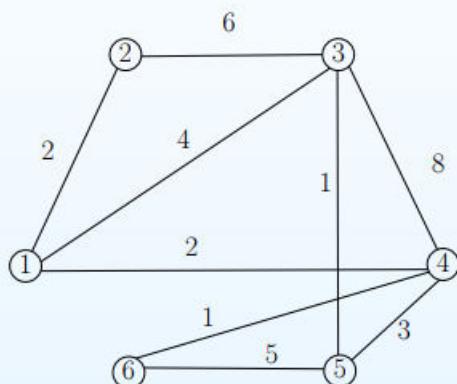
Βασικό Βήμα Έστω $V(T) \subset V$ και $E(T) \subset E$ το σύνολο των κορυφών και ακμών του ΕΣΔ που θέλουμε να υπολογίσουμε. Σε κάθε βήμα οι κορυφές του γραφήματος διαχωρίζονται σε δύο σύνολα: στο σύνολο των κορυφών που ανήκουν στο ΕΣΔ ($V(T)$) και στο σύνολο των υπόλοιπων κορυφών ($V \setminus V(T)$). Σε κάθε βήμα εμπλουτίζεται το σύνολο $V(T)$ προσθέτοντας σε αυτό την κορυφή του συνόλου $V \setminus V(T)$ που συνδέεται με την ‘έλαφρύτερη’ ακμή με τις υπόλοιπες κορυφές του συνόλου $V(T)$.

Ο αλγόριθμος κωδικοποιείται στη συνέχεια. Το σύνολο $V(T)$ αρχικοποιείται από μία οποιαδήποτε κορυφή του V .

- 1: $V(T) \leftarrow \{v\}; E(T) \leftarrow \emptyset;$
- 2: **for** $i \leftarrow 1; i \leq |V| - 1; i + +$ **do**
- 3: $(v^*, u^*) \leftarrow \operatorname{argmin}\{w(v, u) : v \in V(T), u \in V \setminus V(T), (v, u) \in E\};$
- 4: $V(T) \leftarrow V(T) \cup \{u^*\};$
- 5: $E(T) \leftarrow E(T) \cup \{(v^*, u^*)\};$
- 6: **end for**

Παράδειγμα

Παράδειγμα 5 Με τη χρήση του αλγόριθμου του Prim, να βρεθεί το ΕΣΔ του γραφήματος στου Σχήματος 3



Σχήμα 3: Γράφημα $G(V, E, w)$

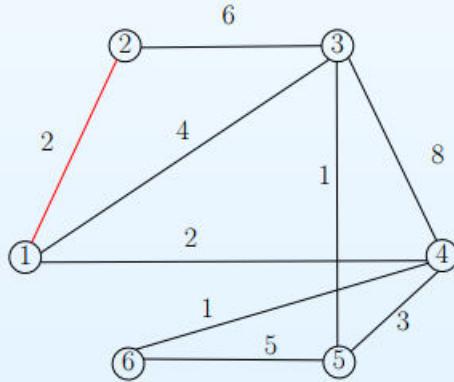
Λύση Αρχικώς έστω $V(T) = \{1\}$

Επανάληψη 1

$$\min\{w(1, 2), w(1, 3), w(1, 4)\} = \min\{2, 4, 2\} = 2.$$

$$V(T) = \{1, 2\}$$

$$E(T) = \{(1, 2)\}$$



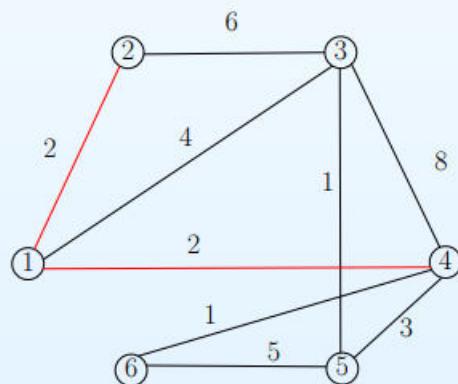
Σχήμα 3: Με κόκκινο οι ακμές του δένδρου

Επανάληψη 2

$$\min\{w(1, 3), w(1, 4), w(2, 3)\} = \min\{4, 2, 6\} = 2.$$

$$V(T) = \{1, 2, 4\}$$

$$E(T) = \{(1, 2), (1, 4)\}$$



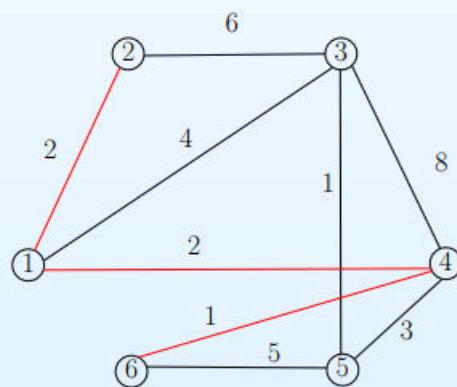
Σχήμα 3: Με κόκκινο οι ακμές του δένδρου

Επανάληψη 3

$$\begin{aligned} & \min\{w(1, 3), w(2, 3), w(4, 3), w(4, 5), w(4, 6)\} \\ &= \min\{4, 6, 8, 3, 1\} = 1. \end{aligned}$$

$$V(T) = \{1, 2, 4, 6\}$$

$$E(T) = \{(1, 2), (1, 4), (4, 6)\}$$



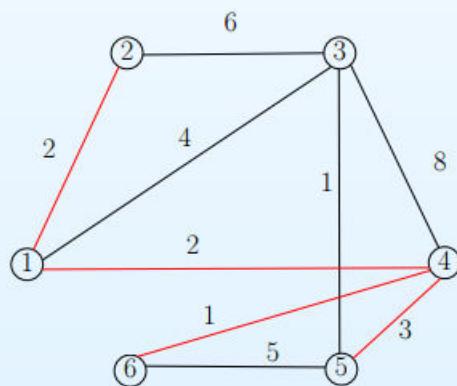
Σχήμα 3: Με κόκκινο οι ακμές του δένδρου

Επανάληψη 4

$$\begin{aligned} & \min\{w(1, 3), w(2, 3), w(4, 3), w(4, 5), w(6, 5)\} \\ &= \min\{4, 6, 8, 3, 5\} = 3. \end{aligned}$$

$$V(T) = \{1, 2, 4, 6, 5\}$$

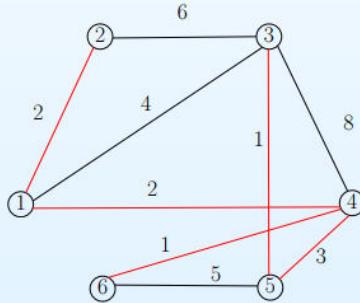
$$E(T) = \{(1, 2), (1, 4), (4, 6), (4, 5)\}$$



Σχήμα 3: Με κόκκινο οι ακμές του δένδρου

Επανάληψη 5

$$\begin{aligned} & \min\{w(1, 3), w(2, 3), w(4, 3), w(5, 3)\} \\ &= \min\{4, 6, 8, 1\} = 1. \\ V(T) &= \{1, 2, 4, 6, 5, 3\} \\ E(T) &= \{(1, 2), (1, 4), (4, 6), (4, 5), (5, 3)\} \end{aligned}$$



Σχήμα 3: Με κόκκινο οι ακμές του δένδρου

Ορθότητα

Πρόταση 6 Έστω T το συνδετικό δένδρο του συνδεδεμένου γραφήματος $G(V, E, w)$ που υπολογίζει ο αλγόριθμος του Prim. Το T είναι ένα ΕΣΔ του G .

Απόδειξη Έστω ότι T^* ένα ΕΣΔ του G . Θα δείξουμε ότι $w(T) = w(T^*)$. Αν $T = T^*$ αυτό ισχύει. Αν $T \neq T^*$ τότε θα υπάρχει (τουλάχιστον) μία ακμή $(u, v) \in E(T) \setminus E(T^*)$. Όταν ο αλγόριθμος πρόσθεσε την ακμή αυτή στο T ήταν η ακμή μικρότερου κόστους από το σύνολο $V(T)$ στο σύνολο $V \setminus V(T)$. Επειδή το T^* είναι συνδετικό δένδρο θα υπάρχει ένα μονοπάτι στο T^* , έστω $P_{u,v}$, από την κορυφή u στην κορυφή v . Στο μονοπάτι αυτό θα πρέπει να υπάρχει ακμή (x, y) τέτοια ώστε $x \in V(T)$, $y \in V \setminus V(T)$. Θα πρέπει

$$w(u, v) \leq w(x, y) \quad (1)$$

(Απόδειξη-συνχ.) Σύμφωνα με το Λήμμα 3 το $\hat{T}^* = T^* + (u, v) - (x, y)$ είναι συνδετικό δένδρο με βάρος

$$w(\hat{T}^*) = w(T^*) + w(u, v) - w(x, y).$$

Από την (1)

$$w(\hat{T}^*) \leq w(T^*) \Rightarrow w(\hat{T}^*) = w(T^*)$$

αφού το \hat{T}^* είναι συνδετικό δένδρο και το T^* είναι ένα ΕΣΔ. Επειδή $|E(T) \setminus E(\hat{T}^*)| = |E(T) \setminus E(T^*)| - 1$, μπορούμε να επαναλάβουμε τη διαδικασία αυτή για κάθε ακμή του συνόλου $E(T) \setminus E(T^*)$ μετατρέποντας το T^* σε T διατηρώντας το ίδιο συνολικό βάρος $w(T^*)$. Έτοιμος $w(T^*) = w(T)$.

Πολυπλοκότητα

Ανάλογα με την υλοποίηση και τη χρήση προχωρημένων δομών δεδομένων ο αλγόριθμος υλοποίεται με πολυπλοκότητες που απεικονίζονται στον Πίνακα 1

Δομή Δεδομένων	Πολυπλοκότητα
λίστα γειτνίασης	$O(V ^2)$
δυαδικός σωρός	$O((V + E) \log V) = O(E \log V)$
σωρός Fibonacci	$O(E + V \log V)$

Πίνακας 1: Αλγόριθμος Prim - Διαφορετικές Υλοποιήσεις

Αλγόριθμος Kruskal

Ο αλγόριθμος κτίζει βήμα-βήμα το συνδετικό δένδρο προσθέτοντας ακμές εφαρμόζοντας κατά γράμμα την άπληστη αρχή: σε κάθε βήμα κάνει την καλύτερη επιλογή (επιλέγει την ακμή με το μικρότερο βάρος) χωρίς να παραβιάζει τους περιορισμούς (που αν την προσθέσει στο υποσύνολο των ακμών που έχει δημιουργηθεί στα προηγούμενα βήματα δεν δημιουργείται κύκλος).

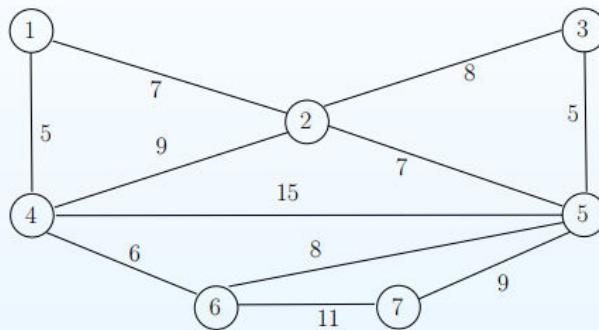
Είσοδος: Συνδεδεμένο γράφημα $G(V, E, w)$

Έξοδος: Ελάχιστο συνδετικό δένδρο T του G

- 1: $m \leftarrow |E|;$
- 2: Διέταξε τις ακμές του $E = \{e_1, e_2, \dots, e_m\}$ ώστε $w(e_1) \leq w(e_2) \leq \dots \leq w(e_m)$.
- 3: $E(T) \leftarrow \emptyset;$
- 4: $i \leftarrow 1;$
- 5: **while** $|E(T)| < |V| - 1$ **do**
- 6: **if** $E(T) \cup \{e_i\}$ δεν περιέχει κύκλο **then**
- 7: $E(T) \leftarrow E(T) \cup \{e_i\};$
- 8: **end if**
- 9: $i++;$
- 10: **end while**

Παράδειγμα

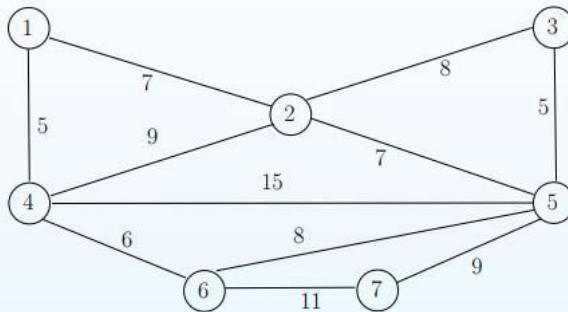
Παράδειγμα 7 Με τη χρήση του αλγόριθμου του Kruskal, να βρεδεί το ΕΣΔ του γραφήματος στου Σχήματος 4



Σχήμα 4: Αλγόριθμος Kruskal - κατασκευή ΕΣΔ

Παράδειγμα - Επίλυση

Λύση

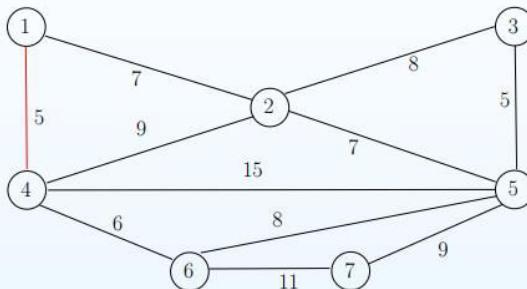


Σχήμα 5: Αλγόριθμος Kruskal - κατασκευή ΕΣΔ

Οι ακμές κατά αύξουσα σειρά βάρους:

$$(1, 4) - (3, 5) - (4, 6) - (1, 2) - (2, 5) - (2, 3) \\ - (6, 5) - (2, 4) - (5, 7) - (6, 7) - (4, 5)$$

