Κεφάλαιο 1

Εισαγωγή

> Ή πραγματική βελτιστοποίηση είναι η επαναστατική συμβολή της σύγχρονης έρευνας στις διαδικασίες λήψης αποφάσεων.¨
> - George Dantzig

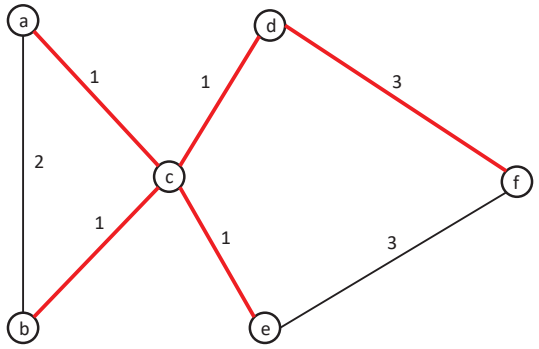
Ας ξεκινήσουμε αυτό το βιβλίο από ένα θεμελιώδες ερώτημα και ας σας πούμε τι θα αποτελέσει αυτό το βιβλίο.

1.1 Τι είναι Συνδυαστική Βελτιστοποίηση;

Στόχος της συνδυαστικής βελτιστοποίησης είναι να βρεθεί ένα βέλτιστο αντικείμενο από ένα πεπερασμένο σύνολο αντικειμένων. Αυτά τα υποψήφια αντικείμενα ονομάζονται *εφικτές λύσεις* ενώ το βέλτιστο ονομάζεται *βέλτιστη λύση*. Για παράδειγμα, θεωρήστε το ακόλουθο πρόβλημα.

Πρόβλημα 1.1.1 (Ελάχιστο Τανύον Δένδρο)**.** *Δίνεται ένα συνεκτικό γράφημα $G = (V, E)$ με μη αρνητικό βάρος τόξων $c : E \to R_+$, να βρείτε ένα δένδρο με ελάχιστο συνολικό βάρος, όπου 'τανύον' σημαίνει ότι όλοι οι κόμβοι περιλαμβάνονται στο δένδρο και συνεπώς ένα δένδρο που εκτείνεται διασυνδέει όλους τους κόμβους του V. Ένα παράδειγμα είναι όπως φαίνεται στο Σχήμα 1.1.*

Προφανώς, το σύνολο όλων των δένδρων που εκτείνονται είναι πεπερασμένο και ο στόχος αυτού του προβλήματος είναι να βρεθεί ένα από αυτά με ελάχιστο συνολικό βάρος (κόστος). Κάθε δένδρο είναι μια εφικτή λύση και η βέλτιστη λύση είναι το δένδρο με το ελάχιστο συνολικό βάρος (κόστος), το



Σχήμα 1.1: Παράδειγμα ελαχίστου τανυόντος δένδρου.

οποίο ονομάζεται επίσης *ελάχιστο τανύον δένδρο*. Επομένως, πρόκειται για ένα πρόβλημα συνδυαστικής βελτιστοποίησης.

Ένα πρόβλημα συνδυαστικής βελτιστοποίησης μπορεί να έχει περισσότερες από μία βέλτιστες λύσεις. Για παράδειγμα, στο Σχήμα 1.1, υπάρχουν δύο δέντρα με ελάχιστο συνολικό μήκος. (Το δεύτερο μπορεί να ληφθεί χρησιμοποιώντας την ακμή (e, f) για να αντικαταστήσει την ακμή (d, f)). Επομένως, η βέλτιστη λύση όπως αναφέρεται παραπάνω, εννοείται ένα γενικό μέλος του συνόλου των βέλτιστων λύσεων.

