

01_02_03_ΚΑΤΕΥΘΥΝΟΜΕΝΑ & ΜΗ ΓΡΑΦΗΜΑΤΑ / 04_graphtransexer

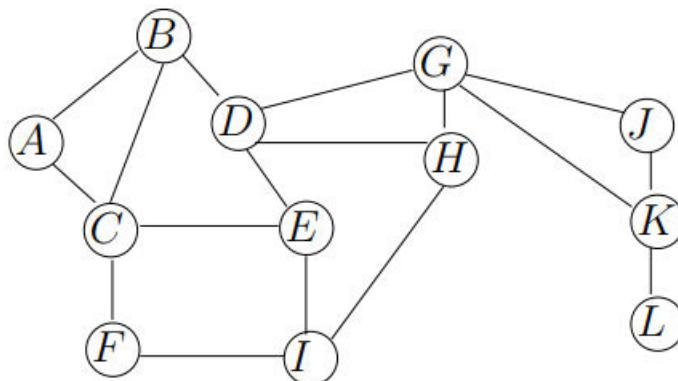
ΔκΒ σε Μη-Κατευθυνόμενο Γράφημα

Εκφώνηση

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

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

Σχήμα 1. Γράφημα



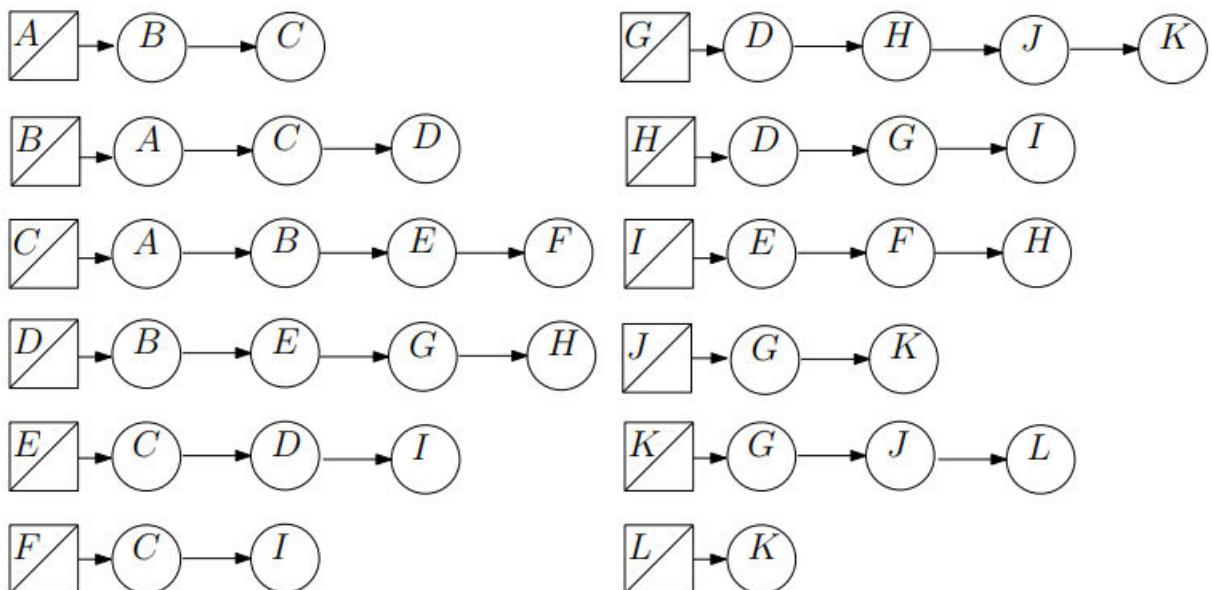
Λύση

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.

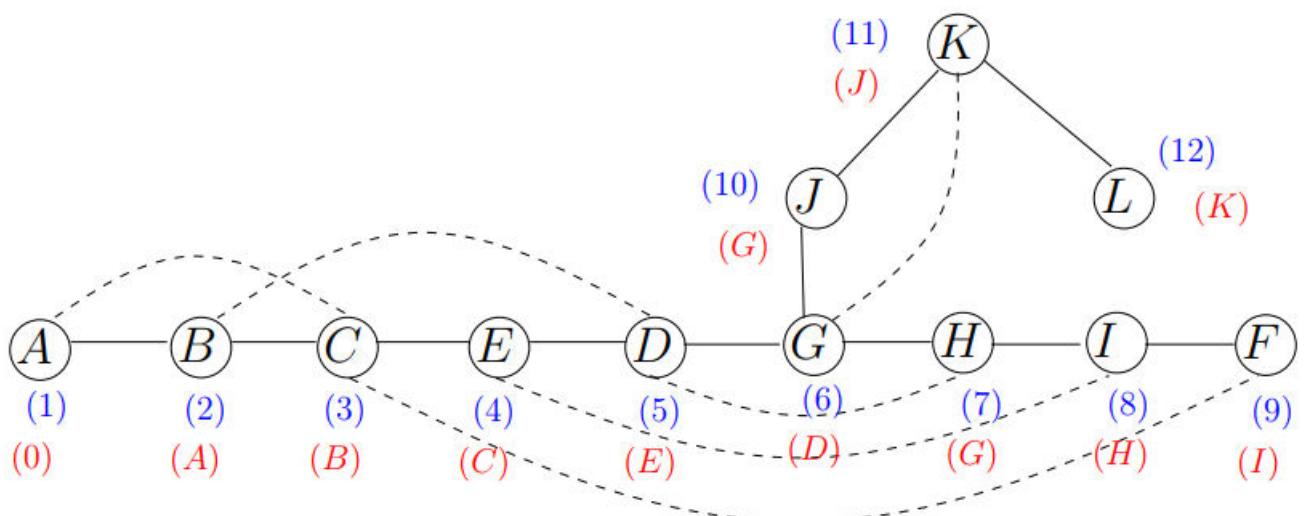
Για να μπορέσουμε να απεικονίσουμε το γεννητορικό δένδρο που υπολογίστηκε προηγουμένως εν είδη λίστας γειτνίασης, αρκεί να υπολογίσουμε για κάθε κορυφή v το σύνολο $N_T(v)$ των γειτόνων της στο δένδρο αυτό. Προφανώς $N_T(v) \subseteq N(v)$. Εκκινώντας από ένα άδειο $N_T(v)$, προσθέτουμε σε αυτό κάθε κορυφή u ∈ 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.

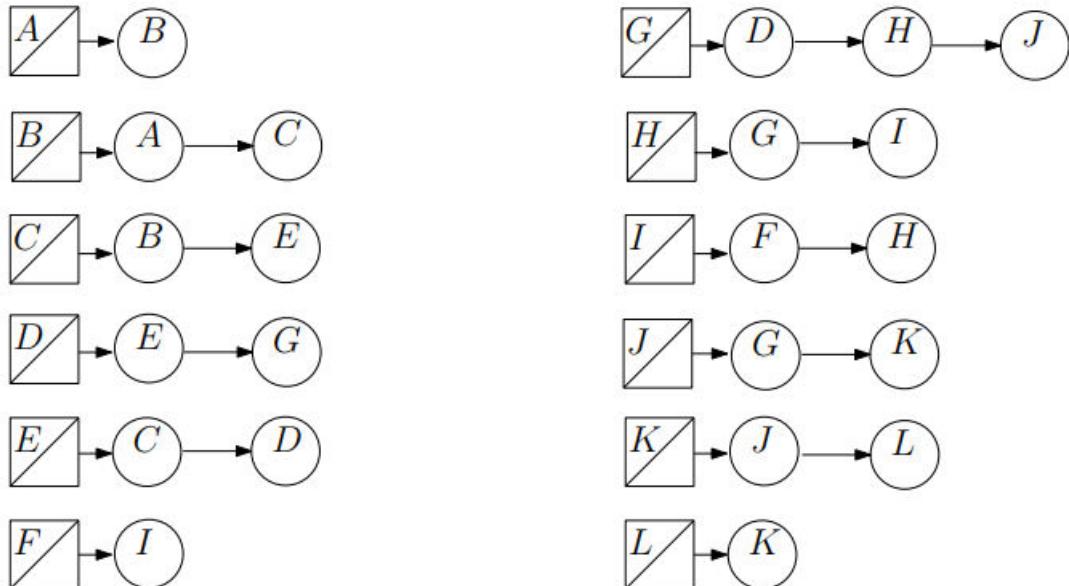
Σχήμα 2. Λίστα γειτνίασης λεξικογραφικά ταξινομημένη



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: $G(V, E)$, $parent[]$

```
1: void spanning_tree_dfs(int  $parent[]$ , 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  $parent[v] \neq 0$  then
7:      $N_T[v] \leftarrow N_T[v] \cup \{parent[v]\}$ ;
8:      $N_T[parent[v]] \leftarrow N_T[parent[v]] \cup \{v\}$ ;
9:   end if
10: end for
```

Μονοπάτι μεταξύ 2 κορυφών (ΔκΒ σε Μη-Κατευθυνόμενο)

Εκφώνηση

Άσκηση 2 Να διατυπώσετε μία έκδοση της ΔκΒ που να δέχεται σαν είσοδο δύο κορυφές s, t ενός μη-κατευθυνόμενου γραφήματος και να βρίσκει αν υπάρχει μονοπάτι που να τις συνδέει. Αν υπάρχει τέτοιο μονοπάτι να το καταγράφει.

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:  $visited[v] \leftarrow true$ ;
3:  $length++$ ;
4:  $path[length] \leftarrow v$ ;
5: if  $v = t$  then
6:    $found \leftarrow true$ ;
7: end if
8: if not  $found$  then
9:   for  $u \in N(v)$  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:    $path[length] \leftarrow 0$ ;  $length--$ ;
21: end if
```

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

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

Κύκλος (ΔκΒ σε Μη-Κατευθυνόμενο)

Εκφώνηση

Άσκηση 3 Να εκπουνήσετε μία έκδοση της ΔκΒ που να διαπιστώνει αν ένα μη-κατευθυνόμενο γράφημα έχει κύκλο και αν αυτό συμβαίνει να τον καταγράψει.

Λύση

Λύση Αρχικώς εκτελούμε ΔκΒ προκειμένου να πάρουν τιμές ο πίνακας *parent*. Όπως έχουμε ήδη δει αν υπάρχει οπισθοακμή αυτό σημαίνει ότι υπάρχει κύκλος. Όμως πως μπορούμε να αποτυπώσουμε τον κύκλο; Ο αλγόριθμος 3 αφού εντοπίζει την οπισθοακμή (v, u) μετά βρίσκει μονοπάτι από την κορυφή u στη v χρησιμοποιώντας τον Αλγόριθμο 2. Η πολυπλοκότητα του αλγόριθμου είναι $O(n + m)$.

Αλγόριθμος 3. ΔκΒ – Κύκλος

Algorithm 3 ΔκΒ - Κύκλος

Require: $G(V, E)$;

```
1: void find_cycle(int  $v$ , bool visited[], int parent[], int cycle[], int length, bool found)
2: clock  $\leftarrow 0$ ;
3: for  $v \in V$  do
4:   visited[ $v$ ]  $\leftarrow false$ ;
5:   preorder[ $v$ ]  $\leftarrow parent[v] \leftarrow 0$ ;
6: end for
7:  $s \leftarrow t \leftarrow 0$ ;
8: for  $v \in V$  do
9:   if not visited[ $v$ ] then
10:    explore(clock,  $v$ , visited, preorder, parent);
11:   end if
12: end for
13: // Ελέγχεται αν υπάρχει οπισθοακμή ( $v, u$ )//
14: backedge_found  $\leftarrow false$ ;
15: for  $v \in V$  and not backedge_found do
16:   for  $u \in N(v)$  do
17:      $s \leftarrow v$ ;  $t \leftarrow u$ ;
18:     if parent[ $t$ ]  $\neq s$  and parent[ $s$ ]  $\neq t$  then
19:       backedge_found  $\leftarrow true$ ;
20:       break;
21:     end if
22:   end for
23: end for
24: found  $\leftarrow false$ ;
25: if backedge_found then
26:   // Αναζήτηση μονοπατιού από  $s$  σε  $t$ //
27:   for  $v \in V$  do
28:     visited[ $v$ ]  $\leftarrow false$ ;
29:   end for
30:   length  $\leftarrow 0$ ;
31:   find_path( $s, t, visited, cycle, length, found$ );
32: else
33:   Print "There is no cycle: the graph is a tree"
34: end if
```

Γραφικές Συνιστώσες (ΔκΒ σε Μη-Κατευθυνόμενο)

Εκφώνηση

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

Δύση 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 $v \in V$ 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 πίστες γειτνίασης περιέχουν τις γειτονικές κορυφές σε αύξουσα σειρά επικέτας. Με τον ίδιο τρόπο (δηλαδή κατά αύξουσα σειρά επικέτας) γίνεται η απαριθμηση 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  $v \in V$  do
3:   visited[ $v$ ]  $\leftarrow$  false;
4:   comp_num  $\leftarrow$  0;
5:   comp[ $v$ ]  $\leftarrow$  0;
6: end for
7: for  $v \in V$  do
8:   if not visited[ $v$ ] then
9:     comp_num  $\leftarrow$  comp_num + 1;
10:    explore1( $v$ , visited, comp, comp_num);
11:   end if
12: end for
```

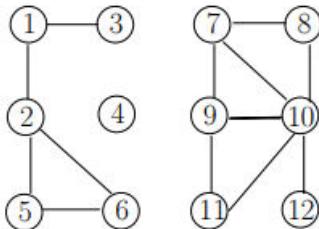
Αλγόριθμος 5. ΔκΒ - Γραφικές Συνιστώσες

Algorithm 5 ΔκΒ - Γραφικές συνιστώσες

Require: $G(V, E)$;

```
1: void explore1(int  $v$ , bool visited[], int comp[], int comp_num);
2: visited[ $v$ ]  $\leftarrow$  true;
3: comp[ $v$ ]  $\leftarrow$  comp_num;
4: for  $u \in N(v)$  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;
```

Δισυνεκτικές Συνιστώσες (ΔκΒ σε Μη-Κατευθυνόμενο)

Εκφόνηση

Άσκηση 5 Ένα γράφημα ονομάζεται δισυνεκτικό αν δεν έχει σημείο κοπής- για παράδειγμα το γράφημα στο Σχήμα 8. Όταν ένα γράφημα έχει σημεία κοπής τότε μπορεί να αποσυνδεθεί σε δισυνεκτικές συνιστώσες. Μία δισυνεκτική συνιστώσα είναι ένα μεγιστοτικό δισυνεκτικό υπογράφημα. Κάθε σημείο κοπής ανήκει σε τουλάχιστον δύο δισυνεκτικές συνιστώσες. Στο Σχήμα 9 απεικονίζεται ένα γράφημα με ένα σημείο κοπής (κορυφή D) και η αποσύνθεση του σε δύο δισυνεκτικές συνιστώσες (η κορυφή D ανήκει και στις δύο.) Μία ακμή με τις προσπίπτουσες κορυφές αποτελεί μία τετραμένη περίπτωση δισυνεκτικής συνιστώσας (εξ' ορισμού αφού η διαγραφή μίας εκ' των δύο κορυφών δεν αυξάνει τον αριθμό των γραφικών συνιστώσων). Επίσης, εκ' του ορισμού, ισχύει ότι δύο κορυφές που ανήκουν στην ίδια δισυνεκτική συνιστώσα (εξαιφουμένης της τετραμένης περίπτωσης) συνδέονται με δύο μονοπάτια διαζευγμένα ως προς τις κορυφές και τις ακμές τους και άρα υπάρχει ένας απλός κύκλος που τις περιλαμβάνει. Συνεπώς, και για κάθε ζευγάρι ακμών που βρίσκονται στην ίδια δισυνεκτική συνιστώσα υπάρχει ένας απλός κύκλος που τις περιλαμβάνει. Οι ιδιότητες αυτές είναι αρκετά χρήσιμες και έχουν αρκετές εφαρμογές. Για παράδειγμα, δεδομένων δύο κορυφών θέλουμε να γνωρίζουμε άν υπάρχει μία τρίτη κορυφή η διαγραφή της οποίας να αποσυνδέει τις κορυφές αυτές. Προφανώς η απάντηση είναι αρνητική στην περίπτωση που οι κορυφές αυτές βρίσκονται στην ίδια δισυνεκτική συνιστώσα.

Να εκπουνήσετε αλγόριθμο ο οποίος να υπολογίζει τις δισυνεκτικές συνιστώσες ενός γραφήματος.

Λύση

Λύση Θεωρούμε ότι γνωρίζουμε τα σημεία κοπής του γραφήματος (έχοντας εκτελέσει τον αλγόριθμο που έχουμε δει στα προηγούμενα). Παρατηρούμε ότι στο δένδρο της ΔκΒ, οι κορυφές που ανήκουν στην ίδια δισυνεκτική συνιστώσα εμπίπτουν σε μία από τις παρακάτω περιπτώσεις. Βρίσκονται είτε

- ανάμεσα στη ρίζα και σε ένα φύλλο,
- ανάμεσα στη ρίζα και σε σημείο κοπής
- ανάμεσα σε σημείο κοπής και σε φύλλο
- ανάμεσα σε δύο σημεία κοπής

Αξιοποιώντας την παραπάνω παρατήρηση μπορούμε να εμπλουτίσουμε την αλγορίθμική διαδικασία εύρεσης σημείων κοπής προκειμένου να εντοπίσουμε (καταγράψουμε) τις κορυφές κάθε δισυνεκτικής συνιστώσας ενός γραφήματος (μαζί μετην εύρεση των σημείων κοπής). Η εμπλουτισμένη εκδοσή της διαδικασίας comp_low απεικονίζεται στον Αλγόριθμο 6.

Ο Αλγόριθμος 6 χρησιμοποιεί μία αφηρημένη δομή δεδομένων - το σύνολο Q - στο οποίο αποθηκεύει όλες κάθε κορυφή αφού ολοκληρωθεί η επίσκεψη σε αυτή από τη ΔκΒ (Γραμμή 6). Όταν ανακαλυφθεί ένα σημείο κοπής (κορυφή v - Γραμμή 7) τότε όλες οι κορυφές που ανήκουν στο σύνολο Q ΚΑΙ βρίσκονται στο υποδένδρο της ΔκΒ με ρίζα την τελευταία κορυφή της οποίας την επίσκεψη έχει ολοκληρώση η διάσχιση (κορυφή u) αποτελούν τις κορυφές μίας δισυνεκτικής συνιστώσας μαζί με την v^1 . Οι κορυφές αυτές αφαιρούνται μία-μία από το σύνολο Q και τυπώνονται (Γραμμές 11-12) αφού προηγουμένως έχει τυπωθεί και η v . Σημειώνουμε ότι μετά από την αφαίρεση των κορυφών από το Q αυτό δεν είναι απαραίτητα άδειο αφού ενδέχεται να έχουν προηγουμένως προστεθεί σε αυτό και κορυφές από άλλη δισυνεκτική συνιστώσα. Ο Αλγόριθμος 6 καλείται στα πλαίσια του κύριου προγράμματος για τον υπολογισμό των σημείων κοπής (δες αντίστοιχη routina main στην παρουσίαση) με την προσθήκη της αρχικοποίησης του συνόλου Q (δηλαδή $Q \leftarrow \emptyset$). Για την υλοποίηση του Αλγόριθμου 6 με γλώσσα προγραμματισμού χρίζει ιδιαίτερης προσοχής ο χειρισμός της δομής Q καθώς και το πως θα προγραμματιστεί ο βρόχος των Γραμμών 10-13.

Παράδειγμα 2 Το γράφημα του Σχήματος 10 έχει ένα σημείο κοπής και δύο δισυνεκτικές συνιστώσες. Μια εικόνα των τιμών των βασικών δομών (μεταβλητών, συνόλο Q) του αλγόριθμου όταν η κορυφή 4 χαρακτηρίζεται σημείο τομής δίνεται στο Σχήμα 11.

Στο σχήμα, οι τιμές του διανύσματος preorder εμφανίζονται με μπλέ χρώμα, οι τιμές του low με πράσινο (οι προηγούμενες τιμές εμφανίζονται διαγραμμένες). Καθώς

$$low[6] = 6 \geq 3 = preorder[4]$$

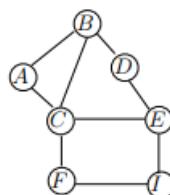
η κορυφή 4 αναγνωρίζεται σαν σημείο κοπής. Ακολούθως, το υπόδεινδρο της ΔΚΒ με ρίζα την κορυφή 6 περιέχει μόνο την κορυφή αυτή η οποία βρίσκεται αποδηκευμένη στο σύνολο Q . Η κορυφή αυτή μαζί με την κορυφή 4 αποτελούν τη μία δισυνεκτική συνιστώσα.

Παρατηρήστε ότι αφού θα ολοκληρωθεί η εξερεύνηση της κορυφής 4 αυτή θα προστεθεί στο σύνολο Q (κάτηση της συνάρτησης $compr_low_biccon$ με $v = 2, u = 4$.) Η δεύτερη δισυνεκτική συνιστώσα ανακαλύπτεται όταν επαναπροσεγγίζεται η ρίζα του δένδρου. Η κατάσταση αυτή ($v = 1, u = 2$) απεικονίζεται στο Σχήμα 12. Παρατηρούμε ότι

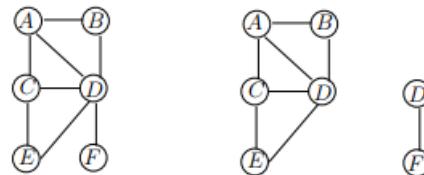
$$low[2] = 1 \geq 1 = preorder[1]$$

και επομένως οι κορυφές του συνόλου Q μαζί με την κορυφή 1 αποτελούν τη δεύτερη δισυνεκτική συνιστώσα. Οι δύο δισυνεκτικές συνιστώσες απεικονίζονται στο Σχήμα 13.