Επανάληψη 1

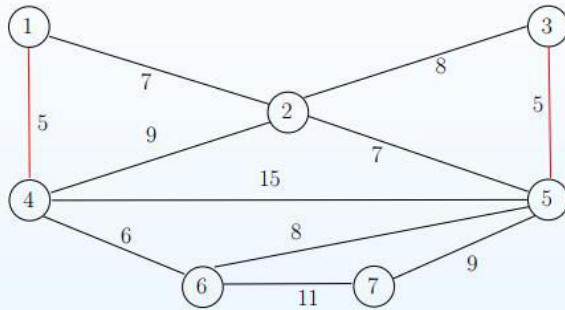


Σχήμα 5: Αλγόριθμος Kruskal - κατασκευή ΕΣΔ

Οι ακμές κατά αύξουσα σειρά βάρους:

$$(1, 4) - (3, 5) - (4, 6) - (1, 2) - (2, 5) - (2, 3) \\ - (6, 5) - (2, 4) - (5, 7) - (6, 7) - (4, 5)$$

Επανάληψη 2

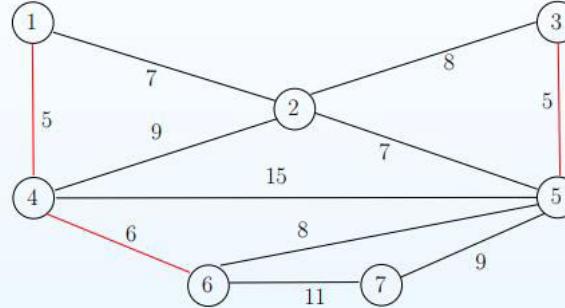


Σχήμα 5: Αλγόριθμος Kruskal - κατασκευή ΕΣΔ

Οι ακμές κατά αύξουσα σειρά βάρους:

$$\begin{aligned} & (1, 4) - (3, 5) - (4, 6) - (1, 2) - (2, 5) - (2, 3) \\ & - (6, 5) - (2, 4) - (5, 7) - (6, 7) - (4, 5) \end{aligned}$$

Επανάληψη 3

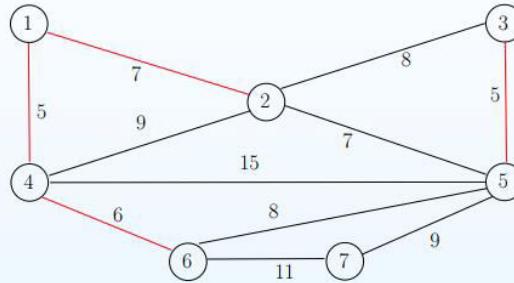


Σχήμα 5: Αλγόριθμος Kruskal - κατασκευή ΕΣΔ

Οι ακμές κατά αύξουσα σειρά βάρους:

$$\begin{aligned} & (1, 4) - (3, 5) - (4, 6) - (1, 2) - (2, 5) - (2, 3) \\ & - (6, 5) - (2, 4) - (5, 7) - (6, 7) - (4, 5) \end{aligned}$$

Επανάληψη 4

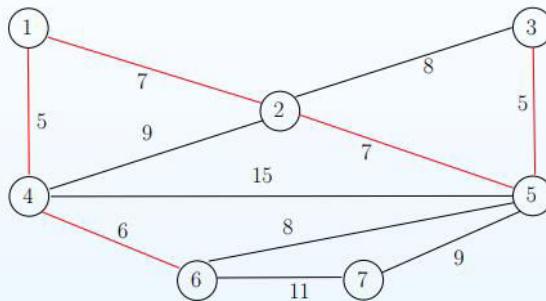


Σχήμα 5: Αλγόριθμος Kruskal - κατασκευή ΕΣΔ

Οι ακμές κατά αύξουσα σειρά βάρους:

$$\begin{aligned} & (1, 4) - (3, 5) - (4, 6) - (1, 2) - (2, 5) - (2, 3) \\ & - (6, 5) - (2, 4) - (5, 7) - (6, 7) - (4, 5) \end{aligned}$$

Επανάληψη 5

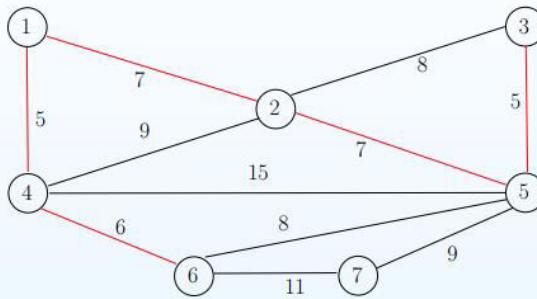


Σχήμα 5: Αλγόριθμος Kruskal - κατασκευή ΕΣΔ

Οι ακμές κατά αύξουσα σειρά βάρους:

$$\begin{aligned} & (1,4) - (3,5) - (4,6) - (1,2) - (2,5) - (2,3) \\ & - (6,5) - (2,4) - (5,7) - (6,7) - (4,5) \end{aligned}$$

Επανάληψη 6

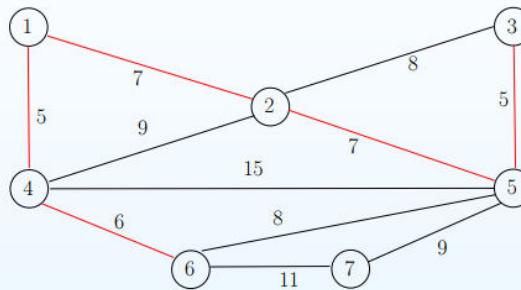


Σχήμα 5: Αλγόριθμος Kruskal - κατασκευή ΕΣΔ

Οι ακμές κατά αύξουσα σειρά βάρους:

$$\begin{aligned} & (1,4) - (3,5) - (4,6) - (1,2) - (2,5) - (2,3) \\ & - (6,5) - (2,4) - (5,7) - (6,7) - (4,5) \end{aligned}$$

Επανάληψη 7

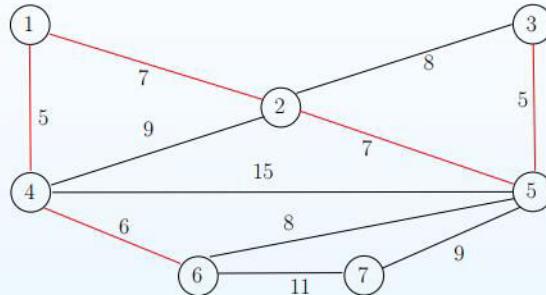


Σχήμα 5: Αλγόριθμος Kruskal - κατασκευή ΕΣΔ

Οι ακμές κατά αύξουσα σειρά βάρους:

$$\begin{aligned} & (1,4) - (3,5) - (4,6) - (1,2) - (2,5) - (2,3) \\ & - (6,5) - (2,4) - (5,7) - (6,7) - (4,5) \end{aligned}$$

Επανάληψη 8

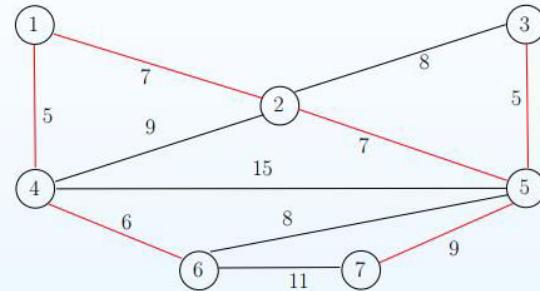


Σχήμα 5: Αλγόριθμος Kruskal - κατασκευή ΕΣΔ

Οι ακμές κατά αύξουσα σειρά βάρους:

$$\begin{aligned} &(1, 4) - (3, 5) - (4, 6) - (1, 2) - (2, 5) - (2, 3) \\ &\quad -(6, 5) - (2, 4) - (5, 7) - (6, 7) - (4, 5) \end{aligned}$$

Επανάληψη 9



Σχήμα 5: Αλγόριθμος Kruskal - κατασκευή ΕΣΔ

Οι ακμές κατά αύξουσα σειρά βάρους:

$$\begin{aligned} &(1, 4) - (3, 5) - (4, 6) - (1, 2) - (2, 5) - (2, 3) \\ &\quad -(6, 5) - (2, 4) - (5, 7) - (6, 7) - (4, 5) \end{aligned}$$

Ορθότητα

Πρόταση 8 Ο Αλγόριθμος του Kruskal παράγει ένα ΕΣΔ T του συνδεδεμένου βεβαρυμένου μη-κατευθυνομένου γραφήματος $G(V, E, w)$

Απόδειξη

- Το T δεν περιέχει κύκλους από κατασκευής
- Το T είναι γεννητορικό υπογράφημα του G (δηλαδή $V(T) = V(G)$ - δες ορισμό γεννητορικού υπογραφήματος). Στην αντίθετη περίπτωση θα υπάρχει κορυφή v η οποία δεν ανήκει στο σύνολο $V(T)$. Όμως από τις προσπιπουσες ακμές στην v αυτή που θα είχε το μικρότερο βάρος θα έπρεπε να προστεθεί σε κάποια επανάληψη στο $E(T)$ χωρίς να δημιουργείται κύκλος αφού $v \notin V(T)$.

Απόδειξη (συνεχ.)

- Το T είναι συνδεδεμένο. Στην αντίθετη περίπτωση θα έχει k συνιστώσες και έστω n_1, \dots, n_k ο αριθμός των κορυφών της κάθε συνιστώσας. Όμως τότε

$$|E(T)| \leq n_1 - 1 + n_2 - 1 + \dots + n_k - 1 = |V| - k < |V| - 1.$$

Επομένως ο αλγόριθμος δεν θα είχε τερματίσει αφού η συνθήκη στη γραμμή 4 θα ήταν αληθής και θα υπήρχαν ακμές με άκρα σε διαφορετικές συνιστώσες η προσθήκη των οπίων δεν θα δημιουργούσε κύκλο σε κάποια επόμενη επανάληψη (αφού το G είναι συνδεδεμένο)

Από τις τρεις παραπάνω παρατηρήσεις συνεπάγεται ότι το T είναι συνδετικό δένδρο του G .

Απόδειξη (συνεχ.) Στη συνέχεια θα δείξουμε ότι το T είναι ένα ΕΣΔ.

Έστω T^* ένα ΕΣΔ του G . Αν $T^* = T$ η απόδειξη ολοκληρώνεται εδώ. Αν $T^* \neq T$ θεωρούμε την ακμή $e \in E(T) \setminus E(T^*)$ που έχει το μικρότερο βάρος από όλες τις ακμές του συνόλου της διαφοράς. Το υπογράφημα $T^* \cup \{e\}$ περιέχει κύκλο C και επειδή το T είναι δένδρο υπάρχει ακμή $f \in E(C) \setminus E(T) \Rightarrow f \in E(T^*) \setminus E(T)$. Παρατηρήστε ότι

$$w(f) \geq w(e), \quad (1)$$

γιατί διαφορετικά ($w(f) < w(e)$) η ακμή f θα είχε εξετασθεί από τον αλγόριθμο σε επανάληψη πριν από την e και άρα θα είχε προστεθεί στο σύνολο $E(T)$ αντί της e .

Απόδειξη (συνεχ.) Το υπογράφημα $T' = T^* - f + e$ είναι συνδετικό δένδρο (Λήμμα 3) και περιέχει μία παραπάνω ακμή κοινή με το T από ότι το T^* . Από (1) έχουμε ότι

$$w(T') \leq w(T^*).$$

Επαναλαμβάνοντας την ίδια διαδικασία μπορούμε να μετατρέψουμε το T^* σε T χωρίς αύξηση του αθροίσματος των συντελεστών των ακμών. Επομένως,

$$w(T) \leq w(T^*),$$

και επειδή T^* είναι ΕΣΔ η παραπάνω σχέση ισχύει σαν ισότητα και άρα και το T είναι ΕΣΔ.

Πολυπλοκότητα

Πρόταση 9 Ο αλγόριθμος του Kruskal εκτελεί $O(m \lg m)$ βήματα, όπου $m = |E|$.

Απόδειξη Ο αλγόριθμος χρειάζεται $m \lg m$ βήματα για να διατάξει τις ακμές κατά αύξουσα σειρά βάρους (π.χ. ταξινόμηση μέσω της μεθόδου mergesort) στη Γραμμή 1. Για κάθε ακμή πρέπει να ελέγχεται αν το υπογράφημα με σύνολο ακμών $E(T) \cup \{e_i\}$ περιέχει κύκλο (Γραμμή 5). Αυτό μπορεί να γίνει σε σταθερό χρόνο (το πολύ δύο συγκρίσεις) ελέγχοντας αν και οι δυο κορυφές της ακμής e_i ανήλουν στο σύνολο $V(T)$.

Ο παραπάνω έλεγχος το πολύ να γίνει m φορές. Άρα συνολικά οι στοιχειώδεις πράξεις των γραμμάων 4-9 είναι τάξης $O(m)$.

06_ΔΥΝΑΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ / 08a_Dynamic Programming Notes

Εισαγωγή

Ο Δυναμικός Προγραμματισμός (ΔΠ) εισήχθη από τον R. Bellman [1] σαν μια μεθοδολογία για την εύρεση λύσης σε προβλήματα βελτιστοποίησης διακριτού τύπου· η έρευνα που ακολουθησε απεκάλυψε πληθώρα εφαρμογών και ανέδειξε τον ΔΠ σαν μία από τις ισχυρότερες υπολογιστικές τεχνικές. Σε ένα τέτοιο πρόβλημα, το σύνολο των λύσεων διαμορφώνεται από διακριτές δομές (π.χ., μονοπάτια σε κάποιο γράφημα, συνεκτικά δένδρα, συμβολοσειρές, κλπ) από τις οποίες αναζητείται η βέλτιστη ως προς κάποιο κριτήριο. Η διαδικασία επίλυσης πραγματοποιείται σε στάδια. Σε κάθε στάδιο επιλύεται μία σειρά από υποπροβλήματα - μικρότερα στιγμιότυπα του αρχικού προβλήματος. Για κάθε υποπρόβλημα εξετάζονται όλες οι πιθανές επιλογές που οδηγούν σε επίλυση και η καλύτερη μαζί με την λύση που παράγει αποθηκεύεται. Σε επόμενο στάδιο η λύση αυτή θα χρησιμοποιηθεί για την επίλυση μεγαλύτερων υποπροβλημάτων. Στο τελευταίο στάδιο η λύση του αρχικού προβλήματος προκύπτει από την σύνθεση των λύσεων κάποιων από τα υποπροβλήματα των προηγουμένων σταδίων.

