

01_ΓΡΑΦΗΜΑΤΑ**Chapter 1****1a. Εισαγωγή στα Γραφήματα****Διαφάνειες**

Slides_and_Exercises → 01_graphsintro.pdf σελ. 1-13

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: $m = |E(G)|$

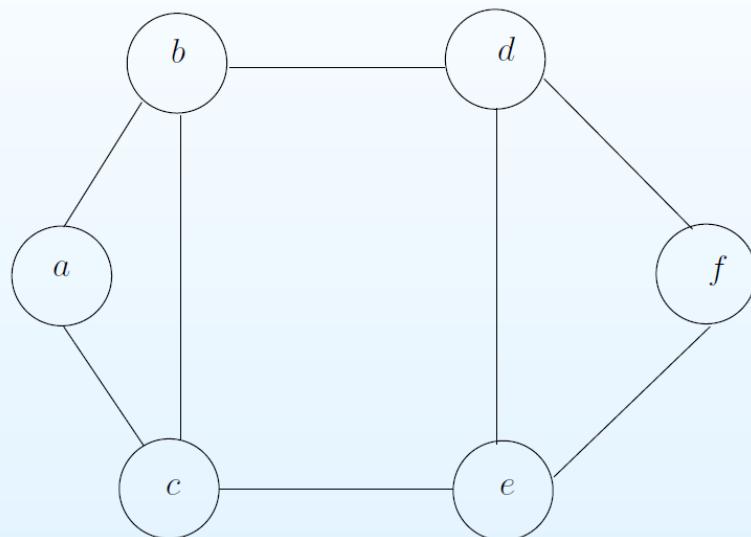
Αν $n = 0 \Rightarrow$ κενό γράφημα

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

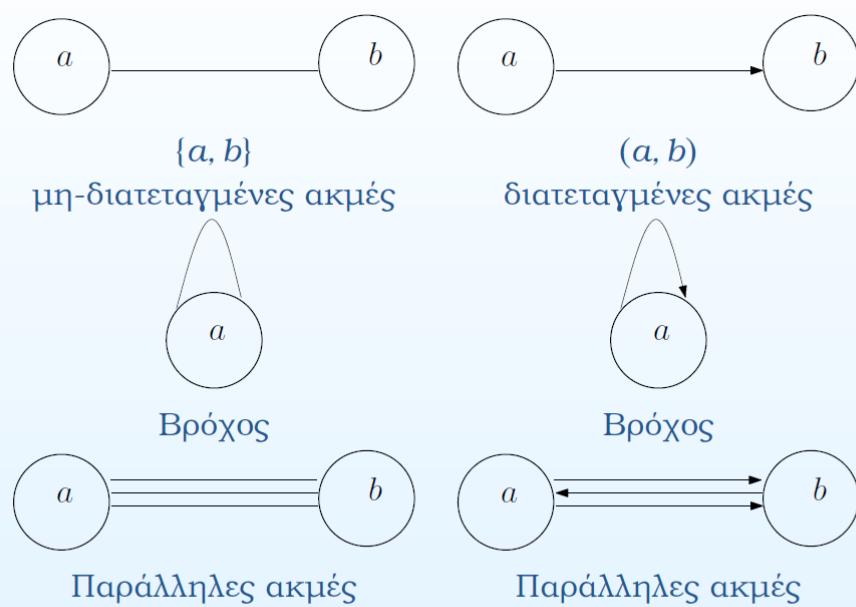
Παράδειγμα

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

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



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



Απλό γράφημα: δεν περιέχει βρόχους ή παράλληλες ακμές

Βαθμός Κορυφής

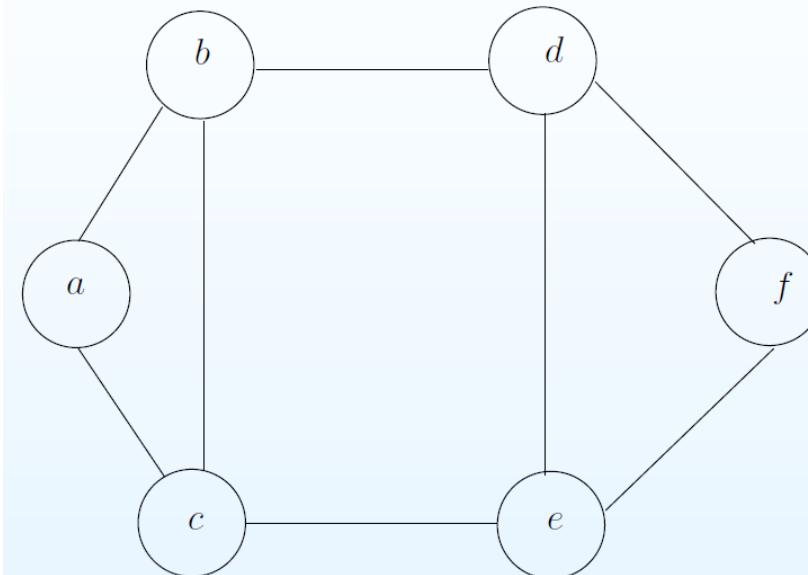
Μη-κατευθυνόμενο γράφημα:

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

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

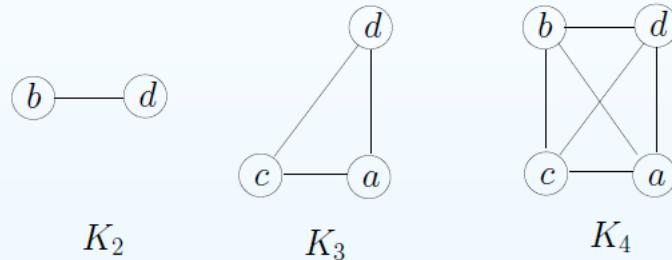


$$d(a) = 2, d(b) = 3, d(c) = 3, d(d) = 3, d(e) = 3, d(f) = 2.$$

Λήμμα 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

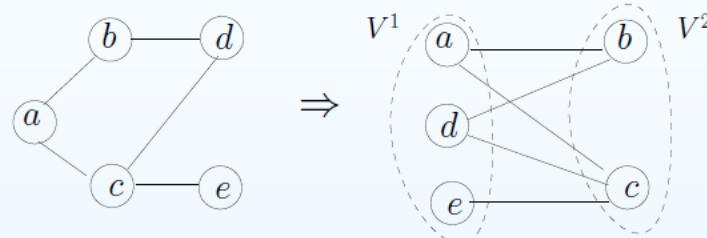
Αριθμός ακμών K_n : $\frac{n(n-1)}{2}$.

Για κάθε απλό γράφημα ισχύει

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

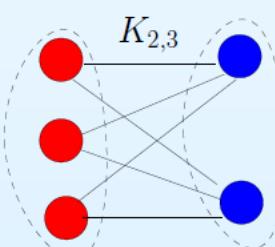
Διμερές Γράφημα

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

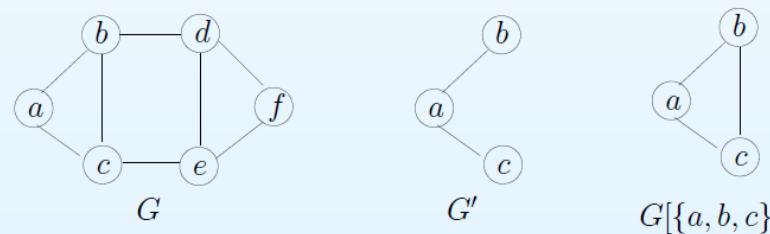
Ένα διμερές γράφημα με $|V^1| = n_1$, $|V^2| = n_2$ που έχει $n_1 * n_2$ ακμές ονομάζεται πλήρες διμερές και συμβολίζεται με K_{n_1, n_2} .



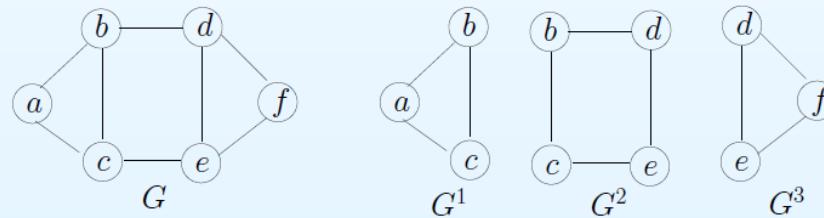
Υπογράφημα

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 $G[V']$.

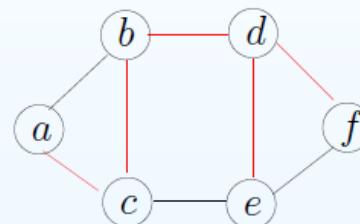


Σχήμα 5: Γράφημα G , Υπογράφημα G' , Επαγόμενο υπογράφημα $G[\{a, b, c\}]$



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

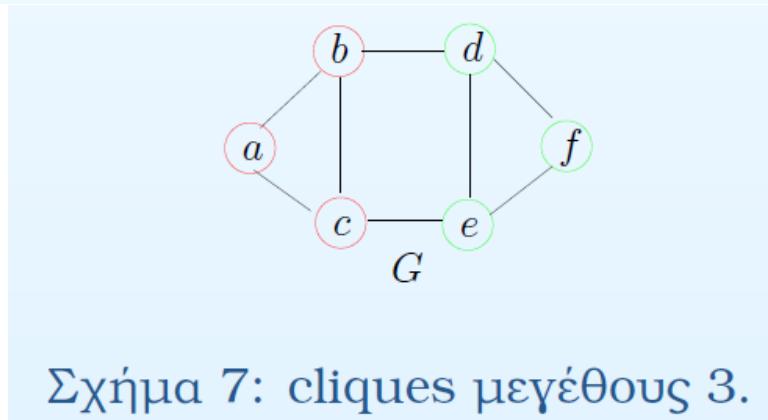
Ένα γεννητορικό υπογράφημα $G'(V', E')$ του $G(V, E)$ έχει $V = V'$ και $E' \subseteq E$. Άρα το G' είναι μεγιστοτικό ως προς το σύνολο E' .



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.

Σημειώσεις

Kώδικας Major

$|A|$: Πληθυμής ή ολίγους των στοιχίων των γνωλών A

$A \setminus B$: Τα στοιχία των A που ΔΕΝ ανήκουν στο B

$\arg\min \{f(x_p) : p \in P\}$: Πάιρνε το x_p πω ελαχιστοποιεί το $f(x_p)$

$\arg\max \{f(x_p) : p \in P\}$: Πάιρνε το x_p πω μεγιστοποιεί το $f(x_p)$

$\min \{f(x_p) : p \in P\}$: Πάιρνε το ελάχιστο $f(x_p)$

$\max \{f(x_p) : p \in P\}$: Πάιρνε το μέγιστο $f(x_p)$

Ο1 - ΓΡΑΦΗΜΑΤΑ / O1-graphsintro

Γράφημα: $G(V, E)$

Πληθυσμός κορυφών: $|V|$

Σύνολο κορυφών των γενεγμάτων G_e : $V(G_e)$

Πληθυσμός ακρών: $|E|$

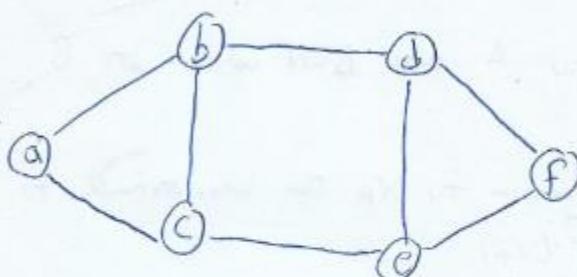
Σύνολο ακρών των γενεγμάτων G_e : $E(G_e)$

-1-

$$V = \{a, b, c, d, e, f\} \quad \text{Παραδείγματα}$$

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

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



Κέντρα Major

$$|V| = n \quad d_v = d(v)$$

$$|E| = m$$

$$N(v) = \{u \in V(G) : \{v, u\} \in E(G)\} : \text{γειτονικές κορυφές}$$

της v

$$|N(v)| = d(v) : \text{βαθμός κορυφής}$$

$$\sum_{v \in V} d_v = 2m : \text{Λογίζει τη Σύμμετρη Χαρακτηριστική } (1) \text{ λογιζεται σε κάθε κορυφή με διπλόν γραφημα}$$

K_n : ολίγες γενιές (Αντί γενιέρα με αρεβίς αντίστροφα σε όλες τους γενιές)

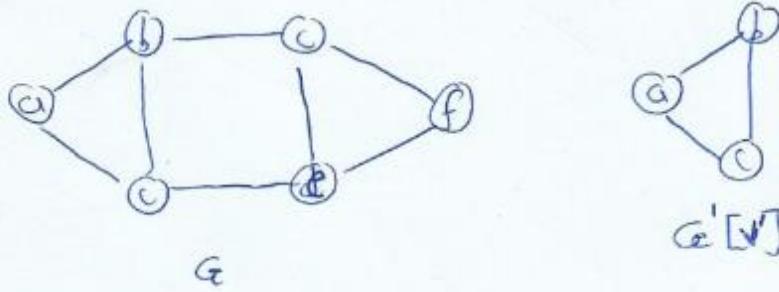
$$|E(K_n)| = \frac{n(n-1)}{2}$$

$$(1) = (n-1) + (n-2) + \dots + (n-1) = 2 \cdot m \quad -9-$$

$G(v, E)$ και $G'(v', E')$

To G' λέγεται υπογράφημα των G ενόσων ($G' \subseteq G$)
 $v' \subseteq v$, $E' \subseteq E$

Μεριστοκύ Υπογράφημα: Περιλήψης ~~διαδικτυακών~~ κορυφών των γενερικών και οιλες της αρχής ή απέντασης ή κανέτες αυτής



Chapter 2

2a. Συνδεσιμότητα και Δένδρα

Διαφάνειες

Slides_and_Exercises → 01_graphsintro.pdf σελ. 16-33

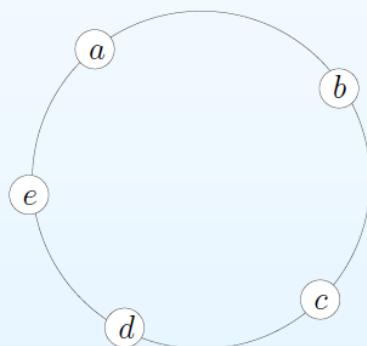
Διαδρομή και Μονοπάτια

Διαδρομή: Μία ακολουθία κορυφών $W = < v_0, v_1, \dots, v_k >$ 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.

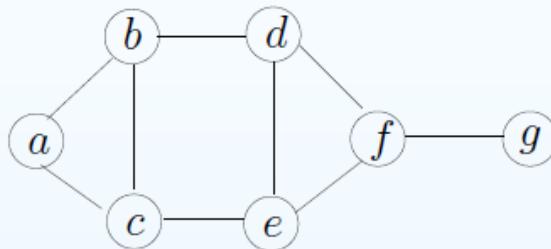


Σχήμα 8: Κύκλος C_5

Ένα γράφημα που δεν περιέχει κύκλο ονομάζεται άκυκλο

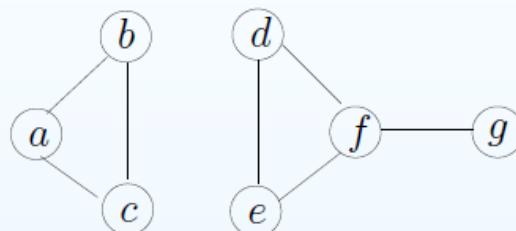
Συνδεδεμένο Γράφημα

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.

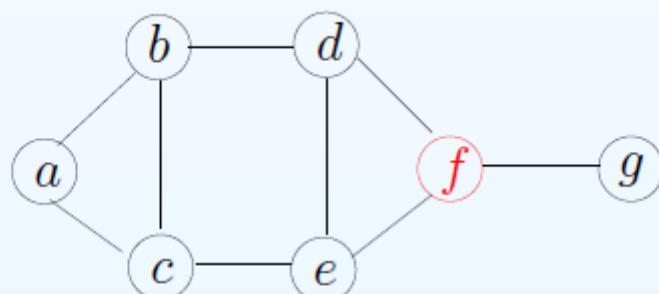


Σχήμα 9: Μη-συνεκτικό γράφημα με δύο συνιστώσες

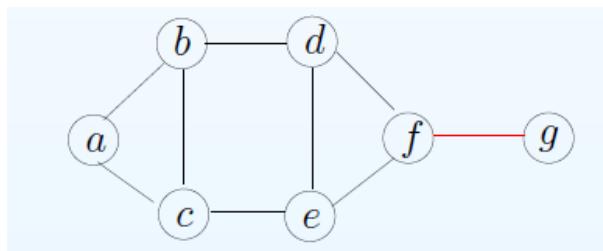
Τομές

Μία κορυφή ονομάζεται σημείο κοπής αν η αφαίρεση της (μαζί με τις προσπίπτουσες 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 $\{f, g\}$

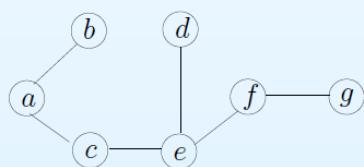
Παρατηρήσεις

- 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.

Ορισμοί Δένδρων

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.

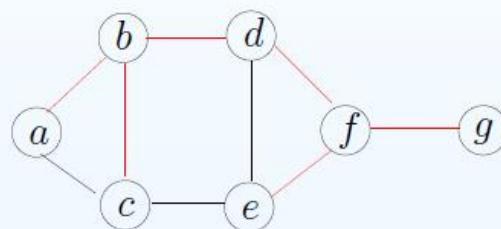
Ειδικά δένδρα.



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 .

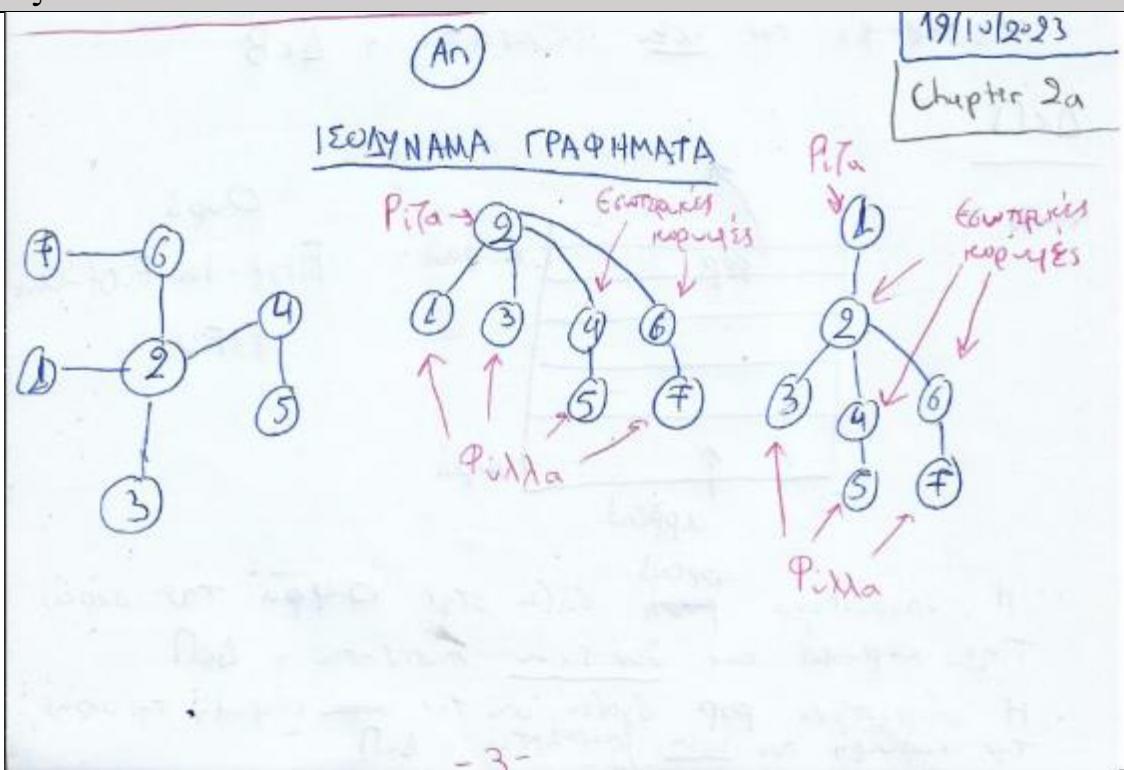
Ρίζα Δένδρου

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.
- Ύψος 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.



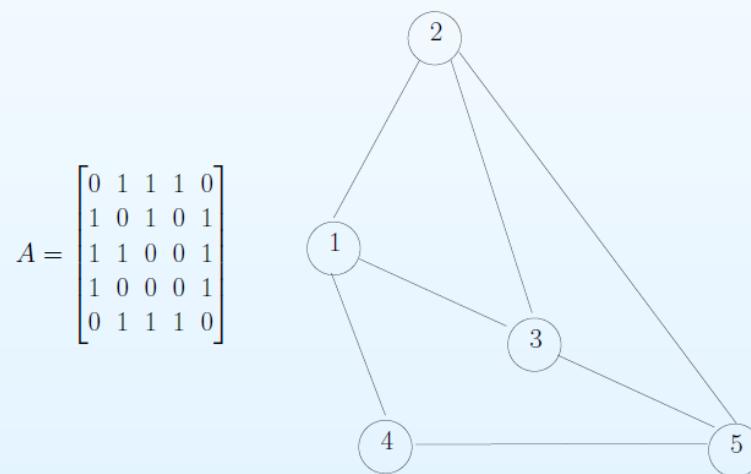
2b. ΔκΒ, ΔκΠ, Σημείο Κοπής στα Μη-Κατευθυνόμενα

Διαφάνειες

[Slides_and_Exercises→02_graphsreptransv.pdf](#) σελ. 1-85

Πίνακας

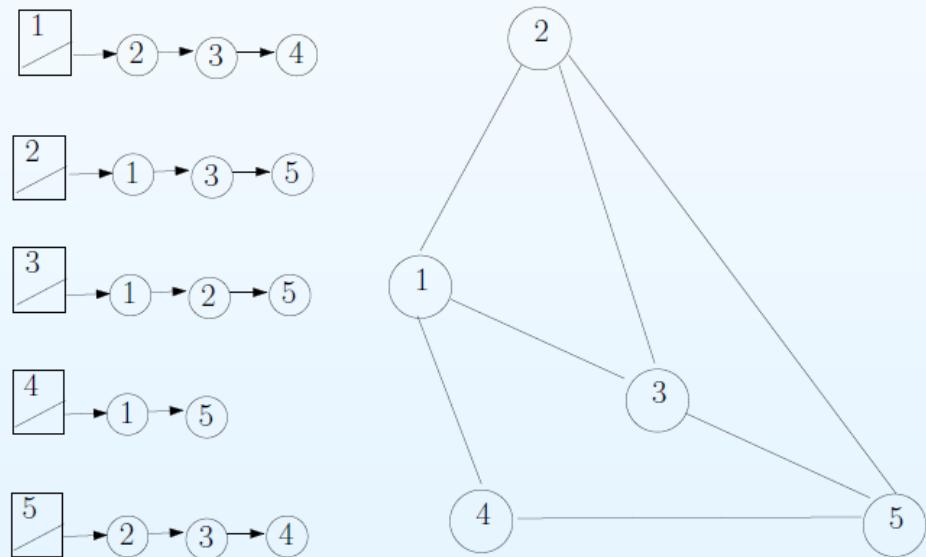
Ο πρώτος τρόπος αναπαράστασης είναι μέσω του πίνακα γειτνίασης ή μητρώο σύνδεσης. Ο πίνακας έχει n γραμμές και n στήλες - αντιστοιχούν στις κορυφές του γραφήματος. στην γραμμή v , στήλη u υπάρχει η τιμή 1 αν $\{v, u\} \in E$ και 0 διαφορετικά.



Σχήμα 1: Πίνακας Γειτνίασης

Λίστα

Εναλλακτικά μπορούμε να χρησιμοποιήσουμε απεικόνιση μέσω λίστων: 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

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()

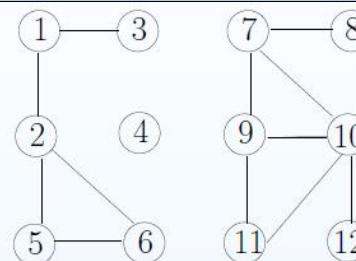
for $v \in V$

$visited[v] \leftarrow false;$

for $v \in V$

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

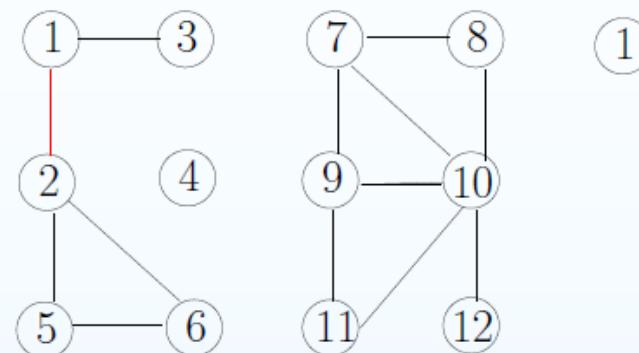
Παράδειγμα

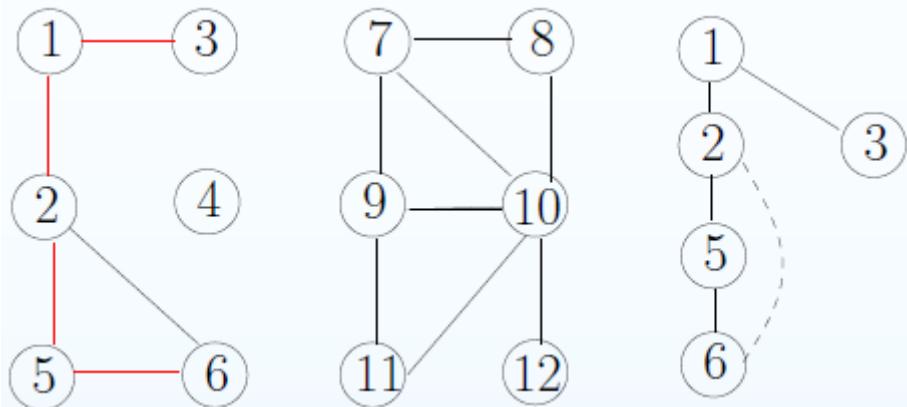
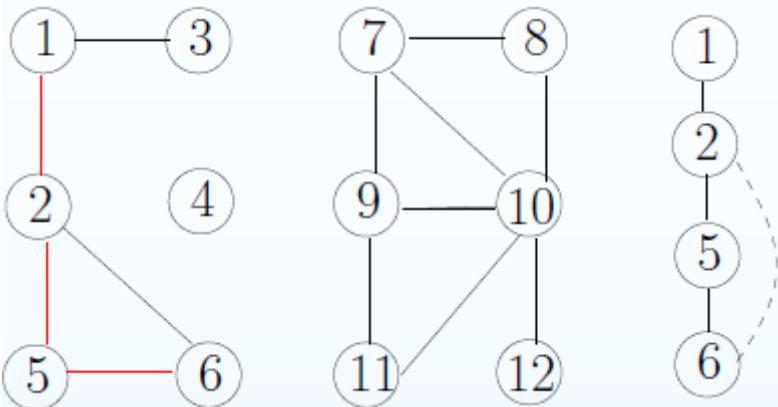
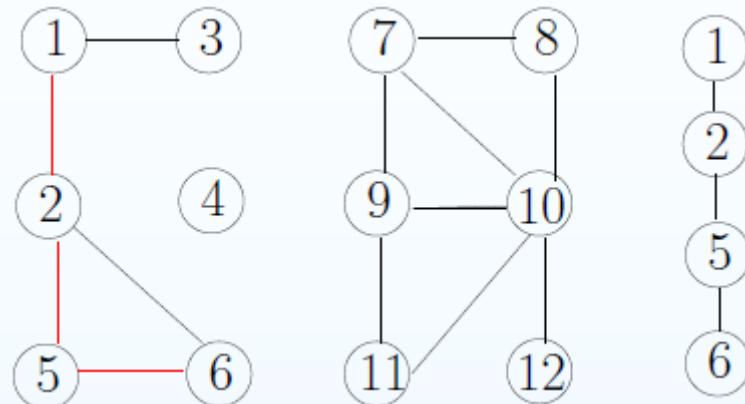
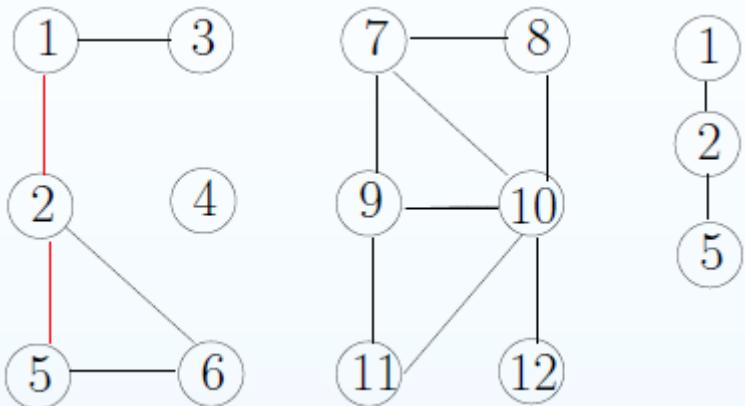
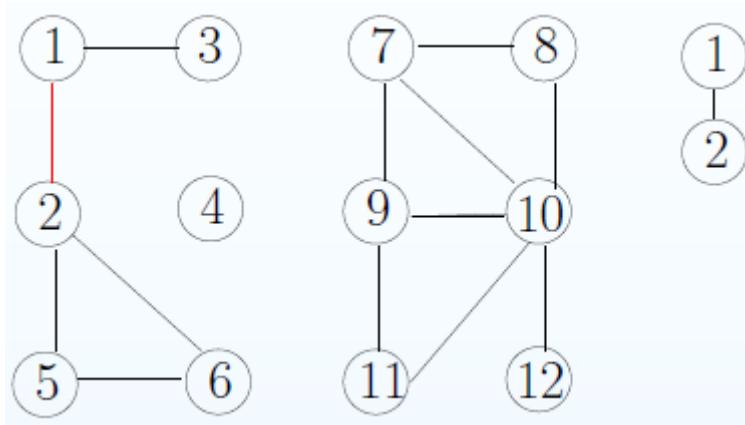


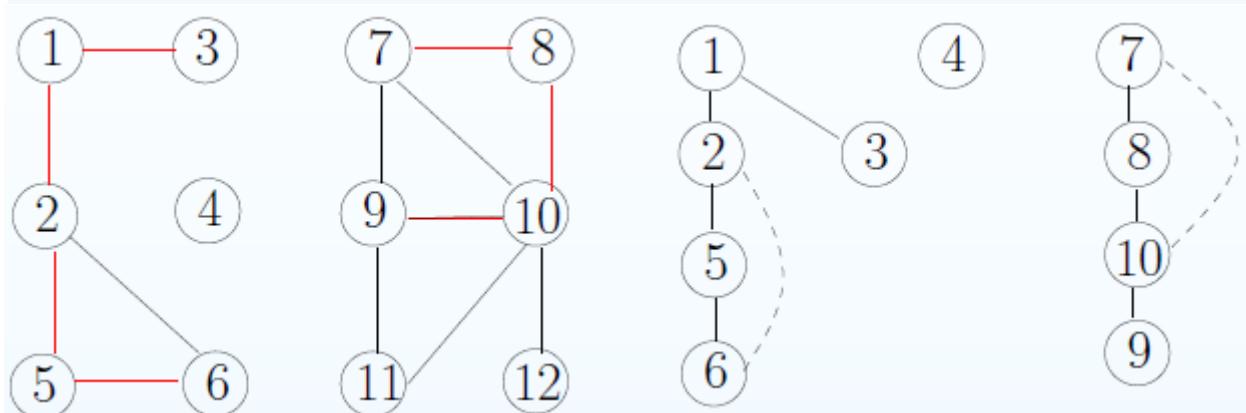
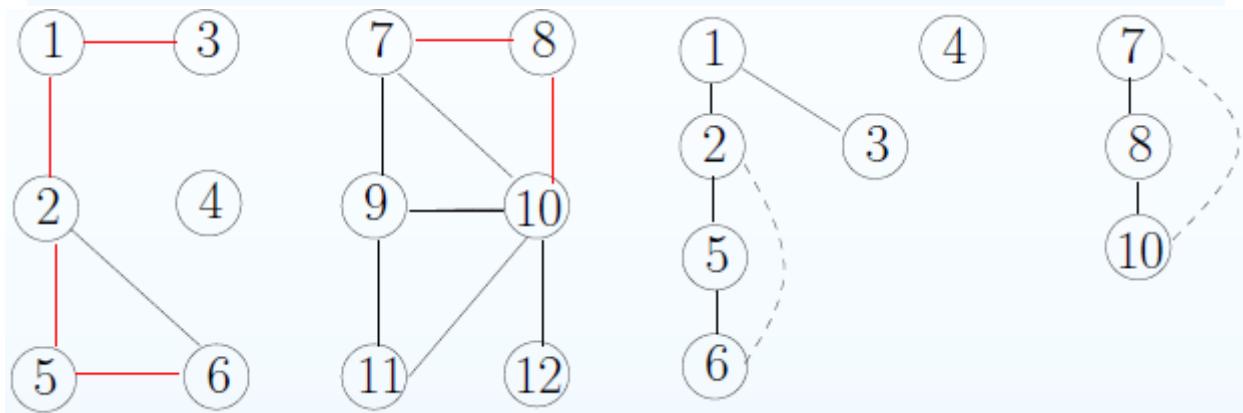
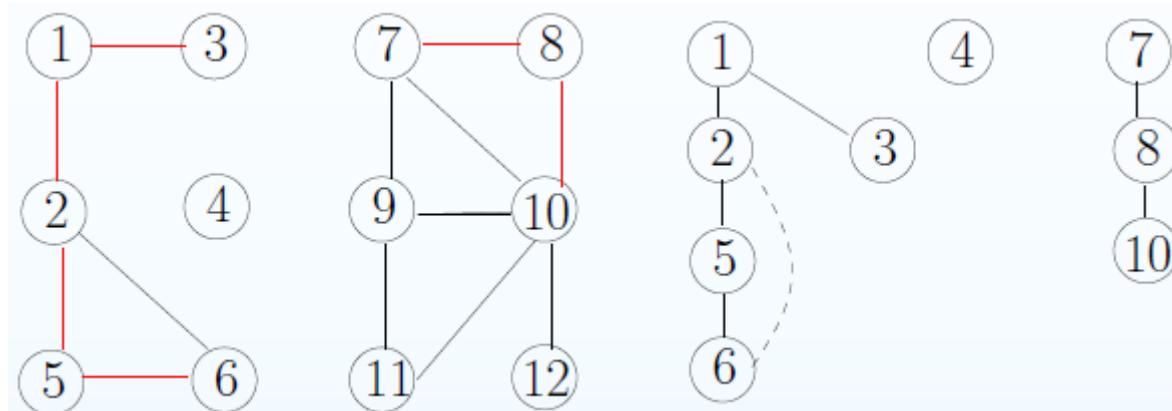
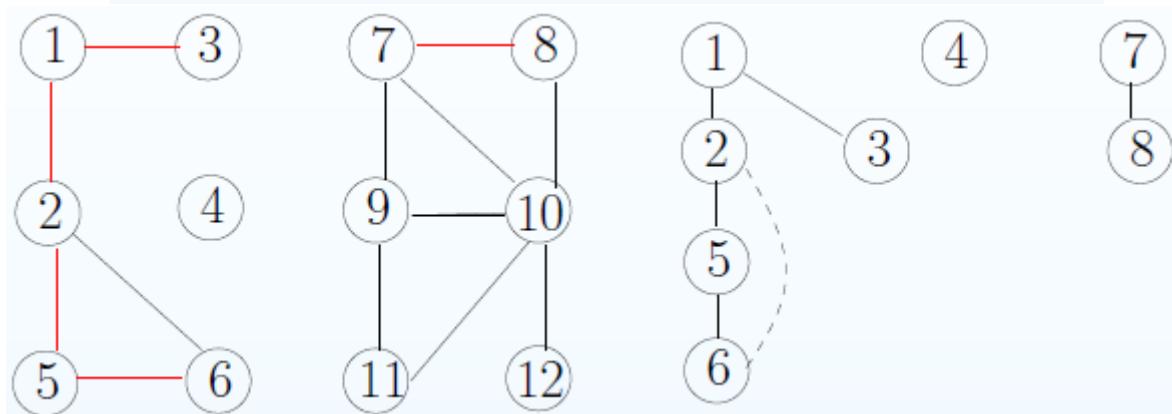
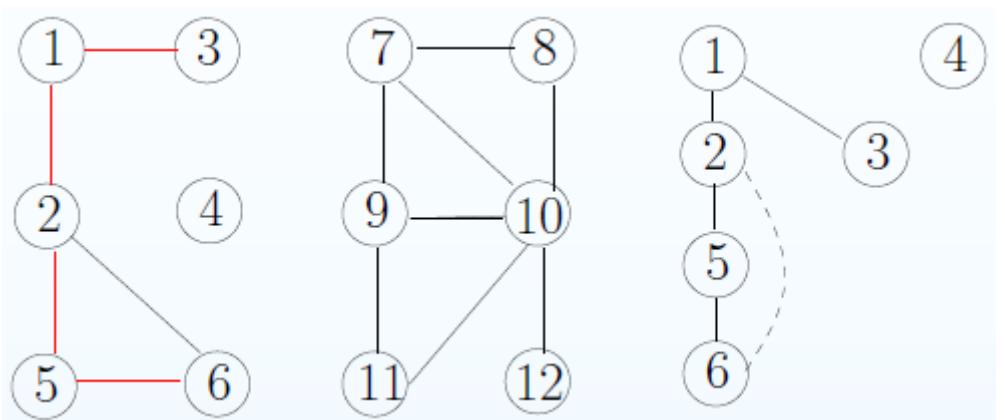
Σχήμα 3: Γράφημα

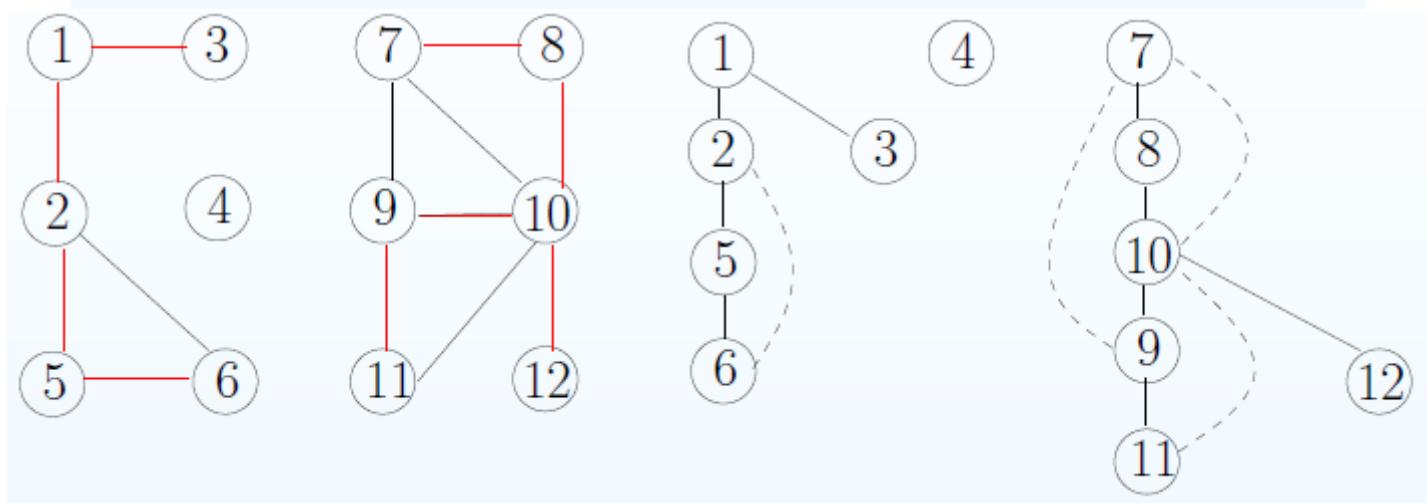
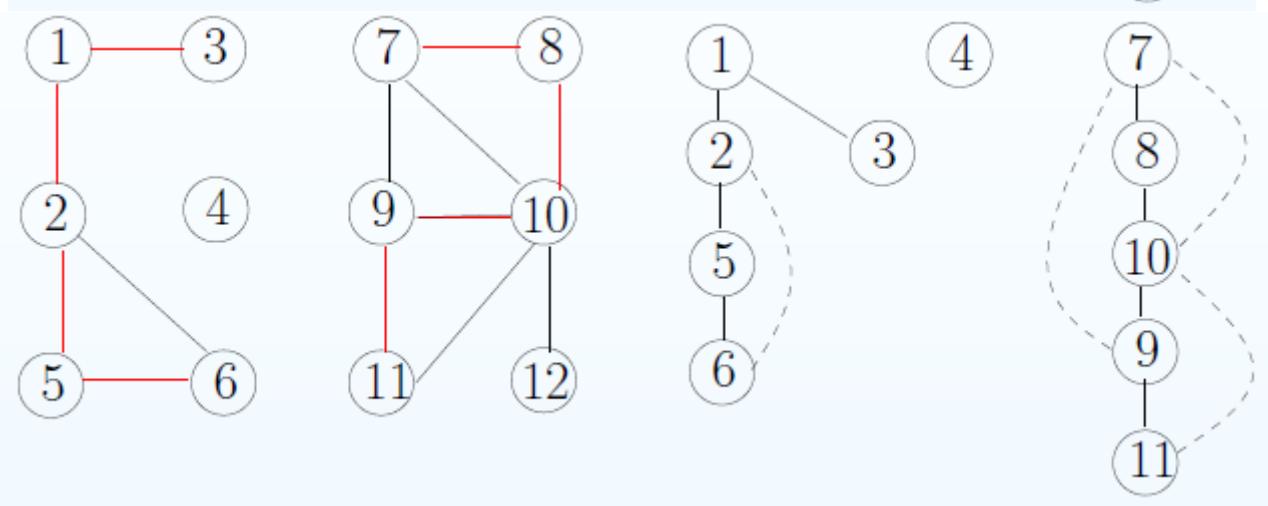
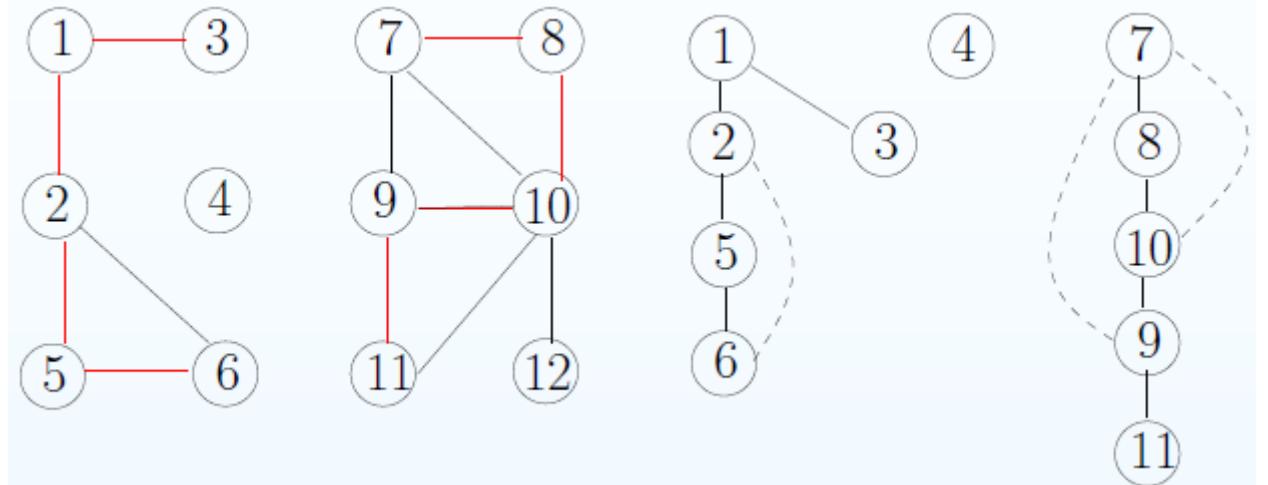
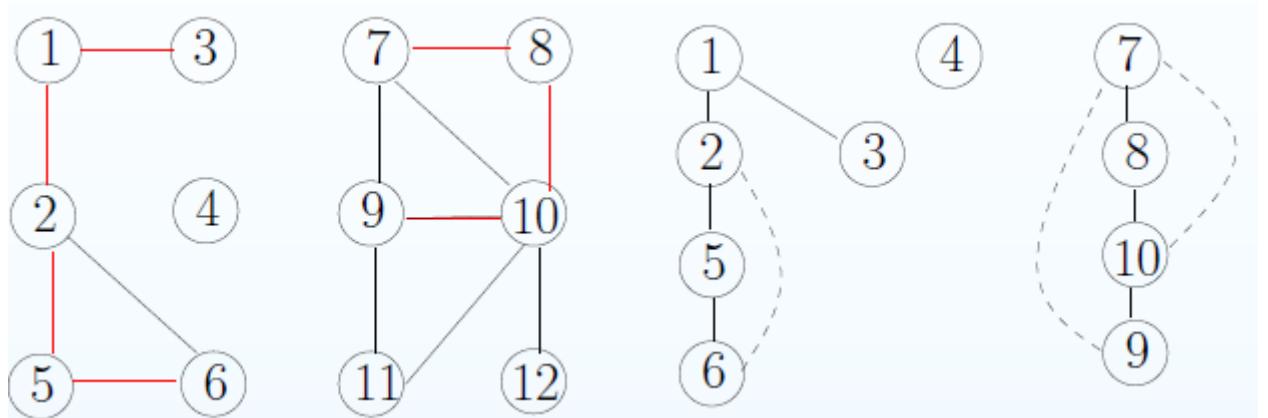
Θα εκτελέσουμε ΔκΒ στο παραπάνω γράφημα θεωρώντας ότι οι κόμβοι στη λίστα γειτνίασης εμφανίζονται με λεξικογραφική σειρά.

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









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**

$visited[v] \leftarrow \text{true};$

$preorder[v] \leftarrow ++clock;$

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 $v \in V$

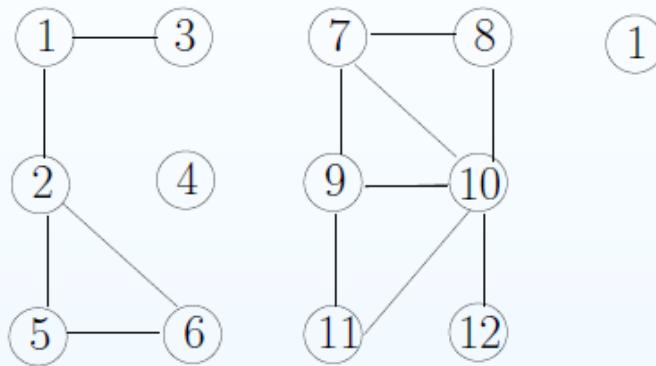
if not $visited[v]$ **then**

$\text{push}(v, Q)$;

explore($clock, Q, visited, preorder$);

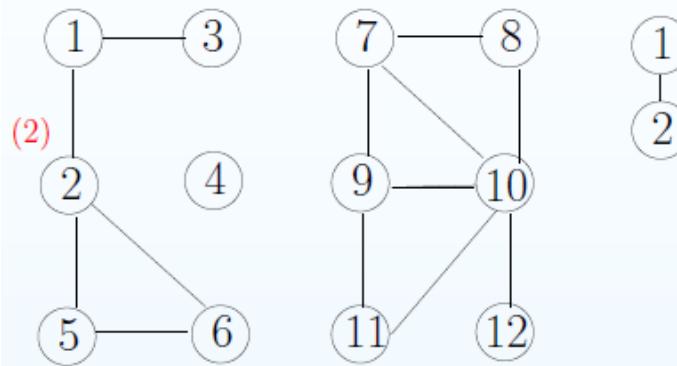
Παράδειγμα

(1)



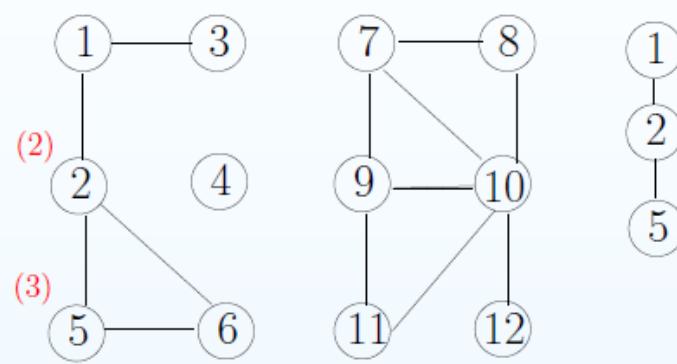
$$Q : 2 - 3$$

(1)



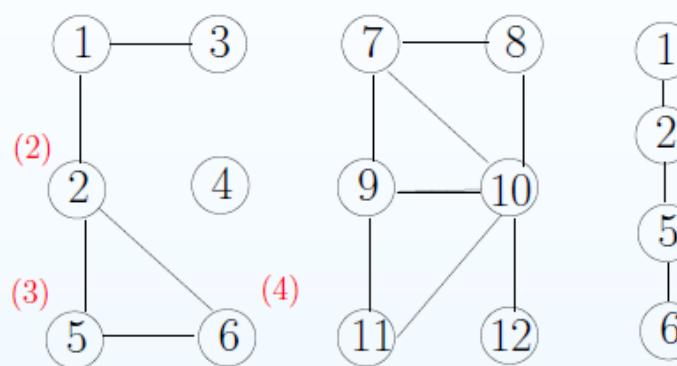
$$Q : 5 - 6 - 3$$

(1)



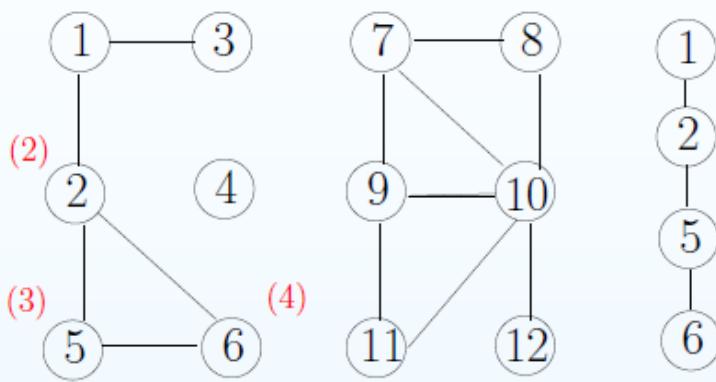
$$Q : 6 - 6 - 3$$

(1)



$$Q : 6 - 3$$

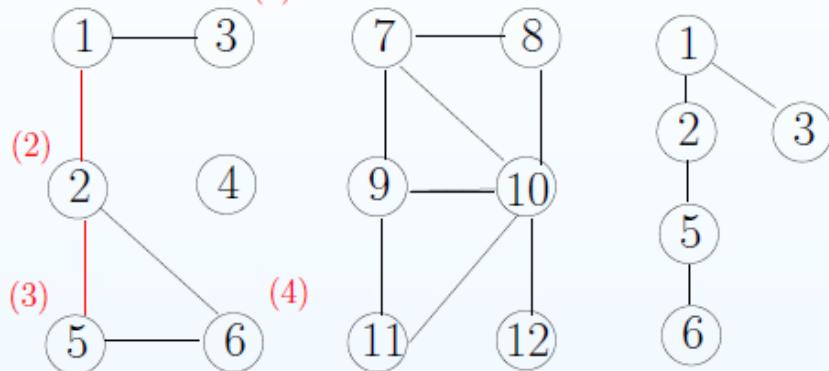
(1)



$Q : 3$

(1)

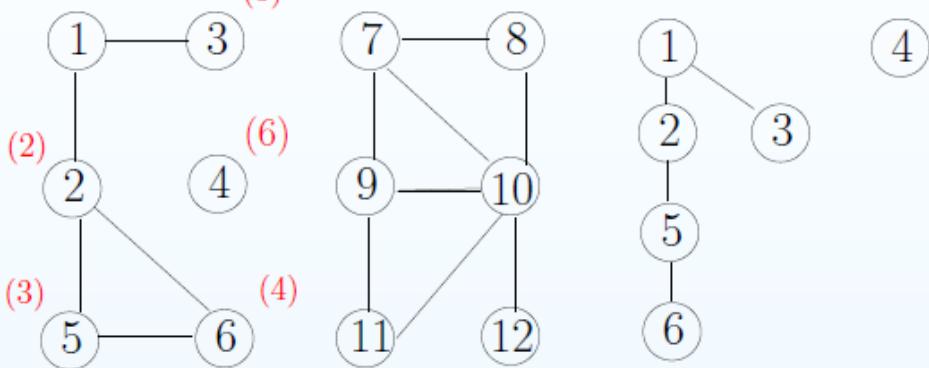
(5)



$Q :$

(1)

(5)

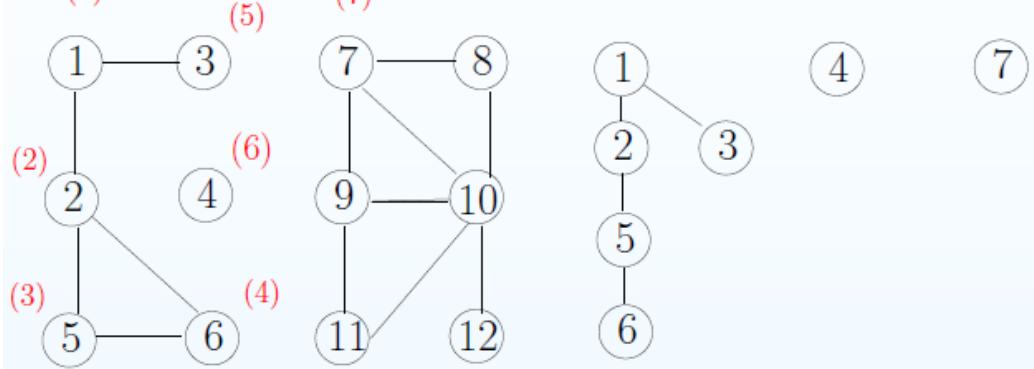


$Q :$

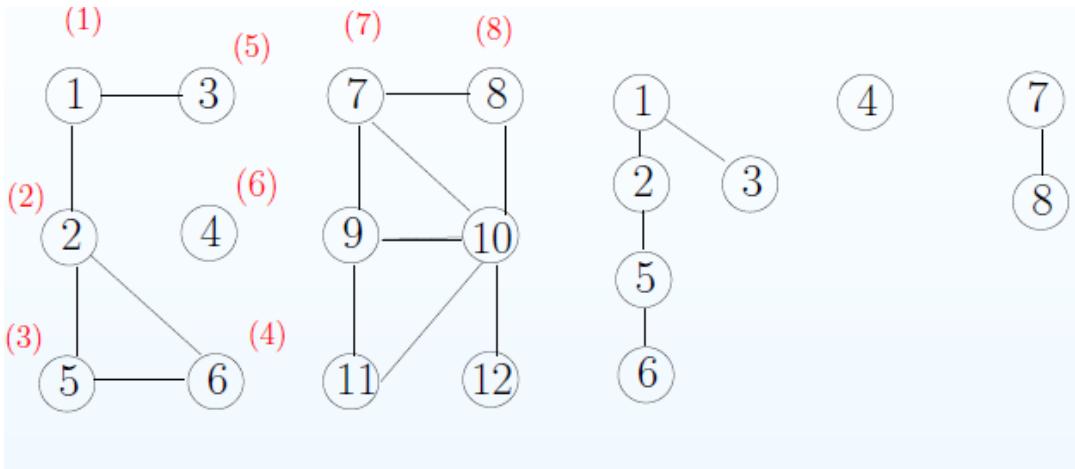
(1)

(5)

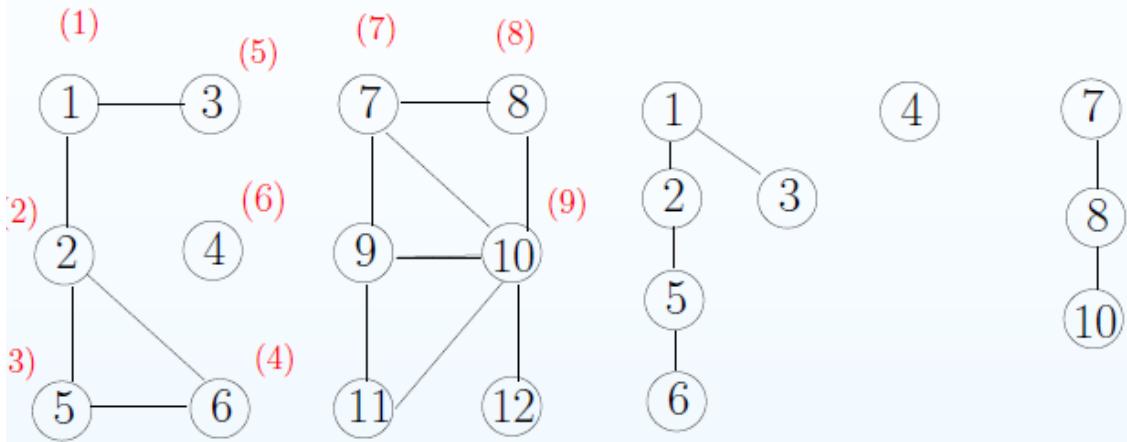
(7)



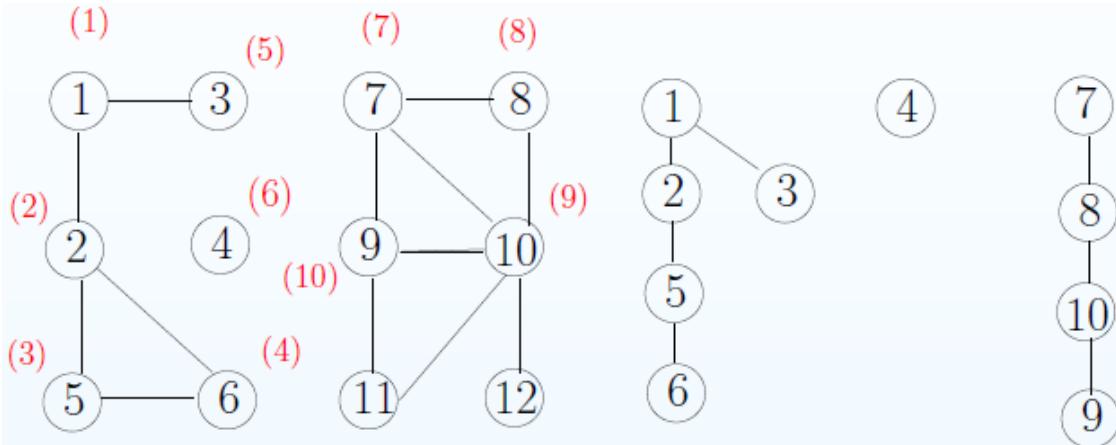
$Q : 8 - 9 - 10$



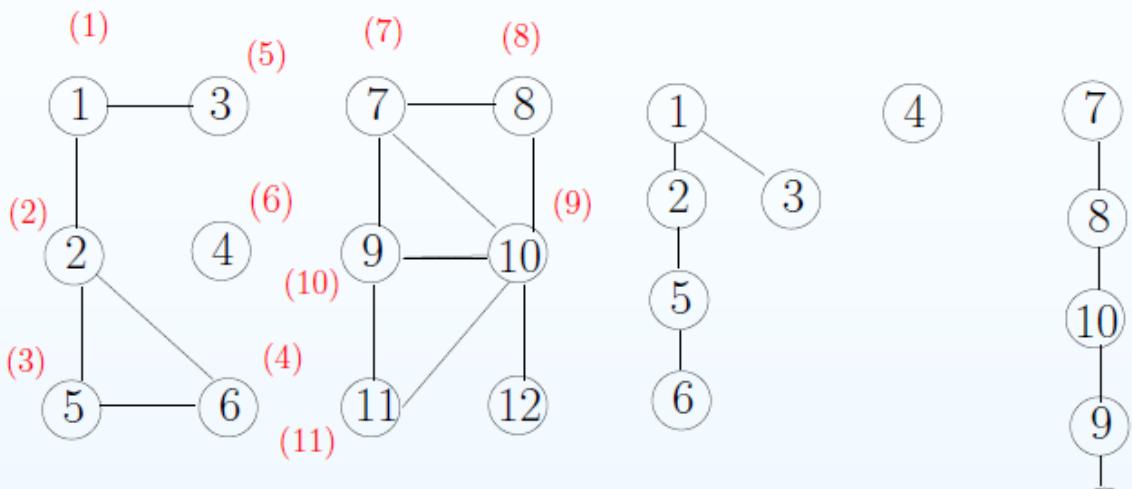
$$Q : 10 - 9 - 10$$



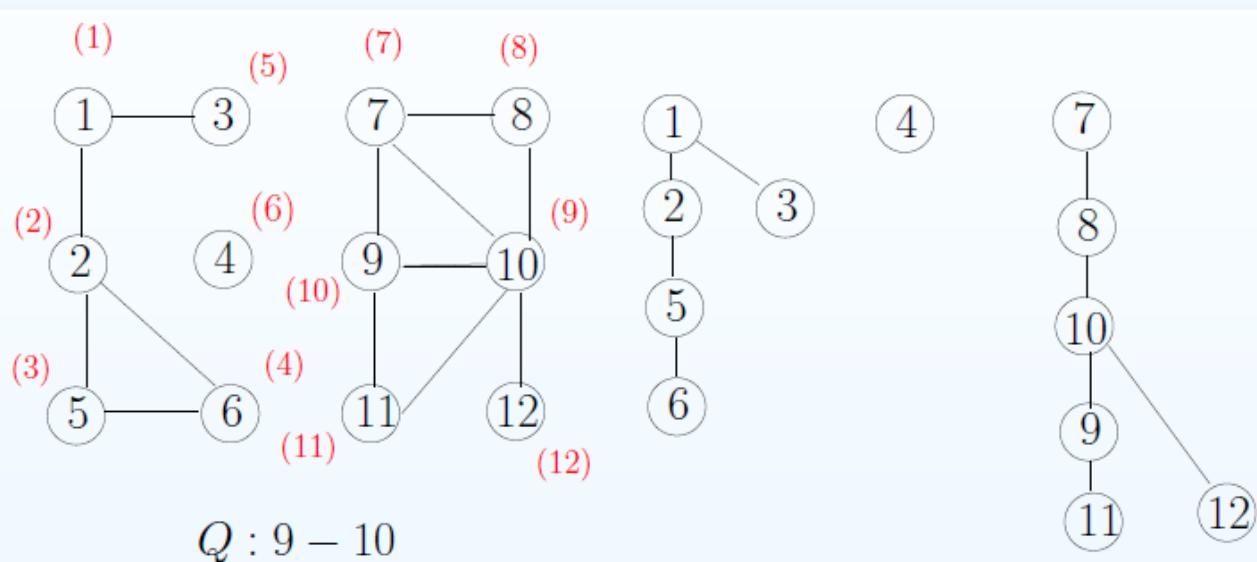
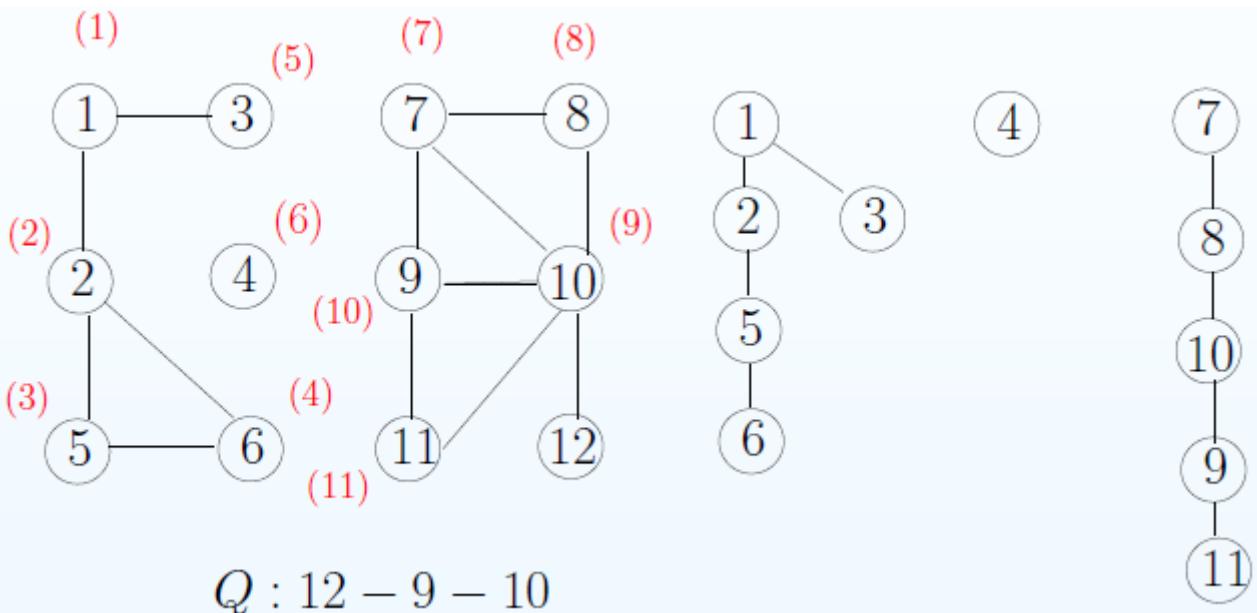
$$Q : 9 - 11 - 12 - 9 - 10$$



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



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



ΔκΠ

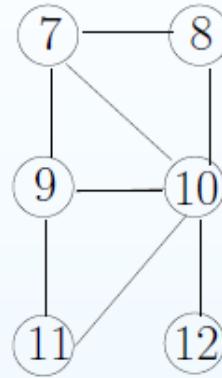
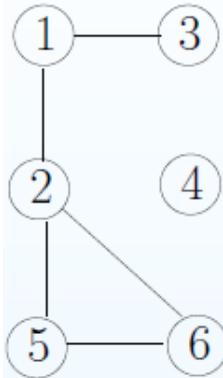
Η Διάσχιση κατά Πλάτος (ΔκΠ) πρώτα επισκέπτεται όλους τους κόμβους που βρίσκονται στο ίδιο επίπεδο και μετά προχωρά στους επόμενους κόμβους. Προκειμένου να υλοποιηθεί αυτή η στρατηγική η μόνη αλλαγή που χρειάζεται να γίνει είναι στην επεξεργασία της ουράς Q . Αντί για LIFO εφαρμόζουμε FIFO. Έτσι όταν 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( $u$ ,  $Q$ );
  
```

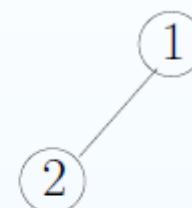
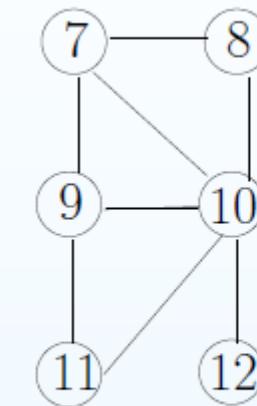
Παράδειγμα

(1)



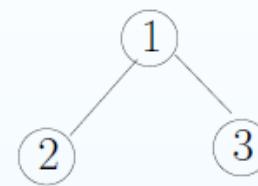
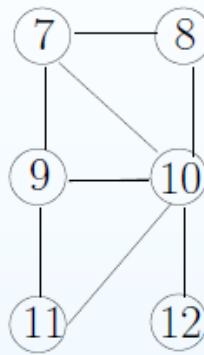
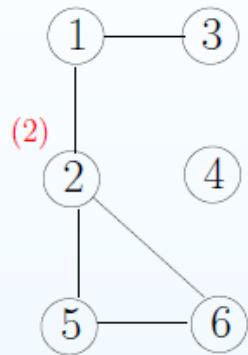
$$Q : 2 - 3$$

(1)



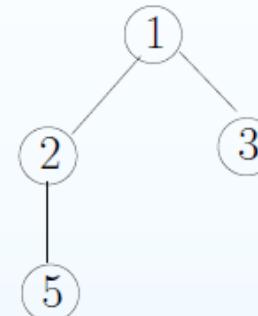
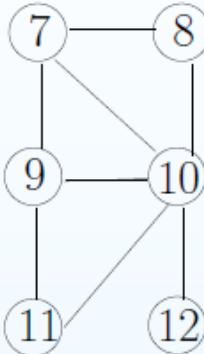
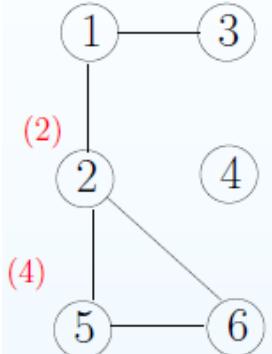
$$Q : 3 - 5 - 6$$

(1) (3)

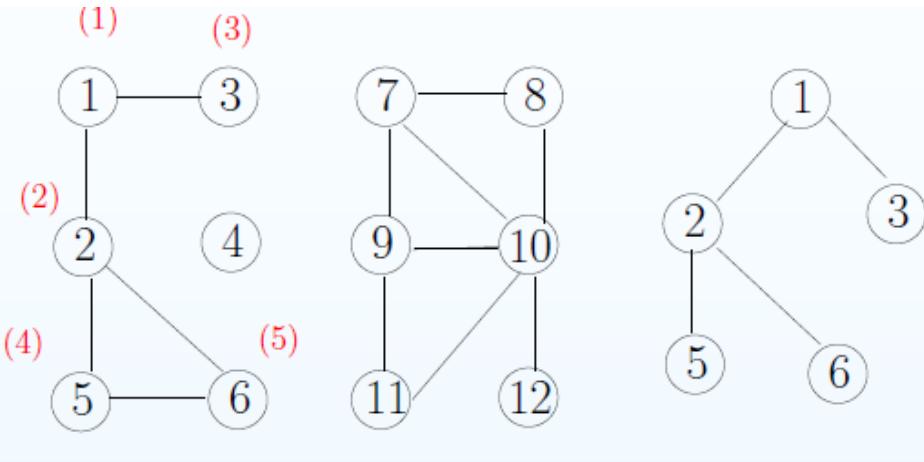


$$Q : 5 - 6$$

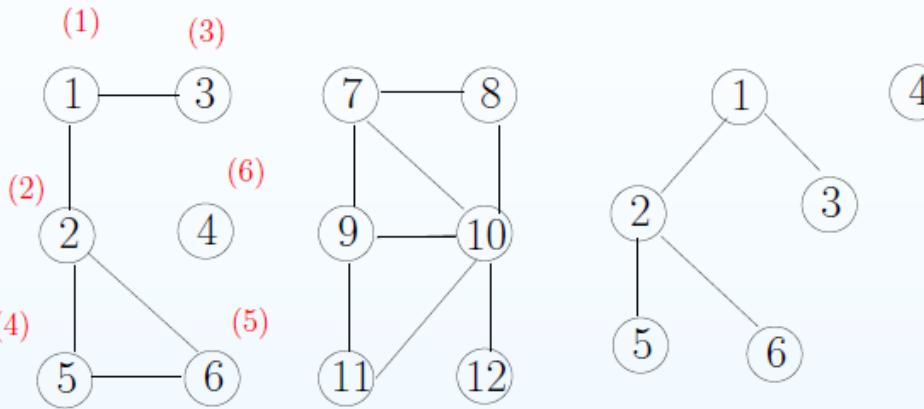
(1) (3)



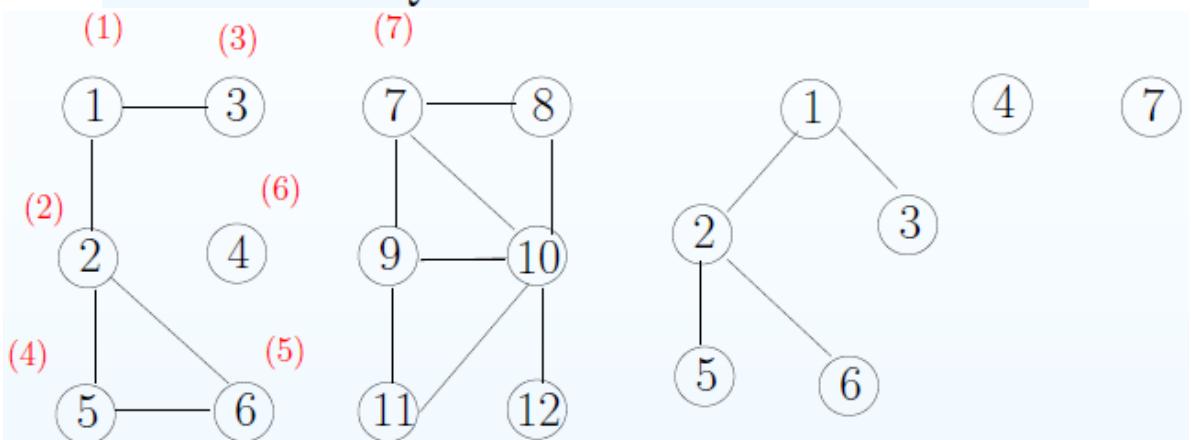
$$Q : 6 - 6$$



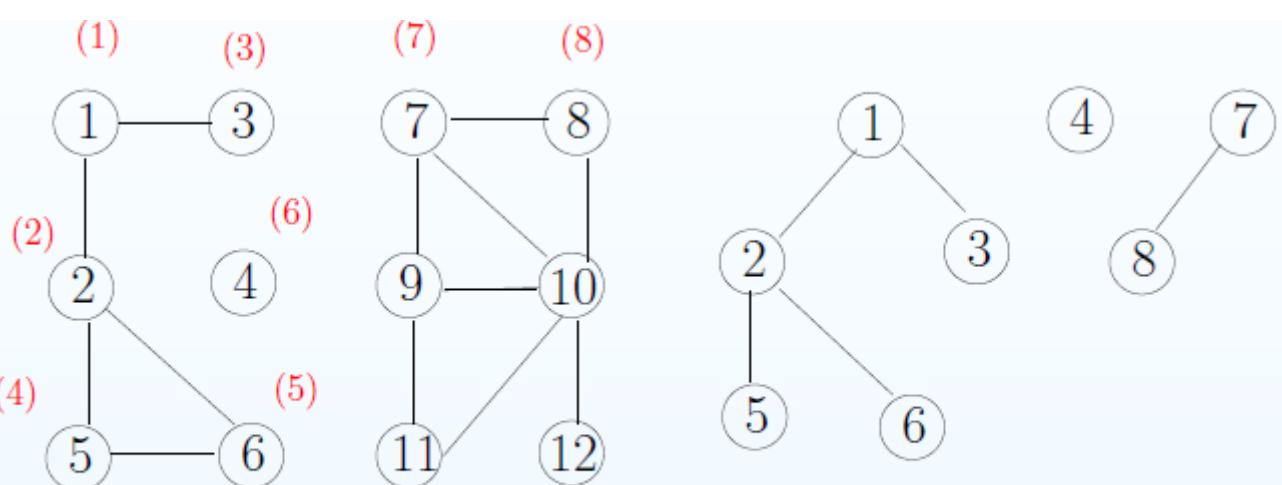
$Q : 6$



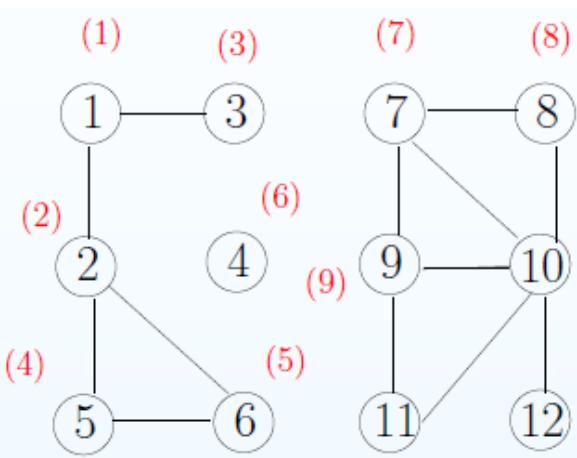
$Q :$



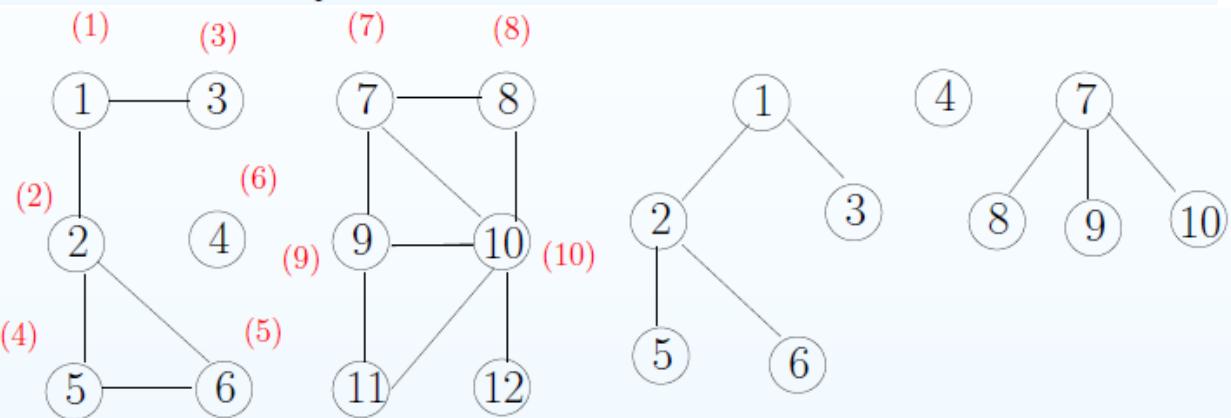
$Q : 8 - 9 - 10$



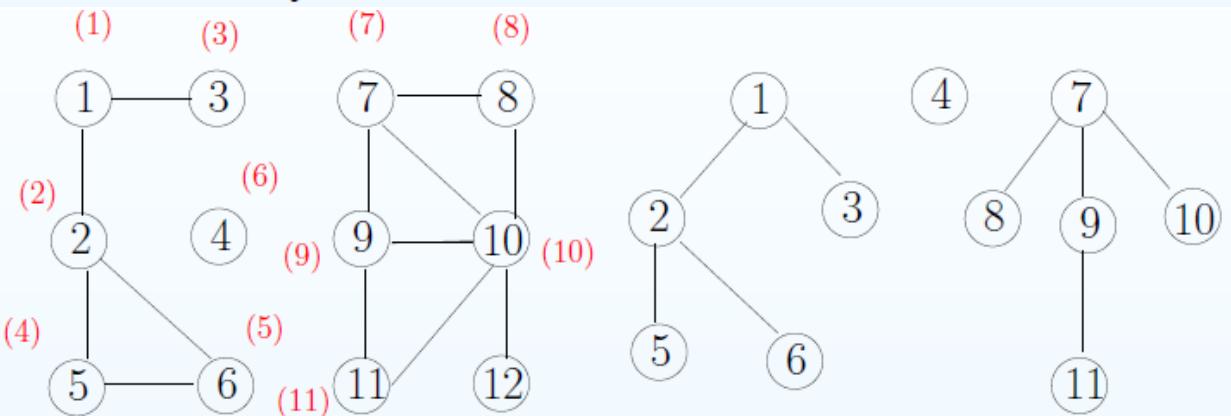
$Q : 9 - 10 - 10$



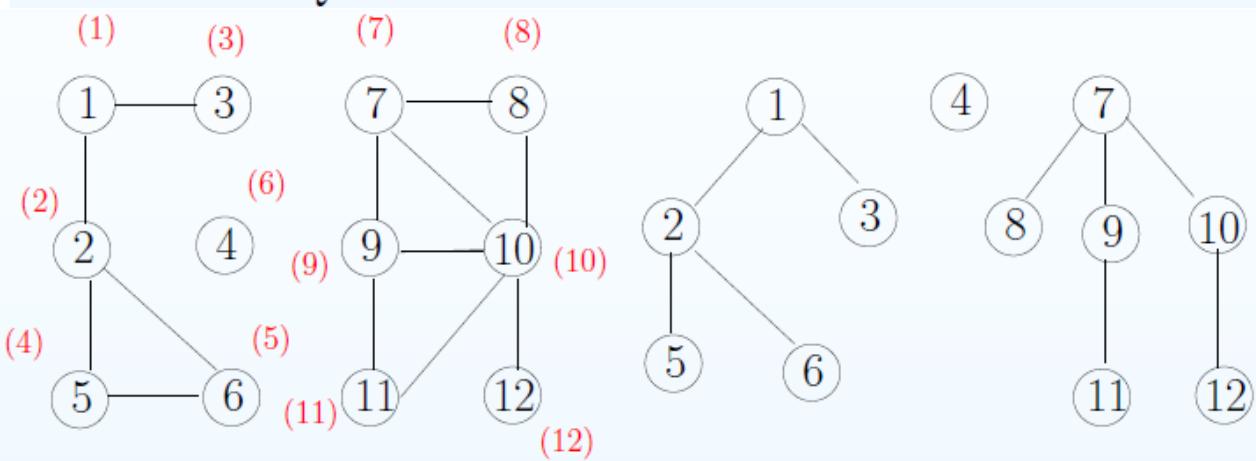
$$Q : 10 - 10 - 10 - 11$$



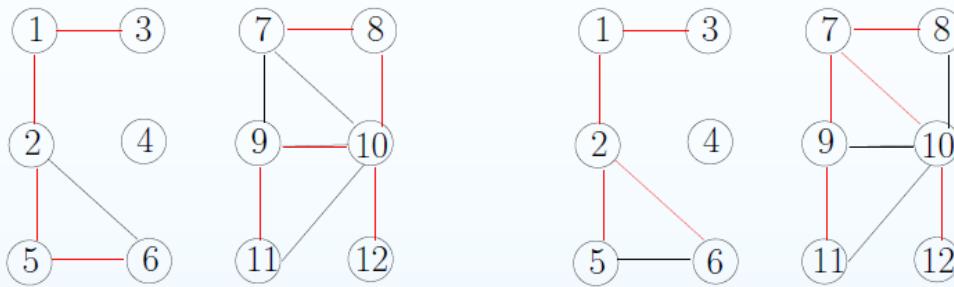
$$Q : 10 - 10 - 11 - 11 - 12$$



$$Q : 11 - 12$$



$$Q :$$



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 επισκέπτονται 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 προσπίπτουσα 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 αν $v = parent[u]$. Η ρίζα του δένδρου r θα έχει $parent[r] = 0$.

explore(*clock*, *v*, *visited*, *preorder*, *parent*)

visited[*v*] \leftarrow true;

preorder[*v*] \leftarrow ++ *clock*;

for *u* \in *N(v)*

if not *visited*[*u*] **then**

parent[*u*] \leftarrow *v*;

explore(*clock*, *u*, *visited*, *preorder*, *parent*);

main()

clock \leftarrow 0;

for *v* \in *V*

visited[*v*] \leftarrow false;

preorder[*v*] \leftarrow 0; *parent*[*v*] \leftarrow 0;

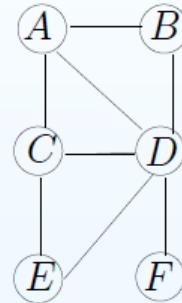
for *v* \in *V*

if not *visited*[*v*] **then**

explore(*clock*, *v*, *visited*, *preorder*, *parent*);

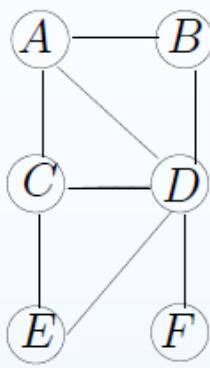
Παράδειγμα

Θα εφαρμόσουμε τον αλγόριθμο στο γράφημα του Σχήματος 8.



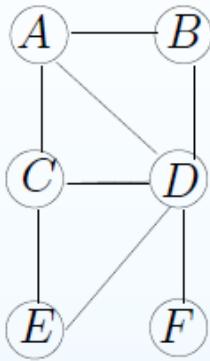
Σχήμα 8: Γράφημα - ΔΚΒ

Οι τιμές του πίνακα preorder απεικονίζονται με μπλέ χρώμα.

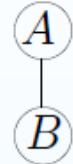


(1) (0) A

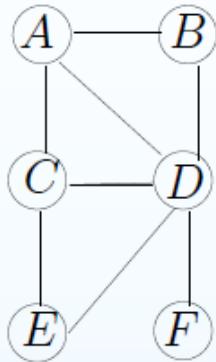
parent= (0, 0, 0, 0, 0, 0)



(1)
(2)



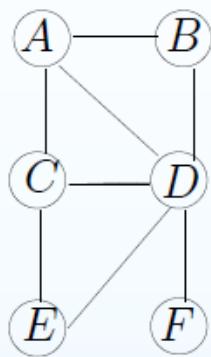
parent= (0, A, 0, 0, 0, 0)



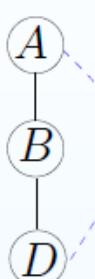
(1)
(2)
(3)



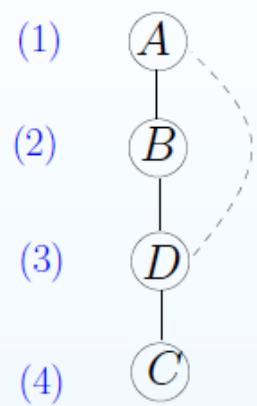
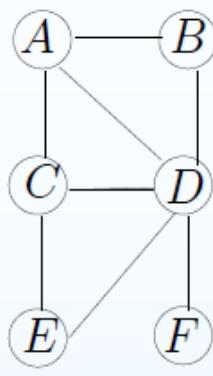
parent= (0, A, 0, B, 0, 0)



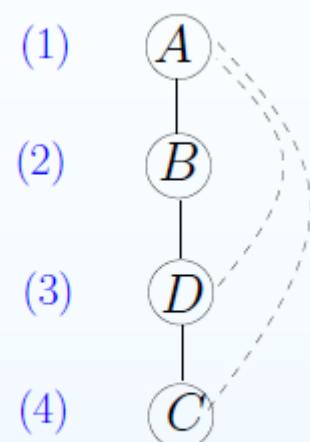
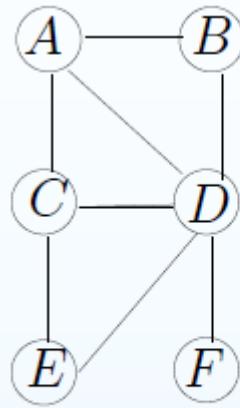
(1)
(2)
(3)



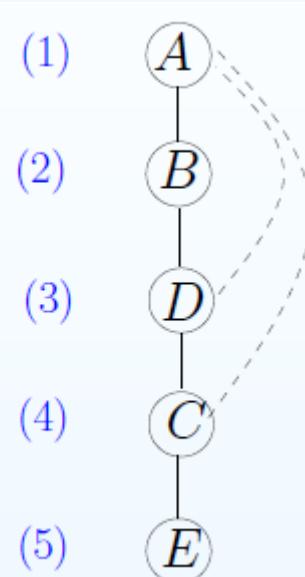
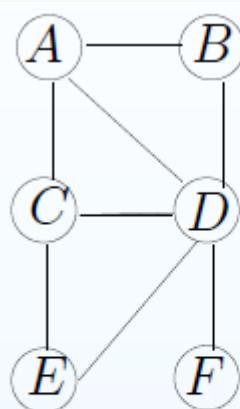
parent= (0, A, 0, B, 0, 0)



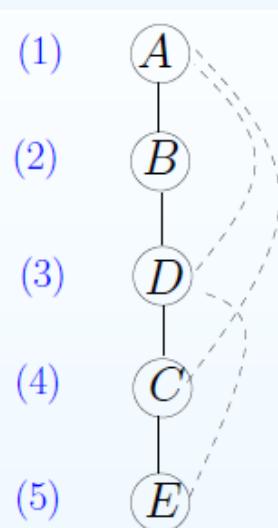
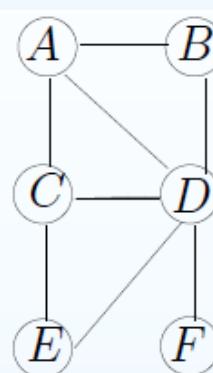
parent= (0, A, D, B, 0, 0)



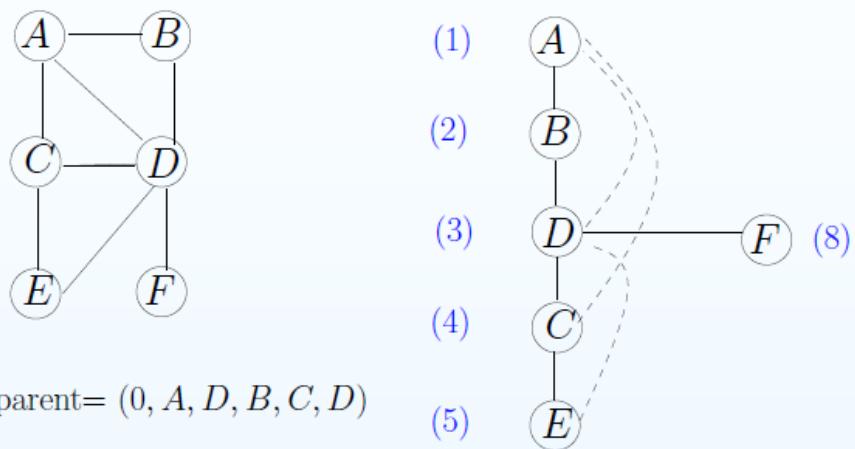
parent= (0, A, D, B, 0, 0)



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



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



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) $\text{parent}[u] = v$, αν η ακμή ανήκει στο T .

Ιδιότητες

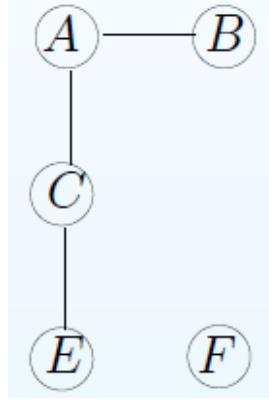
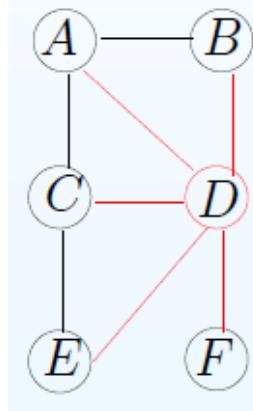
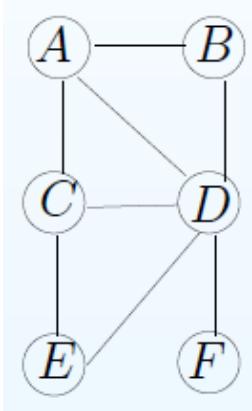
Μετά την εκτέλεση της ΔκΒ, οι ακμές του G που δεν ανήκουν στο σύνολο του παραχθέντος γεννητορικού δένδρου ονομάζονται *οπισθοακμές*. Αυτές απεικονίζονται στο Σχήμα 9 σαν διακεκομμένες.

Μία 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.



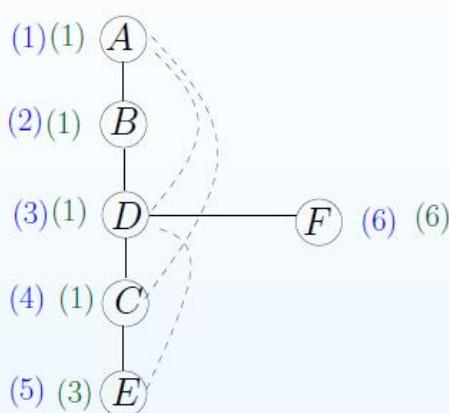
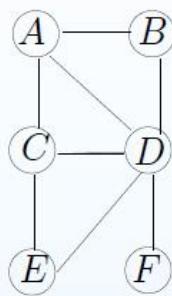
Θεωρούμε ότι έχει εκτελεστεί η ΔΚΒ και σε κάθε κορυφή v έχει αποδοθεί τιμή $preorder[v]$.

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*)

$\text{visited}[v] \leftarrow \text{true};$

$\text{low}[v] \leftarrow \text{preorder}[v];$

for $u \in N(v)$

if not $\text{visited}[u]$ **then**

comp_low(u , *visited*, *preorder*, *parent*, *low*, *cut_node*)

if $\text{low}[v] > \text{low}[u]$ **then**

$\text{low}[v] \leftarrow \text{low}[u];$

if $\text{low}[u] \geq \text{preorder}[v]$ **then**

$\text{cut_node}[v] \leftarrow \text{true};$

else /* $\text{visited}[u] = \text{true}$ */

if $u \neq \text{parent}[v]$ **then** /*(v, u) οπισθοακμή */

if $\text{low}[v] > \text{preorder}[u]$ **then**

$\text{low}[v] \leftarrow \text{preorder}[u];$

main()

$\text{clock} \leftarrow 0;$

for $v \in V$

$\text{visited}[v] \leftarrow \text{cut_node}[v] \leftarrow \text{false};$

$\text{preorder}[v] \leftarrow \text{parent}[v] \leftarrow 0;$

for $v \in V$

if not $\text{visited}[v]$ **then**

$\text{explore}(\text{clock}, v, \text{visited}, \text{preorder}, \text{parent});$

for $v \in V$ $\text{visited}[v] \leftarrow \text{false};$

for $v \in V$

if not $\text{visited}[v]$ **then**

$\text{comp_low}(v, \text{visited}, \text{preorder}, \text{parent}, \text{low}, \text{cut_node});$

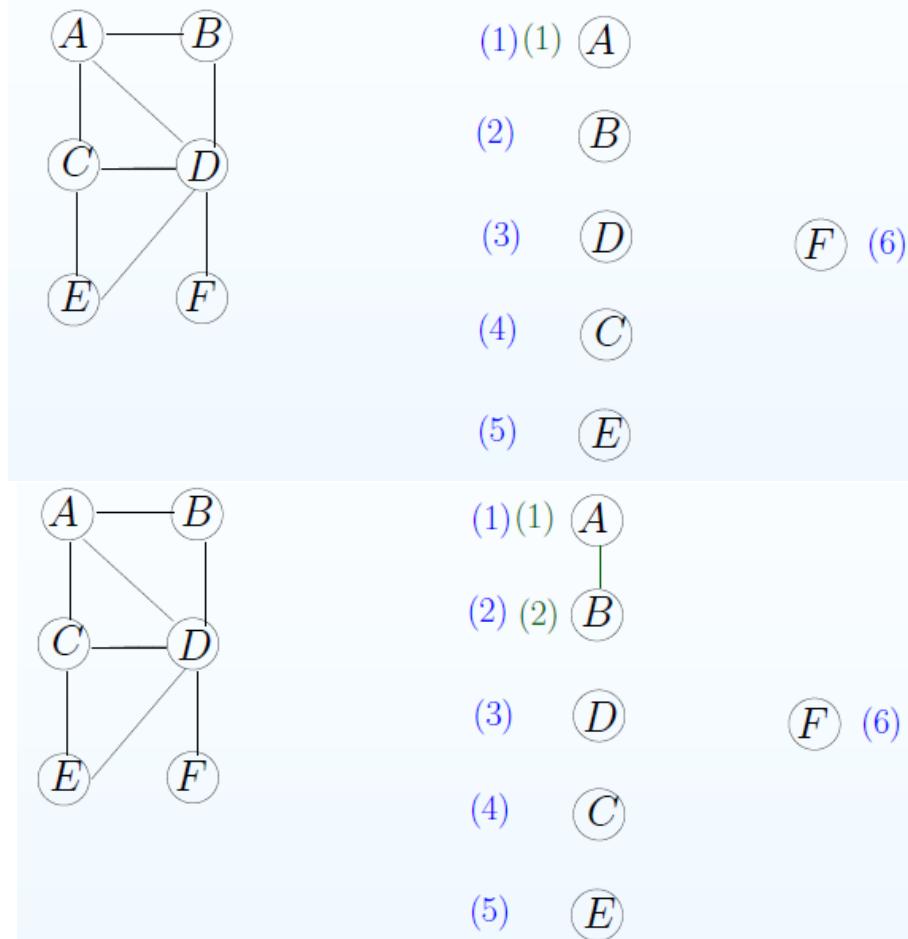
```