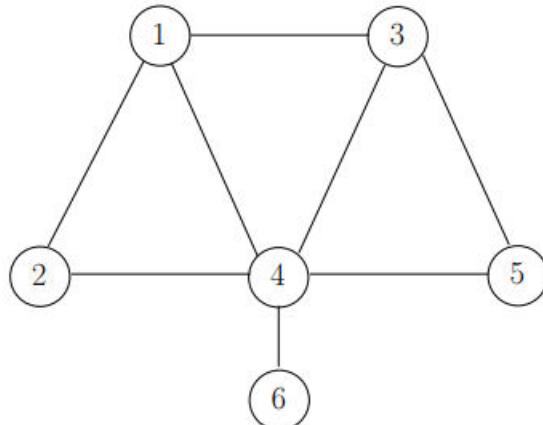
Σχήμα 8. Δισυνεκτικό Γράφημα



Σχήμα 9. Γράφημα με δύο Δισυνεκτικές Συνιστώσες



Σχήμα 10. Εύρεση δισυνεκτικών συνιστώσεων



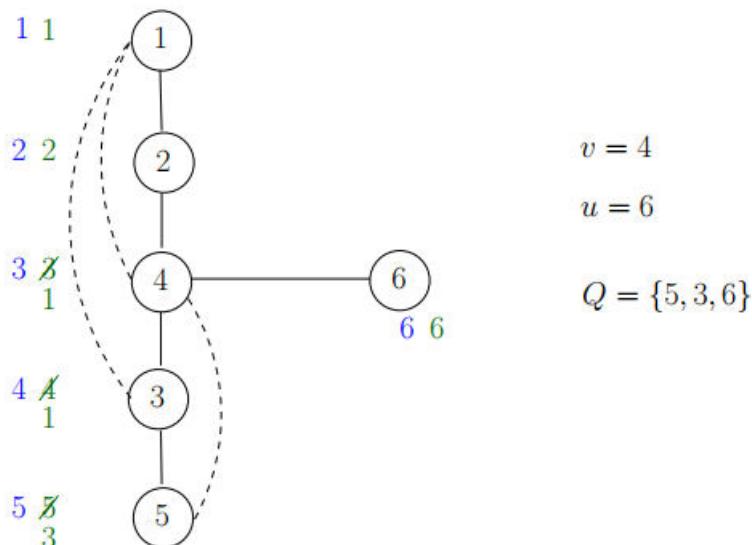
Αλγόριθμος 6. Δισυνεκτικές συνιστώσες

Algorithm 6 Δισυνεκτικές συνιστώσες

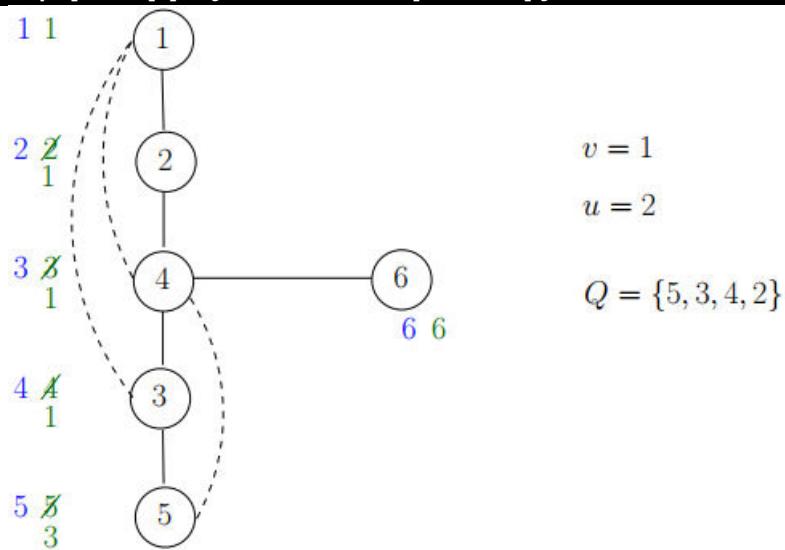
Require: $set Q$

```
1: void comp_low_biconn(int v, bool visited[], int preorder, int parent, bool cut_vertex[], set Q);  
2: visited[v] ← true; low[v] ← preorder[v];  
3: for  $u \in N(v)$  do  
4:   if not visited[u] then  
5:     comp_low_biconn(u, visited, preorder, parent, cut_vertex, Q);  
6:      $Q \leftarrow Q \cup \{u\}$ ;  
7:   if  $low[u] \geq preorder[v]$  then  
8:     cut vertex[v] ← true;  
9:     Print  $v$ ;  
10:    for all  $w \in Q : w$  κορυφή του υποδένδρο της ΔΚΒ με ρίζα το  $u$  do  
11:       $Q \leftarrow Q - \{w\}$ ;  
12:      Print  $w$ ;  
13:    end for  
14:   end if  
15: else  
16:   if not  $v = parent[u]$  then  
17:     if  $low[v] > preorder[u]$  then  
18:        $low[v] = preorder[u]$   
19:     end if  
20:   end if  
21: end if  
22: end for
```

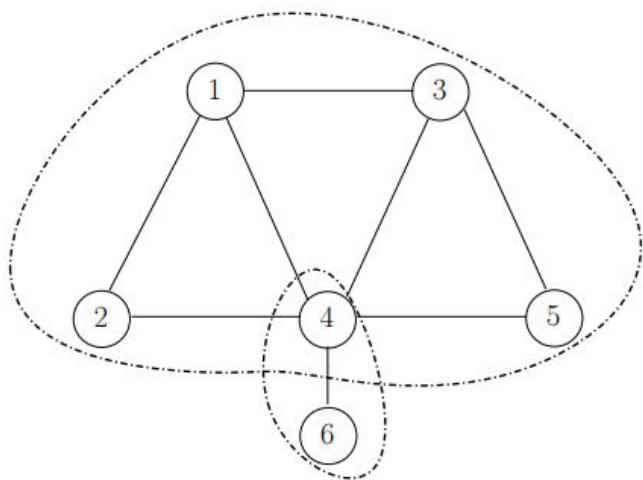
Σχήμα 11. Η κορυφή 4 αναγνωρίζεται σαν σημείο κοπής



Σχήμα 12. Επιστροφή στη ρίζα του δένδρου της ΔκΒ



Σχήμα 13. Οι δύο δισυνεκτικές συνιστώσεις του γραφήματος του σχήματος 10

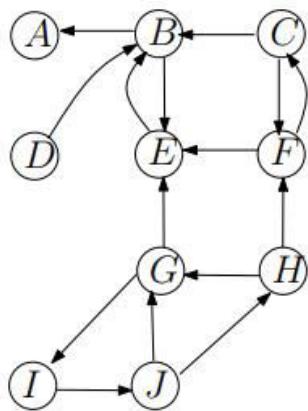


ΔκΒ σε Κατευθυνόμενο Γράφημα

Εκφώνηση

Άσκηση 6 Να εκτελέσετε ΔκΒ στο γράφημα του Σχήματος 14 θεωρώντας ότι οι κορυφές βρίσκονται στη λίστα γειτνίασης σε πλεικογραφική σειρά. Αφού υπολογίσετε τους αριθμούς preorder και postorder να χαρακτηρίσετε τις ακμές. Να απαντήσετε με βάση την κατηγοριοποίηση αν το γράφημα είναι άκυκλο.

Σχήμα 14. Κατευθυνόμενο Γράφημα

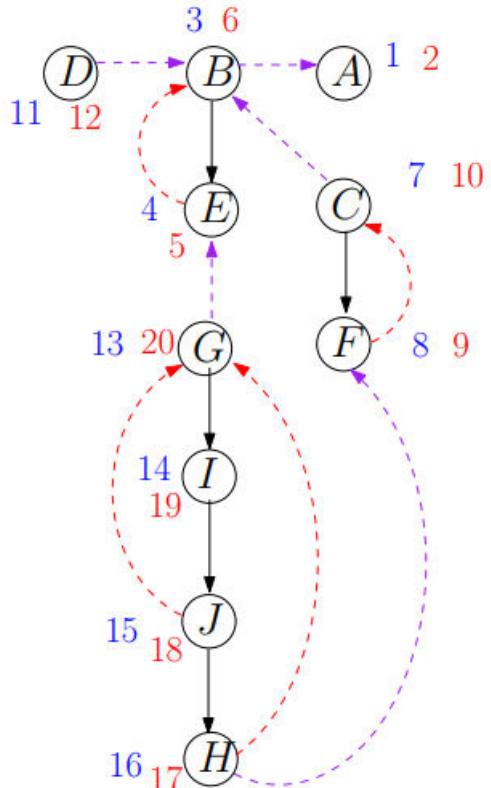


Σχήμα 14: Κατευθυνόμενο Γράφημα

Λύση

Λύση Στο Σχήμα 15 απεικονίζεται το δένδρο της ΔκΒ (μπλε για τις τιμές *preorder* και κάκκινο για τις τιμές *postorder*). Οι εμπροσθοακμές απεικονίζονται με συμπαγείς γραμμές. Οι ακμές με μώβ χρώμα είναι εγκάρσιες ενώ αυτές με κόκκινο οπισθοακμές. Η ύπαρξη οπισθοακμών υπονοεί ότι το γράφημα δεν είναι άκυκλο (περιέχει (κατευθυνόμενο) κύκλο).

Σχήμα 15. Δένδρο ΔκΒ σε Κατευθυνόμενο Γράφημα



Κύκλος (ΔκΒ σε Κατευθυνόμενο)

Εκφώνηση

Άσκηση 7 Να εκπονήσετε αλγόριθμο για τον εντοπισμό ενός κύκλου σε ένα κατευθυνόμενο γράφημα. Αν το γράφημα δεν είναι Ακυκλο Κατευθυνόμενο Γράφημα ο αλγόριθμος πρέπει να επιστρέψει έναν κύκλο

Λύση

Λύση Ο ζητούμενος αλγόριθμος μπορεί να προκύψει με μικρές αλλαγές του Αλγόριθμου 3. Η πρώτη αλλαγή είναι ότι όπου αναφέρεται $N(v)$ αντικαθίσταται από το $N^+(v)$ (το ίδιο και στη διαδικασία `find_path`.)

Η επικεφαλίδα του ζητούμενου αλγόριθμου (γραμμή 1, Αλγόριθμος 3) επαναδιατυπώνεται ως

```
void find_cycle(int v, bool visited[], int preorder[], int postorder[], int cycle[], int length, bool found);
```

Από την παραπάνω δήλωση είναι προφανές ότι προϋπόθεση αποτελεί να έχει εκτελεστεί ΔκΒ και οι πίνακες *preorder*, *postorder* έχουν πάρει τιμές. Στη συνέχεια αρκεί να αντικαταστήσουμε τη γραμμή 7 (Αλγόριθμος 3) με την

```
if preorder[u] < preorder[v] AND postorder[v] < postorder[u] then
```

Ουσιαστικά η παραπάνω γραμμή εντοπίζει αν η ακμή (v, u) είναι οπισθοακμή.

Αλγόριθμος 3. ΔκΒ – Κύκλος (Ανανεωμένος)

Algorithm 3 ΔκΒ - Κύκλος

Require: $G(V, E)$;

```
1: void find_cycle(int  $v$ , bool  $visited[]$ , int  $preorder[]$ , int  $postorder[]$ , int  $cycle[]$ , int  $length$ , bool  $found$ );
2:  $clock \leftarrow 0$ ;
3: for  $v \in V$  do
4:    $visited[v] \leftarrow false$ ;
5:    $preorder[v] \leftarrow parent[v] \leftarrow 0$ ;
6: end for
7:  $s \leftarrow t \leftarrow 0$ ;
8: for  $v \in V$  do
9:   if not  $visited[v]$  then
10:    explore( $clock$ ,  $v$ ,  $visited$ ,  $preorder$ ,  $parent$ );
11:   end if
12: end for
13: // Ελέγχεται αν υπάρχει ομοθοακμή ( $v, u$ ) //
14:  $backedge\_found \leftarrow false$ ;
15: for  $v \in V$  and not  $backedge\_found$  do
16:   for  $u \in N^+(v)$  do
17:     if  $preorder[u] < preorder[v]$  AND  $postorder[v] < postorder[u]$  then
18:       if  $parent[t] \neq s$  and  $parent[s] \neq t$  then
19:          $backedge\_found \leftarrow true$ ;
20:         break;
21:       end if
22:     end for
23:   end for
24:    $found \leftarrow false$ ;
25:   if  $backedge\_found$  then
26:     // Αναζητηση μονοπατου από  $s$  σε  $t$  //
27:     for  $v \in V$  do
28:        $visited[v] \leftarrow false$ ;
29:     end for
30:      $length \leftarrow 0$ ;
31:     find_path( $s, t, visited, cycle, length, found$ );
32:   else
33:     Print "There is no cycle: the graph is a tree"
34:   end if
```

Τοπολογική Ταξινόμηση (ΔκΒ σε Κατευθυνόμενο)

Εκφώνηση

Άσκηση 8 Σε ένα ΑΚΓ οι κορυφές μπορούν να διαταχθούν κατά τρόπο ώστε να μην υπάρχει ακμή από μία κορυφή που να έπειται στη διάταξη προς μία ακμή που να προηγείται. Μία τέτοια διάταξη ονομάζεται τοπολογική ταξινόμηση. Για παράδειγμα στο Σχήμα 16 απεικονίζεται ένα ΑΚΓ και μία τοπολογική ταξινόμηση των κορυφών. Παρατηρήστε ότι η ταξινόμηση αυτή δεν είναι μοναδική. Να εκπονήσετε αλγόριθμο ο οποίος να δέχεται σαν εισοδού ένα ΑΚΓ και να παράγει μια τοπολογική ταξινόμηση.

Λύση

Λύση Ο αλγόριθμος στον πυρήνα του αποτελείται από την αναδρομική διαδικασία που απεικονίζεται στον Αλγόριθμο 7. Η διαδικασία χρησιμοποιεί μία δομή στοίβας Q στην οποία αποθηκεύεται διαδοχικά τις κορυφές.

Αν ανακαλύψει ότι υπάρχει κορυφή u που δεν έχει επισκεφτεί και είναι γείτονας της κορύφης v που μόλις εισήγαγε στη στοίβα, βγάζει την v από τη στοίβα και την προσθέτει αφού ολοκληρωθεί η εξερεύνηση στους γείτονες της u . Η διαδικασία `top_ordering` καλείται από τον Αλγόριθμο 8 όπου αρχικοποιούνται οι βασικές δομές.

Αλγόριθμος 7. Τοπολογική Ταξινόμηση

Algorithm 7 Τοπολογική Ταξινόμηση

Require: $G(V, E)$

```
1: void top_ordering(int v, bool visited[], stack Q);
2: visited[v] ← true;
3: push(v, Q);
4: for  $u \in N^+(v)$  do
5:   if not visited[u] then
6:     pop(Q);
7:     top_ordering(u, visited, Q);
8:     push(v, Q);
9:   end if
10: end for
```

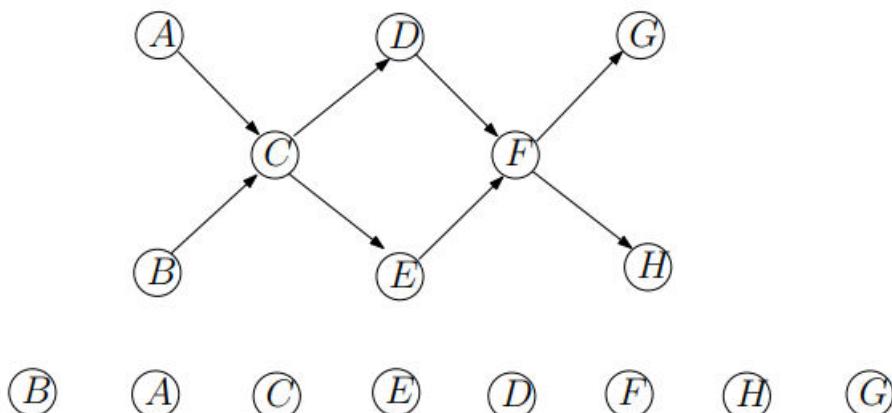
Αλγόριθμος 8. Τοπολογική Ταξινόμηση (Κύριο Πρόγραμμα)

Algorithm 8 Τοπολογική Ταξινόμηση (Κύριο πρόγραμμα)

Require: $G(V, E)$

```
1: void top_ord()
2: create Q;
3: for  $v \in V$  do
4:   visited[v] ← false;
5: end for
6: for  $v \in V$  do
7:   if not visited[v] then
8:     top_ordering(v, visited, Q);
9:   end if
10: end for
11: print Q;
```

Σχήμα 16. ΑΚΓ και μία τοπολογική ταξινόμηση των κορυφών του



Θεωρία πάνω στο Δένδρο Συντομότερων Μονοπατιών

Εκφόνηση

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.

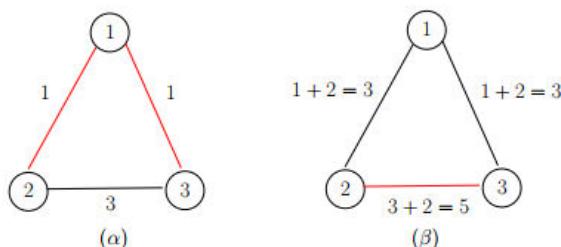
1. Πολλαπλασιασμός των συντελεστών των ακμών με αριθμό $b > 0$.
2. Αύξηση του συντελεστή των ακμών κατά $b > 0$.
3. Αντιστροφή των συντελεστών των ακμών.

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 1a 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.

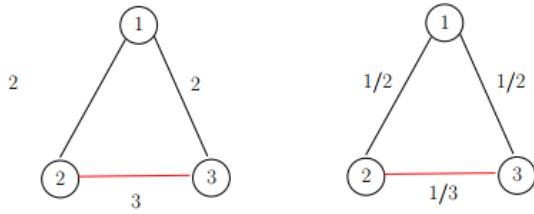


Σχήμα 1: Συντομότερα μονοπάτια - αύξηση των συντελεστών των ακμών κατά μία σταθερά

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 1a 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

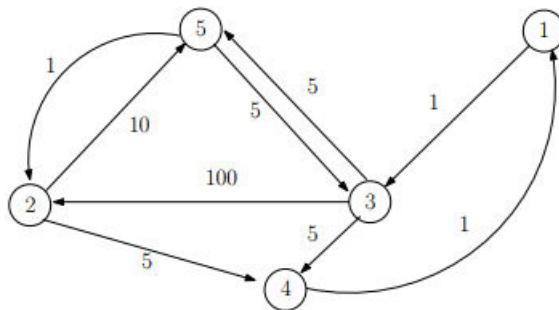
Εφαρμογή Dijkstra σε γράφημα

Εκφώνηση

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

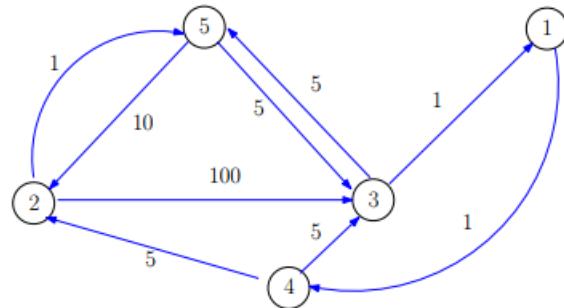


Λύση

Δεδομένου του κατευθυνόμενου γραφήματος $G(V, E, w)$, δημιουργούμε το γράφημα \bar{G} ως εξής. $V(\bar{G}) = V(G)$ 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:

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

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:

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

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:

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

Επανάληψη 4

Φάση I

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

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

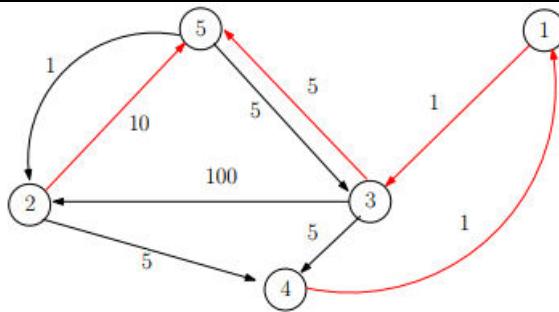
Φάση II

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

Τέλος Επανάληψης:

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

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



Μοντελοποίηση ενός προβλήματος με εφαρμογή του Dijkstra

Εκφώνηση

Άσκηση 3 Σε μία πόλη υπάρχουν κάποιες δομές που παρέχουν υπηρεσίες κοινής αφέλειας (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 όμεσες 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 διάδεση 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

Άστοι Μπορούμε να μοντελοποιήσουμε το πρόβλημα με τη χρήση ενός κατευθυνομένου συνεκτικού γραφήματος $G(V, E, w)$ 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 συνέχεια θα 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 συνέχεια 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 .

Το πρόβλημα του χτισίματος ενός σπιτιού (DAG)

Εκφώνηση

Άσκηση 4 Για το χτίσιμο ενός σπιτιού απαιτούνται επιμέρους εργασίες οι οποίες απεικονίζονται στον Πίνακα 1. Να βρεθεί ο επιλάχιστος αριθμός των ημερών που χρειάζεται προκειμένου να ολοκληρωθεί το χτίσιμο του σπιτιού.