Από μία άποψη η μεθοδολογία αυτή προσομοιάζει με τη «Διαίρει και Κυρίευε» (ΔΚ) την οποία είδαμε σε προηγούμενο κεφάλαιο: το αρχικό πρόβλημα διασπάται σε μικρότερα που μπορούν να επιλυθούν ευκολότερα και η σύνθεση των λύσεων τους οδηγεί στη λύση του αρχικού προβλήματος. Όμως στην περίπτωση του ΔΠ η διαδικασία είναι πιο σύνθετη: η λύση σε κάθε υποπρόβλημα προκύπτει εξετάζοντας μία σειρά από εναλλακτικές αποφάσεις τα αποτελέσματα των οποίων υπολογίζονται με βάση την αποθηκευμένη πληροφορία που αφορά λύσεις υποπροβλημάτων που έχουν επιλυθεί σε προηγούμενα στάδια. Δηλαδή τα δύο σημεία που ο ΔΠ διαφέρει από τη ΔΚ είναι α) η χρήση μνήμης για την αποθήκευση λύσεων προηγουμένων υποπροβλημάτων και β) η αξιοποίηση τους προκειμένου να εκτιμηθούν τα αποτελέσματα διαφορετικών εναλλακτικών αποφάσεων και να επιλεγεί η καλύτερη που οδηγεί στη λύση του παρόντος υποπροβλήματος.

Μεθοδολογία

Από την προηγούμενη ενότητα είναι εμφανή τα βασικά χαρακτηριστικά της μεθοδολογίας του ΔΠ. Βασική ιδέα αποτελεί η διάσπαση του αρχικού προβλήματος σε μικρότερα προβλήματα τα οποία επειδή μπορεί να είναι αλληλοκαλυπτόμενα επιλύονται μόνο μία φορά - η λύση τους αποθηκεύεται στη μνήμη και στη συνέχεια χρησιμοποιείται όσες φορές χρειάζεται. Περαιτέρω, η σειρά επίλυσης των υποπροβλημάτων πρέπει να διασφαλίζει ότι όταν επιχειρείται να υπολογιστεί η λύση κάποιου από αυτά να υπάρχουν διαθέσιμες (και αποθηκευμένες στη μνήμη) οι λύσεις των υποπροβλημάτων στις οποίες αυτή βασίζεται. Αυτό υποδηλώνει μία ιεράρχηση στα υποπροβλήματα που μπορεί να αναπαρασταθεί σε ένα κατευθυνόμενο γράφημα. Οι κορυφές του γραφήματος αντιστοιχούν σε υποπροβλήματα. Για κάθε ζευγάρι κορυφών που σχετίζονται με υποπροβλήματα στα οποία η λύση του πρώτου χρησιμοποιείται στον υπολογισμό της λύσης του δεύτερου θα υπάρχει κατευθυνόμενη ακμή από την κορυφή του πρώτου προς την κορυφή του δεύτερου. Προφανώς για να μπορεί να επιλυθεί το αρχικό πρόβλημα θα πρέπει το γράφημα να είναι άκυκλο κατευθυνόμενο (*AKG*). Η τοπολογική ταξινόμηση των κορυφών του γραφήματος υποδεικνύει τη σειρά με την οποία θα πρέπει να επιλύονται τα υποπροβλήματα. Για παράδειγμα, στην περίπτωση του Αλγόριθμου 46, το γράφημα των υποπροβλημάτων για την επίλυση του προβλήματος $F(5)$ απεικονίζεται στο Σχήμα 8.1. Η τοπολογική ταξινόμηση των κορυφών στην περίπτωση αυτή είναι εμφανής αφού ακολουθεί την αυξητική φορά του δείκτη i στο συμβολισμό $F(i)$. Γενικότερα επειδή υπάρχει ιεραρχία στη σειρά επίλυσης, ο ΔΠ μπορεί να χαρακτηριστεί

σαν μία αλγορίθμική τεχνική Bottom-Up.

Σε σχέση με την πολυπλοκότητα των αλγορίθμικών σχημάτων που βασίζονται στο ΔΠ, παρατηρούμε ότι καθοριστικό ρόλο παίζει ο αριθμός των υποπροβλημάτων· ο συνολικός αριθμός των SYB που εκτελούνται είναι ίσος με το γινόμενο του αριθμού αυτού επί το πλήθος των SYB που εκτελείται για την επίλυση του κάθε υποπροβλήματος. Στην περίπτωση του υπολογισμού του n -ιοστού όρου της σειράς Fibonacci μέσω του Αλγόριθμου 46, έχουμε $\Theta(n)$ - το πλήθος - υποπροβλήματα, το καθένα από τα οποία λύνεται σε σταθερό χρόνο και άρα η πολυπλοκότητα του αλγόριθμου είναι επίσης $\Theta(n)$.

Τα πέντε στάδια

Η μεθοδολογία του ΔΠ βασίζεται στα ακόλουθα στάδια:

1. Ορισμός υποπροβλημάτων
2. Καθορισμός επιλογών (αποφάσεις)
3. Ορισμός αναδρομικής σχέσης
4. Τοπολογική ταξινόμηση υποπροβλημάτων
5. Επίλυση αρχικού προβλήματος

Στο Στάδιο 1 θα πρέπει να οριστούν τα υποπροβλήματα. Οι ορισμοί περιλαμβάνουν μία λεκτική περιγραφή αλλά και μία συμβολική. Η συμβολική περιγραφή συνήθως γίνεται μέσω δεικτών που ορίζουν *πρόθεμα* (*prefix*), *επίθεμα* (*suffix*) ή μέρος της συμβολοσειράς εισόδου. Στο Στάδιο 2 θα πρέπει για κάθε υποπρόβλημα να καθοριστούν ποιες είναι οι επιλογές που υπάρχουν και που θα οδηγήσουν στη λύση ενός επόμενου υποπροβλήματος. Έτσι φτάνουμε στο Στάδιο 3 όπου μπορούμε πλέον να διατυπώσουμε μία αναδρομική σχέση η οποία συνδέει τις λύσεις των υποπροβλημάτων. Στο Στάδιο 4 αποδεικνύουμε ότι το γράφημα που προκύπτει από την αναδρομική σχέση που συνδέει τα υποπροβλήματα είναι ΑΚΓ. Έτσι προκύπτει τη σειρά που πρέπει αυτά να επιλυθούν - σειρά που προκύπτει από την τοπολογική ταξινόμηση. Έτσι αποθηκεύονται στη μνήμη οι λύσεις των υποπροβλημάτων με σειρά που να τις κάνει διαθέσιμες για την επίλυση των επόμενων υποπροβλημάτων. Στο τελευταίο στάδιο (Στάδιο 5), περιγράφεται το πως επιλύεται το αρχικό πρόβλημα με βάση τις λύσεις των υποπροβλημάτων.

Ας προσπαθήσουμε να εφαρμόσουμε τα παραπάνω στο πρόβλημα εύρεσης του μεγαλύτερου αθροίσματος συνεχόμενων ακεραίων που είναι αποθηκευμένοι στις θέσεις 1 ως n ενός πίνακα A . Το πρόβλημα αυτό παρουσιάστηκε στην Ενότητα 6.2.1 μαζί με έναν αλγόριθμο επίλυσης (Αλγόριθμος 34). Ο αλγόριθμος αυτός υλοποιεί την παραπάνω μεθοδολογία και επομένως αποτελεί έναν αλγόριθμο ΔΠ. Στην περίπτωση αυτή τα παραπάνω στάδια εξειδικεύονται ως εξής.

1. (Ορισμός υποπροβλημάτων) Ορίζουμε ως $S(j)$ το υποπρόβλημα εύρεσης του μεγαλύτερου αθροίσματος συνεχόμενων ακεραίων της υποσειράς $A[1], \dots, A[j]$, όπου $j \in \{1, \dots, n\}$. Δηλαδή ο ορίζοντας σε σχέση με το υποπρόβλημα αυτό περιορίζεται στα υποπροβλήματα που ορίζονται για τιμές μικρότερες-ίσες της τιμής του j . Συμβολίζουμε το σύνολο των προβλημάτων αυτών ως $S(: j)$. Επομένως είναι φανερό ότι έχουμε μία προθεματική περιγραφή των υποπροβλημάτων.
2. (Καθορισμός επιλογών) Οι επιλογές που έχουμε όταν θέλουμε να επιλύσουμε το πρόβλημα $S(j)$ είναι είτε να χρησιμοποιήσουμε τη λύση του υποπροβλήματος $S(j - 1)$ επαυξάνοντας τη σειρά αυτή κατά τον ακέραιο $A[j]$ είτε όχι. Στη

δεύτερη περίπτωση ξεκινάει μία νέα σειρά. Η σειρά αυτή αποτελεί λύση για το υποπρόβλημα $S(j)$ και περιέχει μόνο την τιμή $A[j]$.

3. (Αναδρομική σχέση) Συμβολίζουμε με S_j τη λύση του υποπροβλήματος $S(j)$. Άμεσα από την ανάλυση του προηγούμενου βήματος προκύπτει η αναδρομική σχέση

$$S_j = \begin{cases} \max\{A[j], A[j] + S_{j-1}\}, & j = 2, \dots, n, \\ A[1], & j = 1. \end{cases} \quad (8.1)$$

4. (Τοπολογική ταξινόμηση) Το γράφημα των υποπροβλημάτων έχει κορυφές τα υποπροβλήματα $S(j)$ και ακμές $(S(j-1), S(j))$ για $j = 2, \dots, n$. Δηλαδή είναι ένα κατευθυνόμενο μονοπάτι το οποίο εκκινεί από το $S(1)$ και καταλήγει στο $S(n)$. Άρα οι κορυφές είναι τοπολογικά ταξινομημένες κατά αύξουσα σειρά των τιμών του δείκτη j .

5. (Επίλυση του αρχικού προβλήματος) Η τιμή του μεγαλύτερου αθροίσματος είναι η μεγαλύτερη από τις τιμές S_j . Δηλαδή, η λύση δίνεται από τη παράσταση

$$\max\{S_j : j = 1, \dots, n\} \quad (8.2)$$

την οποία και υπολογίζει ο Αλγόριθμος 34. Παρατηρούμε ότι η πολυπλοκότητα του προκύπτει αν στον αριθμό των SYB που εκτελούνται στην (8.2) το γινόμενο του αριθμού των υποπροβλημάτων επί τον αριθμό των SYB που εκτελούνται για να επιλυθεί το κάθε υποπρόβλημα. Συγκεκριμένα εκτελούνται $\Theta(n)$ SYB για τον υπολογισμό της (8.2), ενώ για την εύρεση λύσης σε κάθε υποπρόβλημα εκτελείται σταθερός αριθμός SYB και υπάρχουν n υποπροβλήματα. Επομένως η συνολική διαδικασία έχει πολυπλοκότητα $\Theta(n)$.

Για να μπορέσουμε να βρούμε ποια είναι η υπακολουθία των συνεχόμενων αριθμών η οποία μεγιστοποιεί το άθροισμα των στοιχείων της θα πρέπει να αποθηκεύσουμε σε κάποια δομή την απόφαση που παίρνουμε σε κάθε βήμα. Για το σκοπό αυτό χρησιμοποιούμε το διάνυσμα p . Το στοιχείο p_j ($j = 1, \dots, n$) έχει τιμή τη θέση του πρώτου στοιχείου της σειράς που έχει άθροισμα S_j . Οι τιμές του διανύσματος δίνονται, για $j = 1, \dots, n$, από τον τύπο:

$$p_j = \begin{cases} p_{j-1}, & \text{av } S_j = A[j] + S_{j-1}, \\ j, & \text{av } S_j = A[j]. \end{cases} \quad (8.3)$$

Η χρήση μίας δομής για την ανάκτηση της λύσης του αρχικού προβλήματος - πέρα από την τιμή της - αποτελεί βασικό προαπαιτούμενο σχεδόν σε κάθε αλγορίθμικό σχήμα ΔΠ. Συνήθως στη δομή αυτή κωδικοποιείται η πληροφορία που αφορά τη βέλτιστη επιλογή που γίνεται για τη λύση του κάθε υποπροβλήματος.

Παράδειγμα

Παράδειγμα 18. Θέλουμε να λύσουμε το πρόβλημα του μεγαλύτερου αθροίσματος υπακολουθίας για τη σειρά

$$A = [-9, \quad 1, \quad -5, \quad 4, \quad 3, \quad -6, \quad 7, \quad 8, \quad -2]$$

Για $j = 1, \dots, 9$, υπολογίζουμε από (8.1), (8.3)

$$\begin{aligned} S_1 &= A[1] = -9, & p_1 &= 1, \\ S_2 &= \max\{A[2], A[2] + S_1\} = \{1, 1 - 9\} = 1, & p_2 &= 2, \\ S_3 &= \max\{A[3], A[3] + S_2\} = \{-5, 1 - 5\} = -4, & p_3 &= p_2 = 2, \\ S_4 &= \max\{A[4], A[4] + S_3\} = \{4, 4 - 4\} = 4, & p_4 &= 4, \\ S_5 &= \max\{A[5], A[5] + S_4\} = \{3, 3 + 4\} = 7, & p_5 &= p_4 = 4, \\ S_6 &= \max\{A[6], A[6] + S_5\} = \{-6, -6 + 7\} = 1, & p_6 &= p_5 = 4, \\ S_7 &= \max\{A[7], A[7] + S_6\} = \{7, 1 + 7\} = 8, & p_7 &= p_6 = 4, \\ S_8 &= \max\{A[8], A[8] + S_7\} = \{8, 8 + 8\} = 16, & p_8 &= p_7 = 4, \\ S_9 &= \max\{A[9], A[9] + S_8\} = \{-2, -2 + 16\} = 14, & p_9 &= p_8 = 4. \end{aligned}$$

Η λύση του προβλήματος δίνεται από την (8.2), ήτοι

$$\max\{S_j : j = 1, \dots, 9\} = \max\{-9, 1, -4, 4, 7, 1, 8, 16, 14\} = 16.$$

Αυτό είναι το άθροισμα της υπακολουθίας η οποία εκκινεί από το στοιχείο που βρίσκεται στη θέση $p_8 = 4$ και τελειώνει με το στοιχείο στη θέση 8. Πράγματι,

$$A[4] + A[5] + A[6] + A[7] + A[8] = 4 + 3 + (-6) + 7 + 8 = 16.$$

Βέλτιστη υποδομή

Ο ΔΠ είναι κατάλληλος για την επίλυση προβλημάτων τα οποία έχουν την ιδιότητα της βέλτιστης υποδομής. Ένα πρόβλημα έχει αυτή την ιδιότητα αν μία βέλτιστη λύση του εμπεριέχει βέλτιστες λύσεις σε υποπροβλήματα. Για παράδειγμα, στο στιγμιότυπο του προβλήματος εύρεσης του μεγαλύτερου αθροίσματος υπακολουθίας που επιλύθηκε στην προηγούμενη ενότητα, η βέλτιστη λύση του αρχικού προβλήματος - λύση του $S(8)$ - εμπεριέχει τη βέλτιστη λύση του υποπροβλήματος $S(7)$ η οποία εμπεριέχει τη βέλτιστη λύση του $S(6)$ η οποία εμπεριέχει τη βέλτιστη λύση του $S(5)$ η οποία εμπεριέχει τη βέλτιστη λύση του $S(4)$.