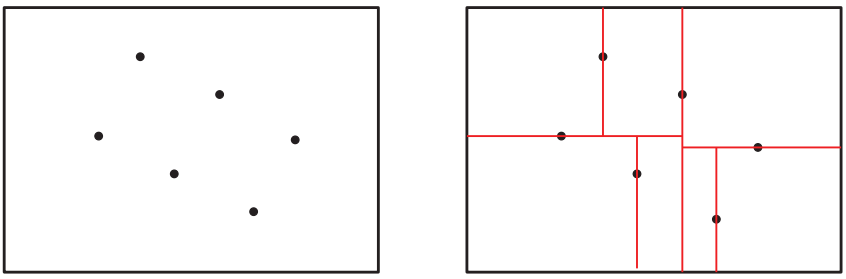
Η συνδυαστική βελτιστοποίηση είναι ένα κατάλληλο υποπεδίο της διακριτής βελτιστοποίησης. Στην πραγματικότητα, υπάρχουν κάποια προβλήματα στη διακριτή βελτιστοποίηση, τα οποία δεν ανήκουν στη συνδυαστική βελτιστοποίηση. Για παράδειγμα, θεωρήστε τον ακέραιο προγραμματισμό. Ανήκει πάντα στη διακριτή βελτιστοποίηση. Ωστόσο, όταν το εφικτό πεδίο είναι άπειρο, δεν ανήκει στη συνδυαστική βελτιστοποίηση. Όμως, μια τέτοια διαφορά δεν αναγνωρίζεται πολύ καλά στη βιβλιογραφία. Στην πραγματικότητα, αν μια εργασία σχετικά με τη βελτιστοποίηση σημείων πλέγματος υποβληθεί στο *Journal of Combinatorial Optimization*, τότε συνήθως, δεν θα απορριφθεί λόγω του ότι είναι εκτός πεδίου εφαρμογής.

Όσον αφορά τις μεθοδολογίες, η συνδυαστική βελτιστοποίηση και η διακριτή βελτιστοποίηση έχουν πολύ κοντινή σχέση. Για παράδειγμα, για να αποδείξουμε την NP-δυσκολία του ακέραιου προγραμματισμού, πρέπει να κόψουμε το απείρως μεγάλο εφικτό πεδίο του σε ένα πεπερασμένο υπο-

σύνολο που περιέχει τη βέλτιστη λύση (βλ. Κεφάλαιο 8 για λεπτομέρειες), δηλαδή να το μετατρέψουμε σε πρόβλημα συνδυαστικής βελτιστοποίησης.

Η γεωμετρική βελτιστοποίηση είναι ένα άλλο παράδειγμα. Σκεφτείτε το ακόλουθο πρόβλημα.

Πρόβλημα 1.1.2 (Ελάχιστο μήκος χωρίσματος γκιλοτίνας)**.** *Δεδομένου ενός ορθογωνίου με οπές στο εσωτερικό του, χωρίστε το σε μικρότερα ορθογώνια χωρίς οπές στο εσωτερικό του με μια ακολουθία κοπών γκιλοτίνας ώστε να ελαχιστοποιηθεί το συνολικό μήκος των κοπών. Εδώ, η κοπή γκιλοτίνας είναι ένα κατακόρυφο ή οριζόντιο ευθύγραμμο τμήμα που χωρίζει ένα ορθογώνιο σε δύο μικρότερα ορθογώνια. Ένα παράδειγμα φαίνεται στο σχήμα 1.2.*



Σχήμα 1.2: Ορθογώνιο χώρισμα γκιλοτίνας.

Υπάρχει άπειρος αριθμός κατατμήσεων. Επομένως, δεν είναι ένα πρόβλημα συνδυαστικής βελτιστοποίησης. Ωστόσο, μπορούμε να αποδείξουμε ότι η βέλτιστη κατάτμηση μπορεί να βρεθεί από ένα πεπερασμένο σύνολο κατατμήσεων ειδικού τύπου (για λεπτομέρειες, βλ. Κεφάλαιο 3). Συνεπώς, για την επίλυση του προβλήματος, χρειάζεται να μελετήσουμε μόνο κατατμήσεις αυτού του ειδικού τύπου, δηλ, ένα πρόβλημα συνδυαστικής βελτιστοποίησης.

Λόγω των παραπάνω, δεν κάνουμε ένα διαχωρισμό για να αποκλείσουμε κάποια μέρη της διακριτής βελτιστοποίησης. Στην πραγματικότητα, αυτό το βιβλίο είναι προσανατολισμένο στη μεθοδολογία. Τα προβλήματα επιλέγονται για να απεικονίσουν τη μεθοδολογία. Ειδικότερα, για κάθε μέθοδο, μπορούμε να επιλέξουμε ένα τυπικό πρόβλημα ως συνοδευτικό για να διερευνήσουμε τη μέθοδο, τις απαιτήσεις της και τις εφαρμογές της.

Για παράδειγμα, χρησιμοποιούμε το πρόβλημα της συντομότερης διαδρομής για να εξηγήσουμε τον δυναμικό προγραμματισμό, χρησιμοποιούμε το πρόβλημα του ελάχιστου τανυόντος δέντρου για να εξηγήσουμε τους άπληστους αλγορίθμους, κ.λπ.