Πίνακας 1. Λίστα δραστηριοτήτων

| Αρ. Δραστ. | Περιγραφή | Διάρκεια (Ημέρες) | πριν από Δραστ. |
|------------|-----------------|-------------------|-----------------|
| 1 | Προεργασία | 2 | 2 |
| 2 | Θεμέλια | 4 | 3 |
| 3 | Ανέγερση Τοίχων | 10 | 4,6,7 |
| 4 | Υδραυλικά (Εξ.) | 4 | 5,9 |
| 5 | Υδραυλικά (Εσ.) | 5 | 10 |
| 6 | Ηλεκτρολογικά | 7 | 10 |
| 7 | Στέγη | 6 | 8 |
| 8 | Σοβάτισμα | 7 | 9 |
| 9 | Βάψιμο (Εξ.) | 9 | 14 |
| 10 | Κουφώματα | 8 | 11,12 |
| 11 | Πατώματα | 4 | 13 |
| 12 | Βάψιμο (Εσ.) | 5 | 13 |
| 13 | Ολοκλήρωση Εσ. | 6 | |
| 14 | Ολοκλήρωση Εξ. | 2 | |

Πίνακας 1: Λίστα δραστηριοτήτων

Λύση

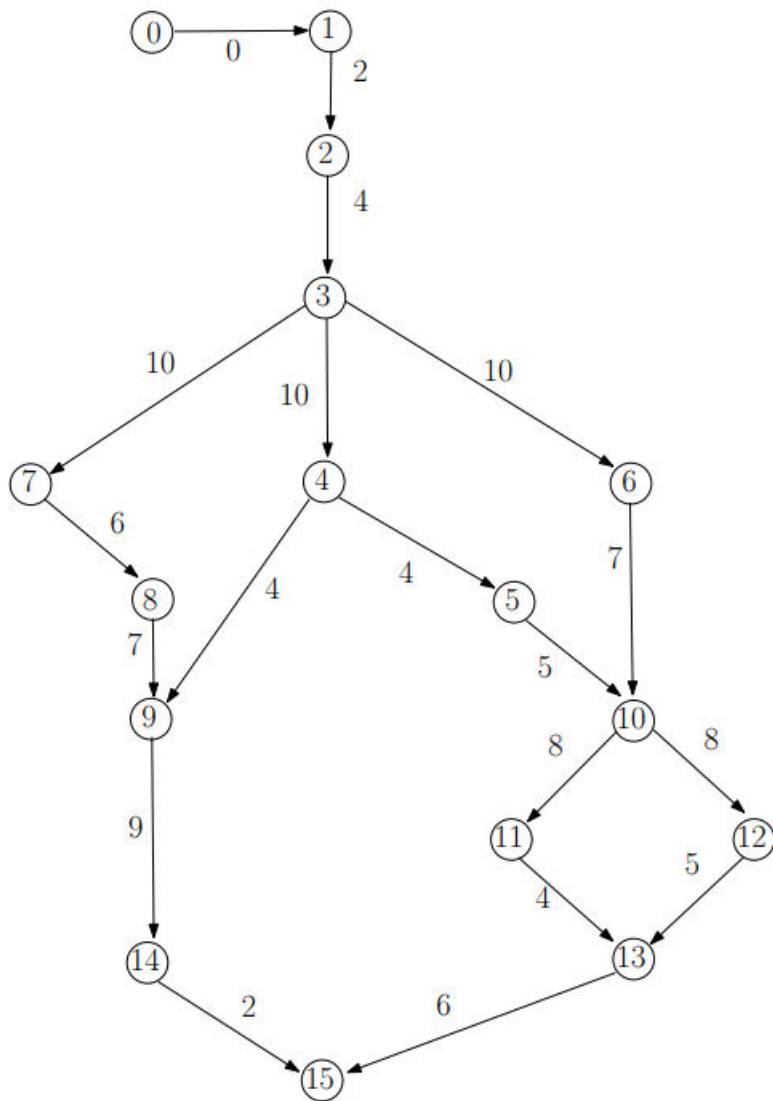
Λύση Κατασκευάζουμε ένα γράφημα με κορυφές τις δραστηριότητες. Για κάθε ζευγάρι δραστηριοτήτων i,j για το οποίο η δραστηριότητα i προηγείται από τη δραστηριότητα j εισάγουμε την ακμή (i,j) με συντελεστή (βάρος) τη διάρκεια της δραστηριότητας i . Επίσης εισάγουμε δυο ψευδοδραστηριότητες: η δραστηριότητα 0 η οποία προηγείται από κάθε δραστηριότητα με διάρκεια 0 και μία δραστηριότητα η οποία έπαιτε από κάθε άλλη δραστηριότητα. Το γράφημα που αντιστοιχεί στον Πίνακα 1 απεικονίζεται στο Σχήμα 6.

Παρατηρήστε ότι το project είναι δυνατό αν το γράφημα των δραστηριοτήτων είναι άκυκλο. Επίσης ο συντομότερος χρόνος που μπορεί να ολοκληρωθεί το έργο δίνεται από το μήκος του μακρύτερου (μεγαλύτερου) μονοπατιού που συνδέει τις δυο ψευδοκορυφές (από την κορυφή 0 μέχρι την κορυφή 15 στο παράδειγμα μας).

Από τα παραπάνω προκύπτει ότι μπορούμε να απαντήσουμε στο ερώτημα της Άσκησης υπολογίζοντας τη συντομότερη διαδρομή από την κορύφη 0 μέχρι την κορυφή 15 στο γράφημα δραστηριοτήτων με βάρη τα αρχικά πολλαπλασιάζοντας τα με -1. Για τον υπολογισμό μπορούμε να χρησιμοποιήσουμε τον αλγόριθμο επίλυσης του προβλήματος ΜΠΣΔ σε ΚΑΓ (δες αντίστοιχη παρουσίαση). Για την εκτέλεση του αλγόριθμου χρησιμοποιούμε την ακόλουθη σειρά για τις κορυφές (που προέρχεται από την τοπολογική ταξινόμηση των κορυφών αφού το γράφημα είναι ΚΑΓ).

0, 1, 2, 3, 4, 6, 7, 8, 5, 9, 10, 14, 11, 12, 13, 15.

Σχήμα 6. Γράφημα δραστηριοτήτων για το χτίσιμο σπιτιού (Πίνακας 1)



Εκτέλεση του αλγορίθμου DAG στο γράφημα του Σχήματος 6

Οι αποστάσεις των ελάχιστων διαδρομών από την κορυφή 0 υπολογίζονται ως εξής.

$$d_0 = 0,$$

$$d_1 = \min\{d_0 - w(0, 1)\} = 0 + 0 = 0, \quad p_1 = 0,$$

$$d_2 = \min\{d_1 - w(1, 2)\} = 0 - 2 = -2, \quad p_2 = 1,$$

$$d_3 = \min\{d_2 - w(2, 3)\} = -2 - 4 = -6, \quad p_3 = 2,$$

$$d_4 = \min\{d_3 - w(3, 4)\} = -6 - 10 = -16, \quad p_4 = 3,$$

$$d_6 = \min\{d_3 - w(3, 6)\} = -6 - 10 = -16, \quad p_6 = 3,$$

$$d_7 = \min\{d_3 - w(3, 7)\} = -6 - 10 = -16, \quad p_7 = 3,$$

$$d_8 = \min\{d_7 - w(7, 8)\} = -16 - 6 = -22, \quad p_8 = 7,$$

$$d_5 = \min\{d_4 - w(4, 5)\} = -16 - 4 = -20, \quad p_5 = 4,$$

$$d_9 = \min\{d_4 - w(4, 9), d_8 + w(8, 9)\} = \min\{-16 - 4, -22 - 7\} = -29, \quad p_9 = 8,$$

$$d_{10} = \min\{d_5 - w(5, 10), d_6 + w(6, 10)\} = \min\{-20 - 5, -16 - 7\} = -25, \quad p_{10} = 5,$$

$$d_{14} = \min\{d_9 - w(9, 14)\} = -29 - 9 = -38, \quad p_{14} = 9,$$

$$d_{11} = \min\{d_{10} - w(10, 11)\} = -25 - 8 = -33, \quad p_{11} = 10,$$

$$d_{12} = \min\{d_{10} - w(10, 12)\} = -25 - 8 = -33, \quad p_{12} = 10,$$

$$d_{13} = \min\{d_{11} - w(11, 13), d_{12} + w(12, 13)\} = \min\{-33 - 4, -33 - 5\} = -38, \quad p_{13} = 12,$$

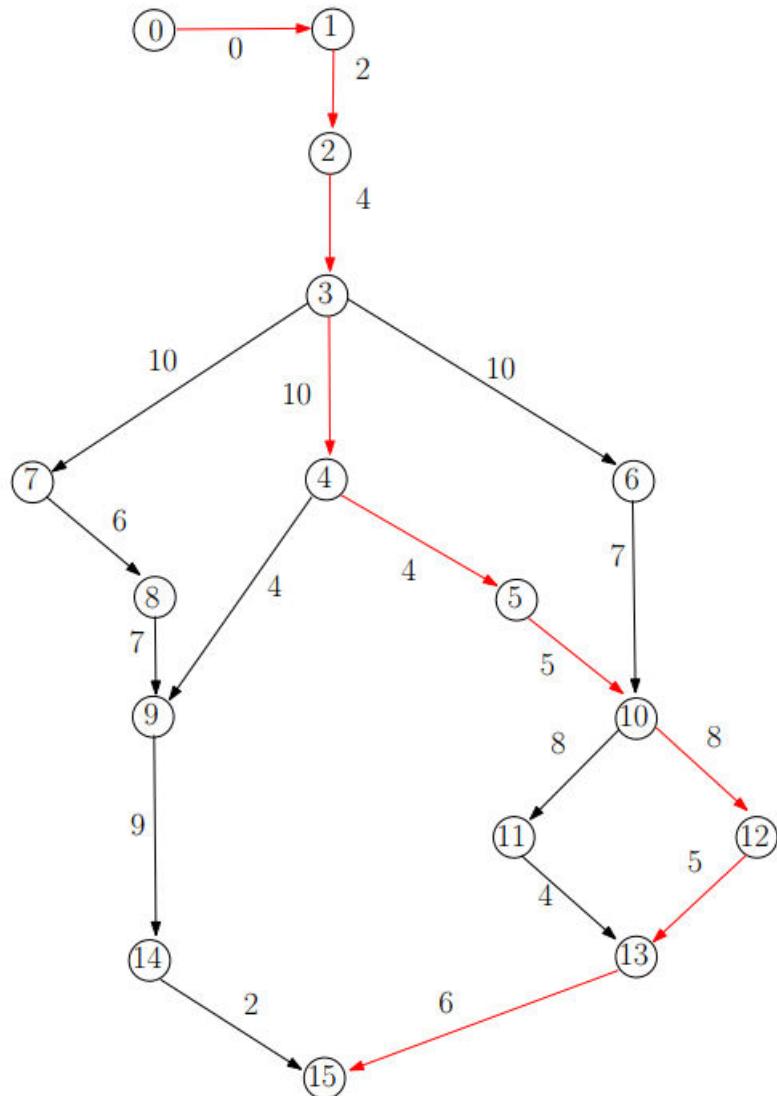
$$d_{15} = \min\{d_{13} - w(13, 15), d_{14} + w(14, 15)\} = \min\{-38 - 6, -38 - 2\} = -44, \quad p_{15} = 13.$$

Σύνοψη

Επομένως το έργο (χτίσιμο σπιτιού) μπορεί να ολοκληρωθεί το συντομότερο σε 44 ημέρες. Οι δραστηριότητες η διάρκεια των οποίων πρέπει να μειωθεί προκειμένου να συντομεύθει το έργο (κρίσιμες δραστηριότητες) δίνονται από το διάνυσμα p , ήτοι,

$$p = [0, 1, 2, 3, 4, 3, 3, 7, 8, 5, 10, 10, 12, 9, 13].$$

Σχήμα 7. Γράφημα δραστηριοτήτων – με κόκκινο οι ακμές των κρίσιμων δραστηριοτήτων



Το πρόβλημα του ταξιδιώτη (Dijkstra)

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 $v \in V$. Υποθέστε 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 φδιωτέρο 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 επιδυνυμία του ταξιδιώτη.

Υποθέστε 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 επιδυνυμία του.

Λύση

Δύση 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 $1, 1 \cdot Dijkstra(V, E, w, s, t) \geq Dijkstra(V, E, w, s, u) + Dijkstra(V, E, w, u, t)$ **then**

Ο ταξιδιώτης μπορεί να μείνει στην πόλη u

else

Ο ταξιδιώτης δεν μπορεί να μείνει στην πόλη u

end if

Θεωρία πάνω στον αλγόριθμο των Bellman-Ford

Εκφώνηση

Άσκηση 6 Να αποδείξετε ότι αν υπάρχει ένας αρνητικός κύκλος σε ένα εμβαρές κατευθυνόμενο γράφημα, ο αλγόριθμος των Bellman-Ford δύναται να το εντοπίσει.

Λύση

Δύση 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 προσπελαθεί από ένα μονοπάτι που εκκινεί από το s . Το μήκος του κύκλου είναι

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

Ο αλγόριθμος βρίσκει ότι υπάρχει αρνητικός κύκλος αν υπάρχει ακμή (v, u) για την οποία ισχύει ότι¹

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

Έστω λοιπόν ότι δεν ισχυεί αυτό για καμμία ακμή του αρνητικού κύκλου. Έχουμε δηλαδή ότι

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

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

$$\sum_{i=1}^k d_{v_i}^{|V|-1} \leq \sum_{i=1}^k d_{v_{i-1}}^{|V|-1} + \sum_{i=1}^k w(v_{i-1}, v_i). \quad (2)$$

Παρατηρούμε ότι

$$\sum_{i=1}^k d_{v_i}^{|V|-1} = \sum_{i=1}^{k-1} d_{v_i}^{|V|-1} + d_{v_k}^{|V|-1},$$

$$\sum_{i=1}^k d_{v_{i-1}}^{|V|-1} = \sum_{i=1}^{k-1} d_{v_i}^{|V|-1} + d_{v_0}^{|V|-1}$$

και επομένως η (2) γίνεται

$$d_{v_k}^{|V|-1} \leq d_{v_0}^{|V|-1} + \sum_{i=1}^k w(v_{i-1}, v_i). \quad (3)$$

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

$$\sum_{i=1}^k w(v_{i-1}, v_i) \geq 0 \quad (4)$$

το οποίο έρχεται σε αντίφαση με την (1).

Το πρόβλημα του arbitrage (Bellman-Ford)

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 συνίσταται στον εντοπισμό μίας σειράς ανταλλαγών που ξεκινώνται από κάποιο υόμισμα καταλήγει στο ίδιο υόμισμα 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

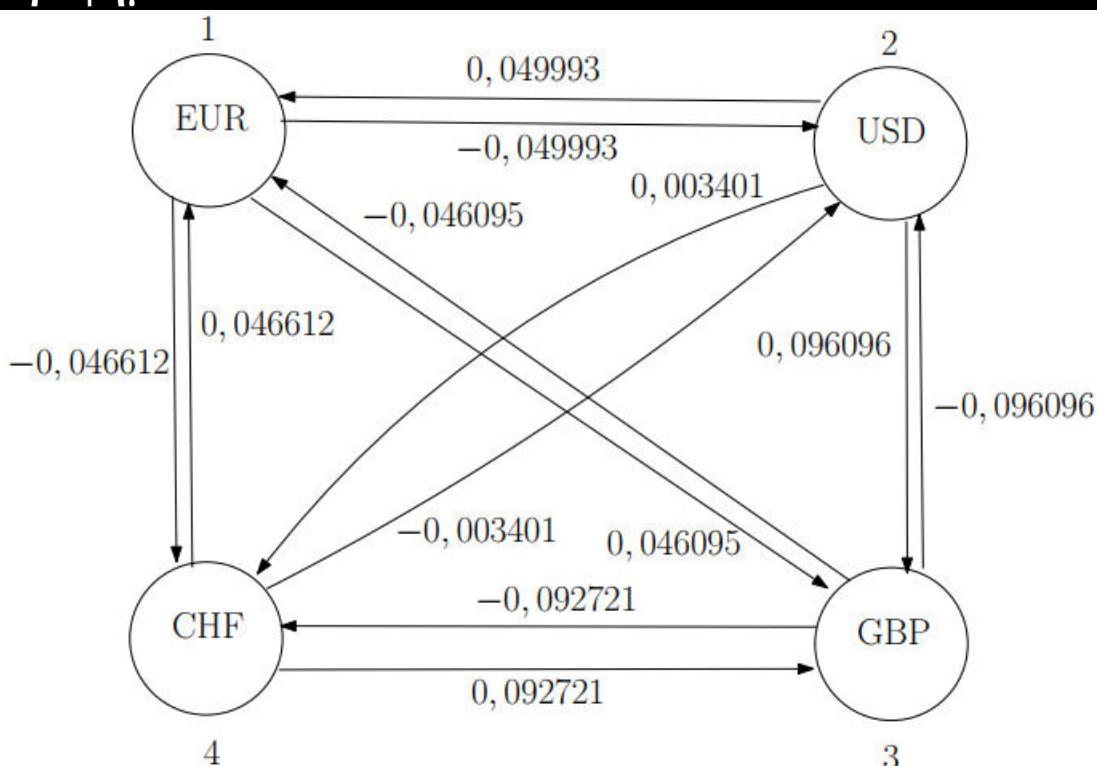
Λύση Για να αντιληφθούμε καλύτερα τους αριθμούς (τις ισοτιμίες) του Πίνακα 2 αν έχουμε 500 ευρώ μπορούμε να πάρουμε $500 * 1,122 = 561$ δολλάρια. Εναλλακτικά (μία περίπτωση), θα μπορούσαμε να κάνουμε τα ευρώ λίρες και τις λίρες δολλάρια. Στην περίπτωση αυτή θα είχαμε $500 * 0,8993 * 1,247661 = 561,01$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

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

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

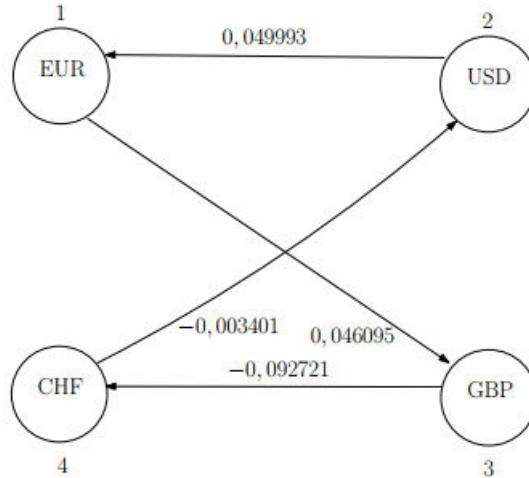
$$d^3 = [-0.00002, -0, 050027, 0, 046083, -0, 046626] \quad (5)$$

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

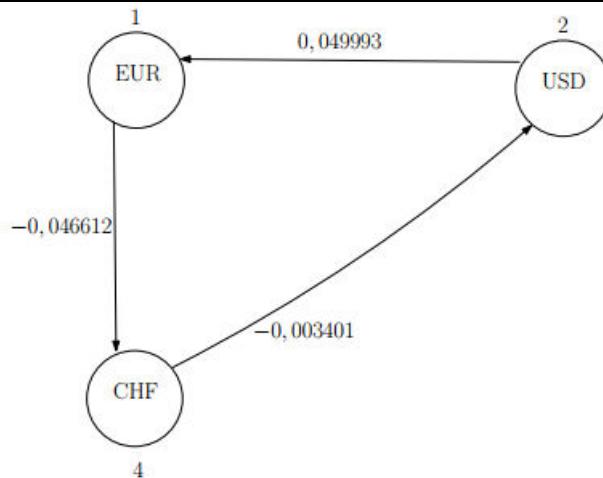
Πίνακας 4. Αρνητικοί κύκλοι στους οποίους έχει πρόσβαση η κορυφή 1

| # | (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) | $d_4^3 = -0, 046626$ | $d_3^3 + w(3, 4)$ | $4 \leftarrow 3 \leftarrow 2 \leftarrow 4 \leftarrow 1$ | $0, 046083 + (-0, 092721)$ $= -0.046638$ |

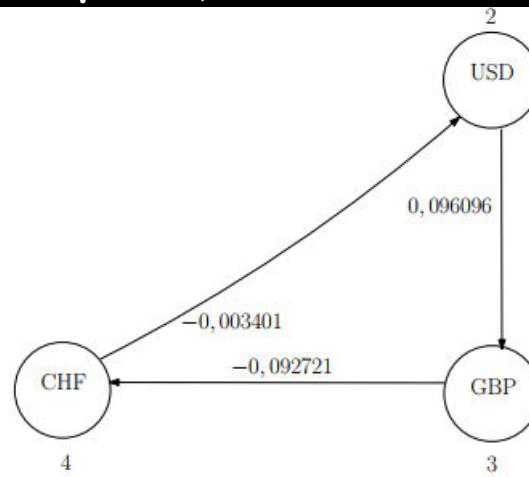
**Σχήμα 9. Κύκλος που αντιστοιχεί στην πρώτη γραμμή του Πίνακα 4.
Άθροισμα συντελεστών ακμών -0,000034**



Σχήμα 10. Κύκλος που αντιστοιχεί στην δεύτερη και τέταρτη γραμμή του Πίνακα 4. Άθροισμα 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 $d_v^3 > d_u^3 + w(u, v)$. Οι ακμές για τις οποίες συνθαίνει αυτό παρατίθενται στο Πίνακα 4. Οι κύκλοι αρνητικού μήκους που εντοπίστηκαν στον Πίνακα 4 απεικονίζονται στα Σχήματα 9, 10 και 11.

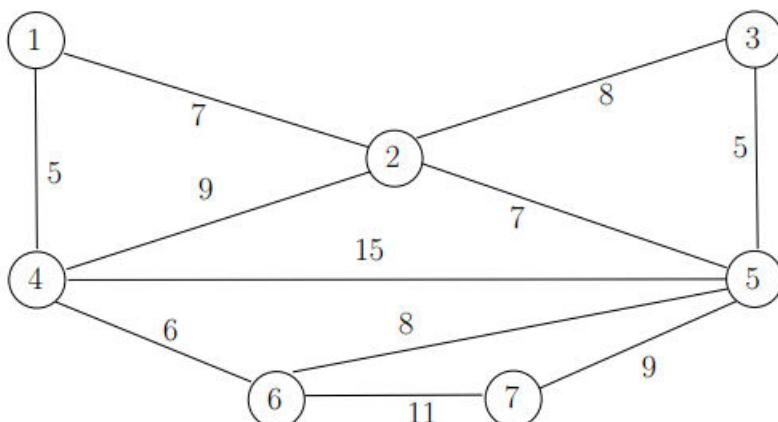
Μόνο οι κύκλοι που εμφανίζονται στα Σχήματα 9, 10 περιέχουν την κορυφή 1 (θέση σε ευρώ). Από αυτούς ο πρώτος έχει πολλαπλασιαστή (γινόμενο αρχικών ισοτιμιών) $10^{0.000034} = 1,00007829$ ενώ ο δεύτερος $10^{0.00002} = 1,000046$. Επομένως, επιλέγουμε τον πρώτο κύκλο αφού ο πολλαπλασιαστής του είναι μεγαλύτερος 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.

Εύρεση ΕΣΔ (Αλγόριθμος Prim)

Εκφώνηση

Άσκηση 8 Να περιγράψετε μία υλοποίηση του αλγόριθμου του Prim πολυπλοκότητας $O(n^2)$ όπου $n = |V|$. Στη συνέχεια να χρησιμοποιήσετε τον αλγόριθμο αυτό για την εύρεση του ΕΣΔ στο γράφημα του Σχήματος 12

Σχήμα 12. Εύρεση ΕΣΔ



Λύση

Λύση Ο Αλγόριθμος 1 δέχεται σαν είσοδο ένα βεβαρυμένο μη-κατευθυνόμενο γράφημα $G(V, E, w)$ και παράγει σαν έξοδο ένα ΕΣΔ $T(V_T, E_T)$.