Επειδή κατασκευάζουμε τη βέλτιστη λύση του αρχικού προβλήματος από τις βέλτιστες λύσεις κάποιων από τα υποπροβλήματα θα πρέπει να φροντίσουμε ώστε αυτά να ανήκουν στο σύνολο των υποπροβλημάτων που επιλύονται.

Στις επόμενες ενότητες θα χρησιμοποιήσουμε ΔΠ για να λύσουμε και άλλα προβλήματα τα οποία παρουσιάζουν την ιδιότητα της βέλτιστης υποδομής.

Εισαγωγή

Ο ΔΠ που παρουσιάστηκε σε προηγούμενο κεφάλαιο αποτελεί μία πολύ δυνατή μεθοδολογία για την επίλυση προβλημάτων διακριτής βέλτιστοποίησης. Όμως σε αρκετές περιπτώσεις ο φόρτος για την επίλυση με τη χρήση της μεθοδολογίας αυτής είναι περιττός. Σε κάποια προβλήματα αρκεί να «χτίσουμε» τη λύση κάνοντας την τοπικά καλύτερη επιλογή. Η ιδέα παραπέμπει σε ένα επαναληπτικό σχήμα που ονομάζεται μέθοδος της απληστίας. Για να περιγράψουμε το σχήμα αυτό θεωρούμε ότι τη λύση ενός προβλήματος βέλτιστοποίησης αποτελεί ένα διακριτό σύνολο στοιχείων.¹ Το σύνολο αυτό διαμορφώνεται μέσα από μία επαναληπτική διαδικασία. Σε μία τυχαία επανάληψη το σύνολο αυτό αποτελεί μία μερική λύση (*partial solution*) του αρχικού προβλήματος. Δεδομένης της μερικής λύσης εξετάζεται μία σειρά από επιλογές σε σχέση με τα στοιχεία που μπορούν να την επαυξήσουν. Όμως δεν είναι κάθε επιλογή νόμιμη· το σύνολο που θα προκύψει από την επαύξηση της μερικής λύσης με κάποιο από τα στοιχεία μπορεί να παραβιάζει τους περιορισμούς του προβλήματος και επομένως καμία λύση δεν μπορεί να κατασκευαστεί με βάση αυτό - πόσο μάλλον η ζητούμενη βέλτιστη. Άρα εξαιρώντας τις μη-νόμιμες επιλογές, ο αλγόριθμος θα πάρει αυτή που στην παρούσα επανάληψη δείχνει καλύτερη - τοπικά βέλτιστη - προκειμένου να επαυξήσει τη μερική λύση. Η διαδικασία ολοκληρώνεται όταν διαπιστωθεί ότι το σύνολο που χτίζεται σταδιακά αποτελεί λύση του αρχικού προβλήματος.

Τα τρία στάδια

Κρίσιμα σημεία στην παραπάνω διαδικασία αποτελούν

- α) ο έλεγχος της νομιμότητας μίας επιλογής,
- β) η συγκριτική αποτίμηση των νομίμων επιλογών,
- γ) το κριτήριο τερματισμού.

Το (α) αποτελεί βασική συνιστώσα ενός άπληστου αλγόριθμου. Συνήθως υλοποιείται από μία ρουτίνα η οποία παίρνει σαν είσοδο το επαυξημένο σύνολο και είτε το αποδέχεται, περίπτωση κατά την οποία δεν παραβιάζει κάποιο περιορισμό του προβλήματος, ή - στην αντίθετη περίπτωση - το απορρίπτει. Η ρουτίνα αυτή συνεισφέρει σε μεγάλο ποσοστό στον υπολογιστικό φόρτο της εκάστοτε επανάληψης. Για το λόγο αυτό συνήθως πρώτα γίνεται η κατάταξη των επιλογών με βάση μία συνάρτηση αποτίμησης (Σημείο (β)) και στη συνέχεια πραγματοποιείται ο έλεγχος της νομιμότητας ξεκινώντας από την καλύτερη. Στη σειρά αυτή, η πρώτη επιλογή που θα οδηγεί σε επαυξημένη λύση χωρίς να παραβιάζονται οι περιορισμοί είναι αυτή που υιοθετείται. Δηλαδή, πραγματοποιείται η επαύξηση της μερικής λύσης με το στοιχείο που αποτελεί την καλύτερη νόμιμη επιλογή και ολοκληρώνεται η επανάληψη. Ακολουθεί έλεγχος του κριτηρίου τερματισμού. Αν αυτό δεν ικανοποιείται, η διαδικασία επαναλαμβάνεται. Στην αντίθετη περίπτωση ο αλγόριθμος ολοκληρώνεται: έξοδος του αλγόριθμου αποτελεί η λύση που σταδιακά κατασκευάστηκε από την καλύτερη νόμιμη, σε κάθε επανάληψη, επιλογή.

Αν και υπάρχουν κάποια προβλήματα στα οποία η μέθοδος της απληστίας παράγει τη βέλτιστη λύση συνήθως δεν συμβαίνει αυτό. Η επιτυχία της εφαρμογής της μεθόδου προϋποθέτει ότι το πρόβλημα έχει την ιδιότητα της βέλτιστης υποδομής αλλά και την ιδιότητα της άπληστης επιλογής: μία βέλτιστη λύση μπορεί να σχηματιστεί από μία σειρά επιλογών που είναι τοπικά βέλτιστες. Στις επόμενες ενότητες θα δούμε παραδείγματα τέτοιων προβλημάτων και θα αναλύσουμε τις ιδιότητες αυτές.

Χαρακτηριστικά της μεθόδου της απληστίας

Η απόδειξη ορθότητας του Αλγόριθμου 49 αποτελεί μία τυπική περίπτωση επαγωγικής απόδειξης ενός άπληστου αλγόριθμου. Ουσιαστικά αυτό που χρειάζεται να κάνουμε για να αποδείξουμε ότι ένας τέτοιος αλγόριθμος παράγει τη βέλτιστη λύση σε ένα πρόβλημα είναι αρχικώς (Βήμα 1) να δείξουμε ότι πάντα υπάρχει μία βέλτιστη λύση η οποία περιέχει το τοπικό βέλτιστο στοιχείο που επιλέγει ο αλγόριθμος (ιδιότητα της άπληστης επιλογής). Στη συνέχεια (Βήμα 2), πρέπει να δείξουμε ότι δεδομένης της άπληστης επιλογής, ο αλγόριθμος απομένει να επιλύσει ένα επιμέρους υποπρόβλημα του οποίου η λύση σε συνδυασμό με την άπληστη επιλογή παράγει τη λύση του αρχικού προβλήματος (βέλτιστη υποδομή). Στην περίπτωση του προβλήματος προγραμματισμού των εργασιών, το Λήμμα 6.1 αποτελεί το Βήμα 1 ενώ το Λήμμα 7 το Βήμα 2 της διαδικασίας απόδειξης.

Μία ισοδύναμη προσέγγιση στην απόδειξη ορθότητας είναι να συγκρίνουμε τα στοιχεία της βέλτιστης λύσης με τα στοιχεία της λύσης που παρήγαγε ο άπληστος αλγόριθμος. Συγκεκριμένα έστω C^* μία βέλτιστη λύση και C η λύση του αλγόριθμου. Αρχικώς θα πρέπει να δείξουμε ότι $|C| = |C^*|$. Στη συνέχεια, θεωρούμε ότι τα στοιχεία των δύο συνόλων είναι διατεταγμένα σε βάση το κριτήριο της άπληστης επιλογής που χρησιμοποιήθηκε. Αν οι δύο λύσεις δεν ταυτίζονται θεωρούμε ότι κάποιο στοιχείο στη διάταξη αυτή είναι το πρώτο στο οποίο τα δύο σύνολα διαφέρουν. Έστω ότι αυτό είναι το i -οστό στη διάταξη στοιχείο. Δηλαδή αν το i -ιοστό στοιχείο του συνόλου C είναι το x και του συνόλου C^* είναι το y έχουμε ότι $x \neq y$. Στη συνέχεια θα πρέπει να αποδείξουμε ότι αν το στοιχείο x αντικαταστήσει το y στο C^* το σύνολο που θα προκύψει θα συνεχίσει να αποτελεί βέλτιστη λύση στο πρόβλημα. Δηλαδή αρκεί να δειχθεί ότι το σύνολο $C^* \cup \{x\} \setminus y$ αποτελεί βέλτιστη λύση. Αν ισχύει αυτό τότε μπορούμε επαγωγικά να εφαρμόσουμε την ίδια διαδικασία για κάθε στοιχείο που τα δύο σύνολα διαφέρουν και να παράγουμε έτσι το σύνολο C το οποίο και αυτό θα είναι βέλτιστο.

Έχοντας σαν παράδειγμα τον προγραμματισμό εργασιών, παρατηρούμε ότι ένας άπληστος αλγόριθμος είναι απλός στο σχεδιασμό, εύκολος στην υλοποίηση του και αποδοτικός σε σχέση με τους πόρους που χρησιμοποιεί. Όμως προκειμένου να παράγει τη βέλτιστη λύση ενός προβλήματος απαιτεί να ισχύει τόσο η ιδιότητα της βέλτιστης υποδομής όσο και της άπληστης επιλογής. Δηλαδή είναι πιο απαιτητικός από το ΔΠ σε σχέση με τις προϋποθέσεις που πρέπει να πληρούνται προκειμένου να παράγει τη βέλτιστη λύση. Από την άλλη βέβαια ο ΔΠ αποτελεί μία μεθοδολογία που είναι πιο δύσκολη στην εφαρμογή της από ότι η μέθοδος της απληστίας.

Γενικότερα, οι δύο μεθοδολογίες διαφέρουν στο έξης σημείο. Ένας αλγόριθμος ΔΠ κάνει μία επιλογή σε κάθε βήμα αλλά η επιλογή αυτή βασίζεται στις λύσεις κάποιων υποπροβλημάτων που έχουν παραχθεί νωρίτερα. Δηλαδή, ένας αλγόριθμος ΔΠ πρώτα επιλύει διάφορα υποπροβλήματα και μετά αποφασίζει με βάση αυτά την καλύτερη επιλογή. Από την άλλη ένας άπληστος αλγόριθμος πρώτα επιλέγει ένα κριτήριο με βάση το οποίο θα κάνει την επιλογή του στα πλαίσια της εκάστοτε «παρούσας κατάστασης». Η παρούσα κατάσταση διαμορφώνεται κάθε φορά με βάση της επιλογές που έχουν γίνει από την εφαρμογή του ίδιου κριτηρίου σε προηγούμενα βήματα. Με την έννοια αυτή η άπληστη επιλογή ακολουθεί μία top-down προσέγγιση στην επίλυση ενός προβλήματος αφού κάνοντας μία τοπικά βέλτιστη επιλογή δημιουργεί ένα επιμέρους υποπρόβλημα το οποίο προσπαθεί να επιλύσει εφαρμόζοντας και πάλι το ίδιο κριτήριο. Όπως είδαμε η επιλογή του κριτηρίου απληστίας είναι κρίσιμη για την επιτυχία της μεθόδου.

Εισαγωγή

Είδαμε ότι

- ο Δυναμικός Προγραμματισμός (ΔΠ) είναι μία πολύ ισχυρή τεχνική για την επίλυσης (κυρίως προβλημάτων βελτιστοποίησης) η οποία βασίζεται σε μία ακολουθία βημάτων (σταδίων) σε καθένα από τα οποία υπάρχει ένα σύνολο επιλογών.

Όμως σε πολλές περιπτώσεις η εφαρμογή του ΔΠ δεν είναι καθόλου απλή. Αρκετά προβλήματα φαίνεται διαισθητικά να επιδέχονται απλούστερης επίλυσης από την εφαρμογή μίας τόσο ισχυρής μεθόδου. Μια αρκετά προφανής στρατηγική είναι αυτή της άπληστης αρχής.

Άπληστη Αρχή: Σε κάθε βήμα κάνε την καλύτερη επιλογή χωρίς να παραβιάζεις τους περιορισμούς του προβλήματος.

Η παραπάνω αρχή βασίζεται στην ιδέα ότι μία τοπικά βέλτιστη επιλογή θα οδηγήσει σε μία καθολικά βέλτιστη λύση.

Εφαρμογές

Στα προβλήματα που έχουμε δει μέχρι τώρα σαν άπληστοι μπορούν να θεωρηθούν οι αλγόριθμοι

Εύρεση ΕΣΔ - αλγόριθμος Kruskal: σε κάθε βήμα επιλέγεται η ακμή με το μικρότερο βάρος (καλύτερη επιλογή) η οποία αν προστεθεί στο προς-κατασκευή δένδρο δεν δημιουργεί κύκλο (δεν παραβιάζει τους περιορισμούς)

Εύρεση Δένδρου Ελάχιστων Διαδρομών - αλγόριθμος Dijkstra: σε κάθε επανάληψη το δένδρο των ελάχιστων διαδρομών επεκτείνεται με την προσθήκη της ελαφρύτερης ακμής αφού επιλέγεται για μονιμοποίηση η μη-μόνιμη κορυφή που βρίσκεται εγγύτερα προς την αφετηριακή κορυφή.

Στη συνέχεια θα δούμε ένα ακόμα πρόβλημα για το οποίο η άπληστη αρχή είναι ο ενδεδειγμένος τρόπος επίλυσης.