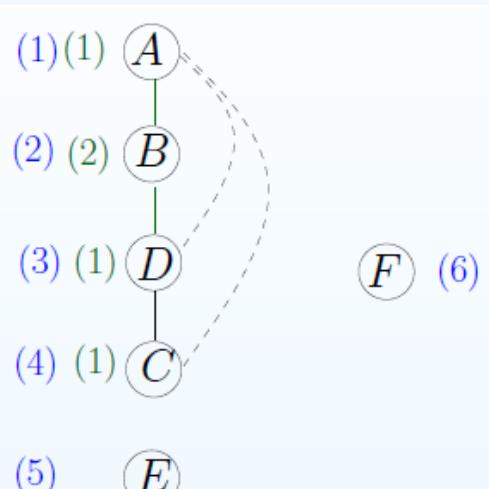
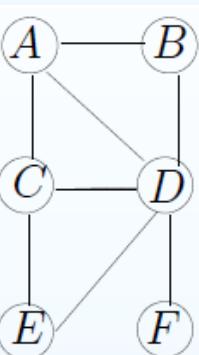
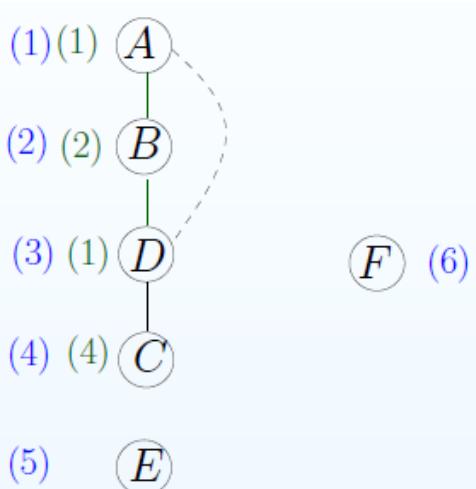
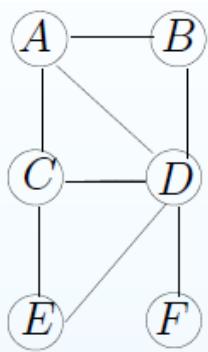
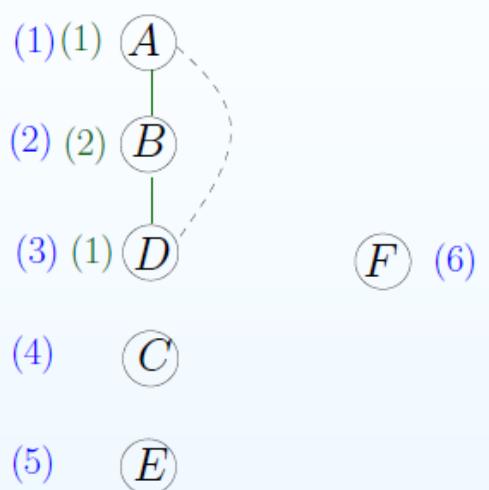
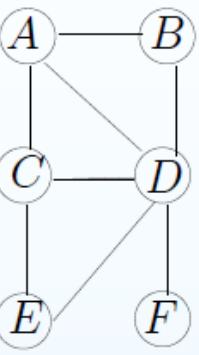
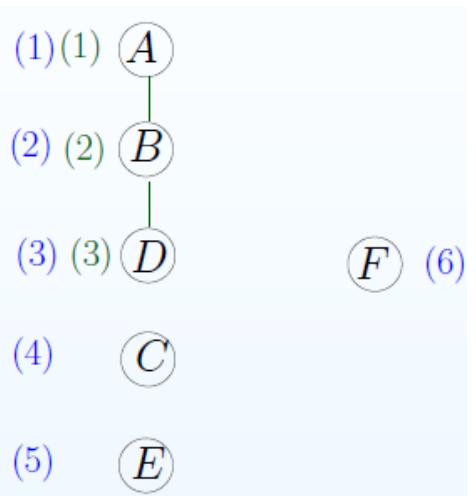
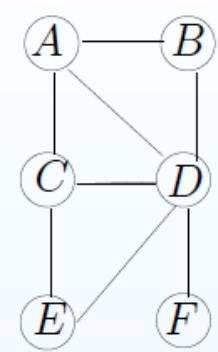
/* main continued */
for  $v \in V$ 
  if  $cut\_node[v] \neq 0$  then
    if  $parent[v] \neq 0$  then
      print( $v$ );
    else /*κορυφή  $v$  είναι ρίζα της ΔκΒ*/
       $num\_root\_children \leftarrow 0$ ;
      for  $u \in N(v)$ 
        if  $parent[u] = v$  then  $num\_root\_children ++$ ;
        if  $num\_root\_children \geq 2$  then
          print( $v$ );

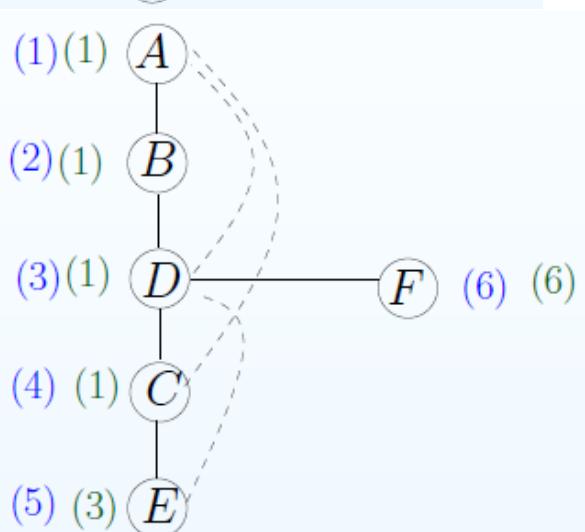
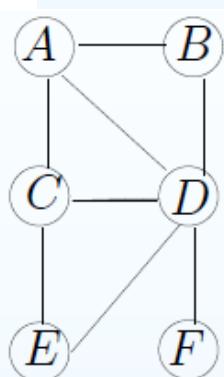
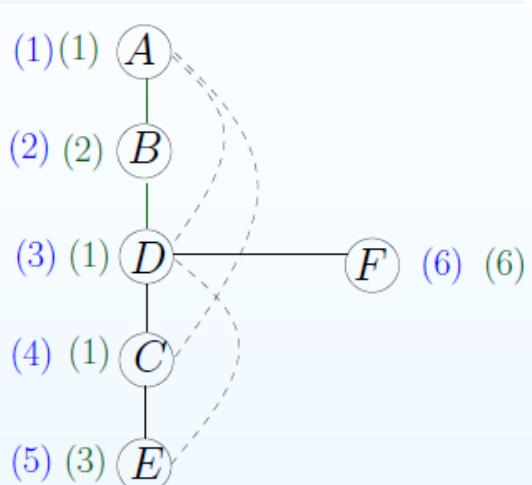
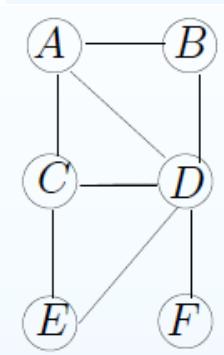
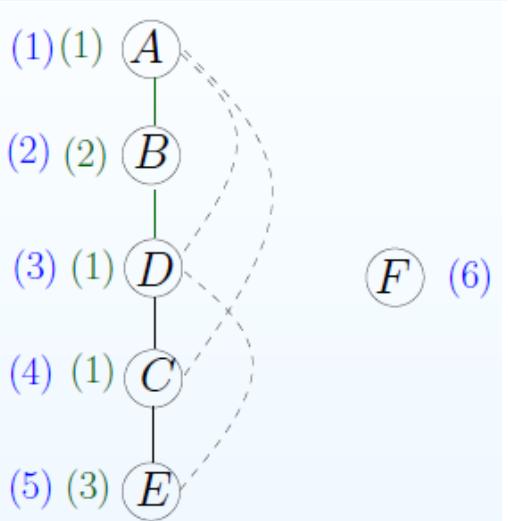
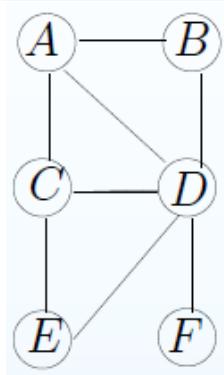
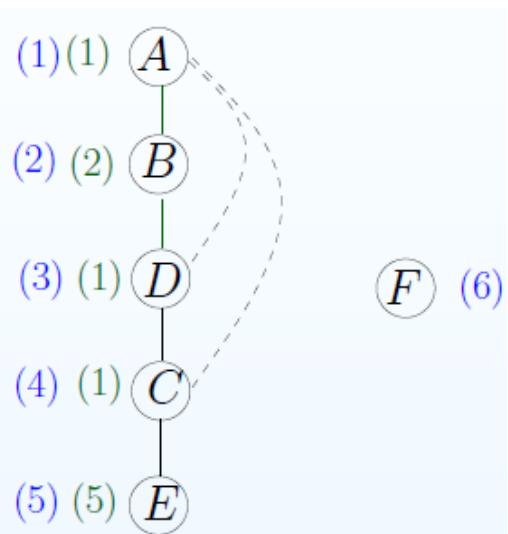
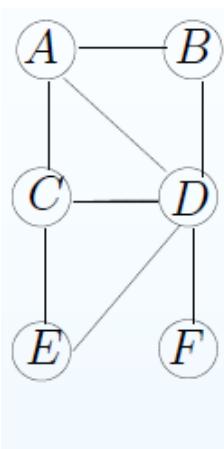
```

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







Επειδή

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

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.

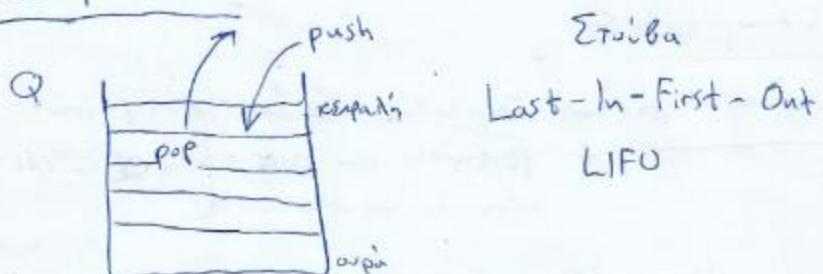
Μία ακμή (v, u) ($v = parent[u]$) είναι γέφυρα ANN

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

(γιατί ;)

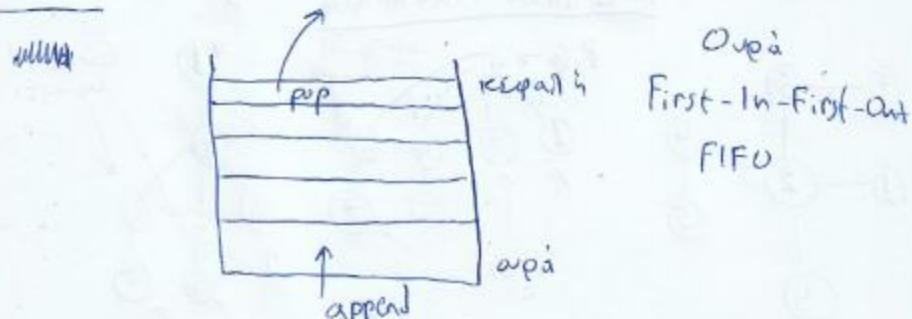
Σημειώσεις

ΔεΒ με στοίβα



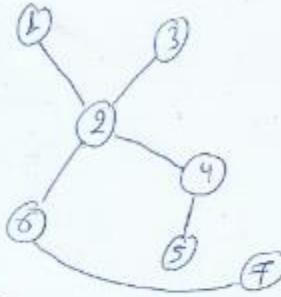
- Η υποριτίνα push βάζει στην κεφαλή της στοίβας την κορυφή που δεν έχει επικεντρωθεί στη ΔεΒ
- Η υποριτίνα pop βγάζει από την κεφαλή της στοίβας την κορυφή που έχει επικεντρωθεί στη ΔεΒ

ΔΚΠ



- Η υποριτίνα append βάζει στην ορά ορά της ουράς την κορυφή που δεν έχει επικεντρωθεί στη ΔΚΠ
- Η υποριτίνα pop βγάζει από την ορά κεφαλή της ουράς την κορυφή που έχει επικεντρωθεί στη ΔΚΠ

Πίνακας πατέντ



Παίρνω τη γράμμη
την κορυφή $\underline{\textcircled{2}}$
ως αριθμητική¹
κορυφή

$$P = [\textcircled{1} \textcircled{2} \textcircled{3} \textcircled{4} \textcircled{5} \textcircled{6} \textcircled{7}]$$

$$P = [2, , , , , ,]$$

$$P = [2, \times, , , , ,]$$

$$P = [2, \times, 2, , , ,]$$

$$P = [2, \times, 2, 2, , ,]$$

$$P = [2, \times, 2, 2, 4, ,]$$

$$P = [2, \times, 2, 2, 4, 2,]$$

$$P = [2, \times, 2, 2, 4, 2, 6]$$

- 5 -

Chapter 3

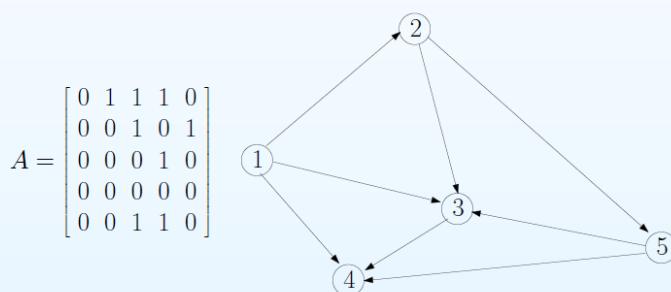
3a. ΔΚΒ σε Κατευθυνόμενα Γραφήματα, ΙΣΓΣ

Διαφάνειες

Slides_and_Exercises → 03_graphtransdir.pdf σελ. 1-43

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.

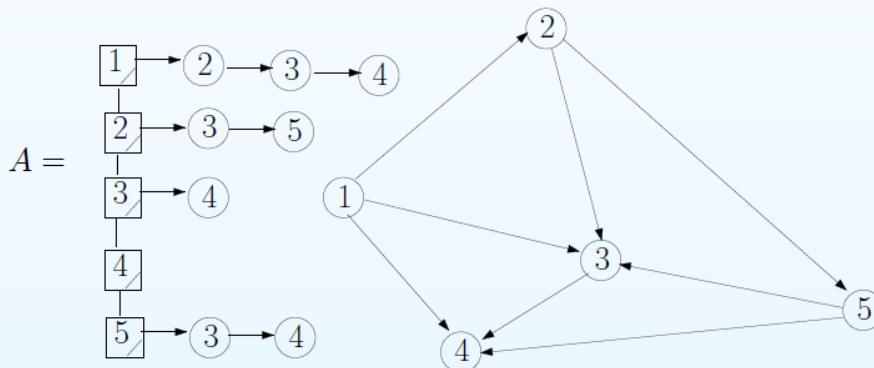


Σχήμα 1: Πίνακας Γειτνίασης

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

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

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

explore($clock, v, visited, preorder, postorder$)

```

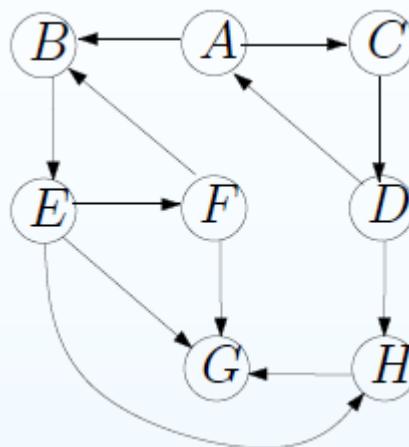
 $visited[v] \leftarrow true;$ 
 $preorder[v] \leftarrow ++clock;$ 
for  $u \in N^+(v)$ 
    if not  $visited[u]$  then
        explore( $clock, u, visited, preorder, postorder$ );
     $postorder[v] \leftarrow ++clock;$ 
```

main()

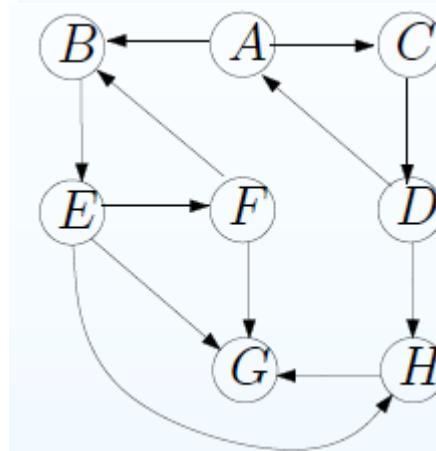
```

 $clock \leftarrow 0;$ 
for  $v \in V$ 
     $visited[v] \leftarrow false;$ 
     $preorder[v] \leftarrow 0; postorder[v] \leftarrow 0;$ 
for  $v \in V$ 
    if not  $visited[v]$  then
        explore( $clock, v, visited, preorder, postorder$ );
```

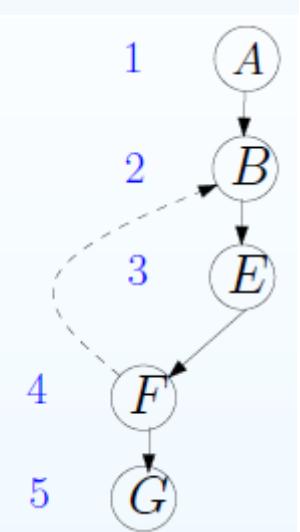
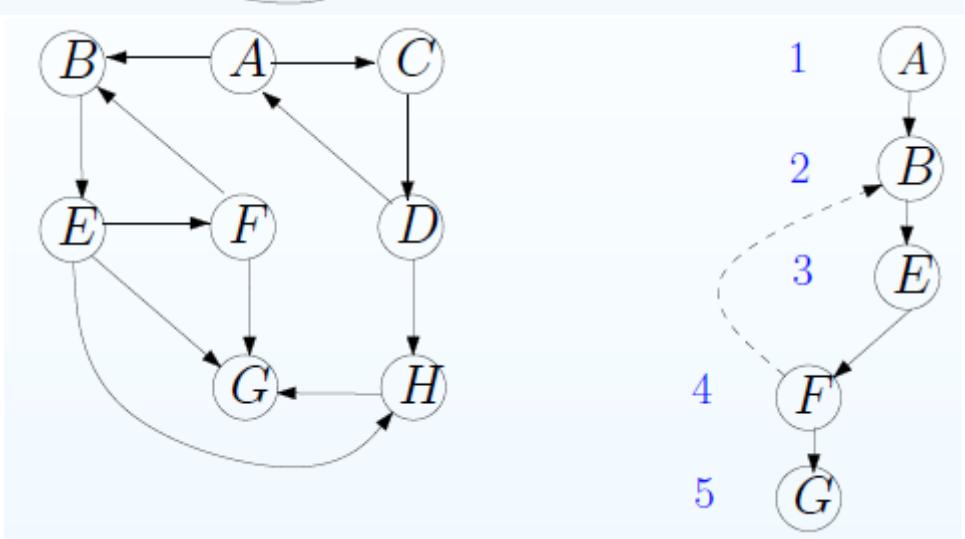
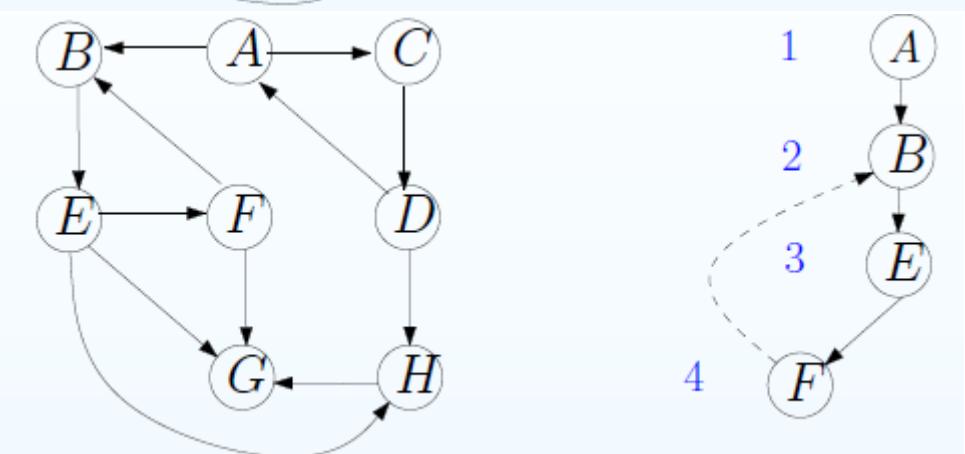
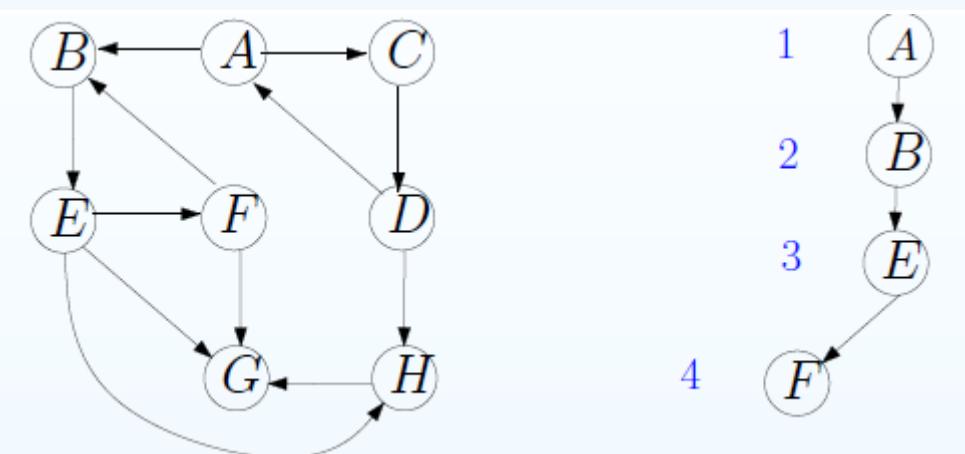
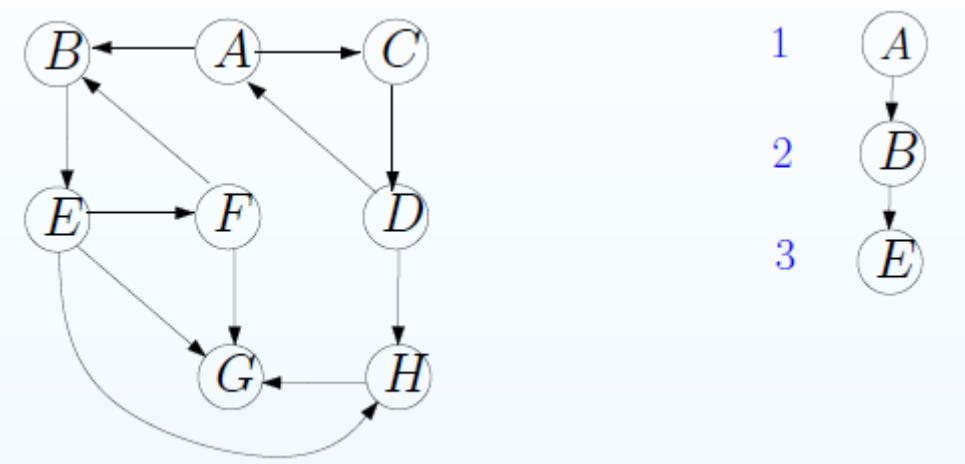
Παράδειγμα

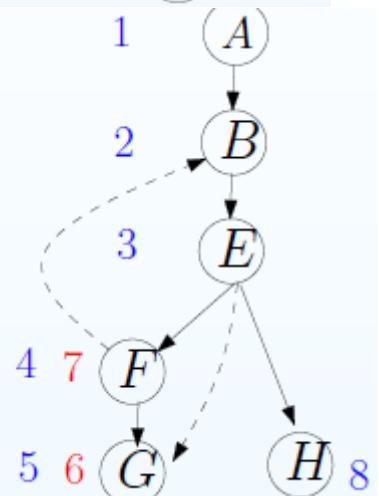
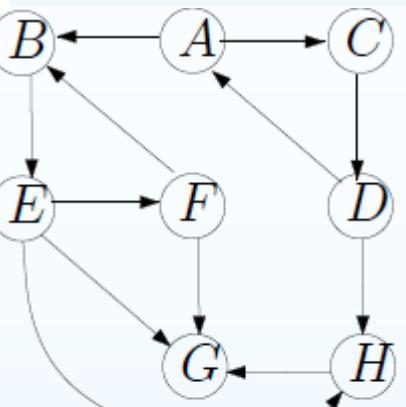
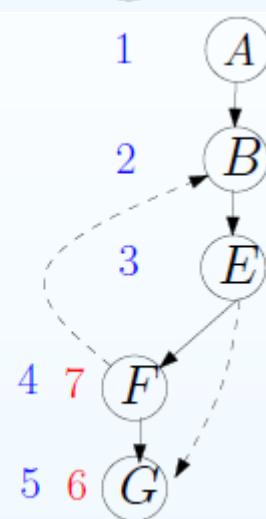
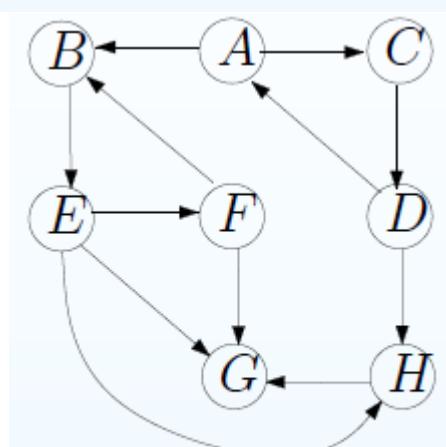
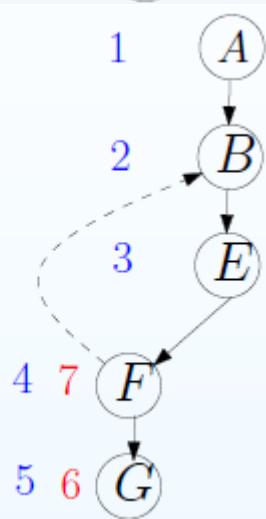
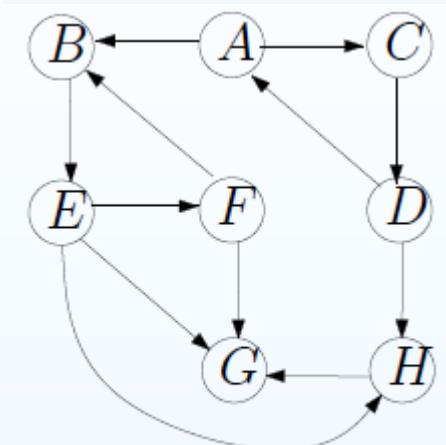
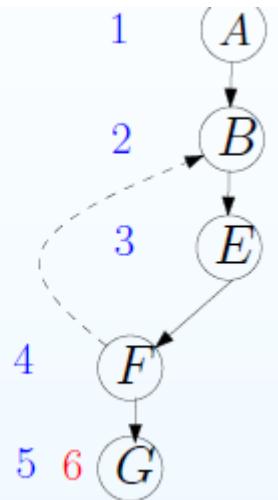
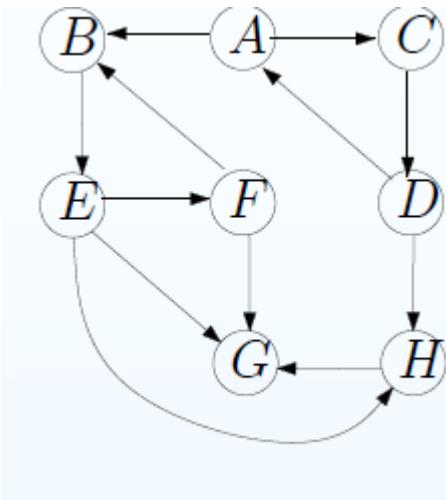


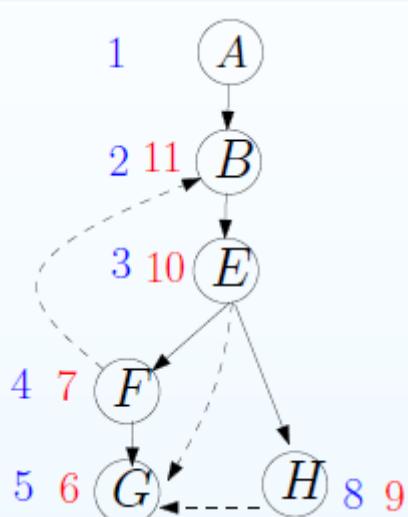
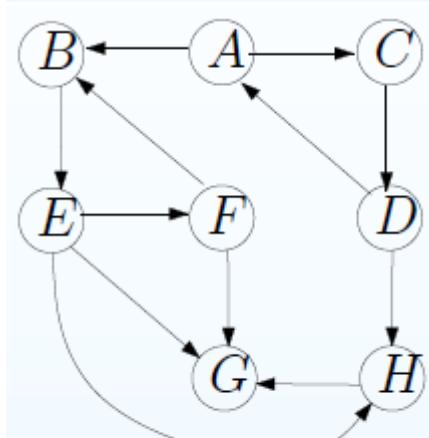
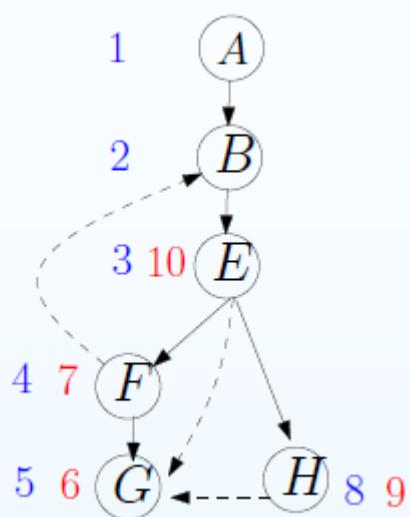
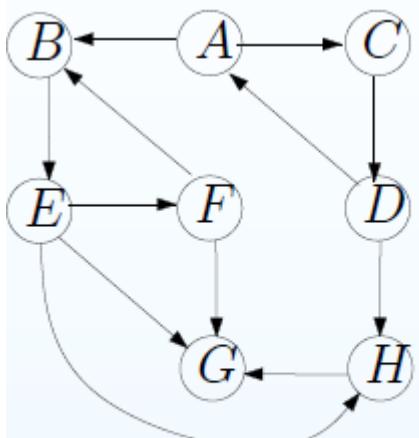
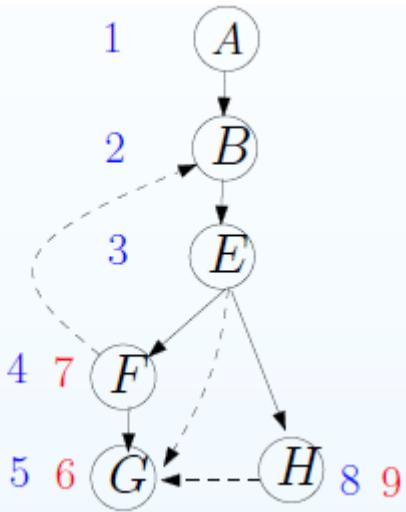
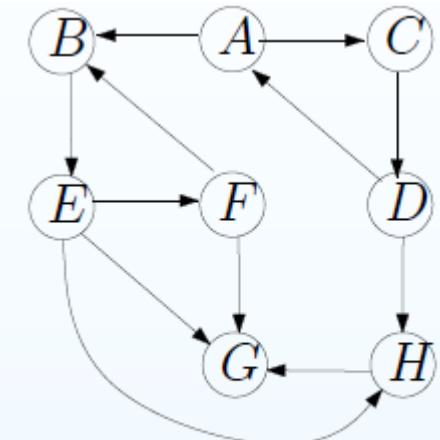
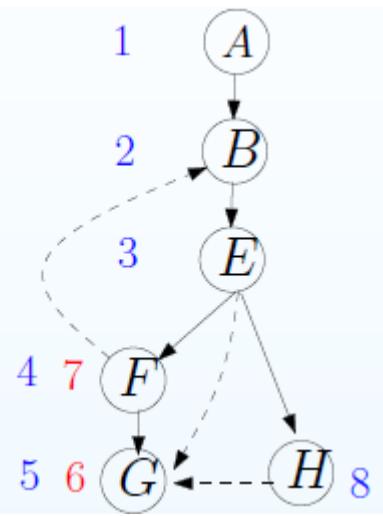
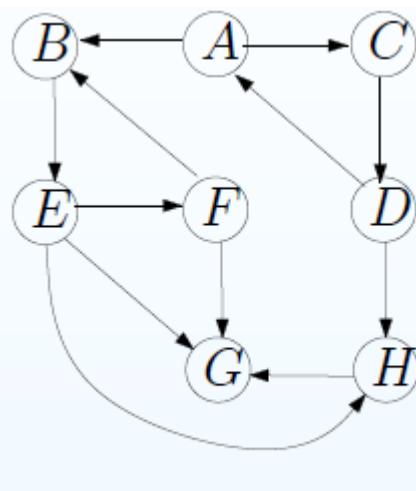
1 (A)

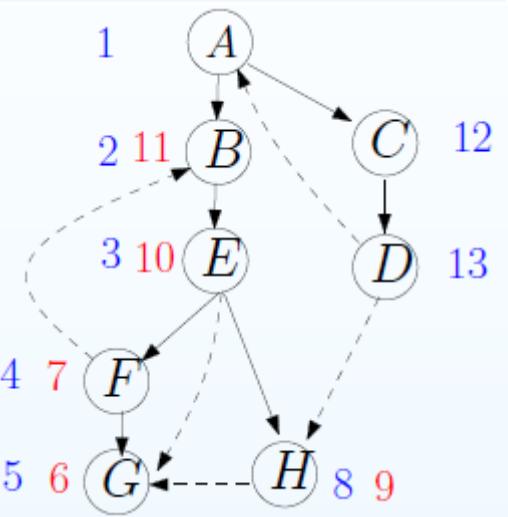
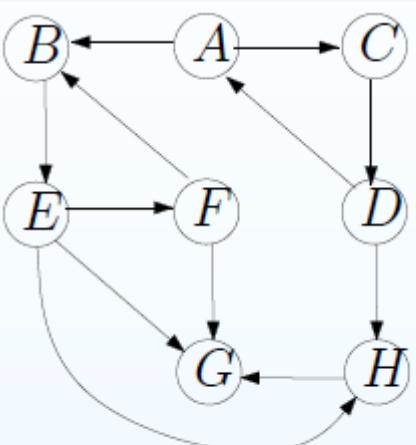
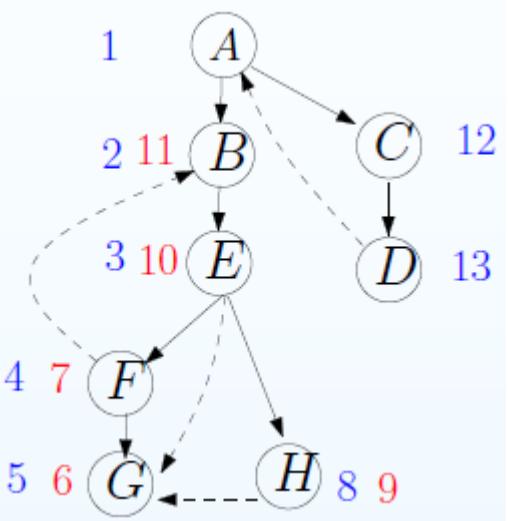
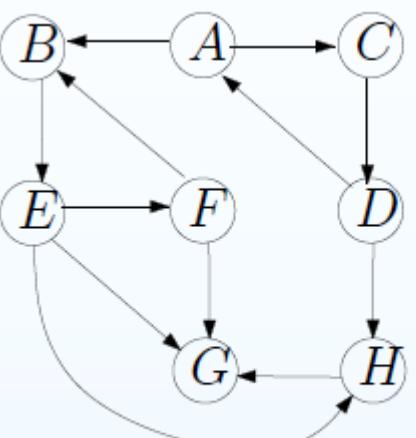
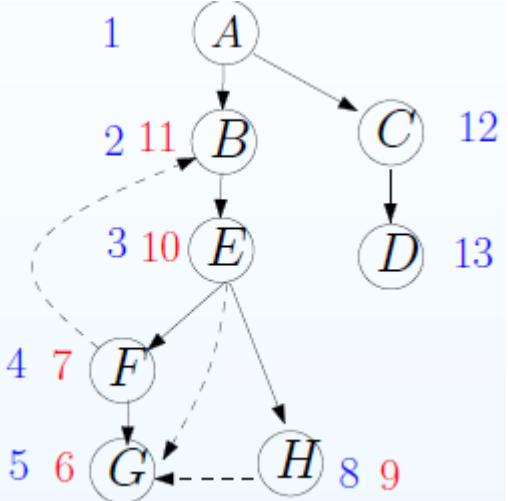
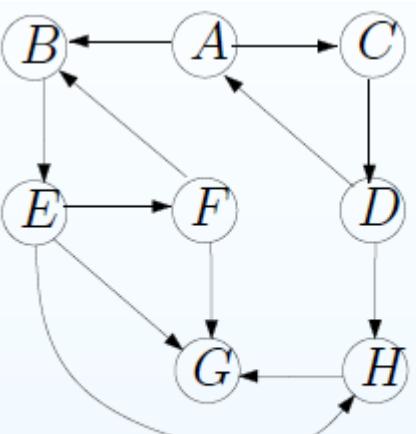
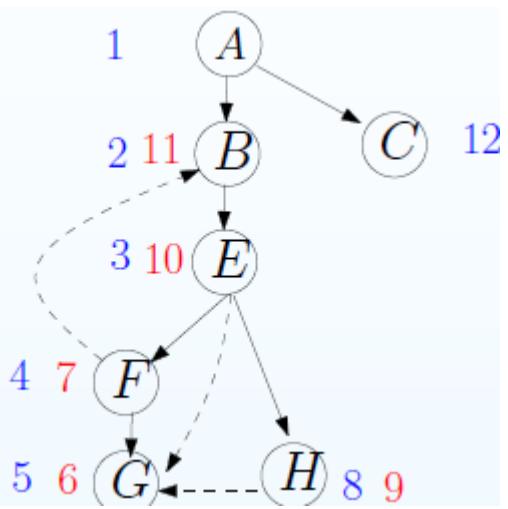
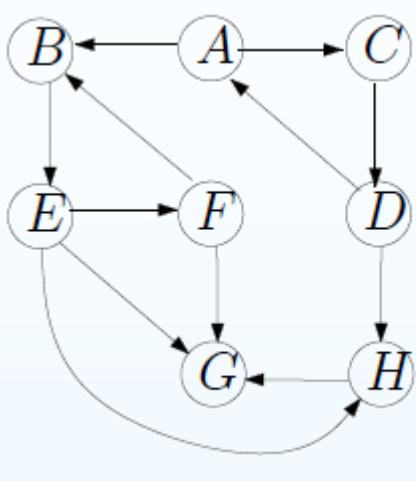


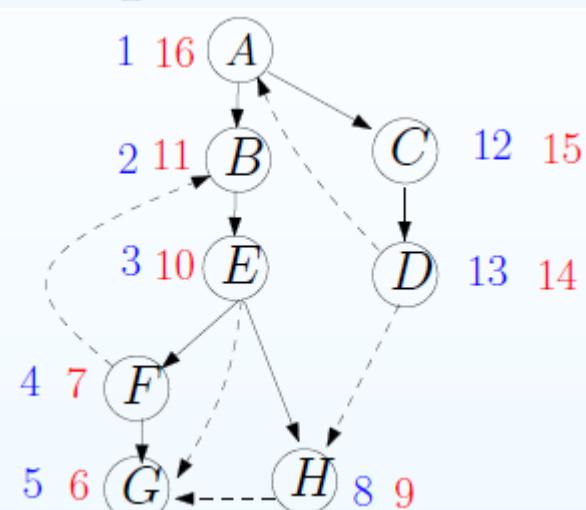
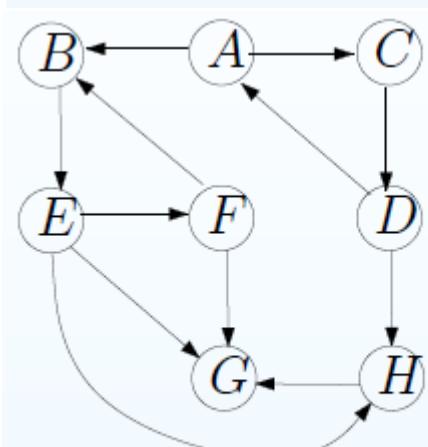
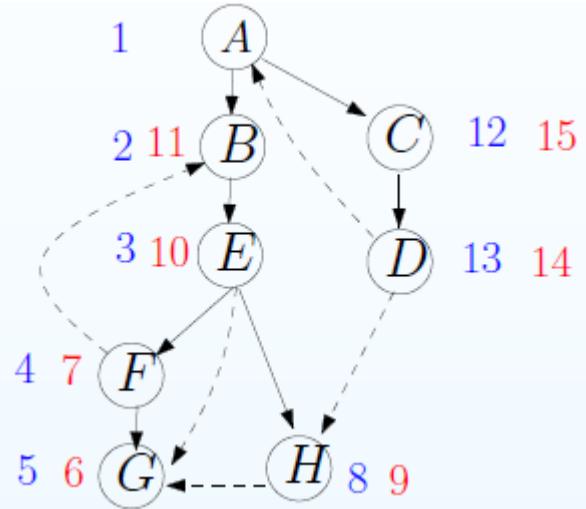
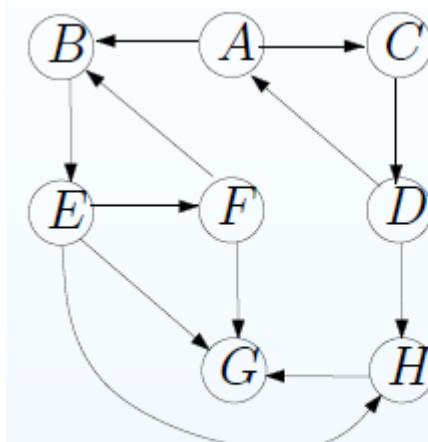
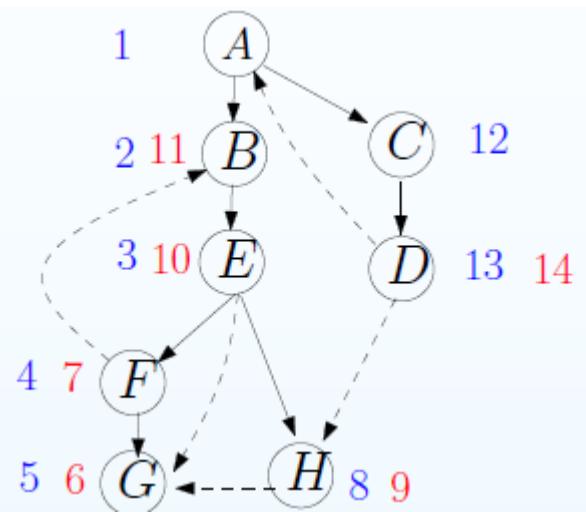
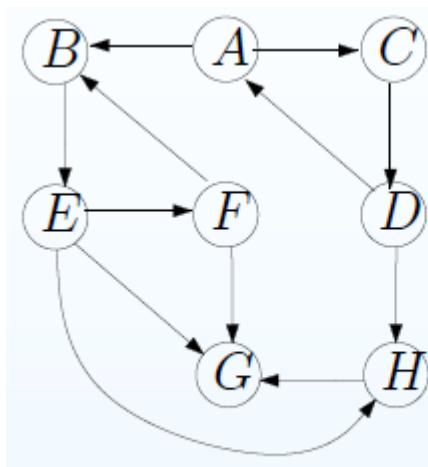
1 (A)
2 (B)











ΔικΒ - Ακμές

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 ισχύει ότι κάθε εμπροσθοακμή είναι δενδρική ακμή (δες ακμή (E, G) στο Σχήμα 5)

Παρατηρήσεις

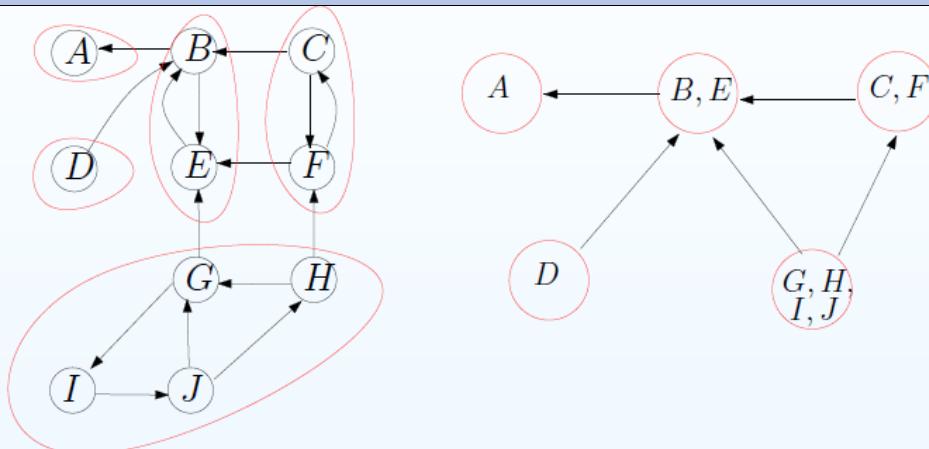
- Η ύπαρξη οπισθοακμής υποδηλώνει την ύπαρξη κατευθυνόμενου κύκλου
- Η ΔκΒ μπορεί να εντοπίσει αν υπάρχει κατευθυνόμενο μονοπάτι που να συνδέει δύο διθείσες κορυφές καθώς και κύκλο.
- Αν το γράφημα δεν έχει (κατευθυνόμενο) κύκλο τότε ονομάζεται άκυκλο κατευθυνόμενο γράφημα (ΑΚΓ).

Συνδεσιμότητα ΙΣΓΣ

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 την 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 (δεξί 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*.

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 $v \in V$ $\text{visited}[v] \leftarrow \text{false};$

while $\exists u \in V$ such that $\text{visited}[u] = \text{false}$

$v \leftarrow \text{argmax}\{\text{postorder}[u] : u \in V, \text{visited}[u] = \text{false}\};$

$Q \leftarrow \emptyset;$

explore($v, \text{visited}, Q$);

print Q ;

όπου

explore($v, \text{visited}, Q$);

$\text{visited}[v] \leftarrow \text{true};$

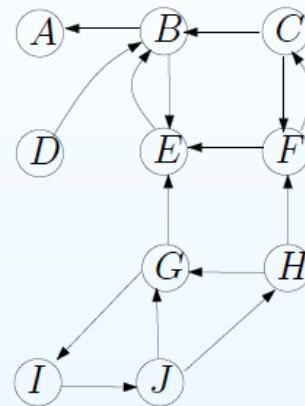
$Q \leftarrow Q \cup \{v\};$

for $u \in N^-(v)$

if not $\text{visited}[u]$ **then**

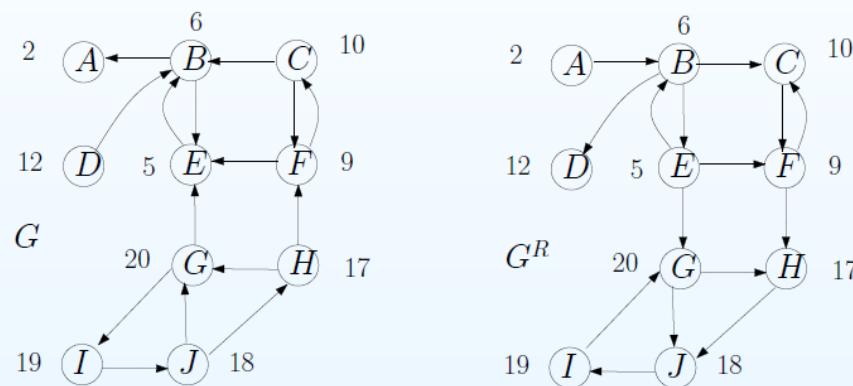
explore($v, \text{visited}, Q$);

Να βρεθούν οι ΙΣΓΣ στο γράφημα του Σχήματος 7



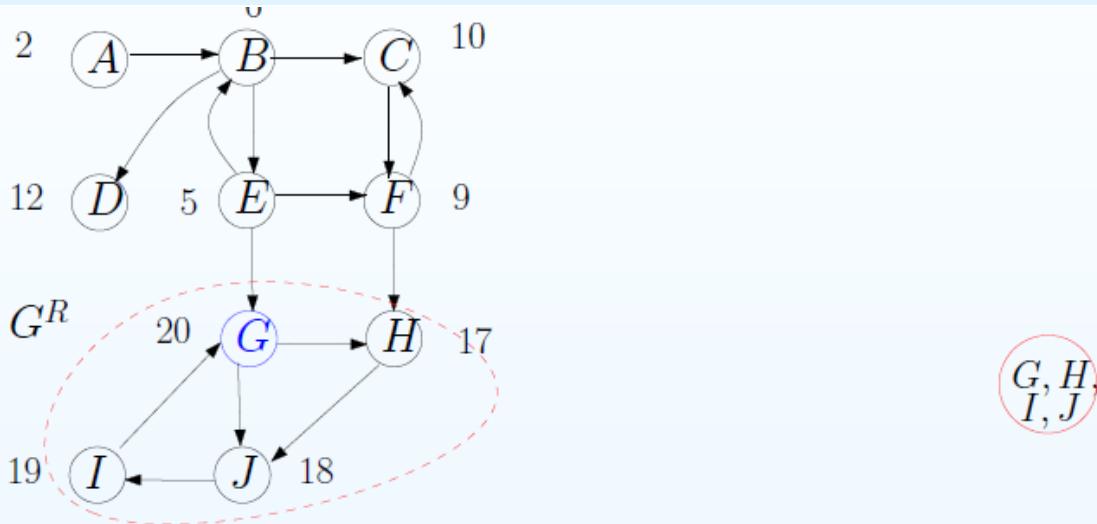
Σχήμα 7: Κατευθυνόμενο Γράφημα

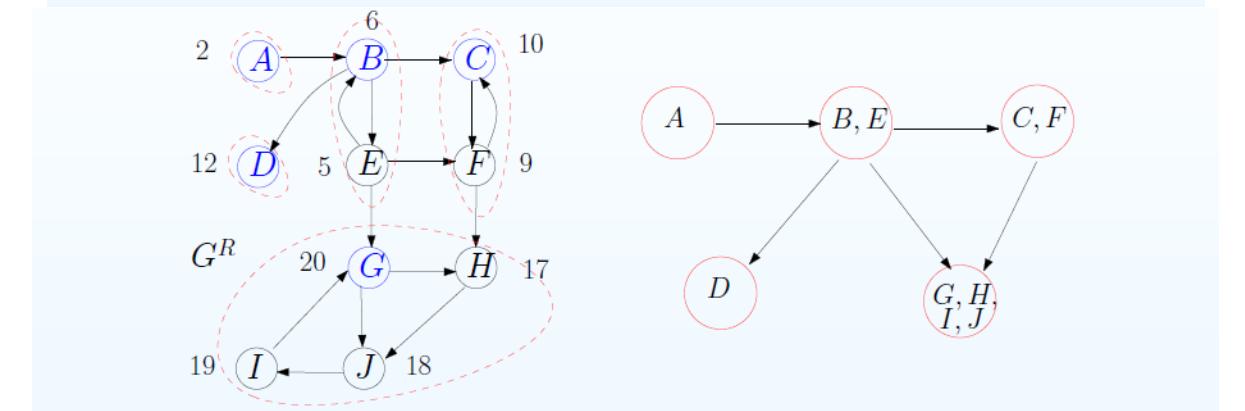
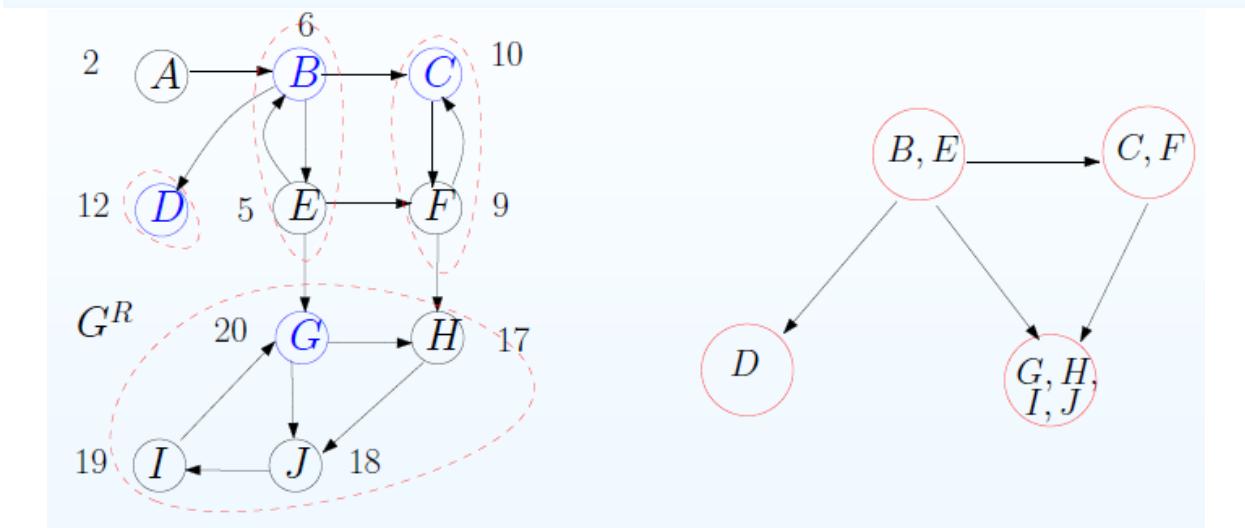
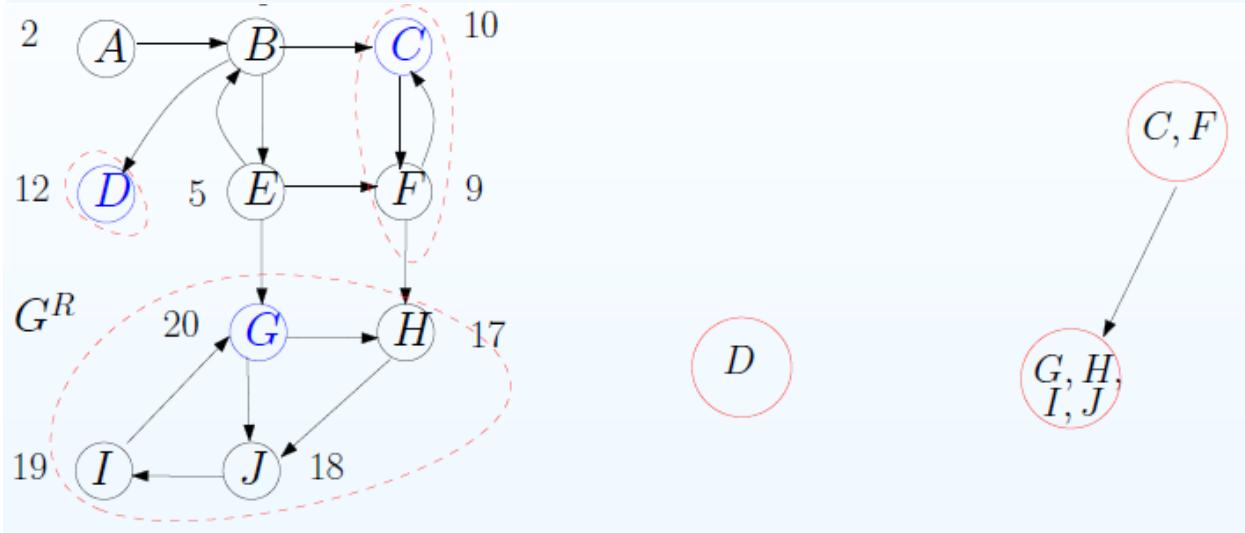
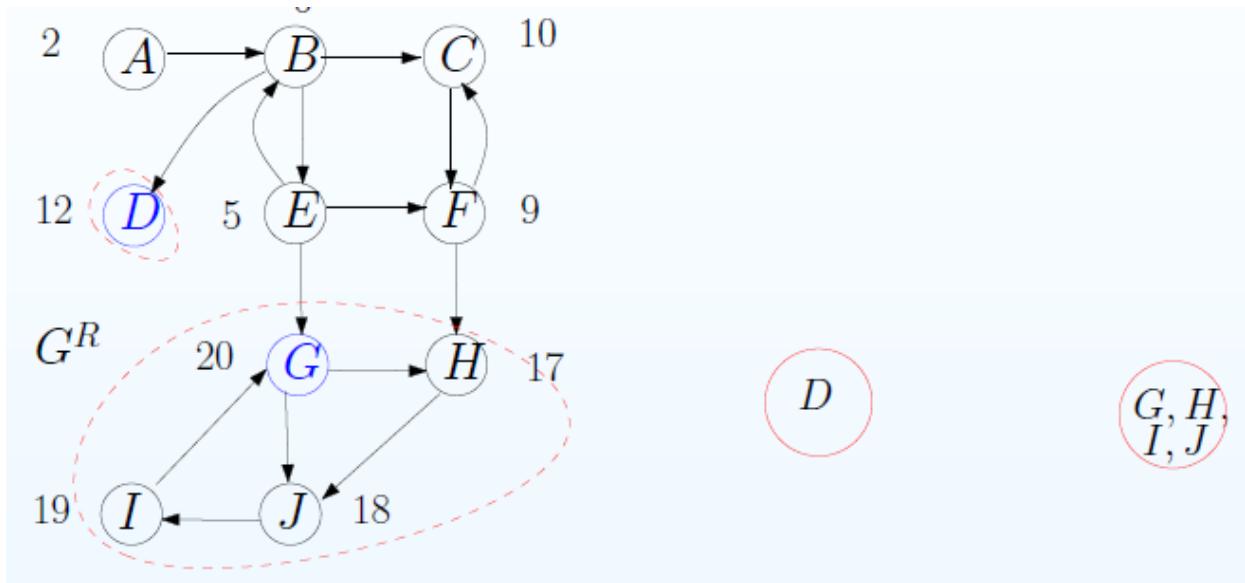
Εκτελούμε ΔΚΒ προκειμένου να υπολογιστούν τα στοιχεία του πίνακα *postorder*. Οι τιμές του πίνακα σημειώνονται δίπλα σε κάθε κορυφή στο Σχήμα 8.



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.

3b. Τοπολογική Ταξινόμηση

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 31-34

ΚΑΓ

Ένα κατευθυνόμενο γράφημα ονομάζεται **άκυκλο** αν δεν περιέχει κατευθυνόμενο κύκλο. Με τη χρήση της ΔκΒ (Άλγόριθμος 82) μπορούμε να διαπιστώσουμε αν το γράφημα είναι άκυκλο· αυτό συμβαίνει αν δεν περιέχει οπισθοακμή (Ενότητα 12.3.2). Αν λοιπόν ισχύει αυτό τότε το γράφημα ονομάζεται **κατευθυνόμενο άκυκλο γράφημα (ΚΑΓ)**.

Ιδέα

Σε ένα ΚΑΓ μπορούμε να διατάξουμε τις κορυφές του σε μία σειρά έτσι ώστε από κάθε κορυφή να υπάρχουν εξερχόμενες ακμές μόνο προς κορυφές που έπονται αυτής στη σειρά. Μία τέτοια διάταξη ονομάζεται **τοπολογική ταξινόμηση**. Με δεδομένο ότι ένα δοθέν κατευθυνόμενο γράφημα είναι άκυκλο μπορούμε να ταξινομήσουμε τοπολογικά τις κορυφές του με τη χρήση μίας στοίβας. Έστω, λοιπόν, Q μία στοίβα που μπορεί να «φιλοξενεί» κορυφές. Αυτή εμπλουτίζεται κατά τη διάρκεια της ΔκΒ ως εξής. Έστω v η πιο πρόσφατη κορυφή που έχει προστεθεί σε αυτή. Όταν ανακαλυφθεί κορυφή u γειτονική της v , εξέρχεται από τη στοίβα v , εισέρχεται η u , συνεχίζεται η ΔκΒ από τη u και όταν επιστρέψει ο έλεγχος από την (αναδρομική) διάσχιση επαναπροστίθεται στη στοίβα v . Αρχικά προστίθεται στη στοίβα μία κορυφή v η οποία δεν έχει καμία εισερχόμενη ακμή - τουλάχιστον μία τέτοια κορυφή πρέπει να υπάρχει με δεδομένο ότι το γράφημα είναι ΚΑΓ.

Άλγόριθμος

Άλγόριθμος 87 Τοπολογική Ταξινόμηση

Απαιτείται: Κατευθυνόμενο γράφημα G , πίνακας Q , ακέραιος size_Q , πίνακας visited , κορυφή v .

Επιστρέφεται: Στοίβα Q , ακέραιος size_Q . πίνακας visited .