Ο Αλγόριθμος 1 εκτελεί $n-1$ επαναλήψεις στο εξωτερικό βρόχο while (γραμμή 8). Σε μία τυχαία επανάληψη k το σύνολο αυτό έχει $n-k$ στοιχεία. Στη χειρότερη περίπτωση όλα αυτά τα στοιχεία βρίσκονται και στο σύνολο $N(v)$ (δηλαδή στη χειρότερη περίπτωση $S = N(v)$). Άρα εκτελούνται $n-k$ συγκρίσεις (γραμμές 8-12). Στη συνέχεια για να βρεθεί το ελάχιστο από τα $n-k$ στοιχεία απαιτούνται $n-k-1$ συγκρίσεις (γραμμές 14-18).

Αλγόριθμος 1. Αλγόριθμος Prim

Algorithm 1 Αλγόριθμος Prim

```
1:  $V_T \leftarrow \emptyset; E_T \leftarrow \emptyset; S \leftarrow V;$ 
2: for  $v \in V$  do
3:    $min\_V(v) \leftarrow NULL; min\_V\_weight(v) \leftarrow \infty;$ 
4: end for
5:  $v \leftarrow 1;$ 
6: while  $|V_T| \leq |V| - 1$  do
7:    $V_T \leftarrow V_T \cup \{v\}; S \leftarrow S - \{v\}$ 
8:   for  $u \in N(v) \cap S$  do
9:     if  $min\_V\_weight(u) > w(v, u)$  then
10:       $min\_V\_weight(u) \leftarrow w(v, u); min\_V(u) \leftarrow v;$ 
11:    end if
12:   end for
13:    $min\_S \leftarrow NULL; min\_S\_weight \leftarrow \infty;$ 
14:   for  $s \in S$  do
15:     if  $min\_S\_weight > min\_V\_weight(s)$  then
16:        $min\_S\_weight \leftarrow min\_V\_weight(s); min\_S \leftarrow s;$ 
17:     end if
18:   end for
19:    $E_T \leftarrow E_T \cup \{(min\_V(min\_S), min\_S)\}$ 
20:    $v \leftarrow min\_S;$ 
21: end while
22:  $V_T \leftarrow V_T \cup \{v\};$ 
```

Πολυπλοκότητα

Συνολικά σε κάθε επανάληψη γίνονται $2(n - k) - 1$ συγκρίσεις. Επειδή το k παίρνει τιμές από 1 ως $n - 1$, ο συνολικός αριθμός των συγκρίσεων είναι

$$\begin{aligned} T(n) &= \sum_{k=1}^{n-1} (2(n - k) - 1) = 2 \sum_{k=1}^{n-1} (n - k) - n \\ &= 2((n - 1) \cdot n - \sum_{k=1}^{n-1} k) - n = 2(n(n - 1) - n(n - 1)/2) - n = n(n - 1) - n \\ &= n^2 - 2n \Rightarrow T(n) = O(n^2). \end{aligned}$$

Κάνοντας χρήση ειδικών δομών δεδομένων (σωρός Fibonacci) ο αλγόριθμος μπορεί να υλοποιηθεί με πολυπλοκότητα $O(|E| + |V| \lg |V|)$.

Αρχικοποίηση

```
 $V_T = \emptyset; E_T = \emptyset; S = V;$ 
 $min\_V\_weight = [\infty, \infty, \infty, \infty, \infty, \infty, \infty];$ 
 $min\_V = [ , , , , , , ];$ 
 $v = 1; |V_T| = 0;$ 
```

Επανάληψη 1

```
 $V_T = \{1\}; S = \{2, 3, 4, 5, 6, 7\};$ 
 $min\_V\_weight = [\infty, 7, \infty, 5, \infty, \infty, \infty];$ 
 $min\_V = [ , 1, , 1, , , ];$ 
 $min\_S = 4; min\_S\_weight = 5;$ 
 $E_T = \{(1, 4)\}; v = 4;$ 
```

Επανάληψη 2

$V_T = \{1, 4\}; S = \{2, 3, 5, 6, 7\};$
 $min_V_weight = [\infty, 7, \infty, 5, 15, 6, \infty];$
 $min_V = [, 1, , 1, 4, 4,];$
 $min_S = 6; min_S_weight = 6;$
 $E_T = \{(1, 4), (4, 6)\}; v = 6;$

Επανάληψη 3

$V_T = \{1, 4, 6\}; S = \{2, 3, 5, 7\};$
 $min_V_weight = [\infty, 7, \infty, 5, 8, 6, 11];$
 $min_V = [, 1, , 1, 6, 4, 7];$
 $min_S = 2; min_S_weight = 7;$
 $E_T = \{(1, 4), (4, 6), (1, 2)\}; v = 2;$

Επανάληψη 4

$V_T = \{1, 4, 6, 2\}; S = \{3, 5, 7\};$
 $min_V_weight = [\infty, 7, 8, 5, 7, 6, 11];$
 $min_V = [, 1, 2, 1, 2, 4, 7];$
 $min_S = 5; min_S_weight = 7;$
 $E_T = \{(1, 4), (4, 6), (1, 2), (2, 5)\}; v = 5;$

Επανάληψη 5

$V_T = \{1, 4, 6, 2, 5\}; S = \{3, 7\};$
 $min_V_weight = [\infty, 7, 5, 5, 7, 6, 9];$
 $min_V = [, 1, 5, 1, 2, 4, 5];$
 $min_S = 3; min_S_weight = 5;$
 $E_T = \{(1, 4), (4, 6), (1, 2), (2, 5), (5, 3)\}; v = 3;$

Επανάληψη 6

$V_T = \{1, 4, 6, 2, 5, 3\}; S = \{7\};$
 $min_V_weight = [\infty, 7, 5, 5, 7, 6, 9];$
 $min_V = [, 1, 5, 1, 2, 4, 5];$
 $min_S = 5; min_S_weight = 9;$
 $E_T = \{(1, 4), (4, 6), (1, 2), (2, 5), (5, 3), (5, 7)\}; v = 7;$

Θεωρία πάνω στο Ελάχιστο Συνδετικό Δένδρο

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.

1. Το δένδρο των ελάχιστων διαδρομών από οποιαδήποτε κορυφή s είναι συνδετικό δένδρο του γραφήματος.
2. Αν προσθέσουμε μία σταθερά b στο βάρος κάθε ακμής το ΕΣΔ αλλάζει.
3. Το ΕΣΔ και το δένδρο των ελάχιστων διαδρομών ανεξάρτητα της αφετηριακής κορυφής s ταυτίζονται σε κάθε γράφημα G .
4. Υπάρχει για κάθε γράφημα G μία αφετηριακή κορυφή s για την οποία το δένδρο των ελαχίστων διαδρομών ταυτίζεται με το ΕΣΔ.

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

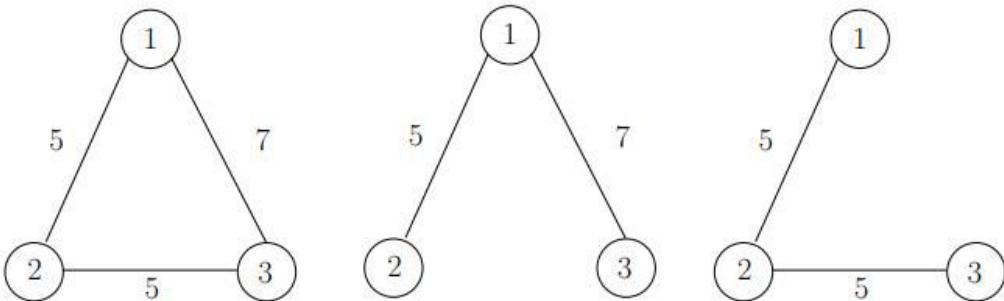
1. Σωστό

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

2. Λάθος

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

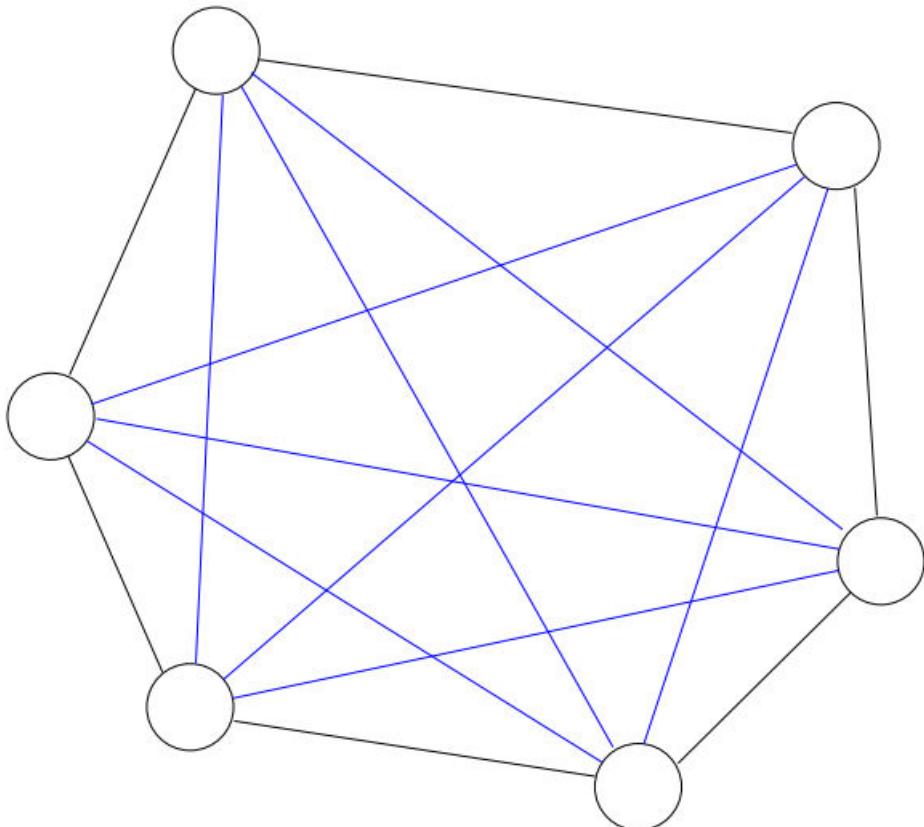
3. Λάθος. Στο γράφημα του Σχήματος 13 (αριστερή εικόνα) το δένδρο των ελαχίστων διαδρομών με αφετηριακή κορυφή την 1 απεικονίζεται στην κεντρική εικόνα ενώ το ΕΣΔ στη δεξιά.



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 ελαχίστων 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 μπλέ κορυφής θα έχουμε κόστος 2.



Σχήμα 14: Δένδρο Συντ. Διαδρομών, Ελάχιστο Συνδετικό Δένδρο

06_ΔΥΝΑΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ / 09_DynamicExer

Η μεγαλύτερη αύξουσα υπακολουθία (maxsubseq)

Εκφώνηση

Άσκηση 1 Δίνεται μία σειρά αριθμών a_1, \dots, a_n . Μία 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

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 $a_i < a_j$.

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

2. Καθορισμός Επιλογών

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

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

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

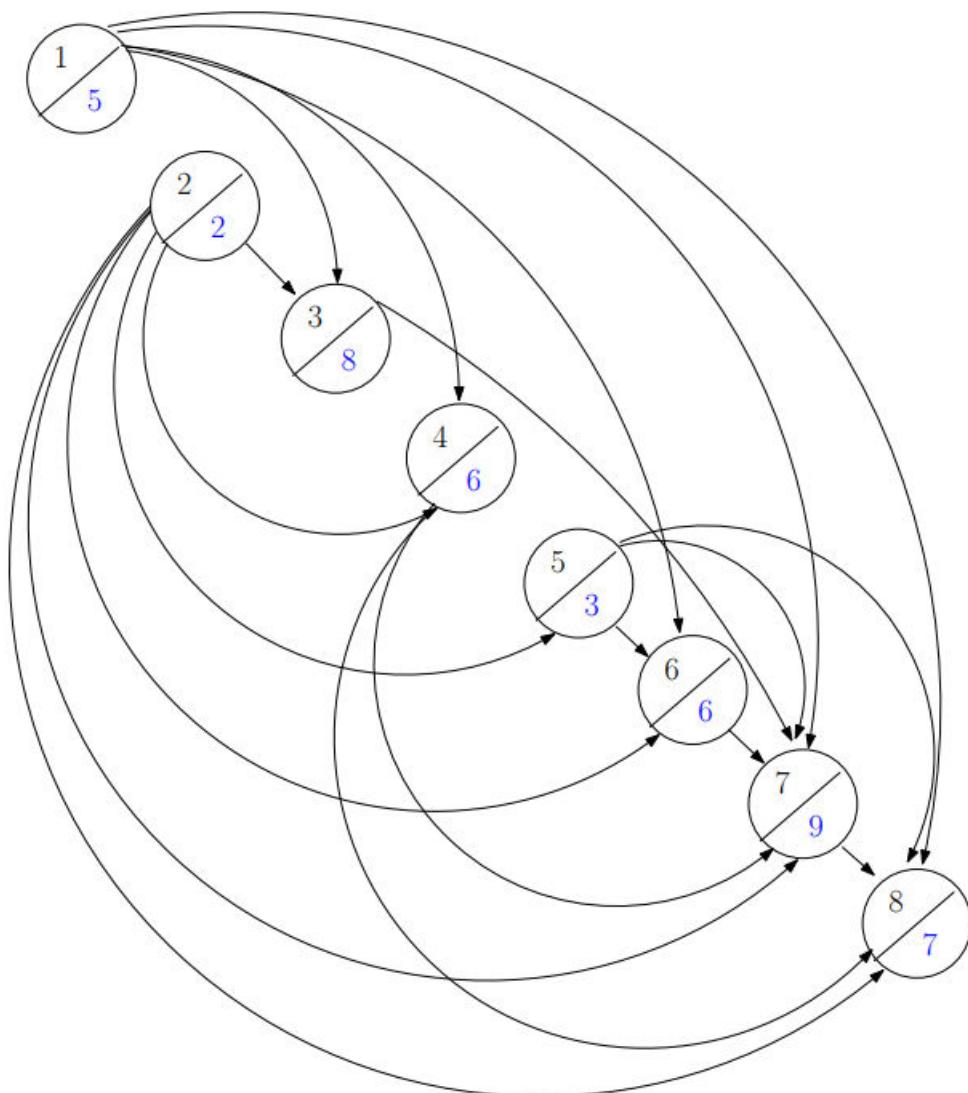
όπου $N^-(j) : \{i : i \in V(G), (i,j) \in E(G)\}$.

4. Τοπολογική Ταξινόμηση Υποπροβλημάτων

Παράδειγμα 1 Για τη σειρά

5 2 8 6 3 6 9 7

το αντίστοιχο γράφημα απεικονίζεται στο Σχήμα 1.



Σχήμα 1: Κατευθυνόμενο γράφημα διάταξης G

Στο παραπάνω παράδειγμα,

$$\begin{aligned}L(1) &= 1, \\L(2) &= 1, \\L(3) &= 1 + \max\{L(1), L(2)\} = 2 \\L(4) &= 1 + \max\{L(1), L(2)\} = 1 + \max\{1, 1\} = 2 \\L(5) &= 1 + \max\{L(2)\} = 1 + \max\{1\} = 2 \\L(6) &= 1 + \max\{L(5), L(2), L(1)\} = 1 + \max\{2, 1, 1\} = 3 \\L(7) &= 1 + \max\{L(6), L(5), L(4), L(3), L(2), L(1)\} = 1 + \max\{3, 2, 2, 2, 1, 1\} = 4 \\L(8) &= 1 + \max\{L(6), L(5), L(4), L(2), L(1)\} = 1 + \max\{3, 2, 2, 1, 1\} = 4\end{aligned}$$

5. Επίλυση Αρχικού Προβλήματος

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

| | | | |
|---|---|---|---|
| 2 | 3 | 6 | 7 |
| 2 | 3 | 6 | 9 |

Ιδέα Αλγορίθμου

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 Αλγορίθμου

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

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

Το μεγαλύτερο άθροισμα στοιχείων ενός πίνακα (maxseqsum)

Εκφώνηση

Άσκηση 2 Θεωρούμε ότι ο πίνακας A περιέχει ακέραιους στις θέσεις από 1 ως n . Να βρεθεί η υποπεριοχή του πίνακα με το μεγαλύτερο άθροισμα στοιχείων.

1. Ορισμός Υποπροβλημάτων

Έστω S_i το μεγαλύτερο άθροισμα από τους συνεχόμενους αριθμούς της σειράς που τελειώνουν στη θέση i .

2. Καθορισμός Επιλογών

Τότε ισχύει ότι $S_{i+1} = S_i + A_{i+1}$ αν $S_i + A_{i+1} > A_{i+1}$. Στην αντίθετη περίπτωση, $S_{i+1} = A_{i+1}$.

3. Ορισμός Αναδρομικής Σχέσης

Οπότε η σχέση που συνδέει τα αθροίσματα S είναι

$$S_{i+1} = \begin{cases} S_i + A_{i+1} & S_i > 0, \\ A_{i+1} & S_i \leq 0. \end{cases}$$

Η παραπάνω σχέση γράφεται πιο συνοπτικά σαν

$$S_{i+1} = \max\{A_{i+1}, S_i + A_{i+1}\}, \quad (3)$$

με $S_1 = A_1$.

4. Τοπολογική Ταξινόμηση Υποπροβλημάτων

Λύση Σαν παράδειγμα θεωρούμε τη σειρά

$$-1 \quad -3 \quad 6 \quad -1 \quad -1 \quad -1 \quad 5 \quad -4$$

5. Επίλυση Αρχικού Προβλήματος

Προφανώς, η ζητούμενη λύση δίνεται από τη σχέση

$$MXSUM = \max\{S_i : i \in \{1, \dots, n\}\}$$

Ιδέα Αλγορίθμου

Ο Αλγόριθμος 2 επιλύει το συγκεκριμένο πρόβλημα. Υλοποιείται σαν συνάρτηση η οποία επιστρέφει την τιμή του μεγαλύτερου αθροίσματος της υπακολουθίας των συνεχών όρων.

Αλγόριθμος

Algorithm 2 Υπακολουθία συνεχών όρων μεγαλύτερου αθροίσματος.

```
1: int maxseqsum(int n, int A[], int previous[])
2: S[1] ← A[1];
3: for i ← 1; i ≤ n – 1; i + + do
4:   S[i + 1] ← max{A[i + 1], S[i] + A[i + 1]};
5: end for
6: max_sum ← –∞;
7: for i ← 1; i ≤ n; i + + do
8:   max_sum ← max{S[i]};
9: end for
10: return max_sum;
```

Πολυπλοκότητα Αλγορίθμου

Ο αριθμός των στοιχειωδών πράξεων είναι τάξης $\Theta(n)$.

Ο μικρότερος αριθμός χαρτονομισμάτων για ένα ποσό (MinNotes)

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 επλάχιστο 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 ανα-

δρομική 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

Παράδειγμα 2 Θεωρούμε νομίσματα του ευρώ $A[1] = 1, A[2] = 2, A[3] = 5$, κλπ.

Έστω $X = 8$ ευρώ. Για να βρεθεί ο ελάχιστος αριθμός χαρτονομισμάτων αντίστοιχης αξίας εκτιμούμε την 4 ξεκινώντας από $x = 2$ έως $x = 8$. Έχουμε,

$x = 1 :$

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

$x = 2 :$

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

$x = 3 :$

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

$x = 4 :$

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

$x = 5 :$

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

$x = 6 :$

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

$x = 7 :$

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

$x = 8 :$

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

5. Επίλυση Αρχικού Προβλήματος

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

Ιδέα Αλγορίθμου

Ο Αλγόριθμος 3 υλοποιεί την παραπάνω διαδικασία.