Προγραμματισμός Εργασιών

Έστω ένα σύνολο n εργασιών $T = \{1, \dots, n\}$ τέτοια ώστε κάθε εργασία $t \in T$ να χαρακτηρίζεται από ένα χρόνο έναρξης s_t και ένα χρόνο περαιώσης f_t . Όλες οι εργασίες εκτελούνται σε μία μηχανή, μία εργασία κάθε χρονική στιγμή και κατά τρόπο ώστε αν έχει ξεκινήσει η επεξεργασία της εργασίας θ πρέπει να ολοκληρωθεί (δηλαδή δεν μπορεί να μείνει στη μέση και να ξεκινήσει μία επόμενη εργασία).

Ζητείται να βρεθεί το μεγαλύτερο σύνολο εργασιών T' που μπορούν να εκτελεστούν στη μηχανή.

Παράδειγμα 1 Θεωρούμε 5 εργασίες με χρόνους έναρξης-περαιώσης όπως απεικονίζονται στον Πίνακα 1

<i>Εργασ. j</i>	1	2	3	4	5
<i>Χρόνοι Εναρξ. s_j</i>	2	1	3	6	10
<i>Χρόνοι Περ. f_j</i>	4	8	9	10	11

Πίνακας 1: Ζητούμενο σύνολο: $T' = \{1, 4, 5\}$

Δυναμικός Προγραμματισμός

Ορίζουμε το σύνολο των εργασιών $T_{i,j}$ οι οποίες μπορούν να εκτελεστούν μετά τη λήξη της εργασίας i και πριν την έναρξη της εργασίας j , ήτοι,

$$T_{i,j} = \{k \in T : f_i \leq s_k < f_k \leq s_j\}.$$

Επιπλέον θεωρούμε δύο ψευδοεργασίες $0, n+1$: η πρώτη περαιώνεται στο χρόνο 0 και η δεύτερη εκκινεί το χρόνο ∞ . Παρατηρούμε ότι ο ορισμός των $T_{i,j}$ βγάζει περισσότερο νόημα αν θεωρήσουμε ότι οι εργασίες είναι διατεταγμένες σε αύξουσα σειρά ως προς το χρόνο περαιώσης τους, δηλαδή,

$$f_0 \leq f_1 \leq f_2 \leq \dots \leq f_{n-1} \leq f_n < f_{n+1}. \quad (1)$$

Θεωρώντας τη δεικτοδότηση των εργασιών ώστε να ισχύει η (1) είναι προφανές ότι αν $j > i$ ή $j = i + 1$, έχουμε $T_{i,j} = \emptyset$.

Κάθε σύνολο $T_{i,j}$ αντιστοιχεί σε ένα (υπο)πρόβλημα:

Ορισμός Υποπροβλήματος

Θέλουμε να βρούμε το μεγαλύτερο σύνολο των εργασιών που ανήκουν στο σύνολο $T_{i,j}$ (δηλαδή των εργασιών που εκκινούν μετά την λήξη της εργασίας i και πριν την έναρξη της j) που μπορούν να εκτελεστούν στη μηχανή (είναι αμοιβαία συμβατές μεταξύ τους). Θα αναφερόμαστε σε ένα τέτοιο σύνολο σαν $A_{i,j}$

Το αρχικό πρόβλημα ορίζεται για το σύνολο $T_{0,n+1}$ και αναζητούμε το σύνολο $A_{0,n+1}$.

Αν εξετάσουμε πιο προσεκτικά ένα μη-κενό σύνολο $T_{i,j}$. Έστω ότι $k \in A_{i,j} \subseteq T_{i,j}$. Τότε προφανώς ισχύει

$$A_{i,j} = A_{i,k} \cup \{k\} \cup A_{k,j}. \quad (1)$$

Δηλαδή αν στη λύση του υποπροβλήματος $T_{i,j}$ υπάρχει η εργασία k τότε η λύση αυτή μπορεί να κατασκευαστεί από τη λύση του υποπροβλήματος $T_{i,k}$, του $T_{k,j}$ και την προσθήκη της k . Αυτό σημαίνει ότι το πρόβλημα παρουσιάζει την ιδιότητα της βέλτιστης υποδομής και επομένως μπορούμε να εφαρμόσουμε την τεχνική του ΔΠ.

Για το $|A_{i,j}|$ ισχύει η αναδρομική σχέση

$$|A_{i,j}| = \begin{cases} \max_{i < k < j, k \in A_{i,j}} \{|A_{i,k}| + |A_{k,j}| + 1\}, & A_{i,j} \neq \emptyset, \\ 0, & A_{i,j} = \emptyset. \end{cases} \quad (1)$$

Απληστία

Έστω

$$k = \operatorname{argmin}\{f_t : t \in T_{i,j}\} \quad (2)$$

Λήμμα 2

1. Η δραστηριότητα k περιλαμβάνεται στο $A_{i,j}$
2. $T_{i,k} = \emptyset$ και επομένως μόνο το σύνολο (υποπρόβλημα) $T_{k,j}$ μπορεί να είναι διάφορο του κενού

Απόδειξη

1. Αν η $A_{i,j}$ περιέχει εργασία με συντομότερο χρόνο περαίωσης που είναι μεγαλύτερος από την k τότε μπορεί να αφαιρεθεί από το σύνολο αυτό και να προστεθεί η k χωρίς να εμφανίζονται συγκρούσεις καθώς η k τελειώνει πρώτη από όλες τις υπόλοιπες εργασίες του συνόλου $A_{i,j}$.
2. Προφανές.

Λήμμα 3 Το πρόβλημα διαδέτει την ιδιότητα της βέλτιστης υποδομής.

Απόδειξη Έστω ότι όποια βέλτιστη λύση περιέχει την άπληστη επιλογή (όπως αυτή ορίζεται από τη (2)) δεν περιέχει τη βέλτιστη υποδομή. Έστω S μία τέτοια λύση. Προφανώς το σύνολο εργασιών $S' = S \setminus \{k\}$ δεν αποτελεί βέλτιστη λύση για το υποπρόβλημα που αφορά το σύνολο των εργασιών που είναι συμβατές με την k (αφού δεν ισχύει η βέλτιστη υποδομή). Έστω S'' μία βέλτιστη λύση για το υποπρόβλημα αυτό. Άρα

$$|S''| > |S'| \quad (3)$$

Το σύνολο εργασιών

$$S''' = S'' \cup \{k\} \quad (4)$$

αποτελεί παραδεκτή λύση για το πρόβλημα. Όμως από (3),(4) έχουμε

$$|S'''| = |S''| + 1 > |S'| + 1 = |S|$$

(αντίφαση αφού S αποτελεί βέλτιστη λύση του προβλήματος)

Τα Λήμματα 2, 3 αποδεικνύουν ότι ισχύει η αναδρομική σχέση

$$A_{i,j} = \{k\} \cup A_{k,j},$$

όπου το k δίνεται από τον τύπο (2).

Θέτοντας $i = 0, j = n + 1$ στην παραπάνω σχέση έχουμε τη λύση στο πρόβλημα μας με τη μορφή του άπληστου αλγόριθμου, ήτοι,

Require: $f_1 \leq f_2 \leq \dots \leq f_{n-1} \leq f_n$

- 1: $A \leftarrow \{1\}; i \leftarrow 1;$
- 2: **for** $m \leftarrow 2; m \leq n; m++$ **do**
- 3: **if** $s_m \geq f_i$ **then**
- 4: $A \leftarrow A \cup \{m\};$
- 5: $i \leftarrow m;$
- 6: **end if**
- 7: **end for**

Παρατηρήσεις

Το γεγονός ότι η άπληστη αρχή βρίσκει τη λύση επιλέγοντας διαδοχικά την εργασία με το μικρότερο χρόνο περαίωσης δεν σημαίνει ότι το αυτό συμβαίνει αν εφαρμόζουμε την άπληστη αρχή σε κάποιο άλλο κριτήριο. Συγκεκριμένα τα παρακάτω κριτήρια ΔΕΝ οδηγούν με τη χρήση της άπληστης επιλογής στη βέλτιστη λύση

- επιλογή δραστηριότητας που ξεκινάει νωρίτερα
- επιλογή δραστηριότητας με την μικρότερη διάρκεια
- επιλογή δραστηριότητας με βάση τη μικρότερη μη-συμβατότητα σε αριθμό δραστηριοτήτων

Μεθοδολογία

Βήματα

Ένας άπληστος αλγόριθμος προσδιορίζει μια λύση σε ένα πρόβλημα μέσω διαδοχικών επιλογών. Σε κάθε ‘σημείο απόφασης’ η επιλογή που προκρίνεται είναι η καλύτερη στη δεδομένη στιγμή με βάση κάποιο κριτήριο. Είδαμε ότι η επιλογή του κριτηρίου είναι κρίσιμη για την επιτυχία του αλγόριθμου. Για να αποδείξουμε ότι ένας άπληστος αλγόριθμος βρίσκει την βέλτιστη λύση θα πρέπει να δείξουμε

1. άπληστη επιλογή: τουλάχιστον μία βέλτιστη λύση περιέχει την άπληστη επιλογή
2. βέλτιστη υποδομή: μία βέλτιστη λύση μπορεί να χτιστεί με βάση την άπληστη επιλογή και τη βέλτιστη λύση για το εναπομείναν υποπρόβλημα (το υποπρόβλημα που προκύπτει αφού έχει γίνει η άπληστη επιλογή)

Ακριβώς τα βήματα αυτά ακολουθήθηκαν στο πρόβλημα προγραμματισμού εργασιών. Στη συνέχεια θα δούμε μία ακόμα εφαρμογή

Κώδικες Συμπίεσης

Πρόβλημα

Τυπική Κωδικοποίηση Χαρακτήρων

Σε ένα τυπικό σχήμα κωδικοποίησης (π.χ. ascii) κάθε χαρακτήρας αποθηκεύεται στον ίδιο αριθμό bits.

Πρόβλημα

Μπορούμε να βρούμε ένα σχήμα κωδικοποίησης το οποίο να χρησιμοποιεί λιγότερα bits για χαρακτήρες οι οποίοι εμφανίζονται πιο συχνά σε μία δομή (π.χ. ένα αρχείο) και επομένως να χρησιμοποιούνται λιγότερα συνολικά bits για την αποθήκευση της δομής αυτής.

Απροθηματικοί Κώδικες

Αν διαφορετικοί χαρακτήρες χρησιμοποιούν διαφορετικό αριθμό bits πως γίνεται η συσχέτιση αυτή :

Απάντηση

Με τη χρήση απροθηματικής κωδικοποίησης: κάθε χαρακτήρας απεικονίζεται σαν μία συμβολοσειρά από bits έτσι ώστε καμία τέτοια σειρά να είναι αρχική υποσυμβολοσειρά κάποιας άλλης συμβολοσειράς της κωδικοποίησης. Με τον τρόπο αυτό η αποκωδικοποίηση γίνεται με μοναδικό τρόπο.

Παράδειγμα 4 Οι συχνότητες με τις οποίες εμφανίζονται κάποιοι χαρακτήρες σε ένα αρχείο απεικονίζονται στον Πίνακα 2

	α	β	γ	δ	ε	ζ
Συχνότητα	344	96	12	16	9	5
Σταθ. Μήκος	000	001	010	011	100	101
Μετ. Μήκος	0	101	100	111	1101	1100

a.gr 11 / 18
Πίνακας 2: Απροθηματική Κωδ. Σταθ. και Μετ. Μήκους

Παράδειγμα

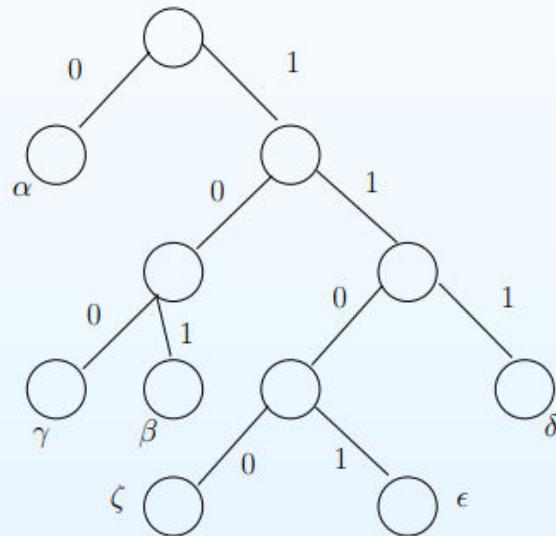
	α	β	γ	δ	ε	ζ
Συχνότητα	344	96	12	16	9	5
Σταθ. Μήκος	000	001	010	011	100	101
Μετ. Μήκος	0	101	100	111	1101	1100

Παρακάτω απεικονίζονται, σαν παράδειγμα, συμβολοσειρές όπως κωδικοποιούνται στο σχήμα σταθερού και μεταβλητού μήκους.

Συμβολοσειρά	Σταθ. Μήκος	Μετ. Μήκος
$\alpha\beta\gamma$	000001010	0101100
$\varepsilon\zeta\zeta$	101100100	110011011101
$\alpha\alpha\alpha$	000000001000	001010

Παρατηρούμε ότι και στις δύο περιπτώσεις η αποκωδικοποίηση των συμβολοσειρών γίνεται με μοναδικό τρόπο

Μπορούμε να απικονίσουμε την απροθηματική κωδικοποίηση του Πίνακα 2 σαν ένα δυαδικό δένδρο



Σχήμα 1: Δυαδικό δένδρο Κωδ. Μετ. Μήκους

- αβααγ
- 11010010111000100
- αβααγ \Rightarrow 010100100
- 11010010111000100 \Rightarrow εααβζαγ

	α	β	γ	δ	ε	ζ
Συχνότητα	344	96	12	16	9	5
Σταθ. Μήκος	000	001	010	011	100	101
Μετ. Μήκος	0	101	100	111	1101	1100

Ο αριθμός των bits για καθένα από τα δύο σχήματα κωδικοποίησης είναι:

- Σταθ. Μήκος:

$$3 \cdot (344 + 96 + 12 + 16 + 9 + 5) = 3 \cdot 482 = 1446$$
- Μετ. Μήκος:

$$1 \cdot 344 + 3 \cdot 96 + 3 \cdot 12 + 3 \cdot 16 + 4 \cdot 9 + 4 \cdot 5 = 772$$

Κόστος Δένδρου

Όπως είδαμε και στο παράδειγμα, το δένδρο μίας απροθηματικής κωδικοποίησης για ένα συγκεκριμένο αρχείο παράγει ένα κωδικοποιημένο αρχείο με συγκεκριμένο αριθμό bits. Ο αριθμός αυτός ονομάζεται και κόστος του δένδρου για το συγκεκριμένο αρχείο και υπολογίζεται από τον τύπο

$$B(T) = \sum_{c \in T} f(c)d_T(c),$$

όπου $f(c)$ είναι η συχνότητα εμφάνισης του χαρακτήρα c στο αρχείο και $d_T(c)$ το βάθος (αριθμός ακμών του μονοπατιού από την ρίζα του δένδρου μέχρι την κορυφή) του χαρακτήρα c στο δένδρο T .