```
1: function TOP_ORDER(graph  $G$ , int  $Q[]$ , int  $\text{size}_Q$ , logical  $\text{visited}[]$ ,  
    int  $v$ )  
2:    $\text{visited}[v] \leftarrow \text{True};$   
3:   PUSH( $Q$ ,  $\text{size}_Q$ ,  $v$ );  
4:   for all  $u \in N^+(v)$  do  
5:     if not  $\text{visited}[u]$  then  
6:        $v \leftarrow \text{POP}(Q, \text{size}_Q);$   
7:       TOP_ORDER( $G$ ,  $Q$ ,  $\text{size}_Q$ ,  $\text{visited}$ ,  $u$ );  
8:       PUSH( $Q$ ,  $\text{size}_Q$ ,  $v$ );  
9:     end if  
10:   end for  
11: end function
```

Αλγόριθμος 88 Τοπολογική Ταξινόμηση - Καλών αλγόριθμος

Απαιτείται: Κατευθυνόμενο γράφημα G .

Επιστρέφεται: Εκτύπωση των κορυφών του G σε τοπολογικά ταξινομημένη διάταξη.

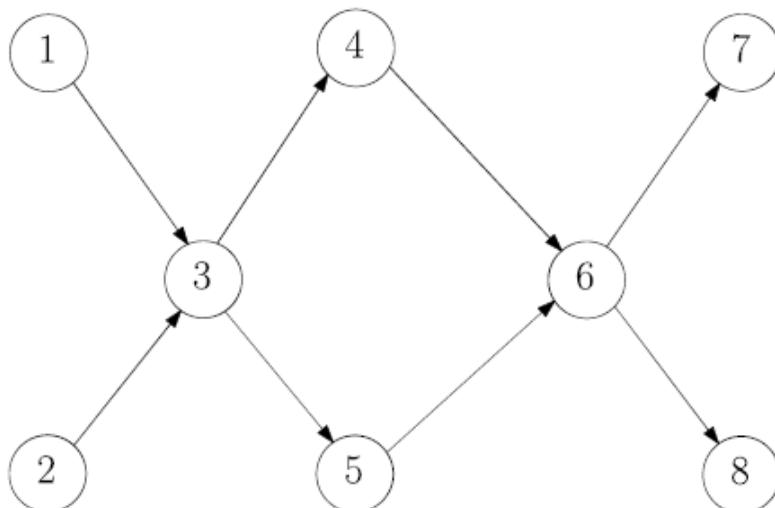
```
1: function TOP_ORDER_MAIN(graph  $G$ )
2:   CREATE_STACK( $Q$ , size_  $Q$ );
3:   for all  $v \in V(G)$  do
4:     visited[ $v$ ]  $\leftarrow$  False;
5:   end for
6:   for all  $v \in V(G)$  do
7:     if not visited[ $v$ ] and  $|N^-(v)| = 0$  then
8:       TOP_ORDER( $G, Q, size_Q, visited, v$ );
9:     end if
10:   end for
11:   while size_  $Q > 0$  do
12:     print POP( $Q, size_Q$ );
13:   end while
14: end function
```

Λειτουργία

Η λειτουργία που περιγράφηκε παραπάνω πραγματοποιείται από τη συνάρτηση TOPORDER (Αλγόριθμος 87) η οποία καλείται από τον Αλγόριθμο 88. Μετά την ολοκλήρωση των κλήσεων της συνάρτησης TOPORDER οι κορυφές του γράφηματος έχουν αποθηκευτεί στη στοίβα Q κατά τρόπο ώστε, κάθε κορυφή v έχει εξερχόμενες ακμές προς κορυφές που βρίσκονται κάτω από αυτή στη στοίβα και εισερχόμενες ακμές από κορυφές που βρίσκονται πάνω από αυτή. Δηλαδή οι κορυφές έχουν τοποθετηθεί στη στοίβα τοπολογικά ταξινομημένες. Στη συνέχεια, στα πλαίσια του Αλγόριθμου 88 γίνεται εκτύπωση των περιεχομένων της στοίβας.

Παράδειγμα

Παράδειγμα 70. Στο Σχήμα 12.7 απεικονίζεται ένα ΚΑΓ. Στην εκτέλεση των αλ-



Σχήμα 12.7: Ένα κατευθυνόμενο άκυκλο γράφημα - Παράδειγμα 70

γορίθμων θεωρούμε τα στοιχεία των συνόλων των κορυφών $N^+(v)$ και $V(G)$ σε αύξουσα σειρά. Έτσι ο Αλγόριθμος 88 καλεί μία φορά την συνάρτηση `TOPORDER` με $v = 1$ και στη συνέχεια με $v = 2$. Η πρώτη κλήση πυροδοτεί μία σειρά αναδρομικών κλήσεων η οποία απεικονίζεται στο Σχήμα 12.8. Στο ίδιο σχήμα παρουσιάζεται η εικόνα της στοίβας Q στο τέλος κάθε κλήσης.

Στη δεύτερη κλήση της `TOPORDER` απλώς προστίθεται η κορυφή 2 στην κεφαλή της στοίβας. Έτσι μετά την ολοκλήρωση των αναδρομικών κλήσεων η στοίβα έχει `TOP_ORDER(v=1)`

`TOP_ORDER(v=3)`

`TOP_ORDER(v=4)`

`TOP_ORDER(v=6)`

`TOP_ORDER(v=7)`

$$Q = \{7\}$$

`TOP_ORDER(v=8)`

$$Q = \{8\}$$

$$Q = \{6\}$$

$$Q = \{4\}$$

`TOP_ORDER(v=5)`

$$Q = \{5\}$$

$$Q = \{4\}$$

$$Q = \{3\}$$

$$Q = \{5\}$$

$$Q = \{4\}$$

$$Q = \{3\}$$

$$Q = \{6\}$$

$$Q = \{7\}$$

$$Q = \{1\}$$

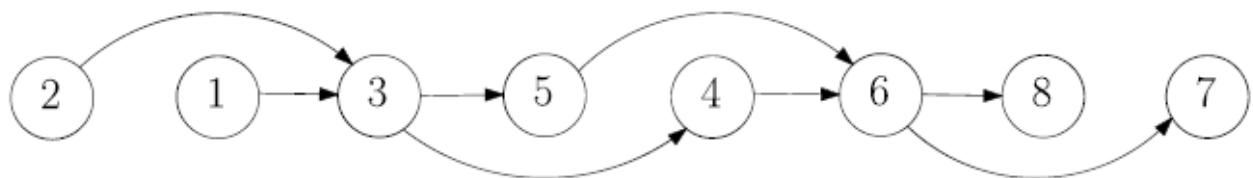
την εικόνα

$$Q = \begin{vmatrix} 2 \\ 1 \\ 3 \\ 5 \\ 4 \\ 6 \\ 8 \\ 7 \end{vmatrix}.$$

Επομένως ο Αλγόριθμος 88 θα εκτυπώσει τις κορυφές με τη σειρά

2 1 3 5 4 6 8 7

Το Σχήμα 12.9 απεικονίζει το γράφημα του Σχήματος 12.7 με τις κορυφές διατεταγμένες με την παραπάνω σειρά. Έτσι φαίνεται και εποπτικά ότι η παραπάνω σειρά είναι τοπολογικά ταξινομημένη.



Σχήμα 12.9: Εναλλακτική απεικόνιση του γραφήματος του Σχήματος 12.7 - Παράδειγμα 70

Είναι σημαντικό να παρατηρήσουμε ότι δεν είναι απαραίτητο η τοπολογικά ταξινομημένη σειρά να είναι μοναδική. Για παράδειγμα εκτός από την παραπάνω διάταξη, οι σειρές

2 1 3 4 5 6 7 8

1 2 3 5 4 6 8 7

είναι επίσης τοπολογικά ταξινομημένες.

(ΚΑΓ)

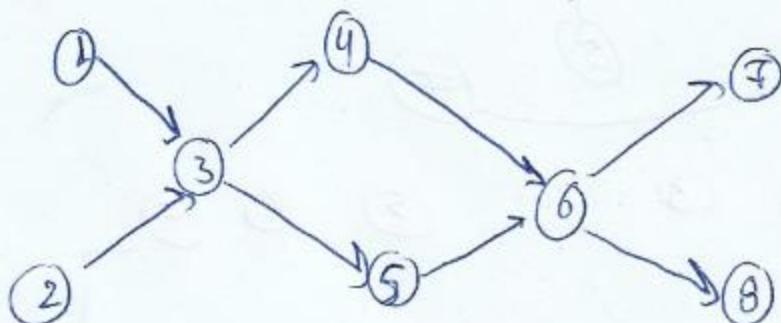
Κατευθύνσεων Ακύλωτο Γράφημα

03_ΚΑΤΕΥΘΥΝΩΜΕΝΑ_ΓΡΑΦΗΜΑΤΑ

24/10/2023

Chapter 3b

~~ΣΟΣ~~ ② Θα δώσω Τοπολογική ταξίδια και στον Διαχρόνιο Προγραμματισμό ΣΤΟΝΙΖΟΥΧΙ Ταξιδιώματα : Γράψω τις καρυφές ώστε να υπάρχουν εξερχόμενες ακριβείς μιαν πριν καρυφές τιν είναιται αυτής στη διεύθυνση. Ισχύει μιαν για ΚΑΓ



Chapter 4

4a. ΔκΒ σε Μη-Κατευθυνόμενο Γράφημα

Διαφάνειες

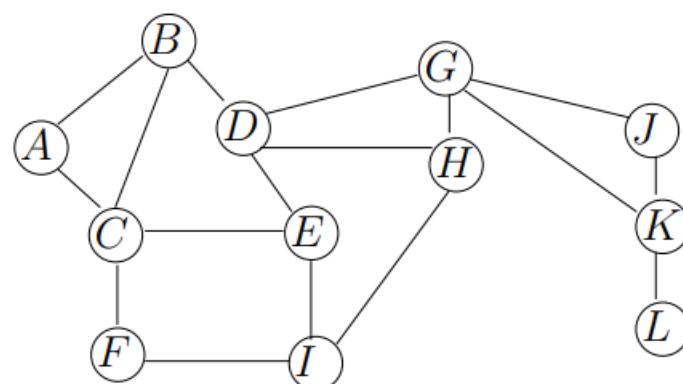
Slides_and_Exercises→04_graphtranser.pdf σελ. 1-5

Εκφώνηση

Άσκηση 1 Για το γράφημα που απεικονίζεται στο Σχήμα 1 να απεικονίσετε το γενυνητορικό (συνδετικό) δένδρο που προκύπτει από τη ΔκΒ αν στη λίστα γειτνίασης οι καρυφές εμφανίζονται σε μλεξικογραφική σειρά. Επίσης να υπολογίσετε τους αριθμούς προδιάταξης κάθε καρυφής καθώς και τις οπισθοακμές που προκύπτουν από τη διάσχιση.

Στη συνέχεια να εκπονήσετε αλγόριθμο ο οποίος να υπολογίζει τη λίστα γειτνίασης του γενυνητορικού δένδρου που υπολόγισες η ΔκΒ.

Σχήμα 1. Γράφημα



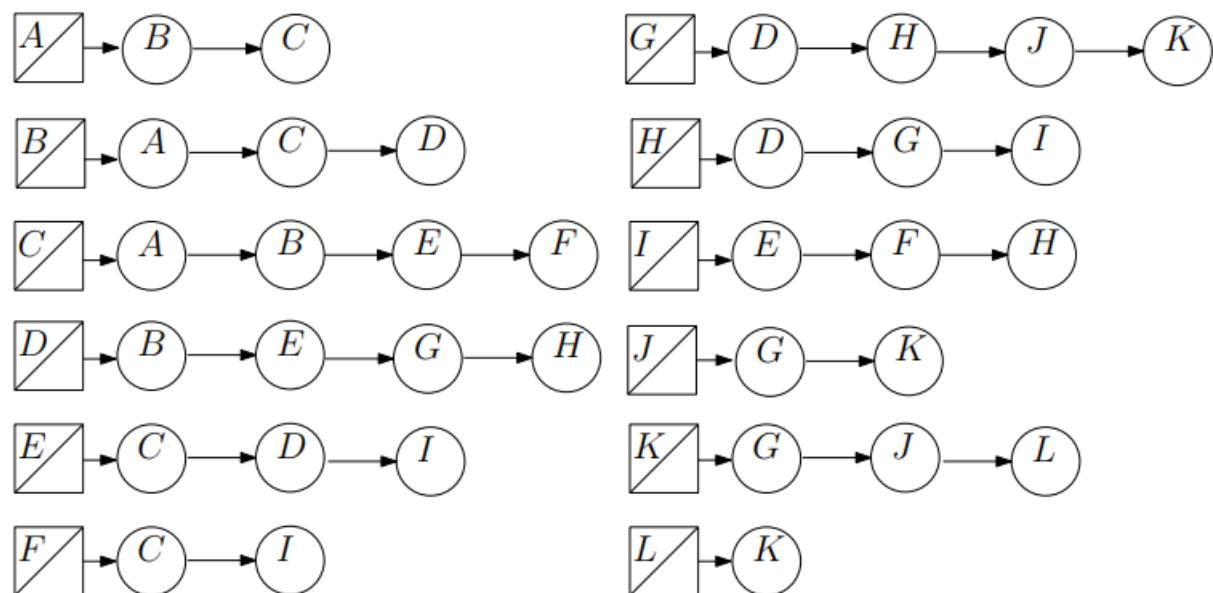
Λύση

Λύση Η λίστα γειτνίασης όπου για κάθε κορύφη οι γειτονικές κορυφές εμφανίζονται σε λεξικογραφική σειρά απεικονίζεται στο Σχήμα 2. Το γεννητορικό δένδρο της ΔκΒ όταν εκκινεί από την κορυφή A απεικονίζεται στο Σχήμα 3. Οι οπισθοακμές σημειώνονται με διακεκομένες γραμμές, οι αριθμοί προδιάταξης με μπλε και τα στοιχεία του πίνακα $parent$ με κόκκινο.

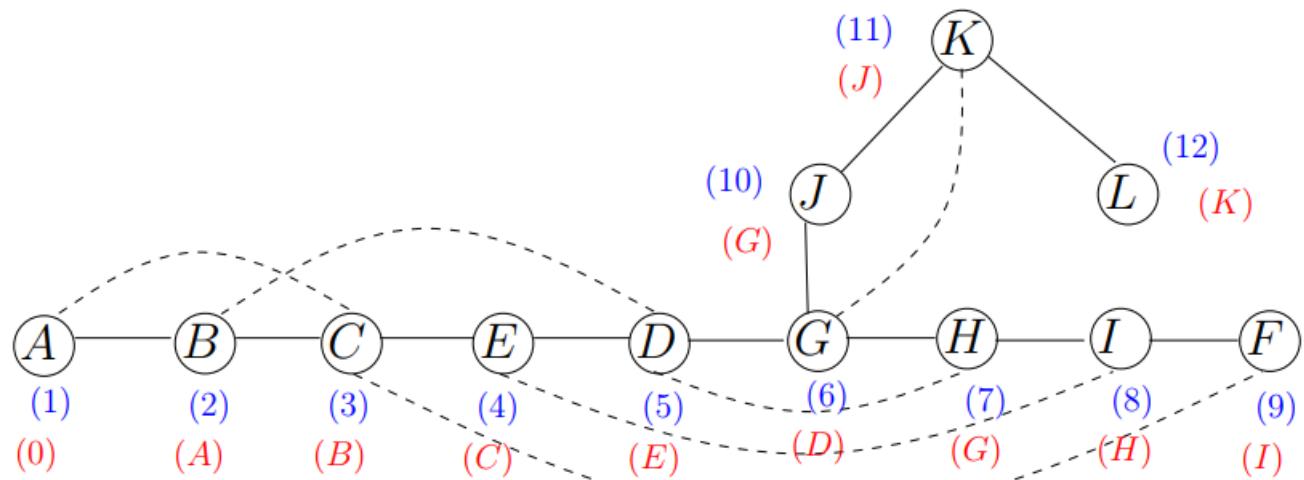
Για να μπορέσουμε να απεικονίσουμε το γεννητορικό δένδρο που υπολογίστηκε προηγουμένως εν είδη λίστας γειτνίασης, αρκεί να υπολογίσουμε για κάθε κορυφή v το σύνολο $N_T(v)$ των γειτόνων της στο δένδρο αυτό. Προφανώς $N_T(v) \subseteq N(v)$. Εκκινώντας από ένα άδειο $N_T(v)$, προσθέτουμε σε αυτό κάθε κορυφή $u \in N(v)$ για την οποία ισχύει ότι είτε $v = parent[u]$ ή $u = parent[v]$. Σε οποιαδήποτε από τις δύο περιπτώσεις η ακμή $\{v, u\}$ ανήκει στο συνδετικό δένδρο που υπολόγισε η ΔκΒ. Ο αλγόριθμος 1 υλοποιεί αυτή την ιδέα. Θα πρέπει να έχει πρώτα εκτελεστεί η ΔκΒ προκειμένου το διάνυσμα $parent$ να έχει ενημερωθεί.

Τα σύνολα N_T με στοιχεία λεξικογραφικά ταξινομημένα για το δένδρο του Σχήματος 3 απεικονίζονται στο Σχήμα 4.

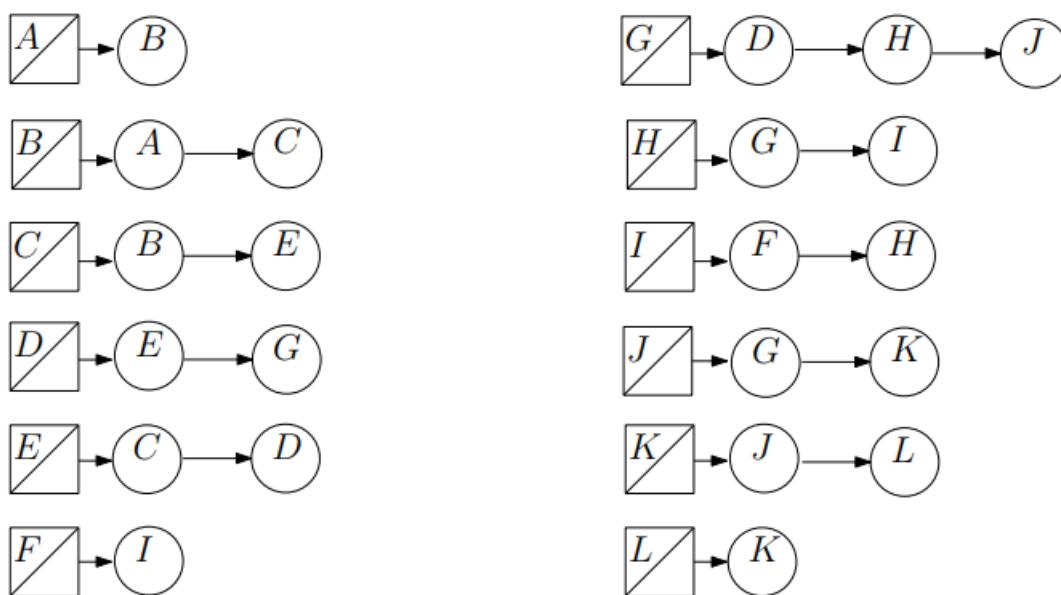
Σχήμα 2. Λίστα γειτνίασης λεξικογραφικά ταξινομημένη



Σχήμα 3. ΔκΒ για το γράφημα του Σχήματος 1



Σχήμα 4. Αναπαράσταση συνδετικού δένδρου ΔκΒ Σχήματος 3 με λίστα γειτνίασης



Αλγόριθμος 1. Λίστα γειτνίασης δένδρου ΔκΒ

Algorithm 1 Λίστα γειτνίασης δένδρου ΔκΒ

Require: $G(V, E)$, $\text{parent}[]$

```
1: void spanning_tree_dfs(int  $\text{parent}[]$ , set  $N_T[]$ )
2: for  $v \in V$  do
3:    $N_T[v] \leftarrow \emptyset$ ;
4: end for
5: for  $v \in V$  do
6:   if  $\text{parent}[v] \neq 0$  then
7:      $N_T[v] \leftarrow N_T[v] \cup \{\text{parent}[v]\}$ ;
8:      $N_T[\text{parent}[v]] \leftarrow N_T[\text{parent}[v]] \cup \{v\}$ ;
9:   end if
10: end for
```

4b. Μονοπάτι μεταξύ δύο κορυφών (ΔκΒ σε Μη-Κατευθυνόμενο Γράφημα)

Διαφάνειες

Slides_and_Exercises → 04_graphtransexer.pdf σελ. 1-6

Εκφώνηση

Άσκηση 2 Να διατυπώσετε μία έκδοση της ΔκΒ που να δέχεται σαν είσοδο δύο κορυφές s, t ενώ οι μεταβλητές $length$ (μήκος μονοπατιού) και $found$ αρχικοποιούνται στις τιμές 0 και $false$ αντίστοιχα. Αν η συνάρτηση επιστρέψει με τιμή $true$ στη μεταβλητή $found$ τότε στον πίνακα $path$ υπάρχουν αποθηκευμένες οι κορυφές του μονοπατιού ενώ το μήκος του (αριθμός ακμών του) είναι η τιμή $length - 1$; Ο Αλγόριθμος 2 μπορεί να εκτελεστεί με τις λίστες γειτνίασης που έχει παράγει η ΔκΒ. Δηλαδή στη γραμμή 9 αντι των συνόλων $N(v)$ να χρησιμοποιηθούν τα σύνολα $N_T(v)$.

Λύση

Λύση Η ζητούμενη υλοποίηση εμφανίζεται στον Αλγόριθμο 2. Τα δύο πρώτα ορίσματα της συνάρτησης αντιστοιχούν στις κορυφές s, t , ενώ οι μεταβλητές $length$ (μήκος μονοπατιού) και $found$ αρχικοποιούνται στις τιμές 0 και $false$ αντίστοιχα. Αν η συνάρτηση επιστρέψει με τιμή $true$ στη μεταβλητή $found$ τότε στον πίνακα $path$ υπάρχουν αποθηκευμένες οι κορυφές του μονοπατιού ενώ το μήκος του (αριθμός ακμών του) είναι η τιμή $length - 1$; Ο Αλγόριθμος 2 μπορεί να εκτελεστεί με τις λίστες γειτνίασης που έχει παράγει η ΔκΒ. Δηλαδή στη γραμμή 9 αντι των συνόλων $N(v)$ να χρησιμοποιηθούν τα σύνολα $N_T(v)$.

Η αναζήτηση μονοπατιού από τον Αλγόριθμο 2, που συνδέει τις κορυφές C, D στο γράφημα του Σχήματος 1, χρησιμοποιώντας τη λίστα γειτνίασης του Σχήματος 3, περιλαμβάνει τα βήματα που απεικονίζονται στο Σχήμα 5.

Αλγόριθμος 2. ΔκΒ – Εύρεση μονοπατιού s, t

Algorithm 2 ΔκΒ - Εύρεση μονοπατιού s, t

Require: G(V, E);

```
1: void find_path(int v, int t, bool visited[], int path[], int length, bool found)
2: visited[v] ← true;
3: length++;
4: path[length] ← v;
5: if v = t then
6:   found ← true;
7: end if
8: if not found then
9:   for u ∈ N(v) do
10:    if not visited[u] and not found then
11:      find_path(u, t, visited, path, length, found);
12:      if found then
13:        break;
14:      end if
15:    end if
16:  end for
17: end if
18: if not found then
19:   // ο κόμβος v δεν ανήκει στο μονοπάτι από το s στο t και άρα θα πρέπει να αφαιρεθεί (προστέθηκε στη γραμμή 3) //
20:   path[length] ← 0; length--;
21: end if
```

Σχήμα 5. Εκτέλεση αλγορίθμου εύρεσης μονοπατιού ανάμεσα στις κορυφές C, D του Σχήματος 1

```
length ← 0;
find_path(C, D, visited, path, 0, false)
  visited[C] ← true; length ← 1; path[1] ← C
  visited[B] = false;
  find_path(B, D, visited, path, 1, false)
    visited[2] ← true; comp[2] ← 1;
    visited[B] ← true; length ← 2; path[2] ← B;
    visited[A] = false;
    find_path(A, D, visited, path, 2, false)
      visited[A] ← true; length ← 3; path[3] ← A
      visited[B] = true;
      found = false;
      path[3] ← 0; length ← 2;
      visited[C] = true;
      found = false;
      path[2] ← 0; length ← 1;
      visited[E] = false;
      find_path(E, D, visited, path, 1, false)
        visited[E] ← true; length ← 2; path[2] ← E;
        visited[C] = true;
        visited[D] = false;
        find_path(D, D, visited, path, 2, false)
          visited[D] ← true; length ← 3; path[3] ← D
          found ← true;
```

4c. Γραφικές Συνιστώσες (ΔκΒ σε Μη-Κατευθυνόμενο)

Διαφάνειες

Slides_and_Exercises→04_graphtransexer.pdf σελ. 5-8

Εκφόνηση

Άσκηση 4 Να διατυπώσετε μία έκδοση της ΔκΒ που να υπολογίζει τις γραφικές συνιστώσες ενός μη-κατευθυνόμενου γραφήματος. Ποια είναι η πολυπλοκότητα του αλγόριθμου.

Λύση

Λύση Ο αλγόριθμος χρησιμοποιεί μία μεταβλητή *comp_num* η οποία, στο τέλος της εκτέλεσης, θα περιέχει τον αριθμό των γραφικών συνιστώσων και ένα μονοδιάστατο πίνακα *comp[]* η οποία για κάθε κορυφή *v* ∈ *V* θα δείχνει τον αριθμό που απαριθμεί τη γραφική συνιστώσα στην οποία ανήκει η κορυφή *v*. Η βασική διαδικασία απεικονίζεται στο αλγορίθμικό σχήμα 5. Το “κύριο πρόγραμμα” περιγράφεται από το σχήμα 4.

Η πολυπλοκότητα του αλγορίθμικου σχήματος είναι (και πάλι) $O(n + m)$.

Παράδειγμα 1 Για το γράφημα που απεικονίζεται στο Σχήμα 6, ο Αλγόριθμος 4 εκτελεί τα βήματα που απεικονίζονται στο Σχήμα 7. Για την εκτέλεση αυτή, θεωρούμε ότι οι μίστες γειτνίασης περιέχουν τις γειτονικές κορυφές σε αύξουσα σειρά επικέτας. Με τον ίδιο τρόπο (δηλαδή κατά αύξουσα σειρά επικέτας) γίνεται η απαρίθμηση των κορυφών στη γραμμή 7 του Αλγόριθμου 4.

Αλγόριθμος 4. Γραφικές Συνιστώσες

Algorithm 4 Γραφικές συνιστώσες

Require: $G(V, E)$

```
1: void graph_comp(int n, int comp[], int comp_num)
2: for v ∈ V do
3:   visited[v] ← false;
4:   comp_num ← 0;
5:   comp[v] ← 0;
6: end for
7: for v ∈ V do
8:   if not visited[v] then
9:     comp_num + +;
10:    explore1(v, visited, comp, comp_num);
11:   end if
12: end for
```

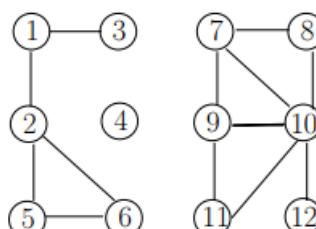
Αλγόριθμος 5. ΔκΒ – Γραφικές Συνιστώσες

Algorithm 5 ΔκΒ - Γραφικές συνιστώσες

Require: $G(V, E)$;

```
1: void explore1(int v, bool visited[], int comp[], int comp_num);
2: visited[v] ← true;
3: comp[v] ← comp_num;
4: for u ∈ N(v) do
5:   if not visited[u] then
6:     explore(u, visited, comp, comp_num);
7:   end if
8: end for
```

Σχήμα 6. Υπολογίζοντας γραφικές συνιστώσες



Σχήμα 7. Εκτέλεση αλγορίθμου εύρεσης γραφικών συνιστωσών

```
comp_num ← 1; vistied[1] = false;  
explore1(1, visited, comp, 1);  
    vistied[1] ← true; comp[1] ← 1;  
    vistied[2] = false;  
    explore1(2, visited, comp, 1);  
        vistied[2] ← true; comp[2] ← 1;  
        vistied[1] = true;  
        vistied[5] = false;  
        explore1(5, visited, comp, 1);  
            vistied[5] ← true; comp[5] ← 1;  
            vistied[2] = true;  
            vistied[6] = false;  
            explore1(6, visited, comp, 1);  
                vistied[6] ← true; comp[6] ← 1;  
                vistied[2] = true;  
                vistied[5] = true;  
                vistied[6] = true;  
                vistied[3] = false;  
                explore1(3, visited, comp, 1);  
                    vistied[3] ← true; comp[3] ← 1;  
                    vistied[1] = true;  
  
comp_num ← 2; vistied[4] = false;  
explore1(4, visited, comp, 2);  
    vistied[4] ← true; comp[4] ← 2;  
  
comp_num ← 3; vistied[7] = false;  
explore1(7, visited, comp, 3);  
    vistied[7] ← true; comp[7] ← 3;  
    vistied[8] = false;  
    explore1(8, visited, comp, 3);  
        vistied[8] ← true; comp[8] ← 3;  
        vistied[7] = true;  
        vistied[10] = false;  
        explore1(10, visited, comp, 3);  
            vistied[10] ← true; comp[10] ← 3;  
            vistied[7] = true;  
            vistied[8] = true;  
            vistied[9] = false;  
            explore1(9, visited, comp, 3);  
                vistied[9] ← true; comp[9] ← 3;  
                vistied[7] = true;  
                vistied[10] = true;  
                vistied[11] = false;  
                explore1(11, visited, comp, 3);  
                    vistied[11] ← true; comp[11] ← 3  
                    vistied[9] = true;  
                    vistied[10] = true;  
                    vistied[12] = false;  
                    explore1(12, visited, comp, 3);  
                        vistied[12] ← true; comp[12] ← 3;  
                        vistied[10] = true;  
                        vistied[9] = true;  
                        vistied[10] = true;
```

4d. ΔκΒ σε Κατευθυνόμενο Γράφημα

Διαφάνειες
exams→lyseis→Σεπτέμβριος 2021 σελ. 1-2

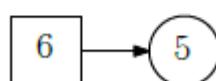
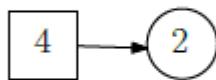
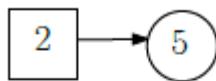
Εκφώνηση

Ερώτημα 1. (Βαθμοί 3)

Οι λίστες γειτνίασης του Σχήματος 1 αναπαριστούν ένα κατευθυνόμενο γράφημα. Να πραγματοποιήσετε διάσχιση κατά βάθος (ΔκΒ)

ξεκινώντας από την κορυφή με τον αριθμό 1 παραθέτοντας για κάθε κορυφή τον αριθμό προδιάταξης και μεταδιάταξης της. Στη συνέχεια να κατηγοριοποιήσετε τις ωρμές του γραφήματος ανάλογα με το αν είναι προς-τα-εμπρός ωρμές, οπισθοωρμές ή εγκάρσιες.

Σχήμα 1. Λίστες γειτνίασης κατευθυνόμενου γραφήματος

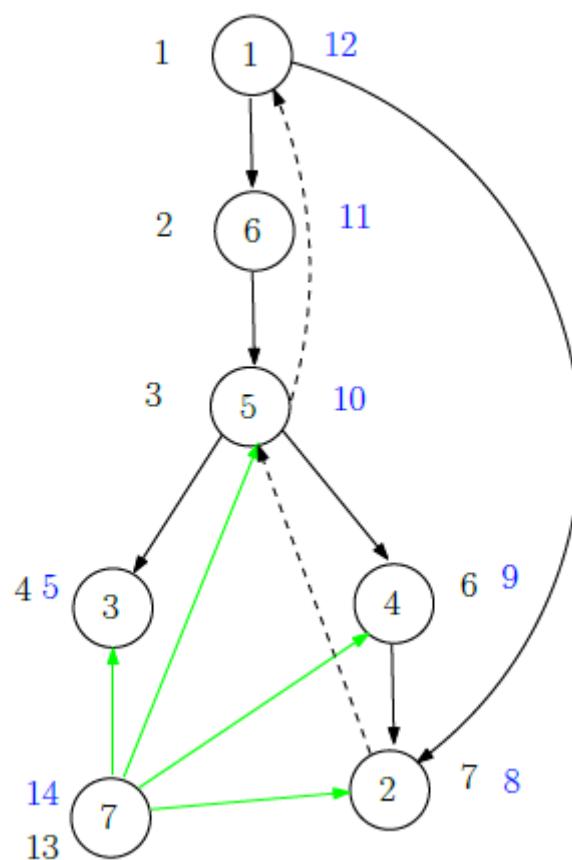


Λύση

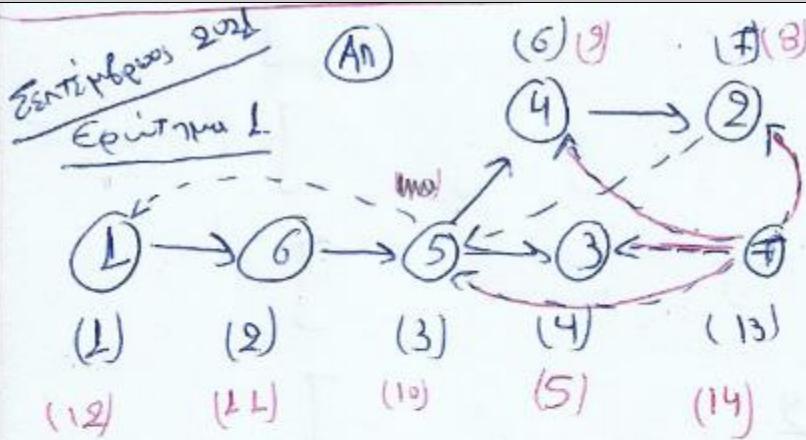
Δύση

Στο Σχήμα 2 απεικονίζεται το δένδρο της ΔκΒ μαζί με τους αριθμούς προδιάταξης και μεταδιάταξης. Οι πράσινες ακμές είναι εγκάρφοσεις ενώ οι διωχεικομμένες είναι οπισθοακμές. Οι υπόλοιπες είναι προς-τα-εμπρός ακμές.

Σχήμα 2. ΔκΒ με αριθμούς προδιάταξης και μεταδιάταξης κορυφών



Σημειώσεις



26/10/2023

Chapter 4 J

Μηλέ αριξ: Εμπροσθυακή

Διατεκτορικής: οπισθυακής

Αριξ (v, u)

Κόκκινη: σγκίρρεις

Εμπροσθυακή: $\text{pre}[v] < \text{pre}[u] < \text{post}[u] < \text{post}[v]$

Μηλέ αριξ: αριθμός προδιατάξης

οπισθυακή: $\text{pre}[u] < \text{pre}[v] < \text{post}[v] < \text{post}[u]$

Κόκκινης αριθμός: αριθμός μεταδιατάξης

Ερίαρχη: $[\text{pre}[v], \text{post}[v]] \cap [\text{pre}[u], \text{post}[u]] = \emptyset$

μεταδιατάξης

- 6 -

4e. Εύρεση συντομότερης διαδρομής με γραφήματα

Διαφάνειες

exams → lyseis → Φεβρουάριος 2021 σελ. 1, 3

Εκφόνηση

Ερώτημα 1.

Θέλουμε να πραγματοποιήσουμε ένα ταξίδι από μία πόλη (αφετηρία) σε μία άλλη (προορισμός) χρησιμοποιώντας ένα αυτοκίνητο με αυτονομία D χιλιομέτρων δηλαδή το αυτοκίνητο με γεμάτο ντεπόζιτο μπορεί να καλύψει μία απόσταση D χιλιομέτρων. Κατά τη διάρκεια του ταξιδιού περνάμε από διάφορες πόλεις στις οποίες μπορούμε να σταματήσουμε για ανεφοδιασμό καυσίμου. Θεωρούμε ότι οι πόλεις βρίσκονται σε μία ευθεία που ξεχίνα από την πόλη αφετηρία και καταλήγει στην πόλη προορισμού και δεν επιτρέπεται να παρακαμφθεί καμία πόλη. Θέλουμε να κάνουμε το ταξίδι ελαχιστοποιώντας τον αριθμό των στάσεων για ανεφοδιασμό.

Παράδειγμα: Το ταξίδι ξεχίναε από την ΑΘΗΝΑ και καταλήγει στα ΙΩΑΝΝΙΝΑ και το αυτοκίνητο μπορεί να διανύσει με γεμάτο ντεπόζιτο 177 χιλιόμετρα. Οι αποστάσεις των ενδιάμεσων πόλεων από την ΑΘΗΝΑ παρουσιάζονται στον Πίνακα 1. Στο

παράδειγμα είναι προφανές ότι δεν μπορούμε να πραγματοποιήσουμε το ταξίδι με έναν ανεφοδιασμό αφού η απόσταση ΑΘΗΝΑ-ΙΩΑΝΝΙΝΑ ξεπερνά τα 177 χιλιόμετρα. Επίσης είναι προφανές ότι αν κάνουμε, για παράδειγμα, επτά στάσεις για ανεφοδιασμό έχουμε σταματήσει παραπάνω φορές από το απαιτούμενο προκειμένου να πραγματοποιήσουμε το ταξίδι.

3. Να διατυπώσετε έτερο ωλγόριθμο της αρεσκείας σας ο οποίος να επιλύει το παραπάνω πρόβλημα και να τον εκτελέσετε στο παραπάνω παράδειγμα.

Τύποδειξη: Να περιγράψετε τον ωλγόριθμο σας αναλυτικά (και με λόγια).

(βαθμοί 2)

Πίνακας 1. Ενδεικτικές αποστάσεις από Αθήνα – πόλεις σε λεξικογραφική σειρά σε σχέση με το όνομα τους

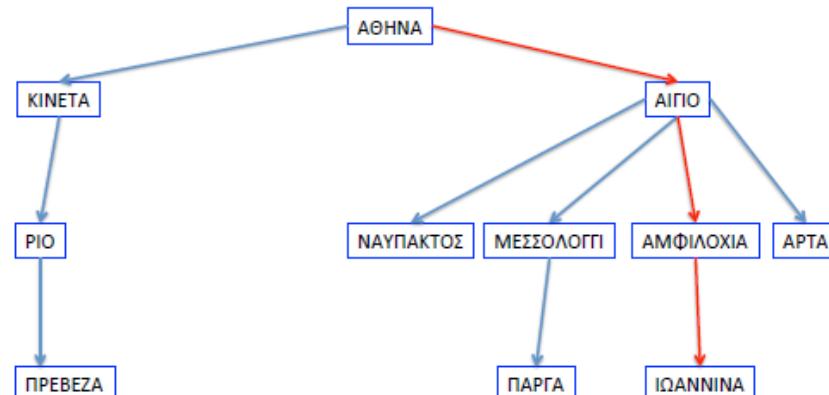
	ΠΟΛΗ	ΧΙΛΙΟΜΕΤΡΑ
1	ΑΘΗΝΑ	0
2	ΑΙΓΑΙΟ	174
3	ΑΜΦΙΛΟΧΙΑ	303
4	ΑΡΤΑ	349
5	ΙΩΑΝΝΙΝΑ	421
6	ΚΙΝΕΤΑ	53
7	ΜΕΣΟΛΟΓΓΙ	240
8	ΝΑΥΠΑΚΤΟΣ	218
9	ΠΑΡΓΑ	415
10	ΠΡΕΒΕΖΑ	357
11	ΡΙΟ (γέφυρα)	211

Λύση

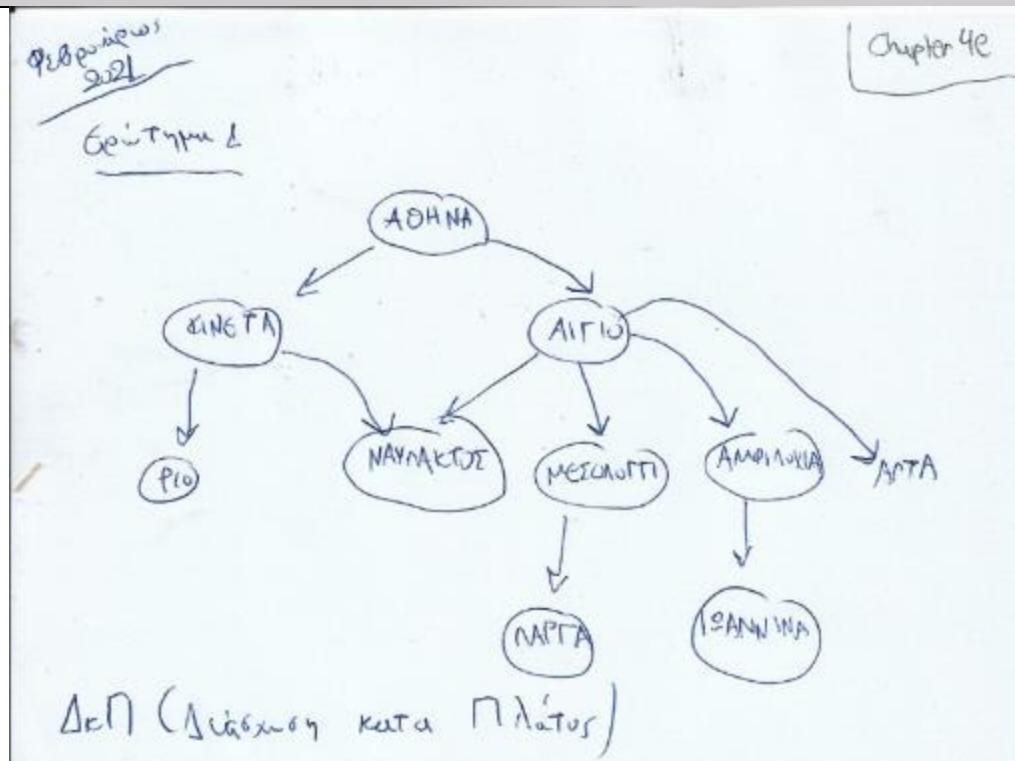
3. Μπορούμε να αναπαραστήσουμε το πρόβλημα σε ένα χατευθυνόμενο γράφημα $G(V, E)$ ως εξής. Οι κορυφές του γραφήματος αντιστοιχούν στις πόλεις. Δύο κορυφές v, u συνδέονται με αριθμή αν ανεφοδιάζονται το αυτοκίνητο στην πόλη v μπορούμε χωρίς άλλη στάση να φτάσουμε στην πόλη u .

Στο γράφημα αυτό θεωρούμε το συντομότερο μονοπάτι (μονοπάτι με το μικρότερο αριθμό ασφών) που συνδέει την κορυφή αφετηρία με την κορυφή-προορισμό. Το μονοπάτι αυτό περιέχει το μικρότερο αριθμό ασφών (και άρα κορυφών) προκειμένου να μεταβούμε από την αφετηρία στο προορισμό. Άρα ο αριθμός των πόλεων αυτών αποτελεί το μικρότερο αριθμό στάσεων. Ως εκ' τούτου αρχεί να εκτελέσουμε στο γράφημα αυτό τη ΔκΠ προκειμένου να εντοπίσουμε το μονοπάτι αυτό. Το δένδρο της διάσχισης κατά πλάτος παρουσιάζεται στο Σχήμα 1.

Σχήμα 1. Διάσχιση κατά Πλάτος



Σημειώσεις



5a. Ιδέα, Αλγόριθμος και Παράδειγμα Dijkstra

Διαφάνειες

[Slides_and_Exercises→05_shpaths.pdf](#) σελ. 1-36

Διαδρομή και Μονοπάτια

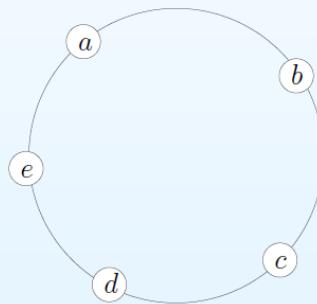
Έστω ένα μη-κατευθυνόμενο γράφημα $G(V, E)$.

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

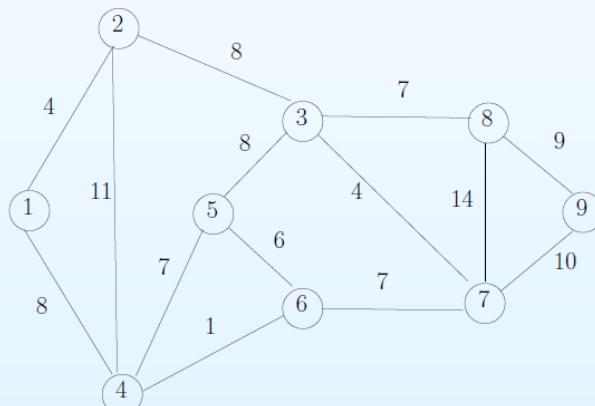
Ένα νοάφημα που δεν περιέχει κύκλο ονομάζεται άκυκλο

Εμβαρές Γράφημα

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

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

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



Σχήμα 2: Εμβαρές Γράφημα

- **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} .

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 ($\Delta\kappa\Pi$) 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.

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

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

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

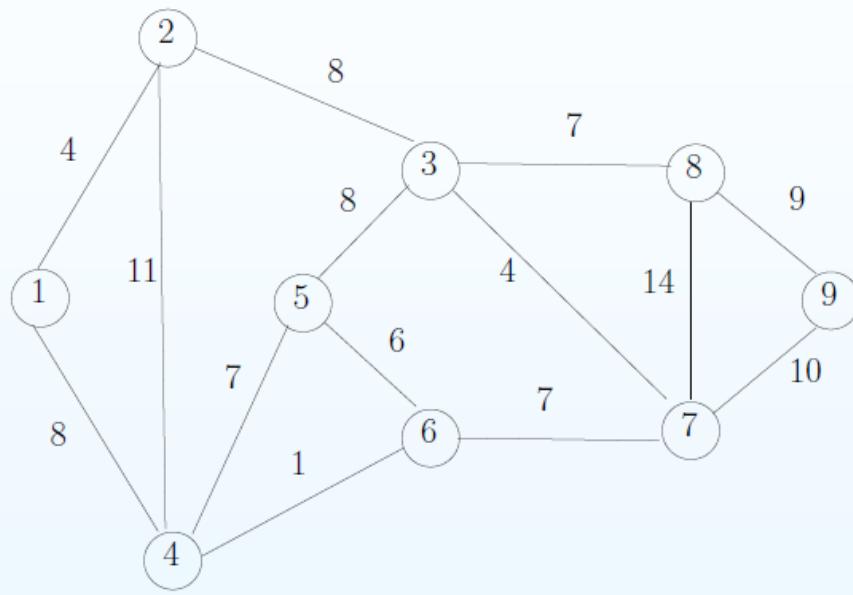
```

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.



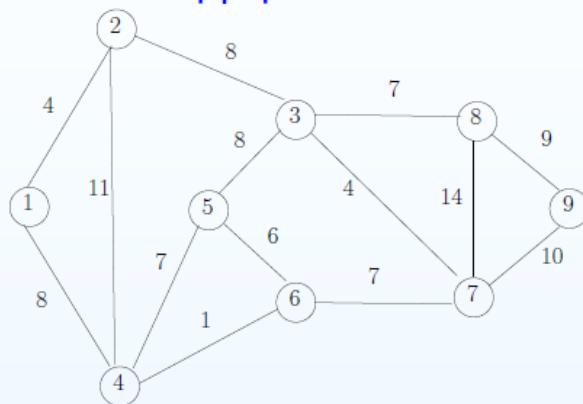
Αρχικοποίηση

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

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

$$S = \emptyset$$

Επανάληψη 1



ΦΑΣΗ I

$$v^* \leftarrow \operatorname{argmin}\{d_v : v \in \{1, \dots, 9\}\}$$

$$= \operatorname{argmin}\{0, \infty, \infty, \infty, \infty, \infty, \infty, \infty, \infty\} = 1,$$

$$S \leftarrow S \cup \{1\} = \emptyset \cup \{1\} = \{1\}$$

ΦΑΣΗ II

$$d_2 \leftarrow \min\{d_2, d_1 + w(1, 2)\} = \min\{\infty, 0 + 4\} = 4, \quad p_2 \leftarrow 1,$$

$$d_4 \leftarrow \min\{d_4, d_1 + w(1, 4)\} = \min\{\infty, 0 + 8\} = 8, \quad p_4 \leftarrow 1.$$

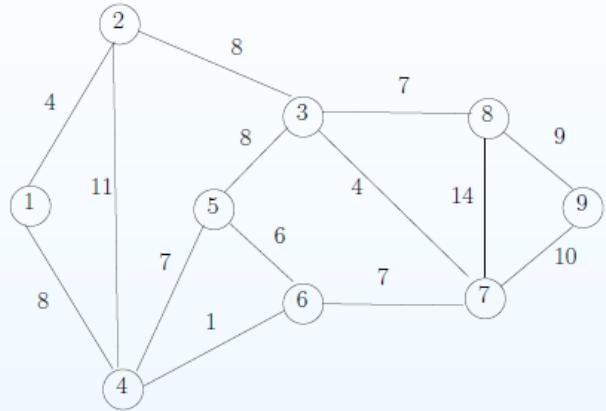
Έξοδος Επανάληψης

$$d = [0, 4, \infty, 8, \infty, \infty, \infty, \infty, \infty]$$

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

$$S = \{1\}$$

Επανάληψη 2



ΦΑΣΗ I

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

ΦΑΣΗ II

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

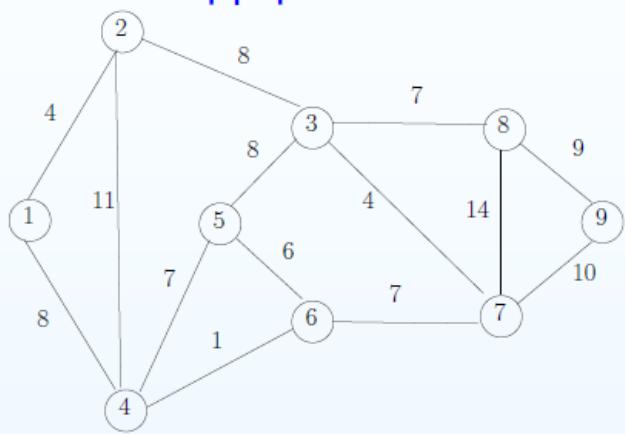
Έξοδος Επανάληψης

$$d = [0, 4, 12, 8, \infty, \infty, \infty, \infty, \infty]$$

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

$$S = \{1, 2\}$$

Επανάληψη 3



ΦΑΣΗ I

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

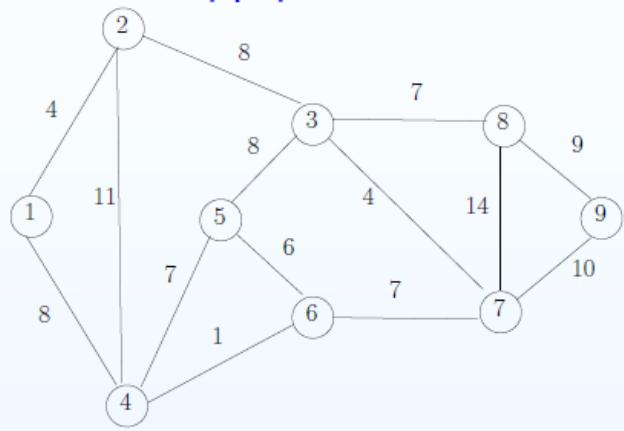
ΦΑΣΗ II

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

Έξοδος Επανάληψης

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

Επανάληψη 4



ΦΑΣΗ I

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

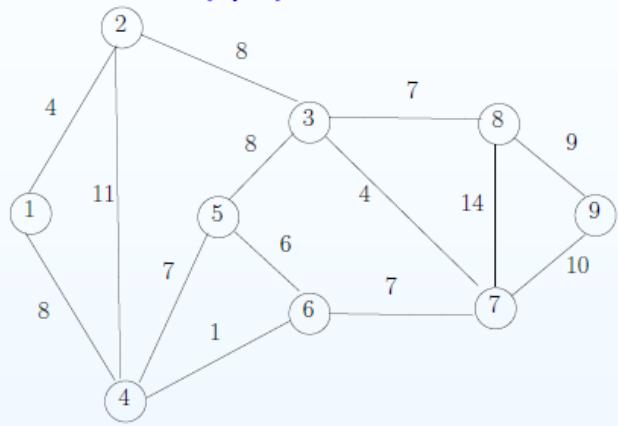
ΦΑΣΗ II

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

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

$$d_5 \leftarrow \min\{d_5, d_3 + w(3, 5)\} = \min\{15, 12 + 8\} = 15,$$

$$d_7 \leftarrow \min\{d_7, d_3 + w(7, 3)\} = \min\{16, 12 + 4\} = 16,$$

wa.gr $d_8 \leftarrow \min\{d_8, d_3 + w(3, 8)\} = \min\{\infty, 12 + 7\} = 19 \quad p_8 \leftarrow \frac{14}{3}.$

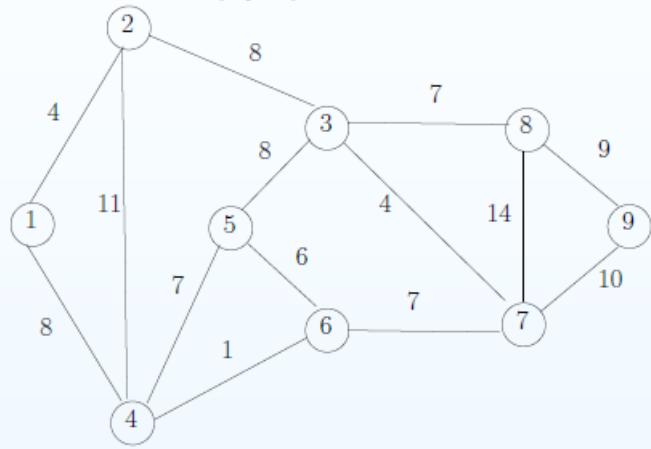
Έξοδος Επανάληψης

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

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

$$S = \{1, 2, 4, 6\}$$

Επανάληψη 6



ΦΑΣΗ I

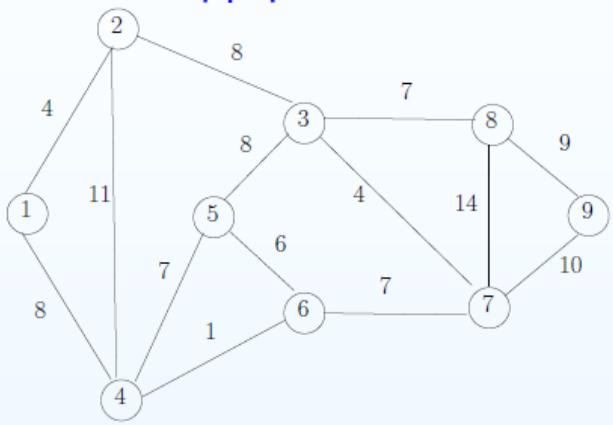
$$\begin{aligned}v^* &\leftarrow \operatorname{argmin}\{d_v : v \in \{5, 7, 8, 9\}\} \\&= \operatorname{argmin}\{15, 16, 19, \infty\} = 5, \\S &\leftarrow S \cup \{5\} = \{1, 2, 3, 4, 6\} \cup \{5\} = \{1, 2, 3, 4, 5, 6\}\end{aligned}$$

ΦΑΣΗ II

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 I

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

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

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

ΦΑΣΗ II

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

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

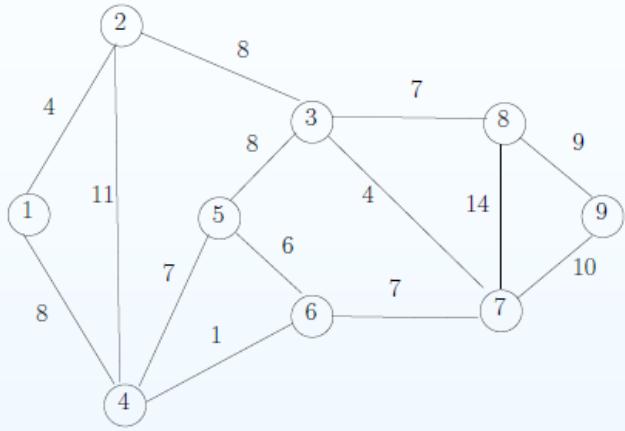
Έξοδος Επανάληψης

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

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

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

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

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

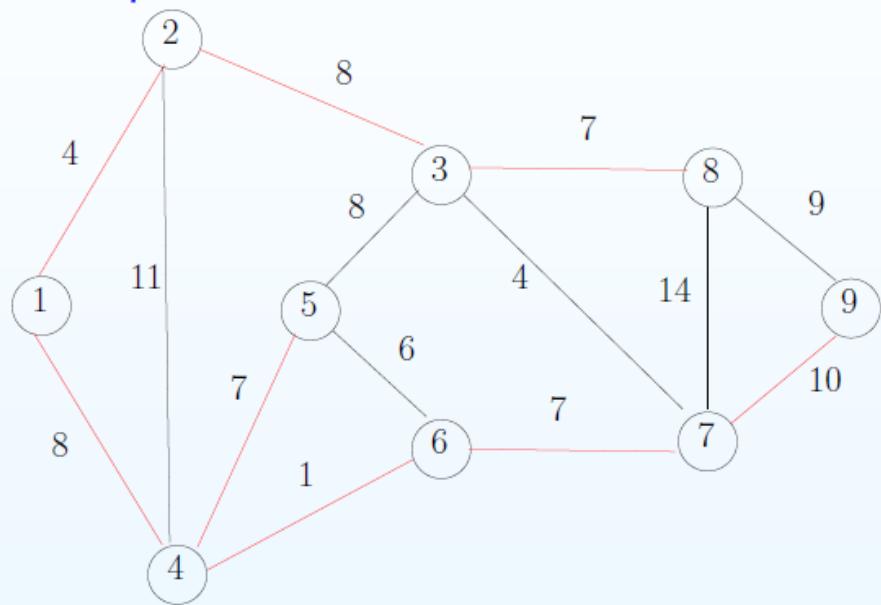
Έξοδος Επανάληψης

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

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

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

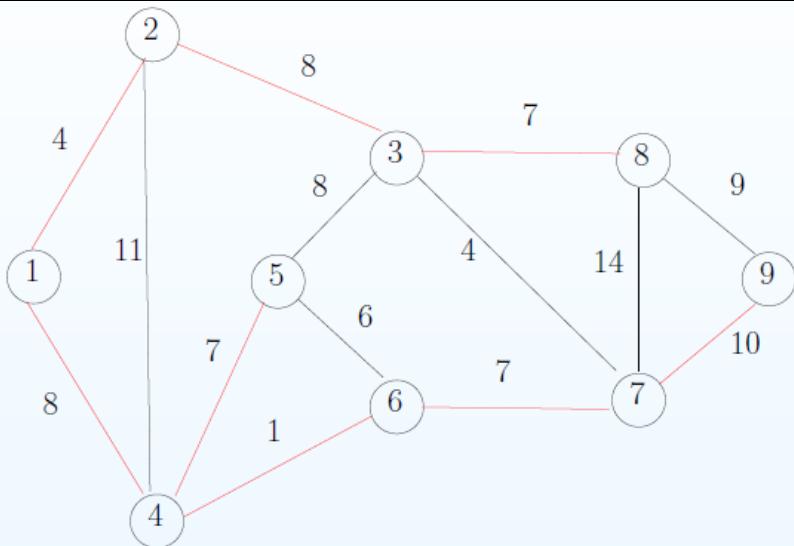
Λύση



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

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

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 (ΔSM) (λόγω της ιδιότητας της βέλτιστης υποδομής). Η πληροφορία αυτή βρίσκεται αποθηκευμένη στον πίνακα p . Στο συγκεκριμένο παράδειγμα έχουμε

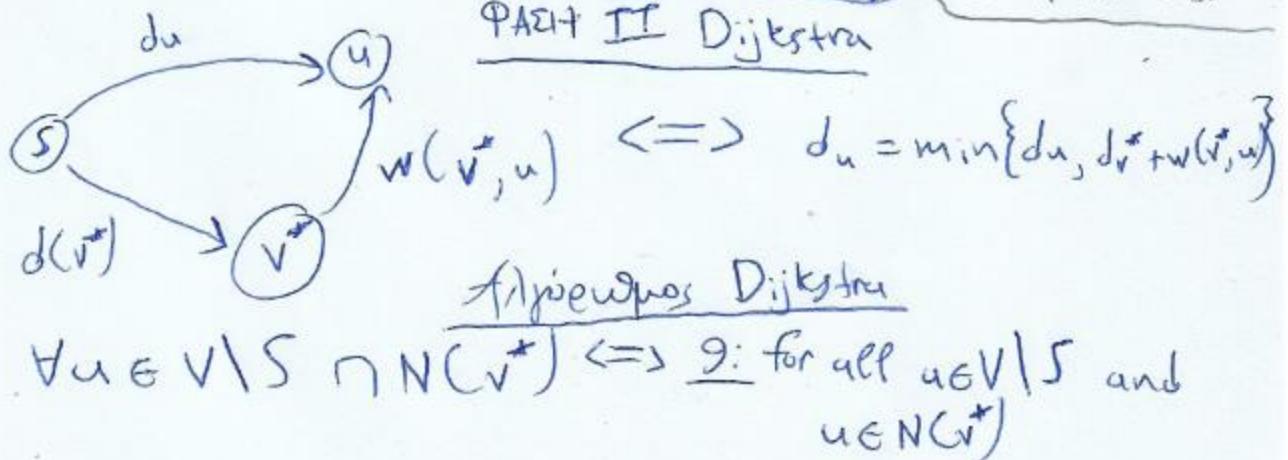
$$p = [\quad, 1, 2, 1, 4, 4, 6, 3, 7].$$

③

Ο4_ΣΥΝΤΟΜΟΤΕΡΑ ΜΟΝΟΠΑΤΙΑ

31/10/2023

Chapter 5a



- F -

5b. Ορθότητα και Πολυπλοκότητα Dijkstra

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 53-55

Λήμμα Ορθότητας

Λήμμα 28. Σε ένα βεβαρυμένο γράφημα $G(V, E, w)$, όπου $w : E(G) \rightarrow \mathbb{R}_{\geq 0}$, ο Αλγόριθμος 91 σωστά υπολογίζει το μήκος του συντομότερου μονοπατιού από μία διακεκριμένη κορυφή s προς κάθε άλλη κορυφή του γραφήματος

Απόδειξη Ορθότητας

Απόδειξη. Θα συμβολίσουμε, για κάθε $v \in V(G)$, ως

$d(s, v)$: το μήκος του (πραγματικού) συντομότερου μονοπατιού από την κορυφή s ως την κορυφή v ,

d_v : το μήκος του συντομότερου μονοπατιού από την κορυφή s ως την κορυφή v που παράγει ο αλγόριθμος (δηλαδή, $d_v \equiv dist[v]$).

Θα αποδείξουμε με επαγωγή ότι $d(s, v) = d_v$.

Βάση επαγωγής: Αρχικώς $S = \emptyset$, $d_s = 0$, $d_v = \infty$, για κάθε $v \in V(G) \setminus \{s\}$.

Στην έξοδο της πρώτης επανάληψης $S = \{s\}$ που σημαίνει ότι η τιμή d_s δεν θα αλλάξει μέχρι τον τερματισμό του αλγόριθμου και η τιμή αυτή είναι ίση με μηδέν η οποία είναι ακριβώς η τιμή $d(s, s)$.

Επαγωγική υπόθεση: Στην αρχή μίας τυχαίας επανάληψης θα υποθέσουμε ότι για κάθε κορυφή $v \in S$, $d_v = d(s, v)$.

Επαγωγικό βήμα: Θα δείξουμε ότι $d_{v^*} = d(s, v^*)$, όπου v^* η κορυφή που επιλέγει να προσθέσει στο σύνολο S ο αλγόριθμος κατά τη διάρκεια της τυχαίας επανάληψης. Από την (13.1) έχουμε

$$v^* = \operatorname{argmin}\{d_v : v \in V(G) \setminus S\}. \quad (13.2)$$

Ας υποθέσουμε ότι ισχύει

$$d(s, v^*) < d_{v^*}, \quad (13.3)$$

δηλαδή υπάρχει συντομότερο μονοπάτι από την κορυφή s στην κορυφή v^* από αυτό που έχει ανακαλύψει ο αλγόριθμος. Στο μονοπάτι αυτό, ονομαστικά P , έστω x η τελευταία κορυφή που ανήκει στο S και y η αμέσως επόμενη κορυφή. Παρατηρούμε ότι θα πρέπει $y \neq v^*$. διαφορετικά ($y = v^*$) το μονοπάτι P είναι ακριβώς αυτό που ανακαλύπτει ο αλγόριθμος - η επιλογή του στην παρούσα επανάληψη είναι η κορυφή v^* η οποία ταυτίζεται με την y . Επομένως υπάρχει τουλάχιστον μία ακόμη κορυφή στο P μετά την y . Από την επαγωγική υπόθεση έχουμε ότι $d_x = d(s, x)$ (αφού $x \in S$) και επειδή οι συντελεστές των ακμών είναι μη-αρνητικοί αριθμοί από την παραπάνω παρατήρηση θα πρέπει να ισχύει ότι

$$d_x + w(x, y) \leq d(s, v^*). \quad (13.4)$$

Όμως θα πρέπει

$$d_y \leq d_x + w(x, y) \quad (13.5)$$

αφού στην επανάληψη που η κορυφή x προστέθηκε στο σύνολο S το d_y υπολογίστηκε από τη σχέση

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

Από (13.5), (13.4), έχουμε ότι $d_y \leq d(s, v^*)$ το οποίο σε συνδυασμό με την (13.3) συνεπάγεται ότι $d_y < d_{v^*}$. Όμως αυτό είναι άτοπο αφού από την (13.2) έχουμε ότι $d_{v^*} \leq d_y$.

□

Παρατηρούμε ότι στην παραπάνω απόδειξη παίζει βασικό ρόλο η υπόθεση ότι τα βάρη στις ακμές είναι μη-αρνητικοί αριθμοί (αλλιώς δεν ισχύει η (13.4)). Επομένως ο αλγόριθμος δεν παράγει απαραίτητα το σωστό αποτέλεσμα αν δεν ισχύει ο περιορισμός αυτός.

Λήμμα Πολυπλοκότητας

Λήμμα 29. Ο Αλγόριθμος 91 εκτελεί $O(|V(G)|^2)$ ΣΥΒ.

Απόδειξη Πολυπλοκότητας

Απόδειξη. Θέτουμε $n = |V(G)|$. Για την αρχικοποίηση των δομών (πίνακες $dist$, $parent$) εκτελούνται $\Theta(n)$ ΣΥΒ. Κύριο μέρος του αλγόριθμου αποτελεί ο βρόχος της Γραμμής 6. Παρατηρούμε ότι στην τελευταία επανάληψη - το σύνολο S διαφέρει από το $V(G)$ κατά μία κορυφή - εκτελείται ένας σταθερός αριθμός ΣΥΒ: η κορυφή αυτή απλώς προστίθεται στο S . Άρα μπορούμε να θεωρήσουμε ότι ο αριθμός των επαναλήψεων του βρόχου της Γραμμής 6 είναι ίσος με $|V(G)| - 1 = n - 1$. Επομένως η Γραμμή 6 μπορεί να αντικατασταθεί από τη γραμμή

```
for  $i \leftarrow 1; i \leq n - 1; i++$  do
```

Θα μελετήσουμε τον αριθμό των ΣΥΒ σε μία τυχαία επανάληψη: έστω $i = k$.

Στη Φάση I της επανάληψης, έχουμε $|S| = k - 1$ και επομένως $|V(G) \setminus S| = n - k + 1$. Ο Αλγόριθμος αναζητάει ανάμεσα σε $n - k + 1$ στοιχεία να βρει το μικρότερο. Άρα εκτελεί $n - k$ συγκρίσεις και άρα ο αριθμός των ΣΥΒ φράζεται από τα επάνω από την ποσότητα $a \cdot (n - k)$ για κάποιο $a > 0$.

Στη Φάση II, εφόσον έχει προστεθεί η κορυφή v^* στο S έχουμε $|S| = k$ και $|V(G) \setminus S| = n - k$. Για κάθε κορυφή $u \in N(v^*) \cap (V(G) \setminus S)$ εκτελείται μία πρόσθεση, μία σύγκριση και πιθανόν δύο (τουλάχιστον) εκχωρήσεις (αναθεώρηση των τιμών $dist[u]$, $parent[u]$). Επειδή το σύνολο $N(v^*) \cap (V(G) \setminus S)$ μπορεί να περιέχει όλες τις $n - k$ κορυφές, ο αριθμός των ΣΥΒ φράζεται από τη $b \cdot (n - k)$, όπου $b \geq 4$. Επειδή $k \leq n - 1$, για το συνολικό αριθμό $T(n)$ των ΣΥΒ που εκτελεί ο αλγόριθμος ισχύει ότι

$$T(n) \leq \Theta(n) + \sum_{k=1}^{n-1} a \cdot (n - k) + \sum_{k=1}^{n-1} b \cdot (n - k) \Rightarrow T(n) \in O(n^2).$$

□

Μία παρατήρηση στην απόδειξη του Λήμματος 29 είναι ότι το σύνολο $N(v^*) \cap (V(G) \setminus S)$ θεωρήθηκε διαθέσιμο σε κάθε επανάληψη χωρίς να χρειαστεί να εκτελεστούν κάποια ΣΥΒ για τον υπολογισμό του. Αυτό δεν επηρεάζει το τελικό αποτέλεσμα αφού ο υπολογισμός του συνόλου αυτού μπορεί να γίνει σε $O(n)$ ΣΥΒ (π.χ. χρησιμοποιώντας μία απεικόνιση των συνόλων με πίνακες). Γενικότερα, η πολυπλοκότητα του αλγόριθμου του Dijkstra μπορεί να μειωθεί περαιτέρω χρησιμοποιώντας πιο εξειδικευμένες δομές δεδομένων, π.χ. ουρά προτεραιότητας ελαχίστου με χρήση δυαδικού σωρού, κλπ [7].

Chapter 6

6a. Ιδέα, Αλγόριθμος και Παράδειγμα Bellman-Ford

Διαφάνειες

Slides_and_Exercises → 05_shpaths.pdf σελ. 42-45, 50-57

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"

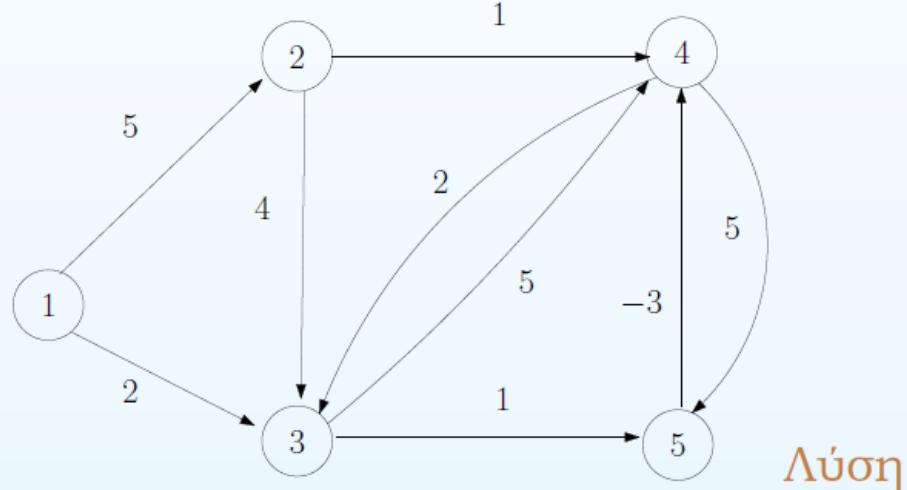
end if

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.

Παράδειγμα

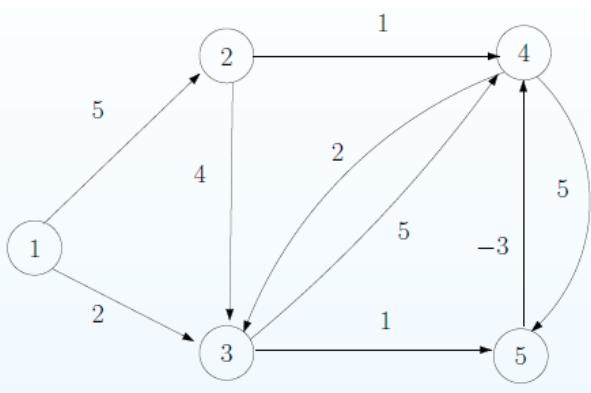
Με τη χρήση του αλγόριθμου των Bellman-Ford να βρεθούν τα συνομότερα μονοπάτια από την κορυφή 1 προς τις υπόλοιπες κορυφές στο γράφημα



Αρχικοποίηση

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

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



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

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

Επανάληψη $k = 1$

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

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

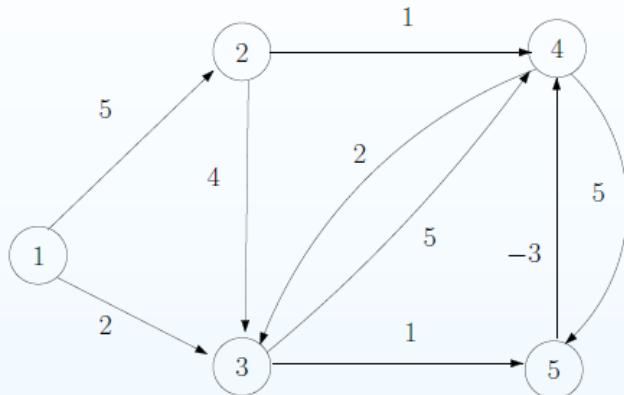
$$\begin{aligned} d_3^1 &= \min\{d_3^0, d_1^0 + w(1, 3), d_2^0 + w(2, 3), d_4^0 + w(4, 3)\} \\ &= \min\{\infty, 0 + 2, \infty + 4, \infty + 2\} = 2, \quad p_3^1 = 1, \end{aligned}$$

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

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

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

29 /



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

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

Επανάληψη $k = 2$

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

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

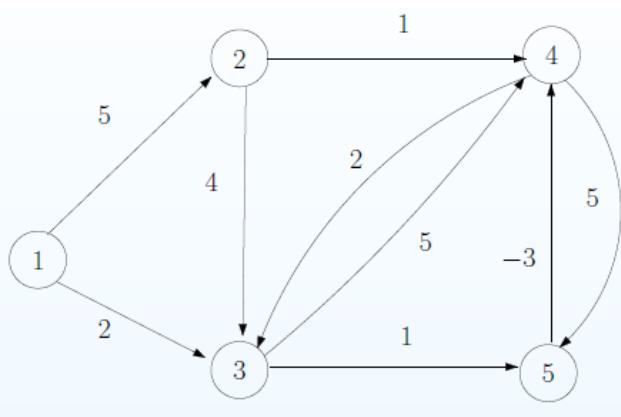
$$\begin{aligned} d_3^2 &= \min\{d_3^1, d_1^1 + w(1, 3), d_2^1 + w(2, 3), d_4^1 + w(4, 3)\} \\ &= \min\{2, 0 + 2, 5 + 4, \infty + 2\} = 2, \end{aligned}$$

$$\begin{aligned} d_4^2 &= \min\{d_4^1, d_2^1 + w(2, 4), d_3^1 + w(3, 4), d_5^1 + w(5, 4)\} \\ &= \min\{\infty, 5 + 1, 2 + 5, \infty - 3\} = 6, \quad p_4^2 = 2, \end{aligned}$$

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

$$= \min\{\infty, 2 + 1, \infty + 5, \} = 3 \quad p_5^2 = 3$$

29



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

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

Επανάληψη $k = 3$

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

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

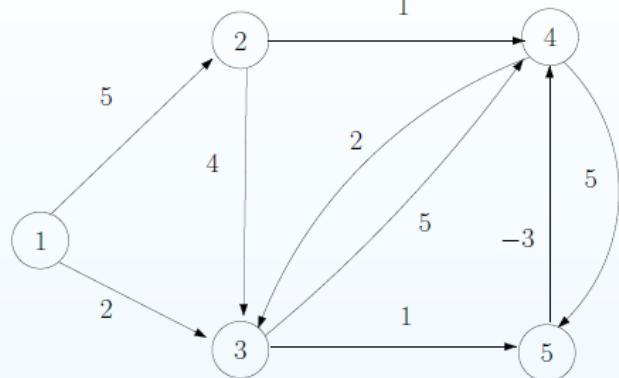
$$\begin{aligned} d_3^3 &= \min\{d_3^2, d_1^2 + w(1, 3), d_2^2 + w(2, 3), d_4^2 + w(4, 3)\} \\ &= \min\{2, 0 + 2, 5 + 4, 6 + 2\} = 2, \end{aligned}$$

$$\begin{aligned} d_4^3 &= \min\{d_4^2, d_2^2 + w(2, 4), d_3^2 + w(3, 4), d_5^2 + w(5, 4)\} \\ &= \min\{6, 5 + 1, 2 + 5, 3 - 3\} = 0, \quad p_4^3 = 5, \end{aligned}$$

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

a.gr $= \min\{3, 2 + 1, 6 + 5, \} = 3$

29



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

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

Επανάληψη $k = 4$

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

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

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

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

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

a.gr $= \min\{3, 2 + 1, 0 + 5, \} = 3$

29

Output

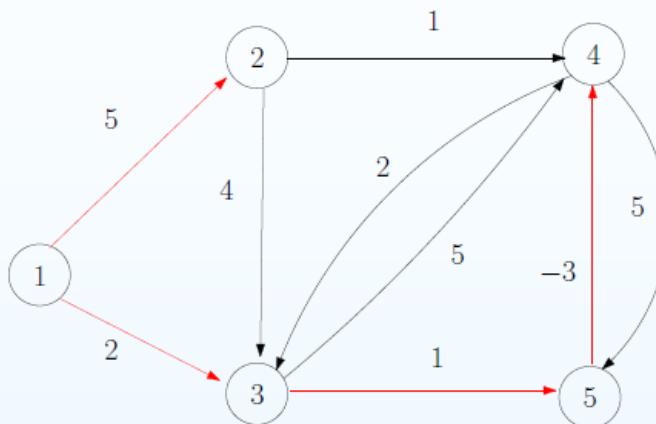
$$d = [0, 5, 2, 0, 3]$$

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

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'.

Σημειώσεις

Θ

| 7/11/2023
| Chapter 6a

Bellman-Ford

Έλεγχος υποψης αρνητικών κύκλων

$d_v^{N+1} > d_u^{N+1} + w(u, v) + (u, v)$

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

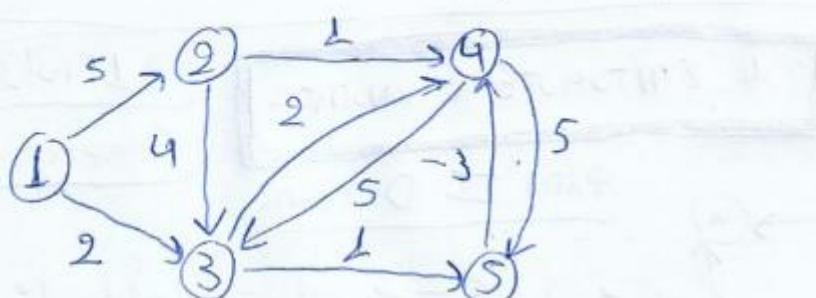
$u \quad v$

$(5, 4)$

$d_4 > d_5 + w(5, 4)$

$0 > 3 - 3 \Rightarrow 0 > 0$ Άρα δεν υπάρχει αρνητικός κύκλος

Ο έλεγχος γίνεται με κάθε ακμή



Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 62-64

Αήματα Ορθότητας

Αήμα 30. Αν το κατευθυνόμενο γράφημα $G(V, E)$ δεν περιέχει αρνητικό κύκλο ο αλγόριθμος των Bellman-Ford υπολογίζει το συντομότερο μονοπάτι από την αφετηρία προς κάθε άλλη κορυφή του γραφήματος.

Απόδειξη Ορθότητας

Απόδειξη. Για $v \in V(G)$, έστω d_v^k το μήκος του μονοπατιού από την κορυφή s στη v που υπολογίζει ο αλγόριθμος των Bellman-Ford και αποτελείται από το πολύ k ακμές. Αρκεί να δείξουμε ότι το d_v^k είναι ίσο με το μήκος του (πραγματικού) συντομότερου μονοπατιού από την κορυφή s στην κορυφή v με αριθμό ακμών μικρότερο-ίσο του k . Θα χρησιμοποιήσουμε επαγωγή στην τιμή του k .

Βάση επαγωγής: Για $k = 0$, ο αλγόριθμος επιστρέφει το σωστό αποτέλεσμα αφού $d_s^0 = 0 = d(s, s)$, $d_v^0 = \infty$, για κάθε $v \in V(G) \setminus \{s\}$.

Επαγωγική υπόθεση: Για κάθε κορυφή v , d_v^{k-1} είναι το μήκος του (πραγματικού) συντομότερου μονοπατιού από την s στη v , χρησιμοποιώντας το πολύ $k - 1$ ακμές.

Επαγωγικό βήμα: Έστω P το (πραγματικό) συντομότερο μονοπάτι από την s στην v , με το-πολύ k ακμές. Συμβολίζουμε με $w(P)$ το μήκος του.

Αν το P αποτελείται από το-πολύ $k - 1$ ακμές, τότε από την επαγωγική υπόθεση, $w(P) = d_v^{k-1}$ και επειδή $d_v^k \leq d_v^{k-1}$ (λόγω της (13.8)), έχουμε

$$d_v^k \leq w(P). \quad (13.9)$$

Στην περίπτωση που το P αποτελείται από k ακμές, 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 το-πολύ k κορυφές και άρα το-πολύ $k - 1$ ακμές. Έστω $w(Q)$ το μήκος του μονοπατιού αυτού. Από την επαγωγική υπόθεση

$$w(Q) = d_u^{k-1}. \quad (13.10)$$

Επίσης από κατασκευής ισχύει ότι

$$w(P) = w(Q) + w(u, v). \quad (13.11)$$

Από την (13.8) έχουμε ότι $d_v^k \leq d_u^{k-1} + w(u, v)$ η οποία σε συνδυασμό με τις (13.10) συνεπάγεται

$$d_v^k \leq w(Q) + w(u, v).$$

Από την (13.11), το δεξί μέλος της παραπάνω ανισότητας είναι ίσο με $w(P)$. Άρα και στην περίπτωση αυτή ισχύει η (13.9).

Όμως επειδή το $w(P)$ είναι το μήκος του (πραγματικού) συντομότερου μονοπατιού από την s στη v με το-πολύ k ακμές, η (13.9) ισχύει σαν ισότητα, δηλαδή $d_v^k = w(P)$.



Λήμμα Αρνητικού Κύκλου

Λήμμα 31. Αν υπάρχει ένας αρνητικός κύκλος σε ένα κατευθυνόμενο βεβαρυμένο γράφημα, ο Αλγόριθμος 92 τον εντοπίζει.

Απόδειξη Αρνητικού Κύκλου

Απόδειξη. Έστω s η αφετηριακή κορυφή στον αλγόριθμο Bellman-Ford και έστω v_0, v_1, \dots, v_k ένας κύκλος αρνητικού μήκους με $v_0 = v_k$ ο οποίος μπορεί να προσπελασθεί από ένα μονοπάτι που εκκινεί από το s . Το μήκος του κύκλου είναι

$$\sum_{i=1}^k w(v_{i-1}, v_i) < 0. \quad (13.12)$$

Ο αλγόριθμος βρίσκει ότι υπάρχει αρνητικός κύκλος αν υπάρχει ακμή (v, u) για την οποία ισχύει ότι²

$$d_v^{|V(G)|-1} > d_u^{|V(G)|-1} + w(u, v).$$

Έστω λοιπόν ότι δεν ισχύει αυτό για καμία ακμή του αρνητικού κύκλου. Έχουμε δηλαδή ότι

$$d_{v_i}^{|V(G)|-1} \leq d_{v_{i-1}}^{|V(G)|-1} + w(v_{i-1}, v_i), \forall i = 1, \dots, k.$$

Αθροίζοντας για κάθε $i = 1, \dots, k$ έχουμε

$$\sum_{i=1}^k d_{v_i}^{|V(G)|-1} \leq \sum_{i=1}^k d_{v_{i-1}}^{|V(G)|-1} + \sum_{i=1}^k w(v_{i-1}, v_i). \quad (13.13)$$

Παρατηρούμε ότι

$$\begin{aligned} \sum_{i=1}^k d_{v_i}^{|V(G)|-1} &= \sum_{i=1}^{k-1} d_{v_i}^{|V(G)|-1} + d_{v_k}^{|V(G)|-1}, \\ \sum_{i=1}^k d_{v_{i-1}}^{|V(G)|-1} &= \sum_{i=1}^{k-1} d_{v_i}^{|V(G)|-1} + d_{v_0}^{|V(G)|-1} \end{aligned}$$

και επομένως η (13.13) γίνεται

$$d_{v_k}^{|V(G)|-1} \leq d_{v_0}^{|V(G)|-1} + \sum_{i=1}^k w(v_{i-1}, v_i). \quad (13.14)$$

Όμως επειδή $v_0 = v_k (\Rightarrow d_{v_k}^{|V(G)|-1} = d_{v_0}^{|V(G)|-1})$ η παραπάνω σχέση δίνει

$$\sum_{i=1}^k w(v_{i-1}, v_i) \geq 0 \quad (13.15)$$

το οποίο έρχεται σε αντίφαση με την (13.12). □

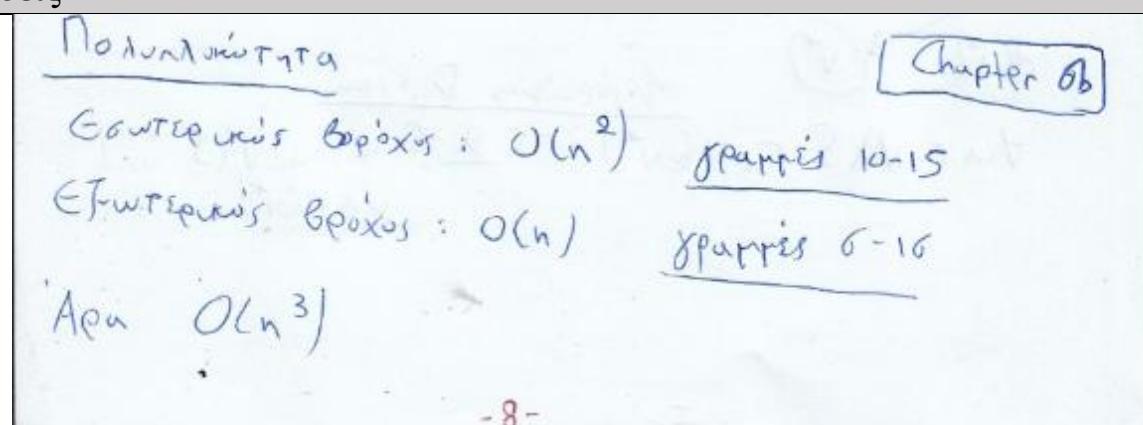
Λήμμα Πολυπλοκότητας

Λήμμα 32. Ο Αλγόριθμος 92 εκτελεί $O(|V(G)|^3)$ ΣΥΒ.

Απόδειξη Πολυπλοκότητας

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 σε πολυωνυμικό αριθμό ΣΥΒ μέσω της εύρεσης ενός τέλειου ταιριάσματος.

Σημειώσεις



6c. Το πρόβλημα εύρεσης μακρύτερου μονοπατιού

Διαφάνειες

Slides_and_Exercises→05_shpaths.pdf σελ. 62-64

Μακρύτερα Μονοπάτια

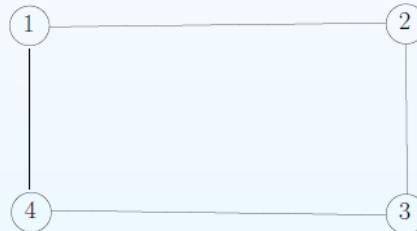
Το συμπληρωματικό του προβλήματος εύρεσης του συντομότερου μονοπατιού είναι το πρόβλημα εύρεσης του μακρύτερου (απλού) μονοπατιού:

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)

7a. Θεωρία πάνω στο Δένδρο Συντομότερων Μονοπατιών

Διαφάνειες

Slides_and_Exercises→07_spstexer.pdf σελ. 1-2

Εκφώνηση

Άσκηση 1 Να μελετήσετε την επίπτωση που έχει στο δένδρο των συντομότερων μονοπατιών ενός μη-κατευθυνόμενου βεβαρημένου γραφήματος οι παρακάτω τροποποιήσεις στους συντελεστές των ακμών.

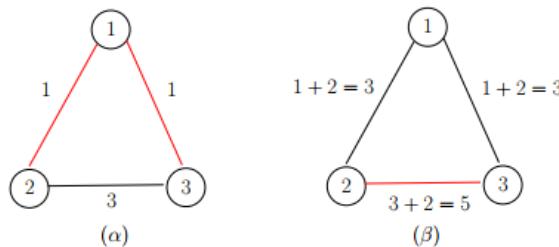
1. Πολλαπλασιασμός των συντελεστών των ακμών με αριθμό $b > 0$.
2. Αύξηση του συντελεστή των ακμών κατά $b > 0$.
3. Αντιστροφή των συντελεστών των ακμών.

1. Πολλαπλασιασμός των συντελεστών των ακμών με αριθμό $b > 0$

1. Το δένδρο των συντομότερων μονοπατιών παραμένει το ίδιο αφού οι συντελεστές των ακμών μεταβάλλονται αναλογικά.

2. Αύξηση του συντελεστή των ακμών κατά $b > 0$

2. Το δένδρο των ελάχιστων μονοπατιών μπορεί να αλλάξει. Για παράδειγμα, στο Σχήμα 1α το συντομότερο μονοπάτι από την κορυφή 2 έως την κορυφή 3 είναι διαμέσου της κορυφής 1. Αν αυξήσουμε κατά ένα τα βάρη των ακμών τότε το συντομότερο μονοπάτι περιλαμβάνει μόνο την ακμή που συνδέει απευθείας την κορυφή 2 με την 3. Αυτό συμβαίνει επειδή ο αριθμός των ακμών που περιέχει το συντομότερο μονοπάτι (ανάμεσα σε δύο κορυφές) μπορεί να είναι μεγάλος οπότε η αύξηση του βάρους κάθε ακμής οδηγεί σε μεγαλύτερη επιβάρυνση και άρα μονοπάτια με μικρότερο αριθμό ακμών γίνονται συντομότερα.

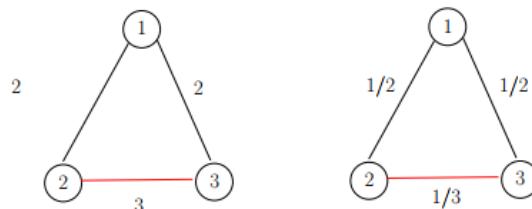


Σχήμα 1: Συντομότερα μονοπάτια - αύξηση των συντελεστών των ακμών κατά μία σταθερά

Γενικότερα, για να μην μεταβάλλεται το συντομότερο μονοπάτι από μία κορυφή s σε μία κορυφή t με την προσθήκη μίας σταθεράς είναι (α) όλα τα μονοπάτια από το s στο t να έχουν τον ίδιο αριθμό ακμών και (β) να μην δημιουργούνται (με την προσθήκη της σταθεράς) αρνητικοί κύκλοι.

3. Αντιστροφή των συντελεστών των ακμών

3. Η αντιστροφή των συντελεστών των ακμών φαίνεται να οδηγεί σε αλλαγή των συντομότερων μονοπατιών. Για παράδειγμα στο γράφημα του Σχήματος 1α θεωρώντας τα αντίστροφα βάρη αλλάζει πρακτικά μόνο το βάρος της ακμής (2, 3) (γίνεται $1/3$). Επομένως το συντομότερο μονοπάτι από την κορυφή 2 στην 3 περιέχει, μετά την αντιστροφή, μόνο την ακμή (2, 3). Όμως αυτό δεν συμβαίνει πάντα (Σχήμα 2)



Σχήμα 2: Συντομότερα μονοπάτια - αντιστροφή των συντελεστών των ακμών

7b. Εφαρμογή Dijkstra σε γράφημα

Διαφάνειες

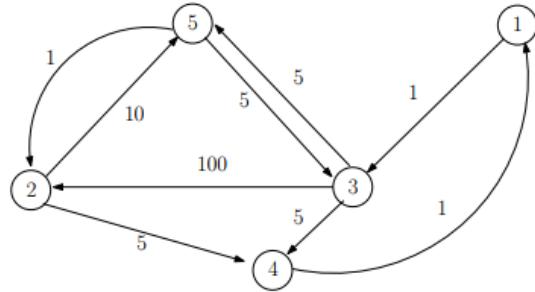
Slides_and_Exercises→07_spstexer.pdf σελ. 2-4

Εκφόνηση

Άσκηση 2 Δίνεται ένα κατευθυνόμενο βεβαρημένο γράφημα $G(V, E, w, t)$, όπου $w : E \rightarrow \mathbb{R}_+$, και $t \in V$. Να εκπονήσετε αλγόριθμο ο οποίος υπολογίζει τα συντομότερα μονοπάτια από κάθε κορυφή $v \in V$ προς την κορυφή t προσορισμού t .

Με την βοήθεια του αλγόριθμου που εκπονήσατε παραπάνω να υπολογίσετε τα συντομότερο μονοπάτι στο γράφημα του Σχήματος 3 από κάθε κορυφή προς την κορυφή 5.