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
```

Πολυπλοκότητα Αλγορίθμου

Η πολυπλοκότητα του αλγόριθμου

είναι $O(nX)$.

Η διορθωτική απόσταση ανάμεσα σε δύο συμβολοσειρές (EditDistance)

Εκφώνηση

Άσκηση 4 Έστω $x = x_1x_2 \cdots x_n$, $y = y_1y_2 \cdots y_m$ δύο συμβολοσειρές με τα σύμβολα $x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_m$ να ανήκουν σε κάποιο αλφάριθμο Γ . Η διορθωτική απόσταση ανάμεσα στις δύο συμβολοσειρές είναι ο ελάχιστος αριθμός αλλαγών - εισαγωγών, διαγραφών και αντικαταστάσεων - χαρακτήρων οι οποίες είναι απαραίτητες για το μετασχηματισμό της πρώτης συμβολοσειράς στη δεύτερη. Να διατυπώσετε αλγόριθμο οποίος υπολογίζει την παράμετρο αυτή και να παράγει τις αλλαγές που αντιστοιχούν.

1. Ορισμός Υποπροβλημάτων

Λύση Σαν παράδειγμα μπορούμε να χρησιμοποιήσουμε τις συμβολοσειρές του Πίνακα 1 Γενικότερα στόχος μας είναι η εύρεση της διορθωτικής απόστασης ανάμεσα στις συμβολοσειρές $x[1 \dots n]$ και $y[1 \dots m]$. Θεωρούμε ένα πρόδεμα της πρώτης συμβολοσειράς, έστω $x[1 \dots i]$, και ένα της δεύτερης έστω $y[1 \dots j]$, όπου $i \in \{1, \dots, n\}$, $j \in \{1, \dots, m\}$. Ονομάζουμε αυτό το πρόβλημα ως $E[i, j]$. Στο παράδειγμα μας, για $i = 3, j = 4$, τα δύο προθέματα εμφανίζονται στον Πίνακα 2.

| | | | | | |
|---|---|---|---|---|---|
| P | A | I | P | N | Ω |
| P | E | P | N | Ω | |

Πίνακας 1: Παράδειγμα διορθωτικής απόστασης

| | | | | | |
|------------------|---|---|---|---|--|
| $x[1 \dots 3] =$ | P | A | I | | |
| $y[1 \dots 4] =$ | P | E | P | N | |

Πίνακας 2: Παράδειγμα προθεμάτων

2. Καθορισμός Επιλογών

Προσπαθούμε να εκφράσουμε το $E(i, j)$ σε συνάρτηση με μικρότερα υποπροβλήματα. Έχουμε τρεις περιπτώσεις σε σχέση με τη στοίχιση των $x[i], y[j]$ όπως φαίνεται στον Πίνακα 3. Για το παράδειγμα μας οι αντίστοιχες περιπτώσεις απεικονίζονται στον Πίνακα 4.

Η πρώτη περίπτωση επιφέρει κόστος 1 για τη συγκεκριμένη στήλη και μετά μένει να στοιχίσουμε την υποσυμβολοσειρά $x[1 \dots i-1]$ με την $y[1 \dots j]$. Αυτό όμως είναι το υποπρόβλημα $E[i-1, j]$. Αντίστοιχα, η δεύτερη περίπτωση επιφέρει και αυτή κόστος 1 μετά μένει να στοιχίσουμε την υποσυμβολοσειρά $x[1 \dots i]$ με την $y[1 \dots j-1]$. Αυτό όμως είναι το υποπρόβλημα $E[i, j-1]$. Τέλος η τρίτη περίπτωση έχει και αυτή κόστος 1 αν $x[i] \neq y[j]$ και 0 αν $x[i] = y[j]$. Μετά μένει να επιλύσουμε το πρόβλημα $E[i-1, j-1]$.

| | | | | |
|------------------|---|------------------|---|---------------------|
| $\frac{x[i]}{-}$ | ή | $\frac{-}{y[j]}$ | ή | $\frac{x[i]}{y[j]}$ |
|------------------|---|------------------|---|---------------------|

Πίνακας 3: Τρεις περιπτώσεις

| | | | | |
|---------------|---|---------------|---|---------------|
| $\frac{I}{-}$ | ή | $\frac{-}{N}$ | ή | $\frac{I}{N}$ |
|---------------|---|---------------|---|---------------|

Πίνακας 4: Τρεις περιπτώσεις (Παράδειγμα)

3. Ορισμός Αναδρομικής Σχέσης

Τα παραπάνω συνεπάγονται την αναδρομική σχέση

$$E[i,j] = \min\{1 + E[i-1,j], 1 + E[i,j-1], d_{ij} + E[i-1,j-1]\},$$

όπου $d_{ij} = 1$ αν $x[i] \neq y[j]$ και 0 αν $x[i] = y[j]$. Οριακές συνθήκες είναι

$$E[0,0] = 0, E[i,0] = i, E[0,j] = j.$$

4. Τοπολογική Ταξινόμηση Υποπροβλημάτων

| | Π | E | P | N | Ω |
|----------|-------|-----|-----|-----|----------|
| | 0 | 1 | 2 | 3 | 4 |
| Π | 1 | 0 | 1 | 2 | 3 |
| A | 2 | 1 | 1 | 2 | 3 |
| I | 3 | 2 | 2 | 2 | 4 |
| P | 4 | 3 | 3 | 2 | 3 |
| N | 5 | 4 | 4 | 3 | 2 |
| Ω | 6 | 5 | 5 | 4 | 3 |
| | (a) | | | | |

| | Π | E | P | N | Ω |
|----------|-------|-------|-------|-------|----------|
| | (0,0) | (0,0) | (0,0) | (0,0) | (0,0) |
| Π | (0,0) | (0,0) | (1,1) | (1,2) | (1,3) |
| A | (0,0) | (1,1) | (1,1) | (2,2) | (2,3) |
| I | (0,0) | (2,1) | (2,2) | (2,2) | (3,3) |
| P | (0,0) | (3,1) | (3,2) | (3,2) | (4,3) |
| N | (0,0) | (4,1) | (4,2) | (4,3) | (4,3) |
| Ω | (0,0) | (5,1) | (5,2) | (5,3) | (5,3) |
| | (b) | | | | |

Πίνακας 5: Επίλυση Παραδείγματος

5. Επίλυση Αρχικού Προβλήματος

Για το παράδειγμά μας, ο Πίνακας 5 απεικονίζει τον E καθώς και πως έχει γίνει το ταίριασμα. Συγκεκριμένα ξεκινώντας από την κάτω δεξιά γωνία του Πίνακα 5β (ταίριασμα ‘Ω’ με ‘Ω’) βλέπουμε ότι το προηγούμενο ταίριασμα είναι το στοιχείο της πέμπτης γραμμής με της τέταρτης στήλης (ταίριασμα ‘Ν’ με ‘Ν’). Το προηγούμενο ταίριασμα γίνεται ανάμεσα στο στοιχείο της τέταρτης γραμμής και της τρίτης στήλης (ταίριασμα ‘Ρ’ με ‘Ρ’). Ακολούθως το στοιχείο (4,3) του Πίνακα 5β παραπέμπει στο στοιχείο (3,2) και αυτό στο στοιχείο (2,2). Αυτό σημαίνει ότι ο χαρακτήρας ‘Τ’ αντιστοιχείται σε μία κενή θέση που παρεισφρύει στη συμβολοσειρά “ΠΕΡΝΩ” ανάμεσα στα γράμματα ‘Ε’ και ‘Ρ’. Στη συνέχεια το ‘Α’ αντιστοιχείται στο ‘Ε’ και τέλος το ‘Π’ στο ‘Ρ’. Η διορθωτική απόσταση είναι 2 όπως φαίνεται και από το κάτω-δεξιά στοιχείο του Πίνακα 5α. Στον Πίνακα 6β απεικονίζεται πιο παραστατικά η συγκεκριμένη διόρθωση.

| | Π | E | P | N | Ω |
|----------|-------|-----|-----|-----|----------|
| | 0 | 1 | 2 | 3 | 4 |
| Π | 1 | 0 | 1 | 2 | 3 |
| A | 2 | 1 | 1 | 2 | 3 |
| I | 3 | 2 | 2 | 2 | 4 |
| P | 4 | 3 | 3 | 2 | 3 |
| N | 5 | 4 | 4 | 3 | 2 |
| Ω | 6 | 5 | 5 | 4 | 3 |
| | (a) | | | | |

| | Π | E | P | N | Ω |
|----------|-------|-----|-----|-----|----------|
| Π | | • | | | |
| A | | | • | | |
| I | | | | • | |
| P | | | | | • |
| N | | | | | • |
| Ω | | | | | • |
| | (b) | | | | |

Πίνακας 6: Επίλυση Παραδείγματος

Ιδέα Αλγορίθμου

Ο Αλγόριθμος 4 υλοποιεί την διαδικασία.

Αλγόριθμος

Algorithm 4 Υπολογισμός Διορθωτικής Απόστασης

```
1: void EditDistance(int n, int x[], int m, int y[])
2: for i ← 0; i ≤ n; i + + do
3:   E[i][0] ← i;
4:   keep_1[i][0] ← 0;
5:   keep_2[i][0] ← 0;
6: end for
7: for j ← 0; j ≤ m; j + + do
8:   E[0][j] ← j;
9:   keep_1[0][j] ← 0;
10:  keep_2[0][j] ← 0;
11: end for
12: for i ← 0; i ≤ n; i + + do
13:   for j ← 0; j ≤ m; j + + do
14:     if x[i] = y[j] then
15:       d ← 0;
16:     else
17:       d ← 1;
18:     end if
19:     a ← E[i - 1][j] + 1; b ← E[i][j - 1] + 1; c ← d + E[i - 1][j - 1];
20:     if a > b then
21:       if b > c then
22:         E[i][j] ← c; keep_1[i][j] ← i - 1; keep_2[i][j] ← j - 1;
23:       else
24:         E[i][j] ← b; keep_1[i][j] ← i; keep_2[i][j] ← j - 1;
25:       end if
26:     else
27:       if a > c then
28:         E[i][j] ← c; keep_1[i][j] ← i - 1; keep_2[i][j] ← j - 1;
29:       else
30:         E[i][j] ← a; keep_1[i][j] ← i - 1; keep_2[i][j] ← j;
31:       end if
32:     end if
33:   end for
34: end for
```

Πολυπλοκότητα Αλγορίθμου

Ο αριθμός των στοιχειωδών πράξεων που εκτελεί ο αλγόριθμος είναι τάξης $\Theta(n \cdot m)$.

Μεγιστοποίηση κέρδους μιας εταιρίας εστίασης από τις τοποθεσίες εγκατάστασης εστιατορίων (locations)

Εκφώνηση

Άσκηση 5 Μια εταιρεία εστίασης μπορεί να ανοίξει μία σειρά από εστιατόρια κατά μήκος μίας διαδρομής. Οι πιθανές τοποθεσίες σχηματίζουν μία ευδειά 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 . Το ζητούμενο είναι το $P(n)$.

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

5. Επίλυση Αρχικού Προβλήματος

Ιδέα Αλγορίθμου

Ο Αλγόριθμος 5 υπολογίζει τα $P(j)$ με βάση τον προηγούμενο τύπο.

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{pj + P[previous[j]], P[j - 1]};
11: end for$$ 
```

Πολυπλοκότητα Αλγορίθμου

Παρατηρήστε ότι πριν τον υπολογισμό αυτό, γίνεται υπολογισμός των j' (πίνακας *prev[]* - γραμμές 2-8). 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

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

Το πρόβλημα του διαρρήκτη (Knapsack integer problem)

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 wt_i και αξία v_i .

| 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.

- 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

Δύση Υπάρχουν δύο περιπτώσεις: να μπορεί ο κλέφτης να προσθέσει στο σακίδιο του όσα αντικείμενα από το ίδιο είδος θέλει (knapsack integer problem) ή να προσθέσει το πολύ μόνο ένα αντικείμενο από το κάθε είδος (knapsack 0-1 problem). Η δεύτερη περίπτωση είναι πιο γενική και προφανώς η πρώτη ανάγεται σε αυτή

Αν το $K(w)$ έχει σχηματιστεί χωρίς να περιέχεται το αντικείμενο i , μπορούμε να το προσθέσουμε αν $wt_i \leq w$. Τότε η μέγιστη συνολική αξία γίνεται $K(w - wt_i) + v_i$. Σε κάθε βήμα λοιπόν επιλέγουμε να προσθέσουμε το αντικείμενο i το οποίο μεγιστοποιεί την παράσταση αυτή.

3. Ορισμός Αναδρομικής Σχέσης

$$K(w) = \begin{cases} \max\{K(w - wt_i) + v_i : i \in \{1, \dots, n\}, wt_i \leq w\}, & w \in \{1, \dots, W\}, \\ 0, & w = 0. \end{cases} \quad (5)$$

4. Τοπολογική Ταξινόμηση Υποπροβλημάτων

$$w = 0 : K(0) = 0.$$

$$w = 1 : K(1) = 0, \text{ αφού κανένα αντικείμενο δεν έχει βάρος 1.}$$

$$w = 2 :$$

$$K(2) = \max\{K(2 - wt_4) + v_4\} = 1.$$

$$w = 3 :$$

$$K(3) = \max\{K(3 - wt_2) + v_2, K(3 - wt_4) + v_4\} = \max\{0 + 14, 1 + 1\} = 14.$$

$w = 4$:

$$\begin{aligned} K(4) &= \max\{K(4 - wt_2) + v_2, K(4 - wt_3) + v_3, K(4 - wt_4) + v_4\} \\ &= \max\{K(4 - 3) + 14, K(4 - 4) + 16, K(4 - 2) + 1\} \\ &= \max\{0 + 14, 0 + 16, 1 + 1\} = 16. \end{aligned}$$

$w = 5$:

$$\begin{aligned} K(5) &= \max\{K(5 - wt_2) + v_2, K(5 - wt_3) + v_3, K(5 - wt_4) + v_4\} \\ &= \max\{K(5 - 3) + 14, K(5 - 4) + 16, K(5 - 2) + 1\} \\ &= \max\{K(2) + 14, K(1) + 16, K(3) + 1\} \\ &= \max\{1 + 14, 0 + 16, 14 + 1\} = 16. \end{aligned}$$

$w = 6$:

$$\begin{aligned} K(6) &= \max\{K(6 - wt_1) + v_6, K(6 - wt_2) + v_2, K(6 - wt_3) + v_3, K(6 - wt_4) + v_4\} \\ &= \max\{K(6 - 6) + 30, K(6 - 3) + 14, K(6 - 4) + 16, K(6 - 2) + 1\} \\ &= \max\{K(0) + 30, K(3) + 14, K(2) + 16, K(4) + 1\} \\ &= \max\{0 + 30, 14 + 14, 1 + 16, 16 + 1\} = 30. \end{aligned}$$

$w = 7$:

$$\begin{aligned} K(7) &= \max\{K(7 - wt_1) + v_6, K(7 - wt_2) + v_2, K(7 - wt_3) + v_3, K(7 - wt_4) + v_4\} \\ &= \max\{K(7 - 6) + 30, K(7 - 3) + 14, K(7 - 4) + 16, K(7 - 2) + 1\} \\ &= \max\{K(1) + 30, K(4) + 14, K(3) + 16, K(5) + 1\} \\ &= \max\{0 + 30, 16 + 14, 14 + 16, 16 + 1\} = 30. \end{aligned}$$

$w = 8$:

$$\begin{aligned} K(8) &= \max\{K(8 - wt_1) + v_6, K(8 - wt_2) + v_2, K(8 - wt_3) + v_3, K(8 - wt_4) + v_4\} \\ &= \max\{K(8 - 6) + 30, K(8 - 3) + 14, K(8 - 4) + 16, K(8 - 2) + 1\} \\ &= \max\{K(2) + 30, K(5) + 14, K(4) + 16, K(6) + 1\} \\ &= \max\{1 + 30, 16 + 14, 16 + 16, 30 + 1\} = 32. \end{aligned}$$

5. Επίλυση Αρχικού Προβλήματος

Επομένως η μέγιστη αξία των αντικειμένων που μπορεί να μεταφέρει ο κλέφτης είναι $K(8) = 32$ και επιτυγχάνεται με δύο αντικείμενα τύπου 3.

Ιδέα Αλγορίθμου

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 Αλγορίθμου

Ο Αλγόριθμος 6 είναι τάξης $O(n \cdot W)$ (γραμμές 9-19).

Το πρόβλημα του διαρρήκτη (Knapsack 0-1 problem)

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 wt_i και αξία v_i .

| | | | | |
|--------|----|----|----|---|
| 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.

- Στην περίπτωση αυτή τροποποιούμε τον ορισμό της (5). Προσθέτουμε μία επιπλέον παράμετρο στη συνάρτηση K . Συγκεκριμένα, 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

Λύση Υπάρχουν δύο περιπτώσεις: να μπορεί ο κλέφτης να προσθέσει στο σακίδιο του όσα αντικείμενα από το ίδιο είδος θέλει (knapsack integer problem) ή να προσθέσει το πολύ μόνο ένα αντικείμενο από το κάθε είδος (knapsack 0-1 problem). Η δεύτερη περίπτωση είναι πιο γενική και προφανώς η πρώτη ανάγεται σε αυτή

Το υποπρόβλημα $K(w, i)$ εκφράζεται σαν συνάρτηση μικρότερων υποπροβλημάτων ως εξής. Η τιμή $K(w, i)$ είτε επιτυγχάνεται με τη χρήση του αντικειμένου i είτε όχι. Στην πρώτη περίπτωση έχουμε $K(w, i) = K(w - wt_i, i - 1) + v_i$, ενώ στη δεύτερη $K(w, i) = K(w, i - 1)$.

3. Ορισμός Αναδρομικής Σχέσης

Η ανα

δοριμική συνάρτηση, για $w \in \{1, \dots, W\}$, $i \in \{1, \dots, n\}$, έχει τη μορφή

$$K(w, i) = \begin{cases} 0, & w = 0 \text{ ή } i = 0, \\ K(w, i - 1), & wt_i > w, \\ \max\{K(w - wt_i, i - 1) + v_i, K(w, i - 1)\} & wt_i \leq w. \end{cases}$$

4. Τοπολογική Ταξινόμηση Υποπροβλημάτων

| w | i | 1 | 2 | 3 | 4 |
|-----|-----|--|--|---|---|
| 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

5. Επίλυση Αρχικού Προβλήματος

Προφανώς η βέλτιστη λύση δίνεται από το $K(W, n)$. 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.

Ιδέα Αλγορίθμου

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  $i \leftarrow 0; i \leq n; i++$  do
3:    $x[i] \leftarrow 0;$ 
4:   for  $w \leftarrow 0; w \leq W; w++$  do
5:      $K[w][i] \leftarrow 0; xweight[w][i] \leftarrow 0;$ 
6:   end for
7: end for
8: for  $w \leftarrow 1; w \leq W; w++$  do
9:   for  $i \leftarrow 1; i \leq n; i++$  do
10:    if  $wt[i] > w$  then
11:       $K[w][i] \leftarrow K[w][i - 1]; xweight[w][i] \leftarrow 0;$ 
12:    else
13:      if  $K[w][i - 1] > K[w - wt[i]][i - 1] + v[i]$  then
14:         $K[w][i] \leftarrow K[w][i - 1]; xweight[w][i] \leftarrow 0;$ 
15:      else
16:         $K[w][i] \leftarrow K[w - wt[i]][i - 1] + v[i]; xweight[w][i] \leftarrow 1;$ 
17:      end if
18:    end if
19:  end for
20: end for
21:  $w = W$ 
22: for  $i = n; i \geq 1; i--$  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 Αλγορίθμου

Ο Αλγόριθμος 7 είναι τάξης $O(n \cdot W)$ (γραμμές 8-20).

07_ΑΠΛΗΣΤΟΙ ΑΛΓΟΡΙΘΜΟΙ / 11_greedyexer

Ο μικρότερος αριθμός κλειστών διαστημάτων μοναδιαίας ακτίνας σ' ένα σύνολο

Εκφώνηση

Άσκηση 1 Θεωρούμε ένα συνολό 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.