Βέλτιστο Δένδρο

Η κωδικοποίηση Huffman κατασκευάζει ένα βέλτιστο δένδρο - επιλύει το πρόβλημα $\min_T B(T)$. Για να περιγράψουμε την κατασκευή αυτή χρησιμοποιούμε τους ακόλουθους συμβολισμούς

- C : αρχείο (αλφάθητο) όπου κάθε χαρακτήρας c εμφανίζεται $f(c)$ φορές
- T_c : δένδρο που αποτελείται μόνο από την κορυφή c
- \mathcal{T} : σύνολο δένδρων που κατασκευάζει ο αλγόριθμος Huffman. Στο τέλος της εκτέλεσης το σύνολο αυτό περιέχει μόνο το βέλτιστο δένδρο.
- $T = T_1 \bigoplus T_2$: το δένδρο T που προκύπτει από την υπαγωγή των ριζών των δένδρων T_1, T_2 υπό μίας κοινής ρίζας. Αν t, t_1, t_2 οι ρίζες των δένδρων T, T_1, T_2 , αντίστοιχα, τότε $f(t) = f(t_1) + f(t_2)$

Ο αλγόριθμος (κατασκευής του δένδρου) Huffman δίνεται παρακάτω.

- 1: $\mathcal{T} \leftarrow \emptyset$
- 2: **for** $c \in C$ **do**
- 3: $\mathcal{T} \leftarrow \mathcal{T} \cup \{T_c\}$
- 4: **end for**
- 5: **while** $|\mathcal{T}| > 1$ **do**
- 6: $T_1 \leftarrow \text{argmin}\{B(T) : T \in \mathcal{T}\}$
- 7: $\mathcal{T} \leftarrow \mathcal{T} - \{T_1\}$
- 8: $T_2 \leftarrow \text{argmin}\{B(T) : T \in \mathcal{T}\}$
- 9: $\mathcal{T} \leftarrow \mathcal{T} - \{T_2\}$
- 10: $T \leftarrow T_1 \bigoplus T_2$
- 11: $\mathcal{T} \leftarrow \mathcal{T} \cup \{T\}$
- 12: **end while**

Παράδειγμα

Θεωρούμε τους χαρακτήρες με τις συχνότητες που απεικονίζονται στον Πίνακα 3.

Γράμμα	a	b	c	d	e	f
Συχνότητες	3	7	40	20	15	13

Πίνακας 3: Παράδειγμα Κατασκευής δένδρου Huffman

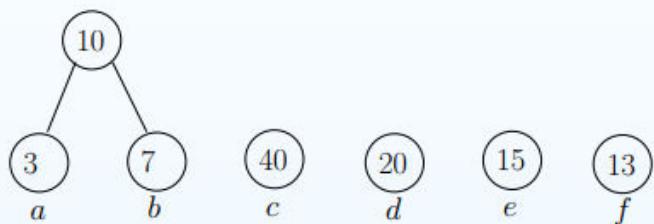
Για να κατασκευάσουμε το δένδρο Huffman σύμφωνα με τον παραπάνω αλγόριθμο ακολουθούμε τα εξής βήματα.

Επανάληψη 1



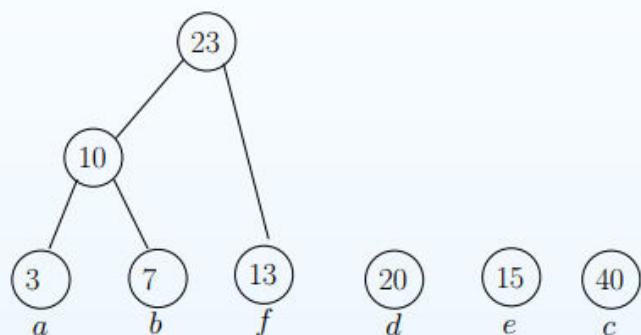
$$\mathcal{T} = \{T_a, T_b, T_c, T_d, T_e, T_f\}$$

Επανάληψη 2



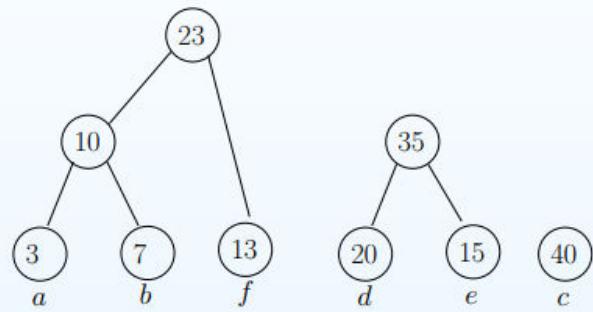
$$\mathcal{T} = \{T_{ab}, T_c, T_d, T_e, T_f\}$$

Επανάληψη 3



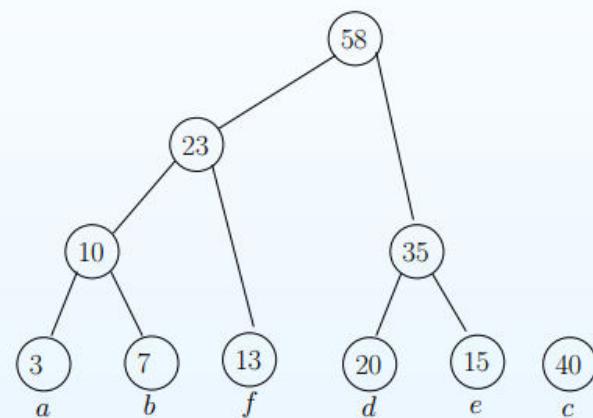
$$\mathcal{T} = \{T_{abf}, T_c, T_d, T_e\}$$

Επανάληψη 4



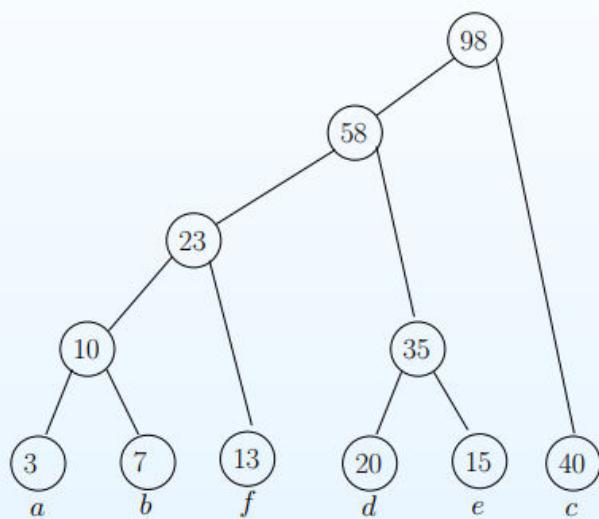
$$\mathcal{T} = \{T_{abf}, T_c, T_{de}\}$$

Επανάληψη 5

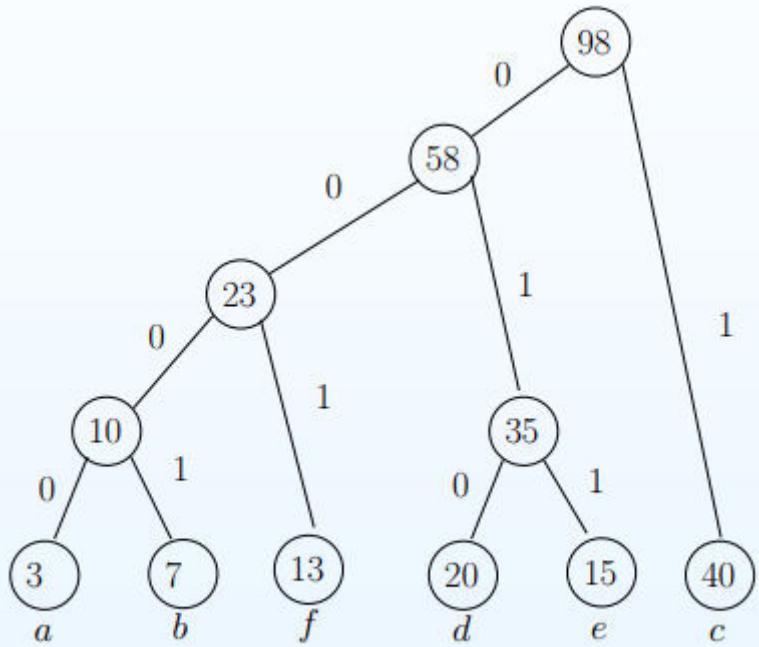


$$\mathcal{T} = \{T_{abfde}, T_c, \}$$

Επανάληψη 6



$$\mathcal{T} = \{T_{abfdec}\}$$



Κωδικοποίηση Huffman

Ορθότητα

Για να δείξουμε ότι η παραπάνω μέθοδος παράγει βέλτιστο δένδρο αρκεί να δείξουμε ότι ισχύουν

- άπληστη επιλογή
- βέλτιστη υποδομή

Άπληστη Επιλογή

Έστω T ένα βέλτιστο δένδρο με x, y δύο χαρακτήρες οι οποίοι βρίσκονται χαμηλότερα στο T - η απόσταση του καθενός από τη ρίζα είναι η μεγαλύτερη δυνατή και η κωδικοποίηση τους διαφέρει στο τελευταίο ψηφίο. Έστω a, b δύο χαρακτήρες με τη μικρότερη συχνότητα.

Εναλλάσουμε το a με το x και το b με το y παράγοντας έτσι το δένδρο T' .

Έχουμε:

$$\begin{aligned}
 B(T) - B(T') &= \sum_{c \in T} f(c) \cdot d_T(c) - \sum_{c \in T'} f(c) \cdot d_{T'}(c) \\
 &= f(a)(d_T(a) - d_{T'}(a)) + f(b)(d_T(b) - d_{T'}(b)) \\
 &\quad + f(x)(d_T(x) - d_{T'}(x)) + f(y)(d_T(y) - d_{T'}(y)) \\
 &= f(a)(d_T(a) - d_{T'}(a)) + f(x)(d_T(x) - d_{T'}(x)) \\
 &\quad + f(b)(d_T(b) - d_{T'}(b)) + f(y)(d_T(y) - d_{T'}(y)).
 \end{aligned}$$

Επειδή

$$d_T(x) - d_{T'}(x) = -(d_T(a) - d_{T'}(a))$$

$$d_T(y) - d_{T'}(y) = -(d_T(b) - d_{T'}(b))$$

η παραπάνω εξίσωση γίνεται

$$\begin{aligned} B(T) - B(T') &= (f(a) - f(x))(d_T(a) - d_{T'}(a)) \\ &\quad + (f(b) - f(y))(d_T(b) - d_{T'}(b)) \end{aligned}$$

Επειδή,

$$f(a) \leq f(x),$$

$$f(b) \leq f(y),$$

$$d_T(a) \leq d_{T'}(a),$$

$$d_T(b) \leq d_{T'}(b),$$

η προηγούμενη σχέση δίνει

$$B(T) - B(T') \geq 0.$$

Επομένως αφού $B(T) \geq B(T')$ αν το T είναι βέλτιστο δένδρο το ίδιο ισχύει και για το T' .

Βέλτιστη Υποδομή

Έστω C ένα αλφάβητο και x, y δύο χαρακτήρες του με τις μικρότερες συχνότητες. Ορίζουμε $C' = C - \{x, y\} \cup \{z\}$, όπου ο νέος χαρακτήρας z έχει την ιδιότητα ότι $f(z) = f(x) + f(y)$. Στη συνέχεια ορίζουμε T' ένα βέλτιστο απροθηματικό δένδρο για το αλφάβητο C' και T το δένδρο που προκύπτει από το T' αν αντικαταστήσουμε την κορυφή φύλλο z με μία εσωτερική κορυφή και δύο απογόνους που αντιστοιχούν στους χαρακτήρες x, y αντίστοιχα.

Θα δείξουμε ότι το δένδρο T αποτελεί ένα βέλτιστο απροθηματικό δένδρο για το αλφάβητο C .

Παρατηρούμε ότι

$$f(c)d_T(c) = f(c)d_{T'}(c), \forall c \in C - \{x, y\} \Rightarrow$$
$$\sum_{c \in C - \{x, y\}} f(c)d_T(c) = \sum_{c \in C - \{x, y\}} f(c)d_{T'}(c)$$

Επίσης

$$f(x)d_T(x) + f(y)d_T(y) = (f(x) + f(y))(d_{T'}(z) + 1)$$
$$= f(z)d_{T'}(z) + f(x) + f(y). \quad (3)$$

Προσθέτοντας κατά μέλη την (3) στα αθροίσματα της προηγούμενης ισότητας, παίρνουμε

$$B(T) = B(T') + f(x) + f(y) \Rightarrow B(T') = B(T) - f(x) - f(y).$$

Έστω ότι το T δεν είναι βέλτιστο και επομένως υπάρχει ένα δένδρο T'' με $B(T'') < B(T)$. Αν αντικαταστήσουμε την κοινή πατρική κορυφή των x, y στο T'' από μία κορυφή φύλλο z με $f(z) = f(x) + f(y)$ παίρνουμε το δένδρο T''' . Έχουμε

$$B(T''') = B(T'') - f(x) - f(y)$$
$$< B(T) - f(x) - f(y) = B(T')$$

και άρα το δένδρο T' δεν είναι βέλτιστο για το C' (αντίφαση).

Εισαγωγή

Κατηγοριοποίηση προβλημάτων

Μια βασική κατηγοριοποίηση των προβλημάτων αποτελούν οι κλάσεις R, EXP, P. Οι κλάσεις αυτές ορίζονται απόπως ως εξής.