Σχήμα 3. Συντομότερα μονοπάτια από κάθε κορυφή προς την κορυφή 5

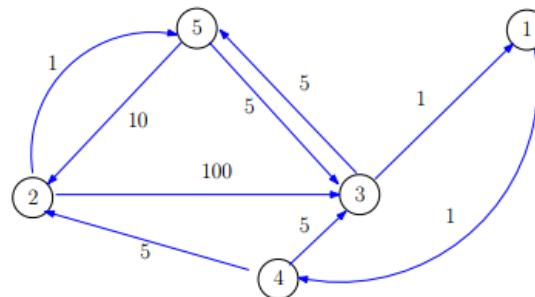


Λύση

Δεδομένου του κατευθυνόμενου γραφήματος $G(V, E, w)$, δημιουργούμε το γράφημα \bar{G} ως εξής. $V(G) = V(\bar{G})$ και για κάθε κατευθυνόμενη ακμή $(v, u) \in E(G)$ εισάγουμε την ακμή $(u, v) \in E(\bar{G})$. Εφαρμόζοντας τον αλγόριθμο του Dijkstra στο γράφημα \bar{G} με αφετηριακή κορυφή t έχουμε τη λύση στο αρχικό πρόβλημα.

Στο γράφημα του Σχήματος 4 έχουν αντιστραφεί οι ακμές. Θα λύσουμε το πρόβλημα εύρεσης των συντομότερων διαδρομών από την κορυφή 5 προς τις υπόλοιπες κορυφές στο γράφημα αυτό.

Σχήμα 4. Γράφημα $\bar{G}(V, E)$ προερχόμενο από το γράφημα του Σχήματος 3 με αντιστροφή στην φορά των ακμών



Αρχικοποίηση

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

Επανάληψη 1

Φάση I

$$u^* = \operatorname{argmin}\{d_1, d_2, d_3, d_4, d_5\} = \operatorname{argmin}\{\infty, \infty, \infty, \infty, 0\} = 5$$
$$S = \{5\}$$

Φάση II

$$d_2 = \min\{d_2, d_5 + w(5, 2)\} = \min\{\infty, 0 + 10\} = 10, \quad p_2 = 5,$$
$$d_3 = \min\{d_3, d_5 + w(5, 3)\} = \min\{\infty, 0 + 5\} = 5, \quad p_3 = 5$$

Τέλος Επανάληψης:

$$d = [\infty, 10, 5, \infty, 0], \quad p = [, 5, 5, ,]$$

Επανάληψη 2

Φάση I

$$u^* = \operatorname{argmin}\{d_1, d_2, d_3, d_4\} = \operatorname{argmin}\{\infty, 10, 5, \infty\} = 3$$

$$S = \{5, 3\}$$

Φάση II

$$d_1 = \min\{d_1, d_3 + w(3, 1)\} = \min\{\infty, 5 + 1\} = 6, \quad p_1 = 3$$

Τέλος Επανάληψης:

$$d = [6, 10, 5, \infty, 0], \quad p = [3, 5, 5, \dots]$$

Επανάληψη 3

Φάση I

$$u^* = \operatorname{argmin}\{d_1, d_2, d_4\} = \operatorname{argmin}\{6, 10, \infty\} = 1$$

$$S = \{5, 3, 1\}$$

Φάση II

$$d_4 = \min\{d_4, d_1 + w(1, 4)\} = \min\{\infty, 6 + 1\} = 7, \quad p_4 = 1$$

Τέλος Επανάληψης:

$$d = [6, 10, 5, 7, 0], \quad p = [3, 5, 5, 1, \dots]$$

Επανάληψη 4

Φάση I

$$u^* = \operatorname{argmin}\{d_2, d_4\} = \operatorname{argmin}\{10, 7\} = 4$$

$$S = \{5, 3, 1, 4\}$$

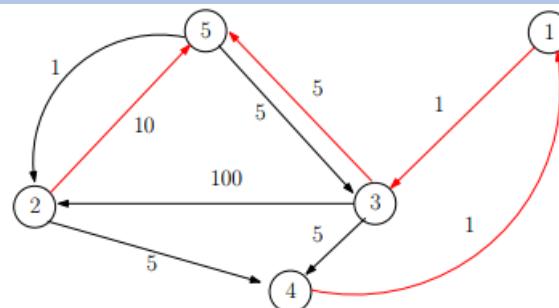
Φάση II

$$d_2 = \min\{d_2, d_4 + w(4, 2)\} = \min\{10, 7 + 5\} = 10,$$

Τέλος Επανάληψης:

$$d = [6, 10, 5, 7, 0], \quad p = [3, 5, 5, 1, \dots]$$

Σχήμα 5. Το δένδρο των συντομότερων διαδρομών προς την κορυφή 5 – κόκκινες ακμές



7c. Μοντελοποίηση ενός προβλήματος με εφαρμογή του Dijkstra

Διαφάνειες

Slides_and_Exercises→07_spstexer.pdf σελ. 4

Εκφώνηση

Άσκηση 3 Σε μία πόλη υπάρχουν κάποιες δομές που παρέχουν υπηρεσίες κοινής αφελείας (π.χ. πυροσβεστικοί σταθμοί). Επίσης υπάρχουν σημεία ενδιαφέροντος (π.χ. σχολεία) τα οποία θα μπορούσαν να είναι χρήστες των υπηρεσιών των παραπάνω δομών. Δεδομένου ότι όλες οι αποστάσεις είναι γνωστές, να εκπονήσετε έναν αλγόριθμο ο οποίος για κάθε σημείο ενδιαφέροντος να υπολογίζει την κοντινότερη προς αυτό δομή κοινής αφέλειας.

Υποδέστε ότι έχουμε στη διάθεσή μας (ως υπορουτίνα) μια υλοποίηση Dijkstra(V, E, w, s) του αλγορίθμου του Dijkstra που δέχεται ως είσοδο ένα κατευθυνόμενο γράφημα $G = (V, E)$ με μη αρνητικά βάρη στις ακμές του ($w(e) \geq 0$ για κάθε $e \in E$) και μία κορυφή $s \in V$. Ως έξοδο δίνει ένα διάνυσμα $d = [d(v), v \in V]$ με τα μήκη των συντομότερων διαδρομών από την s προς κάθε $v \in V$ και ένα διάνυσμα $p = [p(v), v \in V]$ με τη χρήση του οποίου μπορούμε να απεικονίσουμε τις συντομότερες διαδρομές

Σημείωση: Στον αλγόριθμό σας επιτρέπεται να καλέσετε μόνο μία φορά την υπορουτίνα Dijkstra(V, E, w, s)

Λύση

Λύση Μπορούμε να μοντελοποιήσουμε το πρόβλημα με τη χρήση ενός κατευθυνομένου συνεκτικού γράφηματος $G(V, E, w)$ όπου $w : E \rightarrow \mathbb{R}^+$. Το σύνολο των κορυφών του γραφήματος V περιέχει τόσο τα σημεία ενδιαφέροντος όσο και τις δομές παροχής υπηρεσιών οι οποίες αποτέλουν ένα σύνολο $S \subset V$. Οι δρόμοι ανάμεσα στις κορυφές του V αποτελούν τις κατευθυνόμενες ακμές του γραφήματος κάθε μία από τις οποίες φέρει βάρος ίσο με την απόσταση που χωρίζει τις δύο κορυφές που ενώνει η ακμή. Προφάνως, θέλουμε για κάθε κορυφή $u \in V \setminus S$ να βρούμε την κοντινότερη κορυφή του συνόλου S .

Αν μπορούσαμε να καλέσουμε πολλές φορές την υπορουτίνα Dijkstra(V, E, w, s) θα το κάναμε ξεχωριστά για κάθε $s \in S$ και στη συγκρίναμε για κάθε κορυφή $u \in V \setminus S$ συντομότερες απόστασεις και θα επιλέγαμε το s που αντιστοιχεί στη μικρότερη. Όμως κάτι τέτοιο δεν επιτρέπεται (από την εκφώνηση).

Για να λύσουμε το πρόβλημα εισάγουμε στο γράφημα μίσ νέα κορυφή, έστω v_s , την οποία τη συνδέουμε με ακμές προς κάθε κορυφή του συνόλου S . Κάθε τέτοια ακμή φέρει μηδενικό βάρος. Στη συγκέντρωση καλούμε τη ρουτίνα Dijkstra(V, E, w, v_s). Προφανώς, για κάθε $u \in V \setminus S$ το στοιχείο $d(u)$ δίνει το μήκος της διαδρομής από το κόντινότερη στη u κορυφή του S . Η κορυφή αυτή μπορεί να ιχνηλατηθεί μέσω του διανύσματος p .

7d. Το πρόβλημα του ταξιδιώτη (Dijkstra)

Διαφάνειες

Slides_and_Exercises→07_spstexer.pdf σελ. 7-8

Εκφώνηση

Άσκηση 5 Δίνεται ένα απλό συνεκτικό γράφημα $G(V, E)$ με μη-αρνητικά βάρη στις **κορυφές** του και τρεις διακεκριμένες κορυφές του $s, t, u \in V$. Συμβολίζουμε με $c(v)$ το βάρος της κορυφής $v \in V$. Υποδέστε ότι οι κορυφές $v \in V \setminus \{s, t\}$ αντιστοιχούν σε διαφορετικές πόλεις και το κόστος παραμονής σε ξενοδοχείο της κάθε πόλης δίνεται από τον συντελεστή $c(v)$. Ένας ταξιδιώτης θέλει να ταξιδέψει με το φεινότερο τρόπο από την πόλη s στην πόλη t αλλά θα ήθελε να μείνει και στο ξενοδοχείο της πόλης u (κάνοντας μία παράκαμψη) αν αυτό δεν αύξανε κατά πολύ το κόστος του ταξιδιού του. Μπορούμε να θεωρήσουμε ότι μία αύξηση της τάξης μέχρι 10% στο κόστος του ταξιδίου είναι μία αποδεκτή αύξηση προκειμένου να ικανοποιηθεί η επιδυμία του ταξιδιώτη.

Υποδέστε ότι έχετε στη διάθεση σας μία υλοποίηση Dijkstra(V, E, w, s, t) του αλγορίθμου του Dijkstra που δέχεται ως είσοδο ένα κατευθυνόμενο γράφημα $G = (V, E)$ με μη-αρνητικά βάρη στις **ακμές** του ($w(e) \geq 0$ για κάθε $e \in E$) και κορυφές $s, t \in V$ και παράγει σαν έξοδο το μήκος της συντομότερη διαδρομής από την κορυφή s στην κορυφή t . Να εκπονήσετε αλγόριθμο ο οποίος να απαντά αν ο ταξιδιώτης μπορεί να ικανοποιήσει την επιδυμία του.

Λύση

Άσωη Για να μπορέσουμε να χρησιμοποιήσουμε την ρουτίνα του αλγόριθμου του Dijkstra θα πρέπει να αντιστοιχήσουμε βάρη στις ακμές του γραφήματος. Σε κάθε ακμή (v, u) αντιστοιχούμε το βάρος $w(v, u) = c(u)$ - αφού ο ταξιδιώτης καταλήξει στην πόλη u θα πρέπει να πληρώσει και το κόστος παραμονής στο ξενοδοχείο της πόλης. Στη συνέχεια υπολογίζουμε το κόστος της συντομότερης διαδρομής από την πόλη s στην πόλη t και το συγκρίνουμε με το άθροισμα του κόστους της ελάχιστης διαδρομής από την πόλη s στην πόλη u και από την πόλη u στην πόλη t . Δηλαδή ο ζητούμενος αλγόριθμος αποτελείται από τις παρακάτω γράμμες κώδικα

if $1 \cdot \text{Dijkstra}(V, E, w, s, t) \geq \text{Dijkstra}(V, E, w, s, u) + \text{Dijkstra}(V, E, w, u, t)$ **then**

Ο ταξιδιώτης μπορεί να μείνει στην πόλη u

else

Ο ταξιδιώτης δεν μπορεί να μείνει στην πόλη u

end if

7e. Το πρόβλημα του arbitrage (Bellman-Ford)

Διαφάνειες

Slides_and_Exercises → 07_spstexer.pdf σελ. 8-13

Εκφώνηση

Άσκηση 7 Στην αγορά συναλλάγματος νομίσματα ανταλλάσσονται με συγκεκριμένες ισοτιμίες. Για παράδειγμα ο Πίνακας 2 περιέχει τις ισοτιμίες ανάμεσα σε ευρώ, δολλάριο, λίρα (αγγλίας) και ελβετικό φράνκο. Το πρόβλημα του arbitrage συνιστάται στον εντοπισμό μίας σειράς ανταλλαγών που ξεκινώνται από κάποιο νόμισμα καταλήγει στο ίδιο νόμισμα με μεγαλύτερο ποσό από το αρχικό. Να εντοπίσετε αν υπάρχει μία τέτοια σειρά με βάση τις παραπάνω ισοτιμίες έχοντας σαν αρχικό ποσό ένα εκατομύριο ευρώ.

Πίνακας 2. Ισοτιμίες

	EUR	USD	GBP	CHF
EUR	1	1,122000	0,899300	1,113300
USD	0,891266	1	0,801500	0,992200
GBP	1,111976	1,247661	1	1,238000
CHF	0,89823	1,007861	0,807754	1

Πίνακας 3. Λογάριθμοι ισοτιμιών πολλαπλασιασμένοι με -1

	EUR	USD	GBP	CHF
EUR	0	-0,049993	0,046095	-0,046612
USD	0,049993	0	0,096096	0,003401
GBP	-0,046095	-0,096096	0	-0,092721
CHF	0,046612	-0,003401	0,092721	0

Λύση

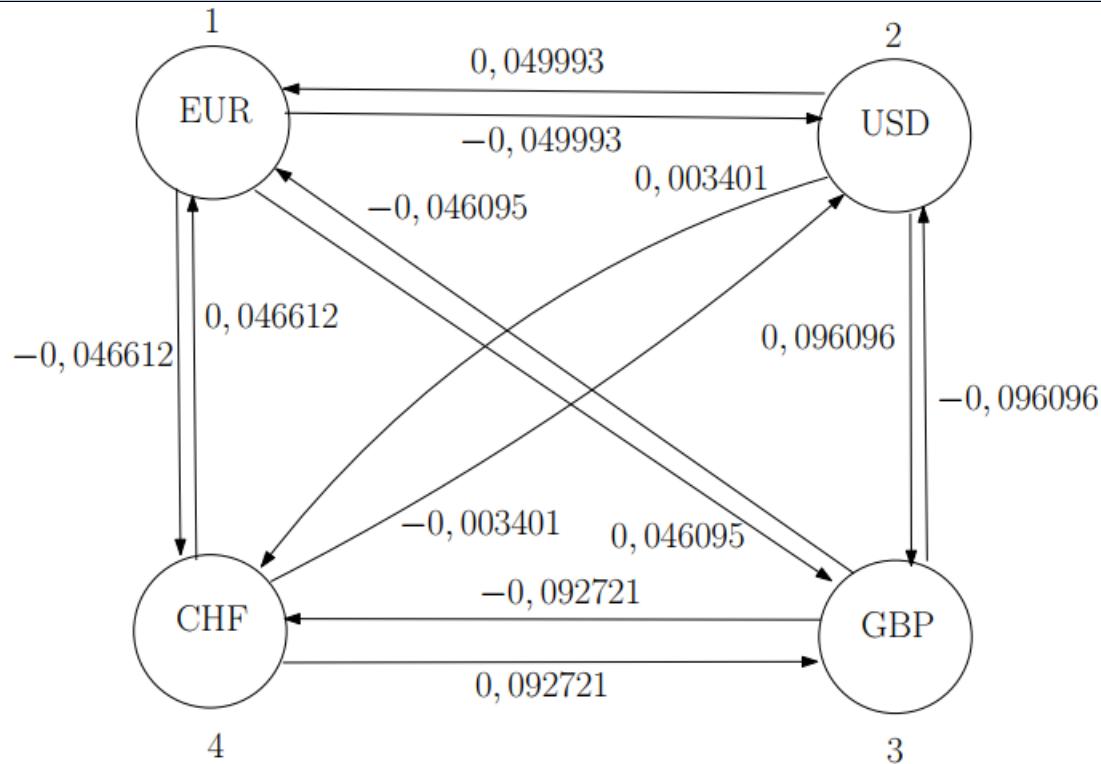
Λύση Για να αντιληφθούμε καλύτερα τους αριθμούς (τις ισοτιμίες) του Πίνακα 2 αν έχουμε 500 ευρώ μπορούμε να πάρουμε $500 * 1,122 = 561$ δολλάρια. Εναλλακτικά (μία περίπτωση), θα μπορούσαμε να κάνουμε τα ευρώ λίρες και τις λίρες δολλάρια, Στην περίπτωση αυτή θα είχαμε $500 * 0,8993 * 1,247661 = 561,01$ δολλάρια. Δηλαδή στην εναλλακτική περίπτωση θα είχαμε κέρδος 1 cent παραπάνω.

Το παράδειγμα που μόλις είδαμε οδηγεί στη μοντελοποίηση του προβλήματος σε όρους γραφημάτων. Θεωρούμε ένα βεβαρημένο γράφημα με κορυφές τα νομίσματα. Έχουμε ακμή από μία κορυφή σε μία άλλη με βάρος την ισοτιμία που μετατρέπει μία μονάδα του ενός νομίσματος στο άλλο. Έτσι στο γράφημα αυτό μας ενδιαφέρει να βρούμε τον κύκλο (αν υπάρχει) που ξεκινάει από συγκεκριμένη κορυφή αφετηρία (την κορυφή του ευρώ στην περίπτωση μας) και επιστρέφει σε αυτή μεγιστοποιώντας το γινόμενο των συντελεστών των ακμών (ισοτιμιών) που συμμετέχουν. Αν το γινόμενο αυτό είναι μεγαλύτερο της μονάδας τότε έχουμε κίνητρο να ακολουθήσουμε τις υποδεικνυόμενες συναλλαγές αφού θα καταλήξουμε με αυξημένη θέση από αυτή στην οποία ξεκινήσαμε.

Για να αποφύγουμε την μεγιστοποίηση του γινομένου μπορούμε, θεωρώντας τους λογάριθμους των ισοτιμιών, να πάρουμε το ισοδύναμο πρόβλημα με μεγιστοποίηση άθροισματος των συντελεστών των ακμών. Περαιτέρω πολλαπλασιάζοντας κάθε λογάριθμο με -1 μπορούμε να ζητήσουμε την εύρεση των συντομότερων διαδρομών από την κορυφή που αντιστοιχεί στο νόμισμα με το οποίο ξεκινάμε. Αν το γράφημα περιέχει αρνητικό κύκλο (που ξεκινάει και τελειώνει στο νόμισμα με το οποίο ξεκινάμε) έχουμε την απάντηση στο αρχικό μας πρόβλημα. Προφανώς για να επιλύσουμε το τελευταίο πρόβλημα λόγω των αρνητικών συντελεστών σε κάποιες από τις ακμές θα χρησιμοποιήσουμε τον αλγόριθμο των Bellman-Ford..

Οι λογάριθμοι των ισοτιμιών του Πίνακα 2 πολλαπλασιάσμενοι με -1 απεικονίζονται στον Πίνακα 3. Στη συνέχεια κατασκευάζουμε το γράφημα του Σχήματος 8. Στη συνέχεια εκτελούμε τον αλγόριθμο των Bellman-Ford.

Σχήμα 8. Γράφημα συντελεστών Πίνακα 3



Αρχικοποίηση

$$d_1^0 = 0, \quad d_2^0 = \infty, \quad d_3^0 = \infty, \quad d_4^0 = \infty$$

Επανάληψη 1

$$\begin{aligned}d_1^1 &= \min\{d_1^0, d_2^0 + w(2, 1), d_3^0 + w(3, 1), d_4^0 + w(4, 1)\}, \\&= \min\{0, \infty + (0, 049993), \infty + (-0, 046095), \infty + (-0, 046612)\} = 0 \\d_2^1 &= \min\{d_2^0, d_1^0 + w(1, 2), d_3^0 + w(3, 2), d_4^0 + w(4, 2)\}, \\&= \min\{\infty, 0 + (-0, 049993), \infty + (-0, 096096), \infty + (-0, 003401)\} = -0, 049993, \quad p_2^1 = 1 \\d_3^1 &= \min\{d_3^0, d_1^0 + w(1, 3), d_2^0 + w(2, 3), d_4^0 + w(4, 3)\}, \\&= \min\{\infty, 0 + (0, 046095), \infty + (-0, 96096), \infty + (0, 092721)\} = 0, 046095, \quad p_3^1 = 1 \\d_4^1 &= \min\{d_4^0, d_1^0 + w(1, 4), d_2^0 + w(2, 4), d_3^0 + w(3, 4)\}, \\&= \min\{\infty, 0 + (-0, 046612), \infty + (0, 003401), \infty + (-0, 092721)\} = -0, 046612, \quad p_4^1 = 1\end{aligned}$$

Συνολικά, υπολογίσαμε,

$$\begin{aligned}d^1 &= [0, -0, 049993, 0, 046095, -0, 046612] \\p^1 &= [, 1, 1, 1]\end{aligned}$$

Επανάληψη 2

$$\begin{aligned}d_1^2 &= \min\{d_1^1, d_2^1 + w(2, 1), d_3^1 + w(3, 1), d_4^1 + w(4, 1)\}, \\&= \min\{0, -0, 049993 + (0, 049993), 0, 046095 + (-0, 046095), 0, 046612 + (-0, 046612)\} \\&= 0 \\d_2^2 &= \min\{d_2^1, d_1^1 + w(1, 2), d_3^1 + w(3, 2), d_4^1 + w(4, 2)\}, \\&= \min\{-0, 049993, 0 + (-0, 049993), 0, 046095 + (-0, 096096), -0, 046612 + (-0, 003401)\} \\&= -0, 050013, \quad p_2^2 = 4 \\d_3^2 &= \min\{d_3^1, d_1^1 + w(1, 3), d_2^1 + w(2, 3), d_4^1 + w(4, 3)\}, \\&= \min\{0, 046095, 0 + (0, 046095), -0, 049993 + (0, 96096), -0, 046612 + (0, 092721)\} \\&= 0, 046095, \quad p_3^2 = 1 \\d_4^1 &= \min\{d_4^0, d_1^1 + w(1, 4), d_2^1 + w(2, 4), d_3^1 + w(3, 4)\}, \\&= \min\{-0, 046612, 0 + (-0, 046612), -0, 049993 + (0, 003401), 0, 046095 + (-0, 092721)\} \\&= -0, 046626, \quad p_4^2 = 3\end{aligned}$$

Συνολικά, υπολογίσαμε,

$$\begin{aligned}d^2 &= [0, -0, 050013, 0, 046095, -0, 046626] \\p^2 &= [, 4, 1, 3]\end{aligned}$$

Επανάληψη 3

$$\begin{aligned}d_1^3 &= \min\{d_1^2, d_2^2 + w(2, 1), d_3^2 + w(3, 1), d_4^2 + w(4, 1)\}, \\&= \min\{0, -0, 050013 + (0, 049993), 0, 046095 + (-0, 046095), -0, 046626 + (-0, 046612)\} \\&= -0.00002, \quad p_1^3 = 2 \\d_2^3 &= \min\{d_2^2, d_1^2 + w(1, 2), d_3^2 + w(3, 2), d_4^2 + w(4, 2)\}, \\&= \min\{-0, 050013, 0 + (-0, 049993), 0, 046095 + (-0, 096096), -0, 046626 + (-0, 003401)\} \\&= -0, 050027, \quad p_2^3 = 4 \\d_3^3 &= \min\{d_3^2, d_1^2 + w(1, 3), d_2^2 + w(2, 3), d_4^2 + w(4, 3)\}, \\&= \min\{0, 046095, 0 + (0, 046095), -0, 050013 + (0, 96096), -0, 046626 + (0, 092721)\} \\&= 0, 046083, \quad p_3^3 = 2 \\d_4^3 &= \min\{d_4^2, d_1^2 + w(1, 4), d_2^2 + w(2, 4), d_3^2 + w(3, 4)\}, \\&= \min\{-0, 046626, 0 + (-0, 046612), -0, 050013 + (0, 003401), 0, 046095 + (-0, 092721)\} \\&= -0, 046626, \quad p_4^3 = 3\end{aligned}$$

Συνολικά, υπολογίσαμε,

$$\begin{aligned}d^3 &= [-0.0002, -0, 050027, 0, 046083, -0, 046626] \\p^3 &= [2, 4, 2, 3]\end{aligned}$$

Έξοδος Αλγορίθμου

Το output του αλγόριθμου για τον υπολογισμό των συντομότερων διαδρομών είναι

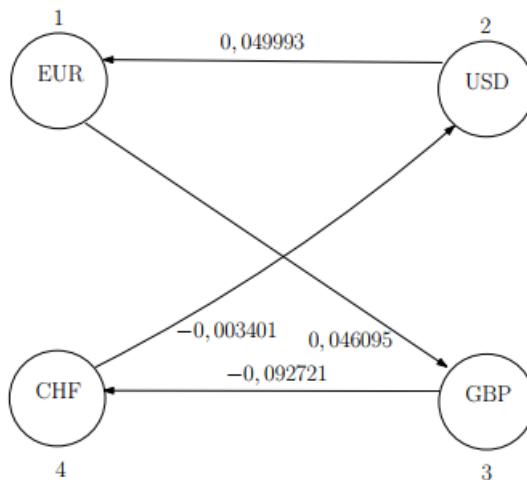
$$d^3 = [-0.00002, -0.050027, 0.046083, -0.046626] \quad (5)$$

$$p = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 1 & 3 \\ 2 & 4 & 2 & 3 \end{bmatrix} \quad (6)$$

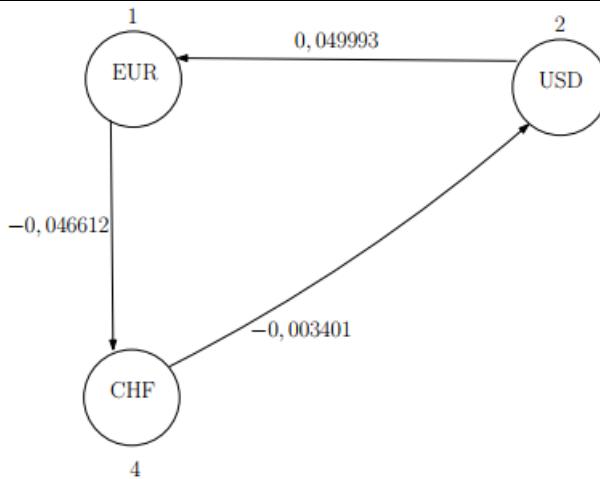
Πίνακας 4. Αρνητικοί κύκλοι στους οποίους έχει πρόσβαση η κορυφή 1

#	(u, v)	d_v^3	$d_u^3 + w(u, v)$	Κύκλος	Μήκος Κύκλου
1	(2, 1)	$d_1^3 = -0.0002$	$d_2^3 + w(2, 1)$	$1 \leftarrow 2 \leftarrow 4 \leftarrow 3 \leftarrow 1$	$-0.050027 + 0.049993 = -0.000034$
2	(1, 3)	$d_3^3 = 0.046083$	$d_1^3 + w(1, 3)$	$3 \leftarrow 1 \leftarrow 2 \leftarrow 4 \leftarrow 1$	$-0.0002 + (0.046095) = 0.046075$
3	(2, 3)	$d_3^3 = 0.046083$	$d_2^3 + w(2, 3)$	$3 \leftarrow 2 \leftarrow 4 \leftarrow 3 \leftarrow 1$	$-0.050027 + 0.096096 = 0.046069$
4	(1, 4)	$d_4^3 = -0.046626$	$d_1^3 + w(1, 4)$	$4 \leftarrow 1 \leftarrow 2 \leftarrow 4 \leftarrow 1$	$-0.00002 + (-0.046612) = -0.046632$
5	(3, 4)	$d_4^3 = -0.046626$	$d_3^3 + w(3, 4)$	$4 \leftarrow 3 \leftarrow 2 \leftarrow 4 \leftarrow 1$	$0.046083 + (-0.092721) = -0.046638$

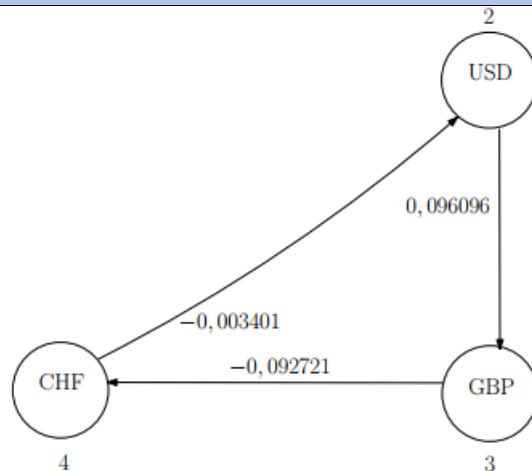
Σχήμα 9. Κύκλος που αντιστοιχεί στην πρώτη γραμμή του Πίνακα 4. Αθροισμα συντελεστών ακμών -0.000034



Σχήμα 10. Κύκλος που αντιστοιχεί στην δεύτερη και τετάρτη γραμμή του Πίνακα 4. Αθροισμα συντελεστών ακμών -0.00002



**Σχήμα 11. Κύκλος που αντιστοιχεί στην τρίτη και πέμπτη γραμμή του Πίνακα 4.
Άθροισμα συντελεστών ακμών -0.000015**

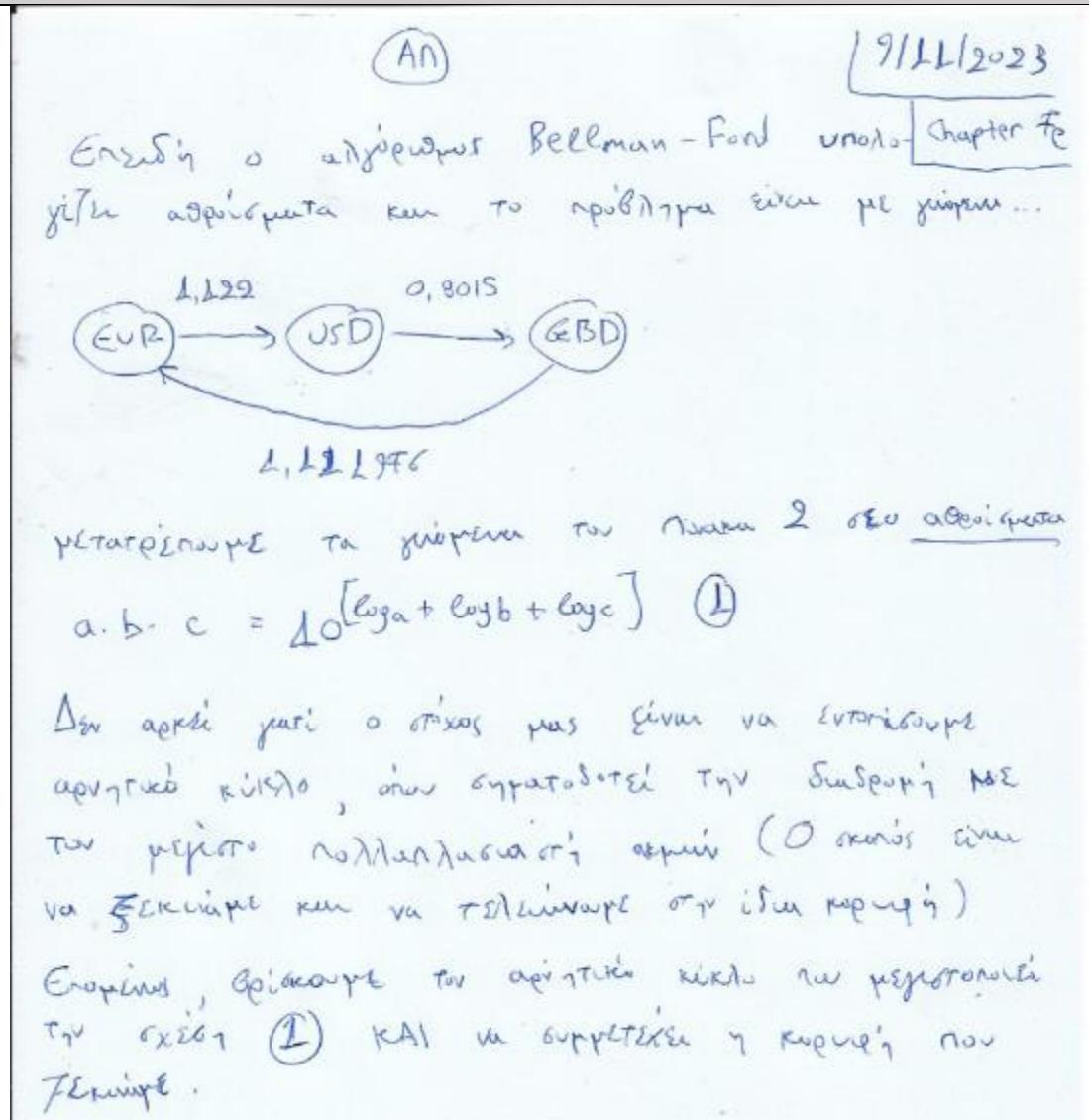


Σύνοψη

Τυπικά για να διαπιστώσουμε αν υπάρχουν αρνητικοί κύκλοι στο γράφημα θα πρέπει να υπάρχει ακμή ($u, v) \in E$ τέτοια ώστε $d_v^3 > d_u^3 + \omega(u, v)$. Οι ακμές για τις οποίες συνθίσται αυτό παρατίθενται στο Πίνακα 4. Οι κύκλοι αρνητικού μήκους που εντοπίστηκαν στον Πίνακα 4 απεικονίζονται στα Σχήματα 9, 10 και 11.

Μόνο οι κύκλοι που εμφανίζονται στα Σχήματα 9, 10 περιέχουν την κορυφή 1 (θέση σε ευρώ). Από αυτούς ο πρώτος έχει πολλαπλασιαστή (γινόμενο αρχικών ισοτιμιών) $10^{0.000034} = 1,00007829$ ενώ ο δεύτερος $10^{0.00002} = 1,000046$. Επομένως, επιλέγουμε τον πρώτο κύκλο αφού ο πολλαπλασιαστής του είναι μεγαλύτερος από αυτόν του δεύτερου. Άρα αν η αρχική μας θέση είναι 1.000.000 ευρώ μας συμφέρει να τα μετατρέψουμε σε δολλάρια, τα δολλάρια σε ελβετικό φρανκο, το ελβετικό φράνκο σε λίρες Αγγλίας και τέλος τις λίρες Αγγλίας πίσω σε ευρώ. Συνολικά από την αλυσίδα των συναλλαγών θα αποκομίσουμε $1.000.000 * 1,00007829 = 1.000.078,29$ ευρώ. Δηλαδή θα έχουμε κέρδος 78,29 ευρώ.

Σημειώσεις



Chapter 8

8a. Ιδέα και Αλγόριθμος Prim

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 64-68

Πρόβλημα

Στην Ενότητα 2.12.1 είδαμε ότι ένα γεννητορικό υπογράφημα ενός συνδεδεμένου μη-κατευθυνόμενου γραφήματος $G(V, E)$ το οποίο είναι δένδρο ονομάζεται συνεκτικό δένδρο του G . Προφανώς δεν υπάρχει μόνο ένα συνεκτικό δένδρο σε ένα μη-κατευθυνόμενο γράφημα G - ένα άνω φράγμα στο πλήθος των συνεκτικών δένδρων του G είναι το $|V(G)|^{|V(G)|-2}$ το οποίο αποτελεί τον αριθμό των συνεκτικών δένδρων σε ένα πλήρες γράφημα (τύπος του Cayley). Αν το μη-κατευθυνόμενο, συνδεδεμένο γράφημα G είναι βεβαρυμένο μας ενδιαφέρει να βρούμε το συνεκτικό δένδρο του μικρότερου βάρους, δηλαδή αυτό που ελαχιστοποιεί το άθροισμα των συντελεστών των ακμών του. Τυπικά, αν T είναι ένα συνεκτικό δένδρο του μη-κατευθυνόμενου, συνδεδεμένου, βεβαρυμένου γραφήματος $G(V, E, w)$, ορίζουμε το βάρος του δένδρου T ως

$$w(T) = \sum_{e \in E(T)} w(e).$$

Έστω $\mathcal{T}(G)$ το σύνολο των συνεκτικών δένδρων του $G(V, E, w)$. Το πρόβλημα του ελάχιστου συνεκτικού δένδρου (*ΕΣΔ*) συνίσταται στην εύρεση του συνεκτικού δένδρου T^* για το οποίο ισχύει ότι $T^* = \operatorname{argmin}\{w(T) : T \in \mathcal{T}(G)\}$.

Βασικές Ιδιότητες

Έστω $S \subseteq G$ ένα υπογράφημα ενός μη-κατευθυνόμενου γραφήματος $G(V, E)$. Θα συμβολίσουμε με $S + e$ το υπογράφημα (του G) που προκύπτει από την προσθήκη της ακμής $e \in E(G) \setminus E(S)$. Αντίστοιχα, το υπογράφημα που προκύπτει από την διαγραφή (αφαίρεση) μίας ακμής $e \in E(S)$ από το S θα συμβολίζεται με $S - e$.

Λήμμα 1^{ης} Βασικής Ιδιότητας

Λήμμα 33. Έστω T ένα συνεκτικό δένδρο του G και $e \in E(G) \setminus E(T)$. Το υπογράφημα $T + e$ περιέχει κύκλο.

Απόδειξη 1^{ης} Βασικής Ιδιότητας

Απόδειξη. Έστω v, u οι κορυφές που συνδέει η κορυφή e . Επειδή το T αποτελεί συνεκτικό δένδρο του G υπάρχει μονοπάτι $\langle v, \dots, u \rangle$ που συνδέει τις δύο αυτές κορυφές αποτελούμενο αποκλειστικά από ακμές του συνόλου $E(T)$. Προφανώς η προσθήκη της ακμής $e = \{u, v\} \notin E(T)$ δημιουργεί κύκλο. \square

Ο κύκλος που δημιουργείται από την προσθήκη της ακμής e στο συνεκτικό δένδρο T ονομάζεται θεμελιώδης.

Λήμμα 2^{ης} Βασικής Ιδιότητας

Λήμμα 34. Έστω T ένα συνεκτικό δένδρο του G , ακμή $e \in E(G) \setminus E(T)$ και ακμή $e' \neq e$ η οποία ανήκει στο θεμελιώδη κύκλο που περιέχει το υπογράφημα $T + e$. Το υπογράφημα $T' = T + e - e'$ αποτελεί συνεκτικό δένδρο του G .

Απόδειξης 2^{ης} Βασικής Ιδιότητας

Απόδειξη. Επειδή $e \notin E(T)$ και $e' \in E(T + e)$ έχουμε

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

Στη συνέχεια αρκεί να δείξουμε ότι T' είναι συνδεδεμένο. Έστω s, t δύο κορυφές του G . Επειδή το T είναι συνδεδεμένο - αποτελεί συνεκτικό δένδρο του G - περιέχει μονοπάτι ανάμεσα σε κάθε ζευγάρι κορυφών και άρα και ανάμεσα στις s, t . Αν το μονοπάτι αυτό δεν περιέχει την ακμή e' τότε το ίδιο μονοπάτι υπάρχει και στο T' . Στην αντίθετη περίπτωση, έστω $e = \{v, u\}$ και $e' = \{x, y\}$. Επειδή οι ακμές αυτές ανήκουν στον ίδιο κύκλο (Λήμμα 33), υπάρχει ένα μονοπάτι που εκκινεί από την s φτάνει στην κορυφή x (χρησιμοποιώντας ακμές του T) διανύει τις ακμές του κύκλου χωρίς να διασχίσει την ακμή $\{x, y\}$, φτάνει στην κορυφή y και στη συνέχεια χρησιμοποιώντας πάλι ακμές του T καταλήγει στην κορυφή t . Παρατηρούμε ότι όλες οι ακμές που χρησιμοποιήθηκαν ανήκουν στο T' . Συνεπώς, το T' είναι συνδεδεμένο. \square

Ιδέα

Έστω $S \subset G$, T_S ένα συνδετικό δένδρο του S , κορυφές v, u τέτοιες ώστε $v \in V(S), u \in V(G) \setminus V(S)$ και ακμή $\{v, u\} \in E(G) \setminus E(S)$. Παρατηρούμε ότι το $T_S + \{v, u\}$ αποτελεί συνεκτικό δένδρο του επαγόμενου υπογραφήματος του G με σύνολο κορυφών $V(S) \cup \{u\}$. Η παρατήρηση αυτή χρησιμοποιείται από τον αλγόριθμο του Prim προκειμένου να κατασκευαστεί ένα συνεκτικό δένδρο του G με τον εξής τρόπο. Αρχικώς θεωρούμε ένα υπογράφημα S κενό ακμών $E(S) = \emptyset$ το οποίο επάγεται από μία μόνο (τυχαία επιλεγμένη) κορυφή, έστω v . Προφανώς, η κορυφή αυτή ανήκει και στο συνεκτικό δένδρο T_S το οποίο σε αυτή τη φάση είναι τετριμένο αφού ισχύει $V(T_S) = V(S) = \{v\}$ και $E(T_S) = E(S) = \emptyset$. Προσθέτοντας στο T_S μία ακμή $e = \{v, u\} \in E(G) \setminus E(S)$, όπου $v \in V(S), u \in V(G) \setminus V(S)$ προκύπτει το υπογράφημα $T_S + e$ το οποίο (από την προηγούμενη παρατήρηση) αποτελεί συνεκτικό δένδρο του υπογραφήματος του G το οποίο επάγεται από το σύνολο κορυφών $V(S) \cup \{u\}$. Επαυξάνοντας σε κάθε επανάληψη το T_S με την εκάστοτε ακμή e και το σύνολο των κορυφών του S (και επομένως και του T_S) με την εκάστοτε κορυφή u , καταλήγουμε, μετά την ολοκλήρωση $|V(G)| - 1$ επαναλήψεων, να έχουμε $V(T_S) = V(S) = V(G)$ και άρα $S \equiv G$. Συνεπώς το αντίστοιχο γράφημα T_S αποτελεί συνεκτικό δένδρο του G .

Ο αλγόριθμος του Prim, χρησιμοποιεί τον παραπάνω μηχανισμό, επιλέγοντας σε κάθε βήμα από τις ακμές $\{v, u\} \in E(G) \setminus E(S)$, όπου $v \in V(S), u \in V(G) \setminus V(S)$, αυτή με το μικρότερο συντελεστή $w(v, u)$.

Αλγόριθμος

Αλγόριθμος 93 Αλγόριθμος Prim

Απαιτείται: Μη-κατευθυνόμενο συνδεδεμένο βεβαρυμένο γράφημα $G(V, E, w)$,
 $w : E(G) \rightarrow \mathbb{R}$.

Επιστρέφεται: Πίνακας *neighbor* που αποτυπώνει ένα ελάχιστο συνεκτικό δένδρου T του G .

```
1: function PRIM(graph  $G$ , float  $w[]$ , int  $neighbor[]$ )
2:    $V(S) \leftarrow \emptyset; E(T) \leftarrow \emptyset;$      $\triangleright$  Αρχικοποίηση συνόλων κορυφών και ακμών
   του  $T$ 
3:   for all  $v \in V(G)$  do
4:      $wneighbor[v] \leftarrow \infty;$ 
5:   end for
6:    $\triangleright$  Χωρίς βλάβη της γενικότητας η κορυφή 1 επιλέγεται να προσαρτηθεί
   πρώτη στο δένδρο
7:    $v \leftarrow 1;$ 
8:    $neighbor[v] \leftarrow 0;$ 
9:    $\bar{V}(S) \leftarrow V(G) \setminus \{v\};$ 
10:   $V(S) \leftarrow \{v\};$ 
11:  while  $|V(S)| \leq |V(G)| - 1$  do
12:    for all  $u \in N(v) \cap \bar{V}(S)$  do
13:      if  $wneighbor[u] > w(v, u)$  then
14:         $wneighbor[u] \leftarrow w(v, u);$ 
15:         $neighbor[u] \leftarrow v;$ 
16:      end if
17:    end for
18:     $\triangleright$  Επιλογή της επόμενης κορυφής  $v$  που θα προσαρτηθεί στο δένδρο
19:     $min\_weight \leftarrow \infty;$ 
20:    for all  $u \in \bar{V}(S)$  do
21:      if  $min\_weight > wneighbor[u]$  then
22:         $min\_weight \leftarrow wneighbor[u];$ 
23:         $v \leftarrow u;$ 
24:      end if
25:    end for
26:     $\bar{V}(S) \leftarrow \bar{V}(S) \setminus \{v\};$ 
27:     $V(S) \leftarrow V(S) \cup \{v\};$ 
28:  end while
29: end function
```

Λειτουργία

Μία κωδικοποίηση της παραπάνω διαδικασίας αποτελεί ο Αλγόριθμος 93. Ο αλγόριθμος διατηρεί και ενημερώνει το σύνολο κορυφών $V(S)$ - το οποίο ταυτίζεται με το $V(T)$ σε κάθε επανάληψη - και το συμπληρωματικό του $\bar{V}(S) = V(G) \setminus V(S)$. Εκτελούνται $|V(G)| - 1$ επανάληψεις. Σε κάθε επανάληψη μία κορυφή (ονομαστικά v) αφαιρείται από το σύνολο $\bar{V}(S)$ και προστίθεται στο $V(S)$. Κατά την αρχικοποίηση, η κορυφή 1 προστίθεται στο σύνολο $V(S)$ ενώ στο συμπληρωματικό του προστίθενται οι υπόλοιπες κορυφές (Γραμμές 9-10).

Ο αλγόριθμος χρησιμοποιεί τους μονοδιάστατους πίνακες $aweight$ και $neighbor$. Για κάθε κορυφή $u \in \bar{V}(S)$, στον πρώτο πίνακα αποθηκεύεται ο μικρότερος από τους συντελεστές των ακμών που συνδέουν την u με οποιαδήποτε γειτονική κορυφή του $V(S)$. Στο δεύτερο πίνακα αποθηκεύεται η αντίστοιχη γειτονική (της u) κορυφή στο $V(S)$. Σε κάθε επανάληψη τα στοιχεία των δύο αυτών δομών επανεκτιμούνται σε σχέση με την τελευταία κορυφή v η οποία προστέθηκε στο $V(S)$ (βρόχος - Γραμμές 14-19). Στη συνέχεια (Γραμμές 22-27) επιλέγεται η επόμενη κορυφή v η οποία θα προσαρτηθεί στο συνεκτικό δένδρο T - με αυτή στην επόμενη επανάληψη θα επανέξηθεί το $V(S)$. Το σύνολο των ακμών του δένδρου $E(T)$ επανέξανται με την ακμή $\{neighbor[v], v\}$, όπου η κορυφή v έχει υπολογιστεί από τις Γραμμές 22-27.

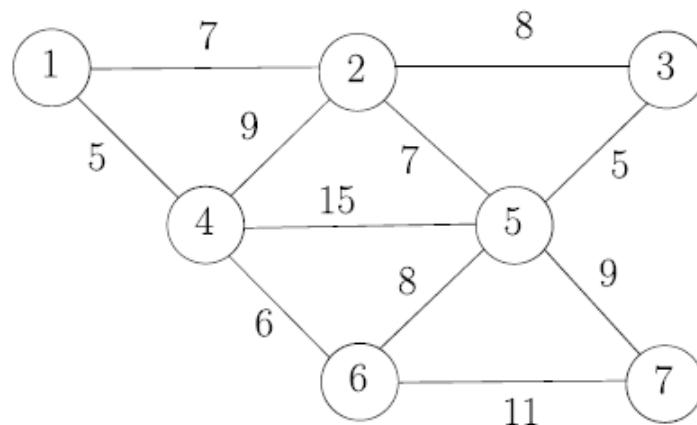
Όταν ο αλγόριθμος τερματίσει όλες οι κορυφές του γραφήματος θα έχουν προσαρτηθεί στο δένδρο ($V(S) = V(T) = V(G)$) ενώ οι ακμές του δένδρου θα έχουν

καταγραφεί στον πίνακα $neighbor$. Δηλαδή,

$$E(T) = \cup_{v \in V(G) \setminus \{1\}} \{neighbor[v], v\}$$

ενώ το βάρος του δένδρου δίνεται από το άθροισμα

$$w(T) = \sum_{v \in V(G) \setminus \{1\}} nweight[v].$$



Σχήμα 13.5: Εύρεση ελάχιστου συνεκτικού δένδρου

Το ελάχιστο συνεκτικό δένδρο σχηματίζεται από τις κόκκινες ακμές στο Σχήμα 13.6 - το άθροισμα των συντελεστών των ακμών που συμμετέχουν σε αυτό είναι ίσο με 39.

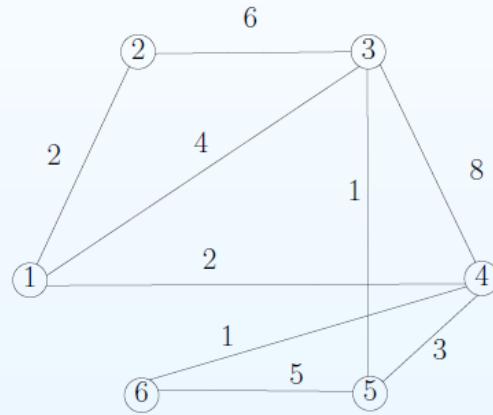
8b. Παράδειγμα Prim

Διαφάνειες

Slides_and_Exercises→06_sptrees.pdf σελ. 13-18

Παράδειγμα

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



Σχήμα 3: Γράφημα $G(V, E, w)$

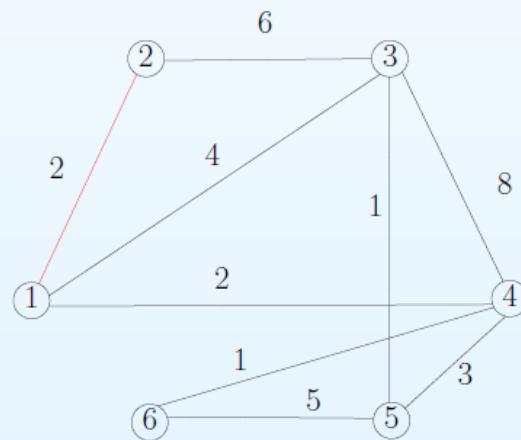
Λύση Αρχικώς έστω $V(T) = \{1\}$

Επανάληψη 1

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

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

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



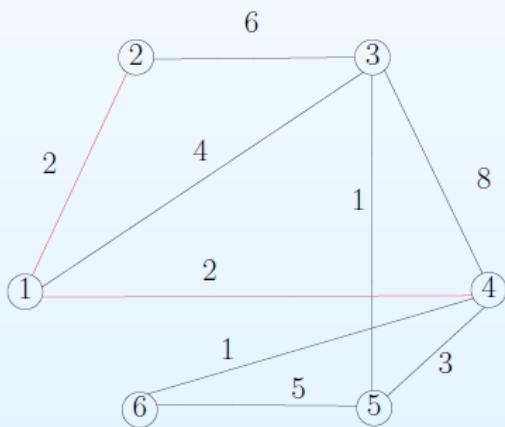
Σχήμα 3: Με κόκκινο οι ακμές του δένδρου

Επανάληψη 2

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

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

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

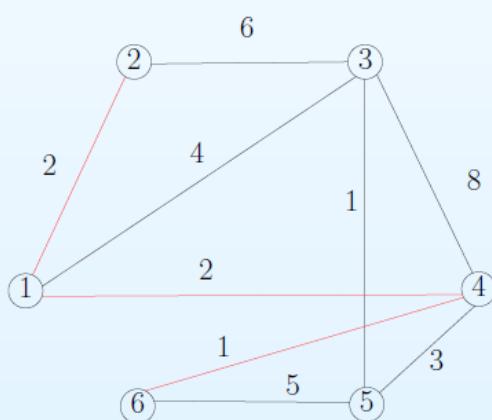


Σχήμα 3: Με κόκκινο οι ακμές του δένδρου
Επανάληψη 3

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

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

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



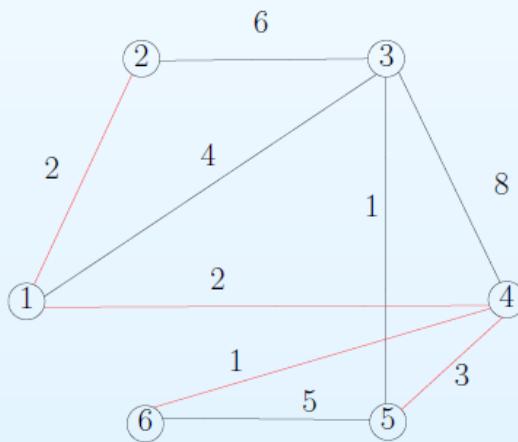
Σχήμα 3: Με κόκκινο οι ακμές του δένδρου

Επανάληψη 4

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

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

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



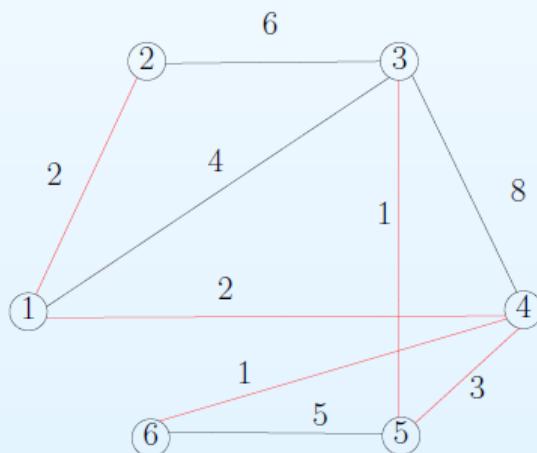
Σχήμα 3: Με κόκκινο οι ακμές του δένδρου

Επανάληψη 5

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

$$V(T) = \{1, 2, 4, 6, 5, 3\}$$

$$E(T) = \{(1, 2), (1, 4), (4, 6), (4, 5), (5, 3)\}$$



Σχήμα 3: Με κόκκινο οι ακμές του δένδρου

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 70-71

Λήμμα Ορθότητας

Είναι προφανές ότι ο αλγόριθμος του Prim παράγει ένα συνδεδεμένο υπογράφημα του G με $|V(G)| - 1$ ακμές. Επομένως παράγει ένα συνεκτικό δένδρο του G . Στην παρακάτω πρόταση αποδεικνύεται ότι το δένδρο αυτό είναι και ελάχιστου βάρους.

Λήμμα 35. Έστω T το συνεκτικό δένδρο που υπολογίζει ο αλγόριθμος του Prim όταν εκτελείται σε ένα συνδεδεμένο μη-κατευθυνόμενο και βεβαρυμένο γράφημα $G(V, E, w)$. Το T είναι ένα ελάχιστο συνεκτικό δένδρο του γραφήματος G .

Απόδειξη Ορθότητας

Απόδειξη. Έστω T^* ένα ελάχιστο συνεκτικό δένδρο του G . Θα δείξουμε ότι $w(T^*) = w(T)$. Αν $T = T^*$ τότε αυτό ισχύει. Στην αντίθετη περίπτωση ($T \neq T^*$) υπάρχει ακμή $\{v, u\} \in E(T) \setminus E(T^*)$.

Στο συνεκτικό δένδρο T^* υπάρχει κάποιο μονοπάτι που συνδέει τις κορυφές v και u . Επειδή η ακμή $\{v, u\}$ δεν ανήκει στο δένδρο αυτό η προσθήκη της σε αυτό δημιουργεί κύκλο (Λήμμα 33). Στον κύκλο αυτό υπάρχει ακμή $\{x, y\}$ για την οποία ισχύει ότι

$$w(v, u) \leq w(x, y). \quad (13.16)$$

Αυτό συμβαίνει γιατί αν $w(v, u) > w(x, y)$ για κάθε ακμή $\{x, y\}$ του κύκλου τότε στην επανάληψη στη οποία ο αλγόριθμος προσέθεσε την ακμή $\{v, u\}$ στο T η ακμή αυτή δεν είχε το μικρότερο συντελεστή ανάμεσα στις ακμές που συνέδεαν το μέχρι εκείνη-τη-στιγμή υποσύνολο του $V(T)$ με τις υπόλοιπες κορυφές του γραφήματος (άτοπο).

Σύμφωνα με το Λήμμα 34 το υπογράφημα $\hat{T}^* = T^* + \{v, u\} - \{x, y\}$ του G είναι συνεκτικό δένδρο. Το βάρος του είναι $w(\hat{T}^*) = w(T^*) + w(v, u) - w(x, y)$. Από την (13.16) έχουμε ότι $w(\hat{T}^*) \leq w(T^*)$ και επειδή εξ' υποθέσεως το T^* είναι συνεκτικό δένδρο ελάχιστου βάρους η σχέση αυτή ισχύει ως ισότητα, δηλαδή $w(\hat{T}^*) = w(T^*)$.

Επειδή $|E(T) \setminus E(\hat{T}^*)| = |E(T) \setminus E(T^*)| - 1$, μπορούμε να επαναλάβουμε τη διαδικασία αυτή μετατρέποντας το T^* στο T χωρίς να αλλάξει το βάρος $w(T^*)$. Συνεπώς, $w(T^*) = w(T)$. \square

Λήμμα Πολυπλοκότητας

Λήμμα 36. Ο Αλγόριθμος 93 εκτελεί $O(|V(G)|^2)$ ΣΥΒ.

Απόδειξη Πολυπλοκότητας

Απόδειξη. Σε κάθε επανάληψη του βρόχου της Γραμμής 11 (εξωτερικός βρόχος) εκτελούνται $(|V(G)|)$ ΣΥΒ από τον βρόχο των Γραμμών 14-19 και $(|V(G)|)$ ΣΥΒ από τον βρόχο των Γραμμών 22-27. Άρα σε κάθε επανάληψη του εξωτερικού βρόχου εκτελούνται $(|V(G)|)$ ΣΥΒ. Ο αριθμός των επαναλήψεων του εξωτερικού βρόχου είναι επίσης $|V(G)|$. \square

Χρησιμοποιώντας πιο σύνθετες δομές δεδομένων (σωροί Fibonacci) μπορούμε να επιτύχουμε πολυπλοκότητα $O(|E(G)| + |V(G)| \lg |V(G)|)$.

8d. Ιδέα, Αλγόριθμος, Παράδειγμα, Ορθότητα και Πολυπλοκότητα Kruskal

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 71-74

Ιδέα

Ίσως έναν πιο διαισθητικό τρόπο για την εύρεση του ελάχιστου συνεκτικού δένδρου προσφέρει ο αλγόριθμος του Kruskal. Η ιδέα πίσω από τη μέθοδο αυτή είναι απλή: το ελάχιστο συνεκτικό δένδρο «χτίζεται» σε μία σειρά από επαναλήψεις: σε κάθε επανάληψη, από το σύνολο των ακμών επιλέγεται αυτή με το μικρότερο συντελεστή η οποία δεν δημιουργεί κύκλο όταν προστεθεί στο σύνολο των ήδη επιλεχθέντων ακμών από τις προηγούμενες επαναλήψεις. Ο αλγόριθμος ολοκληρώνεται μετά από $|V(G)| - 1$ επαναλήψεις - όταν έχουν επιλεγεί $|V(G)| - 1$ ακμές. Μία «υψηλού επιπέδου» κωδικοποίηση της διαδικασίας αυτής αποτελεί ο Αλγόριθμος 94. Η Γραμμή 3 του αλγόριθμου ουσιαστικά περιγράφει την ταξινόμηση των ακμών με βάση τους συντελεστές τους.

Αλγόριθμος

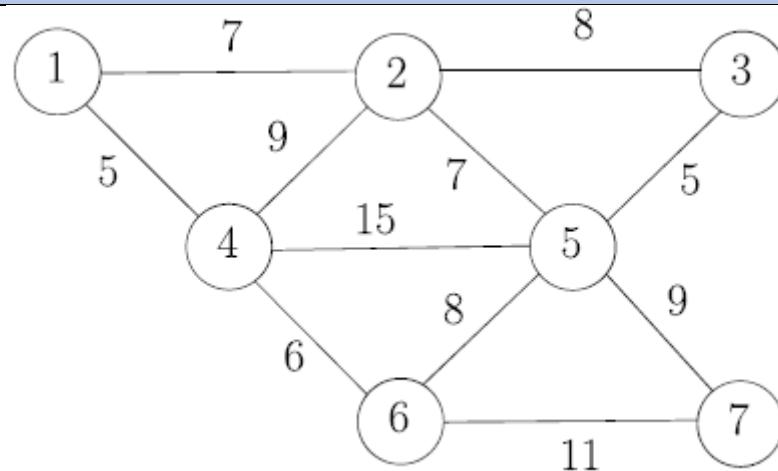
Αλγόριθμος 94 Αλγόριθμος Kruskal

Απαιτείται: Μη-κατευθυνόμενο συνδεδεμένο βεβαρυμένο γράφημα $G(V, E, w)$, $w : E(G) \rightarrow \mathbb{R}$.

Επιστρέφεται: Ελάχιστο συνεκτικό δένδρου T του G .

```
1: function KRUSKAL(graph  $G$ , float  $w[]$ , graph  $T$ )
2:    $m \leftarrow |E(G)|$ ;
3:   Έστω  $E(G) = \{e_{i_1}, e_{i_2}, \dots, e_{i_m}\}$  ώστε  $w(e_{i_1}) \leq \dots \leq w(e_{i_m})$ 
4:    $E(T) \leftarrow \emptyset$ ;
5:    $t \leftarrow 1$ ;
6:   while  $|E(T)| < |V(G)| - 1$  do
7:     if  $E(T) \cup \{e_{i_t}\}$  δεν περιέχει κύκλο then
8:        $E(T) \leftarrow E(T) \cup \{e_{i_t}\}$ 
9:     end if
10:     $t++$ ;
11:   end while
12: end function
```

Σχήμα 13.5 Εύρεση ελάχιστου συνεκτικού δένδρου



Πίνακας 13.1 Ταξινομημένες ακμές του γραφήματος του Σχήματος 13.5

Ακμή	Συντελεστής
{1, 4}	5
{3, 5}	5
{4, 6}	6
{1, 2}	7
{2, 5}	7
{2, 3}	8
{5, 6}	8
{2, 4}	9
{5, 7}	9
{6, 7}	11
{4, 5}	15

Παράδειγμα

Παράδειγμα 75. Θα εκτελέσουμε τον Αλγόριθμο 94 στο γράφημα του Σχήματος 13.5. Η ταξινομημένη λίστα των ακμών με βάση τους συντελεστές τους παρουσιάζεται στον Πίνακα 13.1

- Αρχικοποίηση

$$E(T) = \emptyset$$

- Επανάληψη: 1

Η ακμή {1, 4} δεν δημιουργεί κύκλο με τις ακμές που ανήκουν στο $E(T)$, επομένως προστίθεται

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

- Επανάληψη: 2

Η ακμή $\{3, 5\}$ δεν δημιουργεί κύκλο με τις ακμές που ανήκουν στο $E(T)$, επομένως προστίθεται

$$E(T) = \{\{1, 4\}, \{3, 5\}\}$$

- Επανάληψη: 3

Η ακμή $\{4, 6\}$ δεν δημιουργεί κύκλο με τις ακμές που ανήκουν στο $E(T)$, επομένως προστίθεται

$$E(T) = \{\{1, 4\}, \{3, 5\}, \{4, 6\}\}$$

- Επανάληψη: 4

Η ακμή $\{1, 2\}$ δεν δημιουργεί κύκλο με τις ακμές που ανήκουν στο $E(T)$, επομένως προστίθεται

$$E(T) = \{\{1, 4\}, \{3, 5\}, \{4, 6\}, \{1, 2\}\}$$

- Επανάληψη: 5

Η ακμή $\{2, 5\}$ δεν δημιουργεί κύκλο με τις ακμές που ανήκουν στο $E(T)$, επομένως προστίθεται

$$E(T) = \{\{1, 4\}, \{3, 5\}, \{4, 6\}, \{1, 2\}, \{2, 5\}\}$$

- Επανάληψη: 6

Η ακμή $\{2, 3\}$ δημιουργεί κύκλο με τις ακμές $\{2, 5\}, \{3, 5\}$ που ήδη ανήκουν στο $E(T)$, επομένως δεν προστίθεται σε αυτό.

- Επανάληψη: 7

Η ακμή $\{5, 6\}$ δημιουργεί κύκλο με τις ακμές $\{1, 2\}, \{2, 5\}, \{1, 4\}, \{4, 6\}$ που ήδη ανήκουν στο $E(T)$, επομένως δεν προστίθεται σε αυτό.

- Επανάληψη: 8

Η ακμή $\{2, 4\}$ δημιουργεί κύκλο με τις ακμές $\{1, 2\}, \{1, 4\}$ που ήδη ανήκουν στο $E(T)$, επομένως δεν προστίθεται σε αυτό.

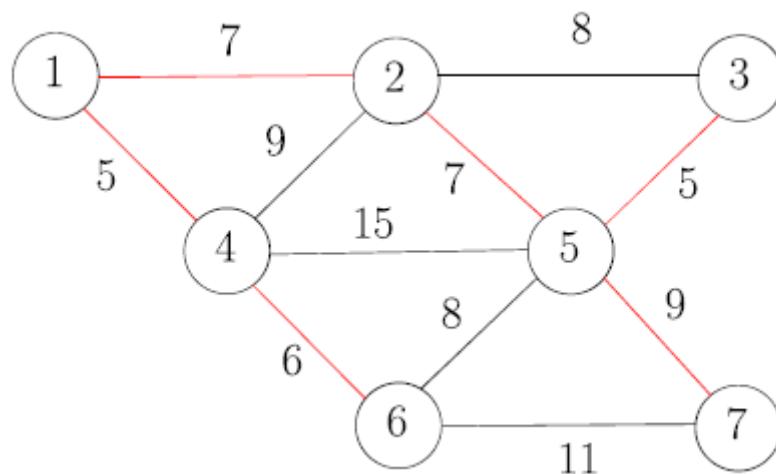
- Επανάληψη: 9

Η ακμή $\{5, 7\}$ δεν δημιουργεί κύκλο με τις ακμές που ανήκουν στο $E(T)$, επομένως προστίθεται

$$E(T) = \{\{1, 4\}, \{3, 5\}, \{4, 6\}, \{1, 2\}, \{2, 5\}, \{5, 7\}\}$$

Στο σημείο αυτό η διαδικασία ολοκληρώνεται αφού ο αριθμός των ακμών του $E(T)$ είναι 7· ίσος με τον αριθμό των κορυφών του γραφήματος μείον 1.

Το συνεκτικό δένδρο που βρήκε ο αλγόριθμος είναι ακριβώς αυτό που υπολόγισε ο Αλγόριθμος 93 - Σχήμα 13.6.



Λήμμα Ορθότητας

Από το παραπάνω παράδειγμα παρατηρούμε ότι το υπογράφημα που αντιστοιχεί στο σύνολο ακμών που υπολογίζει ο Αλγόριθμος 94 δεν είναι σε κάθε επανάληψη συνδεδεμένο.³ Όμως στην έξοδο του αλγόριθμου το σύνολο των ακμών περιγράφει ένα συνδετικό δένδρο αφού από κατασκευής δεν περιέχει κύκλο και έχει πληθάριθμο ίσο με τον αριθμό των κορυφών του γραφήματος G μείον ένα (Ενότητα 2.12.1). Η παρακάτω πρόταση είναι αντίστοιχη του Λήμματος 35.

Λήμμα 37. Έστω T το συνεκτικό δένδρο που υπολογίζει ο Αλγόριθμος 94 όταν εκτελείται σε ένα συνδεδεμένο μη-κατευθυνόμενο και βεβαρυμένο γράφημα $G(V, E, w)$. Το T είναι ένα ελάχιστο συνεκτικό δένδρο του γραφήματος G .

Απόδειξη Ορθότητας

Απόδειξη. Έστω T^* ένα ελάχιστο συνεκτικό δένδρο του G . Θα δείξουμε ότι $w(T^*) = w(T)$. Άν $T = T^*$ τότε αυτό ισχύει. Στην αντίθετη περίπτωση ($T \neq T^*$) έστω $e \in E(T) \setminus E(T^*)$ η ακμή με το μικρότερο συντελεστή από όλες τις ακμές της διαφοράς των δύο συνόλων ακμών. Το υπογράφημα $T^* + e$ περιέχει κύκλο (Λήμμα 33),

ονομαστικά C . Επειδή το T είναι δένδρο υπάρχει ακμή e' που ανήκει στον κύκλο C αλλά όχι στο δένδρο T . Άρα η ακμή αυτή ανήκει στη διαφορά $E(T^*) \setminus E(T)$. Παρατηρούμε ότι

$$w(e) \leq w(e') \quad (13.17)$$

γιατί διαφορετικά η ακμή e' θα είχε εξετασθεί πριν από την e από τον αλγόριθμο και θα είχε προστεθεί αντί αυτής στο $E(T)$.

Το υπογράφημα $\hat{T}^* = T^* - e' + e$ αποτελεί συνεκτικό δένδρο για το G (Λήμμα 34) και περιέχει μία παραπάνω κοινή ακμή - την ακμή e - με το T από ότι το T^* . Από την (13.17) έχουμε ότι $w(\hat{T}^*) \leq w(T^*)$ και επειδή επειδή εξ' υποθέσεως το T^* είναι συνεκτικό δένδρο ελάχιστου βάρους η σχέση αυτή ισχύει ως ισότητα, δηλαδή $w(\hat{T}^*) = w(T^*)$.

Επαναλαμβάνοντας την ίδια διαδικασία για κάθε ακμή του συνόλου $E(T) \setminus E(T^*)$ μπορούμε να παράγουμε από το T^* το T χωρίς να αλλάξει το βάρος $w(T^*)$. Συνεπώς, $w(T^*) = w(T)$. \square

Απόδειξη Πολυπλοκότητας

Ένα δύσκολο σημείο στην υλοποίηση του αλγόριθμου του Kruskal είναι ο έλεγχος περί της υπάρξεως κύκλου σε ένα υποσύνολο των ακμών του G . Επειδή ο έλεγχος αυτός πραγματοποιείται σε κάθε επανάληψη δεν είναι υπολογιστικά αποτελεσματικό να γίνεται μέσω της ΔκΒ όπου μπορεί να διαγνωστεί κύκλος με την ύπαρξη (ή όχι) οπισθοακμής (Κεφάλαιο 12). Προκειμένου να γίνει αποτελεσματικός ο έλεγχος αυτός χρησιμοποιείται η ειδική δομή UnionFind [12, 7, 6]. Με τη χρήση της δομής αυτής ο αλγόριθμος χαρακτηρίζεται από την πολυπλοκότητα της διαδικασίας ταξινόμησης των ακμών σε σχέση με τους συντελεστές τους, ήτοι $O(|E(G)| \lg |E(G)|)$.

8e. Θεωρία πάνω στο Ελάχιστο Συνδετικό Δένδρο

Διαφάνειες

Slides_and_Exercises→07_spstexer.pdf σελ. 15-17

Εκφώνηση

Άσκηση 9 Υποθέστε ότι $G(V, E, w)$ είναι ένα βεβαρυμένο μη-κατευθυνόμενο συνδεδεμένο γραφήμα. Να απαντήσετε αν είναι σωστές ή λάθος οι παρακάτω προτάσεις.

- Το δένδρο των ελάχιστων διαδρομών από οποιαδήποτε κορυφή s είναι συνδετικό δένδρο του γραφήματος.

1. Το δένδρο των ελάχιστων διαδρομών από οποιαδήποτε κορυφή s είναι συνδετικό δένδρο του γραφήματος

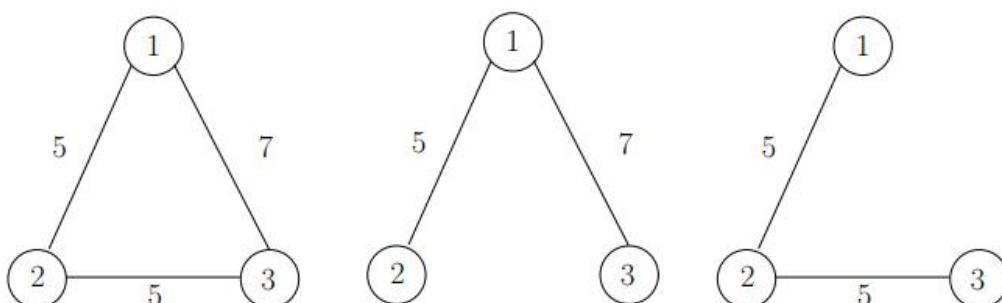
- Σωστό

2. Αν προσθέσουμε μία σταθερά b στο βάρος κάθε ακμής το ΕΣΔ αλλάζει

- Λάθος

3. Το ΕΣΔ και το δένδρο των ελάχιστων διαδρομών ανεξάρτητα της αφετηριακής κορυφής s ταυτίζονται σε κάθε γράφημα G

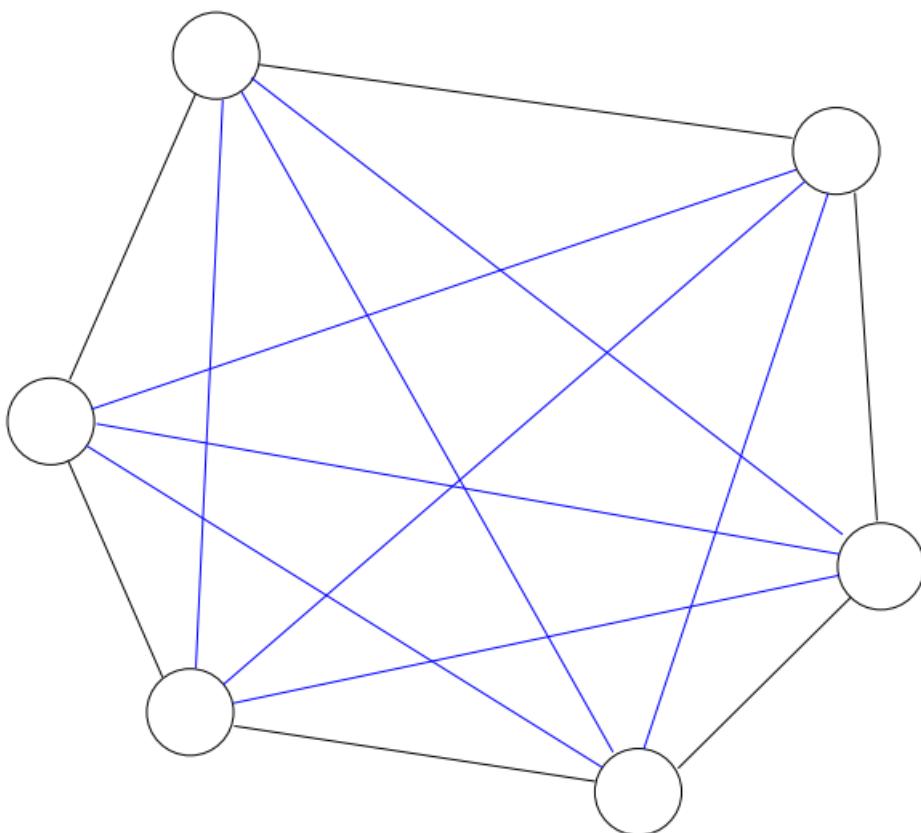
- Λάθος. Στο γράφημα του Σχήματος 13 (αριστερή εικόνα) το δένδρο των ελαχίστων διαδρομών με αφετηριακή κορυφή την 1 απεικονίζεται στην κεντρική εικόνα ενώ το ΕΣΔ στη δεξιά.



Σχήμα 13: Γράφημα, Δένδρο Συντ. Διαδρομών από κορυφή 1, Ελάχιστο Συνδετικό Δένδρο

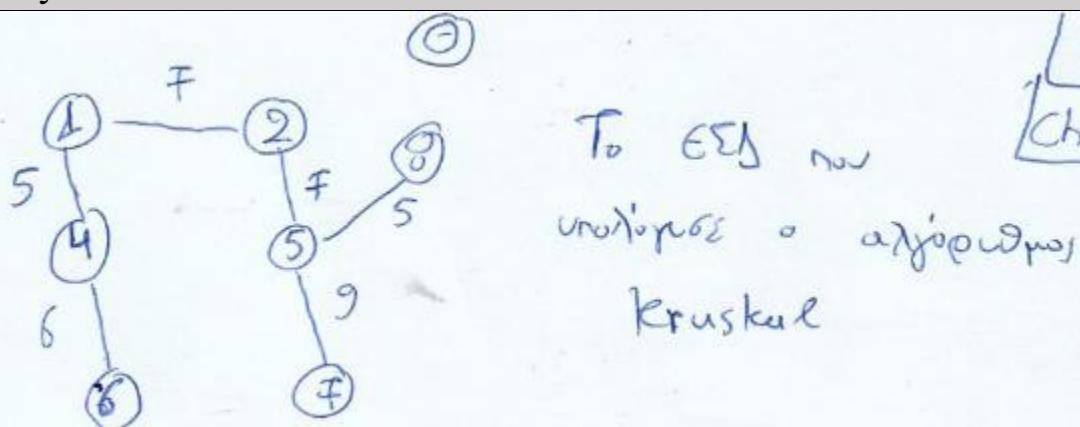
4. Υπάρχει για κάθε γράφημα G μία αφετηριακή κορυφή s για την οποία το δένδρο των ελάχιστων διαδρομών ταυτίζονται με το ΕΣΔ

4. Λάθος. Στο γράφημα του Σχήματος 14 θεωρείστε ότι οι μαύρες ακμές έχουν βάρος 1 ενώ οι μπλε έχουν βάρος 2. Οποιοδήποτε ΕΣΔ αποτελείται από μαύρες ακμές ενώ όποιοδήποτε Δένδρο Συντομότερων διαδρομών θα περιλαμβάνει τουλάχιστον μία μπλέ ακμή αφού για να φτάσουμε στη συμμετρική κορυφή από αυτή της αφετηρίας (όποια και αν είναι αυτή) θα έχουμε κόστος 3 ενώ μέσω της μπλε κορυφής θα έχουμε κόστος 2.



Σχήμα 14: Δένδρο Συντ. Διαδρομών, Ελάχιστο Συνδετικό Δένδρο

Σημειώσεις



Chapter 9

9a. Εισαγωγή και Χρησιμότητα του Δ.Π. μέσα από τον αλγόριθμο Fibonacci

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 79-82

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 ακολούθησε 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.

Διαίρει και Κυρίευε Vs Δυναμικός Προγραμματισμός

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.

Χρησιμότητα Δ.Π μέσα από τον αλγόριθμο Fibonacci

Ένα απλό παράδειγμα, μέσω του οποίου θα αναδειχθούν τα ιδιαίτερα χαρακτηριστικά του ΔΠ αποτελεί η χρήση του στον υπολογισμό του n -ισοτού όρου ($n \geq 0$) της ακολουθίας Fibonacci. Συμβολίζουμε το πρόβλημα αυτό ως $F(n)$. Από την (6.3) γνωρίζουμε ότι για μεγάλες τιμές του n η λύση του προκύπτει από το άθροισμα των λύσεων των υποπροβλημάτων $F(n-1), F(n-2)$ ενώ η λύση του $F(n-1)$ προκύπτει από τη λύση των $F(n-2), F(n-3)$, η λύση του $F(n-2)$ προκύπτει από τη λύση των $F(n-3), F(n-4)$, κοκ. Επίσης από την ίδια σχέση, γνωρίζουμε άμεσα τη λύση του υποπροβλήματος $F(0)$ καθώς και του $F(1)$. Επομένως τα υποπροβλήματα στα οποία αποσυντίθεται το αρχικό πρόβλημα $F(n)$ είναι: $F(0), F(1), F(2), \dots, F(n-2), F(n-1)$. Θεωρούμε τα υποπροβλήματα αυτά σαν στάδια (υπολογισμού): σε καθένα από αυτά - εκτός από τα δύο πρώτα - υπολογίζεται ο αντίστοιχος όρος της ακολουθίας - λύση του υποπροβλήματος που αντιστοιχεί στο στάδιο αυτό - από τη λύση των δύο προηγουμένων σταδίων. Στην περίπτωση αυτή δεν υπάρχει θέμα αξιολόγησης εναλλακτικών αποφάσεων με βάση την οποία θα υπολογιστεί η λύση του υποπροβλήματος. Άν f_i συμβολίζει τη λύση του υποπροβλήματος $F(i)$, για $i = 0, \dots, n$, η λύση των υποπροβλημάτων καθορίζεται από την αναδρομική σχέση

$$\begin{aligned}f_i &= f_{i-1} + f_{i-2}, i = 2, \dots, n, \\f_1 &= 1, \\f_0 &= 0.\end{aligned}$$

Από την παραπάνω σχέση είναι άμεσο ότι προκειμένου να υπολογιστεί η τιμή της f_i , για $i \geq 2$, θα πρέπει να έχουν υπολογιστεί οι τιμές f_{i-1}, f_{i-2} . Άρα θα πρέπει ο υπολογισμός των λύσεων των υποπροβλημάτων να γίνει με τη σειρά $F(0), F(1), \dots, F(n-1), F(n)$. Επίσης γνωρίζοντας τις τιμές f_{i-1}, f_{i-2} (αποθηκεύοντας τις στη μνήμη) μπορούμε να υπολογίσουμε σε σταθερό αριθμό ΣΥΒ την f_i . Ο Αλγόριθμος 95 που επιλύει το πρόβλημα $F(n)$ προκύπτει άμεσα από την παραπάνω ανάλυση.

Αλγόριθμος Fibonacci

Αλγόριθμος 95 Υπολογισμός αριθμών Fibonacci.

Απαιτείται: Ακέραιος $n \geq 0$.

Επιστρέφεται: n -ιστός αριθμός Fibonacci.

```
1: function FIBODYNAMIC(int n)
2:   if  $n \leq 1$  then
3:      $f_n \leftarrow n;$ 
4:   else
5:      $f_{n-2} \leftarrow 0; f_{n-1} \leftarrow 1;$ 
6:      $i \leftarrow 2;$ 
7:     while  $i \leq n$  do
8:        $f_n \leftarrow f_{n-1} + f_{n-2};$ 
9:        $f_{n-2} \leftarrow f_{n-1};$ 
10:       $f_{n-1} \leftarrow f_n;$ 
11:       $i ++;$ 
12:    end while
13:  end if
14:  return  $f_n;$ 
15: end function
```

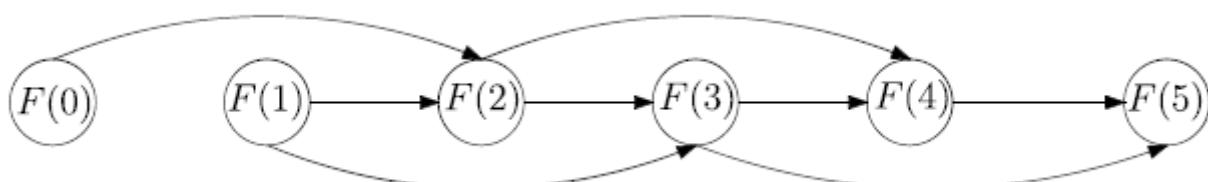
Βελτίωση στην Πολυπλοκότητα

Συγκρίνοντας τον Αλγόριθμο 95 με τον Αλγόριθμο 43 παρατηρούμε ότι ο πρώτος είναι πολύ πιο αποτελεσματικός από το δεύτερο· είναι εύκολο να δείξει κανείς ότι η πολυπλοκότητα του πρώτου είναι $\Theta(n)$ ενώ του δεύτερου $\Theta(2^n)$. Αυτό είναι αποτέλεσμα της χρήσης της μνήμης από τον πρώτο αλγόριθμο αφού όποτε υπολογίζεται μία τιμή f_i αποθηκεύεται στη μνήμη και χρησιμοποιείται τόσο στον υπολογισμό της f_{i+1} όσο και στον υπολογισμό της f_{i+2} . Αντίθετα στην περίπτωση του Αλγόριθμου 43 ο υπολογισμός της τιμής αυτής γίνεται δύο φορές - μία για τον υπολογισμό της f_{i+1} και άλλη μία για τον υπολογισμό της f_{i+2} .