1.2 Βέλτιστη και Προσεγγιστική Λύση

Ας δείξουμε μια συνθήκη βελτιστότητας για το ελάχιστο τανύον δένδρο.

Θεώρημα 1.2.1 (Βελτιστότητα Μονοπατιών)**.** *Ένα τανύον δέντρο T^* είναι ένα ελάχιστο τανύον δέντρο αν και μόνο αν ικανοποιεί την ακόλουθη συνθήκη:*

Συνθήκη Βελτιστότητας Μονοπατιών *Για κάθε ακμή (u, v) που δεν ανήκει στο T^*, υπάρχει ένα μονοπάτι p στο T^*, που συνδέει τις u και v, και επιπλέον, $c(u, v) \geq c(x, y)$ για κάθε ακμή (x, y) στο μονοπάτι p.*

Απόδειξη. Ας υποθέσουμε, για αντίφαση, ότι $c(u, v) < c(x, y)$ για κάποια ακμή (x, y). στο μονοπάτι p. Τότε $T' = (T^* \setminus (x, y)) \cup (u, v)$ είναι ένα τανύον δένδρο με κόστος μικρότερο από $c(T^*)$, γεγονός που αντικρούει την ελαχιστότητα του T^*. Αντιστρόφως, υποθέστε ότι T^* ικανοποιεί τη συνθήκη βελτιστότητας διαδρομών.

Έστω T' ένα ελάχιστο τανύον δένδρο έτσι ώστε μεταξύ όλων των ελάχιστων τανυόντων δέντρων, το T' είναι αυτό με τις περισσότερες κοινές ακμές με το T^*. Ας υποθέσουμε, για αντίφαση, ότι $T' \neq T^*$. Ισχυριζόμαστε ότι υπάρχει μια ακμή (u, v) $sto T^*$ έτσι ώστε το μονοπάτι στο T' μεταξύ u και v να περιέχει μια ακμή (x, y) με μήκος $c(x, y) \geq c(u, v)$. Αν αυτός ο ισχυρισμός είναι αληθής, τότε $(T' \setminus (x, y)) \cup (u, v)$ εξακολουθεί να είναι ένα ελάχιστο τανύον δένδρο, σε αντίθεση με τον ορισμό του T'.

Τώρα, αποδεικνύουμε τον ισχυρισμό με αντίφαση. Ας υποθέσουμε ότι ο ισχυρισμός δεν είναι αληθής. Θεωρήστε μια ακμή $(u_1, v_1) \in T^* \setminus T'$. Το μονοπάτι στο T' που συνδέει τα u_1 και v_1 πρέπει να περιέχει μια ακμή (x_1, y_1) που δεν βρίσκεται στο T^*. Εφόσον ο ισχυρισμός δεν είναι αληθής, έχουμε $c(u_1, v_1) < c(x_1, y_1)$. Ακολούθως, εξετάζουμε το μονοπάτι στο T^* που συνδέει τις x_1 και y_1, το οποίο πρέπει να περιέχει μια ακμή $(u_2, v_2) \notin T'$. Εφόσον το T^* ικανοποιεί τη συνθήκη βελτιστότητας διαδρομών, έχουμε $c(x_1, y_1) \leq c(u_2, v_2)$. Συνεπώς, $c(u_1, v_1) < c(u_2, v_2)$. Καθώς

το επιχείρημα αυτό συνεχίζεται, θα βρούμε μια ακολουθία ακμών στο T^* έτσι ώστε $c(u_1, v_2) < c(u_2, v_2) < c(u_3, v_3) < \cdots$, γεγονός που αντιφάσκει το πεπερασμένο του T^*. □

Ένας αλγόριθμος μπορεί να σχεδιαστεί με βάση τη συνθήκη βελτιστότητας διαδρομών.

Αλγόριθμος Kruskal
input: Ένα συννεκτικό γράφημα $G = (V, E)$ με μη αρνητικό βάρος τόξων $c : E \to R_+$.
output: Ένα ελάχιστο τανύον δένδρο T.
Ταξινόμησε όλα τα τόξα $e_1, e_2, ..., e_m$ σε αύξουσα σειρά με βάση τα βάρη, i.e., $c(e_1) \leq c(e_2) \leq \cdots \leq c(e_m)$·
$T \leftarrow \emptyset$·
for $i \leftarrow 1$ **to** m **do**
  if $T \cup e_i$ δεν περιέχει κύκλο
    then $T \leftarrow T \cup e_i$·
return T.