P : το σύνολο των προβλημάτων τα οποία επιλύονται σε (το-πολύ) πολυωνυμικό χρόνο, δηλαδή σε χρόνο $O(n^c)$,

EXP : το σύνολο των προβλημάτων τα οποία επιλύονται σε (το-πολύ) εκθετικό χρόνο, δηλαδή σε χρόνο $O(2^{n^c})$,

R : το σύνολο των προβλημάτων τα οποία λύνονται σε πεπερασμένο χρόνο.

Τα περισσότερα από τα προβλήματα με τα οποία ασχοληθήκαμε στα προηγούμενα κεφάλαια ανήκουν στην τάξη P: είναι τα προβλήματα για τα οποία παρουσιάστηκαν πολυωνυμικοί αλγόριθμοι επίλυσης. Προβλήματα αυτής της κατηγορίας είναι σχεδόν κάθε πρόβλημα που παρουσιάσαμε στα προηγούμενα κεφάλαια: για παράδειγμα, το πρόβλημα εύρεσης των συντομότερων μονοπατιών από μία κορυφή προς κάθε άλλη σε ένα εμβαρές κατευθυνόμενο γράφημα, το πρόβλημα εύρεσης ενός ελάχιστου συνεκτικού δένδρου, το πρόβλημα του υπολογισμού ενός γινομένου πινάκων με το μικρότερο αριθμό πολλαπλασιασμών κλπ.

Προβλήματα που ανήκουν στην κατηγορία EXP είναι συνήθως προβλήματα που αφορούν παίγνια. Θεωρούμε για παράδειγμα το γενικευμένο πρόβλημα των σκακιού: δεδομένης μίας διάταξης λευκών και μαύρων κομματιών πάνω σε μία σκακιέρα διάστασης $n \times n$, μπορεί να κερδίσει ο παίκτης με τα λευκά κομμάτια (αν παίζει πρώτος); Το πρόβλημα αυτό έχει αποδειχθεί ότι ανήκει στην κλάση [7] - το ίδιο ισχύει και για το αντίστοιχο πρόβλημα που αφορά την ντάμα [13].

Μη-υπολογίσιμα προβλήματα

Από τους παραπάνω ορισμούς είναι προφανής η ιεραρχία $P \subseteq EXP \subseteq R$.¹ Όμως τι συμβαίνει με τα προβλήματα τα οποία είναι έξω από την τάξη R; Τα προβλήματα αυτά είναι μη-υπολογίσιμα: δεν υπάρχει αλγόριθμος ο οποίος να μπορεί να επιλύσει κάθε στιγμιότυπο ενός τέτοιου προβλήματος σε πεπερασμένο χρόνο. Ένα τέτοιο πρόβλημα είναι αυτό του τερματισμού ενός αλγόριθμου: δοθέντος ενός αλγόριθμου και ενός συνόλου δεδομένων που αποτελούν είσοδο στον αλγόριθμο, θα ολοκληρωθεί η εκτέλεση του προγράμματος με τη συγκεκριμένη είσοδο σε πεπερασμένο χρόνο; Είναι γνωστό ότι το πρόβλημα αυτό είναι εκτός των ορίων του υπολογισμού [8, Κεφάλαιο 9]. Δηλαδή δεν μπορεί να υπάρξει αλγόριθμος που δέχεται σαν είσοδο έναν **οποιοδήποτε αλγόριθμο M** μαζί με ένα σύνολο δεδομένων εισόδου του και να αποφαίνεται αν ο αλγόριθμος M τερματίζει όταν εκτελεστεί πάνω στο σύνολο αυτό.

Προβλήματα απόφασης

Το πρόβλημα του τερματισμού είναι ένα πρόβλημα απόφασης (*decision problem*)· υπάρχουν δυο πιθανές απαντήσεις:

1. ΝΑΙ αν το πρόγραμμα τερματίζει,
2. ΟΧΙ αν δεν τερματίζει.

Τυπικά οι προαναφερθείσες κλάσεις προβλημάτων αφορούν προβλήματα απόφασης, δηλαδή προβλήματα στα οποία το σύνολο των λύσεων είναι το {ΝΑΙ, ΟΧΙ}. Ανάλογα με τη λύση, τα στιγμιότυπα ενός τέτοιου προβλήματος χωρίζονται σε ΝΑΙ-στιγμιότυπα και σε ΟΧΙ-στιγμιότυπα.

Προβλήματα απόφασης θα μας απασχολήσουν από εδώ και στο εξής. Η περιγραφή ενός τέτοιου προβλήματος έχει δύο μέρη. Στο μέρος «**Στιγμιότυπο:**» απαριθμούνται τα δεδομένα του προβλήματος ενώ στο μέρος «**Ερώτηση:**» αφορά την ερώτηση της οποίας η απάντηση είναι είτε ΝΑΙ ή ΟΧΙ (λύση του προβλήματος). Κάτω από αυτή τη σύμβαση το πρόβλημα του τερματισμού γράφεται ως

HALT

Στιγμιότυπο: Αλγόριθμος M , δεδομένα κωδικοποιημένα σε συμβολοσειρά w .

Ερώτηση: Τερματίζει η εκτέλεση του M με είσοδο το w ;

Δυστυχώς τα περισσότερα προβλήματα απόφασης δεν ανήκουν στο R . Αυτό μπορεί να αποδειχθεί με τον εξής τρόπο. Κάθε αλγόριθμος μπορεί να θεωρηθεί σαν συμβολοσειρά πεπερασμένου μήκους - φανταστείτε την υλοποίηση ενός αλγόριθμου ως πρόγραμμα σε οποιαδήποτε γλώσσα προγραμματισμού. Περαιτέρω, η συμβολοσειρά αυτή μπορεί να θεωρηθεί δυαδική - θυμηθείτε ότι το εκτελέσμα αρχείο ενός προγράμματος είναι μία πεπερασμένη συμβολοσειρά από σύμβολα του συνόλου {0, 1}. Άρα κάθε αλγόριθμος έχει σε μία αναπαράσταση στο δυαδικό σύστημα και επομένως μπορούμε να τον αντιστοιχήσουμε στο σύνολο των φυσικών αριθμών \mathbb{N} . Από την άλλη ένα πρόβλημα απόφασης μπορεί να θεωρηθεί σαν μία συνάρτηση που απεικονίζει κάθε στιγμιότυπο σε ένα στοιχείο από το σύνολο {ΝΑΙ = 1, ΟΧΙ = 0}. Άρα αν το πρόβλημα περιέχει άπειρο αριθμό στιγμιοτύπων και κάθε ένα απεικονίζεται σε ένα στοιχείο από το {0, 1}, τότε το πρόβλημα περιγράφεται από μία δυαδική συμβολοσειρά απείρου μήκους. Κάθε τέτοια συμβολοσειρά μπορεί να αναπαρασταθεί από έναν πραγματικό αριθμό στο διάστημα $(0, 1) \subset \mathbb{R}$ - θεωρήστε ότι τα δυαδικά ψηφία έπονται μίας υποδιαστολής και επομένως η συμβολοσειρά (απείρου μήκους) εκφράζει έναν πραγματικό στο διάστημα $(0, 1)$. Όμως είναι γνωστό ότι το σύνολο των τιμών που εμπεριέχονται στο διάστημα αυτό είναι άπειρο και μη-αριθμήσιμο ενώ το σύνολο \mathbb{N} είναι άπειρο αλλά αριθμήσιμο και άρα υπάρχουν πολύ λιγότεροι αλγόριθμοι από προβλήματα απόφασης.

Παρόλο που οι κλάσεις των προβλημάτων P , EXP , R τυπικά αφορούν προβλήματα απόφασης, είναι κοινή πρακτική να κατατάσσουμε και προβλήματα άλλων τύπων στις τάξεις αυτές. Για παράδειγμα, το πρόβλημα εύρεσης ενός ελάχιστου συνεκτικού δένδρου σε ένα βεβαρυμένο συνδεδεμένο γράφημα, ονομαστικά $E\Delta$, είναι ένα πρόβλημα βελτιστοποίησης για το οποίο υπάρχει πολυωνυμικός αλγόριθμος επίλυσης. Δηλώνουμε την ιδιότητα αυτή κατηγοριοποιώντας το (ίσως καταχρηστικά) στην κλάση P . Αυτό δεν γίνεται αδικαιολόγητα· μπορούμε να μετατρέψουμε το πρόβλημα αυτό σε ένα πρόβλημα απόφασης με βάση το οποίο μπορούμε να το επιλύσουμε. Το πρόβλημα απόφασης

Παράδειγμα

πων στις τάξεις αυτές. Για παράδειγμα, το πρόβλημα εύρεσης ενός ελάχιστου συνεκτικού δένδρου σε ένα βεβαρυμένο συνδεδεμένο γράφημα, ονομαστικά $E\Delta$, είναι ένα πρόβλημα βελτιστοποίησης για το οποίο υπάρχει πολυωνυμικός αλγόριθμος επίλυσης. Δηλώνουμε την ιδιότητα αυτή κατηγοριοποιώντας το (ίσως καταχρηστικά) στην κλάση P . Αυτό δεν γίνεται αδικαιολόγητα· μπορούμε να μετατρέψουμε το πρόβλημα αυτό σε ένα πρόβλημα απόφασης με βάση το οποίο μπορούμε να το επιλύσουμε. Το πρόβλημα απόφασης

ΕΣΔ(ΑΠΟΦΑΣΗ)

Στιγμιότυπο: $k \in \mathbb{N}$, μη-κατευθυνόμενο γράφημα $G(V, E, w)$, με $w : E \rightarrow \mathbb{N}$.

Ερώτηση: Υπάρχει στο γράφημα G ένα συνεκτικό δένδρο με βάρους μικρότερο ή ίσο του k ;

Έστω ότι μπορούμε να λύσουμε το πρόβλημα ΕΣΔ(ΑΠΟΦΑΣΗ). Επιλύοντας το πρόβλημα αυτό για διαφορετικές τιμές της παραμέτρου k θα μπορούσαμε να καταλήξουμε στη λύση του προβλήματος ΕΣΔ. Παρατηρήστε ότι αυτό δεν σημαίνει ότι το πρόβλημα ΕΣΔ(ΑΠΟΦΑΣΗ) θα πρέπει να επιλυθεί για κάθε πιθανή τιμή του k . Αντίστροφα, η λύση του προβλήματος ΕΣΔ συνεπάγεται την άμεση επίλυση του ΕΣΔ(ΑΠΟΦΑΣΗ).

Το παραπάνω παράδειγμα αποτελεί μία ειδική περίπτωση της αρχής: «κάθε υπολογιστικό πρόβλημα σχετίζεται με - μπορεί να διατυπωθεί σαν - ένα πρόβλημα απόφασης». Επομένως η ιεραρχία των κλάσεων που αφορά τα προβλήματα απόφασης, αφορά κάθε πρόβλημα υπολογισμού.

Η κλάση NP

Εισαγωγή

Η ταξινόμηση των προβλημάτων απόφασης δεν εξαντλείται στις κλάσεις που παρατέθηκαν στην προηγουμένη ενότητα. Ισως η σημαντικότερη κλάση η οποία περιέχει την κλάση P είναι η κλάση NP. Η κλάση αυτή περιέχει όλα τα προβλήματα απόφασης τα οποία επιλύονται σε πολυωνυμικό χρόνο από κάποιον «τυχερό» αλγόριθμο. Ένας αλγόριθμος λέγεται τυχερός αν μπορεί να μαντέψει τις σωστές επιλογές - χωρίς να τις δοκιμάσει όλες - στην πορεία του υπολογισμού προκειμένου να επιλύσει το εκάστοτε στιγμιότυπο του προβλήματος.

Ορισμός

Ορισμός 6. Το σύνολο των προβλημάτων απόφασης για τα οποία η λύση των ΝΑΙ-στιγμιοτύπων μπορεί να επαληθευτεί σε πολυωνυμικό χρόνο συνιστά την τάξη NP.

Ο παραπάνω ορισμός υπονοεί ότι κάθε ΝΑΙ-στιγμιότυπο συνοδεύεται από ένα σύντομο πιστοποιητικό το οποίο αποδεικνύει ότι η απάντηση στο στιγμιότυπο είναι ΝΑΙ. Το πιστοποιητικό έχει τη μορφή συμβολοσειράς και η έννοια του «σύντομου» υποδηλώνει ότι το μέγεθος της είναι πολυωνυμική σε σχέση με το μέγεθος του στιγμιότυπου.

Παράδειγμα

Για παράδειγμα, θεωρούμε το στιγμιότυπο του προβλήματος ΕΣΔ(ΑΠΟΦΑΣΗ) όπου το βεβαρυμένο γράφημα $G(V, E, w)$ απεικονίζεται στο Σχήμα 11.1 και ο ακέραιος k είναι ίσος με 80. Η απάντηση στο στιγμιότυπο αυτό είναι ΝΑΙ και τεκμηριώνεται από το πιστοποιητικό που αποτελεί το ελάχιστο συνεκτικό δένδρο $T(V, E)$ με

$$V = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\},$$

$$E = \{\{1, 7\}, \{1, 10\}, \{1, 11\}, \{2, 8\}, \{3, 8\}, \{4, 8\}, \{5, 9\}, \{6, 9\}, \{8, 10\}, \{9, 10\}\}.$$

Πράγματι μπορούμε να διαπιστώσουμε σε χρόνο $O(|V|)$ ότι το σύνολο E αποτελείται από $|V| - 1 = 11 - 1 = 10$ ακμές του συγκεκριμένου γραφήματος. Στη συνέχεια εκτελώντας στο T διάσχιση κατά βάθος μπορούμε σε $O(|V|)$ ΣΥΒ να διαπιστώσουμε

ότι δν περιέχει κύκλο. Αυτή η διαπίστωση σε συνδυασμό με την προηγούμενη πρόταση σημαίνει ότι το T αποτελεί συνεκτικό δένδρο του G . Ακολούθως, εκτελώντας $O(|V|)$ ΣΥΒ (συγκεκριμένα, $|V| - 2 = 11 - 2 = 9$ προσθέσεις και μία σύγκριση του αθροίσματος με την τιμή $k = 80$), διαπιστώνουμε την εγκυρότητα του T ως πιστοποιητικού για την ορθότητα της απάντησης ΝΑΙ σαν λύση του παραπάνω στιγμιότυπου.