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 - 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 κατευθυνόμενο άκυκλο (*KAG*) (Ενότητα 12.4.2). 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 95, το γράφημα των υποπροβλημάτων για την επίλυση του προβλήματος $F(5)$ απεικονίζεται στο Σχήμα 14.1. 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.

Σχήμα 14.1 Κατευθυνόμενο γράφημα υποπροβλημάτων για τον υπολογισμό του πέμπτου όρου της ακολουθίας Fibonacci



Πολυπλοκότητα στον Δ.Π.

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 95 έχουμε $\Theta(n)$ - το πλήθος - υποπροβλήματα, το καθένα από τα οποία λύνεται σε σταθερό χρόνο και άρα η πολυπλοκότητα του αλγόριθμου είναι επίσης $\Theta(n)$.

Σημειώσεις

Θ ① OB - ΔΥΝΑΜΙΚΩΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Fibonacci

21/11/2023
Chapter 9a

$F(0) \rightarrow F(1) \rightarrow F(2) \rightarrow F(3) \rightarrow F(4) \rightarrow F(5)$

$$f(n) = \begin{cases} f(n-1) + f(n-2), & n \geq 2 \\ n, & 0 \leq n \leq 1 \end{cases}$$

Function Fibo(int n)

if $n \leq 1$ then

return n ;

else

return $f(n-1) + f(n-2)$

end if

end function

Fibonacci με αναρροφή

$O(2^n)$!

$f(n)$

$f(n-1)$ $f(n-2)$

$f(n-2)$ $f(n-3)$ $f(n-3)$ $f(n-4)$

Δημιουργία παντία
διαδικτικός γένερος και
θί αυτό ο αλγόριθμος εκτελείται σε εκθετικό χρόνο, καθώς
υπολογίζεται ένα υποπόλυτο 2 φορές \oplus

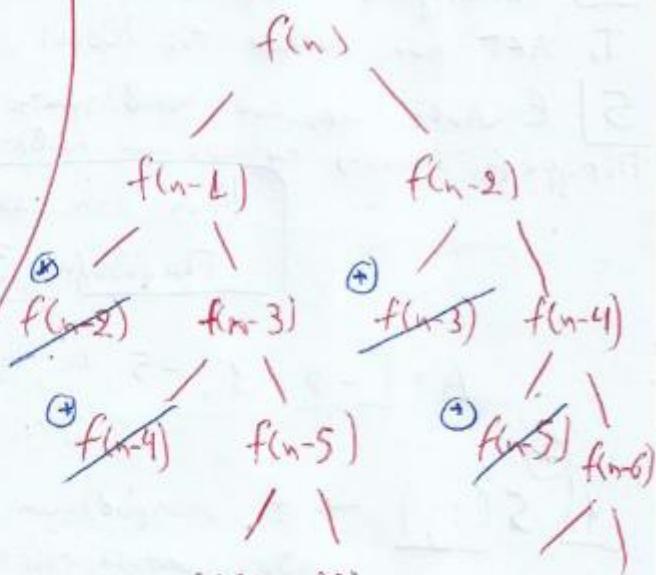
```

function FiboDynamic (int n)
if n ≤ 1 then
    fn ← n;
    return fn;
else
    fn-2 ← 0;
    fn-1 ← 1;
    i ← 2;
    while i ≤ n do
        fn ← fn-2 + fn-1;
        fn-2 ← fn-1;
        fn-1 ← fn;
        i++;
    end while
end if
return fn;
end function

```

Fibonacci με ΔΥΝΑΜΙΚΟ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

$O(n)$ ✓



Ein Ser xerüftan va

Emarginatagracilis Ta unopened specimen

now than unadjusted and adjusted unadjusted.

To ride your bicycle without fear is like flying.

• αλιγωνος εκτελεσην οε νομουνυμει χρων

9b. Τα 5 στάδια μεθοδολογίας του Δυναμικού Προγραμματισμού

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 82-83

1. Ορισμός υποπροβλημάτων

Στο Στάδιο 1 θα πρέπει να οριστούν τα υποπροβλήματα. Οι ορισμοί περιλαμβάνουν μία λεκτική και μία συμβολική περιγραφή. Η συμβολική περιγραφή συνήθως γίνεται μέσω δεικτών που ορίζουν πρόθεμα (*prefix*), επίθεμα (*suffix*) ή μέρος της συμβολοσειράς εισόδου.

2. Καθορισμός επιλογών

Στο Στάδιο 2 θα πρέπει για κάθε υποπρόβλημα να καθοριστούν ποιες είναι οι επιλογές που μπορούν να οδηγήσουν στη λύση του με βάση τις λύσεις υποπροβλημάτων μικρότερου μεγέθους.

3. Ορισμός αναδρομικής σχέσης

Έτσι φτάνουμε στο Στάδιο 3 όπου μπορούμε πλέον να διατυπώσουμε μία αναδρομική σχέση η οποία σχετίζει τη λύση του παρόντος υποπροβλήματος με την καλύτερη από τις επιλογές που καθορίστηκαν στο προηγούμενο στάδιο και συνδυάζουν λύσεις μικρότερων υποπροβλημάτων. Δηλαδή, η αναδρομική σχέση μας δίνει τη λύση του παρόντος υποπροβλήματος ως συνάρτηση της καλύτερης επιλογής συνδυασμού λύσεων που έχουν παραχθεί από προηγούμενες επιλογές.

4. Τοπολογική ταξινόμηση υποπροβλημάτων

Στο Στάδιο 4 καθορίζεται η σειρά με την οποία λύνονται τα υποπροβλήματα έτσι ώστε οι λύσεις των υποπροβλημάτων που χρειάζεται η αναδρομική σχέση για να παράγει τη λύση του εκάστοτε παρόντος υποπροβλήματος να είναι διαθέσιμες στη μνήμη της ΜΑΠ. Όπως είδαμε και παραπάνω, αυτό είναι ισοδύναμο με το να αποδείξουμε ότι το γράφημα που συνδέει τα υποπροβλήματα είναι ΚΑΓ.¹ Η σειρά επίλυσης των υποπροβλημάτων είναι αυτή που υποδεικνύει η τοπολογική ταξινόμηση του γραφήματος αυτού.

5. Επίλυση αρχικού προβλήματος

Στο τελευταίο στάδιο (Στάδιο 5) περιγράφεται το πως επιλύεται το αρχικό πρόβλημα με βάση τις λύσεις των υποπροβλημάτων.

Τα 5 στάδια Μεθοδολογίας Δ.Π. | Chapter 9b

1] Ορισμός Υποπροβλημάτων

Λεκτική και Συρβολική περιγραφή : $S(:j)$ πρότερη

$S(j:)$ επιτέλη

2] Καθορισμός Ενδοξίν

Οι ενδοξίνες για την λύση των επώνευν υποπροβλημάτων

3] Θερμός Αναδρομικής Σειράς

Η σειρά που διαδίδει τις λύσεις των υποπροβλημάτων

4] Τοπολογική Ταχινικότητα Υποπροβλημάτων

Το AFG που διαδίδει τις λύσεις των υποπροβλημάτων

5] Επίλυση αρχικών προβλημάτων

Περικρατή επίλυση των αρχικών προβλημάτων από τις λύσεις των υποπροβλημάτων

9c. Το πρόβλημα εύρεσης της υπακολουθίας με το μέγιστο άθροισμα (Προθεματική Περιγραφή)

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 83-85

Πρόβλημα

Ας προσπαθήσουμε να εφαρμόσουμε τα παραπάνω στο πρόβλημα εύρεσης του μεγαλύτερου αθροίσματος ακεραίων που βρίσκονται σε συνεχόμενες θέσεις σε ένα πίνακα ο οποίος «φιλοξενεί» ακέραιους στις θέσεις 1 ως n . Το πρόβλημα αυτό παρουσιάστηκε στην Ενότητα 8.3.1 μαζί με έναν αλγόριθμο επίλυσης (Αλγόριθμος 58). Ο αλγόριθμος αυτός είναι ένας αλγόριθμος ΔΠ αφού υλοποιεί την παραπάνω μεθοδολογία. Τα στάδια 1-5 εξειδικεύονται, στην περίπτωση αυτή, ως εξής.

1. Ορισμός υποπροβλημάτων

1. (Ορισμός υποπροβλημάτων) Ορίζουμε ως $S(:j)$ το υποπροβλημα εύρεσης του μεγαλύτερου αθροίσματος συνεχόμενων ακεραίων της υποσειράς $A[1], \dots, A[j]$, όπου $j \in \{1, \dots, n\}$. Δηλαδή ο ορίζοντας επίλυσης περιορίζεται για τιμές μικρότερες-ίσες του j . Ο συμβολισμός $\langle\langle : j \rangle\rangle$ υποδηλώνει την προθεματική περιγραφή του υποπροβλήματος: η επίλυση αφορά το υποσύνολο των δεδομένων που περιλαμβάνει τα στοιχεία από την πρώτη μέχρι τη θέση j του πίνακα.

2. Καθορισμός επιλογών

2. (Καθορισμός επιλογών) Οι επιλογές που έχουμε για 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 αντιστοιχεί σε λύση για το υποπρόβλημα $S(: j)$ με τιμή το στοιχείο $A[j]$.

3. Ορισμός αναδρομικής σχέσης

3. (Δημιουργία αναδρομικής σχέσης) Συμβολίζουμε με s_j τη λύση του υποπροβλήματος $S(: j)$. Η αναδρομική σχέση που δίνει την τιμή της s_j , προκύπτει άμεσα από τις επιλογές που περιγράφηκαν στο προηγούμενο στάδιο, ήτοι

$$s_j = \begin{cases} \max\{A[j], A[j] + s_{j-1}\}, & j = 2, \dots, n, \\ A[1], & j = 1. \end{cases} \quad (14.1)$$

4. Τοπολογική ταξινόμηση υποπροβλημάτων

4. (Σειρά επίλυσης υποπροβλημάτων) Η απεικόνιση των σχέσεων των υποπροβλημάτων δίνεται από το γράφημα με κορυφές τα $S(: j)$ 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 και αυτή είναι η σειρά με την οποία επιλύονται τα υποπροβλήματα.

5. Επίλυση αρχικού προβλήματος

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

$$\max\{s_j : j = 1, \dots, n\} \quad (14.2)$$

Πολυπλοκότητα

την οποία και υπολογίζει ο Αλγόριθμος 58. Παρατηρούμε ότι η πολυπλοκότητα του προκύπτει αν στον αριθμό των ΣΥΒ που εκτελούνται από την (14.2) προστεθεί το γινόμενο του αριθμού των υποπροβλημάτων επί τον αριθμό των ΣΥΒ που εκτελούνται για να επιλυθεί το κάθε υποπρόβλημα. Συγκεκριμένα εκτελούνται $\Theta(n)$ ΣΥΒ για ον υπολογισμό της (14.2), ενώ για την εύρεση λύσης σε κάθε υποπρόβλημα εκτελείται σταθερός αριθμός ΣΥΒ και υπάρχουν n υποπροβλήματα. Επομένως ο Αλγόριθμος 58 έχει πολυπλοκότητα $\Theta(n)$.

Δομή απομνημόνευσης

Για να μπορέσουμε να βρούμε ποια είναι η υπασειρά των συνεχόμενων αριθμών η οποία μεγιστοποιεί το άθροισμα των στοιχείων της θα πρέπει να αποθηκεύσουμε σε κάποια δομή την επιλογή που γίνεται σε κάθε βήμα. Για το σκοπό αυτό χρησιμοποιούμε το διάνυσμα p .² Το στοιχείο $p_j (j = 1, \dots, n)$ έχει τιμή τη θέση του πρώτου στοιχείου της σειράς που έχει άθροισμα s_j - το τελευταίο στοιχείο της σειράς δίνεται από την τιμή του δείκτη j . Οι τιμές του διανύσματος δίνονται, για $j = 1, \dots, n$, από τον τύπο:

$$p_j = \begin{cases} p_{j-1}, & \text{αν } s_j = A[j] + s_{j-1}, \\ j, & \text{αν } s_j = A[j]. \end{cases} \quad (14.3)$$

Η χρήση μίας δομής για την ιχνηλάτηση της λύσης του αρχικού προβλήματος - πέρα από την τιμή της - αποτελεί βασικό χαρακτηριστικό σε κάθε αλγορίθμικό σχήμα ΔΠ. Στη δομή αυτή κωδικοποιείται η πληροφορία που αφορά τη βέλτιστη επιλογή που γίνεται για τη λύση του κάθε υποπροβλήματος.³

Παράδειγμα

Παράδειγμα 76. Θέλουμε να λύσουμε το πρόβλημα του μεγαλύτερου αθροίσματος συνεχόμενων ακεραίων του πίνακα

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

Βάσει των (14.1) και (14.3), για $j = 1, \dots, 9$, εκτελούμε τους υπολογισμούς:

$$\begin{aligned} s_1 &= A[1] = -9, & p_1 &= 1, \\ s_2 &= \max\{A[2], A[2] + s_1\} = \{1, 1 - 9\} = 1, & p_2 &= 2, \\ s_3 &= \max\{A[3], A[3] + s_2\} = \{-5, 1 - 5\} = -4, & p_3 &= p_2 = 2, \\ s_4 &= \max\{A[4], A[4] + s_3\} = \{4, 4 - 4\} = 4, & p_4 &= 4, \\ s_5 &= \max\{A[5], A[5] + s_4\} = \{3, 3 + 4\} = 7, & p_5 &= p_4 = 4, \\ s_6 &= \max\{A[6], A[6] + s_5\} = \{-6, -6 + 7\} = 1, & p_6 &= p_5 = 4, \\ s_7 &= \max\{A[7], A[7] + s_6\} = \{7, 1 + 7\} = 8, & p_7 &= p_6 = 4, \\ s_8 &= \max\{A[8], A[8] + s_7\} = \{8, 8 + 8\} = 16, & p_8 &= p_7 = 4, \\ s_9 &= \max\{A[9], A[9] + s_8\} = \{-2, -2 + 16\} = 14, & p_9 &= p_8 = 4. \end{aligned}$$

Η λύση του προβλήματος δίνεται από την (14.2), 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,

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

Max sum subsequence
Παράδειγμα 34

Chapter 9c

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

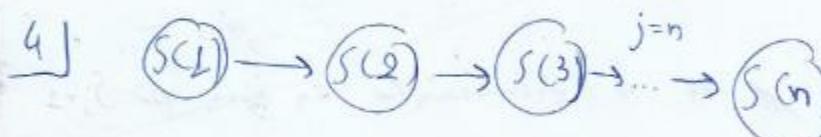
1] $S_{\cdot j}$ → το υπορύθμημα είρεσης των μεγαλύτερων αξεσίδρυτων που έκανε από τη δύση 1 και καταλήγει στην δύση j

Προθετική
Παραγραφή

2] S_j είτε να χρησιμοποιείται για S_{j-1} του S_{j-1}
είτε να ΜΗΝ χρησιμοποιείται για S_{j-1} του S_{j-1}

- 12 -

$$3] S_j = \begin{cases} \max\{S_{j-1} + A[j], \\ A[j]\}, & \forall j=2, \dots, n \\ A[1], & j=1 \end{cases}$$



5] Τυπώ $\max S_j \quad \forall j \in \{1, \dots, n\}$

$$S_1 = A[1] = -9, \quad p_1 = 1$$

$$p_j = \begin{cases} p_{j-1}, & S_j = A[j] \\ j, & S_j = A[j] \end{cases}$$

$$S_2 = \max\{S_1 + A[2], A[2]\} = \max\{-9 + 1, 1\} = 1, \quad p_2 = 2$$

! Για να κάνω trace το pattern σαιρών parent \circledast
array για να θέω ότι να το πάρω. Βρίσκεται την δύση των υπορύθμητων

$$S_3 = \max\{S_2 + A[3], A[3]\} = \max\{1 + (-5), -5\} = -4, \quad p_3 = 2$$

$$S_4 = \max\{S_3 + A[4], A[4]\} = \max\{-4 + 4, 4\} = 4, \quad p_4 = 4$$

$$S_5 = \max\{S_4 + A[5], A[5]\} = \max\{4 + 3, 3\} = 7, \quad p_5 = 4$$

$$S_6 = \max\{S_5 + A[6], A[6]\} = \max\{7 + (-6), -6\} = 1, \quad p_6 = 4$$

$$S_7 = \max\{S_6 + A[7], A[7]\} = \max\{1 + 7, 7\} = 9, \quad p_7 = 4$$

$$S_8 = \max\{S_7 + A[8], A[8]\} = \max\{9 + 8, 8\} = 17, \quad p_8 = 4$$

$$S_9 = \max\{S_8 + A[9], A[9]\} = \max\{17 + (-2), -2\} = 15, \quad p_9 = 4$$

9d. Το πρόβλημα εύρεσης της υπακολονθίας με το μέγιστο άθροισμα (Επιθεματική Περιγραφή)

Σημειώσεις

1 $S(j:)$ → το υπορίθμημα
Επιθεματική Περιγραφή
εύρεσης των μεγαλύτερων αθροίσματων που θεωρούνται από τη σειρά j και καταλήγουν στη σειρά n

Chapter 9d

2 S_j είτε να χρησιμοποιείται τη λύση των S_{j+1}
είτε να ΜΗΝ χρησιμοποιείται τη λύση των S_{j+2}

$$S_j = \begin{cases} \max\{A[j], A[j] + S_{j+1}\}, & \forall j = n-1, \dots, 1 \\ A[n], & j = n \end{cases}$$

4 $S(n) \rightarrow \dots \rightarrow S(3) \rightarrow S(2) \rightarrow S(1)$

$$\max \{S_j : j = 1, \dots, n\}$$

$$P = \begin{cases} P_{j+1}, & \text{av } S_j = A[j] + S_{j+1} \\ j, & \text{av } S_j = A[j] \end{cases}$$

~~A = [9, 1, -5, 4, 3, -6, F, 8, -2]~~

$$A = [-9, 1, -5, 4, 3, -6, F, 8, -2]$$

$$S_9 = A[9] = -2, P_9 = 9$$

$$S_8 = \max\{A[8], A[8] + S_9\} = \max\{8, 8 - 2\} = 8, P_8 = 8$$

$$S_7 = \max\{A[7], A[7] + S_8\} = \max\{F, F + 8\} = 15, P_7 = P_8$$

$$S_6 = \max\{A[6], A[6] + S_7\} = \max\{-6, -6 + 15\} = 9, P_6 = P_7$$

$$S_5 = \max\{A[5], A[5] + S_6\} = \max\{3, 3 + 9\} = 12, P_5 = P_6$$

$$\boxed{S_4 = \max\{A[4], A[4] + S_5\} = \max\{4, 4 + 12\} = 16, P_4 = P_5}$$

$$S_3 = \max\{A[3], A[3] + S_4\} = \max\{-5, -5 + 16\} = 9, P_3 = P_4$$

$$S_2 = \max\{A[2], A[2] + S_3\} = \max\{1, 1 + 9\} = 10, P_2 = P_3$$

$$S_1 = \max\{A[1], A[1] + S_2\} = \max\{-9, -9 + 10\} = 1, P_1 = P_2$$

$$S(:j) \quad \max\{S_j : j = 1, \dots, 9\} = \max\{-9, 1, -4, 4, F, 1, 8, 16, 14\} = \underline{16}$$

P₈ = 4

$$A_{\text{sum}} = A[4] + A[5] + A[6] + A[F] + A[8] = 4 + 3 + (-6) \\ + F + 8 = 16$$

$$S(j) = \max \{ S_j : j = 1, \dots, 1\} = \max \{-2, 8, 15, 9, 12, 16, 9, 10, 1\} = 16$$

$$P_3 = 4$$

$$\text{Aριθμοί } A[4] + A[5] + A[6] + A[7] + A[8] = 4 + 3 + (-6) + 7 + 8 = 16$$

9e. Δυναμικός Προγραμματισμός και Βέλτιστη Υποδομή

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 85

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 μικρότερων υποπροβλήματων. Για παράδειγμα, στο στιγμιότυπο του προβλήματος εύρεσης του μεγαλύτερου αθροίσματος συνεχόμενων ακεραίων, 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 $S(: 6)$ κοκ. Αντίστοιχα, όπως είδαμε και στην Ενότητα 13.1.1, το συντομότερο μονοπάτι που συνδέει δύο κορυφές ενός γραφήματος περιέχει τα συντομότερα μονοπάτια για κάθε ζευγάρι κορυφών που διασχίζει.

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

Chapter 10

10a. Συντομότερο μονοπάτι σε ΚΑΓ

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 85-88

1. Ορισμός υποπροβλημάτων

Μπορούμε εύκολα να επιλύσουμε το πρόβλημα αυτό με ΔΠ αφού η κάθε κορυφή στο γράφημα ορίζει ένα υποπρόβλημα. Συγκεκριμένα, για κάθε κορυφή u , ορίζουμε ως $D(: u)$ το υποπρόβλημα εύρεσης του μήκους του συντομότερου μονοπατιού που εκκινεί από την κορυφή s και καταλήγει στη u (Στάδιο 1). Θα συμβολίζουμε με d_u τη λύση του προβλήματος $D(: u)$.

2. Καθορισμός επιλογών

Κάθε κορυφή v για την οποία υπάρχει η κατευθυνόμενη ακμή (v, u) αποτελεί μία επιλογή από την οπαία μπορούμε να φτάσουμε στη u . Φτάνουμε με ένα μονοπάτι από την s στη v και στη συνέχεια διασχίζουμε την ακμή (v, u) (Στάδιο 2).

3. Ορισμός αναδρομικής σχέσης

Άρα μπορούμε να πάρουμε την καλύτερη επιλογή που μας οδηγεί στη κορυφή u . Δηλαδή επιλέγεται η κορυφή v η οποία ελαχιστοποιεί το άθροισμα του μήκους του μονοπατιού από το s στο v συν το συντελεστή $w(v, u)$. Η σκέψη αυτή μας οδηγεί στη σχέση που συνδέει τη λύση του υποπροβλήματος $D(: u)$ με τις λύσεις των υποπροβλημάτων $D(: v)$, για κάθε $v \in N^-(u)$ (Στάδιο 3). Η σχέση αυτή διατυπώνεται μαθηματικά ως

$$d_u = \min\{d_v + w(v, u) : v \in N^-(u)\}. \quad (14.4)$$

4. Τοπολογική ταξινόμηση υποπροβλημάτων

Η (14.4) υπαγορεύει και τη σειρά με την οποία πρέπει να επιλύονται τα υποπροβλήματα: κάθε κορυφή αντιπροσωπεύει ένα υποπρόβλημα και επομένως η τοπολογική ταξινόμηση των κορυφών του G υποδεικνύει τη ζητούμενη σειρά επίλυσης (Στάδιο 4). Επειδή το G είναι ΚΑΓ πάντα υπάρχει μία διάταξη των κορυφών σε τοπολογικά ταξινομημένη σειρά.

5. Επίλυση αρχικού προβλήματος

Η λύση του αρχικού προβλήματος (Στάδιο 5) θα προκύψει όταν έχουν υπολογιστεί τα ελάχιστα μήκη d_u για κάθε κορυφή u .

Δομή απομνημόνευσης

Για να μπορέσουμε να ιχνηλατήσουμε το κάθε μονοπάτι θα πρέπει μαζί με τον υπολογισμό που γίνεται από την (14.4) να κρατάμε στο διάνυσμα p την κορυφή που οδηγεί στη βέλτιστη επιλογή. Δηλαδή,

$$p_u = \operatorname{argmin}\{d_v + w(v, u) : v \in N^-(u)\}. \quad (14.5)$$

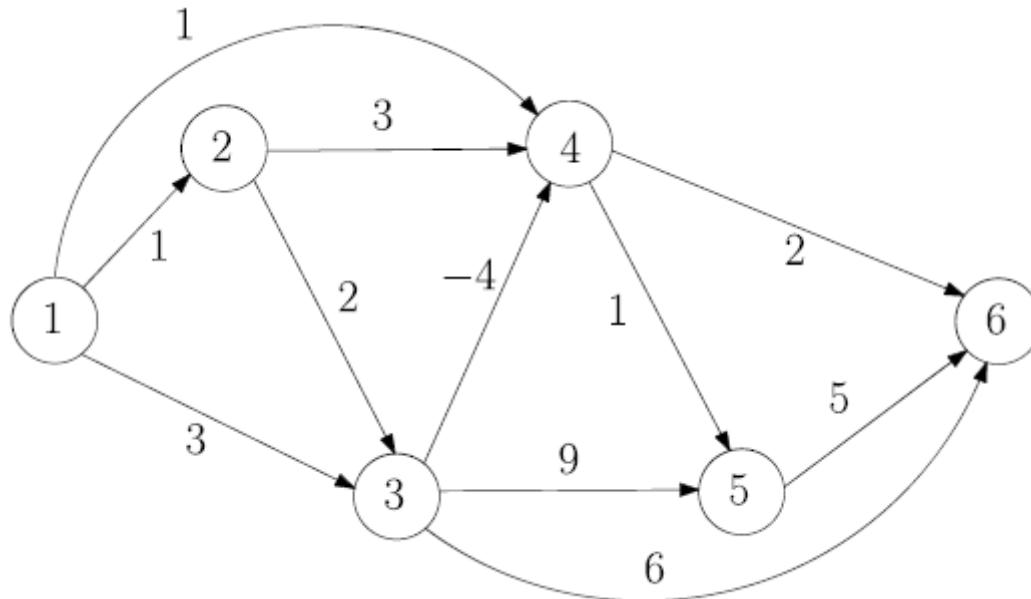
Πολυπλοκότητα

Η παραπάνω ανάλυση οδηγεί σε έναν αλγόριθμο πολυπλοκότητας $O(n^2)$. Είναι εύκολο να δούμε προς προκύπτει αυτό: Θεωρούμε ότι οι κορυφές ταυτοποιούνται από την αριθμητική σειρά $1, 2, \dots$ σύμφωνα με την τοπολογική τους ταξινόμηση όπου $s = 1$. Συνεπώς $N^-(u) \subseteq \{1, \dots, u - 1\}$ και $|N^-(u)| \leq u - 1$. Όμως $O(|N^-(u)|)$ είναι ο αριθμός των ΣΥΒ που εκτελούνται για την επίλυση του στιγμιότυπου $D(: u)$. Αθροίζοντας για $u = 1, \dots, n$ προκύπτει ότι ο αριθμός των ΣΥΒ είναι $O(n^2)$.

Παρατήρηση

Τέλος θα πρέπει να παρατηρήσουμε ότι η υπόθεση ότι $s = 1$ γίνεται χωρίς βλάβη της γενικότητας. Μπορούμε να θεωρήσουμε οποιαδήποτε κορυφή ως s και να υπολογίσουμε το συντομότερο μονοπάτι προς κάθε κορυφή $u > s$. Για κάθε άλλη κορυφή $u < s$ δεν υπάρχει μονοπάτι από την s προς αυτή εφόσον το γράφημα είναι ΚΑΓ.

Σχήμα 14.2 Άκυκλο κατευθυνόμενο γράφημα



Παράδειγμα

Παράδειγμα 77. Στο γράφημα του Σχήματος 14.2 θέλουμε να υπολογίσουμε τα ελάχιστα μονοπάτια από την κορυφή 1 προς όλες τις υπόλοιπες κορυφές. Παρατηρούμε ότι πρόκειται περί ενός ΚΑΓ όπου οι ετικέτες των κορυφών έχουν αποδοθεί σύμφωνα με την τοπολογική ταξινόμηση. Έχουμε,

$$\begin{aligned}N^-(1) &= \emptyset, \\N^-(2) &= \{1\}, \\N^-(3) &= \{1, 2\}, \\N^-(4) &= \{1, 2, 3\}, \\N^-(5) &= \{3, 4\}, \\N^-(6) &= \{3, 4, 5\}.\end{aligned}$$

Από τις (14.4), (14.5),

$$d_1 = 0,$$

$$d_2 = \min\{d_v + w(v, 2) : v \in N^-(2)\}$$

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

$$d_3 = \min\{d_v + w(v, 3) : v \in N^-(3)\}$$

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

$$= \min\{0 + 3, 1 + 2\} = 3, \quad p_3 = 1, \text{ ή } p_3 = 2,$$

$$d_4 = \min\{d_v + w(v, 4) : v \in N^-(4)\}$$

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

$$= \min\{0 + 1, 1 + 3, 3 + (-4)\} = -1, \quad p_4 = 3,$$

$$d_5 = \min\{d_v + w(v, 5) : v \in N^-(5)\}$$

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

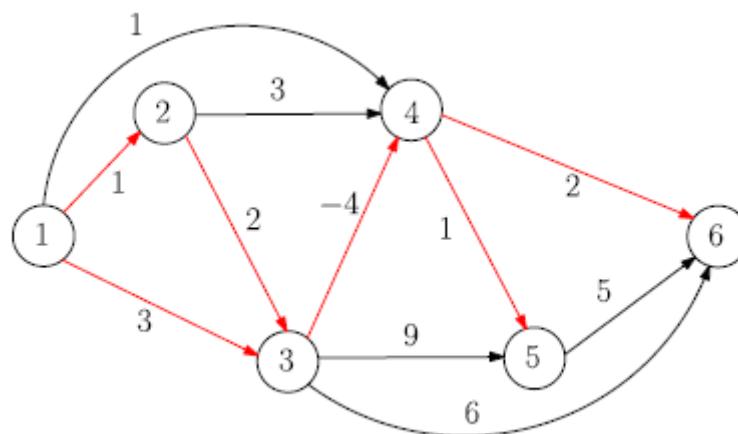
$$= \min\{3 + 9, -1 + 1\} = 0, \quad p_5 = 4,$$

$$d_6 = \min\{d_v + w(v, 6) : v \in N^-(6)\}$$

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

$$= \min\{3 + 6, -1 + 2, 0 + 5\} = 1, \quad p_6 = 4.$$

Σχήμα 14.3 Συντομότερα μονοπάτια από την κορυφή 1



Παρατήρηση

Τα συντομότερα μονοπάτια όπως ιχνηλατούνται μέσω του διανύσματος p απεικονίζονται στο Σχήμα 14.3 (κόκκινες ακμές). Όπως προκύπτει και από το διάνυσμα p υπάρχον δύο συντομότερα μονοπάτια από την κορυφή 1 στην 3 : το ένα αποτελείται μόνο από την ακμή $(1, 3)$ και το άλλο από τις ακμές $(1, 2)$ και $(2, 3)$. Αμφότερα έχουν μήκος 3.

(ΑΠ)

23/11/2023

Chapter 10 a

5 στάδια Δ.Π.

1] Ορισμός Υπορρθλημάτων

2] Κανονισμός επιλογών

3] Ορισμός Ανιδροτυπίας σχετικών

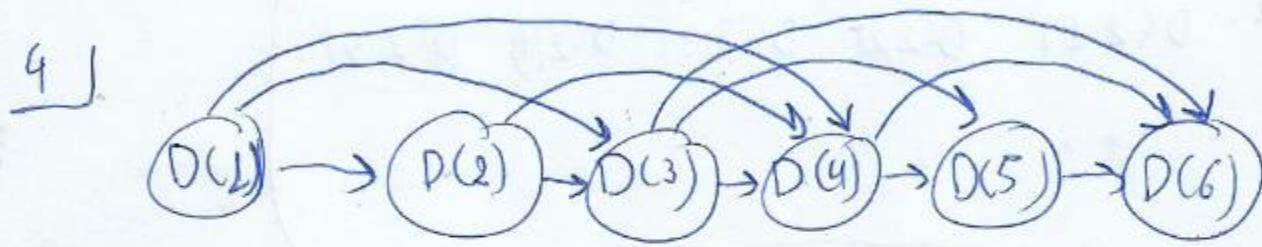
4] Βασική Τατιώνια Υπορρθλημάτων

5] Επίλογοι αρχών προβλημάτων

Συντοπότερο μοντέλο σε KAF

1] $D(v)$ → Εκφράζει την αρχηγική κορυφή S
Πρώτης κατατάξη στην κορυφή v 2] $|N^-(v)| \rightarrow$ Οι επιλογές είναι $\# N^-(v)$

3] $d_v = \begin{cases} \min \{ d_u + w(u, v) : u \in N^-(v), \forall v \in V \setminus \{s\} \} \\ 0 \end{cases}, v = s$



5] d_v

$$\forall v \in V(G)$$

(H) στήλευ σελ. 9 documents \rightarrow Δικτυούς Πραγματοποίησης

10b. Σχέση Bellman-Ford με Δ.Π.

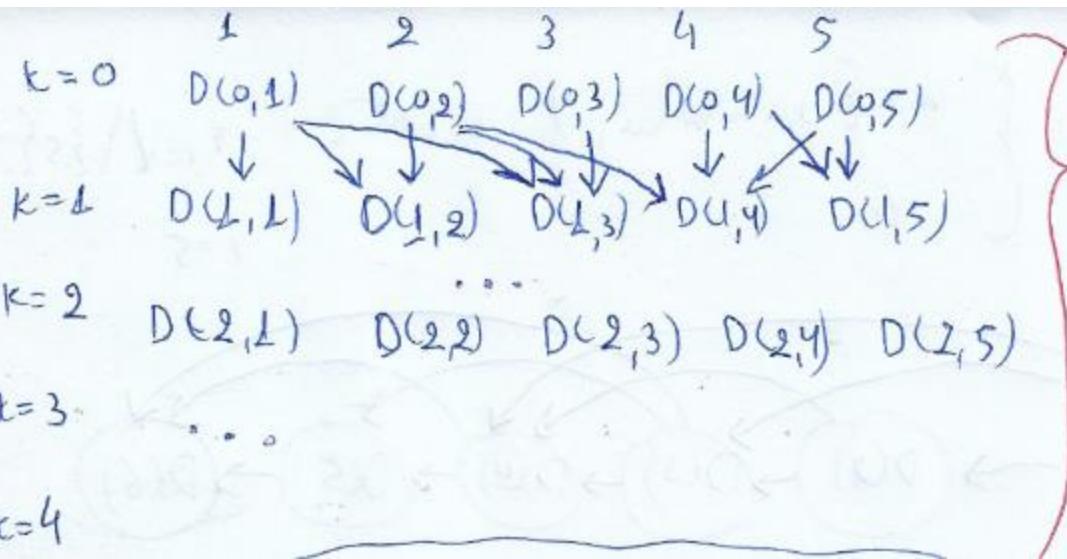
Σημειώσεις

(Σχέση Bellman-Ford - Δ.Π.)

Chapter 10 b

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

1] $D(:, v, :k) \rightarrow$ το νωτίν v και όλες
τις πολύ κ αιρέτες



10c. Η μεγαλύτερη αύξουσα υπακολουθία (Προθεματική Περιγραφή)

Διαφάνειες

Slides_and_Exercises→09_DynamicExer.pdf σελ. 1-3

Εκφώνηση

Άσκηση 1 Δίνεται μία σειρά αριθμών a_1, \dots, a_n . Μια αύξουσα υπακολουθία αποτελείται από τους αριθμούς $a_{i_1}, a_{i_2}, \dots, a_{i_k}$ που ανήκουν στη σειρά και έχουν την ιδιότητα $a_{i_1} < a_{i_2} < \dots < a_{i_k}$ με $i_1 < i_2 < \dots < i_k$. Το πρόβλημα της μεγαλύτερης αύξουσας υπακολουθίας ζητά να βρεθεί η αύξουσα υπακολουθία που μεγιστοποιεί το k (δηλαδή η αύξουσα υπακολουθία με τους περισσότερους όρους).

1. Ορισμός υποπροβλημάτων

Ορίζουμε $L(j)$ το μήκος του μεγαλύτερου μονοπατιού που καταλήγει στον κόμβο j .

2. Καθορισμός επιλογών

Λύση Θεωρούμε ένα γράφημα διάταξης G με n κόμβους: ο κόμβος i αντιστοιχεί στο στοιχείο a_i . Φέρουμε προσανατολισμένη ακμή από τον κόμβο i στον κόμβο j αν $i < j$ και $a_i < a_j$.

3. Ορισμός αναδρομικής σχέσης

Ισχύει η αναδρομική σχέση

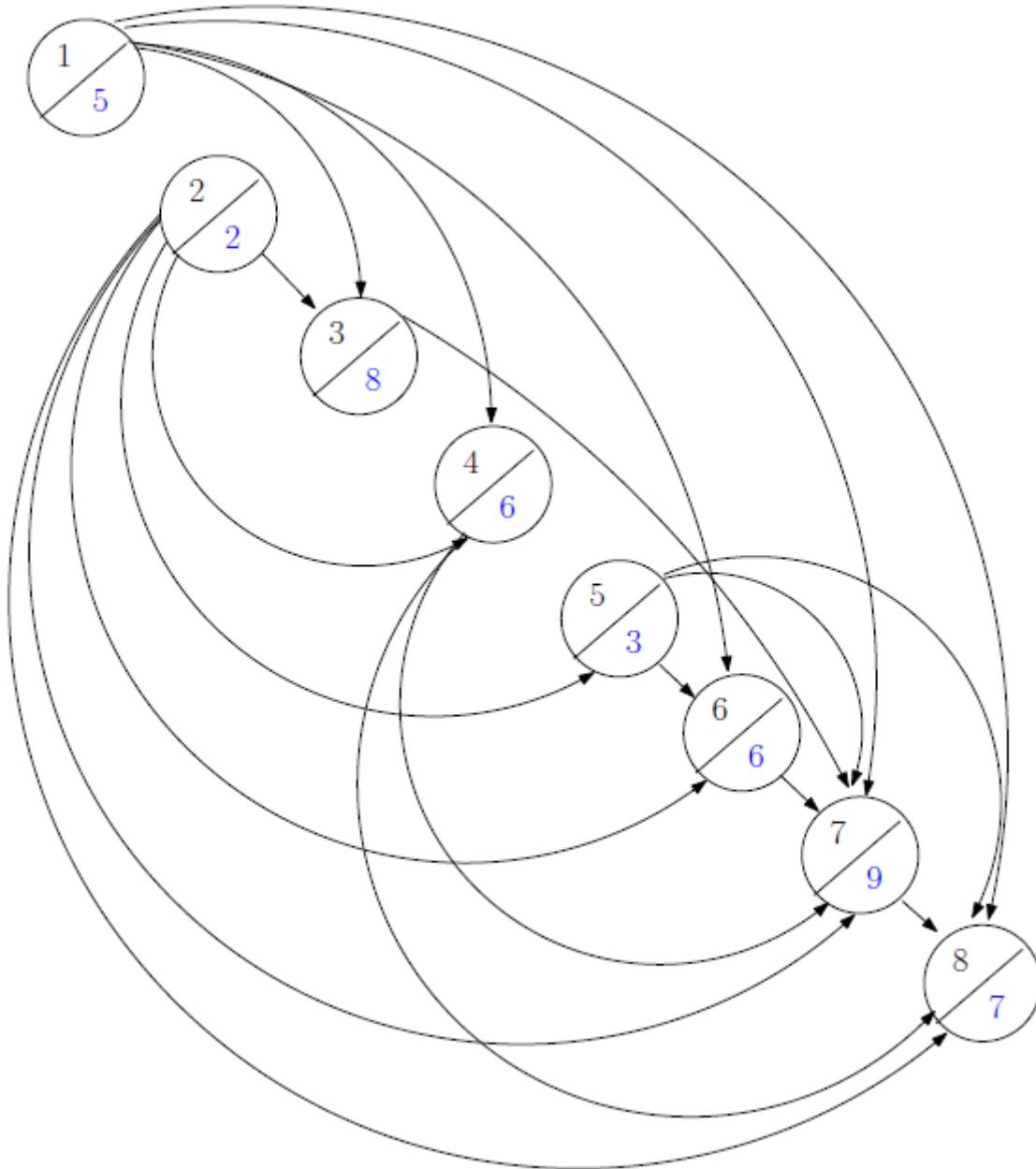
$$L(j) = \begin{cases} 1 + \max_i\{L(i)\}, & i < j, a_i < a_j, \\ 1, & \text{διαφορετικά} \end{cases} \quad (1)$$

Σε όρους γραφήματος η σχέση (1) γράφεται ισοδύναμα ως

$$L(j) = 1 + \max\{L(i) : i \in N^-(j)\}, \quad (2)$$

όπου $N^-(j) : \{i : i \in V(G), (i,j) \in E(G)\}$

4. Τοπολογική ταξινόμιση υποπροβλημάτων



5. Επίλυση αρχικού προβλήματος

Προφανώς το μεγαλύτερο σε αριθμό κορυφών (κατευθυνόμενο) μονοπάτι στο γράφημα G αντιστοιχεί στην μεγαλύτερη αύξουσα υπακολουθία.

Ιδέα

Ο Αλγόριθμος 1 απεικονίζει την παραπάνω διαδικασία: δέχεται σαν είσοδο τη σειρά των αριθμών στις n πρώτες θέσεις του πίνακα A και παράγει τόσο τον πίνακα L όσο και τον πίνακα $prev$ με τη βοήθεια του οποίου μπορούμε να εντοπίσουμε την μεγαλύτερη αύξουσα υπακολουθία..

Αλγόριθμος

Algorithm 1 Μεγαλύτερη αύξουσα υπακολουθία

```
1: void maxsubseq(int n, int A[], int previous[])
2: for j ← 1; j ≤ n; j + + do
3:   L[j] ← 0;
4:   previous[j] ← 999;
5:   for i ← 1; i ≤ j – 1; i + + do
6:     if A[i] < A[j] and L(j) < L(i) then
7:       L[j] ← L[i];
8:       previous[j] ← i;
9:     end if
10:   end for
11:   L[j] ← L[j] + 1;
12: end for
13: length ← 0;
14: last_node ← 0;
15: for j ← 1; j ≤ n; j + + do
16:   if L[j] > length then
17:     length ← L[j];
18:     last_node ← j;
19:   end if
20: end for
21: while last_node < 999 do
22:   print(last_node);
23:   last_node ← previous[last_node];
24: end while
25: print(length);
```

Πολυπλοκότητα

Είναι εύκολο να παρατηρήσουμε ότι ο Αλγόριθμος 1 είναι τάξης $\Theta(n^2)$. Ειδικότερα στις γραμμές 2-12 γίνονται δύο συγκρίσεις (γραμμή 6) για κάθε ζευγάρι τιμών i, j . Άρα ο συνολικός αριθμός των συγκρίσεων σε αυτές τις γραμμές είναι

$$T(n) = \sum_{j=1}^n 2(j-1) = 2[\sum_{j=1}^n j - n] = n(n-1).$$

Παράδειγμα

Παράδειγμα 1 Για τη σειρά

5 2 8 6 3 6 9 7

Στο παραπάνω παράδειγμα,

$$L(1) = 1,$$

$$L(2) = 1,$$

$$L(3) = 1 + \max\{L(1), L(2)\} = 2$$

$$L(4) = 1 + \max\{L(1), L(2)\} = 1 + \max\{1, 1\} = 2$$

$$L(5) = 1 + \max\{L(2)\} = 1 + \max\{1\} = 2$$

$$L(6) = 1 + \max\{L(5), L(2), L(1)\} = 1 + \max\{2, 1, 1\} = 3$$

$$L(7) = 1 + \max\{L(6), L(5), L(4), L(3), L(2), L(1)\} = 1 + \max\{3, 2, 2, 2, 1, 1\} = 4$$

$$L(8) = 1 + \max\{L(6), L(5), L(4), L(2), L(1)\} = 1 + \max\{3, 2, 2, 1, 1\} = 4$$

Επομένως υπάρχουν δύο μεγαλύτερες αύξουσες υπακολουθίες, καθεμία με τέσσερα στοιχεία, ήτοι

$$\begin{matrix} 2 & 3 & 6 & 7 \\ 2 & 3 & 6 & 9 \end{matrix}$$

Σημειώσεις

H μεγαλύτερη αύξουση υπακολουθία

$$a = \{ 5 \ 2 \ 3 \ 6 \ 3 \ 6 \ 9 \ F \} \quad [\text{Chapter 10c}]$$

1] $L(:j)$ → Το μονατικό καταλίγει στον κύριο j για να υπολογίζει την πίνακα l_j των υποκομβήματος. Οι πίνακες να ορίσων.

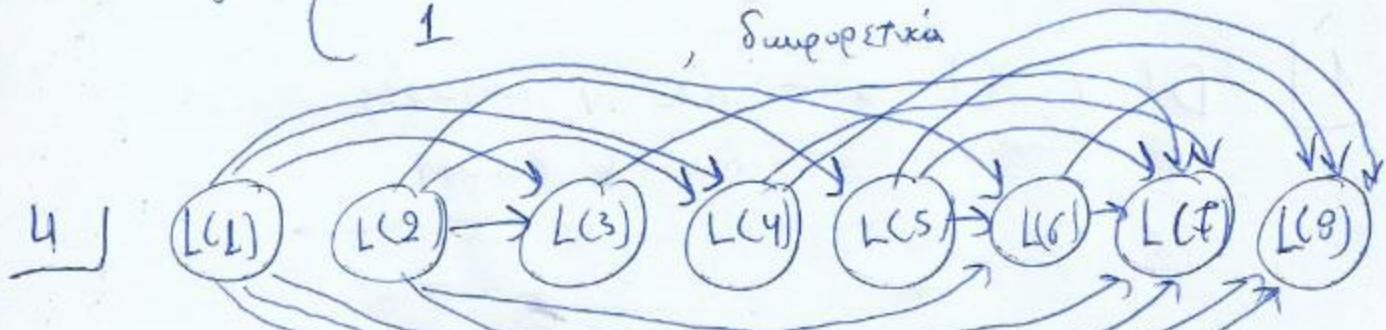
Προεργατική
Περιήγηση

$$L(:i) \quad i < j, \quad a_i < a_j$$

Το μήκος των μεγαλύτερων μονατικών πως καταλίγει στον κύριο j και i αντίστοιχα

2] $|N^-(j)| \leq j - 1$ (Το μεγαλύτερο σε αριθμό κορυφών μονατικών)

3] $l_j = \begin{cases} 1 + \max\{l_i : i < j, a_i < a_j\} \\ 1 \end{cases}$, διαφορετικά



4] $\max l_j, \quad 1 \leq j \leq n$ σελ. 3
-18- ... Slides_and_Exercises → 09_DynamicExer

Σημειώσεις

1] $L(j:)$ → Το μέγκος των μοναδών
 Επιθεματική προς αύξουσα υπακολούθια πως Fekete's
 Περιγραφή αν τον κύριο j

$L(i:)$ → Το μέγκος των μοναδών της αύξουσας
 υπακολούθιας πως Fekete's αν τον κύριο i

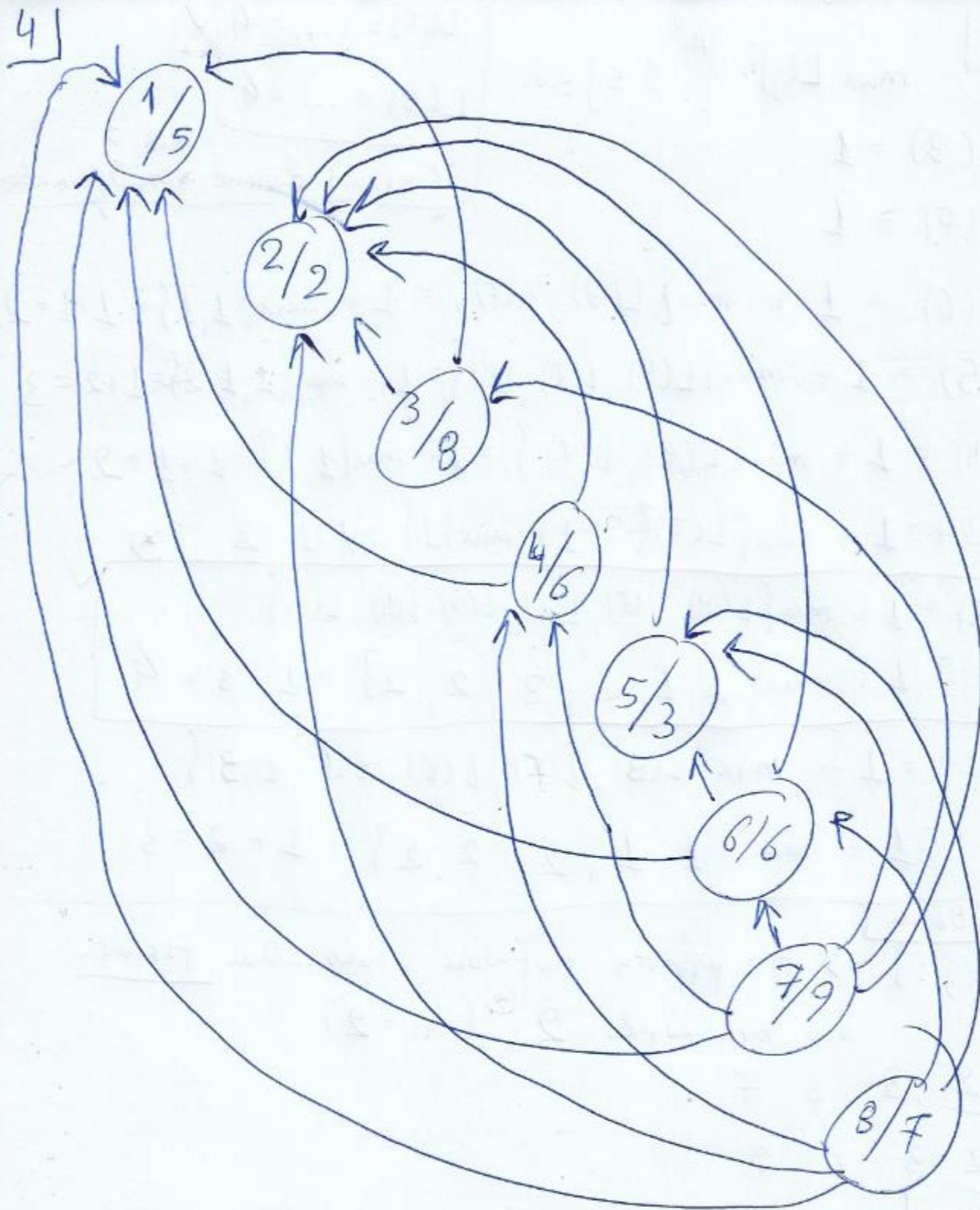
Αν $i > j$ KAI $a_i > a_j$, τότε φέρνεις προβαντοληγράφη
 ακριβή αν τον κύριο j στον i

2] $|N^+(j)| \leq j-1$ (Το μεγαλύτερος αριθμός
 κορυφών μοναδών)

$$L(j) = \begin{cases} L + \max\{L(i) : i > j, a_i > a_j\} \\ L \end{cases}, \quad \text{διαφορετικά}$$

$$L(j) = L + \max\{L(i) : i \in N^+(j)\},$$

οù $N^+(j) : \{i : i \in V(G), (j, i) \in E(G)\}$



5]

$$\max L(j) \quad , \quad 1 \leq j \leq n$$

$$L(8) = 1$$

$$L(F) = 1$$

$$L(6) = 1 + \max\{L(8), L(F)\} = 1 + \max\{1, 1\} = 1+1=2$$

$$L(5) = 1 + \max\{L(8), L(F), L(6)\} = 1 + \max\{1, 1, 2\} = 1+2=3$$

$$L(4) = 1 + \max\{L(8), L(F)\} = 1 + \max\{1, 1\} = 1+1=2$$

$$L(3) = 1 + \max\{L(F)\} = 1 + \max\{1\} = 1+1=2$$

$$\begin{aligned} L(2) &= 1 + \max\{L(8), L(F), L(6), L(5), L(4), L(3)\} \\ &= 1 + \max\{1, 1, 2, 3, 2, 2\} = 1+3=4 \end{aligned}$$

$$\begin{aligned} L(1) &= 1 + \max\{L(8), L(F), L(6), L(4), L(3)\} \\ &= 1 + \max\{1, 1, 2, 2, 2\} = 1+2=3 \end{aligned}$$

Επίθετα |

$L(j:)$ => Η πρώτη αύφορη υποκατάσταση Έξιμη
στον κώνο 2 \oplus ($a_i = 2$)

2 3 6 F

2 3 6 9

Λύπηθεται | $L(:j)$ => Η πρώτη αύφορη υποκατάσταση καταλήγει
στον κώνο F ($a_i = 9$) και στην κώνο Θ
($a_i = F$) \oplus

2 3 6 F

2 3 8 9

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 88-93

Πρόβλημα

Μία παράγραφος κειμένου αποτελείται από n λέξεις. Υποθέτουμε ότι κάθε χαρακτήρας - γράμμα, σύμβολο, διάστημα - που εμφανίζεται στο κείμενο πιάνει τον ίδιο χώρο. Δεδομένου του αριθμού των χαρακτήρων που περιέχει μία γραμμή, έστω m , επιθυμούμε τη διάσπαση της παραγράφου σε γραμμές ώστε το κείμενο να εμφανίζεται όσο το δυνατόν πιο «ομοιόμορφο». Αν θεωρήσουμε ότι η παράγραφος θα πρέπει να εμφανίζεται με αριστερή στοίχιση, τότε ο όρος «ομοιόμορφο» αναφέρεται στην ελαχιστοποίηση των κενών διαστημάτων στο τέλος κάθε γραμμής.

Θεωρώντας ότι η παράγραφος αποτελείται από τις λέξεις w_1, w_2, \dots, w_n , αν μία γραμμή αποτελείται από τις λέξεις $w_i, w_{i+1}, \dots, w_{j-1}, w_j$, για $1 \leq i \leq j \leq n$, μπορούμε να συσχετίσουμε με τη γραμμή αυτή έναν αριθμό, έστω $b(i, j)$, ο οποίος εκφράζει το πόσο κακή (μη-ομοιόμορφη) είναι η γραμμή αυτή. Θα ονομάζουμε το $b(i, j)$ ως συντελεστή ανομοιομορφίας της γραμμής. Συνήθως ο συντελεστής αυτός είναι συνάρτηση του πλήθους των κενών διαστημάτων που υπάρχουν στο τέλος της γραμμής. Μία καλή στοίχιση θα χωρίζει την παραγραφο σε γραμμές κατά τρόπο ώστε να ελαχιστοποιείται το άθροισμα των συντελεστών ανομοιομορφίας των γραμμών (ανομοιομορφία της παραγράφου). Το ζητούμενο είναι το πως θα μπορέσει να επιτευχθεί μία καλή στοίχιση. Δηλαδή ζητείται ο χωρισμός του κειμένου σε γραμμές - καθορισμός των λέξεων που θα περιέχει η κάθε γραμμή - ώστε να ελαχιστοποιείται η ανομοιομορφία της παραγράφου.

Ιδέα

Για να μπορέσουμε να καταστρώσουμε ένα σχήμα ΔΠ που να επιλύει το παραπάνω πρόβλημα, θεωρούμε γνωστούς τους συντελεστές $b(i, j)$. Πράγματι τους συντελεστές αυτούς μπορούμε να τους υπολογίσουμε εκ' των προτέρων. Αν για παράδειγμα μας ενδιαφέρουν τα κενά στο τέλος κάθε γραμμής, τότε θα μπορούσαμε να ορίσουμε την ανομοιομορφία της γραμμής με πρώτη λέξη την w_i και τελευταία την w_j , με $1 \leq i \leq j \leq n$, ως

$$b(i, j) = \begin{cases} (m - s_{ij} - \sum_{t=i}^j |w_t|)^2, & \text{αν } m - s_{ij} - \sum_{t=i}^j |w_t| \geq 0, \\ \infty, & \text{διαφορετικά,} \end{cases} \quad (14.6)$$

όπου $|w_t|$ είναι το πλήθος των χαρακτήρων της λέξης w_t και s_{ij} ο αριθμός των κενών διαστημάτων ανάμεσα στις λέξεις. Προφανώς, $s_{ij} = j - i$.

Η περίπτωση $b(i, j) = \infty$ προκύπτει όταν το πλήθος των χαρακτήρων των λέξεων από w_i έως w_j με τα ενδιάμεσα κενά διαστήματα ξεπερνούν τον αριθμό των χαρακτήρων που μπορεί να χωρέσει στη γραμμή.

1. Ορισμός υποπροβλημάτων

- Θα ορίσουμε τα προβλήματα επιθεματικά. Έστω $D(i :)$, $i = 1, \dots, n$, το υποπρόβλημα εύρεσης της ελάχιστης ανομοιομορφίας αν η παράγραφος αποτελείται από τις λέξεις w_i, w_{i+1}, \dots, w_n . Δηλαδή στο υποπρόβλημα $D(i :)$ να διαχωρίσουμε τις παραπάνω λέξεις σε γραμμές προκειμένου να επιτευχθεί η χαμηλότερη ανομοιομορφία. Συνολικά υπάρχουν n τέτοια υποπροβλήματα.

2. Καθορισμός επιλογών

- Οι επιλογές που έχουμε όταν θέλουμε να επιλύσουμε το πρόβλημα $D(i :)$ αφορά τις λέξεις που θα περιέχει η γραμμή με πρώτη λέξη την w_i . Οι επιλογές είναι $n - i + 1$. Δηλαδή η γραμμή μπορεί να περιέχει μόνο τη λέξη w_i ή τις λέξεις w_i, w_{i+1} ή, ..., ή όλες τις λέξεις w_i, \dots, w_n . Πιθανώς, από κάποια λέξη και πέρα δεν μπορούν οι λέξεις να χωρέσουν στη γραμμή αφού μαζί με τα ενδιάμεσα κενά διαστήματα ξεπερνιέται η τιμή m .

3. Ορισμός αναδρομικής σχέσης

- Συμβολίζουμε με d_i τη λύση του υποπροβλήματος $D(i :)$. Δηλαδή το d_i αποτελεί μία ποσοτικοποίηση της μικρότερης δυνατής ανομοιομορφίας για την παράγραφο με λέξεις w_i, w_{i+1}, \dots, w_n χρησιμοποιώντας για κάθε γραμμή ως μέτρο το συντελεστή ανομοιομορφίας της γραμμής. Προφανώς w_{j-1} , για $j \geq i + 1$, είναι η τελευταία λέξη της γραμμής που ξεκινάει με τη λέξη w_i . Τότε $d_i = b(i, j - 1) + d_j$. Όπως είδαμε προηγουμένως, έχουμε $n - i + 1$ επιλογές ως προς την τελευταία λέξη της γραμμής. Από αυτές θα επιλέξουμε την καλύτερη. Επομένως ισχύει ο αναδρομικός τύπος

$$d_i = \begin{cases} \min\{b(i, j - 1) + d_j : i + 1 \leq j \leq n + 1\}, & 1 \leq i \leq n, \\ 0, & i = n + 1. \end{cases} \quad (14.7)$$

Για να μπορέσουμε να ανακτήσουμε τη λύση χρησιμοποιούμε το διάνυσμα p το οποίο παίρνει τιμές ως εξής:

$$p_i = \operatorname{argmin}\{b(i, j - 1) + d_j : i + 1 \leq j \leq n + 1\}. \quad (14.8)$$

Δηλαδή w_{p_i} είναι η λέξη που ξεκινάει την επόμενη γραμμή από τη γραμμή που ξεκινάει με τη λέξη w_i .

4. Τοπολογική ταξινόμηση υποπροβλημάτων

- Η σειρά με την οποία επιλύονται τα υποπροβλημάτα $D(i :)$ είναι σε φθίνουσα διάταξη των τιμών i εκκινώντας από $i = n$.

5. Επίλυση αρχικού προβλήματος

- Η τιμή d_1 αποτελεί τη λύση του αρχικού προβλήματος.

Αλγόριθμος

Αλγόριθμος 96 Διάσπαση κειμένου σε γραμμές με ελαχιστοποίηση της ανομοιομορφίας

Απαιτείται: Ακέραιος n , δισδιάστατος πίνακας b - το στοιχείο $b[i, j]$ ποσοτικοποιεί την ανομοιομορφία της γραμμής με πρώτη λέξη τη w_i και τελευταία τη w_j .

Επιστρέφεται: Ελάχιστη ανομοιομορφία παραγράφου και πίνακας p κάθε στοιχείο του οποίου υποδεικνύει την πρώτη λέξη της κάθε γραμμής από τη δεύτερη γραμμή και μετά.

```
1: function TEXTJUST(int n, int b[][], int p[])
2:    $d[n + 1] \leftarrow 0;$ 
3:   for  $i \leftarrow n; i \geq 1; i --$  do
4:      $d[i] \leftarrow \infty;$ 
5:     for  $j \leftarrow i + 1; j \leq n + 1; j ++$  do
6:       if  $d[i] > d[j] + b[i, j - 1]$  then
7:          $d[i] \leftarrow d[j] + b[i, j - 1];$ 
8:          $p[i] \leftarrow j;$ 
9:       end if
10:      end for
11:    end for
12:    return  $d[1];$ 
13: end function
```

Πολυπλοκότητα

Με γνωστούς τους συντελεστές ανομοιομορφίας $b(i, j)$ ο υπολογισμός της λύσης του κάθε υποπροβλήματος d_i παίρνει σταθερό αριθμό ΣΥΒ: εκτελούνται $n - i + 1$ προσθέσεις, $n - i + 1$ αφαιρέσεις (υπολογισμός της τιμής $j - 1$), $n - i$ συγκρίσεις και μία εκχώρηση. Άρα ο συνολικός αριθμός των ΣΥΒ που εκτελούνται προκειμένου να υπολογιστεί η τιμή d_1 που αποτελεί τη λύση στο αρχικό πρόβλημα είναι

$$\sum_{i=1}^n 3(n - i + 1) = \Theta(n^2).$$

Εναλλακτικά, μπορούμε να υπολογίσουμε την πολυπλοκότητα του Αλγόριθμου 96 ο οποίος κωδικοποιεί την παραπάνω διαδικασία, ως εξής. Ο αριθμός των υποπροβλημάτων είναι n και η λύση σε κάθε υποπρόβλημα υπολογίζεται θεωρώντας το πολύ $n - 1$ επιλογές ενώ η αποτίμηση της κάθε επιλογής γίνεται σε σταθερό αριθμό ΣΥΒ. Άρα η πολυπλοκότητα είναι $\Theta(n) \cdot O(n - 1) \cdot \Theta(1) = O(n^2)$.

Όμως ο υπολογισμός των συντελεστών $b(i, j)$ από την (14.6) είναι τάξης $\Theta(n^3)$. Μπορεί ο υπολογισμός να γίνει σε $\Theta(n^2)$ παρατηρώντας ότι μπορούμε να υπολογίσουμε τον όρο $b(i, j)$ από τον όρο $b(i, j - 1)$ εκτελώντας σταθερό αριθμό ΣΥΒ.

Πράγματι, για $i < j$,

$$\begin{aligned} b(i, j) &= (m - s_{ij} - \sum_{t=i}^j |w_t|)^2 \\ &= (m - (j - i) - \sum_{t=i}^j |w_t|)^2 \\ &= (m - (j - 1 - i) - 1 - \sum_{t=i}^{j-1} |w_t| - |w_j|)^2 \\ &= (\sqrt{b(i, j - 1)} - 1 - |w_j|)^2. \end{aligned}$$

Επομένως, αντί της (14.6), μπορούμε να υπολογίσουμε τους συντελεστές ανομοιομορφίας για $1 \leq i \leq j \leq n$, από τη σχέση

$$b(i, j) = \begin{cases} (m - |w_i|)^2, & \text{αν } i = j, \\ (\sqrt{b(i, j - 1)} - 1 - |w_j|)^2, & \text{αν } i < j, \sqrt{b(i, j - 1)} - 1 - |w_j| \geq 0, \\ \infty, & \text{διαφορετικά.} \end{cases}$$

Συνεπώς, δεδομένων των λέξεων w_1, \dots, w_n μπορούμε να επιτύχουμε ελαχιστοποίηση της ανομοιομορφίας της παραγράφου (που τις περιέχει) σε $O(n^2)$ βήματα.

Παράδειγμα

Παράδειγμα 78. Θέλουμε να διατάξουμε σε γραμμές των 9 χαρακτήρων με αριστερή στοίχιση τη φράση:

Στη γάτα δεν αρέσει το νερό

Το πλήθος των χαρακτήρων της κάθε λέξης παρουσιάζεται στον Πίνακα 14.1. Οι

i	1	2	3	4	5	6
w_i	Στη	γάτα	δεν	αρέσει	το	νερό
$ w_i $	3	4	3	6	2	4

Πίνακας 14.1: Αριθμός γραμμάτων - Παράδειγμα 78

συντελεστές ανομοιομορφίας όπως προκύπτουν από την (14.6) παρουσιάζονται στον Πίνακα 14.2.

	1	2	3	4	5	6
1	36	1	∞	∞	∞	∞
2		25	1	∞	∞	∞
3			36	∞	∞	∞
4				9	0	∞
5					49	4
6						25

Πίνακας 14.2: Συντελεστές ανομοιομορφίας υπολογισμένοι από (14.6) για $m = 9$
- Παράδειγμα 78

Από τις σχέσεις (14.7), (14.8) έχουμε

$$D_7 = 0,$$

$$D_6 = 25,$$

$$\begin{aligned} D_5 &= \min\{D_6 + b(5, 5), D_7 + b(5, 6)\} \\ &= \min\{25 + 49, 0 + 4\} = 4, \end{aligned} \quad p_5 = 7,$$

$$\begin{aligned} D_4 &= \min\{D_5 + b(4, 4), D_6 + b(4, 5), D_7 + b(4, 6)\} \\ &= \min\{4 + 9, 25 + 0, 0 + \infty\} = 13, \end{aligned} \quad p_4 = 5,$$

$$\begin{aligned} D_3 &= \min\{D_4 + b(3, 3), D_5 + b(3, 4), \\ &\quad D_6 + b(3, 5), D_7 + b(3, 6)\} \\ &= \min\{13 + 36, 4 + \infty, 25 + \infty, 0 + \infty\} = 49, \end{aligned} \quad p_3 = 4,$$

$$\begin{aligned} D_2 &= \min\{D_3 + b(2, 2), D_4 + b(2, 3), D_5 + b(2, 4), \\ &\quad D_6 + b(2, 5), D_7 + b(2, 6)\} \\ &= \min\{49 + 25, 13 + 1, 4 + \infty, 25 + \infty, 0 + \infty\} = 14, \end{aligned} \quad p_2 = 4,$$

$$\begin{aligned} D_1 &= \min\{D_2 + b(1, 1), D_3 + b(1, 2), D_4 + b(1, 3), \\ &\quad D_5 + b(1, 4), D_6 + b(1, 5), D_7 + b(1, 6)\} \\ &= \min\{14 + 36, 49 + 1, 13 + \infty, 4 + \infty, \\ &\quad 25 + \infty, 0 + \infty\} = 50, \end{aligned} \quad p_1 = 2 \text{ ή } p_1 = 3.$$

Επομένως υπάρχουν δύο τρόποι διάσπασης της φράσης σε γραμμές των εννέα χαρακτήρων που ελαχιστοποιούν τη (συνολική) ανομοιομορφία. Οι τρόποι αυτοί μαζί με τα αντίστοιχα διανύσματα p παρουσιάζονται στον Πίνακα 14.3.

Ο απλούστερος κανόνας που λέει να τοποθετήσουμε σε κάθε γραμμή όσο περισσότερες λέξεις μπορούμε - άπληστη επιλογή - οδηγεί σε πολλές περιπτώσεις σε μεγαλύτερη ανομοιομορφία. Ακολουθώντας τον κανόνα αυτό στο κείμενο του Παραδείγματος 78 παίρνουμε τη διάταξη του Πίνακα 14.4. Υπολογίζοντας την ανομοιομορφία της διάταξης αυτής ως το άθροισμα των τετραγώνων των κενών διαστημάτων στο τέλος κάθε γραμμής (με αριθμό χαρακτήρων γραμμής $m = 9$) παίρ-

Στη γάτα δεν αρέσει το νερό

Στη γάτα δεν αρέσει το νερό

$$p = [2, 4, 4, 5, 7, ,] \quad p = [3, 4, 4, 5, 7, ,]$$

Πίνακας 14.3: Διαφορετικές διατάξεις γραμμών που ελαχιστοποιούν την ανομοιομορφία - Παράδειγμα 78

Στη γάτα δεν αρέσει το νερό

Πίνακας 14.4: Στοιχίζοντας με βάση την άπληστη επιλογή

νούμε την τιμή 62 η οποία είναι μεγαλύτερη από την τιμή 50 που υπολογίσαμε παραπάνω. Το αποτέλεσμα αυτό αντικατοπτρίζει την ανομοιομορφία της παραπάνω διάταξης στη δεύτερη και τέταρτη γραμμή.

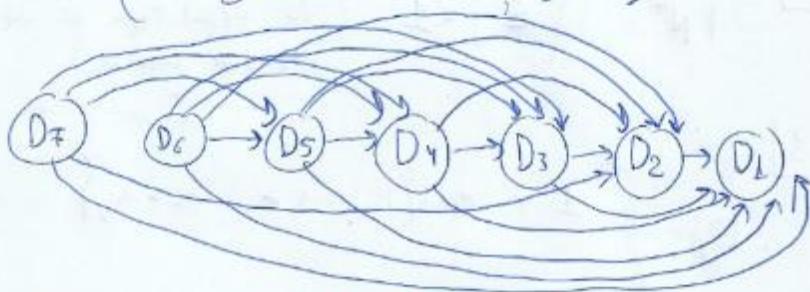
Σημειώσεις

1] $D(i:)$ (Στοιχίσης κεριών) → $f_{lexicin}$ ανά τη λέξη $w(i)$
και καταλήγει στη λέξη $w(n)$

Επεργασίεις
Πλευρών

2] $n = i+1$

3] $D_i = \begin{cases} \min\{b(i, j-1) + D_j : i+1 \leq j \leq n+1\} \\ 0 & i = n+1 \end{cases}$

4] 

5] D_L

* Στη γάτα δεν αρέσει το νερό

ΕΚΤΙΔΕΣ, σελ. 13 documents → Διαφάνεια Προγράμμα

* Στη γάτα δεν αρέσει το νερό

$P_L = 2 \rightarrow$ Στην επίσημη γραμμή η πρώτη λέξη είναι $\gamma 2^{\text{st}}$ (γάτα)

$P_2 = 4 \rightarrow$ Στην επίσημη γραμμή, η πρώτη λέξη είναι $\gamma 4^{\text{th}}$ (αρέσει), αφού καταλαβαίνουμε ότι στην $\gamma 2^{\text{nd}}$ (γάτα) και $\gamma 3^{\text{rd}}$ (νερό)

11a. Ο μικρότερος αριθμός χαρτονομισμάτων για ένα ποσό

Διαφάνειες

Slides_and_Exercises→09_DynamicExer.pdf σελ. 5-7

Εκφώνηση

Άσκηση 3 Να περιγράψετε και να μελετήσετε αλγόριθμο ο οποίος δέχεται σαν είσοδο έναν πίνακα A με n διαφορετικά είδη (χαρτο)νομισμάτων σε κάποιο συναλλαγματικό μέσο (π.χ., ευρώ, δολάριο, κλπ) ταξινομημένα κατά αύξουσα σειρά και ένα χρηματικό ποσό X και υπολογίζει το X σαν άδροισμα με τον ελάχιστο αριθμό (χαρτο)νομισμάτων. Θεωρείστε ότι το μικρότερο (χαρτο)νόμισμα έχει αξία 1.

1. Ορισμός υποπροβλημάτων

Δύση Θα πρέπει να παρατηρήσουμε ότι επιλέγοντας άπληστα «χαρτο»νάμισμα μεγαλύτερης αξίας πρώτη επιλογή (δεν οδηγεί πάντα στη βέλτιστη λύση).

2. Καθορισμός επιλογών

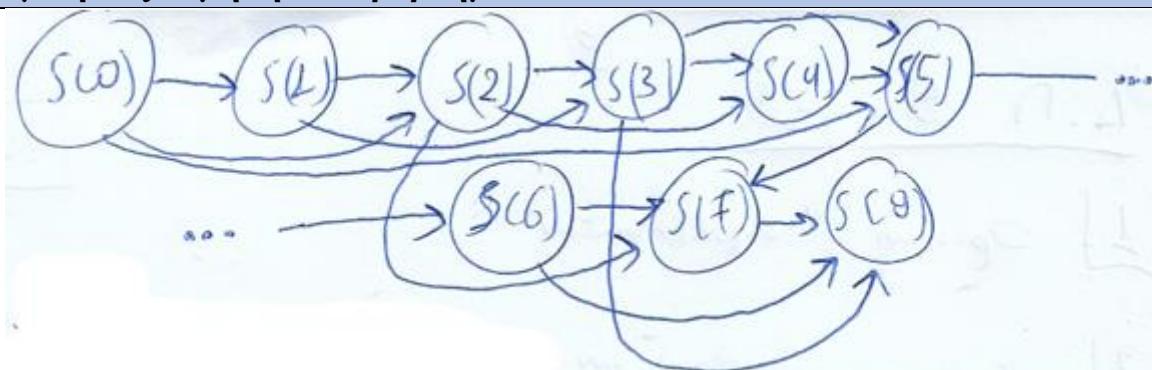
αν $Q = 30$ και $A[1] = 1, A[2] = 15, A[3] = 25$, και πρώτα χρησιμοποιήσουμε το $A[3] = 25$ θα πάρουμε τη λύση $(25, 1, 1, 1, 1, 1)$, δηλαδή τέσσερα νομίσματα, ενώ η βέλτιστη λύση αποτελείται από δύο (χαρτο)νομίσματα είναι δύο ($A[2] = 15$).

3. Ορισμός αναδρομικής σχέσης

Η αναδρομική συνάρτηση που μετράει τον αριθμό των χαρτονομισμάτων είναι

$$p(x) = \begin{cases} 1 + \min\{p(x - A[i]) : i \in \{1, \dots, n\}, x - A[i] \geq 0\}, & X \geq x \geq 1, \\ 0, & x = 0. \end{cases} \quad (4)$$

4. Τοπολογική ταξινόμηση υποπροβλημάτων



5. Επίλυση αρχικού προβλήματος

Ο ελάχιστος αριθμός των (χαρτο)νομισμάτων δίνεται από $p(x)$ (δες υλοποίηση).

Ο ελάχιστος αριθμός των (χαρ-

Παράδειγμα

Παράδειγμα 2 Θεωρούμε νομίσματα του ευρώ $A[1] = 1, A[2] = 2, A[3] = 5$, κ.λπ. Έστω $X = 8$ ευρώ. Για να βρεθεί ο ελάχιστος αριθμός χαρτονομίσματων αντίστοιχης αξίας εκτιμούμε την 4 ξεκινώντας από $x = 2$ έως $x = 8$. Έχουμε,

$x = 1 :$

$$p(1) = 1 + \min\{p(1 - 1)\} = 1 + 0 = 1$$

$x = 2 :$

$$p(2) = 1 + \min\{p(2 - 1), p(2 - 2)\} = 1 + \min\{1, 0\} = 1$$

$x = 3 :$

$$p(3) = 1 + \min\{p(3 - 1), p(3 - 2)\} = 1 + \min\{1, 1\} = 2$$

$x = 4 :$

$$p(4) = 1 + \min\{p(4 - 1), p(4 - 2)\} = 1 + \min\{2, 1\} = 2$$

$x = 5 :$

$$p(5) = 1 + \min\{p(5 - 1), p(5 - 2), p(5 - 5)\} = 1 + \min\{2, 1, 0\} = 1$$

$x = 6 :$

$$p(6) = 1 + \min\{p(6 - 1), p(6 - 2), p(6 - 5)\} = 1 + \min\{1, 2, 1\} = 2$$

$x = 7 :$

$$p(7) = 1 + \min\{p(7 - 1), p(7 - 2), p(7 - 5)\} = 1 + \min\{2, 1, 1\} = 2$$

$x = 8 :$

$$p(8) = 1 + \min\{p(8 - 1), p(8 - 2), p(8 - 5)\} = 1 + \min\{2, 2, 2\} = 3$$

Αλγόριθμος

Algorithm 3 Ελάχιστα χαρτονομίσματα

```
1: void MinNotes(int n, int A[], int X)
2: for x  $\leftarrow$  1; x  $\leq$  X; x ++ do
3:   keep[x]  $\leftarrow$  0;
4: end for
5: p[0]  $\leftarrow$  0;
6: for x  $\leftarrow$  1; x  $\leq$  X; x ++ do
7:   p[x]  $\leftarrow$  X;
8:   min_index  $\leftarrow$   $-\infty$ ;
9:   for i  $\leftarrow$  1; i  $\leq$  n; i ++ do
10:    if x - A[i]  $\geq$  0 and p[x] > 1 + p[x - A[i]] then
11:      p[x]  $\leftarrow$  1 + p[x - A[i]]; min_index  $\leftarrow$  i;
12:    end if
13:   end for
14:   keep[x]  $\leftarrow$  min_index
15: end for
16: println("Min number of notes: ", p[X]);
17: println("List of Notes");
18: x  $\leftarrow$  X;
19: while x>0 do
20:   println("Note: ", A[keep[x]]);
21:   x  $\leftarrow$  x - A[keep[x]]
22: end while
```

είναι $O(nX)$.

Σημειώσεις

(θ)

29/11/2023

Chapter IIa

Δ.Π 5 βράδυ

1] Ορισμός Υπορρεύματων

2] Καθυρεγμένος Εντολή

3] Ορισμός Αντρομενής σχετικών

4] Τοπολογική Ταυτότητα Υπορρεύματων

5] Σπιλυρητικοί αριθμοί προβλημάτων

{Ο μεριότερος αριθμός χαρτονηματίων (MinNotes)}

1] $S(:x)$ → Η λύση των υπορρεύματων που καταλήγουν στο πρόσω x , δηλαδή, το ελάχιστο πλήθος των χαρτονηματίων που μας δίνει το πρόσω x

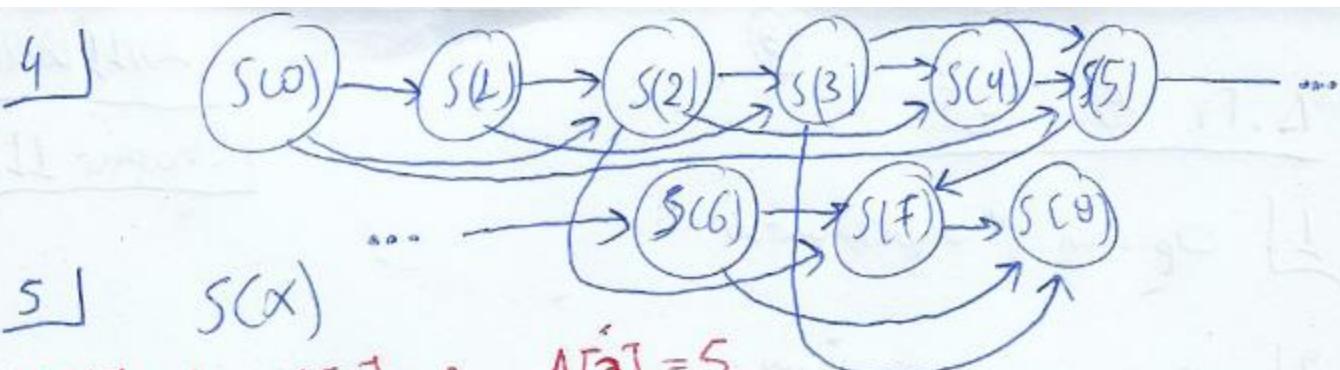
Πρόθετη

Παραδείγματα

2] $x \leq X$ $j \in \{1, n\}$ $x - A[j] \geq 0$

$$\begin{array}{c} 1€ \quad 2€ \quad 5€ \\ A[1] < A[2] < A[3] < \dots < A[n] \\ X = 8€, \quad x \in \{0€, \dots, 8€\} \end{array}$$

3] $S(x) = \begin{cases} 1 + \min \{ S(x - A[j]) : 1 \leq j \leq n, x - A[j] \geq 0 \} & \text{if } 1 \leq x \leq X \\ 0 & \text{if } x = 0 \end{cases}$



5] $S(x)$

$$A[1] = 1, A[2] = 2, A[3] = 5$$

$$S(0) = 0$$

$$S(1) = L + \min\{S(1-A[1])\} = L + S(0) = L, P_1 = L \text{ €}$$

$$\begin{aligned} S(2) &= L + \min\{S(2-A[2]), S(2-A[1])\} = L + \min\{S(0), S(1)\} \\ &= L + \min\{0, L\} = L, P_2 = L \text{ €} \end{aligned}$$

$$\begin{aligned} S(3) &= L + \min\{S(3-A[2]), S(3-A[1])\} = L + \min\{S(1), S(2)\} \\ &= L + \min\{L, L\} = 2, P_3 = \cancel{1 \text{ €}}, 2 \text{ €} \end{aligned}$$

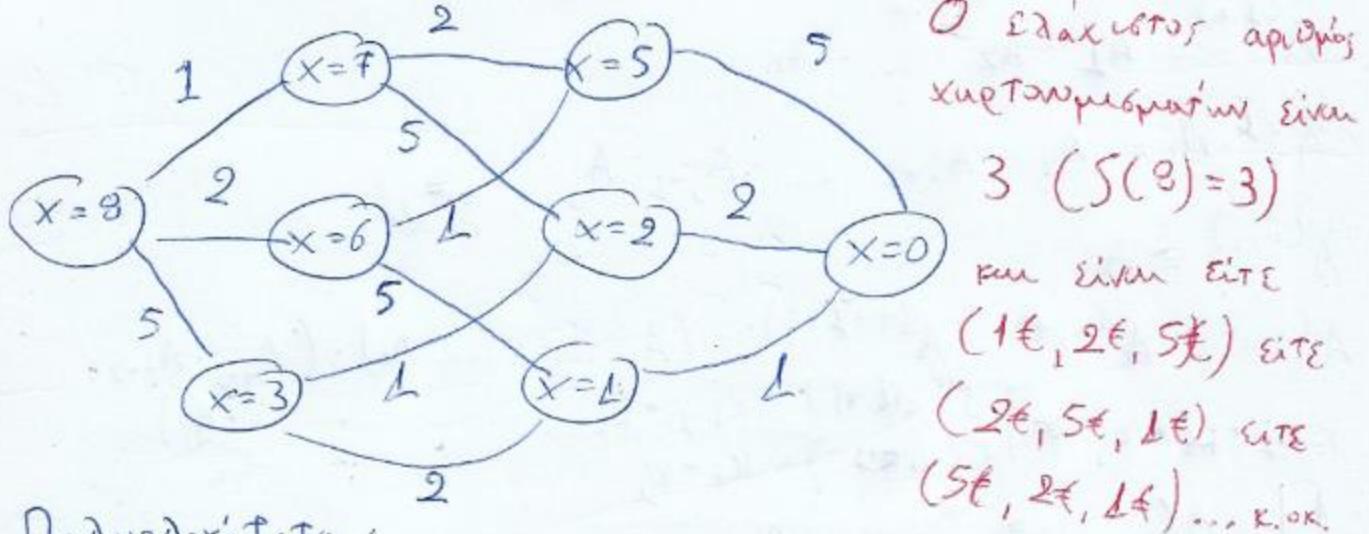
$$\begin{aligned} S(4) &= L + \min\{S(4-A[2]), S(4-A[1])\} = L + \min\{S(2), S(3)\} \\ &= L + \min\{L, 2\} = 2, P_4 = 2 \text{ €} \end{aligned}$$

$$\begin{aligned} S(5) &= L + \min\{S(5-A[3]), S(5-A[2]), S(5-A[1])\} = L + \min\{S(0), S(1), S(2)\}, \\ &= L + \min\{0, 2, 2\} = L, P_5 = 5 \text{ €} \end{aligned}$$

$$\begin{aligned} S(6) &= L + \min\{S(6-A[3]), S(6-A[2]), S(6-A[1])\} \\ &= L + \min\{S(2), S(4), S(5)\} = L + \min\{L, 2, L\} = 2, P_6 = \cancel{1 \text{ €}}, 2 \text{ €} \end{aligned}$$

$$\begin{aligned} S(F) &= L + \min\{S(F-A[3]), S(F-A[2]), S(F-A[1])\} \\ &= L + \min\{S(2), S(5), S(6)\} = L + \min\{L, 2, 2\} = 2, P_F = \cancel{1 \text{ €}}, 2 \text{ €} \end{aligned}$$

$$\begin{aligned} S(8) &= L + \min\{S(8-A[3]), S(8-A[2]), S(8-A[1])\} \\ &= L + \min\{S(3), S(6), S(F)\} = L + \min\{2, 2, 2\} = 3, P_8 = \cancel{1 \text{ €}}, 2 \text{ €}, 5 \text{ €} \end{aligned}$$



Πολυπλοκότητα:

Για κάθε $x \in \{1, n\}$ κάνω η αρχικής, $n-L$ συκρίσεως και L πρόσθιες από στάδιο 3]

$O(n^*X)$

11b. Πολλαπλασιασμός Πινάκων

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 93-100

Πρόβλημα

Από τη Γραμμική Άλγεβρα είναι γνωστό ότι το γινόμενο δύο πινάκων A, B διάστασης $p \times q$ και $q \times r$ είναι ένας πίνακας $C = A \cdot B$ όπου το κάθε στοιχείο του δίνεται από τη σχέση

$$C[i, j] = \sum_{k=1}^q A[i, k] \cdot B[k, j], \text{ για } i = 1, \dots, p, j = 1, \dots, r.$$

Εύκολα προκύπτει ότι κάθε στοιχείο του C παράγεται από q πολλαπλασιασμούς· επειδή η διάσταση του C είναι $p \times r$, για τον υπολογισμό όλων των στοιχείων του εκτελούνται $p \cdot q \cdot r$ πολλαπλασιασμοί. Όμως αν έχουμε μία σειρά από πίνακες A_1, A_2, \dots, A_n με συμβατές μεταξύ τους διαστάσεις - ο αριθμός των στηλών του πίνακα A_i είναι ίσος με τον αριθμό των γραμμών του πίνακα A_{i+1} , για $i = 1, \dots, n-1$ - πόσους πολλαπλασιασμούς χρειάζεται να εκτελέσουμε προκειμένου να βρούμε το γινόμενο

$$A_1 \cdot A_2 \cdots A_n; \quad (14.9)$$

Ιδέα

Η απάντηση στο ερώτημα αυτό ποικίλει ανάλογα με τη σειρά που εκτελούνται οι πολλαπλασιασμοί. Για παράδειγμα, έστω ότι ο πίνακας A_i έχει k_{i-1} γραμμές και k_i στήλες. Τότε για $n = 3$ υπάρχουν δύο τρόποι για να υπολογίσουμε το γινόμενο $A_1 \cdot A_2 \cdot A_3$. Είτε το υπολογίζουμε ως $A_1 \cdot (A_2 \cdot A_3)$ ή ως $(A_1 \cdot A_2) \cdot A_3$. Στην πρώτη περίπτωση ο αριθμός των πολλαπλασιασμών που εκτελούνται είναι $k_1 \cdot k_2 \cdot k_3 + k_0 \cdot k_1 \cdot k_3$ ενώ στη δεύτερη περίπτωση $k_0 \cdot k_1 \cdot k_2 + k_0 \cdot k_2 \cdot k_3$. Δηλαδή βλέπουμε ότι το πλήθος των πολλαπλασιασμών μπορεί να διαφέρει στις δύο περιπτώσεις. Γενικότερα, ο αριθμός των τρόπων που μπορεί να υπολογιστεί το γινόμενο (14.9) είναι εκθετικός.⁴ Από όλους αυτούς τους τρόπους θέλουμε να βρούμε αυτόν που υπολογίζει το γινόμενο (14.9) εκτελώντας το μικρότερο αριθμό πολλαπλασιασμών. Θα επιλύσουμε το πρόβλημα αυτό με ΔΠ. Προς τούτο ακολου-

1. Ορισμός υποπροβλημάτων

- Θα συμβολίσουμε τον πίνακα που προκύπτει από το γινόμενο (14.9) ως $A_{1:n}$.