Ιδέα Αλγορίθμου

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:    $num\_intervs ++$ ;  
6:   while  $V[i] \leq b$  do  
7:      $i ++$ ;  
8:   end while  
9: end while  
10: Print  $num\_intervs$ 
```

Πολυπλοκότητα Αλγορίθμου

Ο Αλγόριθμος 1 υλοποιεί σε $O(n)$ βήματα την ιδέα.

Άπληστη Επιλογή

(Απληστη Επιλογή) Έστω S μία βέλτιστη λύση με πρώτο (αριστερότερο) διάστημα $[s, s + 1]$. Προφανώς το διάστημα αυτό δεν μπορεί να είναι δεξιότερα από το διάστημα $[v_1, v_1 + 1]$ 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 υπολοίπων 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' .

Το οδικό ταξίδι με τις λιγότερες στάσεις για ανεφοδιασμό

Εκφώνηση

Άσκηση 2 Έστω ότι θέλουμε να ταξιδεύσουμε από την Αθήνα στη Θεσσαλονίκη με αυτοκίνητο. Η χωρητικότητα του ρεζερβουάρ μας επιτρέπει να καλύψουμε απόσταση t 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 $d_{i+1} - d_i \leq t$, 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.

Ιδέα Αλγορίθμου

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 $d_1 \leq t$ και το σημείο $n+1$ αντιπροσωπεύει τον προορισμό- δηλαδή την Θεσσαλονίκη- και ισχύει ότι $d_{n+1} - d_n \leq t$).

Αλγόριθμος

Algorithm 2 Άπληστος 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 Αλγορίθμου

Η πολυπλοκότητα του αλγόριθμου 2 είναι $\Theta(n)$. Η απόδειξη της ορθότητας ακολουθεί.

Άπληστη 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 $d_{s_2} - d_v < d_{s_2} - d_{s_1} \leq t$. 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.

Η μέγιστη συνολική θρεπτική αξία τροφίμων που χωράει σ' ένα σακίδιο

Εκφώνηση

Άσκηση 3 Ένας ορειβάτης έχει ένα σακίδιο το οποίο μπορεί να χωρέσει W κιλά. Ο ορειβάτης πρέπει να επιλέξει ανάμεσα σε n συσκευασίες να πάρει μαζί του, καθεμία από τις οποίες περιέχει ένα διαφορετικό τρόφιμο. Η κάθε συσκευασία $i \in \{1, \dots, n\}$ έχει βάρος w_i και θρεπτική αξία v_i θερμίδες. Αυτή να πάρει ολόκληρη τη συσκευασία μπορεί να πάρει ένα κομμάτι (ποσοστό) από αυτή. Για παράδειγμα αν πάρει 90% της συσκευασίας i θα προσθέσει στο σακίδιο βάρος $0,9w_i$ και θα έχει θρεπτική αξία $0,9v_i$. Με ποιο τρόπο ο ορειβάτης μπορεί να μεγιστοποιήσει τη συνολική θρεπτική αξία που θα πάρει μαζί του.

Ιδέα Αλγορίθμου

Λύση Κατατάσουμε τις συσκευασίες σε φθίνουσα σειρά του λόγου της θρεπτικής αξίας προς το βάρος. Έστω λοιπόν

$$\frac{v_1}{w_1} > \frac{v_2}{w_2} > \dots > \frac{v_n}{w_n}. \quad (1)$$

Ισχυριζόμαστε ότι η θρεπτική αξία μεγιστοποιείται αν πάρει όσο περισσότερο μπορεί από τη συσκευασία 1 μετά από την 2 κ.ο.κ. Δηλαδή η βέλτιστη επιλογή γίνεται με άπληστο τρόπο με τη σειρά που υπαγορεύει η έκφραση (1). Ο Αλγόριθμος 3

Αλγόριθμος

Algorithm 3 Άπληστος αλγόριθμος για το κλασματικό πρόβλημα του σακκιδίου - Άσκηση 3

Require: βάρη $w[1, \dots, n]$, αξίες $v[1, \dots, n]$, τέτοιες ώστε $\frac{v[1]}{w[1]} \geq \frac{v[2]}{w[2]} \geq \dots \geq \frac{v[n]}{w[n]}$

- 1: $weight \leftarrow 0$;
- 2: $totval \leftarrow 0$;
- 3: $i \leftarrow 0$;
- 4: **while** $weight < W$ **do**
- 5: $i \leftarrow i + 1$;
- 6: $ratio \leftarrow \frac{W-weight}{w[i]}$;
- 7: **if** $ratio \geq 1$ **then**
- 8: $ratio \leftarrow 1$;
- 9: **end if**
- 10: $x[i] \leftarrow ratio$;
- 11: $weight \leftarrow weight + ratio \cdot w[i]$;
- 12: $totval \leftarrow totval + ratio \cdot v[i]$;
- 13: **end while**
- 14: Print x

Πολυπλοκότητα Αλγορίθμου

Η πολυπλοκότητα του αλγόριθμου 2 είναι $O(n)$.

Απληστη Επιλογή

(Απληστη Επιλογή) Έστω $X = (x_1, \dots, x_n)$ μία βέλτιστη λύση όπου η συσκευασία 1 δεν περιλαμβάνεται σε ποσοστό 100% (δηλαδή $x_1 < 1$). Αν το σακίδιο δεν έχει γεμίσει τότε μπορούμε να αυξήσουμε τη θρεπτική αξία παίρνοντας επιπλέον ποσοστό από τη συσκευασία αυτή μέχρι να φτάσουμε στη χωρητικότητα W (αντίφαση αφού η λύση X είναι βέλτιστη.) Αν το σακίδιο έχει γεμίσει τότε θα περιλαμβάνει κάποια συσκευασία k για την οποία ισχύει ότι

$$\frac{v_1}{w_1} > \frac{v_k}{w_k}$$

Προφανώς υπάρχει ένα ποσοστό της συσκευασίας 1 που δεν έχει χρησιμοποιηθεί. Μπορούμε να μειώσουμε το βάρος από τη συσκευασίας k κατά μία μικρή ποσότητα $\epsilon > 0$ (να τη βγάλουμε από το σακίδιο) και να αυξήσουμε κατά την ίδια ποσότητα το βάρος της 1 (να την προσθέσουμε στο σακίδιο). Στην λύση X το βάρος της συσκευασίας k είναι $x_k \cdot w_k$. Άρα αν αφαιρέσουμε μία ποσότητα (εκφρασμένη σε βάρος) ίση με ένα ποσοστό p_0 του γινομένου αυτού, έχουμε $\epsilon = p_0 \cdot x_k \cdot w_k$. Επειδή θα προσθέσουμε την ποσότητα αυτή στη συσκευασία 1 θα πρέπει

$$(1 - x_1)w_1 > \epsilon = p_0 \cdot x_k \cdot w_k$$

αφού $(1 - x_1)w_1$ είναι η επιπλέον ποσότητα σε βάρος από τη συσκευασία 1 που δεν έχει χρησιμοποιηθεί.

Επειδή γενικότερα το κλάσμα $\frac{v_i}{w_i}$ είναι η θρεπτική αξία που έχει μία μονάδα βάρους (π.χ. κιλό) από τη συσκευασία i , η διαφορά στη θρεπτική αξία από την αλλαγή που κάναμε είναι θετική, δηλαδή ίση με

$$\epsilon \cdot \left(\frac{v_1}{w_1} - \frac{v_k}{w_k} \right) > 0$$

(αντίφαση στη βελτιστότητα της X).

Βέλτιστη Υποδομή

(Βέλτιστη Υποδομή) Αν έχει χρησιμοποιηθεί όλη η συσκευασία 1, το υποπρόβλημα P' που πρέπει να επιλυθεί αφορά βάρος σακιδίου $W' = W - w_1$ για τις υπόλοιπες συσκευασίες. Η θρεπτική αξία της βέλτιστης λύσης σε αυτό αθροιζόμενη με την v_1 δίνει τη βέλτιστη θρεπτική αξία στο αρχικό πρόβλημα.

Προγραμματισμός εργασιών για μεγιστοποίηση κέρδους

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.

Ιδέα Αλγορίθμου

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

Αλγορίτημ 4 Απληστος 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: $t \leftarrow t - 1;$
- 7: **end while**

Πολυπλοκότητα Αλγορίθμου

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' .

08_ΒΑΣΙΚΕΣ ΚΛΑΣΕΙΣ ΠΡΟΒΛΗΜΑΤΩΝ P, NP / 16_npsexe

kIKAN

Πρόβλημα

Λύση Η δομή kIKAN υπονοεί το πρόβλημα IKAN όταν υπάρχουν ακριβώς k όροι σε κάθε πρόταση.

Απόδειξη kIKANεNP

Επομένως η απόδειξη ότι kIKAN ανήκει στην τάξη NP είναι ακριβώς ανάλογη με την αντίστοιχη απόδειξη για 3IKAN και παραλείπεται.

Απόδειξη k IKAN \in NP-hard

τα i . Στη συνέχεια θα δείξουμε $\text{3IKAN} \leq_P k\text{IKAN}$. Θεωρούμε ένα στιγμιότυπο 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 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 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 $T(z_1) = T(z_2) = \dots = T(z_{k-3}) = \text{False}$. Το στιγμιότυπο k IKAN έχει ακριβώς τον ίδιο αριθμό προτάσεων με το στιγμιότυπο 3IKAN ενώ έχει παραπάνω $k - 3$ μεταβλητές και επομένως η αναγωγή είναι πολυωνυμική.

ΔΙΠΛΗ-ΙΚΑΝ

Περιγραφή

Στιγμιότυπο: Έκφραση ϕ σε ΚΣΜ.

Ερώτηση: Υπάρχουν τουλάχιστον δύο αποτυπήσεις που να ικανοποιούν την ϕ .

Απόδειξη ΔΙΠΛΗ-ΙΚΑΝ \in NP

Λύση Η απόδειξη ότι ΔΙΠΛΗ-ΙΚΑΝ \in NP είναι ανάλογη της αντίστοιχης απόδειξης για το πρόβλημα IKAN και επομένως παραλείπεται.

Απόδειξη ΔΙΠΛΗ-ΙΚΑΝ \in NP-hard

Θα δείξουμε $\text{IKAN} \leq_P \text{ΔΙΠΛΗ-ΙΚΑΝ}$. Θεωρούμε ένα στιγμιότυπο IKAN σε ΣΚΜ (βλέπε (1), (2)). Κατασκευάζουμε ένα αντίστοιχο στιγμιότυπο ΔΙΠΛΗ-ΙΚΑΝ ως εξής. Για κάθε πρόταση C_i 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

$$\phi' = \bigwedge_{j=1}^m (A_j \wedge \bar{A}_j).$$

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 . Επίσης είναι πολυωνυμική αφού η ϕ' περιέχει $2m$ προτάσεις και μία επιπλέον μεταβλητή (τη z).

ΟΧΙ-ΟΛΑ-ΙΔΙΑ-4ΙΚΑΝ

Περιγραφή

Στιγμιότυπο: Έκφραση ϕ σε KSM 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;

Απόδειξη ΟΧΙ-ΟΛΑ-ΙΔΙΑ-4ΙΚΑΝΕΝP

Λύση Και για τα δύο προβλήματα είναι εύκολο να δειχθεί ότι ανήκουν στην τάξη NP και επομένως η αντίστοιχη απόδειξη παραλείπεται. Προχωράμε για

Απόδειξη ΟΧΙ-ΟΛΑ-ΙΔΙΑ-4ΙΚΑΝΕΝP-hard

1. Θα δείξουμε ότι $3IKAN \leq_P OXI-OЛА-IDIA-4IKAN$. 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 $T(z) = False$. Προσέξτε 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).

ΟΧΙ-ΟΛΑ-ΙΔΙΑ-3ΙΚΑΝ

Περιγραφή

Στιγμιότυπο: Έκφραση ϕ σε KSM 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;

Απόδειξη ΟΧΙ-ΟΛΑ-ΙΔΙΑ-3ΙΚΑΝΕΝP

Λύση Και για τα δύο προβλήματα είναι εύκολο να δειχθεί ότι ανήκουν στην τάξη NP και επομένως η αντίστοιχη απόδειξη παραλείπεται. Προχωράμε για

Απόδειξη ΟΧΙ-ΟΛΑ-ΙΔΙΑ-3ΙΚΑΝΕΝP-hard

2. Θα δείξουμε ότι $\text{ΟΧΙ-ΟΛΑ-ΙΔΙΑ-4ΙΚΑΝ} \leq_p \text{ΟΧΙ-ΟΛΑ-ΙΔΙΑ-3ΙΚΑΝ}$. 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

$$\begin{aligned} A_j &= a_j \vee b_j \vee w_j, \\ B_j &= c_j \vee d_j \vee \neg w_j, \end{aligned}$$

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. Η έκφραση ϕ' έχει $2 * m$ προτάσεις και m νέες μεταβλητές και άρα ο μετασχηματισμός είναι πολυωνυμικός.

1-3ΙΚΑΝ

Περιγραφή

Στιγμιότυπο: Έκφραση ϕ σε ΚΣΜ με τρεις όρους σε κάθε πρόταση.

Ερώτηση: Υπάρχει κάποια αποτίμηση που να δίνει σε ακριβώς έναν όρο σε κάθε πρόταση τιμή $T(\text{true})$;

Απόδειξη 1-3ΙΚΑΝΕΝP

Και πάλι θα παραλείψουμε την απόδειξη ότι το πρόβλημα ανήκει στην τάξη NP.

Απόδειξη 1-3IKAN \in NP-hard

Λύση Για να γίνει αντιληπτός ο παραπάνω ορισμός παραθέτουμε την παρακάτω 3IKAN έκφραση η οποία δεν είναι 1-3IKAN.

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3).$$

NP. Θα δείξουμε μία αναγωγή από το πρόβλημα 3IKAN. Θεωρούμε λοιπόν ένα 3IKAN στιγμιότυπο όπως περιγράφεται από (1),(2). Η αναγωγή γίνεται ως εξής. Για κάθε πρόταση C_j εισάγουμε στο στιγμιότυπο 1-3IKAN την παρακάτω έκφραση (η οποία αποτελείται τις τέσσερις προτάσεις)

$$(\neg a_j \vee y_j^1 \vee z_j^1) \wedge (\neg b_j \vee y_j^2 \vee z_j^2) \wedge (\neg c_j \vee y_j^3 \vee z_j^3) \wedge (y_j^1 \vee y_j^2 \vee y_j^3)$$

Το παραγόμενο στιγμιότυπο 1-3IKAN περιέχει $4m$ προτάσεις και $6m$ νέες μεταβλητές - άρα η αναγωγή είναι πολυωνυμική. Όμως γιατί δουλεύει; Εφόσον η αποτίμηση T ικανοποιεί την πρόταση C_j τότε ένας τουλάχιστον από τους όρους της πρότασης C_j παίρνει την τιμή *True*. Χωρίς βλάβη της γενικότητας, έστω $T(a_j) = True$. Επεκτείνουμε την αποτίμηση ως εξής