Πόρισμα

Το παραπάνω παράδειγμα αποτελεί άμεση απόδειξη ότι $P \cap NP \neq \emptyset$. Είναι εύκολο να διαπιστώσει κάποιος ότι ισχύει η σχέση $P \subseteq NP$. Το μεγάλο ερώτημα είναι αν η κλάση P αυστηρά εμπεριέχεται στην NP , δηλαδή αν $P \subsetneq NP$ ή αν $P \equiv NP$.

Αναγωγές

Ορισμός

Η έννοια της αναγωγής συνδέει προβλήματα μεταξύ τους. Η ιδέα είναι απλή: αν θέλουμε να λύσουμε κάποιο πρόβλημα το μετατρέπουμε σε ένα πρόβλημα που ήδη γνωρίζουμε πως να επιλύσουμε. Δηλαδή κάθε στιγμιότυπο του προβλήματος A μετατρέπεται σε ένα στιγμιότυπο του προβλήματος B για το οποίο (α) γνωρίζουμε έναν αλγόριθμο επίλυσης και (β) από τη λύση του προκύπτει η λύση του στιγμιότυπου του A . Η διαδικασία μετατροπής ονομάζεται **αναγωγή** και συμβολίζεται με $A \leq_P B$. Παρατηρούμε ότι ο τελεστής της αναγωγής υποσημειώνεται από το P . Αυτό γίνεται προκειμένου να να τονιστεί ότι η αναγωγή - μετατροπή του στιγμιότυπου του προβλήματος A σε ένα στιγμιότυπο του προβλήματος B - πρέπει να γίνεται σε πολυωνυμικό αριθμό βημάτων σε σχέση με το μέγεθος του στιγμιότυπου του προβλήματος A . Ισοδύναμα, το μέγεθος του παραγόμενου στιγμιότυπου είναι πολυωνυμική συνάρτηση του μεγέθους του στιγμιότυπου του A .

Παραδείγματα

Η αναγωγή είναι μία συχνά χρησιμοποιούμενη ιδέα για την επίλυση προβλημάτων. Στον Πίνακα 11.1 δίνονται κάποια παραδείγματα αναγωγής.

Πρόβλημα A	Πρόβλημα B
Εύρεση του δένδρου των συντομότερων μονοπατιών με ρίζα την κορυφή s σε ένα μη-κατευθυνόμενο, μη-βεβαρυμένο γράφημα $G(V, E, s)$	Εύρεση δένδρου συντομότερων μονοπατιών με ρίζα την κορυφή s στο μη-κατευθυνόμενο βεβαρυμένο γράφημα $G(V, E, s, w)$, όπου $w : E \rightarrow \{1\}$.
Εύρεση του συνεκτικού δένδρου μεγαλύτερου βάρους στο γράφημα $G(V, E, w)$	Εύρεση του ελάχιστου συνεκτικού δένδρου στο γράφημα $G(V, E, p)$, όπου $p_e = -w_e, e \in E$
Εύρεση δένδρου συντομότερων μονοπατιών με τερματική κορυφή t στο κατευθυνόμενο βεβαρυμένο γράφημα $G(V, E, t, w)$	Εύρεση δένδρου συντομότερων μονοπατιών με ρίζα την κορυφή t στο κατευθυνόμενο γράφημα $G(V, \bar{E}, t, w)$, όπου η ακμή $(u, v) \in \bar{E}$ να και μόνο αν $(v, u) \in E$

Πίνακας 11.1: Παραδείγματα αναγωγής $A \leq_P B$.

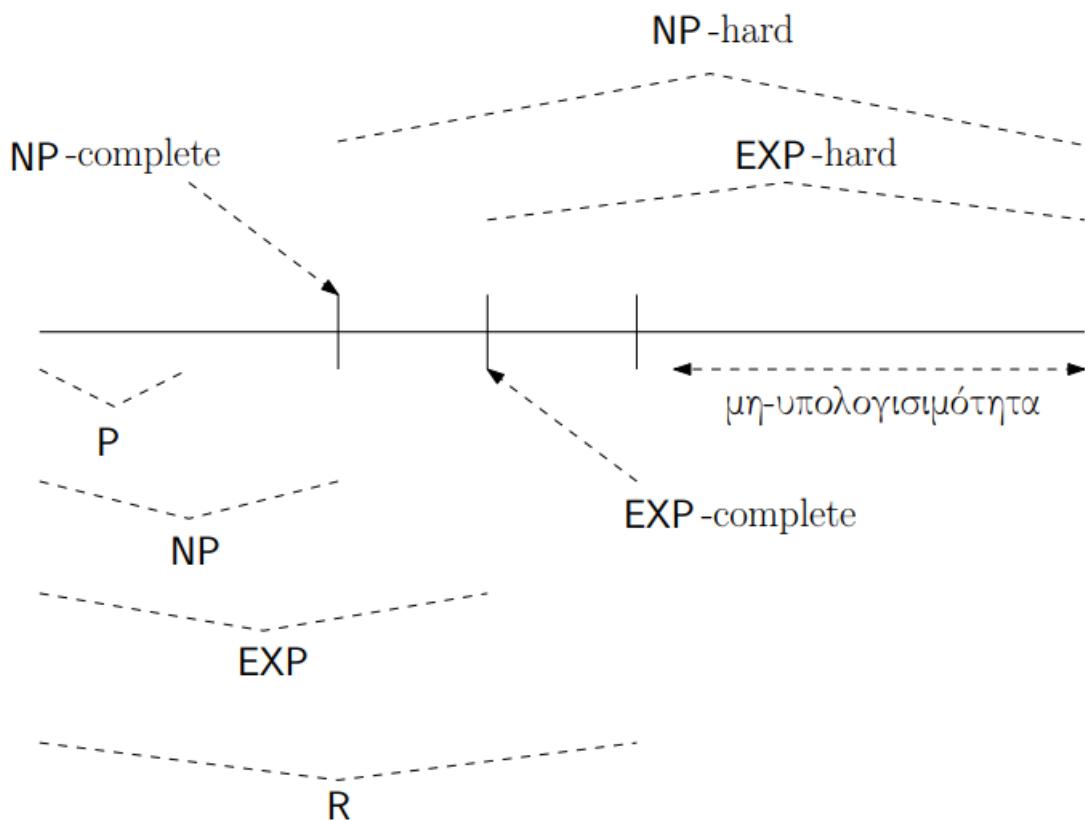
Η κλάση NP-hard

Η έννοια της αναγωγής μπορεί να χαρακτηρίσει προβλήματα ανάλογα με τη «δύσκολία» που παρουσιάζουν. Έστω B ένα πρόβλημα με την ιδιότητα οποιοδήποτε πρόβλημα μίας κλάσης να μπορεί να αναχθεί σε αυτό. Τότε το πρόβλημα χαρακτηρίζεται «δύσκολο» αφού είναι τουλάχιστο τόσο δύσκολο να επιλυθεί όσο το πιο δύσκολο πρόβλημα της συγκεκριμένης κλάσης. Για παράδειγμα, αν οποιοδήποτε πρόβλημα της

κλάσης NP μπορεί να αναχθεί στο B τότε το πρόβλημα B ονομάζεται NP-hard. Αν επιπλέον το πρόβλημα B ανήκει και στην κλάση NP τότε το B ανήκει στην τάξη NP-complete. Δηλαδή, η τάξη NP-complete περιλαμβάνει τα πιο δύσκολα προβλήματα της τάξης NP. Αντίστοιχα, η κλάση EXP-complete περιέχει τα πιο δύσκολα προβλήματα της κλάσης EXP. Παρόμοιοι ορισμοί ισχύουν και για άλλες κλάσεις προβλημάτων. Επίσης είναι γνωστό ότι οι κλάσεις P, NP, EXP είναι κλειστές ως προς την πράξη της αναγωγής.

Ιεραρχία των κλάσεων

Στο Σχήμα 11.2 παρουσιάζεται σχηματικά η ιεραρχία των παραπάνω κλάσεων. Στην απεικόνιση αυτή πρέπει να διαβάζεται με την επιφύλαξη ότι οι ακριβείς σχέσεις ανάμεσα στις κλάσεις P, NP και NP, EXP αποτελούν ανοικτά προβλήματα.



Σχήμα 11.2: Κλάσεις προβλημάτων

Η κλάση NP-complete

Ορισμός

Η κλάση NP-complete παρουσιάζει ιδιαίτερο ενδιαφέρον: για κανένα από τα προβλήματα αυτής της κλάσης δεν είναι γνωστός πολυωνυμικός αλγόριθμος επίλυσης παρόλο που η επαλήθευση του πιστοποιητικού απάντησης ΝΑΙ μπορεί να γίνει σε πολυωνυμικό χρόνο. Αν $P \subsetneq NP$ τότε $NP\text{-complete} \subset NP$ και $P \cup NP\text{-complete} \subsetneq NP$ [12, Θεώρημα 14.1].

Μεθοδολογία

Τα παραπάνω καταδεικνύουν γιατί είναι σημαντικό να μπορούμε να αναγνωρίσουμε αν κάποιο πρόβλημα B ανήκει στην τάξη NP-complete. Για να μπορέσουμε να δείξουμε κάτι τέτοιο θα ακολουθήσουμε την παρακάτω διαδικασία:

1. αποδεικνύουμε ότι $B \in NP$,
2. αποδεικνύουμε ότι $A \leq_P B$, όπου A οποιοδήποτε πρόβλημα της κλάσης NP-complete.

Το δεύτερο βήμα της παραπάνω διαδικασίας αποτελείται από την απόδειξη της πρότασης $B \in NP\text{-hard}$ και εξειδικεύεται ως εξής. Από ένα τυχαίο στιγμιότυπο του προβλήματος A κατασκευάζουμε ένα στιγμιότυπο του προβλήματος B όπου το στιγμιότυπο του προβλήματος A έχει απάντηση ΝΑΙ **αν και μόνο αν** το στιγμιότυπο του προβλήματος έχει απάντηση ΝΑΙ. Επιπλέον, θα πρέπει η τάξη του μεγέθους του στιγμιότυπου του προβλήματος B να είναι πολυωνυμική σε σχέση με το μέγεθος του στιγμιότυπου του προβλήματος A .

Το πρόβλημα SAT

Για αν εφαρμόσουμε την παραπάνω μεθοδολογία θα πρέπει να γνωρίζουμε ένα τουλάχιστον πρόβλημα το οποίο να ανήκει σε αυτή την κλάση. Το πρόβλημα αυτό, ονομάζεται SAT και αφορά την αποτίμηση ενός λογικού τύπου σε κανονική συζευτική μορφή. Το πρόβλημα περιγράφεται με βάση τους επόμενους ορισμούς. Μία λογική μεταβλητή x παίρνει τιμές στο σύνολο {True, False}. Σε κάθε λογική μεταβλητή x αντιστοιχούν δύο όροι: ήτοι x , \bar{x} . Οι όροι αυτοί είναι αντίθετοι: αν μία αποτίμηση δίνει στον έναν όρο την τιμή True τότε αυτομάτως ο άλλος όρος παίρνει την τιμή False. Ο συνδυασμός λογικών όρων με τους τελεστές διάζευξης \vee (OR), σύζευξης \wedge (AND) και με τη χρήση παρενθέσεων δημιουργεί τους λογικούς τύπους. Για παράδειγμα, με τη χρήση των λογικών μεταβλητών x_1, x_2, x_3, x_4 μπορούμε να γράψουμε τους τύπους

$$\phi_1 = (x_1 \vee \bar{x}_2) \wedge (x_1 \wedge \bar{x}_4 \wedge x_3), \quad (11.1)$$

$$\phi_2 = (x_2 \wedge x_4) \vee (\bar{x}_1 \wedge x_3 \wedge \bar{x}_2), \quad (11.2)$$

$$\phi_3 = (\bar{x}_3 \vee \bar{x}_2 \vee x_4) \wedge (x_2 \vee x_1 \vee \bar{x}_4). \quad (11.3)$$

Θα αναφερόμαστε σε κάθε υπο-τύπο που εμπεριέχεται μέσα σε μια παρένθεση ως (*λογική πρόταση*). Για παράδειγμα, στον τύπο που ορίζεται στην (11.1), προτάσεις αποτελούν οι $(x_1 \vee \bar{x}_2)$ και $(x_1 \wedge \bar{x}_4 \wedge x_3)$. Ένας τύπος είναι σε κανονική συζευτική μορφή αν αποτελεί σύζευξη προτάσεων μέσα στις οποίες οι όροι συνδέονται με διάζευξη. Ο τύπος που ορίζεται από την (11.3) είναι σε κανονική συζευτική μορφή. Ο συμμετρικός ορισμός αποτελεί δίνει τύπους σε κανονική διαζευτική μορφή: έχουμε διάζευξη προτάσεων στις οποίες οι όροι συνδέονται με σύζευξη, πχ., τύπος που ορίζεται από την (11.2).

Για κάθε απόδοση τιμών από το σύνολο {True, False} στους λογικούς όρους ενός τύπου, έχουμε μία αποτίμηση του. Για παράδειγμα, αν θέσουμε $x_1 = \text{True}$, $x_2 = \text{False}$, $x_3 = \text{True}$, $x_4 = \text{False}$, οι αποτιμήσεις των παραπάνω τύπων είναι $\phi_1 = \text{True}$, $\phi_2 = \text{False}$, $\phi_3 = \text{True}$. Ένας τύπος ονομάζεται **ικανοποιήσιμος** αν υπάρχει μία απόδοση τιμών στους όρους του ώστε ο τύπος να αποτιμάται σε True. Το πρόβλημα SATαφορά ακριβώς αυτό:

SAT

Στιγμιότυπο: Τύπος ϕ σε κανονική συζευτική μορφή.

Ερώτηση: Είναι ο ϕ ικανοποιήσιμος;

Όπως τονίστηκε προηγουμένως το πρόβλημα SAT είναι το πρώτο πρόβλημα το οποίο αποδείχθηκε ότι ανήκει στην κλάση NP-complete. Θα χρησιμοποιήσουμε το πρόβλημα αυτό σαν «σημείο εκκίνησης» προκειμένου να αποδείξουμε και για κάποια άλλα προβλήματα ότι ανήκουν στην κλάση αυτή. Η μεθοδολογία που θα χρησιμοποιηθεί είναι αυτή που περιγράφηκε στην προηγούμενη ενότητα.