Το πρόβλημα είναι να υπολογίσουμε τον πίνακα αυτό εκτελώντας τον μικρότερο αριθμό πολλαπλασιασμών. Θα ορίσουμε ως $A(i:j)$ το υποπρόβλημα υπολογισμού του πίνακα

$$A_{i,j} = A_i \cdot A_{i+1} \cdot \dots \cdot A_j, \quad 1 \leq i \leq j \leq n. \quad (14.10)$$

Έστω $m_{i,j}$ η λύση στο πρόβλημα αυτό. Δηλαδή, το $m(i,j)$ συμβολίζει το μικρότερο αριθμό πολλαπλασιασμών που εκτελούνται προκειμένου να υπολογιστούν τα στοιχεία του πίνακα $A_{i,j}$. Παρατηρείστε ότι αν $i = j$ ο πίνακας $A_{i,i} = A_i$ και άρα για τον υπολογισμό του δεν λαμβάνει χώρα κανένας πολλαπλασιασμός, δηλαδή $m_{i,i} = 0$.

2. Καθορισμός επιλογών

- Παρατηρούμε ότι για οποιοδήποτε $t \geq i$ και $t < j$, ο πίνακας $A_{i,j}$ μπορεί να υπολογιστεί αν είναι γνωστοί οι πίνακες $A_{i,t}$ και $A_{t+1,j}$ αφού

$$A_{i,j} = A_{i,t} \cdot A_{t+1,j} = (A_i \cdot \dots \cdot A_t) \cdot (A_{t+1} \cdot \dots \cdot A_j). \quad (14.11)$$

Ο αριθμός των πολλαπλασιασμών που εκτελούνται για τον υπολογισμό του $A_{i,j}$ από την (14.11) είναι

$$c_t = k_{i-1} \cdot k_t \cdot k_j. \quad (14.12)$$

Από την παραπάνω σχέση είναι προφανές ότι υπάρχουν $j - i$ επιλογές για την τιμή του δείκτη t .

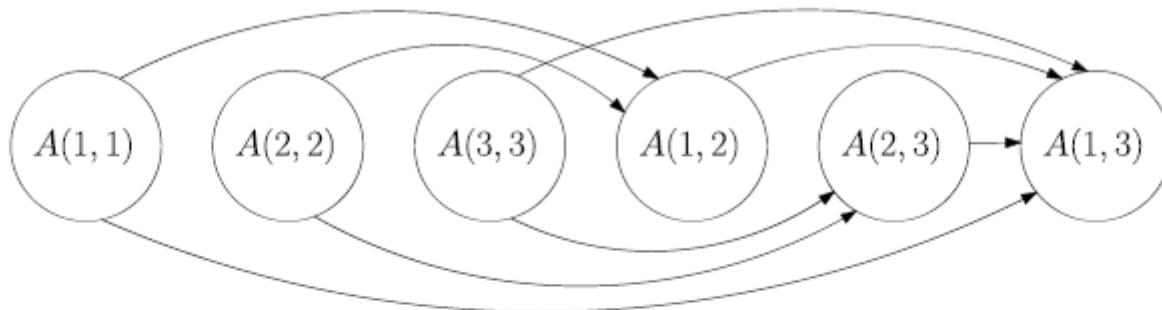
3. Ορισμός αναδρομικής σχέσης

3. Η λύση του υποπροβλήματος $A(i, j)$ προκύπτει από την καλύτερη επιλογή της τιμής t ώστε να προκύψει ο πίνακας $A_{i,j}$ από την (14.11). Δηλαδή για $i, j \in \{1, \dots, n\}$, με $i \leq j$, έχουμε

$$m_{i,j} = \begin{cases} 0, & i = j, \\ \min_{t \in \{i, \dots, j-1\}} \{m_{i,t} + m_{t+1,j} + c_t\}, & i < j, \end{cases} \quad (14.13)$$

όπου ο συντελεστής c_t ορίζεται από την (14.12).

4. Τοπολογική ταξινόμηση υποπροβλημάτων



Σχήμα 14.4: Γράφημα υποπροβλημάτων για τον πολλαπλασιασμό τριών πινάκων

4. Από την (14.13) προκύπτει ότι η σειρά με την οποία θα πρέπει να λυθούν τα υποπροβλήματα υποδεικνύεται από την διαφορά των δεικτών $j - i$. Δηλαδή αρχικώς έχουμε τα υποπροβλήματα που έχουν τετριμμένη λύση ($j = i \Rightarrow j - i = 0$), στη συνέχεια θα λυθούν τα υποπροβλήματα που ορίζονται για $j - i = 1$ μετά αυτά για τα οποία $j - i = 2$ κοκ. Μία οπτική αναπαράσταση του γραφήματος των υποπροβλημάτων για $n = 3$ απεικονίζεται στο Σχήμα 14.4.

5. Επίλυση αρχικού προβλήματος

5. Ο ελάχιστος αριθμός πολλαπλασιασμών δίνεται από την τιμή του $m_{1,n}$.

Ιδέα

Η προηγούμενη ανάλυση οδηγεί στον Αλγόριθμο 97. Επειδή πέρα από τον ελάχιστο αριθμό των πολλαπλασιασμών μας ενδιαφέρει να ιχνηλατήσουμε με ποια σειρά πρέπει να τους εκτελέσουμε προκειμένου να επιτύχουμε τον αριθμό αυτό, χρησιμοποιούμε ένα δισδιάστατο διάνυσμα p . Για $i < j$, η τιμή $p_{i,j}$ (στοιχείο $p[i, j]$ στον Αλγόριθμο 97) είναι η τιμή t για την οποία επιτυγχάνεται ο ελάχιστος αριθμός των πολλαπλασιασμών προκειμένου να υπολογιστεί ο πίνακας $A_{i,j}$ από την (14.11).

Αλγόριθμος 97 Υπολογισμός ελάχιστου αριθμού πολλαπλασιασμών

Απαιτείται: Ακέραιος n , πίνακας k με ακέραιους στις θέσεις 0 ως n . Ο πίνακας A_i έχει $k[i - 1]$ γραμμές και $k[i]$ στήλες.

Επιστρέφεται: Ελάχιστος αριθμός πολλαπλασιασμών για τον υπολογισμό του γινόμενου των πινάκων A_1, A_2, \dots, A_n και δισδιάστατος πίνακας p για την ιχνηλάτηση της λύσης.

```

1: function MINMATMULT(int n, int k[], int p[][])
2:   for  $i \leftarrow 1; i \leq n; i++$  do
3:      $m[i, i] \leftarrow 0;$ 
4:      $p[i, i] \leftarrow i;$ 
5:   end for
6:   for  $r \leftarrow 1; r \leq n - 1; r++$  do
7:     for  $i \leftarrow 1; i \leq n; i++$  do
8:        $j \leftarrow i + r;$ 
9:       if  $j \leq n$  then
10:         $m[i, j] \leftarrow \infty;$ 
11:        for  $t \leftarrow i; t \leq j - 1; t++$  do
12:           $temp \leftarrow m[i, t] + m[t + 1, j] + k[i - 1] \cdot k[t] \cdot k[j];$ 
13:          if  $temp < m[i, j]$  then
14:             $m[i, j] \leftarrow temp;$ 
15:             $p[i, j] \leftarrow t;$ 
16:          end if
17:        end for
18:      end if
19:    end for
20:  end for
21:  return  $m[1, n];$ 
22: end function

```

Πολυπλοκότητα

Για να υπολογίσουμε την πολυπλοκότητα του Αλγόριθμου 97, παρατηρούμε ότι στον πλέον εσωτερικό βρόχο γίνεται σταθερός αριθμός ΣΥΒ. Αν $a > 0$ είναι ο αριθμός αυτός, ο συνολικός αριθμός ΣΥΒ - αγνοώντας την αρχικοποίηση των δομών - δίνεται από τη σχέση

$$\begin{aligned}
& \sum_{i=1}^{n-1} \sum_{j=i+1}^n \sum_{t=i}^{j-1} a = \sum_{i=1}^{n-1} \sum_{j=i+1}^n (j-i)a = \sum_{i=1}^{n-1} \sum_{r=1}^{n-i} ra \\
& = a \cdot \sum_{i=1}^{n-1} \frac{(n-i)(n-i+1)}{2} = \frac{a}{2} \left(\sum_{i=1}^{n-1} (n-i)^2 + \sum_{i=1}^{n-1} (n-i) \right) \\
& = \frac{a}{2} \left(\sum_{i=1}^{n-1} i^2 + \sum_{i=1}^{n-1} i \right) = \Theta(n^3).
\end{aligned}$$

Παράδειγμα 79. Έστω πίνακες:

A_1 διάστασης 30×35 ,

A_2 διάστασης 35×15 ,

Σειρά εκτέλεσης	Αριθμός πολλαπλασιασμών
$A_1 \cdot A_2$	$30 \cdot 35 \cdot 15 = 15750$
$(A_1 \cdot A_2) \cdot A_3$	$30 \cdot 15 \cdot 5 = 2250$
$((A_1 \cdot A_2) \cdot A_3) \cdot A_4$	$30 \cdot 5 \cdot 10 = 1500$
$((((A_1 \cdot A_2) \cdot A_3) \cdot A_4) \cdot A_5$	$30 \cdot 10 \cdot 20 = 6000$

Πίνακας 14.5: Αναλυτικός υπολογισμός του αριθμού των πολλαπλασιασμών με σειρά από αριστερά προς τα δεξιά - Παράδειγμα 79

A_3 διάστασης 15×5 ,

A_4 διάστασης 5×10 ,

A_5 διάστασης 10×20 .

Από τα παραπάνω προκύπτουν οι ακόλουθοι αριθμοί γραμμών-στηλών:

$$k_0 = 30, k_1 = 35, k_2 = 15, k_3 = 5, k_4 = 10, k_5 = 20.$$

Θέλουμε να υπολογίσουμε το γινόμενο

$$A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5$$

με το μικρότερο δυνατό αριθμό πολλαπλασιασμών. Ο αριθμός των πολλαπλασιασμών που απαιτούνται για να προκύψει το παραπάνω γινόμενο αν κάνουμε τους πολλαπλασιασμούς από αριστερά προς τα δεξιά - με την προτεραιότητα που υποδηλώνουν οι παρενθέσεις:

$$(((A_1 \cdot A_2) \cdot A_3) \cdot A_4) \cdot A_5,$$

είναι ίσος με 25500. Ο αριθμός αυτός προκύπτει από το άθροισμα των αριθμών της δεύτερης στήλης του Πίνακα 14.5. Σε κάθε γραμμή του πίνακα αυτού υπολογίζεται ο αριθμός των πολλαπλασιασμών της πράξης που σημειώνεται στο αντίστοιχο κελί της πρώτης στήλης.

Εφαρμογή ΔΠ

Για την εφαρμογή του ΔΠ αρχικώς επιλύουμε τα τετριμμένα υποπροβλήματα $A_{i,i}$, ήτοι

$$\begin{array}{ll} m_{1,1} = 0, & p_{1,1} = 1, \\ m_{2,2} = 0, & p_{2,2} = 2, \\ m_{3,3} = 0, & p_{3,3} = 3, \\ m_{4,4} = 0, & p_{4,4} = 4, \\ m_{5,5} = 0, & p_{5,5} = 5. \end{array}$$

Στη συνέχεια υπολογίζουμε μέσω της (14.13) τις λύσεις των υποπροβλημάτων σύμφωνα με την (θετική) τιμή της παράστασης $j - i$ σε αύξουσα σειρά. Έχουμε,

$$j - i = 1 :$$

$$\begin{aligned} m_{1,2} &= \min\{m_{1,1} + m_{2,2} + k_0 \cdot k_1 \cdot k_2\} \\ &= \min\{0 + 0 + 30 \cdot 35 \cdot 15\} = 15750, \\ p_{1,2} &= 1, \end{aligned}$$

$$\begin{aligned} m_{2,3} &= \min\{m_{2,2} + m_{3,3} + k_1 \cdot k_2 \cdot k_3\} \\ &= \min\{0 + 0 + 35 \cdot 15 \cdot 5\} = 2625, \\ p_{2,3} &= 2, \end{aligned}$$

$$\begin{aligned} m_{3,4} &= \min\{m_{3,3} + m_{4,4} + k_2 \cdot k_3 \cdot k_4\} \\ &= \min\{0 + 0 + 15 \cdot 5 \cdot 10\} = 750, \\ p_{3,4} &= 3, \end{aligned}$$

$$\begin{aligned} m_{4,5} &= \min\{m_{4,4} + m_{5,5} + k_3 \cdot k_4 \cdot k_5\} \\ &= \min\{0 + 0 + 5 \cdot 10 \cdot 20\} = 1000, \\ p_{4,5} &= 4, \end{aligned}$$

$$j - i = 2 :$$

$$\begin{aligned} m_{1,3} &= \min\{m_{1,1} + m_{2,3} + k_0 \cdot k_1 \cdot k_3, \\ &\quad m_{1,2} + m_{3,3} + k_0 \cdot k_1 \cdot k_2 \cdot k_3\} \\ &= \min\{0 + 2625 + 30 \cdot 35 \cdot 5, 15750 + 0 + 30 \cdot 15 \cdot 5\} \\ &= \min\{7875, 18000\} = 7875, \\ p_{1,3} &= 1, \end{aligned}$$

$$\begin{aligned}
m_{2,4} &= \min\{m_{2,2} + m_{3,4} + k_1 \cdot k_2 \cdot k_4, \\
&\quad m_{2,3} + m_{4,4} + k_1 \cdot k_3 \cdot k_4\} \\
&= \min\{0 + 750 + 35 \cdot 15 \cdot 10, 2625 + 0 + 35 \cdot 5 \cdot 10\} \\
&= \min\{6000, 4375\} = 4375, \\
p_{2,4} &= 3,
\end{aligned}$$

$$\begin{aligned}
m_{3,5} &= \min\{m_{3,3} + m_{4,5} + k_2 \cdot k_3 \cdot k_5, \\
&\quad m_{3,4} + m_{5,5} + k_2 \cdot k_4 \cdot k_5\} \\
&= \min\{0 + 1000 + 15 \cdot 5 \cdot 20, 750 + 0 + 15 \cdot 10 \cdot 20\} \\
&= \min\{1500, 3000\} = 1500, \\
p_{3,5} &= 3,
\end{aligned}$$

$j - i = 3 :$

$$\begin{aligned}
m_{1,4} &= \min\{m_{1,1} + m_{2,4} + k_0 \cdot k_1 \cdot k_4, \\
&\quad m_{1,2} + m_{3,4} + k_0 \cdot k_2 \cdot k_4, \\
&\quad m_{1,3} + m_{4,4} + k_0 \cdot k_3 \cdot k_4\} \\
&= \min\{0 + 4375 + 30 \cdot 35 \cdot 10, 15750 + 750 + 30 \cdot 15 \cdot 10, \\
&\quad 7875 + 0 + 30 \cdot 5 \cdot 10\} \\
&= \min\{14875, 21000, 9375\} = 9375, \\
p_{1,4} &= 3,
\end{aligned}$$

$$\begin{aligned}
m_{2,5} &= \min\{m_{2,2} + m_{3,5} + k_1 \cdot k_2 \cdot k_5, \\
&\quad m_{2,3} + m_{4,5} + k_1 \cdot k_3 \cdot k_5, \\
&\quad m_{2,4} + m_{5,5} + k_1 \cdot k_4 \cdot k_5, \} \\
&= \min\{0 + 1500 + 35 \cdot 15 \cdot 20, 2625 + 1000 + 35 \cdot 5 \cdot 20, \\
&\quad 4375 + 0 + 35 \cdot 10 \cdot 20\} \\
&= \min\{12000, 7125, 11375\} = 7125,
\end{aligned}$$

$$p_{2,5} = 3,$$

$j - i = 4 :$

$$\begin{aligned}
m_{1,5} &= \min\{m_{1,1} + m_{2,5} + k_0 \cdot k_1 \cdot k_5, \\
&\quad m_{1,2} + m_{3,5} + k_0 \cdot k_2 \cdot k_5, \\
&\quad m_{1,3} + m_{4,5} + k_0 \cdot k_3 \cdot k_5, \\
&\quad m_{1,4} + m_{5,5} + k_0 \cdot k_4 \cdot k_5\}
\end{aligned}$$

$$\begin{aligned}
&= \min\{0 + 7125 + 30 \cdot 35 \cdot 20, 15750 + 1500 + 30 \cdot 15 \cdot 20, \\
&\quad 7875 + 1000 + 30 \cdot 5 \cdot 20, 9375 + 0 + 30 \cdot 10 \cdot 20\} \\
&= \min\{28125, 26250, 11875, 15375\} = 11875, \\
p_{1,5} &= 3.
\end{aligned}$$

Μπορούμε να παρουσιάσουμε συγκεντρωτικά τους υπολογισμούς στους πίνακες

$$m = \begin{bmatrix} 0 & 15750 & 7875 & 9375 & 11875 \\ & 0 & 2625 & 4375 & 7125 \\ & & 0 & 750 & 1500 \\ & & & 0 & 1000 \\ & & & & 0 \end{bmatrix}, p = \begin{bmatrix} 1 & 1 & 1 & 3 & 3 \\ & 2 & 2 & 3 & 3 \\ & & 3 & 3 & 3 \\ & & & 4 & 4 \\ & & & & 5 \end{bmatrix}.$$

Επομένως μπορούμε να υπολογίσουμε το γινόμενο των πέντε πινάκων εκτελώντας μόλις 11875 πολλαπλασιασμούς.⁵ Όμως με ποια σειρά πρέπει να εκτελέσουμε τους πολλαπλασιασμούς; Βλέπουμε ότι $p_{1,5} = 3$. Αυτό σημαίνει ότι $A_{1,5} = A_{1,3} \cdot A_{4,5}$. Επειδή $p_{1,3} = 1$, έχουμε ότι $A_{1,3} = A_{1,1} \cdot A_{2,3}$ και άρα $A_{1,3} = A_1 \cdot (A_2 \cdot A_3)$. Αντίστοιχα, επειδή $p_{4,5} = 4$ έχουμε ότι $A_{4,5} = A_{4,4} \cdot A_{5,5} = A_4 \cdot A_5$. Επομένως η σειρά με την οποία πρέπει να πολλαπλασιάσουμε τους πίνακες για να επιτύχουμε τον ελάχιστο αριθμό πολλαπλασιασμών των στοιχείων τους είναι

$$(A_1 \cdot (A_2 \cdot A_3)) \cdot (A_4 \cdot A_5).$$

Σημειώσεις

{ Πολλαπλασιασμός Πίνακα (MinMat Mult) } Chapter 12b

$$\left. \begin{array}{l} A_1 \quad S(A_1) = k_0 * k_1 \\ A_2 \quad S(A_2) = k_L * k_2 \\ A_3 \quad S(A_3) = k_2 * k_3 \\ \vdots \\ A_n \quad S(A_n) = k_{n-1} * k_n \end{array} \right\} \quad \begin{array}{l} C = A_1 \cdot A_2 \cdot A_3 \\ C = (A_1 \cdot A_2) \cdot A_3 \\ \quad \downarrow \quad k_0 * k_2 \\ k_0 * k_1 * k_2 + k_0 * k_2 * k_3 \\ \vdots \\ C = A_L \cdot \frac{(A_2 \cdot A_3)}{k_L * k_3} \\ \quad \downarrow \\ k_0 * k_1 * k_2 + k_L * k_2 * k_3 \end{array} \quad 131$$

$k_i \rightarrow$ αριθμός
πολλαπλασιασμού

$$A^{(1,n)} = A_1 \cdot A_2 \cdot \dots \cdot A_n$$

$$A^{(i,j)} = A_i \cdot A_{i+1} \cdot \dots \cdot A_{j-1} \cdot A_j, \quad i \leq j$$

$$A^{(i,i)} = A_i$$

$$A^{(i,j)} = A^{(i,t)} \cdot A^{(t+1,j)} = (A_i \cdot A_{i+1} \cdot \dots \cdot A_t) \cdot (A_{t+1} \cdot A_{t+2} \cdot \dots \cdot A_j)$$

$$k_{i-1} * k_t * k_j \Leftarrow \begin{cases} S(A^{(i,t)}) = k_{i-1} * k_t \\ S(A^{(t+1,j)}) = k_t * k_j \end{cases}$$

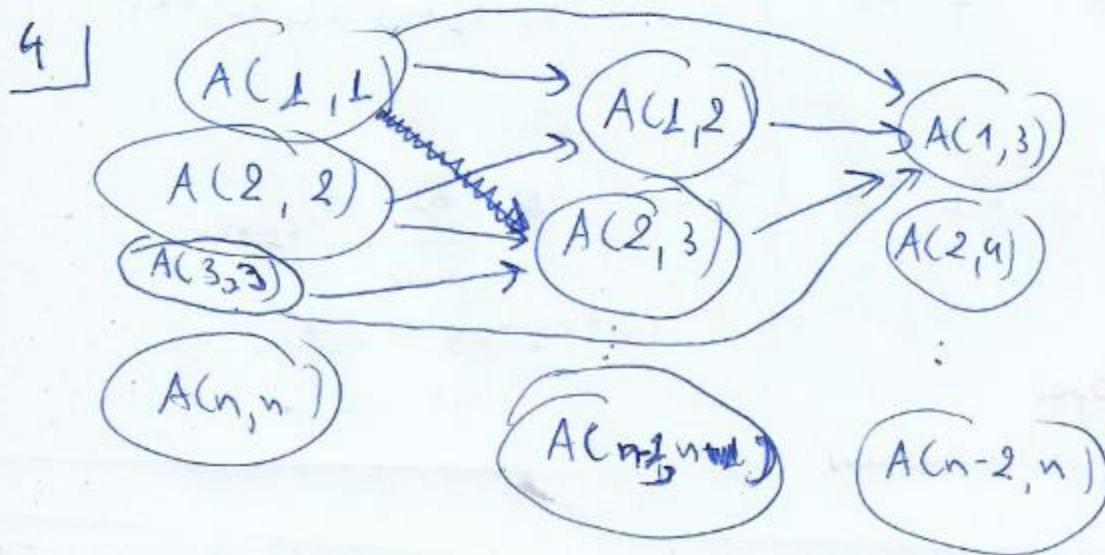
1] $A^{(i,j)} \rightarrow$ To παρόβλημα είρεσης $A^{(i,j)} = A_i \cdot A_{i+1} \cdot \dots \cdot A_j$
 των ελαχιστών αριθμών
 Ορισμένος συρροτοσεράς πολλαπλασιασμών

2] Εστιν $m(i,j)$ η λύση των $A^{(i,j)}$

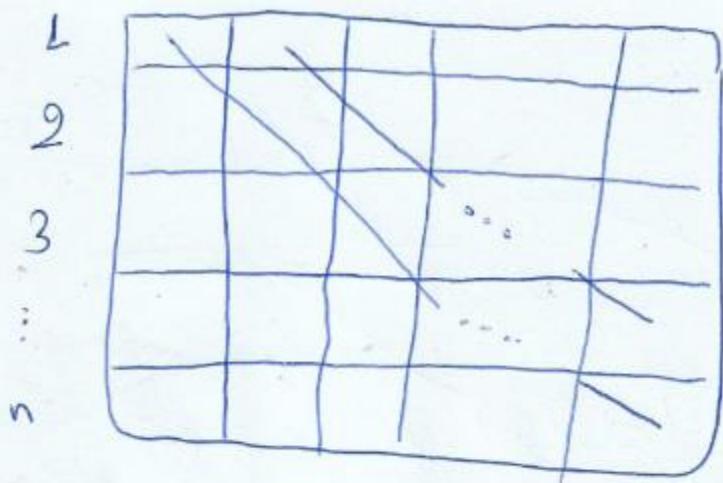
$$t = j - L - i + L = j - i$$

3]

$$m(i,j) = \begin{cases} \min_{i \leq t \leq j-1} \{ k_{i-1} * k_t * k_j + m(i,t) + m(t+1,j) \}, & \text{av } 1 \leq i \leq j \leq n \\ 0, & \text{av } i=j \end{cases}$$



L 2 3 ... n



5] $A(L, n)$

... Extensible documents \rightarrow Documents Programmable
σελ. 18-20

$$\begin{aligned} A^{(1,5)} &= A^{(1,3)} \cdot A^{(4,5)} \\ &\downarrow \\ &= (A^{1,1} \cdot A^{2,3}) \cdot (A_4 \cdot A_5) \\ &= (A_1 \cdot A^{2,3}) \cdot (A_4 \cdot A_5) \\ &= (A_1 \cdot (A_2 \cdot A_3)) \cdot (A_4 \cdot A_5) \\ &= (A_1 \cdot (A_2 \cdot A_3)) \cdot (A_4 \cdot A_5) \end{aligned}$$

12a. Μεγιστοποίηση κέρδους μιας εταιρίας εστίασης από τις τοποθεσίες εγκατάστασης εστιατορίων

Διαφάνειες

Slides_and_Exercises→09_DynamicExer.pdf σελ. 11-12

Εκφώνηση

Άσκηση 5 Μια εταιρεία εστίασης μπορεί να ανοίξει μία σειρά από εστιατόρια κατά μήκος μίας διαδρομής. Οι n πιθανές τοποθεσίες σχηματίζουν μία ευδεία γραμμή και είναι γνωστές οι αποστάσεις από την αρχή της διαδρομής μέχρι την κάθε τοποθεσία καθώς και το κέρδος από το άνοιγμα ενός εστιατορίου σε κάθε τοποθεσία. Οι περιορισμοί είναι οι εξής.

- Σε κάθε τοποθεσία μπορεί να ανοίξει το πολύ ένα εστιατόριο.
- Δύο εστιατόρια θα πρέπει να απέχουν τουλάχιστον κατά μία συγκεκριμένη απόσταση D ανάμεσα τους.

Να διατυπώσετε αλγόριθμο ο οποίος να αποφασίζει τις τοποθεσίες στις οποίες θα πρέπει η εταιρεία να ανοίξει τα εστιατόρια για να μεγιστοποιήσει το κέρδος της.

1. Ορισμός υποπροβλημάτων

Λύση Έστω ότι οι τοποθεσίες αριθμούνται από το 1 ως το n και το κέρδος από το άνοιγμα ενός εστιατορίου στη θέση j είναι p_j . Παριστάνουμε με d_j την απόσταση της τοποθεσίας j από την αρχή της διαδρομής. Ορίζουμε σαν $P(j)$ το συνολικό κέρδος που έχει η εταιρεία από το άνοιγμα των εστιατορίων από τη θέση 1 μέχρι και την θέση j.

2. Καθορισμός επιλογών

Είναι καθαρό ότι αν επιλεγεί η θέση j τότε δεν πρέπει να έχει επιλεγεί κάποια θέση που βρίσκεται σε απόσταση πριν τη θέση αυτή σε απόσταση μικρότερη από D. Επομένως η εταιρεία έχει δύο επιλογές: α) να μην ανοίξει εστιατόριο στη θέση j, ή, β) να ανοίξει. Στην πρώτη περίπτωση $P(j) = P(j - 1)$ ενώ στη δεύτερη έχουμε $P(j) = p_j + P(j')$, όπου j' είναι η εγγύτερη προηγούμενη θέση από τη θέση j με $d_j - d_{j'} \geq D$.

3. Ορισμός αναδρομικής σχέσης

Η αναδρομική συνάρτηση κέρδους γράφεται

$$P(j) = \begin{cases} \max\{p_j + P(j'), P(j - 1)\}, & \text{όπου } j' = \max\{i : d_j - d_i \geq D\}, j \in \{1, \dots, n\}, \\ 0, & j = 0, \end{cases}$$

4. Τοπολογική ταξινόμηση υποπροβλημάτων

for $i=1$ to n
 solve $S(:i)$ $\Leftrightarrow S(1) \rightarrow S(2) \rightarrow \dots \rightarrow S(n)$

5. Επίλυση αρχικού προβλήματος

Το ζητούμενο είναι το $P(n)$.

Ιδέα

Ο Αλγόριθμος 5 υπολογίζει τα $P(j)$ με βάση τον προηγούμενο τύπο. Παρατηρήστε ότι πριν τον υπολογισμό αυτό, γίνεται υπολογισμός των j' (πίνακας $prev[]$ - γραμμές 2-8).

Αλγόριθμος

Algorithm 5 Εστιατόρια κατά μήκος μίας διαδρομής

```

1: void locations(int n, int p[], int d[], int D)
2: for  $j \leftarrow 1; j \leq n; j++$  do
3:    $i \leftarrow j - 1;$ 
4:   while  $d[j] - d[i] < D$  and  $i > 0$  do
5:      $i \leftarrow i - 1;$ 
6:   end while
7:    $previous[j] = i;$ 
8: end for
9: for  $j \leftarrow 1; j \leq n; j++$  do
10:    $P[j] = \max\{p_j + P[previous[j]], P[j - 1]\};$ 
11: end for

```

Πολυπλοκότητα

Ο αλγόριθμος για να υπολογίσει το j' εκτελεί το πολύ $j-1$ αφαιρέσεις, $j-1$ συγκρίσεις

(για τον υπολογισμό $d_j - d_i \geq D$ και $j-2$ συγκρίσεις για να βρει την μεγαλύτερη τιμή i). Άρα συνολικά εκτελούνται $3j-4$ στοιχειώδεις πράξεις ΣΤΗ ΧΕΙΡΟΤΕΡΗ ΠΕΡΙΠΤΩΣΗ για τον υπολογισμό του j' . Στη συνέχεια για τον υπολογισμό $P(j)$ γίνεται μία σύγκριση και μία πρόσθεση. Άρα ο αριθμός των στοιχειωδών πράξεων φράζεται από το επάνω από την

$$\sum_{j=1}^n 3j - 2 = O(n^2).$$



5/12/2023

Chapter 12a

Μεροτονοίηση κέρδους μετα επαγγέλτων
εστιών ανά την τοποθεσία εγκαταστάσης εστιών

$D \rightarrow H$ ανάσταση
ανάπτυξη ανά
εστιώνες

2

$p_i \rightarrow$ το κέρδος
των εστιών
μετα εγκατάστασης i

$d_i \rightarrow$ η απόσταση
της τοποθεσίας i ανά
την απεριφύλακτη

1] $S(:i)$ →

Προθεματική
Περιγραφή 1 και να καταλήξει στη δεσμή i

2] S_i → Είτε ανοίγει εστιώρω στη δεσμή i

$$S_i = p_i + S_j, \quad j < i, \quad d_i - d_j \geq D$$

Είτε δεν ανοίγει εστιώρω στη δεσμή i

$$S_i = S_{i-1}$$

$S_i \rightarrow$ Η λίστα των αντιστοχών υποπεριφύλακτων

$$S_i = \begin{cases} \max\{S_{i-1}, p_i + \max_j \{ S_j : j < i, d_i - d_j \geq D \}\}, \\ 0 \end{cases} \quad 2 \leq i \leq n, \quad i=1$$

4] for $i=1$ to n
solve $S(:i)$ $\Leftrightarrow (S(1)) \rightarrow (S(2)) \rightarrow \dots \rightarrow (S(n))$

5] S_n

Πολυπλοκότητα: $O(n^2)$

Εκτελούνται $i-1$ ανεμόσις } $\Rightarrow d_j - d_i \geq D$
-/-/- $i-1$ συκείσις }

-/-/- $i-2$ συκείσις } $\Rightarrow \max\{s_{i-2} \dots\}$

$$\sum_{i=1}^n z_{i-2} = O(n^2) \quad O \text{ αριθμός των ΣΥΒ}$$

Δ.Π.

Σε πρόβλημα που μως γνωστάται πολυπλοκότητα

- Θα πρέπει να ανανείψει πώς οι ΣΥΒ εκτελούνται
σε καθέτη υπορόβλημα

12b. Μέγιστη κοινή υπακολούθια

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 100-103

Πρόβλημα

Μια συμβολοσειρά $W = \langle w_1, w_2, \dots, w_k \rangle$ αποτελεί μία ακολουθία k συμβόλων (χαρακτήρων) όπου το πρώτο σύμβολο είναι το w_1 , το δεύτερο το w_2 , κοκ. Μια υπακολούθια W' της W είναι μία συμβολοσειρά η οποία προκύπτει από την W αν διαγραφούν από αυτή ένας οποιοσδήποτε αριθμός χαρακτήρων. Για παράδειγμα, αν $W = \langle X, A, I, P, E, T, I, S, M, A, T, A \rangle$, τότε οι συμβολοσειρές $\langle A, I, M, A \rangle$, $\langle P, E, M, A \rangle$, $\langle A, T, A \rangle$, $\langle A, P, I, A \rangle$ αποτελούν υποακολουθίες της. Από το παράδειγμα είναι φανερό ότι οι χαρακτήρες μίας υπακολούθιας πρέπει να εμφανίζονται με την ίδια σειρά στην ακολουθία αλλά όχι απαραίτητα συνεχόμενα. Το μήκος μιας υπακολούθιας είναι ο αριθμός των χαρακτήρων που περιέχει.

Το πρόβλημα με το οποίο θα ασχοληθούμε ορίζεται ως εξής: δεδομένων δύο ακολουθιών $X = \langle x_1, x_2, \dots, x_n \rangle$ και $Y = \langle y_1, y_2, \dots, y_m \rangle$ θέλουμε να βρούμε την κοινή υπακολούθια μέγιστου μήκους. Εκτός των άλλων, το πρόβλημα αυτό βρίσκει εφαρμογή στον τομέα της γενετικής όπου δεδομένων δυο σειρών DNA, καθένα από τα οποία περιγράφει έναν οργανισμό, θέλουμε να καθορίσουμε το βαθμό ομοιότητας τους.

1. Ορισμός υποπροβλημάτων

1. Για να ορίσουμε τα υποπροβλήματα, θα χρησιμοποιήσουμε τη λογική των επιθεμάτων. Έστω $X(i :) = \langle x_i, x_{i+1}, \dots, x_n \rangle$ και $Y(j :) = \langle y_j, y_{j+1}, \dots, y_m \rangle$. Συμβολίζουμε με $Z(i : , j :)$ το υποπρόβλημα εύρεσης του μήκους της μεγαλύτερης κοινής υπακολουθίας των συμβολοσειρών $X(i :), Y(j :)$, όπου $1 \leq i \leq n$, $1 \leq j \leq m$. Θα συμβολίζουμε με $L_{i,j}$ τη λύση του προβλήματος αυτού.

2. Καθορισμός επιλογών

2. Όταν θεωρούμε το υποπρόβλημα $Z(i : , j :)$ υπάρχουν δύο περιπτώσεις: είτε $x_i \neq y_j$ ή $x_i = y_j$. Στην πρώτη περίπτωση υπάρχουν δύο επιλογές: είτε το μήκος της μεγαλύτερης κοινής υπακολουθία αποτελεί λύση του υποπροβλήματος $Z(i, j + 1)$ ή του $Z(i + 1, j)$. Στη δεύτερη περίπτωση δεν υπάρχει επιλογή: αφού ταιριάζουν οι χαρακτήρες x_i και y_j μένει να επιλυθεί το υποπρόβλημα $Z(i + 1, j + 1)$.

3. Ορισμός αναδρομικής σχέσης

3. Στην πρώτη περίπτωση που εξετάσαμε προηγουμένως ($x_i \neq y_j$) υπάρχουν δύο επιλογές σε σχέση με την τιμή της $L_{i,j}$ και άρα θα πάρουμε την καλύτερη. Μαθηματικά, $L_{i,j} = \max\{L_{i+1,j}, L_{i,j+1}\}$. Στην δεύτερη περίπτωση ($x_i = y_j$), το μήκος της μεγαλύτερης κοινής συμβολοσειράς ισούται με την τιμή $L_{i+1,j+1}$ - λύση του υποπροβλήματος $Z(i+1, j+1)$ - επαυξημένη κατά 1. Επίσης αν έχουν εξεταστεί όλοι οι χαρακτήρες της μίας από τις δύο συμβολοσειρές - περίπτωση κατά την οποία είτε $i = n+1$, ή $j = m+1$ - δεν υπάρχουν επιπλέον χαρακτήρες που να ανήκουν σε κοινή συμβολοσειρά. Άρα θέτουμε $L_{n+1,j} = 0$, $j = 1, \dots, m+1$, και $L_{i,m+1} = 0$, $i = 1, \dots, n+1$. Συνολικά,

$$L_{i,j} = \begin{cases} 0, & i = n+1 \text{ ή } j = m+1, \\ 1 + L_{i+1,j+1}, & x_i = y_j, \\ \max\{L_{i+1,j}, L_{i,j+1}\}, & x_i \neq y_j. \end{cases} \quad (14.14)$$

4. Τοπολογική ταξινόμηση υποπροβλημάτων

4. Επειδή ακολουθήθηκε η θεώρηση των υποπροβλημάτων με τη λογική των επιθεμάτων η σειρά επίλυσης γίνεται με τους δείκτες i, j να αρχικοποιούνται στις τιμές n, m , αντίστοιχα, και να βαίνουν μειωμένοι. Δηλαδή η σειρά που θα επιλυθούν τα υποπροβλήματα είναι

$$\begin{aligned} Z(n, m) &\rightarrow Z(n, m-1) \rightarrow \cdots \rightarrow Z(n, 1) \rightarrow Z(n-1, m) \rightarrow \\ &\cdots \rightarrow Z(n-1, 1) \rightarrow Z(n-2, m) \rightarrow \cdots \rightarrow Z(1, 1). \end{aligned}$$

5. Επίλυση αρχικού προβλήματος

5. Το μήκος της μεγαλύτερης κοινής υπακολουθίας είναι η τιμή $L_{1,1}$. Η υπακολουθία μπορεί να ιχνηλατηθεί αν η επιλογή που γίνεται κάθε φορά για τον υπολογισμό της λύσης από την (14.14), καταχωρείται σε ένα δισδιάστατο διάνυσμα p . Για να μπορέσουμε να εκφράσουμε τη λύση σαν ένα μονοπάτι σε γράφημα θα θεωρήσουμε ότι το στοιχείο $p_{i,j}$ θα παίρνει τις παρακάτω τιμές

$$p_{i,j} = \begin{cases} '↖', & \text{av } L_{i,j} = 1 + L_{i+1,j+1}, \\ '↑', & \text{av } L_{i,j} = L_{i+1,j}, \\ '←', & \text{av } L_{i,j} = L_{i,j+1}. \end{cases}$$

Παράδειγμα

Παράδειγμα 80. Θεωρούμε τις ακολουθίες $=<\Pi, O, \Lambda, \Lambda, A>$ και $=<P, O, \Lambda, A>$. Η μεγαλύτερη κοινή υπακολουθία μέσω ΔΠ προκύπτει από τους υπολογισμούς:

$$L_{5,4} = 1 + L_{6,5} = 1 + 0 = 1, \quad p_{5,4} = '↖',$$

$$L_{5,3} = \max\{L_{6,3}, L_{5,4}\} = \max\{0, 1\} = 1 \quad p_{5,3} = '←',$$

$$L_{5,2} = \max\{L_{6,2}, L_{5,3}\} = \max\{0, 1\} = 1 \quad p_{5,2} = '←',$$

$$L_{5,1} = \max\{L_{6,1}, L_{5,2}\} = \max\{0, 1\} = 1 \quad p_{5,1} = '←',$$

$$L_{4,4} = \max\{L_{5,4}, L_{4,5}\} = \max\{1, 0\} = 1, \quad p_{4,4} = '↑',$$

$$L_{4,3} = 1 + L_{5,4} = 1 + 1 = 2 \quad p_{4,3} = '↖',$$

$$L_{4,2} = \max\{L_{5,2}, L_{4,3}\} = \max\{1, 2\} = 2 \quad p_{4,2} = '←',$$

$$L_{4,1} = \max\{L_{5,1}, L_{4,2}\} = \max\{1, 2\} = 2 \quad p_{4,1} = '←',$$

$$L_{3,4} = \max\{L_{4,4}, L_{3,5}\} = \max\{1, 0\} = 1, \quad p_{3,4} = '↑',$$

$$L_{3,3} = 1 + L_{4,4} = 1 + 1 = 2 \quad p_{3,3} = '↖',$$

$$L_{3,2} = \max\{L_{4,2}, L_{3,3}\} = \max\{2, 2\} = 2 \quad p_{3,2} = '↑' \text{ ή } p_{3,2} = '←',$$

$$L_{3,1} = \max\{L_{4,1}, L_{3,2}\} = \max\{2, 2\} = 2 \quad p_{3,1} = '↑' \text{ ή } p_{3,1} = '←',$$

$$L_{2,4} = \max\{L_{3,4}, L_{2,5}\} = \max\{1, 0\} = 1, \quad p_{2,4} = '↑',$$

$$L_{2,3} = \max\{L_{3,3}, L_{2,4}\} = \max\{2, 1\} = 2 \quad p_{2,3} = '↑',$$

$$L_{2,2} = 1 + L_{3,3} = 1 + 2 = 3 \quad p_{2,2} = '↖',$$

$$L_{2,1} = \max\{L_{3,1}, L_{2,2}\} = \max\{2, 3\} = 3 \quad p_{2,1} = '←',$$

$$\begin{aligned}
 L_{1,4} &= \max\{L_{2,4}, L_{1,5}\} = \max\{1, 0\} = 1, & p_{1,4} &= '↑', \\
 L_{1,3} &= \max\{L_{2,3}, L_{1,4}\} = \max\{2, 1\} = 2 & p_{1,3} &= '↑' \\
 L_{1,2} &= \max\{L_{2,2}, L_{1,3}\} = \max\{3, 2\} = 3 & p_{1,2} &= '↑', \\
 L_{1,1} &= \max\{L_{2,1}, L_{1,2}\} = \max\{3, 3\} = 3 & p_{1,1} &= '↑' \text{ ή } p_{1,1} = '←'.
 \end{aligned}$$

Στον Πίνακα 14.6 αποτυπώνεται σε κάθε κελί το μήκος της μέγιστης κοινής υπακολουθίας του αντίστοιχου υποπροβλήματος. Επίσης σε κάθε κελί έχει απεικονιστεί και το αντίστοιχο στοιχείο του πίνακα p το οποίο δείχνει τη λύση του (προηγούμενου) υποπροβλήματος η οποία χρησιμοποιήθηκε προκειμένου να προκύψει η λύση του συγκεκριμένου κελιού. Ακολουθώντας σε ανάποδη φορά από τα βέλη το μονοπάτι που καταλήγει στο κελί της πρώτης γραμμής και πρώτης στήλης μπορούμε να βρούμε την κοινή συμβολοσειρά: κάθε γράμμα της υποδεικνύεται από ένα από τα «διαγώνια βέλη» του μονοπατιού. Στον Πίνακα 14.6 υπάρχον δύο μονοπάτια που καταλήγουν στο κελί της πρώτης γραμμής, πρώτης στήλης τα οποία εκκινούν από το κελί της γραμμής $n = 5$, στήλης $m = 4$ (κόκκινες ακμές στον πίνακα). Και στα δύο μονοπάτια οι διαγώνιες ακμές είναι ίδιες και υποδεικνύουν την υπακολουθία $< O, \Lambda, A >$ με πλήθος χαρακτήρων $L_{1,1} = 3$.

	P	O	Λ	A
P	3 	3 	2 	1 
O	3 	3 	2 	1 
Λ	2 	2 	2 	1 
Λ	2 	2 	2 	1 
A	1 	1 	1 	1 

Πίνακας 14.6: Μέγιστη κοινή υπακολουθία - Παράδειγμα 80

Πολυπλοκότητα

Η πολυπλοκότητα της μεθόδου μπορεί να υπολογιστεί πολύ εύκολα: ο αριθμός των υποπροβλημάτων είναι $n \cdot m$ και η επίλυση καθενός από αυτά απαιτεί σταθερό αριθμό ΣΥΒ. Επομένως η διαδικασία είναι τάξης $\Theta(n \cdot m)$.

{ Μεριστη Κωνή Υπακολουθία }

Chapter 12b

$$X = \langle x_1, x_2, \dots, x_n \rangle \quad \left\{ \begin{array}{l} X, V \rightarrow \text{ακολούθες} \\ x_1, y_1 \rightarrow \text{γράμμα} \end{array} \right.$$

$$Y = \langle y_1, y_2, \dots, y_m \rangle \quad \left\{ \begin{array}{l} X, V \rightarrow \text{ακολούθες} \\ x_1, y_1 \rightarrow \text{γράμμα} \end{array} \right.$$

1

$$x_i: \langle x_1, \dots, x_n \rangle, 1 \leq i \leq n$$

$$y_j: \langle y_1, \dots, y_m \rangle, 1 \leq j \leq m$$

$$\underline{Z(i:, j:) \rightarrow \frac{\text{Ξεκινώ} \text{ αώ τη δεση } i \text{ και καταλήγει}}{\sigma_{ij} \text{ δεση } n}}$$

Ενδεματική
Περιγραφή

$$\frac{\text{Ξεκινώ} \text{ αώ τη δεση } j \text{ και καταλήγει}}{\sigma_{ij} \text{ δεση } m}$$

$L_{i,j} \rightarrow$ To μήκος της μέγιστης κανής διακόλωσίς των υποδοχήματος $Z(i:, j:)$

2]

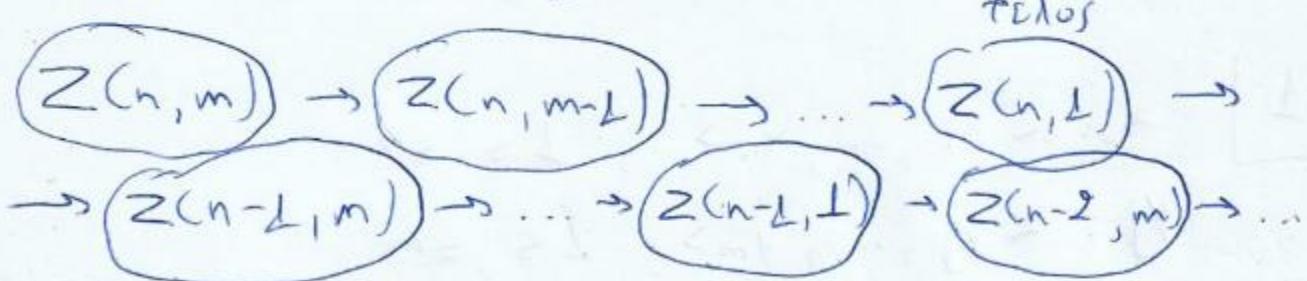
$$\text{Ενδημία } \begin{cases} x_i = y_j, L_{i,j} = L + L_{i+1, j+1} \\ x_i \neq y_j, L_{i,j} = \max\{L_{i+1, j}, L_{i, j+1}\} \end{cases}$$

$$x_i \neq y_j, L_{i,j} = \max\{L_{i+1, j}, L_{i, j+1}\}$$

3]

$$L_{i,j} = \begin{cases} L + L_{i+1, j+1}, & x_i = y_j \\ \max\{L_{i+1, j}, L_{i, j+1}\}, & x_i \neq y_j \\ 0 & , i=n+1 \text{ ή } j=m+1 \end{cases}$$

4] Ενδημία είναι ειδεπευκάνη η οποία ανήκει στο



5] $L_{1,1}$

$$Y = \langle P \circ \Lambda A \rangle \quad m = 4$$

$$X = \langle \cap O \wedge \wedge A \rangle \quad n = 5$$

Equivalent ans to Texos

	i	j	P	O	\wedge	A	$x_i = y_j$
\cap			3	3	2↑	L↑	
O			3	3	2↑	L↑	
\wedge			2	2	2↑	L↑	
\wedge			2	2	2↑	L↑	
A			L	L	L	L↑	

parent array

$$p_{i,j} = \begin{cases} 'R' & , \text{ or } L_{i,j} = L_{i+1,j} + L_{i+2,j+1}, x_i = y_j \\ 'U' & , \text{ or } L_{i,j} = L_{i+1,j}, x_i \neq y_j \\ 'L' & , \text{ or } L_{i,j} = L_{i,j+2}, x_i \neq y_j \end{cases}$$

5] Eniλvov
dexku
nqo6λiμatos

$$L_{1,3} = 3$$

$$\langle O, \wedge, A \rangle$$

$$P_{5,4} = 'R'$$

$$L_{5,4} = L + L_{5,6} = L + 0 = L$$

$$L_{5,3} = \max\{L_{6,3}, L_{5,4}\}$$

$$= \max\{0, L\} \stackrel{P_{5,3} = 'L'}{=} L$$

...

$$L_{4,4} = \max\{L_{5,4}, L_{4,5}\}$$

$$= \max\{L, 0\} = L$$

$$P_{4,4} = 'U'$$

...

$$P_{4,2} = 'L'$$

$$L_{4,2} = \max\{L_{5,2}, L_{4,3}\}$$

$$= \max\{1, 2\} = 2$$

$$P_{4,1} = 'L'$$

$$L_{4,1} = \max\{L_{5,1}, L_{4,2}\}$$

$$= \max\{L, 2\} = 2$$

13a. Το πρόβλημα του διαρρήκτη (Knapsack integer problem)

Διαφάνειες

Slides_and_Exercises→09_DynamicExer.pdf σελ. 12-14, 15

Εκφώνηση

Άσκηση 6 Κατά τη διάρκεια μίας διάρρηξης, ένας κλέφτης βρίσκει περισσότερα αντικείμενα αξίας από αυτά που μπορεί να κουβαλήσει. Έστω n , ο αριθμός των αντικειμένων. Ο κλέφτης μπορεί να εκτιμήσει για κάθε αντικείμενο το βάρος και την αξία του. Ποια αντικείμενα πρέπει να πάρει μαζί του ο κλέφτης προκειμένου να μεγιστοποιήσει την συνολική αξία αυτών που θα κουβαλήσει.

1. Ορισμός υποπροβλημάτων

Σε οποιαδήποτε

περίπτωση μπορούμε να θεωρήσουμε ότι το βάρος που μπορεί να σηκώσει ο κλέφτης είναι W . Έστω ότι τα αντικείμενα (είδη των αντικειμένων) αριθμούνται από το $1, \dots, n$ και κάθε αντικείμενο i έχει βάρος w_i και αξία v_i .

Στην περίπτωση αυτή θεωρούμε κάθε ακέραια τιμή $w \in W$. Έστω $K(w)$ η μέγιστη συνολική αξία των αντικειμένων αν το βάρος που μπορεί να σηκώσει ο κλέφτης είναι w .

2. Καθορισμός επιλογών

Αν το $K(w)$ έχει σχηματιστεί χωρίς να περιέχεται το αντικείμενο i , μπορούμε να το προσθέσουμε αν $w_i \leq w$. Τότε η μέγιστη συνολική αξία γίνεται $K(w - w_i) + v_i$. Σε κάθε βήμα λοιπόν επιλέγουμε να προσθέσουμε το αντικείμενο i το οποίο μεγιστοποιεί την παράσταση αυτή.

3. Ορισμός αναδρομικής σχέσης

Ο τελικός αναδρομικός

τύπος είναι

$$K(w) = \begin{cases} \max\{K(w - w_i) + v_i : i \in \{1, \dots, n\}, w_i \leq w\}, & w \in \{1, \dots, W\}, \\ 0, & w = 0. \end{cases} \quad (5)$$

4. Τοπολογική ταξινόμηση υποπροβλημάτων

for $w=1 ; w \leq W ; w++$
λύσε $K(:w)$

5. Επίλυση αρχικού προβλήματος

$K(W)$

Παράδειγμα

i	1	2	3	4
wt_i	6	3	4	2
v_i	30	14	16	1

Πίνακας 7: Παράδειγμα για το πρόβλημα του σακιδίου

Παράδειγμα 3 Έστω $W = 8$. Το βάρος και η αξία τεσσάρων αντικειμένων απεικονίζονται στον Πίνακα 7.

$$w = 0 : K(0) = 0.$$

$w = 1 : K(1) = 0$, αφού κανένα αντικείμενο δεν έχει βάρος 1.

$$w = 2 :$$

$$K(2) = \max\{K(2 - wt_4) + v_4\} = 1.$$

$$w = 3 :$$

$$K(3) = \max\{K(3 - wt_2) + v_2, K(3 - wt_4) + v_4\} = \max\{0 + 14, 1 + 1\} = 14.$$

$$w = 4 :$$

$$\begin{aligned} K(4) &= \max\{K(4 - wt_2) + v_2, K(4 - wt_3) + v_3, K(4 - wt_4) + v_4\} \\ &= \max\{K(4 - 3) + 14, K(4 - 4) + 16, K(4 - 2) + 1\} \\ &= \max\{0 + 14, 0 + 16, 1 + 1\} = 16. \end{aligned}$$

$$w = 5 :$$

$$\begin{aligned} K(5) &= \max\{K(5 - wt_2) + v_2, K(5 - wt_3) + v_3, K(5 - wt_4) + v_4\} \\ &= \max\{K(5 - 3) + 14, K(5 - 4) + 16, K(5 - 2) + 1\} \\ &= \max\{K(2) + 14, K(1) + 16, K(3) + 1\} \\ &= \max\{1 + 14, 0 + 16, 14 + 1\} = 16. \end{aligned}$$

$$w = 6 :$$

$$\begin{aligned} K(6) &= \max\{K(6 - wt_1) + v_6, K(6 - wt_2) + v_2, K(6 - wt_3) + v_3, K(6 - wt_4) + v_4\} \\ &= \max\{K(6 - 6) + 30, K(6 - 3) + 14, K(6 - 4) + 16, K(6 - 2) + 1\} \\ &= \max\{K(0) + 30, K(3) + 14, K(2) + 16, K(4) + 1\} \\ &= \max\{0 + 30, 14 + 14, 1 + 16, 16 + 1\} = 30. \end{aligned}$$

$$w = 7 :$$

$$\begin{aligned} K(7) &= \max\{K(7 - wt_1) + v_6, K(7 - wt_2) + v_2, K(7 - wt_3) + v_3, K(7 - wt_4) + v_4\} \\ &= \max\{K(7 - 6) + 30, K(7 - 3) + 14, K(7 - 4) + 16, K(7 - 2) + 1\} \\ &= \max\{K(1) + 30, K(4) + 14, K(3) + 16, K(5) + 1\} \\ &= \max\{0 + 30, 16 + 14, 14 + 16, 16 + 1\} = 30. \end{aligned}$$

$$w = 8 :$$

$$\begin{aligned} K(8) &= \max\{K(8 - wt_1) + v_6, K(8 - wt_2) + v_2, K(8 - wt_3) + v_3, K(8 - wt_4) + v_4\} \\ &= \max\{K(8 - 6) + 30, K(8 - 3) + 14, K(8 - 4) + 16, K(8 - 2) + 1\} \\ &= \max\{K(2) + 30, K(5) + 14, K(4) + 16, K(6) + 1\} \\ &= \max\{1 + 30, 16 + 14, 16 + 16, 30 + 1\} = 32. \end{aligned}$$

Επομένως η μέγιστη αξία των αντικειμένων που μπορεί να μεταφέρει ο κλέφτης είναι $K(8) = 32$ και επιτυγχάνεται με δύο αντικείμενα τύπου 3.

Ιδέα

Ο Αλγόριθμος 6 υλοποιεί την διαδικασία. Η διαδικασία επιστρέφει στο όνομα της την μεγαλύτερη αξία και ο πίνακας $x[]$ τα είδη που αντιστοιχούν στην αξία αυτή.

Αλγόριθμος

Algorithm 6 Υπολογισμός σακιδίου - έκδοση με επανάληψη

```
1: int KnapsackRep(int n, int W, int v[], int wt[], int x[])
2: for  $w \leftarrow 0$ ;  $w \leq W$ ;  $w + +$  do
3:    $xweight[w] \leftarrow 0$ ;
4: end for
5: for  $i \leftarrow 0$ ;  $i \leq n$ ;  $i + +$  do
6:    $x[i] \leftarrow 0$ ;
7: end for
8:  $K[0] \leftarrow 0$ ;
9: for  $w \leftarrow 1$ ;  $w \leq W$ ;  $w + +$  do
10:    $max\_val \leftarrow -\infty$ ;  $max\_index \leftarrow 0$ ;
11:   for  $i \leftarrow 1$ ;  $i \leq n$ ;  $i + +$  do
12:     if  $wt[i] \leq w$  then
13:       if  $max\_val < K[w - wt[i]] + v[i]$  then
14:          $max\_val \leftarrow K[w - wt[i]] + v[i]$ ;  $max\_index \leftarrow i$ ;
15:       end if
16:     end if
17:   end for
18:    $K[w] \leftarrow max\_val$ ;  $xweight[w] \leftarrow max\_index$ ;
19: end for
20:  $w = W$ 
21: while  $w > 0$  do
22:    $x[xweight[w]] + +$ ;  $w \leftarrow w - wt[xweight[w]]$ 
23: end while
24: return  $K[W]$ ;
```

Πολυπλοκότητα

Ο Αλγόριθμος 6 είναι τάξης $O(n \cdot W)$ (γραμμές 9-19).

(ΑΠ)

17/12/2023

Chapter 13a

To πρόβλημα των Knapsack integer problem

i	1	2	3	4	→ αντικείμενα
w _{ti}	6	3	4	2	→ βάρος σακιδίου ή kg
v _i	30	14	16	1	→ αξία σακιδίου ή €

i	1	2	3	4	
w _{ti}	1	1	1	1	=> €/kg
v _i	5	4,6	4	0,5	

Προϊστορίας: 1) Πρίντε να λέμε όλα τα διαθέσιμα κύλια των σακιδίων

2) Μαρτίν να λέμε η αρχή το πώς σακίδιων $0 \leq w \leq W$ → W τα μεγάλα με μεριά να σημαίνει

1] $K(:w)$ → οι λύσεις των υποπρόβλημάτων που καταλήγουν σε w κύλια.

$K(w)$ → η γεναλίτερη αξία σακιδίου με μεριά να λέμε μέχρι w κύλια

2] Είστε $i \in \{1, \dots, n\}$ με τη ρυθμίση $w_{ti} \leq w$

3

$$k(w) = \begin{cases} \max \{ v_i + k(w - w_{t,i}) : 1 \leq i \leq n, w_{t,i} \leq w \}, & w \geq w \geq L \\ 0, & w=0 \end{cases}$$

4 for $w=L; w \leq W; w++$
 $\lambda \in \mathbb{C} \setminus K(:w)$

5 $k(w)$

Πολυπλοκότητα: $O(W+n)$

$W \rightarrow$ αριθμός υπορρόφησηών

n αρχαιώσεις και n αναγορέωσις

$$w=0 : k(0) = 0$$

$$w=L : k(L) = 0$$

$$p_2 = 4$$

$$w=2 : k(2) = \max \{ v_4 + k(2-2) \} = L+0 = L, \quad \text{REDACTED}$$

$$\begin{aligned} w=3 : k(3) &= \max \{ v_2 + k(3-3), v_4 + k(3-2) \} \\ &= \max \{ 14+0, L+0 \} = 14, \quad \text{REDACTED} \end{aligned}$$

$$\begin{aligned} w=4 : k(4) &= \max \{ v_2 + k(4-3), v_3 + k(4-4), v_4 + k(4-2) \} \\ &= \max \{ 14+0, 16+0, L+L \} = 16 \end{aligned}$$

$$p_4 = 3$$

$$W=5 : K(5) = \dots = 16, P_5 = 3$$

$$W=6 : K(6) = \dots = 30, P_6 = 1$$

$$W=7 : K(7) = \dots = 30, P_7 = 1, 2, 3$$

$$W=8 : K(8) = \dots = 32, P_8 = 3$$

Αριθμός στα 8 κελιά για μέρηση αφού ωριμός 32 €

Βλέπω στην $P_8 = 3$ αριθμό των ημερών...

i	1	2	3	4
W_{ti}	6	3	4	2
V_i	30	14	16	1

Να προσθέτω το ανταρκτικόν $i=3$ και στην διάρκεια $W_{ti}=4$

($W=8-4=4$). Ο γραμμής στο $P_4=3$, με την οποία θα πρέπει να προσθέτω το ανταρκτικόν 3 ($W=4-4=0$). Ο γραμμής στο $P_0=0$.

Άριθμός των ανταρκτικών και μερών είναι $(3,3)$

13b. Το πρόβλημα των διαρρήκτη (Knapsack 0-1 problem)

Διαφάνειες

Slides_and_Exercises→09_DynamicExer.pdf σελ. 14, 16-18

Εκφώνηση

Άσκηση 6 Κατά τη διάρκεια μίας διάρρηξης, ένας κλέφτης βρίσκει περισσότερα αντικείμενα αξίας από αυτά που μπορεί να κουβαλήσει. Έστω n , ο αριθμός των αντικειμένων. Ο κλέφτης μπορεί να εκτιμήσει για κάθε αντικείμενο το βάρος και την αξία του. Ποια αντικείμενα πρέπει να πάρει μαζί του ο κλέφτης προκειμένου να μεγιστοποιήσει την συνολική αξία αυτών που θα κουβαλήσει.

1. Ορισμός υποπροβλημάτων

Σε οποιαδήποτε

περίπτωση μπορούμε να θεωρήσουμε ότι το βάρος που μπορεί να σηκώσει ο κλέφτης είναι W . Έστω ότι τα αντικείμενα (είδη των αντικειμένων) αριθμούνται από το $1, \dots, n$ και κάθε αντικείμενο i έχει βάρος w_i και αξία v_i .

ορίζουμε $K(w, i)$ τη

μέγιστη αξία η οποία επιτυγχάνεται με τη χρήση ενός σακιδίου χωρητικότητας w με είδη $1, \dots, i$.

2. Καθορισμός επιλογών

Η τιμή $K(w, i)$ είτε επιτυγχάνε-

ται με τη χρήση του αντικειμένου i είτε όχι. Στην πρώτη περίπτωση έχουμε

$K(w, i) = K(w - w_i, i - 1) + v_i$, ενώ στη δεύτερη $K(w, i) = K(w, i - 1)$.

3. Ορισμός αναδρομικής σχέσης

δρομική συνάρτηση, για $w \in \{1, \dots, W\}$, $i \in \{1, \dots, n\}$, έχει τη μορφή

$$K(w, i) = \begin{cases} 0, & w = 0 \text{ ή } i = 0, \\ K(w, i - 1), & w t_i > w, \\ \max\{K(w - w_i, i - 1) + v_i, K(w, i - 1)\} & w t_i \leq w. \end{cases}$$

4. Τοπολογική ταξινόμηση υποπροβλημάτων

for $w = L$; $w \leq W$; $w++$

for $i = L$; $i \leq n$; $i++$

Λύσε $K(w, i)$

5. Επίλυση αρχικού προβλήματος

Προφανώς η βέλτιστη λύση δίνεται από το $K(W, n)$.

Παράδειγμα

Το παράδειγμα επιλύεται

στον Πίνακα 8. Η μεγαλύτερη αξία των πραγμάτων που μπορεί να μεταφέρει ο κλέφτης είναι 31 και προκύπτει από την επιλογή των αντικειμένων 1 και 4.

	i	1	2	3	4
w					
1		0	0	0	0
2		0	0	0	$\max\{K(2 - wt_4, 3) + v_4, K(2, 3)\}$ = $\max\{K(2 - 2, 3) + 1, 0\}$ = $\max\{0 + 1, 0\} = 1$
3		0	$\max\{K(3 - wt_2, 1) + v_2, K(3, 1)\}$ = $\max\{K(3 - 3, 1) + 14, 0\}$ = $\max\{0 + 14, 0\} = 14$	$K(3, 2) = 14$	$\max\{K(3 - wt_4, 3) + v_4, K(3, 4)\}$ = $\max\{K(3 - 2, 3) + 1, 14\}$ = $\max\{0 + 1, 14\} = 14$
4		0	$\max\{K(4 - wt_2, 1) + v_2, K(4, 1)\}$ = $\max\{K(4 - 3, 1) + 14, 0\}$ = $\max\{0 + 14, 0\} = 14$	$\max\{K(4 - wt_3, 2) + v_3, K(4, 2)\}$ = $\max\{K(4 - 4, 2) + 16, 14\}$ = $\max\{0 + 16, 14\} = 16$	$\max\{K(4 - wt_4, 3) + v_4, K(4, 3)\}$ = $\max\{K(4 - 2, 3) + 1, 16\}$ = $\max\{0 + 1, 16\} = 16$
5		0	$\max\{K(5 - wt_2, 1) + v_2, K(5, 1)\}$ = $\max\{K(5 - 3, 1) + 14, 0\}$ = $\max\{0 + 14, 0\} = 14$	$\max\{K(5 - wt_3, 2) + v_3, K(5, 2)\}$ = $\max\{K(5 - 4, 2) + 16, 14\}$ = $\max\{0 + 16, 14\} = 16$	$\max\{K(5 - wt_4, 3) + v_4, K(5, 3)\}$ = $\max\{K(5 - 2, 3) + 1, 16\}$ = $\max\{14 + 1, 16\} = 16$
6		$\max\{K(6 - wt_1, 1) + v_1, K(6, 1 - 1)\}$ = $\max\{K(6 - 6, 1) + 30, 0\}$ = $\max\{0 + 30, 0\} = 30$	$\max\{K(6 - wt_2, 1) + v_2, K(6, 1)\}$ = $\max\{K(6 - 3, 1) + 14, 30\}$ = $\max\{0 + 14, 30\} = 30$	$\max\{K(6 - wt_3, 2) + v_3, K(6, 2)\}$ = $\max\{K(6 - 4, 2) + 16, 30\}$ = $\max\{0 + 16, 30\} = 30$	$\max\{K(6 - wt_4, 3) + v_4, K(6, 3)\}$ = $\max\{K(6 - 2, 3) + 1, 30\}$ = $\max\{16 + 1, 30\} = 30$
7		$\max\{K(7 - wt_1, 1) + v_1, K(7, 1 - 1)\}$ = $\max\{K(7 - 6, 1) + 30, 0\}$ = $\max\{0 + 30, 0\} = 30$	$\max\{K(7 - wt_2, 1) + v_2, K(7, 1)\}$ = $\max\{K(7 - 3, 1) + 14, 30\}$ = $\max\{0 + 14, 30\} = 30$	$\max\{K(7 - wt_3, 2) + v_3, K(7, 2)\}$ = $\max\{K(7 - 4, 2) + 16, 30\}$ = $\max\{14 + 16, 30\} = 30$	$\max\{K(7 - wt_4, 3) + v_4, K(7, 3)\}$ = $\max\{K(7 - 2, 3) + 1, 30\}$ = $\max\{16 + 1, 30\} = 30$
8		$\max\{K(8 - wt_1, 1) + v_1, K(8, 1 - 1)\}$ = $\max\{K(8 - 6, 1) + 30, 0\}$ = $\max\{0 + 30, 0\} = 30$	$\max\{K(8 - wt_2, 1) + v_2, K(8, 1)\}$ = $\max\{K(8 - 3, 1) + 14, 30\}$ = $\max\{0 + 14, 30\} = 30$	$\max\{K(8 - wt_3, 2) + v_3, K(8, 2)\}$ = $\max\{K(8 - 4, 2) + 16, 30\}$ = $\max\{14 + 16, 30\} = 30$	$\max\{K(8 - wt_4, 3) + v_4, K(8, 3)\}$ = $\max\{K(8 - 2, 3) + 1, 30\}$ = $\max\{30 + 1, 30\} = 31$

Πίνακας 8: Επίλυση παραδείγματος σακιδίου χωρίς επανάληψη

Ιδέα

Ο Αλγόριθμος 7 υλοποιεί την διαδικασία. Η διαδικασία επιστρέφει στο όνομα της την μεγαλύτερη αξία και ο πίνακας $x[]$ τα είδη που αντιστοιχούν στην αξία αυτή.

Αλγόριθμος

Algorithm 7 Υπολογισμός σακιδίου - έκδοση χωρίς επανάληψη

```

1: int Knapsack(int n, int W, int v[], int wt[], int x[])
2: for i ← 0; i ≤ n; i ++ do
3:   x[i] ← 0;
4:   for w ← 0; w ≤ W; w ++ do
5:     K[w][i] ← 0; xweight[w][i] ← 0;
6:   end for
7: end for
8: for w ← 1; w ≤ W; w ++ do
9:   for i ← 1; i ≤ n; i ++ do
10:    if wt[i] > w then
11:      K[w][i] ← K[w][i - 1]; xweight[w][i] ← 0;
12:    else
13:      if K[w][i - 1] > K[w - wt[i]][i - 1] + v[i] then
14:        K[w][i] ← K[w][i - 1]; xweight[w][i] ← 0;
15:      else
16:        K[w][i] ← K[w - wt[i]][i - 1] + v[i]; xweight[w][i] ← 1;
17:      end if
18:    end if
19:  end for
20: end for
21: w = W
22: for i = n; i ≥ 1; i -- do
23:   if xweight[w][i] = 1 then

```

24: $x[i] \leftarrow 1$; $w \leftarrow w - wt[i]$
 25: **end if**
 26: **end for**
 27: return $K[W][n]$;

Πολυπλοκότητα

Ο Αλγόριθμος 7 είναι τάξης $O(n \cdot W)$ (γραμμές 8-20).

Σημειώσεις

{ To πρόβλημα των διαρίκτη
 Knapsack 0-1 problem } Chapter 13b

1 $kL(w, :i) \rightarrow$ πάγκη w κιλώ KΛ
 Πρόσθιμη v_i το i αντικείμενο
 $kL(w, i) \rightarrow$ για προσθήμη a_i σακίδιον w προπών
 πάγκη w κιλώ KΛ v_i το i αντικείμενο

2 $Enefisis$ $\left\{ \begin{array}{l} \text{ΕΙΤΕ} \text{ επιλέγω το αντικείμενο } i \\ i \text{ σε } w \text{ το επιλέγω } \end{array} \right\}$ $\left\{ \begin{array}{l} w_{ti} \leq w \\ w = 0 \text{ ή } i = 0 \end{array} \right.$

3 $k(w, i) = \begin{cases} kL(w, :i-1), & w_{ti} > w \\ \max\{kL(w, :i-1), v_i + k(w-w_{ti})\}_{i=1}^n \}, & w_{ti} \leq w \\ 0, & w=0 \text{ ή } i=0 \end{cases}$

4 $\begin{array}{l} \text{for } w=L; w \leq W; w++ \\ \quad \text{for } i=L; i \leq n; i++ \\ \quad \quad \text{if } k(w, :i) \end{array}$

Προϊνοθεστή:
 ΔΕΝ μνησίων
 να προσθέων
 2 λεπτή
 το ίδιο
 αντικείμενο

5 $k(W, n)$

-35-

Πολυπλοκότητα: $\Theta(n \cdot W)$

$n \cdot W$ στονοβλίπτων

$K(g) = 3L$

Πάγκη τα αντικείμενα (4, L)

Σημειώσεις

Πάτα εδώ για το πρόβλημα με την μία πάραμετρο

{ Ο μικρότερος αριθμός χαρτονομισμάτων
με ίδια έπιπληγή το πολὺ 2 ροξές
το ίδια νόμιμα (Παραλλαγή της Αστυνομίας 3)
Sliders-and-Exercises → 09_DynamicExer

Chapter 13c

Παράδειγμα

$A[1] = 1 \leq A[2] = 1 \leq A[3] = 2 \leq A[4] = 2 \leq A[5] = 5 \leq$
 $\leq A[6] = 5$, $X = 4 \text{ €} \rightarrow$ Το πολύ που θέλω να μετέψω
με ταν μικρότερο αριθμό χαρτονομισμάτων
Έχω νομίσματα των 1, των 2 και των 5 ευρώ
και μπορώ να πάρω κάθε νόμιμα το πολύ 2 ροξές.
Άρα έχω 6 αντικείμενα $|A[n]| = 6$

1] $SC(x, :j) \rightarrow$ Η λύση των κάθε υπορεθλητών που
προσέβηται στο πολύ x και καταλήγει
στο νόμιμο j

$SC(x, j) \rightarrow$ Ορίζω ταν ελεύθερο αριθμό των νομίσματων που
χρηγοποιών για να σταύρω στο πολύ x αν έχω
χρηγοποιήσει το νόμιμο j

2] x^* 1 ροξά 2 ροξές
Είτε συλλέγω το νόμιμο j Είτε συλλέγω το νόμιμο j *
Είτε δεν συλλέγω το νόμιμο j Είτε δεν συλλέγω το νόμιμο j
Endeges 0 ροξές
Είτε δεν συλλέγω το νόμιμο j | $x - A[j] < 0$
* $| x - A[j] \geq 0$ $j > 0$

3]

$$S(x, j) = \begin{cases} S(x, j) = 0, & \text{av } x \leq 0 \\ S(x, j) = \infty, & \text{av } x > 0, j = 0 \\ S(x, j-1), & \text{av } x - A[j] \leq 0, j > 0 \\ \min\{L + S(x - A[j], j-1), S(x, j-1)\}, & \text{av } x - A[j] \geq 0, j > 0 \end{cases}$$

4]for $x=1; x \leq X; x++$ for $j=1; j \leq n; j++$ Solve $S(x, j)$ 5] $S(x, n)$

-38-

x	$A[1] = 1 \epsilon$	$A[2] = 1 \epsilon$	$A[3] = 2 \epsilon$	$A[4] = 2 \epsilon$	$A[5] = 5 \epsilon$	$A[6] = 5 \epsilon$
1	$\min\{S(1-1, 1-1) + 1, S(1, 1-1)\} = \min\{S(0, 0) + 1, S(1, 0)\} = \min\{0 + 1, \infty\} = 1$	$\min\{S(1-1, 2-1) + 1, S(1, 2-1)\} = \min\{S(0, 1) + 1, S(1, 1)\} = \min\{0 + 1, 1\} = 1$	$S(1, 3-1) = S(1, 2) = 1$	$S(1, 4-1) = S(1, 3) = 1$	$S(1, 5-1) = S(1, 4) = 1$	$S(1, 6-1) = S(1, 5) = 1$
2	$\min\{S(2-1, 1-1) + 1, S(2, 1-1)\} = \min\{S(1, 0) + 1, S(2, 0)\} = \min\{\infty + 1, \infty\} = \infty$	$\min\{S(2-2, 2-1) + 1, S(2, 2-1)\} = \min\{S(1, 1) + 1, S(2, 1)\} = \min\{1 + 1, \infty\} = 2$	$\min\{S(2-2, 3-1) + 1, S(2, 2-1)\} = \min\{S(1, 2) + 1, S(2, 2)\} = \min\{2 + 1, 2\} = 2$	$\min\{S(2-2, 4-1) + 1, S(2, 3)\} = \min\{S(1, 3) + 1, S(2, 3)\} = \min\{2 + 1, 2\} = 2$	$S(2, 5-1) = S(2, 4) = 1$	$S(2, 6-1) = S(2, 5) = 1$
3	$\min\{S(3-1, 1-1) + 1, S(3, 1-1)\} = \min\{S(2, 0) + 1, S(3, 0)\} = \min\{\infty + 1, \infty\} = \infty$	$\min\{S(3-2, 2-1) + 1, S(3, 2-1)\} = \min\{S(2, 1) + 1, S(3, 1)\} = \min\{\infty + 1, \infty\} = \infty$	$\min\{S(3-2, 3-1) + 1, S(3, 3-1)\} = \min\{S(1, 2) + 1, S(3, 2)\} = \min\{2 + 1, 2\} = 2$	$\min\{S(3-2, 4-1) + 1, S(3, 4-1)\} = \min\{S(1, 3) + 1, S(3, 3)\} = \min\{2 + 1, 2\} = 2$	$S(3, 5-1) = S(3, 4) = 1$	$S(3, 6-1) = S(3, 5) = 2$
4	$\min\{S(4-1, 1-1) + 1, S(4, 1-1)\} = \min\{S(3, 0) + 1, S(4, 0)\} = \min\{\infty + 1, \infty\} = \infty$	$\min\{S(4-2, 2-1) + 1, S(4, 2-1)\} = \min\{S(3, 1) + 1, S(4, 1)\} = \min\{\infty + 1, \infty\} = \infty$	$\min\{S(4-2, 3-1) + 1, S(4, 3-1)\} = \min\{S(2, 2) + 1, S(4, 2)\} = \min\{2 + 1, 2\} = 3$	$\min\{S(4-2, 4-1) + 1, S(4, 4-1)\} = \min\{S(2, 3) + 1, S(4, 3)\} = \min\{2 + 1, 2\} = 2$	$S(4, 5-1) = S(4, 4) = 1$	$S(4, 6-1) = S(4, 5) = 2$

-39-

Η επίλυση^{των} αρχικού προβλήματος σίνα $SC(4,6) = 2$,
δηλ., ο ελάχιστος αριθμός θυμεράτων πως χρησιμοποιούν
για να εργάσουν τα πώς $X=9 \in$ σίνα $2 \{2\text{€}, 2\text{€}\}$

-40-

05_ΑΠΛΗΣΤΟΙ ΑΛΓΟΡΙΘΜΟΙ

Chapter 14

14a. Εισαγωγή

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 107-108

Διαδικασία & Διαφορές με ΔΠ

Ο ΔΠ που παρουσιάστηκε σε προηγούμενο κεφάλαιο αποτελεί μία πολύ δυνατή μεθοδολογία για την επίλυση προβλημάτων διακριτής βελτιστοποίησης. Όμως σε αρκετές περιπτώσεις ο φόρτος που συνεπάγεται η επίλυση με τη χρήση της μεθοδολογίας αυτής είναι μεγάλος και εν κατακλείδι 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 βελτιστοποίησης 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.

Έλεγχος της νομιμότητας μίας επιλογής

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.

Συγκριτική αποτίμηση των νόμιμων επιλογών

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.

Κριτήριο τερματισμού

Ακολουθεί έλεγχος του κριτηρίου τερματισμού (Σημείο (γ')). 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.

Ιδιότητες Απληστων Αλγορίθμων

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.

14b. Προγραμματισμός εργασιών

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 108-112

Πρόβλημα

Έστω ότι έχουμε ένα σύνολο εργασιών $T = \{1, \dots, n\}$. Η κάθε εργασία $t \in T$ χαρακτηρίζεται από έναν χρόνο έναρξης s_t καθώς και από ένα χρόνο περαίωσης f_t - θεωρούμε ότι $s_t < f_t$. Όλες οι εργασίες εκτελούνται σε μία μηχανή, μία εργασία κάθε χρονική στιγμή και κατά τρόπο ώστε αν έχει ξεκινήσει η επεξεργασία της εργασίας θα 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.

Ιδέα

Για τους σκοπούς της ανάλυσης θεωρούμε δύο από τις εργασίες, έστω $i, j \in T$, τέτοιες ώστε η i να προηγηθεί της εργασίας j στην επεξεργασία μέσω της μηχανής. Για να μπορεί να συμβεί αυτό, θα πρέπει $f_i \leq s_j$. Στην περίπτωση αυτή λέμε ότι οι εργασίες i, j είναι συμβατές. Ένα υποσύνολο εργασιών του T που αποτελείται από συμβατές μεταξύ τους εργασίες ονομάζεται συμβατό. Το ζητούμενο είναι να βρούμε το συμβατό υποσύνολο του T με το μεγαλύτερο πληθάριθμο.