$$\begin{aligned} T(y_j^1) &= True, T(y_j^2) = False, T(y_j^3) = False \\ T(z_j^1) &= False, T(z_j^2) = T(b_j), T(z_j^3) = T(c_j). \end{aligned}$$

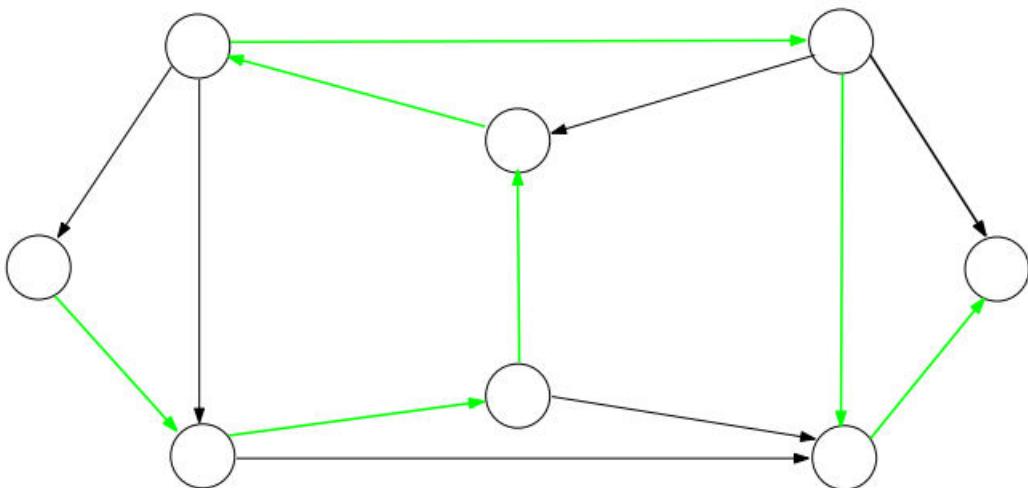
Η αποτίμηση αυτή ικανοποιεί το 1-3IKAN ANN το 3IKAN είναι ικανοποιήσιμο.

KAT-ΜΟΝΟΠΑΤΙ-ΧΑΜΙΛΤΟΝ

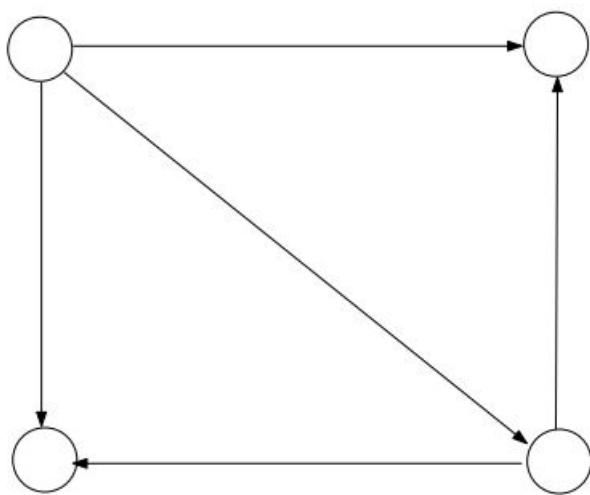
Πρόβλημα

Άσκηση 5 Ένα μονοπάτι Χάμιλτον σε ένα κατευθυνόμενο γράφημα $G(V, E)$, είναι ένα (απλό) μονοπάτι που περιλαμβάνει (επισκέπτεται) όλες τις κορυφές. Είναι προφανές ότι ένα γράφημα δεν έχει απαραίτητα μονοπάτι του Χάμιλτον (για παράδειγμα δες το γράφημα στο Σχήμα 2). Οπότε ορίζουμε το ακόλουθο πρόβλημα απόφασης.

Σχήμα 1. Γράφημα με μονοπάτι Χάμιλτον



Σχήμα 2. Γράφημα χωρίς μονοπάτι Χάμιλτον



Περιγραφή

Στιγμιότυπο: Κατευθυνόμενο γράφημα $G(V, E)$.

Ερώτηση: Υπάρχει ένα κατευθυνόμενο μονοπάτι Χάμιλτον;

Απόδειξη KAT-MONOΠΑΤΙ-ΧΑΜΙΛΤΟΝ \in NP

Λύση Είναι εύκολο να αποδείξουμε ότι ένα προτεινόμενο πιστοποιητικό (αποτελούμενο από μία σειρά κορυφών) μπορεί να ελεγχθεί σε πολυωνυμικό χρόνο (για το αν αποτελεί μονοπάτι Χάμιλτον στο δοθέν γράφημα)- για το λόγο αυτό η απόδειξη παραλείπεται.

Απόδειξη KAT-MONOΠΑΤΙ-ΧΑΜΙΛΤΟΝ \in NP-hard

Στη συνέχεια θα δείξουμε ότι $1\text{-3IKAN} \leq_P \text{KAT-MONOΠΑΤΙ-ΧΑΜΙΛΤΟΝ}$. Θεωρούμε ότι το στιγμιότυπο 1-3IKAN περιέχει n λογικές μεταβλητές (άρα $2n$ σύμβολα τα οποία δεν είναι ανάγκη να εμφανίζονται όλα) και m πράσεις. Θα κατασκευάσουμε ένα κατευθυνόμενο γράφημα $G(V, E)$ το οποίο θα διαθέτει μονοπάτι του Χάμιλτον ANN το στιγμιότυπο 1-3IKAN είναι NAI. Το γράφημα θα περιέχει γραφικές συνιστώσες τύπου "διαμάντι"- μία για κάθε λογική μεταβλητή. Μία συνιστώσα τέτοιου τύπου για τη (λογική) μεταβλητή

x_i αποτελείται καταρχάς από τέσσερις κορυφές, έστω $x_i^{UP}, x_i^{DN}, x_i^{LF}, x_i^{RG}$, και τις κατευθυνόμενες ακμές

$$(x_i^{UP}, x_i^{LF}), (x_i^{LF}, x_i^{DN}), (x_i^{UP}, x_i^{RG}), (x_i^{RG}, x_i^{DN}), \forall i \in \{1, \dots, n\}.$$

Επιπλέον σε κάθε τέτοια συνιστώσα υπάρχει ένα ζευγάρι κορυφών για κάθε πρόταση $C_j, j \in \{1, \dots, m\}$, ήτοι οι κορυφές C_j^a, C_j^b . Κάθε τέτοιο ζευγάρι συνδέεται με δύο ακμές· η μία κατευθύνεται από το C_j^a στο C_j^b και η άλλη αντίστροφα. Επίσης υπάρχουν δύο ακμές ανάμεσα σε κάθε ζευγάρι και στο επόμενο του ως εξής:

$$(C_j^b, C_{j+1}^a), (C_{j+1}^a, C_j^b), \forall j \in \{1, \dots, m-1\}.$$

Παρατηρούμε ότι ανάμεσα στις κορυφές C_j^a, C_j^b δημιουργείται μία "άλυσίδα" που επιτρέπει τη διάσχιση των κορυφών είτε από αριστερά προς τα δεξιά είτε από δεξιά προς τα αριστερά. Το διαμάντι ολοκληρώνεται με τη διασύνδεση της αλυσίδας με τις κορυφές x_i^{LF}, x_i^{RG} . Αυτό γίνεται με την προσθήκη των ακμών $(x_i^{LF}, C_1^a), (C_1^a, x_i^{LF})$ και $(x_i^{RG}, C_m^b), (C_m^b, x_i^{RG})$. Μία συνιστώσα διαμάντι απεικονίζεται στο Σχήμα 3. Ο αριθμός των κορυφών και ακμών σε κάθε τέτοια συνιστώσα είναι

$$|V_i| = 4 + 2m, |E_i| = 4 + 2m + 2(m-1) + 4 = 4m - 6. \quad (4)$$

'Οπως είδαμε και προηγουμένως, οι ακμές είναι τοποθετημένες με τέτοιο τρόπο ώστε για να διασχίσουμε όλους τις κορυφές μίας αλυσίδας είτε θα κινηθούμε από αριστερά προς τα δεξιά είτε ανάποδα. Η φορά αυτή υπονοεί ότι η μεταβλητή που αντιστοιχεί στη συνιστώσα παίρνει τιμή *True* ενώ η ανάποδη φορά σημαίνει τιμή *False*. Οι συνιστώσες "διαμάντι" συνδέονται μεταξύ τους προσθέτοντας τις ακμές

$$(x_i^{DN}, x_{i+1}^{UP}), i \in \{1, \dots, n-1\}, \quad (5)$$

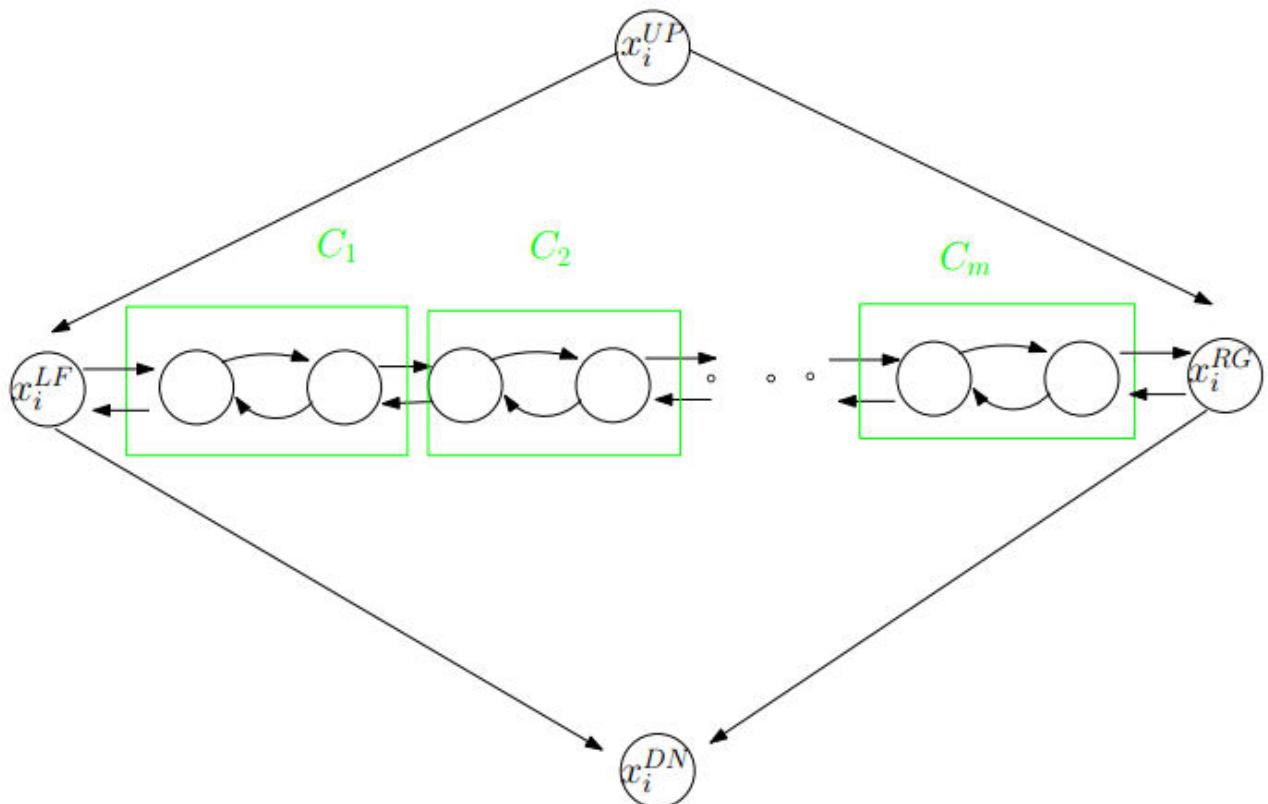
όπως φαίνεται στο Σχήμα 4. Το γράφημα ολοκληρώνεται περιλαμβάνοντας m νέους κόμβους - έναν για κάθε πρόταση C_j . Για κάθε τέτοιο κόμβο εισάγονται έξι ακμές - δύο για κάθε μεταβλητή που εμφανίζεται στην πρόταση. Αν εμφανίζεται ο όρος x στην πρόταση C_j τότε εισάγονται οι ακμές (C_j^a, C_j) και (C_j, C_j^b) , όπου οι κόμβοι C_j^a, C_j^b είναι αυτοί που εμφανίζονται στο διαμάντι της συγκεκριμένης μεταβλητής x για την πρόταση C_j . Συμμετρικά, αν εμφανίζεται ο όρος $\neg x$ στην πρόταση C_j τότε εισάγονται οι ακμές (C_j^b, C_j) και (C_j, C_j^a) . Το γράφημα $G(V, E)$ που απεικονίζεται στο Σχήμα 5 περιέχει όλες τις κορυφές C_j και για τις C_1, C_m περιέχει τις ακμές που υποδηλώνουν ότι ο όρος x_1 εμφανίζεται στην πρώτη ενώ ο $\neg x_1$ στη δεύτερη. 'Οπως έχουμε ήδη πει

το γράφημα θα είναι ολοκληρωμένο όταν για κάθε C_j έχουμε τρία ζευγάρια ακμών - για κάθε όρο που εμφανίζεται στην πρόταση ένα ζευγάρι ακμών προς τις κορυφές C_j^a, C_j^b του διαμαντιού που αντιστοιχεί στη μεταβλητή του όρου αυτού.

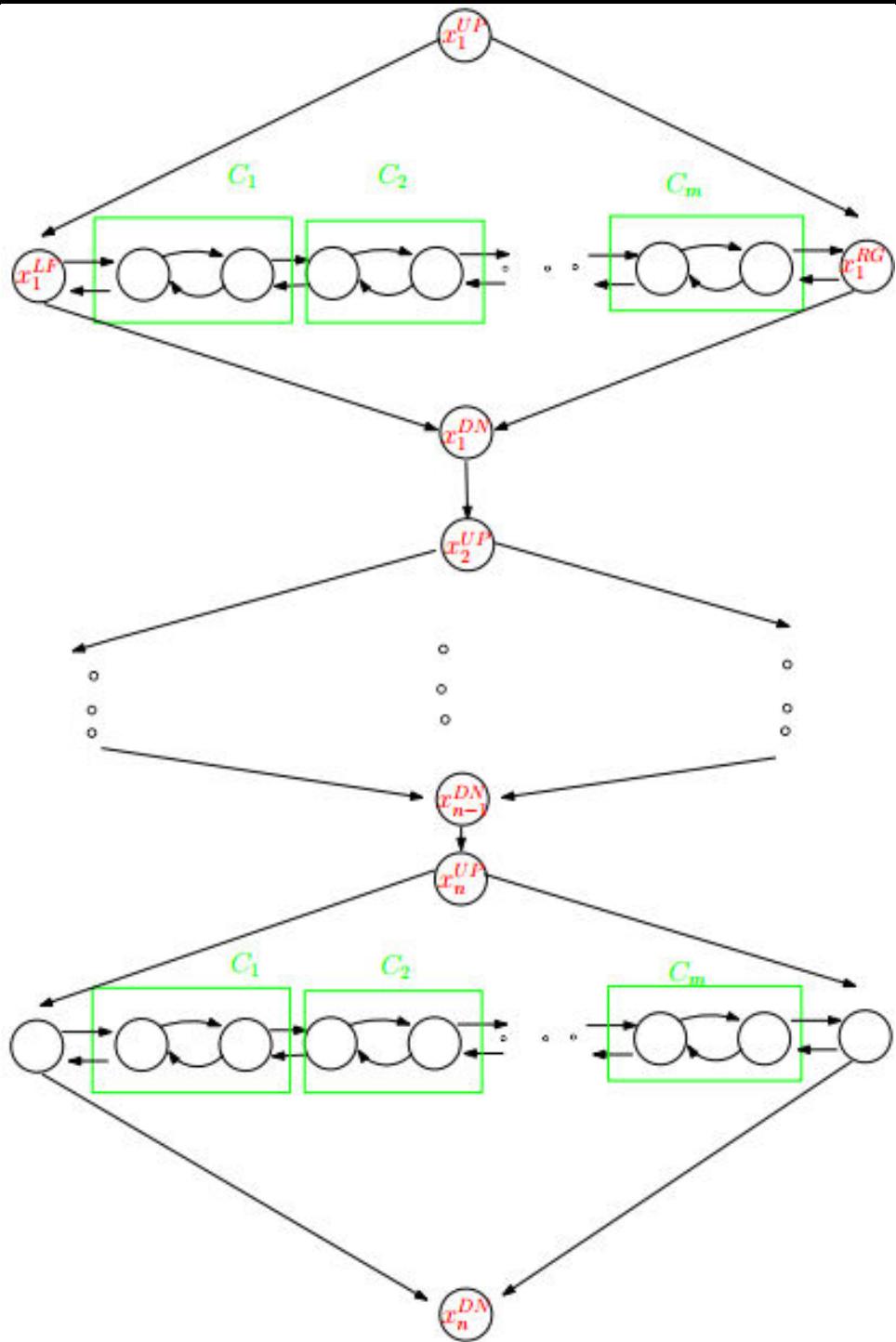
Έστω ότι έχουμε μία αποτίμηση η οποία να ικανοποιεί το στιγμιότυπο 3IKAN. Τότε μπορούμε να θεωρήσουμε ένα μονοπάτι το οποίο ξεκινάει από το x_i^{UP} καταλήγει στο x_i^{DN} και διασχίζει τους κόμβους κάθε "διαμαντιού" είτε από αριστερά προς τα δεξιά αν η μεταβλητή στην συγκεκριμένη αποτίμηση πάρει τιμή *True* και από δεξιά προς τα αριστερά αν πάρει τιμή *False*. Προφανώς το μονοπάτι αυτό περνάει από όλους τους κόμβους των διαμαντιών μία ακριβώς φορά και δεν περιλαμβάνει τους κόμβους που αντιστοιχούν στις προτάσεις. Όμως κάθε τέτοιο κόμβο μπορούμε να τον συμπεριλάβουμε στο μονοπάτι επιλέγοντας για κάθε μία πρόταση τον όρο που σε αυτή γίνεται $T(RUE)$.

Επειδή το στιγμιότυπο είναι 1-3IKAN κάθε εξωτερική κορυφή C_j περιλαμβάνεται μόνο μια φορά (στο μονοπάτι).

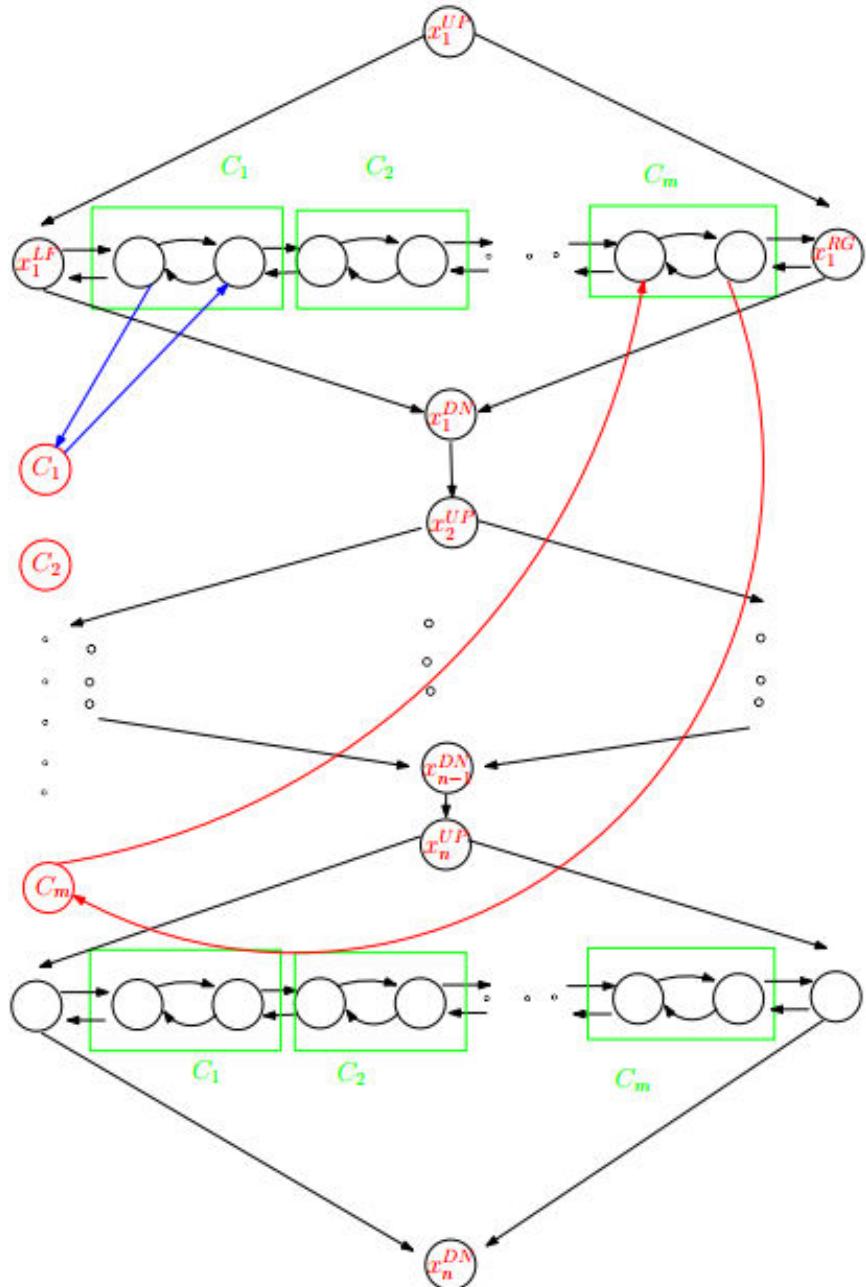
Σχήμα 3. Συνιστώσα τύπου «διαμάντι». Μία τέτοια συνιστώσα για κάθε μεταβλητή



Σχήμα 4. Συνιστώσες τύπου «διαμάντι» σε συνδεδεμένο γράφημα



Σχήμα 5. Γράφημα που αντιστοιχεί σε στιγμιότυπο με x_1 , $\neg x_1$ να εμφανίζονται στις προτάσεις C_1, C_m αντίστοιχα



Παράδειγμα

Παράδειγμα 1 Θεωρούμε το NAI στιγμιότυπο του ZIKAN

$$x_1 \vee x_2 \vee \neg x_3$$

$$x_1 \vee \neg x_2 \vee x_3.$$

Το αντίστοιχο γράφημα απεικονίζεται στο Σχήμα 6. Έστω τώρα η αποτίμηση $T(x_1) = False$, $T(x_2) = True$, $T(x_3) = True$ η οποία ικανοποιεί το στιγμιότυπο του παραδείγματος. Αντί να περιλάβουμε την ακμή (C_1^a, C_1^b) στο μονοπάτι (στο δεύτερο “διαμάντη”) μπορούμε να περιλάβουμε και τον κόμβο C_1 στο μονοπάτι χρησιμοποιώντας τις ακμές (C_1^a, C_1) , (C_1, C_1^b) . Ομοίως αντικαθιστούμε στο μονοπάτι την ακμή (C_2^a, C_2^b) με τις (C_2^a, C_2) , (C_2, C_2^b) . Το μονοπάτι απεικονίζεται στο Σχήμα 7.

Αντίστροφα τώρα, έστω ότι στο γράφημα υπάρχει ένα μονοπάτι του Χάμιλτον. Αυτό δεν μπορεί να εγκαταλείπει ένα διαμάντι χωρίς να διασχίζει όλους τους οριζόντιους κόμβους του. Γιατί όμως δεν μπορεί να συμβαίνει αυτό· Προσέξτε ότι ο μόνος τρόπος να γίνει αυτό είναι μέσω κάποιου κόμβου C_j . Έστω λοιπόν ότι είμαστε στο “διαμάντι” που σχετίζεται με τη μεταβλητή x_i και το μονοπάτι κάνει διάσχιση από τα αριστερά προς τα δεξιά. Έστω ότι το μονοπάτι χρησιμοποιεί την ακμή (C_j^a, C_j) και η επόμενη ακμή οδηγεί έξω από το “διαμάντι”, Κάποια στιγμή το μονοπάτι θα πρέπει να επισκεφθεί και τον κόμβο C_{j+1}^b έχοντας μεταπηδήσει από κάποιο άλλο “διαμάντι” πίσω στο συγκεκριμένο “διαμάντι” (αφού είναι μονοπάτι του Χάμιλτον και άρα όλοι οι κόμβοι πρέπει να υπάρχουν πάνω σε αυτό). Όμως για να γίνει αυτό θα πρέπει να φτάσει στο C_{j+1}^b από τον κόμβο C_{j+1}^a και άρα ο επόμενος κόμβος στο μονοπάτι θα είναι ο C_j^a - άτοπο αφού έχουμε ήδη επισκεφτεί τον κόμβο αυτό αφού χρησιμοποιήσαμε την ακμή (C_j^a, C_j) . Η περίπτωση στην οποία διασχίζουμε το διαμάντι από δεξιά προς τα αριστερά είναι απολύτως συμμετρική.
Επομένως το μονοπάτι του Χάμιλτον διασχίζει τους κόμβους ενός διαμαντιού στη σειρά είτε από αριστερά προς τα δεξιά είτε από τα δεξιά προς τα αριστερά. Στην πρώτη περίπτωση θέτουμε την μεταβλητή στην τιμή *True* ενώ στη δεύτερη στην τιμή *False*. Αν χρησιμοποιείται στη διάσχιση και τον κόμβο της πρότασης C_j τότε η αποτίμηση που δείξαμε χρησιμοποιείται για να ικανοποιεί τη συγκεκριμένη πρόταση.

Τέλος παρατηρούμε ότι στο γράφημα $G(V, E)$ έχουμε

- $4 + 2m$ κορυφές σε κάθε διαμάντι (4) και έχουμε n διαμάντια και
- m κορυφές εξωτερικές (ένας για κάθε πρόταση C_j) από τα διαμάντια.

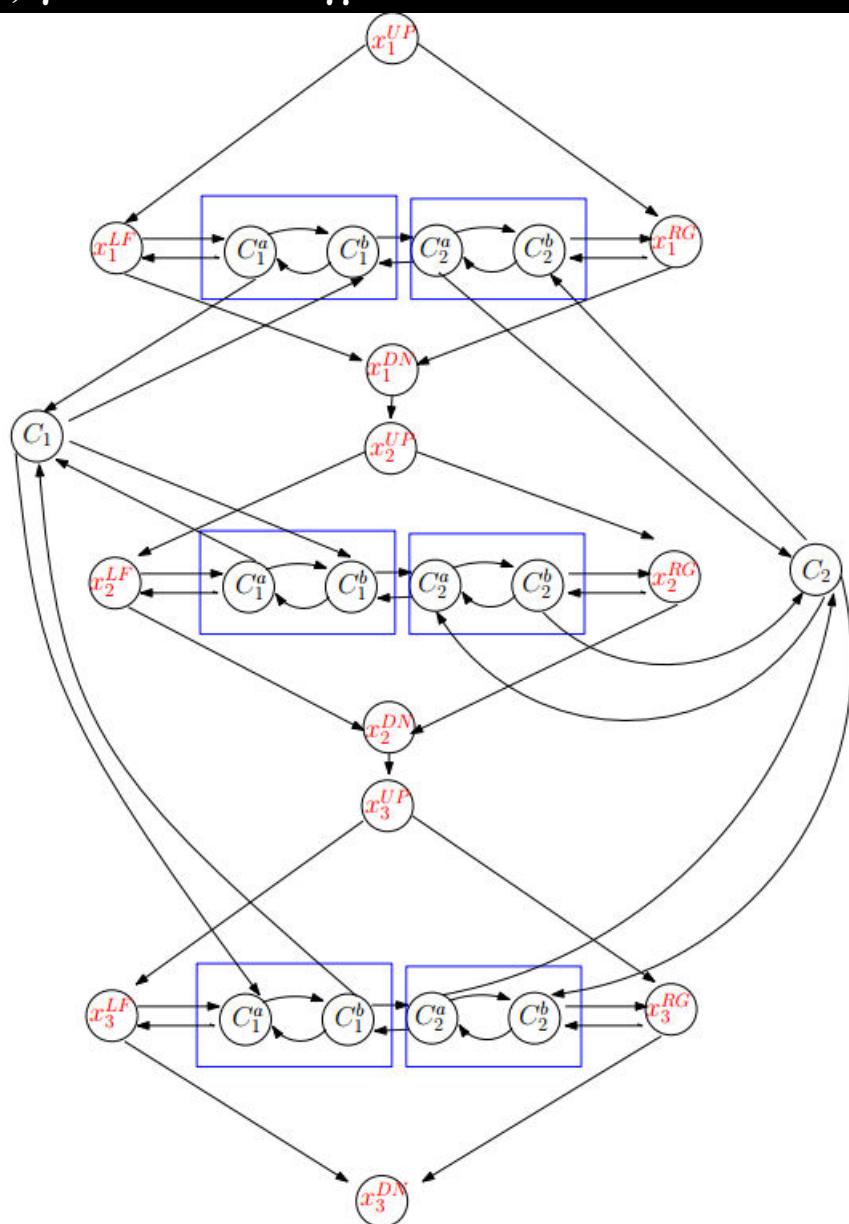
Συνολικά ο αριθμός των κορυφών είναι $(4 + 2m)n + m = O(mn)$. Επίσης στο γράφημα $G(V, E)$ έχουμε

- $4m - 6$ ακμές σε κάθε διαμάντι (4) και έχουμε n διαμάντια και
- $n - 1$ ακμές για την σύνδεση των διαμαντιών(5) και
- $6m$ ακμές οι οποίες συνδέουν τις εξωτερικές κορυφές με τις κορυφές τα διαμάντια.

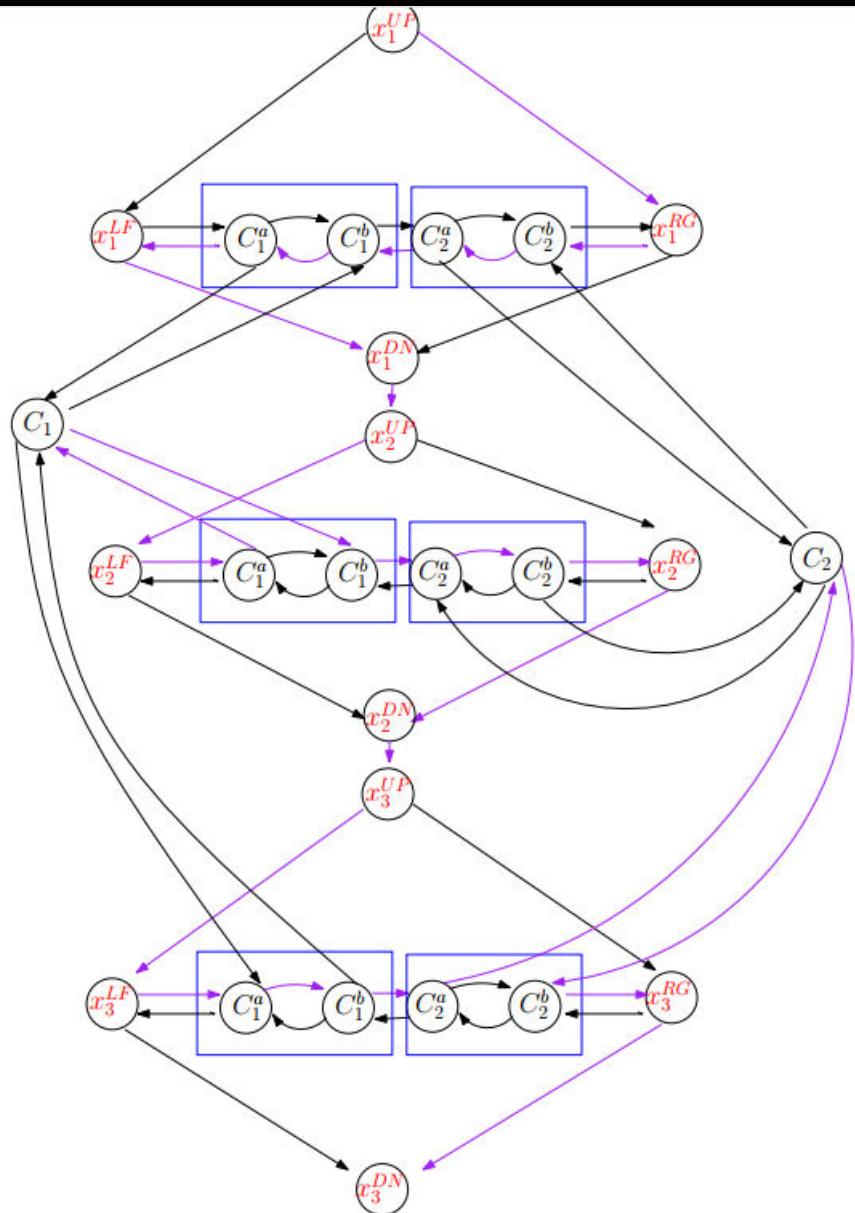
Συνολικά ο αριθμός των ακμών είναι $(4m - 6)n + n - 1 + 6m = 4mn - 5n + 6m - 1 = O(mn)$.

Από τα παραπάνω προκύπτει ότι η τάξη και το μέγεθος του γραφήματος είναι $O(mn)$ και επομένως ο μετασχηματισμός είναι πολυωνυμικός.

Σχήμα 6. $G(V, E)$ για ένα ΝΑΙ στιγμιότυπο 3IKAN



Σχήμα 7. Οι μωβ ακμές ανήκουν στο μονοπάτι Χάμιλτον που προκύπτει από την αποτίμηση $T(x_1) = \text{False}$, $T(x_2) = \text{True}$, $T(x_3) = \text{True}$



01-ΙΣΟΤ

Περιγραφή

Σπιγμιότυπο: 0/1 πίνακας A .

Ερώτηση: Υπάρχει 0/1 διάνυσμα y τέτοιο ώστε $Ay = e$, όπου e είναι ένα διάνυσμα με στοιχεία 1;

Απόδειξη 01-ΙΣΟΤ ∈ NP

Λύση Έστω ότι ο πίνακας A έχει m γραμμές και n στήλες. Είναι εύκολο να δείξουμε ότι το πρόβλημα ανήκει στην τάξη NP αφού ο έλεγχος για το αν ένα δοσμένο διάνυσμα y ικανοποιεί το σύστημα απαιτεί $O(mn)$ πράξεις.

Απόδειξη 01-ΙΣΟΤΕΝP-hard

Στη συνέχεια θα δείξουμε μία αναγωγή από το πρόβλημα 1-3IKAN. Δηλαδή, ξεκινώντας από ένα στιγμιότυπο 1-3IKAN θα κατασκευάσουμε ένα στιγμιότυπο 01ΙΣΟΤ (στην ουσία έναν 0/1 πίνακα A) για τον οποίο θα ισχύει ότι η παράσταση ϕ θα είναι ικανοποιήσιμη ANN το σύστημα $Ay = e$ έχει λύση. Η κατασκευή του πίνακα γίνεται ως εξής. Οι στήλες του πίνακα αντιστοιχούν στους όρους των μεταβλητών του στιγμιότυπου 1-3IKAN. Οι γραμμές του πίνακα χωρίζονται σε δύο σύνολα. Στο πρώτο σύνολο έχουμε μία γραμμή για κάθε μεταβλητή: η γραμμή περιέχει δύο 1 (στις στήλες των όρων της μεταβλητής) και παντού αλλού 0. Στο δεύτερο σύνολο έχουμε μία γραμμή για κάθε πρόταση C_j : η γραμμή περιέχει τρία 1 και όλα τα υπόλοιπα στοιχεία 0. Τα στοιχεία με την τιμή 1 βρίσκονται στις στήλες που αντιστοιχούν ατούς όρους που εμφανίζονται στην πρόταση. Ο πίνακας A έχει $n + m$ γραμμές και $2 * n$ στήλες. Έστω T μία αποτίμηση που ικανοποίει την έκφραση ϕ που περιγράφει το στιγμιότυπο 1-3IKAN. Το διάνυσμα y με τιμές