Για τους σκοπούς της ανάλυσης θεωρούμε δύο από τις εργασίες, έστω $i, j \in T$, τέτοιες ώστε η i να προηγηθεί της εργασίας j στην επεξεργασία μέσω της μηχανής. Για να μπορεί να συμβεί αυτό, θα πρέπει $f_i \leq s_j$. Στην περίπτωση αυτή λέμε ότι οι εργασίες i, j είναι συμβατές. Ένα υποσύνολο εργασιών του T που αποτελείται από συμβατές μεταξύ τους εργασίες ονομάζεται συμβατό. Το ζητούμενο είναι να βρούμε το συμβατό υποσύνολο του T με το μεγαλύτερο πληθάριθμο.

Έλεγχος της νομιμότητας μίας επιλογής

Έστω δύο συμβατές εργασίες i, j . Μία άτερη εργασία k μπορεί να παρεμβληθεί ανάμεσα στις i, j αν

$$f_i \leq s_k \text{ και } f_k \leq s_j. \quad (15.1)$$

Με βάση την (15.1) μπορούμε να ορίσουμε το σύνολο των εργασιών

$$T_{i,j} = \{k \in T : f_i \leq s_k \text{ και } f_k \leq s_j\}. \quad (15.2)$$

Δηλαδή το σύνολο $T_{i,j}$ περιέχει κάθε εργασία η οποία μπορεί να παρεμβληθεί ανάμεσα στις i, j .

Συγκριτική αποτίμηση των νόμιμων επιλογών

Για απλότητα μπορούμε να θεωρήσουμε ότι οι εργασίες δεικτοδοτούνται κατά αύξουσα σειρά σε σχέση με τον χρόνο περαίωσης. Δηλαδή, ισχύει η σειρά

$$f_1 \leq f_2 \leq \cdots \leq f_n. \quad (15.3)$$

Κάτω από την υπόθεση αυτή ισχύει ότι το σύνολο $T_{i,j} = \emptyset$ για $j \leq i + 1$.

Ορίζουμε ως $A(i, j)$ το πρόβλημα εύρεσης του συνόλου συμβατών εργασιών με τον μεγαλύτερο πληθάριθμο για καθεμία από τις οποίες ισχύει ότι έπεται της i και προηγείται της j . Συμβολίζουμε με $A_{i,j}$ το σύνολο αυτό - δηλαδή το $A_{i,j}$ είναι το σύνολο με το μεγαλύτερο αριθμό συμβατών εργασιών από αυτές που ανήκουν στο $T_{i,j}$.²

Αν $T_{i,j} \neq \emptyset$ για οποιοδήποτε στοιχείο $k \in T_{i,j}$ ισχύει ότι

$$T_{i,j} = T_{i,k} \cup \{k\} \cup T_{k,j}. \quad (15.4)$$

Κριτήριο τερματισμού

Επίσης μπορούμε να θεωρήσουμε δύο ψευδοεργασίες, την υπ' αριθμόν 0 και την υπ' αριθμόν $n + 1$ για τις οποίες ισχύει ότι $f_0 < s_1$ και $f_n < s_{n+1}$ και επομένως $T \equiv T_{0,n+1}$. Το αρχικό πρόβλημα ορίζεται ως $A(0, n+1)$: η λύση του, ονομαστικά $A_{0,n+1}$, είναι το σύνολο που περιέχει το μεγαλύτερο αριθμό συμβατών εργασιών που ανήκουν στο $T_{0,n+1}$ (και επομένως στο T).

Λήμμα Άπληστης Επιλογής

Λήμμα 38. Έστω εργασίες $i, j \in T$ τέτοιες ώστε $T_{i,j} \neq \emptyset$ και

$$r = \operatorname{argmin}\{f_k : k \in T_{i,j}\}. \quad (15.5)$$

1. Υπάρχει σύνολο $A_{i,j}$ τέτοιο ώστε $r \in A_{i,j}$.

2. $T_{i,r} = \emptyset$.

Απόδειξη Άπληστης Επιλογής

Απόδειξη.

1. Έστω $t^* = \operatorname{argmin}\{f_t : t \in A_{i,j}\}$. Αν $t^* \neq r$ τότε μπορούμε να αφαιρέσουμε από το $A_{i,j}$ την εργασία t^* και να προσθέσουμε την εργασία r χωρίς να δημιουργηθεί ασυμβατότητα.
2. Εφόσον η εργασία r είναι αυτή που ολοκληρώνεται νωρίτερα από κάθε άλλη εργασία του συνόλου $T_{i,j}$ δεν υπάρχει άλλη εργασία που μπορεί να παρεμβληθεί ανάμεσα σε αυτή και την εργασία i . Επομένως $T_{i,r} = \emptyset$.

□

Πόρισμα

Πόρισμα 10. Έστω συμβατές εργασίες $i, j \in T$. Τότε

$$T_{i,j} = \{r\} \cup T_{r,j},$$

όπου το r δίνεται από την (15.5).

Απόδειξη πορίσματος

Απόδειξη. Εφόσον $r \in T_{i,j}$ και $T_{i,r} = \emptyset$ (Λήμμα 38), από την (15.4), έχουμε

$$T_{i,j} = T_{i,r} \cup \{r\} \cup T_{r,j} = \emptyset \cup \{r\} \cup T_{r,j} = \{r\} \cup T_{r,j}.$$

□

Λήμμα Βέλτιστης Υποδομής

Λήμμα 39. Αν $i, j \in T$ αποτελούν συμβατές εργασίες, τότε υπάρχει σύνολο $A_{i,j}$ που αποτελεί λύση του προβλήματος $A(i, j)$ για το οποίο ισχύει ότι

$$A_{i,j} = \{r\} \cup A_{r,j}, \quad (15.6)$$

όπου το r δίνεται από την (15.5).

Απόδειξη Βέλτιστης Υποδομής

Απόδειξη. Από το Λήμμα 38, γνωρίζουμε ότι για κάποιο $A_{i,j}$ ισχύει ότι $r \in A_{i,j}$. Έστω ότι το $A_{i,j} \setminus \{r\}$ δεν αποτελεί βέλτιστη λύση για το πρόβλημα $A(r, j)$. Άρα οποιοδήποτε σύνολο $A_{r,j}$ το οποίο αποτελεί λύση του προβλήματος $A(r, j)$ περιέχει περισσότερες εργασίες από το $A_{i,j} \setminus \{r\}$. Δηλαδή, έχουμε

$$|A_{r,j}| > |A_{i,j} \setminus \{r\}|. \quad (15.7)$$

Παρατηρούμε ότι κάθε εργασία στο σύνολο $A_{r,j}$ είναι συμβατή με την r . Επομένως το σύνολο $\{r\} \cup A_{r,j}$ περιέχει συμβατές εργασίες για το πρόβλημα με σύνολο εργασιών το $\{r\} \cup T_{r,j} = T_{i,j}$ - η ισότητα προκύπτει από το Πόρισμα 10. Όμως αυτό έρχεται σε αντίφαση με το ότι το σύνολο $A_{i,j}$ είναι η βέλτιστη λύση για το πρόβλημα $A(i, j)$ αφού από την (15.7) έχουμε ότι

$$|A_{r,j} \cup \{r\}| > |(A_{i,j} \setminus \{r\}) \cup \{r\}| = |A_{i,j}|.$$

□

Σύνοψη

Συνοψίζοντας, το Λήμμα 38 υποδηλώνει ότι υπάρχει βέλτιστη λύση στο υποπρόβλημα $A(i, j)$ η οποία περιέχει την εργασία r όπως αυτή περιγράφεται από την (15.5) (άπληστη επιλογή) ενώ το Λήμμα 39 επιβεβαιώνει ότι η βέλτιστη λύση του υποπροβλήματος $A(i, j)$ περιέχει τη βέλτιστη λύση του υποπροβλήματος $A(r, j)$, όπου $i < r$ (βέλτιστη υποδομή). Δηλαδή, το πρόβλημα έχει την ιδιότητα της άπλειστης επιλογής και της βέλτιστης υποδομής όταν χρησιμοποιείται ως κριτήριο επιλογής εργασίας ο συντομότερος χρόνος περαίωσης. Περαιτέρω, βλέπουμε ότι η βέλτιστη λύση του αρχικού προβλήματος μπορεί να προκύψει αν θέσουμε αρχικά $i = 0, j = n + 1$ και επαγγειακά εφαρμόσουμε την (15.6). Δηλαδή, κάνοντας την άπληστη επιλογή - ανάμεσα σε συμβατές εργασίες - σε σχέση με το μικρότερο χρόνο περαίωσης επαναληπτικά μπορούμε να κατασκευάσουμε τη (βέλτιστη) λύση του αρχικού προβλήματος.

Λειτουργία

Ο αλγόριθμος παράγει σαν έξοδο το σύνολο S καθώς και τον πληθάριθμο του.

ρατηρούμε ότι το S αρχικοποιείται από την εργασία 1 και επαυξάνεται από την εργασία $j = \operatorname{argmin}\{f_k : k \in T_{i,n+1}\}$, όπου i είναι η εργασία που είχε προστεθεί στο S πιο πρόσφατα.

Algorithm 98 Εκτύπωση μεγαλύτερου συνόλου εργασιών που μπορούν να εκτελεστούν στη μηχανή.

Απαιτείται: Ακέραιος n , πίνακες s, f που περιέχουν τους χρόνους εκκίνησης και περαίωσης, αντίστοιχα, n εργασιών. Οι εργασίες είναι δεικτοδοτημένες κατά αύξουσα σειρά των χρόνων περαίωσης: $f[1] \leq f[2] \leq \dots \leq f[n]$.

Επιστρέφεται: Το πλήθος του μεγαλύτερου αριθμού των εργασιών που μπορούν να εκτελεστούν από τη μηχανή και γίνεται εκτύπωση τους.

```

1: function JOBSPROG(int n, int s[], int f[])
2:    $i \leftarrow 1; m \leftarrow 0;$ 
3:    $S \leftarrow \{1\};$ 
4:   for  $j \leftarrow 2; j \leq n; j++ \text{ do}$ 
5:     if  $s[j] \geq f[i]$  then
6:        $S \leftarrow S \cup \{j\};$ 
7:        $i \leftarrow j;$ 
8:        $m++;$ 
9:     end if
10:   end for
11:   print S;
12:   return m + 1;
13: end function
```

Πολυπλοκότητα

Ο Αλγόριθμος 98, τάξης $O(n)$, υλοποιεί την ιδέα.

Λήμμα Ορθότητας

Λήμμα 40. Ο Αλγόριθμος 98 υπολογίζει το μέγιστο αριθμό των εργασιών που μπορούν να εκτελεστούν στη μηχανή.

Απόδειξη Ορθότητας

Απόδειξη. Από το Λήμμα 39 έχουμε ότι

$$\begin{aligned}
 A_{0,n+1} &= \{1\} \cup A_{1,n+1} = \{1, r_1\} \cup A_{r_1,n+1} \\
 &= \{1, r_1, r_2\} \cup A_{r_2,n+1} = \dots = \{1, r_1, r_2, \dots, r_m\} \cup A_{r_m,n+1} \quad (15.8) \\
 &= \bigcup_{i=0}^m \{r_i\} \cup A_{r_m,n+1}
 \end{aligned}$$

όπου

$$r_i = \begin{cases} 1, & i = 0, \\ \operatorname{argmin}\{f_k : k \in T_{r_{i-1},n+1}\}, & m \geq i > 0, \end{cases}$$

και $m = \operatorname{argmax}\{i : T_{r_{i-1},n+1} \neq \emptyset\}$. Από τον ορισμό της παραμέτρου m είναι προφανές ότι $A_{r_m,n+1} = \emptyset$ αφού $T_{r_m,n+1} = \emptyset$. Από την (15.8) έχουμε

$$A_{0,n+1} = \{1, r_1, r_2, \dots, r_m\}$$

Παρατηρούμε ότι τα στοιχεία του συνόλου αυτού είναι ακριβώς αυτά που επιλέγει ο αλγόριθμος για να «χτίσει» το S . \square

Παράδειγμα

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

Πίνακας 15.1: Πέντε εργασίες με τους αντίστοιχους χρόνους έναρξης και περαίωσης

Παράδειγμα 81. Θεωρούμε 5 εργασίες με χρόνους έναρξης-περαίωσης όπως απεικονίζονται στον Πίνακα 15.1 Παρατηρούμε ότι οι εργασίες έχουν διαταχθεί σε αύξουσα σειρά σε σχέση με το χρόνο περαίωσης. Εκτελώντας τον Αλγόριθμο 98 βλέπουμε ότι η πρώτη εργασία που θα επιλεγεί είναι η 1, δεύτερη η 4 και τρίτη η 5.

Εναλλακτικά, για να προγραμματίσουμε τις εργασίες στη μηχανή προκειμένου να εκτελεστεί ο μεγαλύτερος αριθμός αυτών θεωρούμε τις ψευδοεργασίες 0 και 6 με $f_0 = 0$ $s_6 = \infty$. Θέλουμε να βρούμε τη βέλτιστη λύση στο πρόβλημα $A(0, 6)$. Η λύση δίνεται από την αναδρομική εφαρμογή της (15.6) ήτοι

$$\begin{aligned} A_{0,6} &= \{r_0 = \operatorname{argmin}\{f_k, k \in T_{0,6}\}\} \cup A_{r_0,6} = \{1\} \cup A_{1,6} \\ &= \{1\} \cup (\{r_1 = \operatorname{argmin}\{f_k, k \in T_{1,6}\}\} \cup A_{r_1,6}) = \{1\} \cup (\{4\} \cup A_{4,6}) \\ &= \{1, 4\} \cup (\{r_2 = \operatorname{argmin}\{f_k, k \in T_{4,6}\}\} \cup A_{r_2,6}) = \{1, 4\} \cup (\{5\} \cup A_{5,6}) \\ &= \{1, 4, 5\} \cup \emptyset = \{1, 4, 5\}. \end{aligned}$$

Η παραπάνω επίλυση αποτελεί εξειδίκευση της διαδικασίας απόδειξης του Λήμματος 40 - στη συγκεκριμένη περίπτωση έχουμε $m = 2$.

Μία βασική παρατήρηση είναι ότι η ιδιότητα της άπληστης επιλογής ισχύει για το πρόβλημα μόνο αν εφαρμόσουμε επιλογή με βάση το μικρότερο χρόνο περαίωσης. Αν επιλεγεί διαφορετικό κριτήριο - π.χ. επιλογή με βάση τον μικρότερο χρόνο έναρξης - δεν είναι απαραίτητο ότι θα ικανοποιείται η ιδιότητα αυτή. Στην περίπτωση αυτή καταλήγουμε με λύση για το Παράδειγμα 81 το σύνολο $\{2, 5\}$.

ΟΦ - ΑΠΛΗΣΤΟΙ ΑΛΓΟΡΙΘΜΟΙ

12/12/2023

Chapter 14b

Περιεγγραφτικός Έργων

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

Ειστέλεση εργασιών $i \rightarrow k \rightarrow j$ $A(i,j) \rightarrow$ Υποτεθλήμα

$$\underline{A_{i,j}} \subseteq \underline{T_{i,j}} \Rightarrow f_r \leq f_t^*$$

Εινώνα

$$T_{i,j} = T_{i,t} \cup \{k\} \cup T_{k,j}, \text{ όπου για } T_{i,j} \neq \emptyset$$

$$r = \underbrace{\arg \min \{f_k : k \in T_{i,j}\}}_{\text{με τον, μικρότερο χρόνο περάτωσης } f_k} \quad r=k, \text{ οπου } k \text{ η εργασία}$$

$$t^* = \underbrace{\arg \min \{f_t : t \in A_{i,j}\}}_{\text{επιλογή}}$$

14c. Χαρακτηριστικά της μεθόδου της απληστίας

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 113-114

Λήμμα Ορθότητας Απληστού Αλγόριθμου

Η απόδειξη ορθότητας του Αλγόριθμου 98 αποτελεί μία τυπική περίπτωση επαγωγικής απόδειξης ενός άπληστου αλγόριθμου. Ουσιαστικά αυτό που χρειάζεται για να αποδείξουμε ότι ένας τέτοιος αλγόριθμος παράγει τη βέλτιστη λύση είναι αρχικώς να δείξουμε ότι πάντα υπάρχει μία βέλτιστη λύση η οποία περιέχει το τοπικά βέλτιστο στοιχείο που επιλέγει ο αλγόριθμος (ιδιότητα της άπληστης επιλογής). Στη συνέχεια, 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, η ιδιότητα της άπληστης επιλογής αποδεικνύεται από το Λήμμα 38.1 ενώ της βέλτιστης υποδομής από το Λήμμα 39.

Απόδειξη Ορθότητας Απληστου Αλγόριθμου

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. Υποθέτουμε ότι τα δύο σύνολα δεν ταυτίζονται και διακρίνουμε τα πρώτα στοιχεία στη διάταξη στα οποία τα δύο σύνολα διαφέρουν. Έστω ότι αυτό είναι το i -οστά στη διάταξη στοιχεία. Δηλαδή αν το i -ιστό 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. Με τον τρόπο αυτό αποδείξαμε την ορθότητα των αλγορίθμων για την εύρεση του ελάχιστου συνεκτικού δένδρου στο Κεφάλαιο 13.

Απληστοι Αλγόριθμοι Vs ΔΠ

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.

! Πρόβλημα να λύνεται με απληστού λύγεται κακά με Δ.Π. οχι το αντιστροφέον

! Η απληστή επιλογή ~~επιλογή~~ σειράς είναι από είναι συγκεκριμένων κριτήρων.

-36-

14d. Χρόνος παραμονής σε κατάστημα

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 114-115

Άλλη προσέγγιση της μεθόδου της απληστίας

Ένας άλλος τρόπος για να δείξουμε ότι η λύση που παράγει η απληστή μέθοδος (χρησιμοποιώντας κάποιο κριτήριο επιλογής) είναι βέλτιστη, είναι να αποδείξουμε ότι οποιαδήποτε άλλη εφικτή λύση στο πρόβλημα είναι χειρότερη. Θα εφαρμόσουμε τη μέθοδο αυτή στο παρακάτω πρόβλημα.

Πρόβλημα

Σε ένα κατάστημα οι πελάτες πληρώνουν σε ένα ταμείο. Έστω ότι ένα σύνολο πελατών $I = \{1, \dots, n\}$ συσσωρεύεται στο ταμείο και είναι γνωστό εκ' των προτέρων ότι ο πελάτης $i \in I$ θα απασχολήσει το ταμείο t_i λεπτά. Θέλουμε να βρούμε τη σειρά με την οποία θα πρέπει να εξυπηρετηθούν οι πελάτες έτσι ώστε ο μέσος χρόνος αναμονής να είναι ο μικρότερος δυνατός. Για να κατανοήσουμε καλύτερα το πρόβλημα θεωρούμε το επόμενο παράδειγμα.

Παράδειγμα

Παράδειγμα 82. Έστω $I = \{1, 2, 3\}$ με χρόνο εξυπηρέτησης (σε λεπτά) $t_1 = 1, t_2 = 5, t_3 = 3$. Αν η σειρά εξυπηρέτησης είναι $3 - 2 - 1$ τότε ο Πελάτης 3 θα περιμένει 3 λεπτά ο Πελάτης 2 ($3 + 5 = 8$ λεπτά) και ο Πελάτης 1 ($5 + 3 + 2 = 10$ λεπτά). Στην περίπτωση αυτή ο μέσος χρόνος αναμονής στο κατάστημα είναι το άθροισμα των χρόνων αναμονής δια του συνολικού αριθμού των πελατών, ήτοι $(3 + 8 + 10)/3 = 7$. Αν θεωρήσουμε ως σειρά εξυπηρέτησης την $2 - 1 - 3$ τότε πρώτα θα εξυπηρετηθεί ο Πελάτης 2 με χρόνο αναμονής 5 λεπτά, στη συνέχεια ο Πελάτης 1 με χρόνο αναμονής $5 + 1 = 6$ λεπτά και στο τέλος ο Πελάτης 3 με χρόνο αναμονής $5 + 6 + 3 = 14$ λεπτά. Επομένως ο μέσος χρόνος αναμονής, στην περίπτωση αυτή, είναι $(5 + 6 + 14)/3 = 25/3 = 8,333$ λεπτά. Είναι φανερό ότι διαφορετική σειρά εξυπηρέτησης οδηγεί σε διαφορετικό μέσο χρόνο αναμονής στο κατάστημα. Αναζητούμε λοιπόν από τους $3! = 6$ διαφορετικούς τρόπους εξυπηρέτησης (σειρές εξυπηρέτησης) αυτόν ο οποίος ελαχιστοποιεί το μέσο χρόνο αναμονής στο κατάστημα.

Κριτήριο τερματισμού

Ένα προφανές κριτήριο για την επίλυση του παραπάνω προβλήματος είναι να δίνεται προτεραιότητα κάθε φορά στον πελάτη με το μικρότερο χρόνο εξυπηρέτησης.

Συγκριτική αποτίμηση των νόμιμων επιλογών

Η εφαρμογή του κριτηρίου είναι πολύ εύκολη: οι πελάτες εξυπηρετούνται με τη σειρά $i_1 - i_2 - \dots - i_n$ όπου $\{i_1, i_2, \dots, i_n\} = I$ και $t_{i_1} \leq t_{i_2} \leq \dots \leq t_{i_n}$. Η εξυπηρέτηση με τη σειρά αυτή περιγράφει έναν άπληστο αλγόριθμο.

Έλεγχος της νομιμότητας μίας επιλογής

ότι ο αλγόριθμος αυτός οδηγεί σε μία σειρά (λύση) στην οποία όλοι οι πελάτες εξυπηρετούνται. Ο συνολικός χρόνος αναμονής στο κατάστημα με βάση αυτή τη σειρά εξυπηρέτησης δίνεται από τον τύπο

$$\begin{aligned} T &= t_{i_1} + (t_{i_1} + t_{i_2}) + \dots \\ &\quad + (t_{i_1} + t_{i_2} + \dots + t_{i_{n-1}}) + (t_{i_1} + t_{i_2} + \dots + t_{i_n}) \\ &= n \cdot t_{i_1} + (n-2) \cdot t_{i_2} + \dots + t_{i_n} \\ &= \sum_{j=1,\dots,n} (n-j+1)t_{i_j} \end{aligned} \tag{15.9}$$

Λήμμα Άπληστης Επιλογής & Βέλτιστης Υποδομή

Πρόταση 3. Οποιαδήποτε σειρά εξυπηρέτησης έχει συνολικό χρόνο αναμονής στο κατάστημα μεγαλύτερο-ίσο με το χρόνο που προκύπτει από την (15.9).

Απόδειξη Άπληστης Επιλογής & Βέλτιστης Υποδομή

Απόδειξη. Θεωρούμε δύο πελάτες i_v, i_u με $v < u$ (και άρα $t_{i_v} \leq t_{i_u}$) όπου ο i_u εξυπηρετείται πριν τον i_v . Αν οι υπόλοιποι πελάτες εξυπηρετούνται με αύξουσα σειρά σε σχέση με το χρόνο εξυπηρέτησης, έχουμε το συνολικό χρόνο αναμονής στο σύστημα

$$T' = (n-v+1)t_{i_u} + (n-u+1)t_{i_v} + \sum_{j \in I \setminus \{v,u\}} t_{i_j} \tag{15.10}$$

Αφαιρώντας τη (15.10) από τη (15.9) παίρνουμε

$$\begin{aligned} T - T' &= (n-v+1)(t_{i_v} - t_{i_u}) + (n-u+1)(t_{i_u} - t_{i_v}) \\ &= (n-v+1)(t_{i_v} - t_{i_u}) - (n-u+1)(t_{i_v} - t_{i_u}) \\ &= (u-v)(t_{i_v} - t_{i_u}) \leq 0, \end{aligned}$$

αφού $u - v > 0$ και $t_{i_v} - t_{i_u} \leq 0$. Άρα $T \leq T'$.

Εφαρμόζοντας επαγωγικά την αλλαγή στη προτεραιότητα εξυπηρέτησης σε οποιοδήποτε ζευγάρι πελατών i_v, i_u , με $v < u$, μπορούμε να παράγουμε οποιαδήποτε άλλη σειρά εξυπηρέτησης από τις $n!$ σειρές. Όμως ο συνολικός χρόνος αναμονής θα είναι στην καλύτερη περίπτωση ίσος με το χρόνο αναμονής της σειράς που παράγει ο άπληστος αλγόριθμος.

□

Εφαρμογή Παραδείγματος

Επομένως για το στιγμιότυπο του Παραδείγματος 82 η σειρά εξυπηρέτησης που ελαχιστοποιεί το μέσο χρόνο αναμονής είναι $1 - 3 - 2$ με μέσο χρόνο εξυπηρέτησης $\frac{1 \cdot 3 + 3 \cdot 2 + 5 \cdot 1}{3} = \frac{14}{3} = 4,666$ λεπτά.

Chapter 15

15a. Απροθηματικοί κώδικες

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 115-122

Πρόβλημα

Ένα από τα βασικότερα προβλήματα στη συμπίεση δεδομένων (data compression) είναι αυτό της αναπαράστασης μίας ακολουθίας χαρακτήρων με τον ελάχιστο αριθμό δυαδικών ψηφίων. Για να επιτύχουμε κάτι τέτοιο μπορούμε να αντιστοιχίσουμε κάθε χαρακτήρα σε μία διαφορετική δυαδική συμβολοσειρά (κωδικολέξη). Για πα-

ράδειγμα, αν χρησιμοποιούνται μόνο οι χαρακτήρες από το σύνολο $\{\alpha, \beta, \gamma, \delta, \varepsilon, \zeta\}$, μπορούμε να αντιστοιχίσουμε τα γράμματα αυτά σε συμβολοσειρές σταθερού μήκους τριών ψηφίων όπως αυτές που απεικονίζονται στην τρίτη γραμμή του Πίνακα 15.2. Αν οι συχνότητες που εμφανίζονται οι χαρακτήρες σε κάποιο κείμενο είναι αυτές που απεικονίζονται στη δεύτερη γραμμή του πίνακα, τότε ο αριθμός των δυαδικών ψηφίων που κωδικοποιεί το κείμενο χρησιμοποιώντας την κωδικοποίηση αυτή είναι

$$(344 + 96 + 12 + 16 + 9 + 5) \cdot 3 = 1446.$$

Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε την κωδικοποίηση μεταβλητού μήκους που παρουσιάζεται στην τελευταία γραμμή του πίνακα. Τότε το κείμενο κωδικοποιείται σε μία δυαδική συμβολοσειρά με μήκος

$$344 \cdot 2 + 96 \cdot 2 + 12 \cdot 3 + 16 \cdot 3 + 9 \cdot 3 + 5 \cdot 3 = 1006.$$

Παρατηρούμε ότι η κωδικοποίηση των χαρακτήρων στην τελευταία γραμμή του πίνακα είναι απροθηματική: καμία συμβολοσειρά δεν αποτελεί πρόθεμα κάποιας άλλης. Πλεονέκτημα της χρήσης μίας τέτοιας κωδικοποίησης είναι η εύκολα αποκωδικοποίηση του κειμένου. Δεδομένου ότι καμία κωδικολέξη δεν συμπίπτει με το πρόθεμα κάποιας άλλης η εναρκτήρια κωδικολέξη σε ένα κωδικοποιημένο κείμενο είναι μονοσήμαντα ορισμένη. Επομένως μπορούμε απλώς να προσδιορίσουμε την πρώτη κωδικολέξη και αφού την αποκωδικοποιήσουμε να επαναλάβουμε τη διαδικασία για το υπόλοιπο κωδικοποιημένο κείμενο.

Πίνακας 15.2 Απροθηματική κωδικοποίηση σταθερού και μεταβλητού μήκους

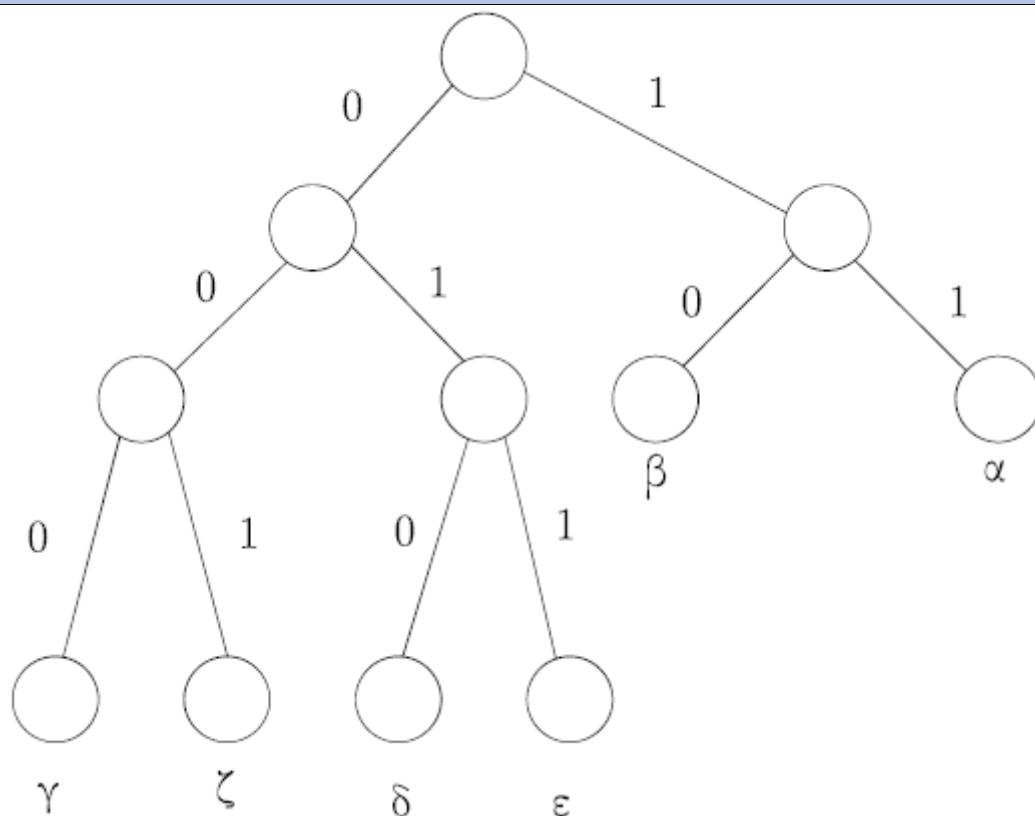
	α	β	γ	δ	ε	ζ
Συχνότητα	344	96	12	16	9	5
Σταθ. Μήκος	000	001	010	011	100	101
Μετ. Μήκος	11	10	000	010	011	001

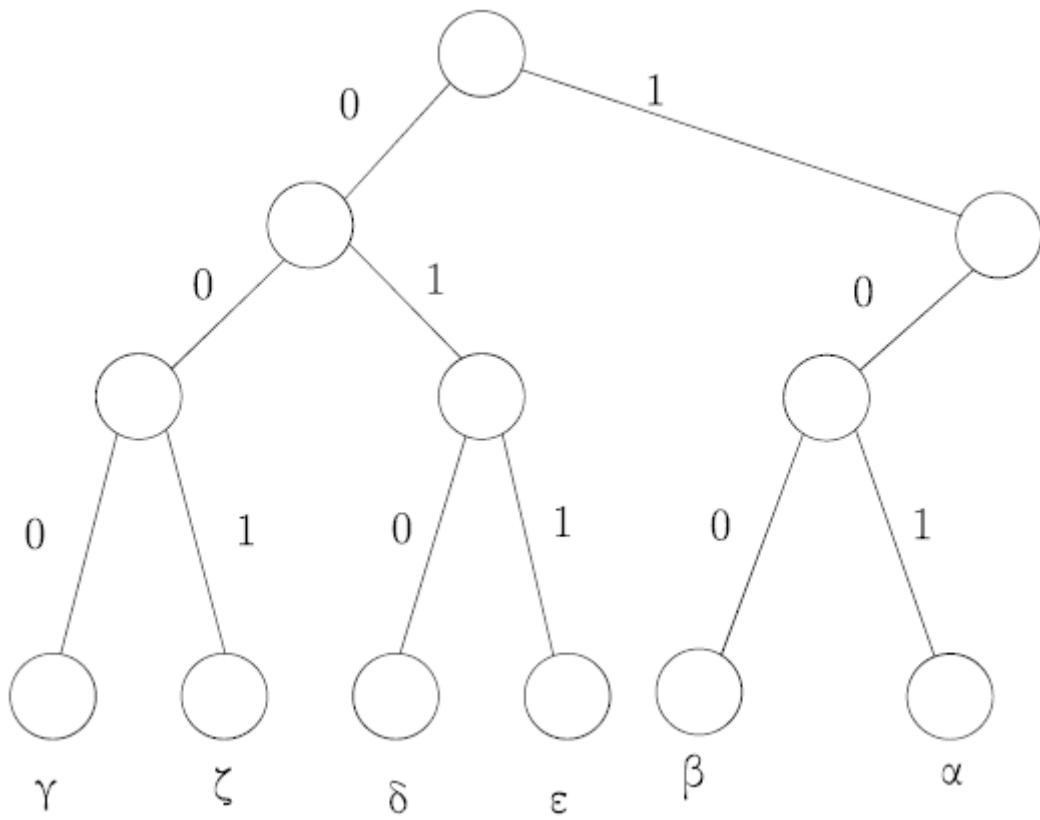
Ιδέα

Για τη διαδικασία αποκωδικοποίησης μπορούμε να χρησιμοποιήσουμε μία αναπαράσταση του σχήματος κωδικοποίησης η οποία θα μας δίνει τη δυνατότητα να εντοπίσουμε με ευκολία την πρώτη κωδικολέξη. Μία προτεινόμενη αναπαράσταση είναι μέσω ενός δυαδικού δένδρου όπου τα φύλλα του αντιστοιχούν στα γράμματα. Σε ένα τέτοιο δένδρο η δυαδική κωδικολέξη ενός χαρακτήρα συνίσταται στη απλή διαδρομή από τη ρίζα του δένδρου μέχρι το αντίστοιχο φύλλο, όπου το 0 σημαίνει μετάβαση κατά μήκος της αριστερής ακμής και 1 μέσω της δεξιάς. Για παράδειγμα, το δένδρο που αντιστοιχεί στην κωδικοποίηση μεταβλητού μήκους του Πίνακα 15.2 απεικονίζεται στο Σχήμα 15.1 Με αντίστοιχο τρόπο η κωδικοποίηση σταθερού μήκους του Πίνακα 15.2 απεικονίζεται στο Σχήμα 15.2.

Παρατηρούμε ότι το δένδρο του Σχήματος 15.1 είναι πλήρες δυαδικό ενώ αυτό του Σχήματος 15.2 είναι μεν δυαδικό αλλά όχι πλήρες δυαδικό. Γενικά ισχύει ότι ένα βέλτιστο σχήμα κωδικοποίησης - δηλαδή μία κωδικοποίηση βάσει της οποίας το παραγόμενο κείμενο έχει το μικρότερο αριθμό bits - αναπαρίσταται πάντα με ένα πλήρες δυαδικό δένδρο.

Σχήμα 15.1 Κωδικοποίηση μεταβλητού μήκους Πίνακα 15.2





Έλεγχος της νομιμότητας μίας επιλογής

Αν γνωρίζουμε το δένδρο T που αντιστοιχεί σε ένα σχήμα κωδικοποίησης κάποιου συνόλου χαρακτήρων C , μπορούμε να υπολογίσουμε το μήκος της συμβολοσειράς του κωδικοποιημένου κειμένου. Αυτό δίνεται από τον τύπο

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

όπου $f(c)$ είναι η συχνότητα εμφάνισης του χαρακτήρα c στο κείμενο και $d_T(c)$ το βάθος του φύλλου - αριθμός ακμών στο μονοπάτι από την κορυφή αυτή μέχρι την ρίζα - που αντιστοιχεί στον χαρακτήρα c . Η ποσότητα $B(T)$ ονομάζεται κόστος του δένδρου T .

Συγκριτική αποτίμηση των νόμιμων επιλογών

Η μέθοδος Huffman [11] υπολογίζει το δένδρο με το ελάχιστο κόστος.

Για να την περιγράψουμε ορίζουμε:

\mathcal{T} : ένα σύνολο από δένδρα με φύλλα (κάποιους από) τους χαρακτήρες του αλφάριθμητου C ,

T_S : ένα δυαδικό δένδρο με φύλλα τους χαρακτήρες του συνόλου $S \subseteq C$,

\oplus : ένας τελεστής ο οποίος όταν εφαρμόζεται στα δένδρα T_{S_1}, T_{S_2} , όπου $S_1 \cup S_2 \subseteq C$, $S_1 \cap S_2 = \emptyset$, τα συνενώνει: δημιουργεί ένα δυαδικό δένδρο $T_{S_1 \cup S_2}$ υπάγοντας τις δύο ρίζες των T_{S_1}, T_{S_2} σε κοινή ρίζα.

Κριτήριο τερματισμού

Επίσης για κάθε δυαδικό δένδρο, η συχνότητα μίας κορυφής v ορίζεται αναδρομικά ως

$$f(v) = \begin{cases} f(c), & \text{αν } v \text{ είναι το φύλλο που} \\ & \text{αντιπροσωπεύει το χαρακτήρα } c \in C, \\ f(u) + f(w), & \text{αν } v \text{ είναι εσωτερική και οι κορυφές } u, w \\ & \text{είναι άμεσοι απόγονοι της } v. \end{cases} \quad (15.11)$$

Η συχνότητα ενός δένδρου T , ονομαστικά $f(T)$, είναι συχνότητα της ρίζας του.

Η μέθοδος Huffman είναι ένας άπληστος αλγόριθμος: σε κάθε επανάληψη συνενώνει τα δύο δένδρα με τη μικρότερη συχνότητα ρίζας.

Λειτουργία

Η μέθοδος Huffman κατασκευάζει ένα δένδρο κωδικοποίησης T επιλέγοντας κάθε φορά, από ένα σύνολο δένδρων \mathcal{T} , να συνενώσει τα δύο δένδρα με τις μικρότερες συχνότητες ρίζας. Στη συνέχεια αντικαθιστά τα δένδρα αυτά με το δένδρο που κατασκεύασε στο σύνολο \mathcal{T} . Αρχικοποιώντας το σύνολο \mathcal{T} με δένδρα που αποτελούνται από μία κορυφή, δηλαδή $\mathcal{T} = \cup_{c \in C} T_{\{c\}}$ και εφαρμόζοντας επαναληπτικά την διαδικασία που περιγράφηκε προηγουμένως, στο τέλος, καταλήγει το σύνολο \mathcal{T} να περιέχει μόνο ένα δένδρο. Αυτό αποτελεί το δένδρο Huffman - έξοδος του Αλγόριθμου 99.

Αλγόριθμος

Algorithm 99 Αλγόριθμος Huffman

Απαιτείται: Αλφάβητο C , πίνακας f όπου στη θέση c περιέχει την συχνότητα εμφάνισης του χαρακτήρα $c \in C$ στα κείμενο.

Επιστρέφεται: Δένδρο Huffman

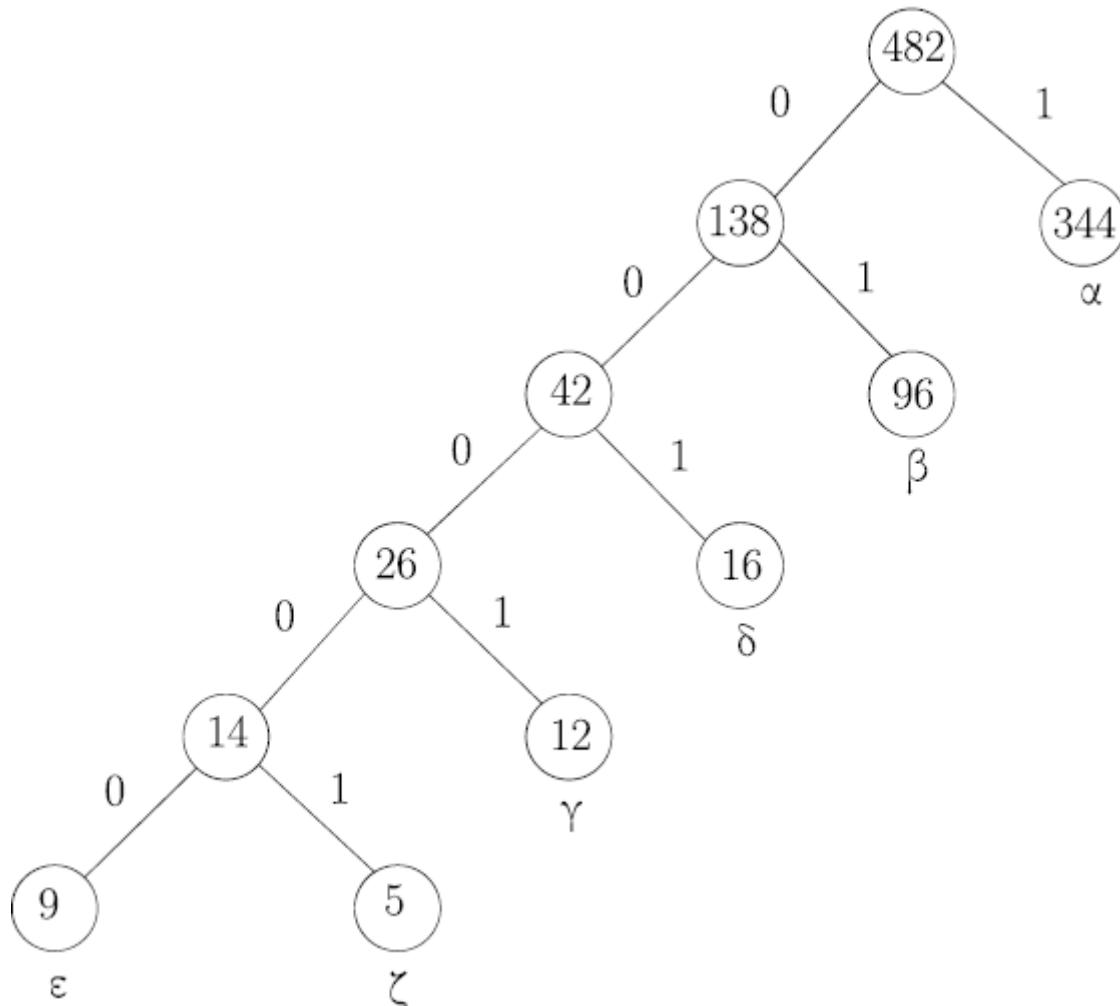
```
1: function HUFFMAN(set  $C$ , int  $f[]$ )
2:    $\mathcal{T} \leftarrow \emptyset$ ;
3:   for all  $c \in C$  do
4:      $\mathcal{T} \leftarrow \mathcal{T} \cup \{T_{\{c\}}\}$ ;
5:   end for
6:   while  $|\mathcal{T}| > 1$  do
7:      $T_{S_1} \leftarrow \text{argmin}\{f(T_S) : T_S \in \mathcal{T}\}$ ;
8:      $\mathcal{T} \leftarrow \mathcal{T} \setminus \{T_{S_1}\}$ ;
9:      $T_{S_2} \leftarrow \text{argmin}\{f(T_S) : T_S \in \mathcal{T}\}$ ;
10:     $\mathcal{T} \leftarrow \mathcal{T} \setminus \{T_{S_2}\}$ ;
11:     $T \leftarrow T_{S_1} \oplus T_{S_2}$ ;
12:     $\mathcal{T} \leftarrow \mathcal{T} \cup \{T\}$ ;
13:   end while
14:   return  $T$ ;
15: end function
```

Εφαρμογή Παραδείγματος

Το δένδρο Huffman για το στιγμιότυπο που περιγράφεται στον Πίνακα 15.2 - σε κάθε κορυφή απεικονίζεται η συχνότητα της από την (15.11) παρουσιάζεται στο Σχήμα 15.3. Το μήκος της δυαδικής συμβολοσειράς που κωδικοποιεί το κείμενο με βάση το σχήμα αυτό είναι

$$344 \cdot 1 + 96 \cdot 2 + 16 \cdot 3 + 12 \cdot 4 + 9 \cdot 5 + 5 \cdot 5 = 702.$$

Σχήμα 15.3 Δένδρο Huffman για το στιγμιότυπο του Πίνακα 15.2



Λήμμα Απληστης Επιλογής

Λήμμα 41. Έστω x', y' , οι χαρακτήρες του αλφάβητου C με τις μικρότερες συχνότητες. Υπάρχει βέλτιστο απροθηματικό δένδρο T για το αλφάβητο C τέτοιο ώστε οι χαρακτήρες αυτοί να αντιπροσωπεύονται από φύλλα μεγαλύτερου βάθους και έχουν κοινή κορυφή ως άμεσο πρόγονο.

Απόδειξη Απληστης Επιλογής

Απόδειξη. Έστω T^* ένα βέλτιστο απροθηματικό δένδρο για το αλφάβητο C και x, y οι χαρακτήρες που αντιστοιχούν σε φύλλα μεγαλύτερου βάθους με κοινό άμεσο πρόγονο. Προφανώς αν $\{x, y\} = \{x', y'\}$ το δένδρο T^* είναι το ζητούμενο δένδρο T . Στην αντίθετη περίπτωση, μπορούμε να υποθέσουμε ότι $f(x) \leq f(y)$ και $f(x') \leq f(y')$. Θεωρούμε ως T το δένδρο που παράγεται από το T^* εναλλάσσοντας τον x με τον x' στα φύλλα που τους αντιπροσωπεύουν και αντίστοιχα τον y με τον y' .

$$\begin{aligned}
B(T^*) - B(T) &= \sum_{c \in C} f(c) \cdot d_{T^*}(c) - \sum_{c \in C} f(c) \cdot d_T(c) \\
&= f(x')(d_{T^*}(x') - d_T(x')) \\
&\quad + f(y')(d_{T^*}(y') - d_T(y')) \\
&\quad + f(x)(d_{T^*}(x) - d_T(x)) \\
&\quad + f(y)(d_{T^*}(y) - d_T(y)).
\end{aligned} \tag{15.12}$$

Επειδή

$$\begin{aligned}
d_{T^*}(x) - d_T(x) &= -(d_{T^*}(x') - d_T(x')), \\
d_{T^*}(y) - d_T(y) &= -(d_{T^*}(y') - d_T(y')),
\end{aligned}$$

η (15.12) γίνεται

$$\begin{aligned}
B(T^*) - B(T) &= (f(x') - f(x))(d_{T^*}(x') - d_T(x')) \\
&\quad + (f(y') - f(y))(d_{T^*}(y') - d_T(y')). \tag{15.13}
\end{aligned}$$

Λαμβάνοντας υπόψη ότι

$$f(x') \leq f(x), \quad f(y') \leq f(y), \quad d_{T^*}(x') \leq d_T(x'), \quad d_{T^*}(y') \leq d_T(y'),$$

η (15.13) συνεπάγεται $B(T^*) - B(T) \geq 0$. Επειδή το T^* είναι βέλτιστο δένδρο κωδικοποίησης η τελευταία σχέση ισχύει σαν ισότητα. \square

Λήμμα Βέλτιστης Υποδομής

Λήμμα 42. Έστω $x, y \in C$ οι δύο χαρακτήρες με τη μικρότερη συγνότητα. Θεωρούμε ένα νέο χαρακτήρα $z \notin C$ με $f(z) = f(x) + f(y)$, βάσει του οποίου ορίζουμε το αλφάριθμο $C' = C \setminus \{x, y\} \cup \{z\}$. Αν T' είναι το βέλτιστο απροθηματικό δένδρο για το αλφάριθμο C' , τότε το βέλτιστο απροθηματικό δένδρο, έστω T , για το αλφάριθμο C προκύπτει από το T' αν η κορυφή που αντιστοιχεί στο χαρακτήρα z γίνει εσωτερική αποκτώντας δύο άμεσους απογόνους (φύλλα) που αντιστοιχούν στους χαρακτήρες x, y .

Απόδειξη Βέλτιστης Υποδομής

Απόδειξη. Παρατηρούμε ότι για κάθε $c \in C \setminus \{x, y\}$ ισχύει $f(c)d_T(c) = f(c)d_{T'}(c)$. Αθροίζοντας κατά μέλη, παίρνουμε

$$\sum_{c \in C \setminus \{x, y\}} f(c)d_T(c) = \sum_{c \in C \setminus \{x, y\}} f(c)d_{T'}(c) \tag{15.14}$$

Επίσης ισχύει ότι

$$\begin{aligned}
f(x)d_T(x) + f(y)d_T(y) &= (f(x) + f(y))(d_{T'}(z) + 1) \\
&= f(z)d_{T'}(z) + f(x) + f(y).
\end{aligned} \tag{15.15}$$

Προσθέτοντας τις (15.14), (15.15) έχουμε

$$B(T) = B(T') + f(x) + f(y) \Rightarrow B(T') = B(T) - f(x) - f(y). \quad (15.16)$$

Έστω ότι το T δεν είναι βέλτιστο. Τότε υπάρχει βέλτιστο δένδρο T^* για το αλφάβητο C τέτοιο ώστε $B(T^*) < B(T)$. Σύμφωνα με το Λήμμα 41, μπορούμε να θεωρήσουμε ότι οι κορυφές που αντιστοιχούν στους χαρακτήρες x, y στο T^* έχουν κοινό άμεσο πρόγονο. Έστω T^{**} το δένδρο που προκύπτει από το T^* αν αντικαταστήσουμε την κορυφή που αποτελεί άμεσα πρόγονο από ένα φύλλο z με $f(z) = f(x) + f(y)$. Στην περίπτωση αυτή

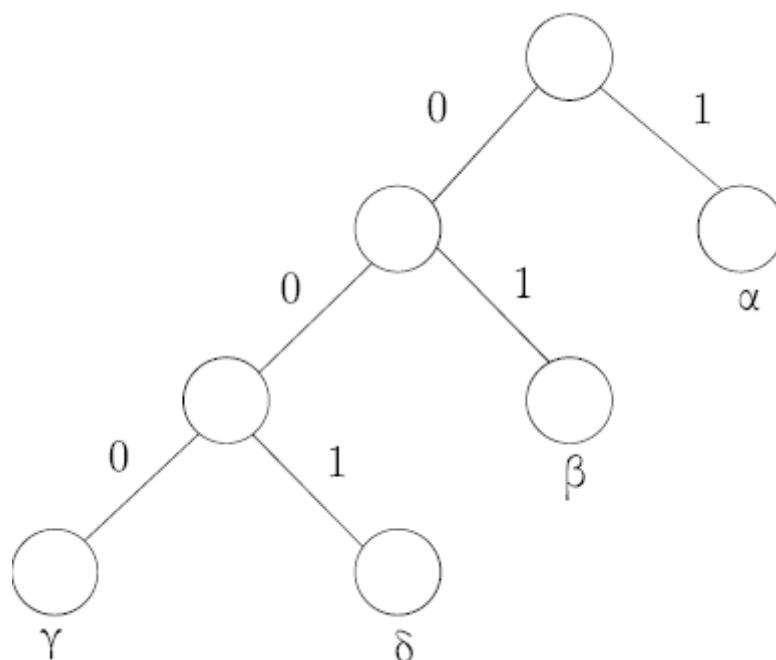
$$B(T^{**}) = B(T^*) - f(x) - f(y) < B(T) - f(x) - f(y) = B(T'),$$

όπου η τελευταία ισότητα προκύπτει άμεσα από τη (15.16). Η ανισότητα $B(T^{**}) < B(T')$ οδηγεί σε άτοπο αφού το T' είναι εξ' υποθέσεως βέλτιστο για το αλφάβητο C' . Επομένως το T είναι βέλτιστο για το αλφάβητο C . \square

Συνέπεια Βέλτιστης Υποδομής

Άμεση συνέπεια του Λήμματος 42 είναι ότι αν T αποτελεί το βέλτιστο δένδρο για το αλφάβητο C τότε το δένδρο αυτό «περιέχει» το βέλτιστο δένδρο για το αλφάβητο $C \setminus \{x, y\}$, όπου x, y οι κορυφές που αντιστοιχούν στα βαθύτερα φύλλα του δένδρου. Το τελευταίο μπορούμε να το ανακτήσουμε αν από το T διαγράψουμε τις κορυφές αυτές μαζί με τον άμεσα πρόγονο τους στο δένδρο και συγχωνεύσουμε την εσωτερική κορυφή που συνδέει το φύλλο που αντιστοιχεί στην κορυφή $p = \operatorname{argmin}\{f(c) : c \in C \setminus \{x, y\}\}$, με τον άμεσο πρόγονο της. Αν, για παράδειγμα, εφαρμόσουμε τα παραπάνω στο δένδρο του Σχήματος 15.2 θα πάρουμε το δένδρο του Σχήματος 15.3.

Σχήμα 15.4 Δένδρο Huffman για το αλφάβητο του Πίνακα 15.2 χωρίς τους χαρακτήρες ‘ε’ και ‘ζ’



Αποδημητικοί
κύρους

14/12/2023

Chapter 15a

	a	b	γ	δ	ε	ζ	T
Συντήρηση	344	96	12	16	9	5	
ΣΤΙΔ. Μήκος	000	00L	020	02L	100	10L	
Μετ. Μήκος	L1	L0	000	010	01L	00L	κανός ρίζα είναι το αριθμόν

1^η loop [1) Παίρνουμε σε κάθε λεπτό το δείγμα με την μεγαλύτερη είτα T_{S1} . Τα αριθμούμε και σημειωθήσουμε την συντήρηση T_{S2}]

$$T = \{ \textcircled{344}, \textcircled{96}, \textcircled{12}, \textcircled{16}, \textcircled{9}, \textcircled{5} \}$$

$$|T| = 6, T_{S1} = \textcircled{5}, T_{S2} = \textcircled{9}$$

2^η loop

$$T = \{ \textcircled{344}, \textcircled{96}, \textcircled{12}, \textcircled{16}, \textcircled{9}, \textcircled{5} \}$$

$$|T| = 5, T_{S1} = \textcircled{12}, T_{S2} = \textcircled{14}$$

3^η loop

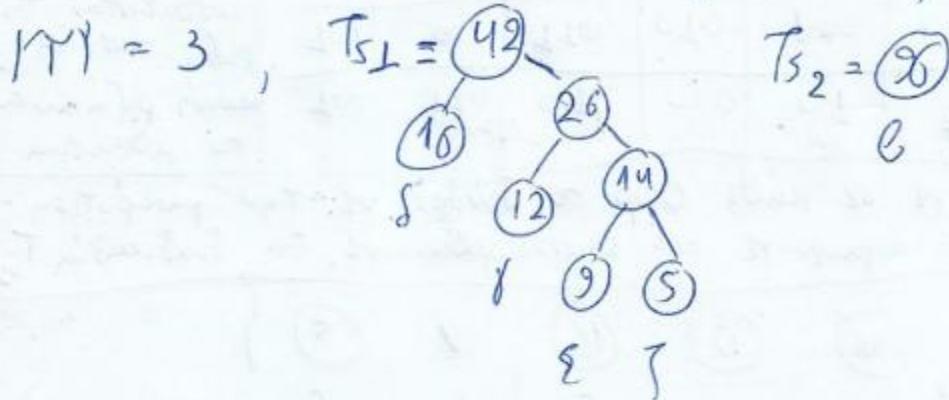
$$T = \{ \textcircled{344}, \textcircled{96}, \textcircled{16}, \textcircled{12}, \textcircled{9}, \textcircled{5} \}$$

$$|T| = 4, T_{S1} = \textcircled{16}, T_{S2} = \textcircled{26}$$

4 = loop

$$\Gamma = \{ \textcircled{344}, \textcircled{96}, \textcircled{16}, \textcircled{12}, \textcircled{9}, \textcircled{5} \}$$

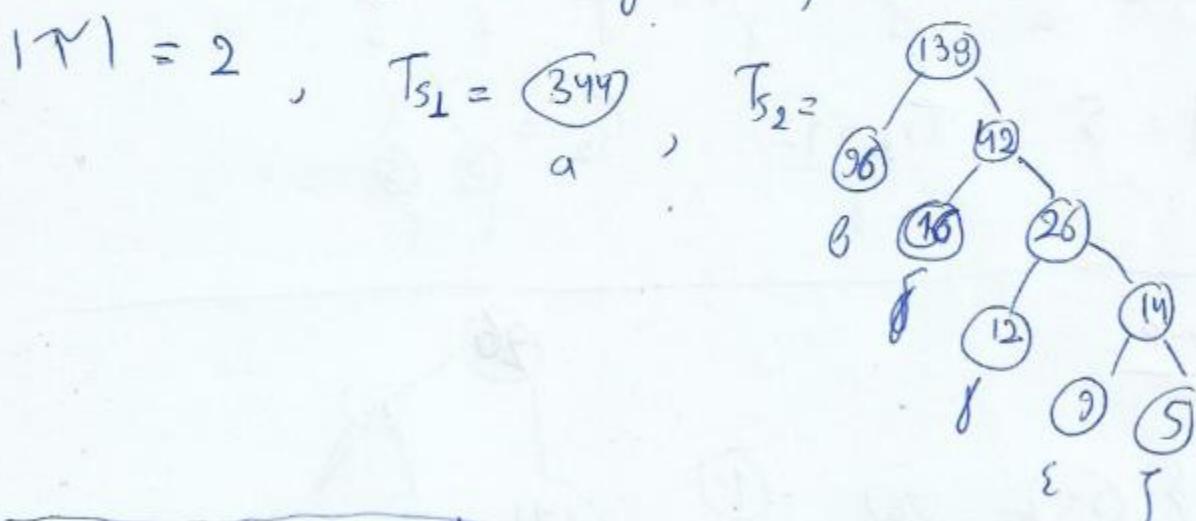
a b c d e f



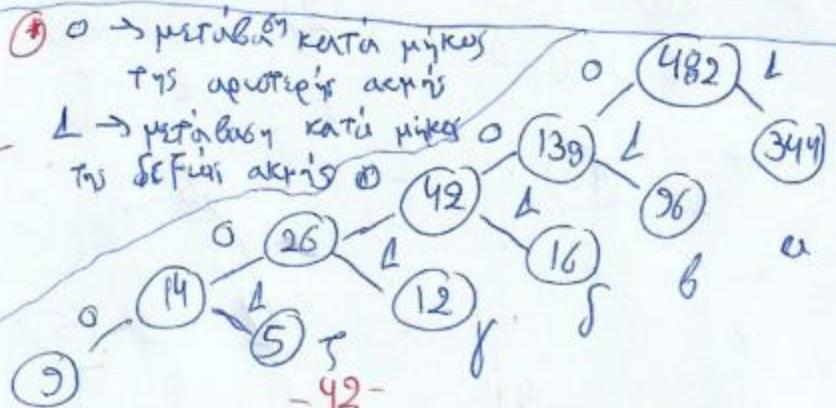
5 = loop

$$\Gamma = \{ \textcircled{344}, \textcircled{96}, \textcircled{16}, \textcircled{12}, \textcircled{9}, \textcircled{5} \}$$

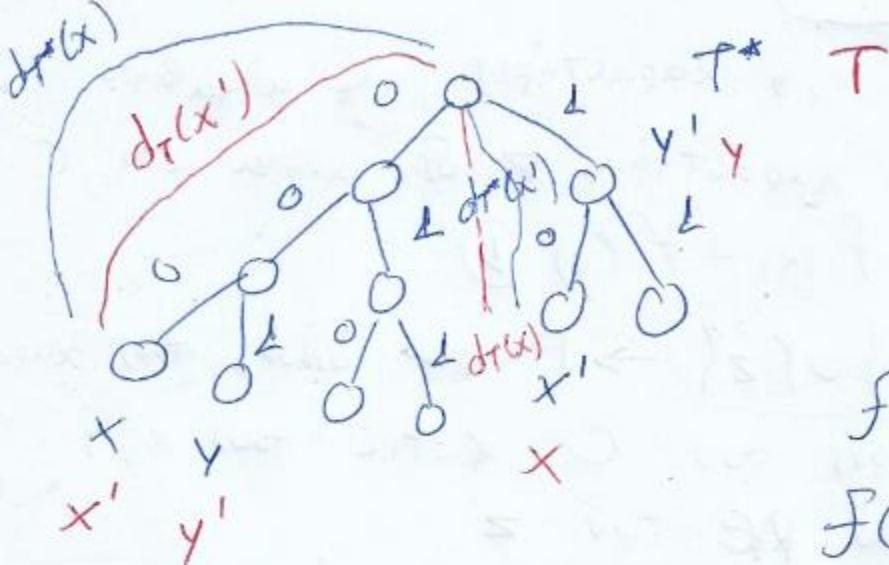
a b c d e f



Dιάδρομος Huffman



[Anology Entropy]



$$f(x) \leq f(x')$$

$$f(x') \leq f(y')$$

$$\begin{aligned}
 B(T^*) - B(T) &= \sum_{c \in C} f(c) \cdot d_T(c) - \sum_{c \in C} f(c) \cdot d_T(c) \\
 &= f(x) (d_{T^*}(x') - d_T(x')) + f(x) (d_{T^*}(x) - d_T(x)) \\
 &\quad + f(y') (d_{T^*}(y') - d_T(y')) + f(y) (d_{T^*}(y) - d_T(y))
 \end{aligned}
 \tag{1}$$

~~Επειδή η διαφορά των πηγών είναι μεγαλύτερη στην δέντρο T*~~

$$d_{T^*}(x) - d_T(x) = d_T(x') - d_{T^*}(x') \quad (=)$$

$$d_{T^*}(x) - d_T(x) = - (d_{T^*}(x') - d_T(x')) \quad (2)$$

Παρόμοια καίσει

$$d_{T^*}(y) - d_T(y) = - (d_{T^*}(y') - d_T(y')) \quad (3)$$

$$\begin{aligned}
 \text{Άρω } (2) + (3) \Rightarrow (1) : B(T^*) - B(T) &= (f(x) - f(x')) (d_{T^*}(x') \\
 &\quad - d_T(x')) + (f(y') - f(y)) (d_{T^*}(y') - d_T(y'))
 \end{aligned}$$

$\Rightarrow B(T^*) - B(T) \geq 0 \rightarrow$ Η καίσει για μικρότερη της αναλογίας
ενέργειας, καθώς T είναι το δέντρο που επιλέγει ως αλγόριθμος
και T^* είναι σεντρούς.

Βιδιτού γενομονί

$x, y \in C$ \rightarrow x, y χαρακτήρες, C αλγίδης
 $z \notin C$ \rightarrow οι χαρακτήρες z δεν ανήκουν στο C
 Τόχων $f(z) = f(x) + f(y)$ ①

$C' = C \setminus \{x, y\} \cup \{z\}$ \rightarrow Πάρων οίλις τους χαρακτήρες των C χαρακτήρες του x αλλιών \neq του z ,

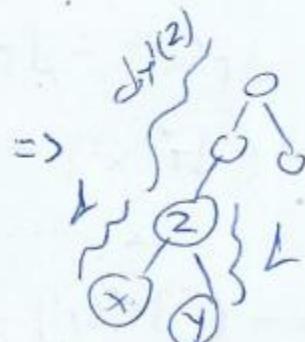
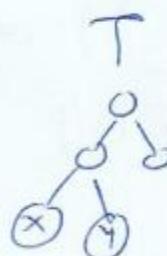
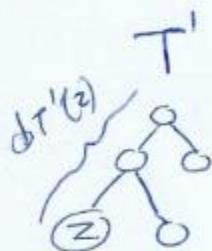
$T' \rightarrow$ το διάτροφο σύνδεσης για το C'

T \rightarrow το διάτροφο σύνδεσης για το C , ~~παραλλαγή~~



②

$$f(x)d_T(x) + f(y)d_T(y) = (f(x) + f(y)) \left(d_{T'}(z) + L \right)$$



②

$$\begin{aligned} d_T(x) &= d_{T'}(z) + L \\ d_T(y) &= d_{T'}(z) + L \end{aligned} \quad \Rightarrow \quad f(x)(d_{T'}(z) + L) + f(y)(d_{T'}(z) + L)$$

15b. Ο μικρότερος αριθμός κλειστών διαστημάτων μοναδιαίας ακτίνας σ' ένα σύνολο

Διαφάνειες

Slides_and_Exercises→11_greedyexer.pdf σελ. 1-2

Εκφώνηση

Άσκηση 1 Θεωρούμε ένα συνόλο από τιμές $V = \{v_1 \leq v_2 \leq \dots \leq v_n\}$ στο σύνολο των πραγματικών. Να περιγράψετε έναν αλγόριθμο που να υπολογίζει το μικρότερο αριθμό κλειστών διαστημάτων μοναδιαίας ακτίνας που να περιέχει όλες τις τιμές του συνόλου.

Κριτήριο Τερματισμού

Λύση Ο αλγόριθμος που προτείνουμε λειτουργεί άπληστα ως εξής. Αρχικά επιλέγουμε το διάστημα $[v_1, v_1 + 1]$. Διαγράφουμε από το V όλες τις τιμές που περιέχονται σε αυτό το διάστημα και επαναλαμβάνουμε.

Αλγόριθμος

Algorithm 1 Άπληστος αλγόριθμος για την κάλυψη τιμών από μοναδιαία διαστήματα

- Άσκηση 1

```
1:  $i \leftarrow 1$ ;
2:  $num\_intervs \leftarrow 0$ ;
3: while  $i \leq n$  do
4:    $a \leftarrow V[i]$ ;  $b \leftarrow V[i] + 1$ ;
5:    $num\_intervs \leftarrow num\_intervs + 1$ ;
6:   while  $V[i] \leq b$  do
7:      $i \leftarrow i + 1$ ;
8:   end while
9: end while
10: Print  $num\_intervs$ 
```

Πολυπλοκότητα

Ο Αλγόριθμος 1 υλοποιεί σε $O(n)$ βήματα την ιδέα.

Απληστη Επιλογή

$[v_1, v_1 + 1]$ γιατί τότε δεν θα καλύπτει την τιμή v_1 . Επομένως θεωρούμε μόνο την περίπτωση όπου το διάστημα αυτό θα βρίσκεται αριστερότερα από το $[v_1, v_1 + 1]$. Αν αντικαταστήσουμε το διάστημα αυτό $[s, s + 1]$ στη λύση S με το διάστημα $[v_1, v_1 + 1]$ παράγουμε τη λύση S' . Η περιοχή που καλύπτεται από το $[s, s + 1]$ και όχι από το $[v_1, v_1 + 1]$ είναι η $[s, v_1]$. Όμως κανένα στοιχείο του συνόλου V δεν βρίσκεται σε αυτό το διάστημα αφού το μικρότερο στοιχείο είναι το v_1 . Άρα η λύση S' έχει τον ίδιο αριθμό διαστημάτων με την S και άρα είναι και αυτή βέλτιστη.

Βέλτιστη Υποδομή

(Βέλτιστη Υποδομή) Έστω P το αρχικό πρόβλημα με βέλτιστη λύση S . Αφού θα περιληφθεί το διάστημα $[v_1, v_1 + 1]$ μένει η επίλυση του υπόλοιπου προβλήματος P' που αφορά την κάλυψη των υπόλοιπων τιμών που δεν καλύπτονται από το πρώτο διάστημα. Έστω S' η λύση του προβλήματος P' . Επειδή $|S| = |S'| + 1$ η βέλτιστη λύση του P περιλαμβάνει τη βέλτιστη λύση του P' .

Ο μεγαλύτερος αριθμός κλειστών
Συστημάτων μοναδικής ακίνητης σ' είναι

Chapter 15b

Θέλω να βρω Συστήματα τύπου $[a, b]$,

όπου θέλω το κεντρίγκεων $b-a=1$

Άρι πιέρνω Συστήματα τύπου $[a, a+1]$

$$V = \left\{ \frac{1.1}{\boxed{[1.1, 2.1]}}, 1.5, 1.8, \frac{2.1}{\boxed{[2.5, 3.5]}}, 2.7, 2.9, 3.2, \frac{3.5}{\boxed{[3.7, 4.7]}} \right\}$$

$$S' = \{ [1.1, 2.1], [2.5, 3.5], [3.7, 4.7] \}$$

$$|S'| = 3$$

Αναλογη Ενεργία

$$S = \{ [s, s+1], \dots \}$$

Ισχυρίζομαι ότι μπορώ να ανακαταστήσω την διέλευση
μέση για την ενέργεια που βρήκε ο αλγόριθμος, ΣΝΣ,

$$S' = \{ [v_1, v_1+1], \dots \}$$

Αναβαθμίζω ότι $|S'| \leq |S|$, σημαίνει, ότι η λύση που
παραχθεί, ο αλγόριθμος αλγόριθμος ⁽⁵⁾ έχει κατέτερη ή νέη
-45- από τη διέλευση (S)

Βέλτιστη Υποδομή

$$P = \left[\begin{array}{cccc} v_1 & v_2 & v_3 & v_4 & \dots & v_n \end{array} \right]$$

v_L v_{L+1}

$[v_L, v_{L+1}]$

P'

Το P είναι βέλτιστη λίστη της S . Το P' είναι τύπος S'
 λογότερη $|S'| = |S| + 1$, καθώς πάχνωρε τα σύντα του P' ($S'|$)
~~και η έπειτα μέτρησε τα συνάρθρωνται~~
 στη βέλτιστη λίστη των αρχικων προβλήματος P

15c. Το οδικό ταξίδι με τις λιγότερες στάσεις για ανεφοδιασμό

Διαφάνειες

Slides_and_Exercises → 11_greedyexer.pdf σελ. 2-3

Εκφώνηση

Άσκηση 2 Έστω ότι θέλουμε να ταξιδεύσουμε από την Αθήνα στη Θεσσαλονίκη με αυτοκίνητο. Η χωρητικότητα του ρεζερβουάρ μας επιτρέπει να καλύψουμε απόσταση τ χιλιομέτρων όταν είναι πλήρες. Στο χάρτη έχουμε σημειώσει τα βενζινάδικα στα οποία μπορούμε να σταματήσουμε για ανεφοδιασμό: έστω $d_1 < d_2 < \dots < d_n$ η απόσταση του βενζινάδικου $1, \dots, n$, από την Αθήνα. Ισχύει ότι $d_{i+1} - d_i \leq m$, για κάθε $i \in \{1, \dots, n-1\}$. Στοπός μας είναι να κάνουμε το ταξίδι πραγματοποιώντας όσο το δυνατόν λιγότερες στάσεις για ανεφοδιασμό. Να διατυπώσετε έναν αλγόριθμο ο οποίος επιλύει το πρόβλημα αυτό και να αποδείξετε την ορθότητα και την πολυπλοκότητα του.

Κριτήριο Τερματισμού

Λύση Η άπληστη στρατηγική υπαγορεύει να ταξιδέψουμε όσο μακρύτερα μπορούμε χωρίς να σταματήσουμε για ανεφοδιασμό. Ο Αλγόριθμος 2 υλοποιεί αυτή τη στρατηγική (θεωρούμε ότι ξεκινάμε από Αθήνα με γεμάτο ρεζερβουάρ, η απόσταση $d_1 \leq m$ και το σημείο $n+1$ αντιπροσωπεύει τον προορισμό- δηλαδή την Θεσσαλονίκη- και ισχύει ότι $d_{n+1} - d_n \leq m$).

Αλγόριθμος

Algorithm 2 Απληστος αλγόριθμος για την πραγματοποίηση των λιγότερων στάσεων ανεφοδιασμού - Άσκηση 2

- 1: $S \leftarrow \emptyset;$
- 2: $last \leftarrow 0;$
- 3: **for** $i \leftarrow 1; i \leq n+1; i++$ **do**
- 4: **if** $d[i] - last > m$ **then**
- 5: $S \leftarrow S \cup \{i-1\};$
- 6: $last \leftarrow d[i-1];$
- 7: **end if**
- 8: **end for**
- 9: Print S

Πολυπλοκότητα

Η πολυπλοκότητα του αλγόριθμου 2 είναι $\Theta(n)$. Η απόδειξη της ορθότητας ακολουθεί.

Απληστη Επιλογή

(Απληστη Επιλογή) Έστω $S = \{s_1, s_2, \dots, s_k\}$ η βέλτιστη λύση, όπου $s_j, j = 1, \dots, k$, τα σημεία ανεφοδιασμού. Έστω v το πρώτο σημείο ανεφοδιασμού του αλγόριθμου 2. Θα δείξουμε ότι υπάρχει μία βέλτιστη λύση με το σημείο v . Αν $s_1 = v$ τότε το σύνολο S αποτελεί μία τέτοια λύση. Έστω $s_1 \neq v$. Επειδή ο Αλγόριθμος 2 επιλέγει το μακρύτερο σημείο ανεφοδιασμού, έχουμε ότι $d_{s_1} < d_v$. Το σύνολο $S' = \{v, s_2, \dots, s_k\}$ αποτελεί μία παραδεκτή λύση αφού $d_{s_2} - d_v < d_{s_2} - d_{s_1} \leq m$. Επίσης παρατηρήστε ότι $|S'| = |S|$ και άρα το σύνολο S αποτελεί και αυτό βέλτιστη λύση.

Βέλτιστη Υποδομή

(Βέλτιστη Υποδομή) Η απόδειξη είναι αντίστοιχη με αυτή της Άσκησης 1.

Σημειώσεις

Το οδικό ταξίδι με
τις λιγότερες στάσεις για
ανεφοδιασμό

Chapter 15c

Αθήνα
 d_0

d_1 d_2 $d_3 \dots d_n$

$\Theta(n^2)$
 d_{n+1}

Ισχίων : $d_1 < d_2 < \dots < d_n \quad \forall i \in \{1, \dots, n\}$

$$d_{i+1} - d_i \leq m$$

$$d_{n+1} - d_n \leq m$$

Last \rightarrow Η απόσταση του τελευταίου βενζινάρικου από την
Αθήνα στο οποίο σημάτησα για ανεφοδιασμό

$S \rightarrow$ Το σίνιον με τα βενζινάρικα που σημάτησα για
ανεφοδιασμό

16a. Προγραμματισμός εργασιών για μεγιστοποίηση κέρδους

Διαφάνειες

Slides_and_Exercises→11_greedyexer.pdf σελ. 5-6

Εκφώνηση

Άσκηση 4 Έχουμε ένα σύνολο δραστηριοτήτων $I = \{1, \dots, n\}$ καθεμία από τις οποίες διαρκεί μία χρονική περίοδο. Σε κάθε χρονική περίοδο το πολύ μία δραστηριότητα μπορεί να εκτελείται. Η δραστηριότητα $i \in I$ θα αποδώσει κέρδος $p_i > 0$ αν εκκινήσει όχι μετά την χρονική περίοδο t_i , όπου t_i ένας τυχαίος πραγματικός (του μηδέν συμπεριλαμβανομένου). Σημειώνεται ότι αν κάποια δραστηριότητα εκκινήσει μετά την περίοδο t_i δεν αποφέρει καθόλου κέρδος και άρα θα μπορούσε και να μην πραγματοποιηθεί. Εκπονήστε έναν αποτελεσματικό αλγόριθμο ο οποίος να προγραμματίζει τις δραστηριότητες αποφέροντας το μεγαλύτερο κέρδος.

Κριτήριο Τερματισμού

Λύση Αρχικά παρατηρήστε ότι πάντα υπάρχει μία βέλτιστη λύση στην οποία όλες οι δραστηριότητες μπορούν να προγραμματιστούν να εκκινήσουν σε ακέραιο χρόνο. Για παράδειγμα για οποιαδήποτε βέλτιστη λύση S μπορούμε να εκκινήσουμε την πρώτη δραστηριότητα στο χρόνο 0, τη δεύτερη στο χρόνο 1 κλπ. Επομένως, σε κάθε βέλτιστη λύση, το γεγονός i θα εκκινεί όχι αργότερα από το χρόνο $\lfloor t_i \rfloor$.

Η παραπάνω παρατήρηση οδηγεί στον παρακάτω άπληστο αλγόριθμο. Ταξινομούμε τις δραστηριότητες κατά φθίνουσα σειρά σε σχέση με το πάτωμα του απαιτούμενου χρόνου έναρξης. Χωρίς βλάβη της γενικότητας θεωρούμε ότι η δραστηριότητα 1 μπορεί να εκκινήσει τελευταία, η 2 μπορεί να εκκινήσει προτελευταία κλπ. Δηλαδή έχουμε τη σειρά

$$\lfloor t_1 \rfloor \geq \lfloor t_2 \rfloor \geq \dots \geq \lfloor t_n \rfloor.$$

Στη συνέχεια για όλες τις δραστηριότητες που εκκινούν στο χρόνο $t = \lfloor t_1 \rfloor$ επιλέγουμε αυτή με το μεγαλύτερο κέρδος και την προγραμματίζουμε να εκκινήσει το χρόνο αυτό. Στη συνέχεια μειώνουμε το t και η διαδικασία επαναλαμβάνεται. Ο Αλγόριθμος 4 υλοποιεί την ιδέα.

Αλγόριθμος

Άλγοριθμος 4 Άπληστος Αλγόριθμος για τη δρομολόγηση δραστηριοτήτων- Άσκηση 4

Require: $\lfloor t_1 \rfloor \geq \lfloor t_2 \rfloor \geq \dots \geq \lfloor t_n \rfloor$

- 1: $S \leftarrow \emptyset;$
- 2: $t \leftarrow \lfloor t_1 \rfloor;$
- 3: **while** $t \geq 0$ **and** $|S| < n$ **do**
- 4: $z \leftarrow \text{argmax}\{p_i : t \leq \lfloor t_i \rfloor, i \notin S\};$
- 5: $S \leftarrow S \cup \{z\};$
- 6: $t --;$
- 7: **end while**

Πολυπλοκότητα

Ο Αλγόριθμος 4 μπορεί να εκτελεστεί σε $O(n \lg n)$ βήματα χρησιμοποιώντας ουρά προτεραιότητας στην εύρεση του μεγίστου (γραμμή 4).

Απληστη Επιλογή

(Απληστη Επιλογή) Θεωρούμε μία βέλτιστη λύση S^* και έστω k η δραστηριότητα η οποία ανήκει στο S^* και εκτελείται τελευταία. Προφανώς $\lfloor t_1 \rfloor \geq \lfloor t_k \rfloor$ αφού ο άπληστος αλγόριθμος επιλέγει τη δραστηριότητα με τον αργότερο απαιτούμενο χρόνο εκκίνησης να εκτελέσει τελευταίο. Θεωρούμε τις δύο ακόλουθες περιπτώσεις.

1. $\lfloor t_1 \rfloor = \lfloor t_k \rfloor$. Αν $1 \notin S^*$ τότε μπορούμε να αντικαταστήσουμε την k με την 1 στο S^* και να επιτύχουμε τουλάχιστον το ίδιο κέρδος (αφού $p_1 \geq p_k$) αφού $p_1 \geq p_k$ (γραμμή 4 του Αλγόριθμου 4). Αν $1 \in S^*$ αυτό σημαίνει ότι έχει δρομολογηθεί να εκκινήσει νωρίτερα από το k στη λύση S^* . Εναλλάσσοντας τους χρόνους εκκίνησης των δραστηριοτήτων 1, k , προκύπτει η βέλτιστη λύση S' η οποία έχει σαν τελευταία δραστηριότητα την 1.
2. $\lfloor t_1 \rfloor > \lfloor t_k \rfloor$. Αν $1 \notin S^*$ αφού η k είναι η τελευταία δραστηριότητα μπορούμε να προσθέσουμε στο S^* την 1 και να επιτύχουμε μεγαλύτερο συνολικό κέρδος - αντίφαση αφού S^* είναι βέλτιστη λύση. Αν $1 \in S^*$ μπορούμε να το εκκινήσουμε το χρόνο $\lfloor t_1 \rfloor$ και να προκύψει η βέλτιστη λύση S' η οποία έχει σαν τελευταία δραστηριότητα την 1.

Βέλτιστη Υποδομή

(Βέλτιστη Υποδομή) Έστω P το αρχικό πρόβλημα με βέλτιστη λύση S . Με δεδομένο ότι η δραστηριότητα 1 θα είναι η πρώτη που θα δρομολογηθεί, μένουμε με το υποπρόβλημα P' που αφορά τη δρομολόγηση των δραστηριοτήτων $2, \dots, n$. Έστω S' η βέλτιστη λύση σε αυτό. Προφανώς, το κέρδος της λύσης S ισούται με το κέρδος της λύσης S' συν p_1 . Άρα η βέλτιστη λύση του προβλήματος P περιλαμβάνει τη βέλτιστη λύση του προβλήματος P' .

(Προγραμματισμός εργασιών με κρίσεις)

19/12/2023

i	a	b	c	d	e	f
p _i	3	5	F	4	3	6
t _i	2,5	3,FS	4,28	4,F3	3,23	4,5
L _{ti}	2	3	4	4	3	4

Chapter 16a

Η διαστήσιμη ι είναι ένας πιο εφεύρεται στην ολικήρυθμη σε χρόνο $t \leq t_i$. Άριστη ολικήρυθμη σε χρόνο $t > t_i$ τοτε είναι κίνδυνος 0.

L_{ti} → Πάντα είναι ο μεγαλύτερος απόμενος που είναι μεγαλύτερος ή ίσος του t_i

Κανονικό sort σε αρθρωτικά τη L_{ti}

$$L_{tL} \geq L_{t2} \geq \dots \geq L_{th}$$

i	c	d	b	e	a	f
p _i	F	3	5	3	3	6
L _{ti}	4	4	3	3	2	4

Μηρών για εκτυπώσης αντί των χρόνων 0 και fέκτων αντί των $t_{i,j}$

$$\text{Αρχικοποίηση } t \leftarrow L_{tL} \Rightarrow t \leftarrow 4 \quad s \in \emptyset$$

Ιδηκώς ($t \leftarrow 4, L_{ti} \geq 4$)

$$z \leftarrow \arg\max \{ p_i : t \leq L_{ti}, i \in S \} \Rightarrow$$

$$z \leftarrow \arg\max \{ F, 4 \} \Rightarrow z \leftarrow 4$$

S ⊂ {e}

2 \Rightarrow loop ($t \Leftarrow 3$, $L[t,i] \geq 3$)

$$z \leftarrow \arg\max\{p_i : t \leq L[t,i], i \notin S\} \Rightarrow$$

$$z \leftarrow \arg\max\{5, 4, 3\} \Rightarrow z \leftarrow b$$

$b \quad d \quad e \quad \dots \quad \star$

$S = \{c, b\}$

Analogi Enarxi

1) Eotw b'etw twn S^* kai iotw $k \in S^*$.

2) H analogi enarxi einai ta arxika sti ~~enarxi~~
 twn S' w L exarxim, kaiis, o upoleipous
 ton twn taftiwpou, $L[t_L] \geq [t_2] \geq \dots \geq L[t_1]$
 Enarxi rjvta twn exarxi, nov teleswsi se $L[t_L]$
 xpw.

3) To S^* neperixa to b'ea stigmatites me to periptero
 appoora kipous kai einai b'eftwto

4) Ota siwtw einai ono. S' nov exei tes b'ea -
 pteres tou S^* kai twn L nov einai kai
 i analogi Enarxi

$$- L[t_L] = L[t_k] \dots$$

$$- L[t_L] > L[t_k] \dots$$

3 \Rightarrow loop ($t \Leftarrow 2$, $L[t,i] \geq 2$)

$$z \leftarrow \arg\max\{p_i : t \leq L[t,i], i \notin S\} \Rightarrow$$

$$z \leftarrow \arg\max\{3, 4, 3\} \Rightarrow z \leftarrow d$$

$a \quad d \quad e \quad -48-$

$S = \{c, b, d\}$

$\text{9} \Rightarrow \text{loop } (t \leftarrow L, [Lt_i] \geq L)$

$$z \leftarrow \operatorname{argmax}\{p_i : t \leq Lt_i, i \notin S\} \Rightarrow$$

$$z \leftarrow \operatorname{argmax}\{3, 3, 6\} \Rightarrow z \leftarrow f$$

a e f

Λύση

$$S = \{c, b, d, f\}$$

Βέβαιοτή Καθόριση

...