$$\begin{aligned} y_i &= 1, y_{i+n} = 0 \text{ αν } T(x_i) = T(\text{true}), \\ y_i &= 0, y_{i+n} = 1 \text{ αν } T(x_i) = F(\text{false}), \end{aligned}$$

για $i \in \{1, \dots, n\}$, ικανοποιεί το σύστημα $Ay = e$.

Αυτό συμβαίνει αφού για κάθε μία από τις n πρώτες γραμμές μόνο ένα από τα δύο στοιχεία (αυτά που αντιστοιχούν στους όρους της μεταβλητής της γραμμής) από το y έχει τιμή 1. Αντίστοιχα, για κάθε μία τις υπόλοιπες γραμμές, εφόσον το στιγμιότυπο είναι 1-3IKAN, μόνο ένα από τα στοιχεία που αντιστοιχούν στους όρους που εμφανίζονται στην πρόταση της γραμμής έχει τιμή 1 στην αντίστοιχή θέση στο διάνυσμα y .

Παράδειγμα

Παράδειγμα 2 Θεωρούμε το NAI στιγμιότυπο του 1-3IKAN

$$\begin{aligned} x_1 \vee \neg x_2 \vee \neg x_3 \\ \neg x_1 \vee x_2 \vee \neg x_3. \end{aligned}$$

Ο αντίστοιχος πίνακας A (πρώτα εμφανίζονται οι στήλες που αντιστοιχούν στους θετικούς όρους) και το διάνυσμα y (τα πρώτα n στοιχεία αντιστοιχούν στους θετικούς όρους) που αντιστοιχεί στην αποτίμηση $T(x_1) = T(x_2) = T(x_3) = T(\text{true})$ είναι

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}, y = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Περιγραφή

Στιγμιότυπο: Ακέραιοι d_1, \dots, d_n, t .

Ερώτηση: Υπάρχει υποσύνολο $J \subseteq \{1, \dots, n\}$ τέτοιο ώστε $\sum_{j \in J} d_j = t$;

Απόδειξη ΥΠΟΣΥΝΑΘΡ \in NP-complete

Λύση Μπορούμε να κάνουμε αναγωγή από το 01ΙΣΟΤ ως εξής. Θεωρούμε κάθε στήλη του πίνακα A σαν κωδικοποίηση ενός αριθμού σε ένα αριθμητικό σύστημα σε κάποια βάση b . Επομένως οι στήλες αντιστοιχούν στους ακέραιους d_1, \dots, d_n (στο αριθμητικό σύστημα με βάση το b) και

$$t = \underbrace{1 \cdots 1}_m_b.$$

Για παράδειγμα, θεωρώντας $b = 10$ ο πίνακας A που περιγράφει το στιγμιότυπο 01ΙΣΟΤ προηγουμένως οδηγεί στο ακόλουθο στιγμιότυπο ΥΠΟΣΥΝΑΘΡ.

$$\begin{aligned}d_1 &= 1 * 10^4 + 0 * 10^3 + 0 * 10^2 + 1 * 10^1 + 0 * 10^0 = 10010, \\d_2 &= 0 * 10^4 + 1 * 10^3 + 0 * 10^2 + 0 * 10^1 + 1 * 10^0 = 1001, \\d_3 &= 0 * 10^4 + 0 * 10^3 + 1 * 10^2 + 0 * 10^1 + 0 * 10^0 = 100, \\d_4 &= 1 * 10^4 + 0 * 10^3 + 0 * 10^2 + 0 * 10^1 + 1 * 10^0 = 10001, \\d_5 &= 0 * 10^4 + 1 * 10^3 + 0 * 10^2 + 1 * 10^1 + 0 * 10^0 = 1010, \\d_6 &= 0 * 10^4 + 0 * 10^3 + 1 * 10^2 + 1 * 10^1 + 1 * 10^0 = 111, \\t &= 1 * 10^4 + 1 * 10^3 + 1 * 10^2 + 1 * 10^1 + 1 * 10^0 = 11111.\end{aligned}$$

Συνεπώς αν το 01ΙΣΟΤ είναι ΝΑΙ στιγμιότυπο, το αντίστοιχο διάνυσμα y είναι το χαρακτηριστικό διάνυσμα του υποσυνόλου J .

Προσοχή: Η παραπάνω αναγωγή ισχύει με δεδομένο ότι το άθροισμα των αριθμών που κωδικοποιούν οι στήλες δεν ξεπερνά το $t = \underbrace{1 \cdots 1}_m_b$. Αυτό μπορεί να συμβεί αν υπάρχουν κρατούμενα στις προσθέσεις του αριστερού μέλους. Για να διασφαλιστεί αυτό θα πρέπει $b \geq n + 1$.

ΑΝΣΥΝΟΛΟ

Πρόβλημα

Άσκηση 8 Σε ένα μη-κατευθυνόμενο γράφημα $G(V, E)$ ένα ανεξάρτητο σύνολο είναι ένα σύνολο $V' \subseteq V$ τέτοιο ώστε για κάθε $v, u \in V'$ δεν υπάρχει ακμή (v, u) . Μια κάλυψη κόμβων είναι ένα σύνολο $\hat{V} \subseteq V$ τέτοιο ώστε για κάθε ακμή (v, u) είτε $v \in \hat{V}$ ή $u \in \hat{V}$. Ορίζουμε τα ακόλουθα προβλήματα

Περιγραφή

Στιγμιότυπο: Μη-κατευθυνόμενο γράφημα $G(V, E)$, ακέραιος k

Ερώτηση: Υπάρχει ένα ανεξάρτητο σύνολο μεγέθους k ;

Απόδειξη ΑΝΣΥΝΟΛΟΕΝP-complete

- Κατασκευάζουμε το συμπληρωματικό γράφημα του $G(V, E)$ το οποίο ορίζεται σαν $G(\bar{V}, \bar{E})$ όπου $\bar{V} = V$ και $\bar{E} = \{(v, u) : v, u \in V, (v, u) \notin E\}$. Προφανώς ένα clique μεγέθους k στο $G(\bar{V}, \bar{E})$ αποτελεί ένα ανεξάρτητο σύνολο στο $G(V, E)$.

ΚΑΛΥΨΗΚΟΜΒΩΝ

Πρόβλημα

Άσκηση 8 Σε ένα μη-κατευθυνόμενο γράφημα $G(V, E)$ ένα ανεξάρτητο σύνολο είναι ένα σύνολο $V' \subseteq V$ τέτοιο ώστε για κάθε $v, u \in V'$ δεν υπάρχει ακμή (v, u) . Μία κάλυψη κόμβων είναι ένα σύνολο $\hat{V} \subseteq V$ τέτοιο ώστε για κάθε ακμή (v, u) είτε $v \in \hat{V}$ ή $u \in \hat{V}$. Ορίζουμε τα ακόλουθα προβλήματα

Περιγραφή

Στιγμιότυπο: Μη-κατευθυνόμενο γράφημα $G(V, E)$, ακέραιος k

Ερώτηση: Υπάρχει μία κάλυψη μεγέθους k ;

Απόδειξη ΚΑΛΥΨΗΚΟΜΒΩΝΕΝP-complete

- Έστω $S \subseteq V$ ένα ανεξάρτητο σύνολο του $G(V, E)$. Τότε το σύνολο $V \setminus S$ αποτελεί μία κάλυψη κόμβων του $G(V, E)$ (γιατί;) Άρα αρκεί να αναζητήσουμε ένα ανεξάρτητο σύνολο μεγέθους $n - k$ στο $G(V, E)$, όπου n ο αριθμός των κόμβων του $G(V, E)$.

3ΧΡΩΜΑΤΙΣΜΟΣ

Περιγραφή

Στιγμιότυπο: Μη-κατευθυνόμενο γράφημα $G(V, E)$, χρώματα T, F, B

Ερώτηση: Υπάρχει αντιστοίχηση χρωμάτων στους κόμβους του γραφήματος τέτοια ώστε κάθε ζευγάρι κόμβων που συνδέεται με ακμή να χρωματίζεται με διαφορετικό χρώμα;

Απόδειξη 3ΧΡΩΜΑΤΙΣΜΟΣεNP-complete

Λύση Θα κάνουμε αναγωγή από το πρόβλημα 3IKAN. Θεωρούμε λοιπόν ένα στιγμιότυπο 3IKAN στη γενική μορφή των (1) και (2). Θα κατασκευάσουμε ένα γράφημα $G(V, E)$ το οποίο θα είναι χρωματίσιμο με τρία χρώματα ANN το 3IKAN είναι ικανοποιήσιμο. Θεωρούμε το σύνολο των τριών χρωμάτων

$$\{T(true), F(false), B(ase)\}. \quad (6)$$

Θα χρησιμοποιήσουμε δύο γραφικές συνιστώσες· η πρώτη αφορά τους όρους και η δεύτερη τις προτάσεις. Η γραφική συνιστώσα που αφορά τους όρους απεικονίζεται στο Σχήμα 8.

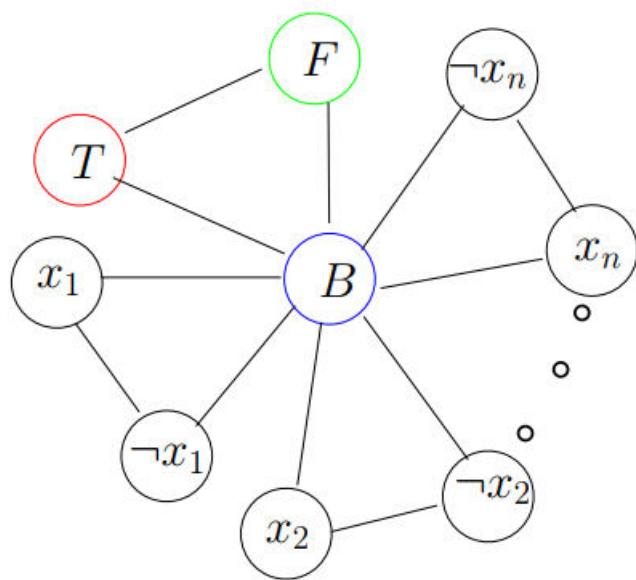
Για κάθε πρόταση $C_j = a_j \vee b_j \vee c_j$ έχουμε μία γραφική (προτασιακή) συνιστώσα που απεικονίζεται στο Σχήμα 9. Παρατηρείστε ότι αν οι κόμβοι a, b, c λάβουν το χρώμα *False* (δηλαδή η C_j και άρα η ϕ δεν είναι ικανοποιήσιμη) τότε ο κόμβος C_j στο γράφημα (κόμβος που αντιστοιχεί στην πρόταση $a \vee b \vee c$) αναγκαστικά πρέπει να πάρει το ίδιο χρώμα (δηλαδή *F(false)*). Διαφορετικά, αν τουλάχιστον ένας από τους όρους πάρει την τιμή *True* υπάρχει χρωματισμός των υπολοίπων κόμβων (από το σύνολο χρωμάτων (6)) ώστε ο κόμβος αυτός να πάρει το χρώμα *True*.

Το γράφημα $G(V, E)$ απεικονίζεται στο Σχήμα 10 με μία συνιστώσα που αφορά την πρόταση C_j . Το γράφημα έχει m τέτοιες συνιστώσες - μία για κάθε τιμή του δείκτη j - με τον αντίστοιχο κόμβο C_j να συνδέεται με ακμή με τους κόμβους *B* και *F(false)*.

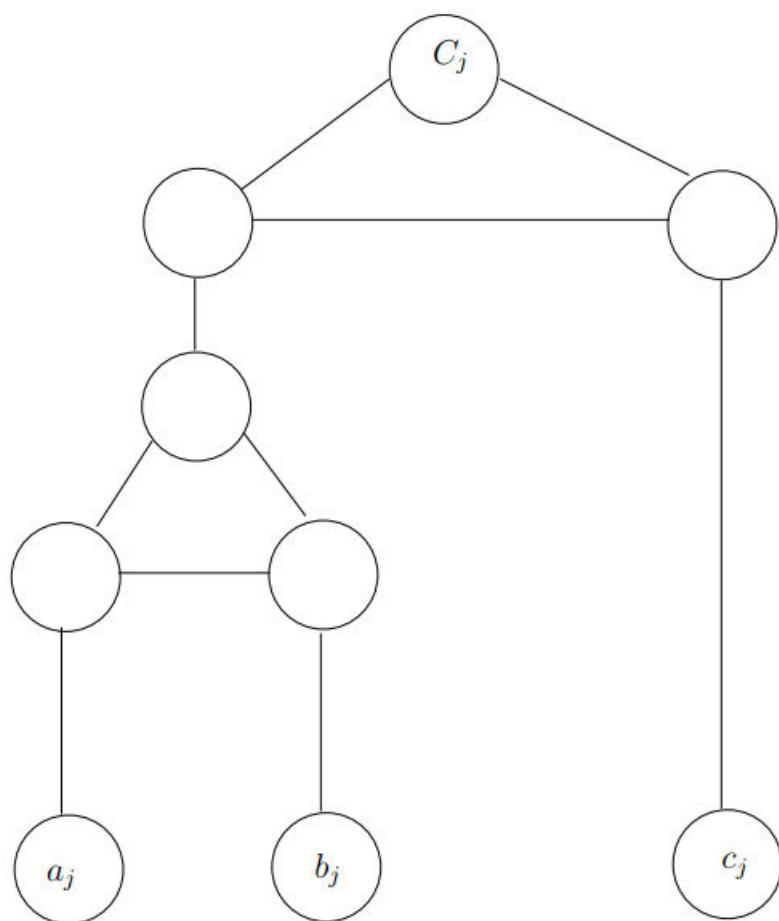
Αν η ϕ (δες (1)) είναι ικανοποιήσιμη τότε οι κόμβοι που αφορούν τις μεταβλητές παίρνουν τα χρώματα *True*, *False* από την αποτίμηση που ικανοποιεί την έκφραση. Το ίδιο συμβαίνει για τους κόμβους a_j, b_j, c_j σε κάθε προτασιακή συνιστώσα - θα πρέπει να χρησιμοποιηθούν τα χρώματα από την αποτίμηση και για τους κόμβους αυτούς. Τέλος οι εσωτερικοί κόμβοι της κάθε προτασιακής συνιστώσας χρωματίζονται κατάλληλα από το σύνολο των χρωμάτων (6) ώστε ο κάθε κόμβος C_j να παίρνει το χρώμα *T(true)*. Αντίστροφα, αν το γράφημα είναι 3-χρωματίσιμο τότε αναγκαστικά οι κόμβοι C_j παίρνουν το ίδιο χρώμα (δηλαδή *T(true)*). Επίσης υπάρχει χρωματισμός όπου κάθε κόμβος με βαθμό 1 (σε κάθε προτασιακή συνιστώσα) παίρνει το ίδιο χρώμα με τον κόμβο που αντιστοιχεί στον ίδιο όρο στη γραφική συνιστώσα που αφορά τις μεταβλητές (αφού οι κόμβοι αυτοί δεν ενώνονται με ακμή).

Το $G(V, E)$ έχει m συνιστώσες που αφορούν προτάσεις - κάθε μία έχει 9 κόμβους, 2 n κόμβους για τους όρους και 3 κόμβους με τα χρώματα *B(ase)*, *F(false)*, *T(true)*. Συνολικά το G περιέχει $9m + 2n + 3$ κόμβους και άρα η μετατροπή είναι πολυωνυμική.

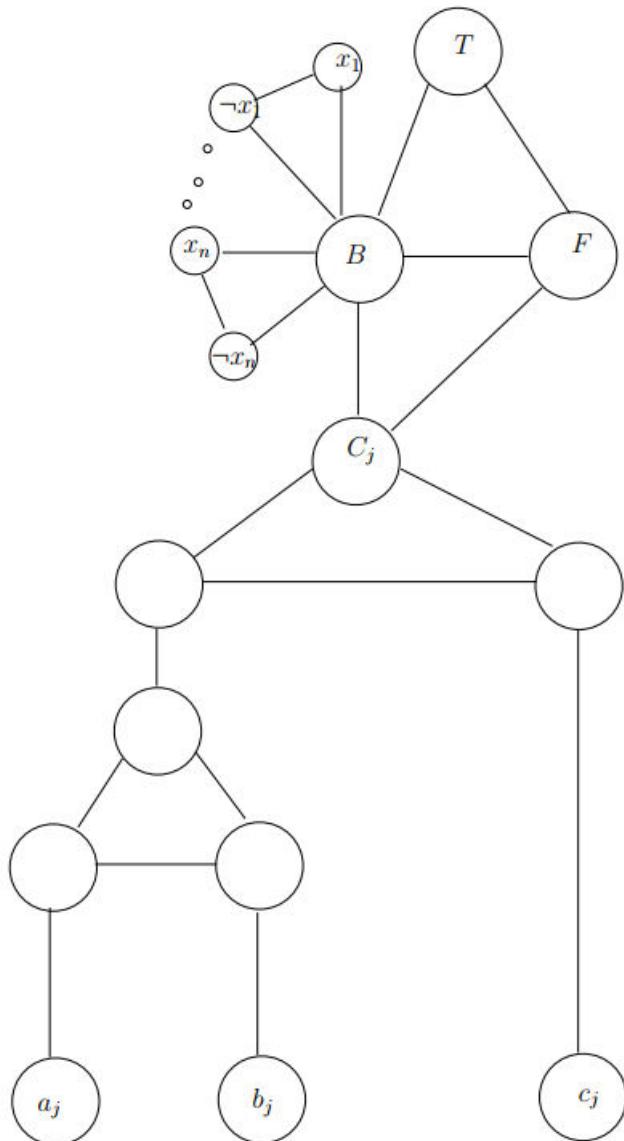
Σχήμα 8. Συνιστώσα που αφορά τους όρους



Σχήμα 9. Συνιστώσα για την πρόταση C_j



Σχήμα 10. Το $G(V, E)$ σε απλοποιημένη έκδοση – με μια πρόταση C_j



ΔΙΑΧΩΡΙΣΜΟΣ

Περιγραφή

Στιγμιότυπο: Σύνολο ακεραιών S

Ερώτηση: Υπάρχουν σύνολα $S_1, S_2 \subset S$ με $S_1 \cup S_2 = S, S_1 \cap S_2 = \emptyset$ τέτοια ώστε $\sum\{s : s \in S_1\} = \sum\{s : s \in S_2\}$; Να δείξετε ότι τα παραπάνω προβλήματα

Απόδειξη ΔΙΑΧΩΡΙΣΜΟΣ \in NP

Λύση: Προφανώς το πρόβλημα ανήκει στην τάξη NP.

Απόδειξη ΔΙΑΧΩΡΙΣΜΟΣ \in NP-hard

νυμική αναγωγή από το πρόβλημα ΥΠΟΣΥΝΑΘΡ. Σε ένα ΝΑΙ στιγμιότυπο του προβλήματος αυτού μας δίνεται μία τιμή t και ένα σύνολο από ακέραιους $D = \{d_1, \dots, d_n\}$ και υπάρχει $J \subset \{1, \dots, n\}$ τέτοιο ώστε

$$\sum_{j \in J} d_j = t. \quad (7)$$

Δημιουργούμε το σύνολο $S = D \cup \{2t - s\}$ όπου $s = \sum_{i=1, \dots, n} d_i$. Το άθροισμα των στοιχείων του S είναι $2t$. Επίσης μέσα στο D υπάρχει ένα υποσύνολο στοιχείων που αθροίζει σε t (αυτά που δεικτοδοτούνται από το J) και άρα τα υπόλοιπα αθροίζουν πάλι σε t (αφού το άθροισμα του συνόλου των στοιχείων του S είναι $2t$). Άρα υπάρχει ο ζητούμενος διαχωρισμός του S αν το στιγμιότυπο του ΥΠΟΣΥΝΑΘΡ είναι NAI.

Αντίστροφα αν ισχύει ότι το S διαχωρίζεται σε δύο σύνολα S_1, S_2 με $\sum\{d : d \in S_1\} = \sum\{d : d \in S_2\}$, θα δείξουμε ότι υπάρχει σύνολο $J \subset \{1, \dots, n\}$ τέτοιο ώστε να ισχύει η (7). Εφόσον ισχύει ο παραπάνω διαχωρισμός του S σε S_1, S_2 , μόνο το ένα από τα δύο αυτά σύνολα έχει το επιπλέον στοιχείο $2t - s$. Έστω ότι αυτό είναι το σύνολο S_2 . Επομένως,

$$\sum\{d : d \in S_1\} = \sum\{d : d \in S_2 \setminus \{2t - s\}\} + 2t - s.$$

Παρατηρούμε ότι $S_2 \setminus \{2t - s\} \subset \{d_1, \dots, d_n\}$ και επομένως

$$\sum\{d : d \in S_1\} = s - \sum\{d : d \in S_2 \setminus \{2t - s\}\}.$$

Από τις δύο τελευταίες σχέσεις προκύπτει ότι

$$s - \sum\{d : d \in S_2 \setminus \{2t - s\}\} = \sum\{d : d \in S_2 \setminus \{2t - s\}\} + 2t - s,$$

και επομένως

$$\begin{aligned} t &= s - \sum\{d : d \in S_2 \setminus \{2t - s\}\} \Rightarrow \\ &\qquad\qquad\qquad t = \sum_{j \in J} d_j, \end{aligned}$$

όπου J δεικτοδοτεί αποκλειστικά στοιχεία από το σύνολο $\{d_1, \dots, d_n\}$.