• O αλγόριθμος συνέχει γίγαντα την $L \Rightarrow$ διαστημάτων.

• Εστω S η βέβαιη λίστα ΗΕ την $L \Rightarrow$ διαστημάτων.

• Εστω S' η βέβαιη λίστα XΟΡΙΣ την $L \Rightarrow$

• Θέλω να αυξήσω ^{διαστημάτων} _{ότι} ωχιές $|S'| = |S| + L$

όπου είναι και η μέτρη της $|S|$ βέβαιων υποδομών.

④ $S \Leftarrow \text{loop } (t \leftarrow 0, [Lt_i] \geq 0)$

$$z \leftarrow \operatorname{argmax}\{p_i : t \leq Lt_i, i \notin S\} \quad \checkmark$$

$$z \leftarrow \operatorname{argmax}\{3, 3\} \Rightarrow z \leftarrow a$$

a e

$$S = \{c, b, d, f, a\}$$

Chapter 16

16b. Εισαγωγή

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 125-127

Βασικές Κλάσεις Προβλημάτων

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

Προβλήματα της κλάσης P

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. Σχεδόν κάθε πρόβλημα που παρουσιάσαμε στα προηγούμενα κεφάλαια εμπίπτει σε αυτή την κατηγορία. Για παράδειγμα, το πρόβλημα εύρεσης των συντομότερων μονοπατιών από μία κορυφή προς κάθε άλλη σε ένα βεβαρυμένο κατευθυνόμενο γράφημα, το πρόβλημα εύρεσης ενός ελάχιστου συνεκτικού δένδρου, το πρόβλημα του υπολογισμού ενός γινομένου πινάκων με το μικρότερο αριθμό πολλαπλασιασμών κλπ.

Προβλήματα της κλάσης EXP

Προβλήματα που ανήκουν στην κλάση 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 EXP [9] - το ίδιο ισχύει και για το αντίστοιχο πρόβλημα που αφορά την ντάμα [17].

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 αλγόριθμου με τη συγκεκριμένη είσοδο σε πεπερασμένο χρόνο; Είναι γνωστό ότι το πρόβλημα αυτό είναι εκτός των ορίων του υπολογισμού [10, 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.

Το πρόβλημα τερματισμού

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

1. ΝΑΙ αν ο αλγόριθμος τερματίζει,
2. ΟΧΙ αν δεν τερματίζει.

Προβλήματα απόφασης

Τυπικά οι προαναφερθείσες κλάσεις προβλημάτων αφορούν προβλήματα τέτοιου τύπου, 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 ;

Προβλήματα απόφασης που δεν ανήκουν στο R

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 εκτελέσιμο αρχείο ενός προγράμματος είναι μία πεπερασμένη συμβολοσειρά από σύμβολα του συνόλου $\{0, 1\}$. Άρα κάθε αλγόριθμος έχει 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 $\{\text{ΝΑΙ} = 1, \text{ΟΧΙ} = 0\}$. 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.

Το πρόβλημα του ΕΣΔ ως πρόβλημα απόφασης

Παρόλο που οι κλάσεις των προβλημάτων P, EXP, R τυπικά αφορούν προβλήματα απόφασης, είναι κοινή πρακτική να κατατάσσουμε και προβλήματα άλλων τύπων στις κλάσεις αυτές. Για παράδειγμα, το πρόβλημα εύρεσης ενός ελάχιστου συνεκτικού δένδρου σε ένα βεβαρυμένο συνδεδεμένο γράφημα, ονομαστικά ΕΣΔ, 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 είναι

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 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.² 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 ως) ένα πρόβλημα απόφασης». Επομένως η ιεραρχία των κλάσεων που αφορά τα προβλήματα απόφασης, θα μπορούσε να αφορά οποιοδήποτε πρόβλημα υπολογισμού.

Chapter 17

17a. Η κλάση NP

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 127-129

«Τυχερός» Αλγόριθμος

Η ταξινόμηση των προβλημάτων απόφασης δεν εξαντλείται στις κλάσεις που παρατέθηκαν στην προηγουμένη ενότητα. Ίσως η σημαντικότερη κλάση η οποία εμπεριέχει 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 της κλάσης NP

Ορισμός 14. Το σύνολο των προβλημάτων απόφασης για τα οποία η λύση των NAI-στιγμιοτύπων μπορεί να επαληθευτεί σε πολυωνυμικό χρόνο συνιστά την τάξη NP.

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

Στιγμιότυπο ΕΣΔ(ΑΠΟΦΑΣΗ)

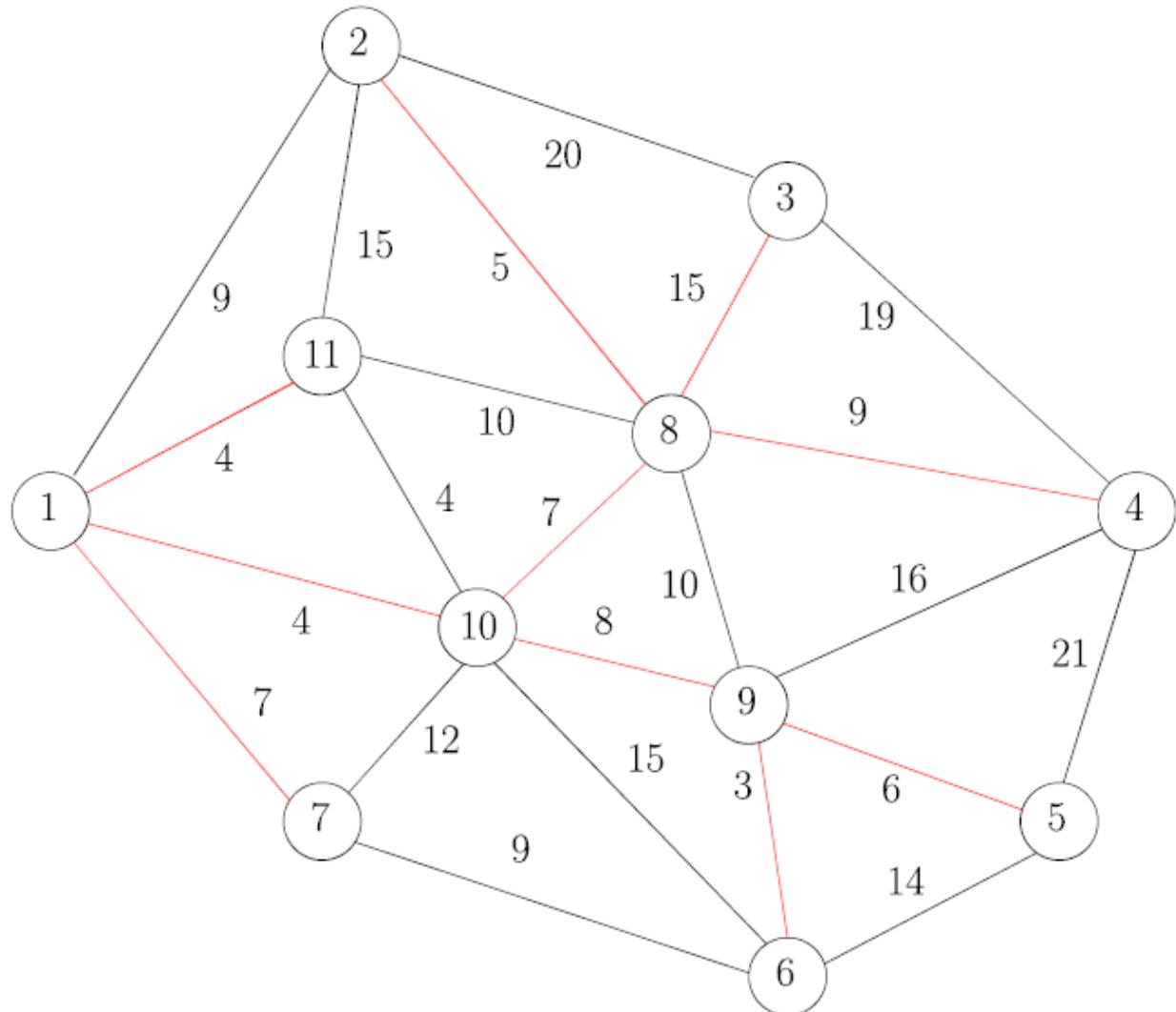
Για παράδειγμα, θεωρούμε το στιγμιότυπο του προβλήματος ΕΣΔ(ΑΠΟΦΑΣΗ) όπου το βεβαρυμένο γράφημα $G(V, E, w)$ απεικονίζεται στο Σχήμα 16.1 και ο ακέραιος k είναι ίσος με 80. Η απάντηση στο στιγμιότυπο αυτό είναι NAI και τεκμηριώνεται από το πιστοποιητικό που αποτελεί το ελάχιστο συ-

συνεκτικό δένδρο $T(V, E)$ με

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

$$\begin{aligned} E(T) = & \{\{1, 7\}, \{1, 10\}, \{1, 11\}, \{2, 8\}, \{3, 8\}, \\ & \{4, 8\}, \{5, 9\}, \{6, 9\}, \{8, 10\}, \{9, 10\}\}. \end{aligned}$$

Σχήμα 16.1 Βεβαρημένο γράφημα με το ελάχιστο συνεκτικό δένδρο του



Πολυπλοκότητα ΕΣΔ(ΑΠΟΦΑΣΗ)

Πράγματι μπορούμε να διαπιστώσουμε σε χρόνο $O(|E(T)|)$ ότι το σύνολο $E(T)$ αποτελείται από $|V(T)| - 1 = 11 - 1 = 10$ ακμές του συγκεκριμένου γραφήματος. Στη συνέχεια εκτελώντας στο T διάσχιση κατά βάθος μπορούμε σε $O(|V(T)| + |E(T)|)$ 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 $O(|V(T)|)$ ΣΥΒ (συγκεκριμένα, $|V(T)| - 2 = 11 - 2 = 9$ 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 στιγμότυπου.

Σχέση της κλάσης P με τη NP

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 $P \equiv NP$.

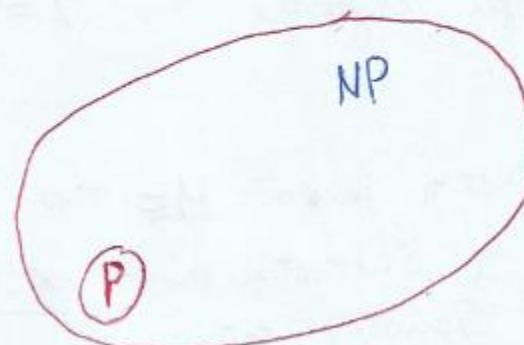
ΟΒ_ΚΛΑΣΕΙΣ ΠΡΟΒΛΗΜΑΤΩΝ

H κλασίς NP

Chapter IIa

21/12/2023

- Το πρόβλημα ανήκει στην κλασή NP αν για κάπει στιγμιότυπο ΝΑΙ υπάρχει εικ. ποσούντυπό επακίνδυνος
- Χρειάζεται «τυχερός» αλγόριθμος να να δράσει λίγων για το πρόβλημα



Δεν γνωρίζουμε αν η κλασή P εμφανίζεται αντιτρόπως στην κλασή NP

$$P \subsetneq NP \quad ; \quad P = NP$$

17b. Αναγωγές

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 129-130

Ιδέα

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 του B είναι πολυωνυμική συνάρτηση του μεγέθους του στιγμιότυπου του A .

Πίνακας 16.1 Παραδείγματα αναγωγής $A \leq_P B$

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 $(u, v) \in \bar{E}$ αν και μόνο αν $(v, u) \in E$

Η κλάση NP-hard

Η έννοια της αναγωγής μπορεί να χαρακτηρίσει προβλήματα ανάλογα με τη «δυσκολία» που παρουσιάζουν. Έστω B ένα πρόβλημα με την ιδιότητα οποιοδήποτε πρόβλημα μίας κλάσης να μπορεί να αναχθεί σε αυτό. Τότε το πρόβλημα χαρακτηρίζεται «δύσκολο» (hard) αφού είναι τουλάχιστο τόσο δύσκολο να επιλυθεί όσο το πιο δύσκολο πρόβλημα της συγκεκριμένης κλάσης. Αν, για παράδειγμα, οποιοδήποτε πρόβλημα της κλάσης NP μπορεί να αναχθεί στο B τότε το πρόβλημα B λέμε ότι ανήκει στην κλάση - είναι - NP-hard. Αν επιπλέον το πρόβλημα B ανήκει και στην κλάση NP τότε το B ανήκει στην κλάση 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.

Αναρρίχηση

①

polynomial

 $A \leq_p B$

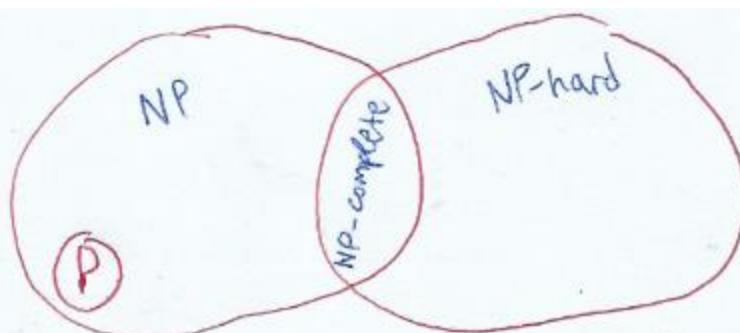
$$\begin{array}{ll} a_1 & \rightarrow b_1 \\ a_2 & b_2 \\ a_3 & b_3 \\ \vdots & \vdots \\ a_n & b_n \end{array}$$

 $A, B \rightarrow$ προβλήματα μάθημας $a_1 \dots a_n \rightarrow$ στυγμένα προβλήματα A $b_1 \dots b_n \rightarrow$ στυγμένα προβλήματα BΤα προβλήματα B λίγες από κάποιαν αλγόριθμο (γρυντό)• Αν κάποια στυγμένη $a_1 \dots a_n$ \rightarrow το πρόβλημα A δεν φέρει απόλυτη λύση, δίνει παρατυγμένη $b_1 \dots b_n$. Τα από κάποιαν αλγόριθμο (γρυντό)

κατατάσσεται σε πολυνυμική

είγηση, τότε λογκά η αναρρίχηση ①

- 50 -



NP-hard: Ανήκουν ~~δεν~~ τα προβλήματα, από
ξεων την ωδήση ότι συνωμβινούν πρόβλημα
της κλασής NP γιατρίνα και αναχθεί σ' ειναντί
αυτά. Έ.χ. $A \in NP$ και $A \leq_p B \Rightarrow$
 $B \in NP\text{-hard}$

NP-complete: Αν $B \in NP$ και $B \in NP\text{-hard} \Rightarrow$
 $B \in NP\text{-complete}$

Τα από σύσκεψη προβλήματα της τάξης NP

17c. Παρατήρηση ως προς την πολυπλοκότητα

Σημειώσεις

{ Παρατήρηση ως προς την πολυπλοκότητα } Chapter 1Fc

→ Ισχυρά πολυωνυμικοί αλγόριθμοι: Ο χρόνος εκτέλεσης είναι πολυωνυμικός τόσο στο μέγεθος των παραμέτρων όσο και στο μέγεθος των αριθμών ή συμβολών στις παραμέτρους. Π.χ.: αλγόριθμοι ταφιώμοις $O(n)$

→ Ψευδοπολυωνυμικοί αλγόριθμοι: Ο χρόνος εκτέλεσης είναι πολυωνυμικός στο μέγευος των παραμέτρων, αλλά όχι στο μέγευος των αριθμών ή συμβολών των παραμέτρων. Π.χ.: Divisors $O(\sqrt{n})$

-5L-

Chapter 18

18a. Η κλάση NP-complete

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 130-131

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 $P \not\subseteq NP$ τότε $NP\text{-complete} \subset NP$ και $P \cup NP\text{-complete} \subseteq NP$ [16, Θεώρημα 14.1].

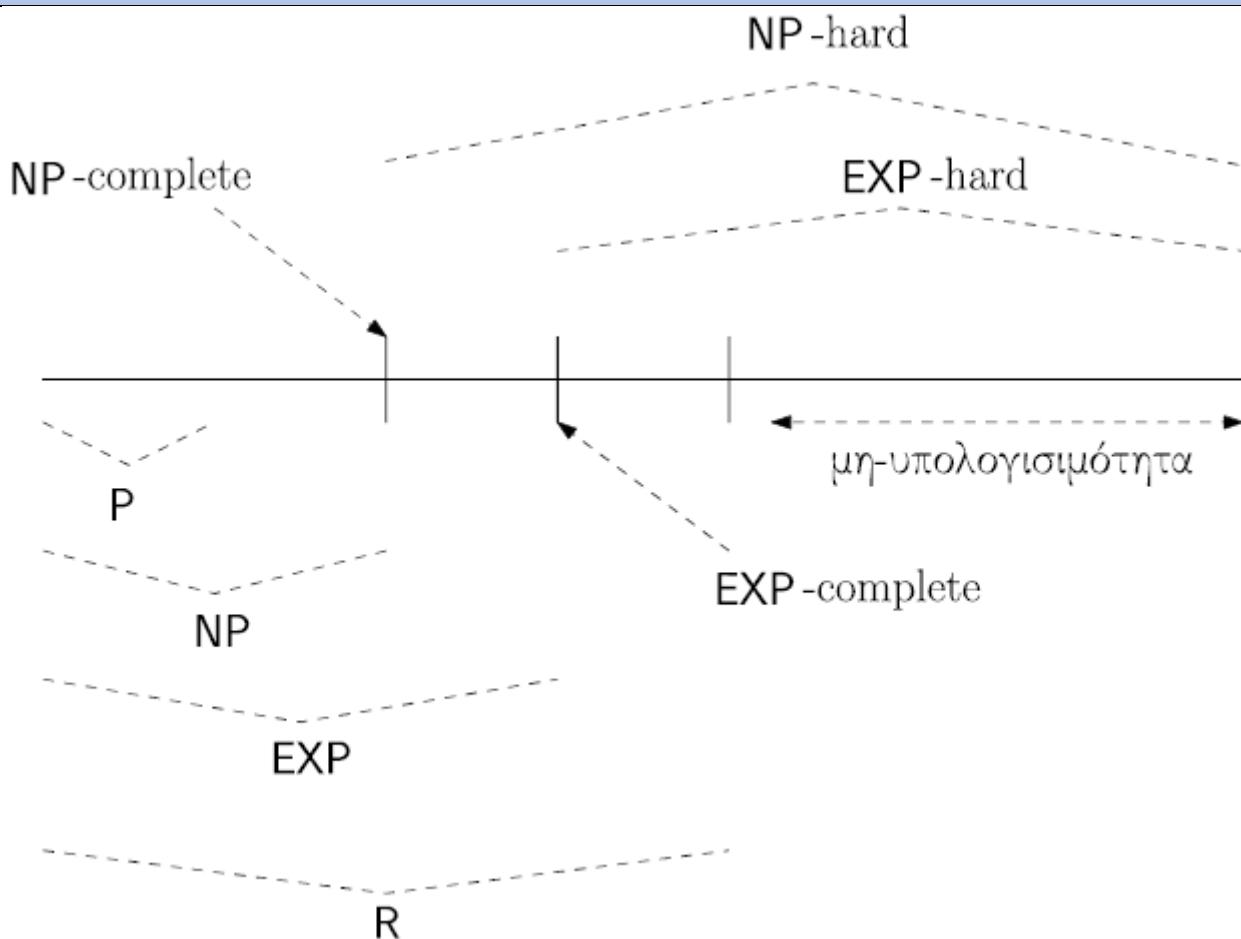
Μεθοδολογία απόδειξης ότι $B \in NP\text{-complete}$

Τα παραπάνω καταδεικνύουν γιατί είναι σημαντικό να μπορούμε να αναγνωρίσουμε αν κάποιο πρόβλημα B ανήκει στην τάξη $NP\text{-complete}$. 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 $NP\text{-complete}$.

Το δεύτερο βήμα της παραπάνω διαδικασίας αποτελείται από την απόδειξη της πρότασης $B \in \text{NP-hard}$ 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 B έχει απάντηση ΝΑΙ. Επιπλέον, θα πρέπει το μέγεθος του στιγμιότυπου του προβλήματος B να είναι πολυωνυμικό σε σχέση με το μέγεθος του στιγμιότυπου του προβλήματος A .

Σχήμα 16.2 Κλάσεις προβλημάτων



Σημειώσεις

(Μεθόδος για Ανίστρητη NP-complete)
Για να αποδώσει ότι ένα πρόβλημα

$B \in \text{NP-complete}$ πρέπει να αποδίξει:

9/1/2024

Chapter 18a

1

$B \in \text{NP}$

ⓐ στηγμιότυπον του B είναι οχι

2.3] Το μέγεθος των στηγμιότυπων του B
είναι πολυωνυμικό

2

$A \leq_p B \iff B \in \text{NP-hard}$

2.1] Αν Το στηγμιότυπον του A είναι ΝΑΙ αν και μόνο αν
Το στηγμιότυπον του B είναι ΝΑΙ

2.2] Το στηγμιότυπον του A είναι οχι αν και μόνο αν Το

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 131-132

Περιγραφή

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 συζευκτική 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

$$\phi_1 = (x_1 \vee \bar{x}_2) \wedge (x_1 \wedge \bar{x}_4 \wedge x_3), \quad (16.1)$$

$$\phi_2 = (x_2 \wedge x_4) \vee (\bar{x}_1 \wedge x_3 \wedge \bar{x}_2), \quad (16.2)$$

$$\phi_3 = (\bar{x}_3 \vee \bar{x}_2 \vee x_4) \wedge (x_2 \vee x_1 \vee \bar{x}_4). \quad (16.3)$$

ΚΣΜ (Κανονική Συζευκτική Μορφή)

Θα αναφερόμαστε σε κάθε υπο-τύπο που εμπεριέχεται μέσα σε μια παρένθεση ως (λογική) πρόταση. Για παράδειγμα, στον τύπο που ορίζεται στην (16.1), προτάσεις αποτελούν οι $(x_1 \vee \bar{x}_2)$ και $(x_1 \wedge \bar{x}_4 \wedge x_3)$. Ένας τύπος είναι σε κανονική συζευκτική μορφή αν αποτελεί σύζευξη προτάσεων μέσα στις οποίες οι όροι συνδέονται με διάζευξη. Ο τύπος που ορίζεται από την (16.3) είναι σε κανονική συζευκτική μορφή. Ο συμμετρικός ορισμός αποτελεί δίνει τύπους σε κανονική διαζευκτική μορφή: έχουμε διάζευξη προτάσεων στις οποίες οι όροι συνδέονται με σύζευξη, πχ., τύπος που ορίζεται από την (16.2).

Απόδοση τιμών

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 SAT αφορά ακριβώς αυτό.

Διατύπωση

SAT

Στιγμιότυπο: Τύπος φ σε κανονική συζευκτική μορφή.

Ερώτηση: Είναι ο φ ικανοποιήσιμος;

Σημειώσεις

(Το πρόβλημα SAT)

Chapter 19b

SAT ∈ NP-complete

(Το Το πρόβλημα να αποτελέσθη
ότι ανήκει στην κατηγορία NP-complete)

Λ → AND

∨ → OR

$$\varphi = (\underline{x_1 \vee x_2 \vee x_3}) \wedge (\underline{\underline{x_4 \vee x_5 \vee x_6}})$$

Κανονική συζευκτική μορφή

Έτσού προτάσεων όπου οι όροι επδιένταν με διάταξη

18c. Το πρόβλημα 3SAT

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 132-135

Περιγραφή

Το πρόβλημα 3SAT είναι μία περιορισμένη έκδοση του SAT όπου οι προτάσεις που συνιστούν τον τύπο φ περιέχουν ακριβώς τρεις όρους για παράδειγμα, ο τύπος που περιγράφεται από την (16.3) είναι τέτοιας μορφής. Τυπικά το πρόβλημα περιγράφεται ως εξής.

Διατύπωση

3SAT

Στιγμιότυπο: Τύπος φ σε κανονική συζευκτική μορφή όπου σε κάθε πρόταση εμφανίζονται τρεις όροι.

Ερώτηση: Είναι ο φ ικανοποιήσιμος;

3SAT ∈ NP

Απόδειξη. Θεωρούμε ότι ο τύπος φ αποτελεί σύζευξη m προτάσεων, όπου $m \in \mathbb{N}$, σε κάθε μία από τις οποίες περιέχονται ακριβώς τρεις όροι. Αν συμβολίσουμε την

i -οστή πρόταση ως ϕ^i τότε μπορούμε να γράψουμε τον τύπο ως $\phi = \wedge_{i=1}^m \phi^i$, όπου $\phi^i = \lambda_1^i \vee \lambda_2^i \vee \lambda_3^i$, και $\lambda_1^i, \lambda_2^i, \lambda_3^i$ οι τρεις όροι της πρότασης. Επομένως, αν αποδώσουμε τιμές στους όρους, για να αποτιμήσουμε κάθε πρόταση ϕ^i εκτελούμε δύο ΣΥΒ - δυο λογικές πράξεις διάζευξης. Στη συνέχεια χρειάζεται να εκτελέσουμε $m-1$ πράξεις σύζευξης για να αποτιμήσουμε τον τύπο ϕ . Άρα για να ελέγχουμε αν μία απόδοση τιμών στους όρους κάνει τον τύπο ϕ True εκτελούνται $2m$ - δηλαδή $\Theta(m)$ - ΣΥΒ. \square

3SAT ∈ NP – hard

Απόδειξη. Θα δείξουμε αναγωγή από το πρόβλημα SAT. Δηλαδή από ένα τυχαίο στιγμιότυπο του προβλήματος SAT θα κατασκευάσουμε ένα στιγμιότυπο του προβλήματος 3SAT κατά τρόπο ώστε το στιγμιότυπο του SAT θα είναι ΝΑΙ αν και μόνο αν το ίδιο συμβαίνει για το στιγμιότυπο 3SAT. Επιπλέον θα δείξουμε ότι το στιγμιότυπο 3SAT έχει πολυωνυμικό μέγεθος σε σχέση με αυτό του στιγμιότυπου SAT.

SAT ≤_P 3SAT

Το στιγμιότυπο SAT είναι ΝΑΙ/OΞΙ ANN το στιγμιότυπο 3SAT είναι ΝΑΙ/OΞΙ

Έστω τύπος $\hat{\phi}$ ο οποίος αποτελεί στιγμιότυπο του προβλήματος SAT. Θεωρούμε ότι ο τύπος αποτελεί τη σύζευξη m διαζευκτικών προτάσεων, δηλαδή

$$\hat{\phi} = \hat{\phi}^1 \wedge \hat{\phi}^2 \wedge \cdots \wedge \hat{\phi}^m,$$

όπου

$$\hat{\phi}^i = \lambda_1^i \vee \lambda_2^i \vee \cdots \vee \lambda_t^i, \forall i \in \{1, \dots, m\}.$$

Δηλαδή η πρόταση $\hat{\phi}^i$ αποτελείται από τη διάζευξη t όρων, ήτοι των $\lambda_1^i, \lambda_2^i, \dots, \lambda_t^i$.

Για κάθε πρόταση $\hat{\phi}^i$, θα γράψουμε είτε μία πρόταση ή μία σύζευξη προτάσεων, ονομαστικά ϕ^i , με τις ιδιότητες

1. η πρόταση (προτάσεις της) ϕ^i να αποτελούνται από τρεις όρους η κάθε μία και
2. υπάρχει αποτίμηση True για την $\hat{\phi}^i$ αν και μόνο αν το ίδιο συμβαίνει για την ϕ^i .

Παρατηρούμε ότι η δεύτερη ιδιότητα διασφαλίζει ότι το παραγόμενο στιγμιότυπο 3SAT είναι ΝΑΙ-στιγμιότυπο αν και μόνο αν το στιγμιότυπο SAT είναι ΝΑΙ-στιγμιότυπο.

Διακρίνουμε τις παρακάτω περιπτώσεις σε σχέση με το πλήθος των όρων (παράμετρος t) που περιέχει η $\hat{\phi}^i$.

$t = 1$: Η ϕ^i περιέχει τη σύζευξη του ιδίου όρου:

$$\phi^i = \lambda_1^i \vee \lambda_2^i \vee \lambda_3^i.$$

$t = 2$: Η ϕ^i αποτελείται από τη σύζευξη των όρων της $\hat{\phi}^i$ επαυξημένη με τον πρώτο από τους δύο:

$$\phi^i = \lambda_1^i \vee \lambda_2^i \vee \lambda_1^i.$$

$t = 3$: Η ϕ^i είναι ακριβώς ίδια με την $\hat{\phi}^i$:

$$\phi^i = \hat{\phi}^i = \lambda_1^i \vee \lambda_2^i \vee \lambda_3^i.$$

$t \geq 4$: Για να γράψουμε την ϕ^i θα χρησιμοποιήσουμε $t - 3$ νέες μεταβλητές - μεταβλητές που δεν εμφανίζονται στον τύπο $\hat{\phi}$ - ονομαστικά z_1, \dots, z_{t-3} . Η ϕ^i θα αποτελείται από τη σύζευξη $t - 4 + 2 = t - 2$ προτάσεων στις οποίες χρησιμοποιούνται $2(t - 3)$ νέοι όροι, ήτοι

$$\begin{aligned} \phi^i = & (\lambda_1^i \vee \lambda_2^i \vee z_1) \wedge (\bar{z}_1 \vee \lambda_3^i \vee z_2) \wedge \\ & \cdots \wedge (\bar{z}_{k-2} \vee \lambda_k^i \vee z_{k-1}) \wedge \cdots \\ & (\bar{z}_{t-4} \vee \lambda_{t-2}^i \vee z_{t-3}) \wedge (\bar{z}_{t-3} \vee \lambda_{t-1}^i \vee \lambda_t^i). \end{aligned} \quad (16.4)$$

Για $t \leq 3$, είναι προφανές ότι ϕ^i αποτιμάται σε True αν και μόνο αν το ίδιο συμβαίνει για την $\hat{\phi}^i$. Θα δείξουμε ότι αυτό ισχύει και στην περίπτωση κατά την οποία $t \geq 4$. Για την περίπτωση αυτή, αρχικώς, θα δείξουμε ότι αν $\hat{\phi}^i$ αποτιμάται σε True μπορούμε να δώσουμε (λογικές) τιμές στους νέους όρους ώστε και ϕ^i να αποτιμάται σε True. Παρατηρούμε ότι για να αποτιμάται $\hat{\phi}^i$ σε True θα πρέπει ένας τουλάχιστον από τους όρους της - δηλαδή ένας από τους όρους $\lambda_1^i, \lambda_2^i, \dots, \lambda_t^i$ - να έχει τιμή True.

Αν $t > 4$, χωρίς βλάβη της γενικότητας θεωρούμε ότι ο όρος αυτός είναι ο λ_k - δες (16.4). Θέτουμε,

$$z_j = \text{True}, \forall j = 1, \dots, k - 2, \quad (16.5)$$

$$z_j = \text{False}, \forall j = k - 1, \dots, t - 3. \quad (16.6)$$

Παρατηρούμε ότι από την παραπάνω ανάθεση τιμών όλες οι προτάσεις που προηγούνται της $(\bar{z}_{k-2} \vee \lambda_k^i \vee z_{k-1})$ στην (16.4) αποτιμώνται σε True αφού σε κάθε μία ο όρος z_j έχει τιμή True. Αντίστοιχα, για όλες τις προτάσεις που βρίσκονται δεξιότερα της παραπάνω πρότασης αποτιμώνται σε True αφού σε κάθε μία ο όρος \bar{z}_j έχει τιμή True. Επίσης η πρόταση $(\bar{z}_{k-2} \vee \lambda_k^i \vee z_{k-1})$ αποτιμάται σε True αφού (εξ' υποθέσεως) ο όρος $\lambda_k^i = \text{True}$.

Αν $t = 4$, η (16.4) γράφεται ως

$$\phi^i = (\lambda_1^i \vee \lambda_2^i \vee z_1) \wedge (\bar{z}_1 \vee \lambda_3^i \vee \lambda_4^i)$$

Αν ο όρος που αποτιμάται σε True είναι κάποιος από τους λ_3^i, λ_4^i η z_1 παίρνει τιμή από την (16.5), διαφορετικά από την (16.6).

Αρα όταν $t \geq 4$, η απόδοση τιμών από τις (16.5), (16.6), διασφαλίζει ότι η ϕ^i να αποτιμάται σε True όταν $\hat{\phi}^i$ αποτιμάται σε True. Στη συνέχεια θα δείξουμε το αντίστροφο: αν $\hat{\phi}^i$ αποτιμάται σε False δεν υπάρχει απόδοση τιμών στους όρους z_1, \dots, z_{t-3} (και επομένως στους $\bar{z}_1, \dots, \bar{z}_{t-3}$) ώστε να αποτιμάται η ϕ^i σε True. Παρατηρούμε ότι αν $\hat{\phi}^i = \text{False}$, θα πρέπει $\lambda_1^i = \lambda_2^i = \dots = \lambda_t^i = \text{False}$. Για να έχουμε $\phi^i = \text{True}$, θα πρέπει κάθε πρόταση της (16.4) να έχει έναν τουλάχιστον

όρο με τιμή True. Άρα θα πρέπει να θέσουμε $z_1 = \text{True}$ για να αποτιμηθεί η πρώτη πρόταση της (16.4) σε True, στη συνέχεια, να θέσουμε $z_2 = \text{True}$ για να αποτιμηθεί η δεύτερη πρόταση της (16.4) σε True, κακ. Όμως έτσι θα φτάσουμε να έχουμε στην τελευταία πρόταση της (16.4) όλους τους όρους με τιμή False.

Από τα παραπάνω προκύπτει ότι ο τύπος $\hat{\phi}$ αποτιμάται σε True αν και μόνο αν το ίδιο συμβαίνει για τον ϕ .

SAT \leq_P 3SAT

Το μέγεθος του στιγμιοτύπου 3SAT είναι πολυφωνυμικό

Έστω n ο αριθμός των μεταβλητών που εμφανίζονται στον τύπο $\hat{\phi}$. Θα δείξουμε ότι ο αριθμός των όρων και των προτάσεων που εμφανίζονται στον τύπο ϕ φράζεται από τα επάνω από ένα πολυώνυμο των n, m . Παρατηρούμε ότι οι όροι σε κάθε πρόταση $\hat{\phi}^i$ δεν μπορούν να ξεπερνούν σε αριθμό το n : διαφορετικά η πρόταση αποτιμάται σε True αφού θα εμφανίζονται και οι δύο όροι για κάποια μεταβλητή - η πρόταση γίνεται ταυτολογία. Άρα εφόσον στην πρόταση $\hat{\phi}^i$ εμφανίζονται τοπολύ n όροι, η ϕ^i αποτελείται από το-πολύ $n - 2$ προτάσεις³ με τρεις όρους η κάθε μία. Επομένως ο αριθμός των προτάσεων στο στιγμιότυπο 3SAT είναι τάξης $O((n - 2)m) = O(nm)$. Αντίστοιχα, σε κάθε ϕ^i εμφανίζονται το-πολύ $2(n - 3)$ νέοι όροι και άρα ο συνολικός αριθμός των όρων στην ϕ^i δεν μπορεί να ξεπερνάει τους $2(n - 3) + n = 3(n - 2)$. Επομένως ο αριθμός των όρων του στιγμιότυπου 3SAT είναι τάξης $O(3(n - 2)m) = O(nm)$. \square

Σημειώσεις

ωτωτή

(Το πέδιλγμα 3SAT)

Chapter 18c

3SAT

Στρατηγική: Τίπος οι σε κανονική συγκετική μορφή στους σε κάθε πρώτην εμφανίζονται 3 οροί

Ερώτηση: Είναι οι υπαναπονηγμένοι;

1] 3SAT ENP ✓

$$\begin{aligned}
 & q^1 \quad \left. \begin{array}{c} (-v_v) \wedge (-v_v) \wedge \dots \wedge (-v_v) \\ \vdots \end{array} \right\} = q_1 \quad q_2 \quad \dots \quad q^m \\
 & q^2 \\
 & \vdots \\
 & q^m
 \end{aligned}$$

$\phi = \bigwedge_{i=1}^m q_i$ nήσος των προτάσεων

\sum καθε q_i εκτελείται \Leftrightarrow OR (\vee) και
nήσος των προτάσεων - 1 AND (\wedge)

Άστεγα, ο αριθμός ΣΥΒ που εκτελείται για $\neg v$ είναι
 λίγος των ΝΑΙ στην πρώτη είναι

$2m+m-1 = 3m-1 = O(m)$. Αναδιπλάσια
 η λογική δωση εκτελείται σε πολυωνύμιο χρόνο ✓

2] SAT \leq_p 3SAT

Ου σίγουρα μη καταρρέει τη στηγμένην SAT
 που θα πρέπει να στηγμένη 3SAT. Το στηγμένο τον τον SAT θα είναι ΝΑΙ αν και μόνο αν
 στηγμένη 3SAT είναι ΝΑΙ. Τέλος, ου σίγουρα
 αναγράφεται σε πολυωνύμιο χρόνο

$$\hat{\varphi} = \hat{\varphi}^1 \wedge \hat{\varphi}^2 \wedge \dots \wedge \hat{\varphi}^m \rightarrow \text{Σύγκριση των SAT}$$

2.1] Av $\hat{\varphi}^i$ NAI Tότε και φ^i NAI

Πλιόνω για την πρόταση $\hat{\varphi}^i$ tί είναι το αντίστοιχο των όπως λί

SAT

3SAT

$$\hat{\varphi}_1^i = \lambda_1^i$$

$$\hat{\varphi}_2^i = \lambda_1^i \vee \lambda_2^i$$

$$\hat{\varphi}_3^i = \lambda_1^i \vee \lambda_2^i \vee \lambda_3^i$$

$$\hat{\varphi}_t^i = \lambda_1^i \vee \lambda_2^i \vee \lambda_3^i \vee \dots \vee \lambda_t^i$$

$$\hat{\varphi}_i^i = \lambda_1^i \vee \lambda_2^i \vee \lambda_3^i \vee \lambda_4^i$$

$$\varphi^i = \lambda_1^i \vee \lambda_2^i \vee \lambda_3^i$$

$$\varphi^i = \lambda_1^i \vee \lambda_2^i \vee \lambda_3^i$$

$$\varphi^i = \lambda_1^i \vee \lambda_2^i \vee \lambda_3^i$$

$$\varphi^i = (\lambda_1^i \vee \lambda_2^i \vee z_L) \wedge$$

$$(\bar{z}_1 \vee \lambda_3^i \vee z_2) \wedge \dots$$

$$\wedge (\bar{z}_{k-2} \vee \lambda_k^i \vee z_{k-1}) \wedge \dots$$

$$\wedge (\bar{z}_{t-4} \vee \lambda_{t-2}^i \vee z_{t-3}) \wedge$$

$$(\bar{z}_{t-3} \vee \lambda_{t-2}^i \vee \lambda_t^i)$$

$$\varphi^i = (\lambda_1^i \vee \lambda_2^i \vee z_L) \wedge$$

$$(\bar{z}_1 \vee \lambda_3^i \vee \lambda_4^i)$$

t=1

t=2

t=3

t≥4

t=4

2.2] Av $\hat{\varphi}^i$ OXI Tότε και φ^i OXI

To $\hat{\varphi}^i$ είναι OXI ή το λύχνο στην λ^i είναι false

Av αναδύονται τύποι στην οποιας του cli μπορεί να είναι

NAI ή καταλήφθει να είναι και αυτό OXI ..

2.3] Το μέγεθος των φ^i είναι πολυαριθμικό

| γρατάσσεται ότι $\varphi^i = O((n-2)m) = O(n \cdot m)$

| όροι των $\varphi^i = O(3(n-2)m) = O(n \cdot m)$

18d. Το πρόβλημα 1-3SAT

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 135-137

Περιγραφή

Το πρόβλημα αυτό αποτελεί παραλλαγή του 3SAT. Όπως και στην περίπτωση εκείνου η ερώτηση αφορά την ικανοποιησιμότητα ενός τύπου $\phi = \wedge_{i=1}^m \phi^i$, όπου κάθε ϕ^i αποτελείται από τρεις όρους. Όμως σε ένα ΝΑΙ-στιγμιότυπο το ϕ αποτιμάται σε True αν και μόνο αν ακριβώς ένας από τους τρεις όρους σε κάθε πρόταση παίρνει την τιμή True. Έχουμε τον ακόλουθο ορισμό.

Διατύπωση

1 – 3SAT

Στιγμιότυπο: Τύπος ϕ σε κανονική συζευκτική μορφή όπου σε κάθε πρόταση εμφανίζονται τρεις όροι.

Ερώτηση: Είναι ο ϕ ικανοποιήσιμος από μία ανάθεση τιμών στους όρους που δίνει τιμή True σε ακριβώς έναν όρο σε κάθε πρόταση;

Επεξήγηση

Για να γίνει ευκολότερα αντιληπτός ο παραπάνω ορισμός, ο τύπος

$$\phi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3)$$

ενώ αποτελεί ένα ΝΑΙ-στιγμιότυπο 3SAT (αφού η απόδοση τιμών $x_1 = \text{True}$, $x_2 = \text{False}$, $x_3 = \text{False}$ τον ικανοποιεί) αποτελεί ένα ΟΧΙ-στιγμιότυπο 1 – 3SAT αφού δεν υπάρχει απόδοση τιμών στους όρους που να τον ικανοποιεί και σε κάθε πρόταση να υπάρχει ένας μόνο όρος να έχει τιμή True.

1 – 3SAT ∈ NP

Μία απόδοση τιμών στους όρους των προτάσεων που απαρτίζουν τον τύπο ϕ αποτελεί ένα πιστοποιητικό για το ότι ο τύπος αποτελεί ένα NAI – στιγμιότυπο. Θεωρώντας ότι το στιγμιότυπο αποτελείται από m προτάσεις, σε χρόνο $\Theta(m)$ επαληθεύεται ότι $\phi = \text{True}$ (δες απόδειξη Λήμματος 43). Στη συνέχεια, για κάθε πρόταση αρχικοποιείται (σε τιμή μηδέν) ένας μετρητής ο οποίος αυξάνει κάθε φορά που συναντάται ένας όρος με τιμή True. Σε ένα NAI – στιγμιότυπο, ο μετρητής της κάθε πρότασης πρέπει να επιστρέφει τιμή ίση με 1. Ο έλεγχος αυτός μπορεί να εκτελεστεί σε $\Theta(m)$ ΣΥΒ αφού για κάθε πρόταση εκτελείται σταθερός αριθμός ΣΥΒ και ο τύπος περιέχει m προτάσεις.

1 – 3SAT ∈ NP – hard

Θα δείξουμε ότι $\text{3SAT} \leq_P 1 - \text{3SAT}$.

$\text{3SAT} \leq_P 1 - \text{3SAT}$

Το στιγμιότυπο 3SAT είναι NAI ANN το στιγμιότυπο 1-3SAT είναι NAI

Θεωρούμε ένα στιγμιότυπο 3SAT με m προτάσεις: δηλαδή, τύπο $\phi = \wedge_{i=1}^m \phi^i$, όπου ϕ^i είναι μία πρόταση με τρεις όρους. Από αυτό θα κατασκευάσουμε ένα στιγμιότυπο 1 – 3SAT τέτοιο ώστε να είναι NAI-στιγμιότυπο αν και μόνο αν το 3SAT είναι NAI-στιγμιότυπο. Μπορούμε να γράψουμε κάθε πρόταση ϕ^i του στιγμιότυπου 3SAT ως

$$\phi^i = a^i \vee b^i \vee c^i, \quad i = 1, \dots, m.$$

Για κάθε τέτοια πρόταση, εισάγουμε στο στιγμιότυπο 1 – 3SAT την σύζευξη τεσσάρων προτάσεων

$$\psi^i = (\bar{a}_i \vee y_1^i \vee z_1^i) \wedge (\bar{b}_i \vee y_2^i \vee z_2^i) \wedge (\bar{c}_i \vee y_3^i \vee z_3^i) \wedge (y_1^i \vee y_2^i \vee y_3^i),$$

όπου $\bar{a}_i, \bar{b}_i, \bar{c}_i$ οι αντίθετοι των όρων a_i, b_i, c_i , αντίστοιχα, και οι y_1^i, y_2^i, y_3^i και z_1^i, z_2^i, z_3^i είναι νέοι όροι που δεν εμφανίζονται στον τύπο ϕ .

Αν το στιγμιότυπο 3SAT είναι ένα NAI-στιγμιότυπο τότε υπάρχει μία απόδοση τιμών στους όρους για την οποία σε κάθε πρόταση ϕ^i υπάρχει ένας τουλάχιστον όρος με τιμή True. Χωρίς βλάβη της γενικότητας θεωρούμε ότι στην πρόταση ϕ^i ο όρος $a^i = \text{True}$. Θα δείξουμε ότι μπορούμε να δώσουμε τιμές στους νέους όρους που εισάγαμε παραπάνω ώστε η ψ^i να αποτιμάται σε True και ταυτόχρονα να παίρνει τιμή True μόνο ένας από τους όρους σε κάθε πρόταση της ψ^i . Η ανάθεση είναι

$$y_1^i = \text{True}, \quad y_2^i = \text{False}, \quad y_3^i = \text{False},$$

$$z_1^i = \text{False}, \quad z_2^i = b^i, \quad z_3^i = c^i,$$

όπου οι δύο τελευταίες σχέσεις υποδηλώνουν ότι οι όροι z_2^i, z_3^i παίρνουν την ίδια τιμή που έχουν οι όροι b^i, c^i , αντίστοιχα, στην απόδοση τιμών που ικανοποιεί τον τύπο ϕ . Παρατηρούμε ότι με δεδομένο ότι $a_i = \text{True}$, η παραπάνω απόδοση τιμών (όποιες και αν είναι οι τιμές των όρων b^i, c^i) αποτιμά την ψ^i σε True θέτοντας ακριβώς έναν όρο σε τιμή True σε κάθε μία από της τέσσερις προτάσεις που την αποτελούν.

$3\text{SAT} \leq_p 1 - 3\text{SAT}$

Το στιγμιότυπο 3SAT είναι ΟΧΙ ANN το στιγμιότυπο $1-3\text{SAT}$ είναι ΟΧΙ

Αντίστροφα, αν το 3SAT είναι ΟΧΙ-στιγμιότυπο, θα υπάρχει πρόταση ϕ^i όπου όλοι οι όροι της θα έχουν την τιμή False . Στην περίπτωση αυτή σε κάθε μία από τις τρεις πρώτες προτάσεις της ψ^i θα έχουμε έναν από τους αντίθετους όρους σε τιμή True . Παρατηρούμε, ότι επειδή απαιτείται να υπάρχει μόνο ένας όρος με τιμή True σε κάθε πρόταση, θα πρέπει όλοι οι νέοι όροι να τεθούν σε τιμή False . Όμως αυτή η απόδοση τιμών κάνει την τέταρτη πρόταση της ψ^i να αποτιμάται σε False και επομένως και η ψ^i αποτιμάται σε False . Συνεπώς, και το παραγόμενο $1 - 3\text{SAT}$ στιγμιότυπο είναι ΟΧΙ-στιγμιότυπο.

$3\text{SAT} \leq_p 1 - 3\text{SAT}$

Το μέγεθος του στιγμιοτύπου $1-3\text{SAT}$ είναι πολυωνυμικό

Όσον αφορά το μέγεθος του $1 - 3\text{SAT}$ στιγμιότυπου που κατασκευάσαμε, παρατηρούμε ότι έχει $4m$ προτάσεις και $6m$ νέους όρους. Επομένως το μέγεθος του είναι πολυωνυμικό σε σχέση με το μέγεθος του 3SAT από το οποίο ξεκινήσαμε.

□

{ Το πρόβλημα 1-3SAT}

Chapter 18d

L] L-3SAT ∈ NP ✓

Παρέργων με 3SAT... ✓

2] 3SAT ≤_P 1-3SAT2.1] $\varphi^i = a^i \vee b^i \vee c^i \rightarrow$ Επιμόνων του 3SAT $\psi^i = (\bar{a}_i \vee y_1^i \vee z_1^i) \wedge (\bar{b}_i \vee y_2^i \vee z_2^i) \wedge (\bar{c}_i \vee y_3^i \vee z_3^i)$
 $\wedge (y_1^i \vee y_2^i \vee y_3^i) \rightarrow$ Επιμόνων του 1-3SATΑνόδοση πρώτη $\rightarrow y_1^i = \text{False} \quad z_1^i = \text{False}$ $y_2^i = \text{False} \quad z_2^i = b^i$ $y_3^i = \text{False} \quad z_3^i = c^i$

Με δεδομένων $a_i = \text{True}$ το στύγμιστων φ_i^i είναι NAI,
δηλαδή $\varphi_i^i = \text{True}$ για ακριβώς είναι όπου και είναι True

2.2] Το φ_i^i για να είναι OXI πρέπει και α
3 όπου a_i^i, b_i^i, c_i^i να είναι False ①

Με την ίδια ανώδοση τυπών στο φ_i^i θα έχει μερικές
έξι στάσεις $a_i^i = \text{False}$

Λόγω των περιορισμών να είναι σίας όπου στο φ_i^i True
για ανώδοση τυπών είναι

$$y_1^i = \text{False}, z_1^i = \text{False}$$

$$y_2^i = \text{False}, z_2^i = b^i$$

$$y_3^i = \text{False}, z_3^i = c^i$$

Με δεδομένων $a_i^i = \text{False}$ ①, το στύγμιστων φ_i^i είναι OXI

2.3] Κατά προτίτις και διανοία όπου στο
στύγμιστων τη 1-3SAT, το οποίο είναι προφέρει
στο είκονα πολυκαρπό

19a. kIKAN

Διαφάνειες

Slides_and_Exercises→16_npcexer.pdf σελ. 1

Διατύπωση

Άσκηση 1 Να δείξετε ότι $k\text{IKAN} \in \text{NP-πλήρες}$ για $k \geq 4$.

Επεξήγηση

Λύση Η δομή $k\text{IKAN}$ υπονοεί το πρόβλημα IKAN όταν υπάρχουν ακριβώς k όροι σε κάθε πρόταση.

$k\text{IKAN} \in \text{NP}$

Επομένως η απόδειξη ότι $k\text{IKAN}$ ανήκει στην τάξη NP είναι ακριβώς ανάλογη με την αντίστοιχη απόδειξη για 3IKAN και παραλείπεται.

$k\text{IKAN} \in \text{NP-hard}$

Στη συνέχεια θα δείξουμε $3\text{IKAN} \leq_P k\text{IKAN}$.

$3\text{IKAN} \leq_P k\text{IKAN}$

Το στιγμιότυπο 3IKAN είναι NAI/OXI ANN το στιγμιότυπο $k\text{IKAN}$ είναι NAI/OXI

3IKAN σε ΣΚΜ, δηλαδή,

$$\phi = \bigwedge_{j=1}^m C_j \quad (1)$$

όπου

$$C_j = a_j \vee b_j \vee c_j, \quad j \in \{1, \dots, m\}. \quad (2)$$

Το αντίστοιχο στιγμιότυπο $k\text{IKAN}$ αποτελείται από σύζευξη των προτάσεων

$$C'_j = a_j \vee b_j \vee c_j, \vee z_1 \vee \dots \vee z_{k-3}, \quad j \in \{1, \dots, m\}, \quad (3)$$

όπου $z_1 \vee \dots \vee z_{k-3}$ νέες μεταβλητές που δεν υπάρχουν στο σύστημα 3IKAN . Μία αποτίμηση T στο στιγμιότυπο $k\text{IKAN}$ είναι ακριβώς ίδια όπως και στο 3IKAN για όλες τις μεταβλητές που υπάρχουν και στα δύο στιγμιότυπα ενώ για τις επιπλέον μεταβλητές z , έχουμε $T(z_1) = T(z_2) = \dots = T(z_{k-3}) = \text{False}$.

$3\text{IKAN} \leq_P k\text{IKAN}$

Το μέγεθος του στιγμιοτύπου $k\text{IKAN}$ είναι πολυωνυμικό

To στιγμιότυπο $k\text{IKAN}$ έχει ακριβώς τον ίδιο αριθμό προτάσεων με το στιγμιότυπο 3IKAN ενώ έχει παραπάνω $k - 3$ μεταβλητές και επομένως η αναγωγή είναι πολυωνυμική.

(N.S.o. k SAT NP-complete, $k \geq 4$)

Chapter 19a

10/11/2024

1] $A \in \text{NP}$

*)

2.3] To πρόβλημα A

είναι non-deterministic

2] $B \leq_p A \Rightarrow A \in \text{NP-hard}$ 2.1] To στυγμότυπο του B είναι NAI ου και μόνο αν
to στυγμότυπο του A είναι NAI2.2] To στυγμότυπο του B είναι OXI ου και μόνο αν to στυγμότυπο του A είναι OXI (✗)1] $\varphi = \bigwedge_{i=1}^m c_i$, οπου $c_i = \lambda_1^i \vee \lambda_2^i \vee \lambda_3^i \vee \dots \vee \lambda_k^i$

$$\frac{m \cdot (k-1) + (m-1)}{\downarrow \# \text{στυγμών} \quad \# \text{στυγμών}} \in \mathcal{O}(mk) \leq \mathcal{O}(mn)$$

k ≤ n

προτάσεων c_i

kSAT in NP ✓

2] $3\text{SAT} \leq_p k\text{SAT}$

$$2.1] \hat{\varphi} = \bigwedge_{i=1}^m \hat{c}_i$$

$$\hat{c}_i = \lambda_1^i \vee \lambda_2^i \vee \lambda_3^i$$

Με την αντικατόταξη στους
οριους των στυγμών
kSAT, to στυγμότυπο 3SAT είναι
NAI (ίνας οριος είναι True)

- 57 -

$$\varphi = \bigwedge_{i=1}^m c_i$$

	False	False
↑	↑	

$$c_i = \lambda_1^i \vee \lambda_2^i \vee \lambda_3^i \vee z_1 \vee \dots \vee z_{k-3}$$

To λειχτότητα είναι οριος η οποία
είναι True όταν μετατίθεται
NAI στυγμότυπο και αυτής
 $z_1 = \dots = z_{k-3} = \text{False}$, τότε ίνας
οντος οριος $\lambda_1^i, \lambda_2^i, \lambda_3^i$ τυλιγμότορ
είναι True

2.2] Με $\lambda_1^i = \lambda_2^i = \lambda_3^i = \text{False}$ ουτός
 το στύριστυρο 3SAT είναι OXI
 αντί για πρώτης και σε τρίτης σέριες
 $\lambda_1^i, \lambda_2^i, \lambda_3^i$ με είναι False για
 να είναι OXI στύριστυρο

2.3] Το μέγεθος των kSAT είναι $\frac{2 \cdot (k-3) \cdot m}{z_1 z_{k-3} \# \text{vars} \# \text{prototypes}} \approx$
 αφού είναι ανλογικό
 kSAT ENP-hard ✓

kSAT GNP-complete ✓

19b. ΔΙΠΛΗ-ΙΚΑΝ

Διαφάνειες

Slides_and_Exercises → 16_npcexer.pdf σελ. 1-2

Διατύπωση

Στιγμιότυπο: Έκφραση φ σε ΚΣΜ.

Ερώτηση: Υπάρχουν τουλάχιστον δύο αποτυπήσεις που να ικανοποιούν την φ.

ΔΙΠΛΗ – IKAN ∈ NP

Λύση Η απόδειξη ότι ΔΙΠΛΗ-ΙΚΑΝ ∈ NP είναι ανάλογη της αντίστοιχης απόδειξης για το πρόβλημα IKAN και επομένως παραλείπεται.

ΔΙΠΛΗ – IKAN ∈ NP – hard

Θα δείξουμε $\text{IKAN} \leq_p \text{ΔΙΠΛΗ-ΙΚΑΝ}$.

$\text{IKAN} \leq_p \text{ΔΙΠΛΗ – IKAN}$

Το στιγμιότυπο IKAN είναι NAI/OXI ANN το στιγμιότυπο ΔΙΠΛΗ-ΙΚΑΝ είναι NAI/OXI

Θεωρούμε ένα στιγμιότυπο IKAN σε ΣΚΜ

(βλέπε (1), (2)). Κατασκευάζουμε ένα αντίστοιχο στιγμιότυπο ΔΙΠΛΗ-ΙΚΑΝ ως εξής. Για κάθε πρόταση C_j εισάγουμε δύο προτάσεις, ήτοι

$$A_j = C_j \vee z, \bar{A}_j = C_j \vee \neg z,$$

όπου z είναι μία νέα μεταβλητή. Το στιγμιότυπο ΔΙΠΛΗ-ΙΚΑΝ είναι

$$\varphi' = \bigwedge_{j=1}^m (A_j \wedge \bar{A}_j).$$

Η αναγωγή είναι σωστή αφού $A_j \wedge \bar{A}_j$ ικανοποιείται ANN C_j ικανοποιείται όποια αποτίμηση και να θεωρήσουμε για τη μεταβλητή z .

IKAN \leq_p ΔΙΠΛΗ – IKAN

Το μέγεθος του στιγμιοτύπου ΔΙΠΛΗ-ΙΚΑΝ είναι πολυωνυμικό

Επίσης είναι πολυωνυμική αφού η φ' περιέχει $2m$ προτάσεις και μία επιπλέον μεταβλητή (τη z).

Σημειώσεις

(N.G.) Double-SAT NP-complete

Chapter 19b

1]

Παροχών με SAT ... Chapter 18b

Double-SAT NP ✓

2]

$SAT \leq_p Double-SAT$

2.1]

\downarrow

2.2]

$$q = \bigwedge_{i=1}^n C_i$$

$$A_i = C_i \vee z, \bar{A}_i = C_i \vee \neg z$$

... (Μόνο η z αλλάζει τις C_i παίρνει την $\neg C_i$!!) τότε

1 απλή ιδέα. 2.1/2.2 ✓

Προσθέτω στο στύριστων των Double-SAT σών νέες προτάσεις

$$\hat{\varphi} = \bigwedge_{i=1}^m (A_i \wedge \bar{A}_i)$$

Ουασίνοτε

πολυωνυμική στα \exists δεν εμφανίζεται την ικανότητα μόντη της $\hat{\varphi}$. Αφού, τα C_i που ανήκει και στο SAT σίνη ικανότητα \exists ...

- 58 -

2.3] Το στύριστων $\hat{\varphi}$ πρέπει $\frac{2m+1}{z}$ ΣΥΒ

αφού το μέγεθος είναι πολυωνυμικό

Double-SAT NP-hard ✓

↑ ↑

$A_i, \bar{A}_i \in z$

Double-SAT NP-complete ✓

Διαφάνειες

Slides_and_Exercises→16_npcexer.pdf σελ. 2-4

Διατύπωση ΟΧΙ-ΟΛΑ-ΙΔΙΑ-4ΙΚΑΝ

Στιγμιότυπο: Έκφραση ϕ σε $K\sigma M$ με ακριβώς τέσσερις όρους σε κάθε πρόταση.

Ερώτηση: Υπάρχει κάποια αποτίμηση που να ικανοποιεί την ϕ και να μην δίνει σε όλους τους όρους κάθε πρότασης την ίδια τιμή;

ΟΧΙ – ΟΛΑ – ΙΔΙΑ – 4ΙΚΑΝ ∈ NP

Λύση Και για τα δύο προβλήματα είναι εύκολο να δειχθεί ότι ανήκουν στην τάξη NP και επομένως η αντίστοιχη απόδειξη παραλείπεται.

ΟΧΙ – ΟΛΑ – ΙΔΙΑ – 4ΙΚΑΝ ∈ NP – hard

Θα δείξουμε ότι $3IKAN \leq_P OXI-OЛА-IDIA-4IKAN$.

 $3IKAN \leq_P OXI-OЛА-IDIA-4IKAN$

Το στιγμιότυπο $3IKAN$ είναι NAI/OXI ANN το στιγμιότυπο $OXI-OЛА-IDIA-4IKAN$ είναι NAI/OXI
Το μέγεθος του στιγμιοτύπου $OXI-OЛА-IDIA-4IKAN$ είναι πολυωνυμικό

Θεωρούμε ένα $3IKAN$ στιγμιότυπο της μορφής (1), (2). Για κάθε πρόταση C_j εισάγουμε στο $OXI-OЛА-IDIA-4IKAN$ στιγμιότυπο την πρόταση

$$A_j = C_j \vee z,$$

όπου z είναι μία νέα μεταβλητή. Για οποιαδήποτε αποτίμηση T , θεωρούμε την επέκταση της με $T(z) = False$. Προοέξτε ότι A_j ικανοποιείται ANN C_j ικανοποιείται στην περίπτωση κατά την οποία τουλάχιστον ένας από τους όρους a_j, b_j, c_j παίρνει την τιμή $True$. Όμως τότε η πρόταση A_j ικανοποιείται και έχει και έναν όρο με άλλη τιμή (αφού $T(z) = False$, για κάθε αποτίμηση T).

Διατύπωση ΟΧΙ-ΟΛΑ-ΙΔΙΑ-3ΙΚΑΝ

Στιγμιότυπο: Έκφραση ϕ σε $K\sigma M$ με ακριβώς τρεις όρους σε κάθε πρόταση.

Ερώτηση: Υπάρχει κάποια αποτίμηση που να ικανοποιεί την ϕ και να μην δίνει σε όλους τους όρους κάθε πρότασης την ίδια τιμή;

ΟΧΙ – ΟΛΑ – ΙΔΙΑ – 3ΙΚΑΝ ∈ NP

Λύση Και για τα δύο προβλήματα είναι εύκολο να δειχθεί ότι ανήκουν στην τάξη NP και επομένως η αντίστοιχη απόδειξη παραλείπεται.

ΟΧΙ – ΟΛΑ – ΙΔΙΑ – 3IKAN ∈ NP – hard

Θα δείξουμε ότι $\text{OXI-OΛA-IΔIA-4IKAN} \leq_p \text{OXI-OΛA-IΔIA-3IKAN}$.

ΟΧΙ – ΟΛΑ – ΙΔΙΑ – 4IKAN \leq_p ΟΧΙ – ΟΛΑ – ΙΔΙΑ – 3IKAN

Το στιγμιότυπο ΟΧΙ-ΟΛΑ-ΙΔΙΑ-4ΙΚΑΝ είναι ΝΑΙ/ΟΧΙ ANN το στιγμιότυπο ΟΧΙ-ΟΛΑ-ΙΔΙΑ-3ΙΚΑΝ είναι ΝΑΙ/ΟΧΙ

Έστω

το στιγμιότυπο ΟΧΙ-ΟΛΑ-ΙΔΙΑ-4ΙΚΑΝ που περιγράφεται από την (1) και από

$$C_j = a_j \vee b_j \vee c_j \vee d_j, \quad j \in \{1, \dots, m\}.$$

Για κάθε τέτοια πρόταση, εισάγουμε στο στιγμιότυπο ΟΧΙ-ΟΛΑ-ΙΔΙΑ-3ΙΚΑΝ τις δύο προτάσεις

$$A_j = a_j \vee b_j \vee w_j,$$

$$B_j = c_j \vee d_j \vee \neg w_j,$$

όπου w_j είναι νέα μεταβλητή. Προσέξτε ότι επειδή η ϕ αποτελεί στιγμιότυπο ΟΧΙ-ΟΛΑ-ΙΔΙΑ-4ΙΚΑΝ τουλάχιστον ένας από τους όρους της C_j θα παίρνει την τιμή *False* και ένας την τιμή *True* σε κάθε αποτίμηση T που την ικανοποιεί. Χωρίς βλάβη της γενικότητας, λόγω συμμετρίας, μπορούμε να θεωρήσουμε ότι $T(a_j) = \text{False}$ και $T(c_j) = \text{True}$. Επεκτείνοντας την αποτίμηση θέτουμε

$$T(w_j) = \begin{cases} \text{True}, & \text{αν } (T(b_j) = \text{False}) \vee (T(d_j) = \text{True}), \\ \text{False}, & \text{διαφορετικά,} \end{cases}$$

το στιγμιότυπο ΟΧΙ-ΟΛΑ-ΙΔΙΑ-3ΙΚΑΝ

$$\phi' = \bigwedge_{j=1}^m (A_j \wedge B_j)$$

ικανοποιείται ANN το ΟΧΙ-ΟΛΑ-ΙΔΙΑ-4ΙΚΑΝ ικανοποιείται. Επίσης τόσο στην πρόταση A_j όσο και στην πρόταση B_j δεν έχουν όλοι οι όροι τις ίδιες τιμές.

3IKAN \leq_p ΟΧΙ – ΟΛΑ – ΙΔΙΑ – 4IKAN

Το μέγεθος του στιγμιούπον ΟΧΙ-ΟΛΑ-ΙΔΙΑ-4ΙΚΑΝ είναι πολυωνυμικό

Η έκφραση ϕ' έχει $2 * m$ προτάσεις και m νέες μεταβλητές και άρα ο μετασχηματισμός είναι πολυωνυμικός.

N.S.o. OXI-ΟΛΑ-ΙΔΙΑ-3ΙΚΑΝ ∈ NP-complete

Chapter 19c

Αποδεκτήνωρε στο OXI-ΟΛΑ-ΙΔΙΑ-4ΙΚΑΝ NP-complete για
και χρησιμοποιείσσει την αναγνώση

$$\text{OXI-ΟΛΑ-ΙΔΙΑ-4ΙΚΑΝ} \leq \rho \text{ OXI-ΟΛΑ-ΙΔΙΑ-3ΙΚΑΝ}$$

1] Και τα Siw παρόμοια με 3SAT... Chapter 18c

$$\text{OXI-ΟΛΑ-ΙΔΙΑ-4ΙΚΑΝ} \in \text{NP} \checkmark$$

$$\text{OXI-ΟΛΑ-ΙΔΙΑ-3ΙΚΑΝ} \in \text{NP} \checkmark$$

2] $\exists \text{IKAN} \leq \rho \text{ OXI-ΟΛΑ-ΙΔΙΑ-4ΙΚΑΝ}$

$\varphi = \bigwedge_{i=1}^m C_i \quad (=)$

$$C_i = \lambda_1^i \vee \lambda_2^i \vee \lambda_3^i$$

OXI

Προσθέτω στο στυγμότυπο την πόσταση

$$A_i = C_i \vee z$$

Δινώ αντίγραφο, $z = \text{False}$

Αρν για να είναι $A_i = \text{True}$ ου

τείνει $C_i = \text{True}$ και σημαίνει στη

σειρά των λεξικών ότις την C_i θέλει

να είναι True

ΝΑ!

2.2] Αν $z = \text{False}$, τότε

και $C_i = \text{False}$ για να είναι

OXI στυγμότυπο αριθ.

$$1^i = \lambda_1^i = \lambda_2^i = \lambda_3^i = \text{False}$$

OXI

2.3] Το στυγμότυπο του OXI-ΟΛΑ-ΙΔΙΑ-4ΙΚΑΝ περιέχει

$$\sum_{i=1}^m \lambda_i^i \geq 3$$

από σύνολο λέξης

$\text{OXI-ΟΛΑ-ΙΔΙΑ-4ΙΚΑΝ} \in \text{NP-hard} \checkmark$

- 59 - $\text{OXI-ΟΛΑ-ΙΔΙΑ-4ΙΚΑΝ} \in \text{NP-complete} \checkmark$

3] OXI-ΟΛΑ-ΙΔΙΑ-4ΙΚΑΝ \leq_p OXI-ΟΛΑ-ΙΔΙΑ-3ΙΚΑΝ

3.2] $C_i = \lambda_1^i \vee \lambda_2^i \vee \lambda_3^i \vee \lambda_4^i$

OXI-ΟΛΑ-ΙΔΙΑ σημαίνει
ότι είναι όρος οχα είναι

True και είναι όρος False

$\lambda_1^i = \text{False}$, $\lambda_2^i = \text{True}$

④ 3.3] P NAI

To σημαίνει
OXI-ΟΛΑ-ΙΔΙΑ-3ΙΚΑΝ
Έχει

$2m + m$ ΣΥΒ
 \downarrow \downarrow
 $A_i, B_i = z$

είναι είναι
να λυθείται

3.2] Για να είναι OXI οι μεταξύ αλογούνται οι οποιοι να είναι
True i ή οι οποιοι να είναι False

ταξιδεύει
 $\lambda_1^i = \text{False}$, $\lambda_2^i = \text{True}$, $\lambda_3^i = \text{False}$, $\lambda_4^i = \text{True}$ και $Z = \text{False}$

$\lambda_3^i = \text{False}$, $\lambda_4^i = \text{True}$ για $Z = \text{True}$

+

OXI

✓

Προσθέτουμε στο σημαίνει της δύο προτάσεων

$$A_i = \lambda_1^i \vee \lambda_2^i \vee z$$

$$B_i = \lambda_3^i \vee \lambda_4^i \vee \neg z$$

Αντίμητο: $\lambda_1^i = \text{False}$

$\lambda_3^i = \text{True}$

$Z = \begin{cases} \text{True}, & (\lambda_2^i = \text{False}) \vee (\lambda_4^i = \text{False}) \\ \text{False}, & \text{διαφορετικά} \end{cases}$

$$\varphi = \bigwedge_{i=1}^m (A_i \wedge B_i)$$

Σε κάθε γεωμετρική κυρίκη είναι
όρος True και είναι όρος False άπαντα
είναι NAI

OXI-ΟΛΑ-ΙΔΙΑ-3ΙΚΑΝ NP-hard

OXI-ΟΛΑ-ΙΔΙΑ-3ΙΚΑΝ NP-complete

Διαφάνειες

documents → Σημειώσεις (29/12/2023) σελ. 137-139

Περιγραφή

Θυμηθείτε ότι σε ένα μη-κατευθυνόμενο γράφημα $G(V, E)$ ένα ανεξάρτητο σύνολο είναι ένα υποσύνολο κορυφών V' τέτοιο ώστε να μην υπάρχει στο γράφημα ακμή ανάμεσα σε οποιοδήποτε ζευγάρι κορυφών του. Το πλήθος των κορυφών του V' ονομάζεται μέγεθος του ανεξάρτητου συνόλου. Με βάση τα παραπάνω ορίζουμε το πρόβλημα INDSET ως:

Διατύπωση

INDSET**Στιγμιότυπο:** Μη-κατευθυνόμενο γράφημα $G(V, E)$, ακέραιος k .**Ερώτηση:** Υπάρχει στο γράφημα G ένα ανεξάρτητο σύνολο μεγέθους μεγαλύτερου-ίσου του k ;

INDSET ∈ NP

Απόδειξη. Αν $k > |V(G)|$ τότε έχουμε ένα ΟΧΙ-στιγμιότυπο. Διαφορετικά (δηλαδή $k \leq |V(G)|$), χωρίς βλάβη της γενικότητας μπορούμε να θεωρήσουμε ότι $V(G) = \{1, 2, \dots\}$. Έτσι σε περίπτωση θετικής απάντησης - το συγκεκριμένο στιγμιότυπο έχει απάντηση ΝΑΙ - ένα πιστοποιητικό αποτελείται από ένα υποσύνολο του $V(G)$ μεγέθους τουλάχιστον k και το-πολύ ίσου με $|V(G)|$. Επομένως σε $O(|V(G)| \lg |V(G)|)$ ΣΥΒ⁴ μπορούμε να ελέγχουμε αν ένα δοσμένο σύνολο S πληροί στις ιδιότητες αυτές. Στη συνέχεια, ελέγχουμε αν για κάθε ζευγάρι κορυφών $v, u \in S$ δεν υπάρχει στο G ακμή που συνδέει τις κορυφές αυτές. Αυτό μπορεί να διαπιστωθεί εκτελώντας $O(|S|^2)$ ΣΥΒ - μία ερώτηση (σύγκριση) για κάθε ζευγάρι κορυφών του S . Επειδή $|S| \leq |V(G)|$ οι παραπάνω έλεγχοι εκτελούνται σε $(|V(G)|^2)$ ΣΥΒ. □

INDSET ∈ NP – hard

Θα δείξουμε ότι $\text{3SAT} \leq_P \text{INDSET}$.]

3SAT \leq_p INDSET

Το στιγμιότυπο 3SAT είναι NAI ANN το στιγμιότυπο INDSET είναι NAI

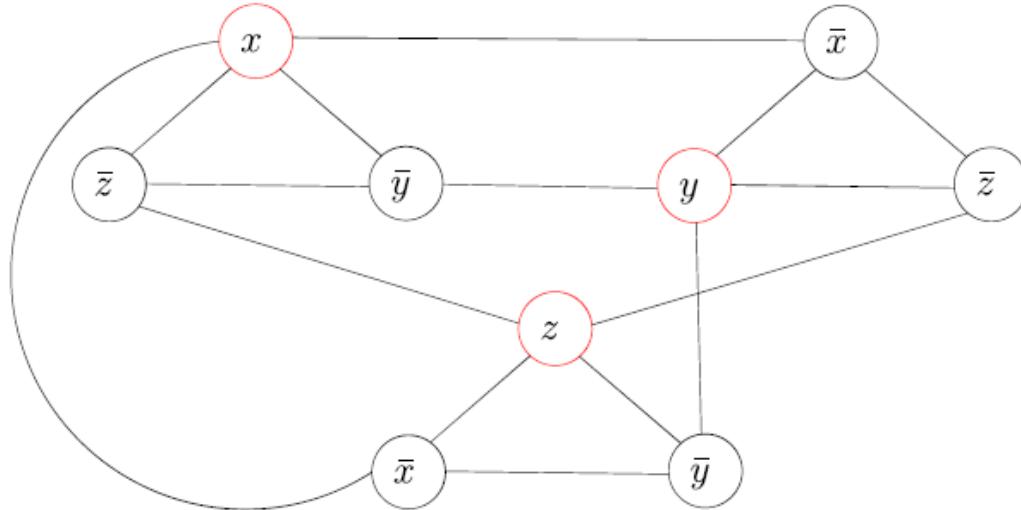
Προς τούτο, χωρίς βλάβη της γενικότητας, θεωρούμε ένα 3SAT στιγμιότυπο με k προτάσεις. Θα κατασκευάσουμε μη-κατευθυνόμενο γράφημα $G(V, E)$ κατά τρόπο ώστε αυτό θα έχει ένα ανεξάρτητο μεγέθους k αν και μόνο αν το 3SAT στιγμιότυπο είναι ικανοποιήσιμο. Η κατασκευή είναι η εξής. Για κάθε πρόταση του 3SAT εισάγουμε μία τριάδα κορυφών, μία κορυφή για κάθε όρο της πρότασης. Στο γράφημα υπάρχουν ακμές ανάμεσα στις κορυφές της ίδιας τριάδας καθώς και ακμές ανάμεσα σε κορυφές διαφορετικών τριάδων που αντιστοιχούν σε αντίθετους όρους. Για παράδειγμα για το στιγμιότυπο 3SAT

$$\phi = (x \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee y \vee \bar{z}) \wedge (\bar{x} \vee \bar{y} \vee z)$$

το γράφημα απεικονίζεται στο Σχήμα 16.3. Παρατηρούμε ότι το γράφημα αυτό περιέχει ένα ανεξάρτητο σύνολο τριών κορυφών (κόκκινες κορυφές). Αν θέσουμε τους αντίστοιχους όρους στο στιγμιότυπο 3SAT σε True παρατηρούμε ότι ο τύπος ϕ αποτιμάται σε True. Θα αποδείξουμε ότι τόσο αυτό όσο και το αντίστροφο ισχύει στη γενική περίπτωση.

Έστω ότι έχουμε ένα NAI-στιγμιότυπο 3SAT· ο αντίστοιχος τύπος $\phi = \bigwedge_{i=1}^k \phi^i$ είναι ικανοποιήσιμος. Τότε, σε κάθε πρόταση ϕ^i ένας τουλάχιστον όρος έχει τιμή True. Αν επιλέξουμε για κάθε πρόταση έναν από αυτούς τους όρους και παρατηρήσουμε τις αντίστοιχες κορυφές στο γράφημα $G(V, E)$ θα δούμε ότι αποτελούν ένα ανεξάρτητο σύνολο με k κορυφές: κάθε κορυφή ανήκει σε διαφορετική τριάδα (αφού έχουμε επιλέξει έναν όρο από κάθε πρόταση) και αν υπήρχε ακμή ανάμεσα σε δύο από αυτές αυτό θα σήμαινε ότι σε δύο αντίθετους όρους θα έχει δοθεί η τιμή True (άτοπο).

Σχήμα 16.3 Μη-κατευθυνόμενο γράφημα που περιέχει ανεξάρτητο σύνολο τριών κορυφών



3SAT \leq_p INDSET

Το στιγμιότυπο 3SAT είναι OXI ANN το στιγμιότυπο INDSET είναι OXI

Αντίστροφα, αν υπάρχει στο γράφημα ένα ανεξάρτητο σύνολο από k κορυφές τότε κάθε κορυφή θα ανήκει σε διαφορετική τριάδα - οι κορυφές της ίδιας τριάδας συνδέονται μεταξύ τους με ακμές - άρα οι αντίστοιχοι όροι εμφανίζονται ο καθένας σε διαφορετική πρόταση ϕ^i . Επίσης μπορούμε να τους θέσουμε όλους στην τιμή

True - αποτιμώντας έτσι τον τύπο ϕ σε True - χωρίς να υπάρχει περίπτωση να έχουμε θέσει δύο αντίθετους όρους στην τιμή True αφού κορυφές που ανήκουν σε διαφορετικές τριάδες συνδέονται με ακμή μόνο αν αντιστοιχούν σε αντίθετους όρους.

3SAT \leq_p INDSET

Το μέγεθος του στιγμιοτύπου INDSET είναι πολυωνυμικό

Το γράφημα έχει $|V| = 3k$ κορυφές και επομένως η αναγωγή είναι πολυωνυμική.

Σημειώσεις

INDSET

Chapter 19d

$$1 \quad \binom{k}{2} = \frac{k!}{2!(k-2)!} = \frac{k \cdot (k-1)}{2} \leq \frac{n(n-1)}{2} = O(n^2)$$

ΜΕΓΑΛΗ Εκτελούνται κ. συγκεισες άνα 2 κερμές.

Ενδισούνται με ακριψή;

INDSET ∈ NP



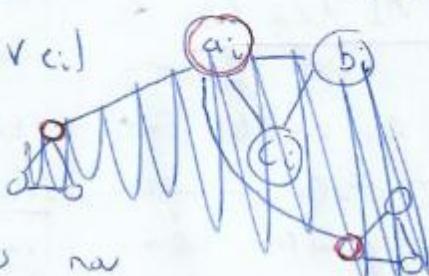
$$2 \quad 3SAT \leq_p INDSET$$

2.1

$$\varphi = \bigwedge_{i=2}^m \psi_i$$

$\rightarrow G(V, E)$

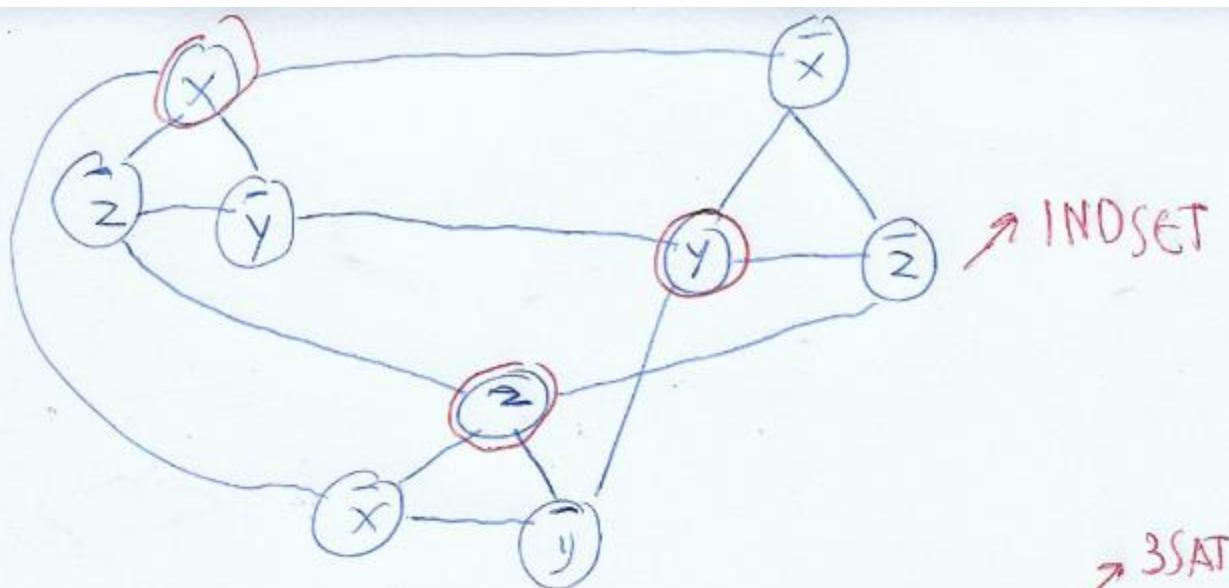
Κάθε πρόταση ψ_i περικλίνει 3 κορυφές
 $\psi_i = (a_i \vee b_i \vee c_i)$



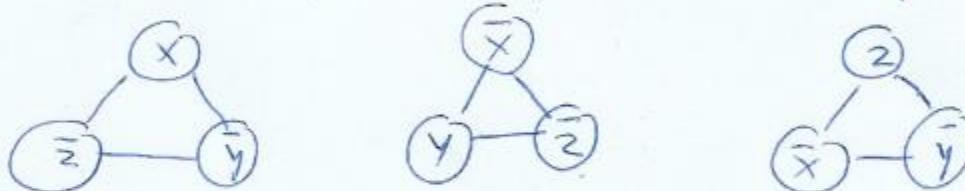
Άν το 3SAT είναι ΝΑΙ

τότε υπάρχει το λίγοτα είναι όπως να

έχει αντικρυψη True

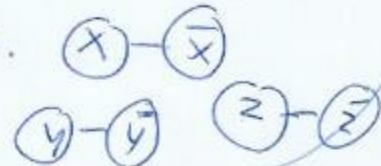


$$\varphi = (x \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee y \vee \bar{z}) \wedge (\bar{x} \vee \bar{y} \vee z) \quad \rightarrow \text{3SAT}$$



Έστω $x = y = z = \text{True}$ τότε αντίστοιχα οι αριθμοί που θέλουμε να κοπεύσουμε είναι $\bar{x}, \bar{y}, \bar{z}$. Σαδες κοπεύσουμε την αριθμητική συμμετρία της γραφής. Σαδες κοπεύσουμε την αριθμητική συμμετρία της γραφής.

Παραπάνω διαβάζουμε ότι τα τρία κομμάτια $\bar{x}, \bar{y}, \bar{z}$ θα γίνουν False . Κατά τη διαδικασία αυτής της κοπής, η γραφή θα γίνει μόνο τρία κομμάτια, τα οποία θα είναι τα τρία κομμάτια που έχουμε από την αριθμητική συμμετρία της γραφής.



2.3 Το γραφήμα έχει $|V| = 3$ κομμάτια, αλλά το γραφήμα δεν είναι ανθεκτικό. Τα τρία κομμάτια είναι ανθεκτικά μεταξύ τους, αλλά τα τρία κομμάτια είναι ανθεκτικά μεταξύ τους.

INDSETNP-hard



INDSETNP-complete

✓