Από αυτόν τον αλγόριθμο, βλέπουμε ότι δεν είναι δύσκολο να βρούμε τη βέλτιστη λύση για το πρόβλημα του ελάχιστου τανύοντος δέντρου. Αν κάθε πρόβλημα συνδυαστικής βελτιστοποίησης μοιάζει με το πρόβλημα του ελάχιστου τανύοντος δένδρου, τότε θα ήμασταν πολύ ευτυχείς να βρούμε τη βέλτιστη λύση για αυτό. Δυστυχώς, υπάρχει ένας μεγάλος αριθμός προβλημάτων που είναι απίθανο να μπορέσουμε να υπολογίσουμε αποτελεσματικά τη βέλτιστη λύση τους. Για παράδειγμα, θεωρήστε το ακόλουθο πρόβλημα.

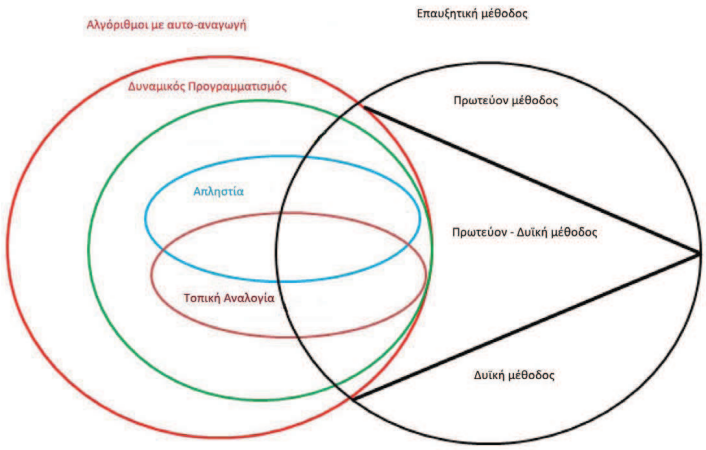
Πρόβλημα 1.2.2 (Πρόβλημα ορθογώνιου χωρίσματος ελαχίστου μήκους)**.** *Δεδομένου ενός ορθογωνίου με οπές στο εσωτερικό του, χωρίστε το σε μικρότερα ορθογώνια χωρίς τρύπες ώστε να ελαχιστοποιηθεί το συνολικό μήκος των κοπών.*

Τα προβλήματα 1.1.2 και 1.2.2 είναι αρκετά διαφορετικά. Το πρόβλημα 1.2.2 είναι δύσκολα επιλύσιμο, ενώ υπάρχει ένας αποτελεσματικός αλγόριθμος για τον υπολογισμό μιας βέλτιστης λύσης για το πρόβλημα 1.1.2. Στην πραγματικότητα, στη θεωρία της συνδυαστικής βελτιστοποίησης, πρέπει να μελετήσουμε όχι μόνο πώς να σχεδιάζουμε και να αναλύουμε αλγορίθμους για την εύρεση βέλτιστων λύσεων, αλλά και πώς να σχεδιάζουμε και να αναλύουμε αλγορίθμους για τον υπολογισμό προσεγγιστικών λύσεων. Πότε βάζουμε τις προσπάθειές μας στην εύρεση της βέλτιστης λύσης; Πότε πρέπει

να να δώσουμε προσοχή στην εύρεση προσεγγιστικών λύσεων; Η ικανότητα για τη λήψη μιας τέτοιας απόφασης αναπτύσσεται από τη μελέτη της υπολογιστικής πολυπλοκότητας.

Το βιβλίο αποτελείται από τρία δομικά σύνολα, το πρώτο είναι ο σχεδιασμός και η ανάλυση υπολογιστικών αλγορίθμων για ακριβή βέλτιστη λύση, το δεύτερο ο σχεδιασμός και η ανάλυση προσεγγιστικών αλγορίθμων και το τρίτο η μη γραμμική συνδυαστική βελτιστοποίηση.

Το πρώτο σύνολο περιλαμβάνει τα κεφάλαια 2-7, τα οποία μπορούν να χωριστούν σε δύο μέρη (Εικ.1.3).

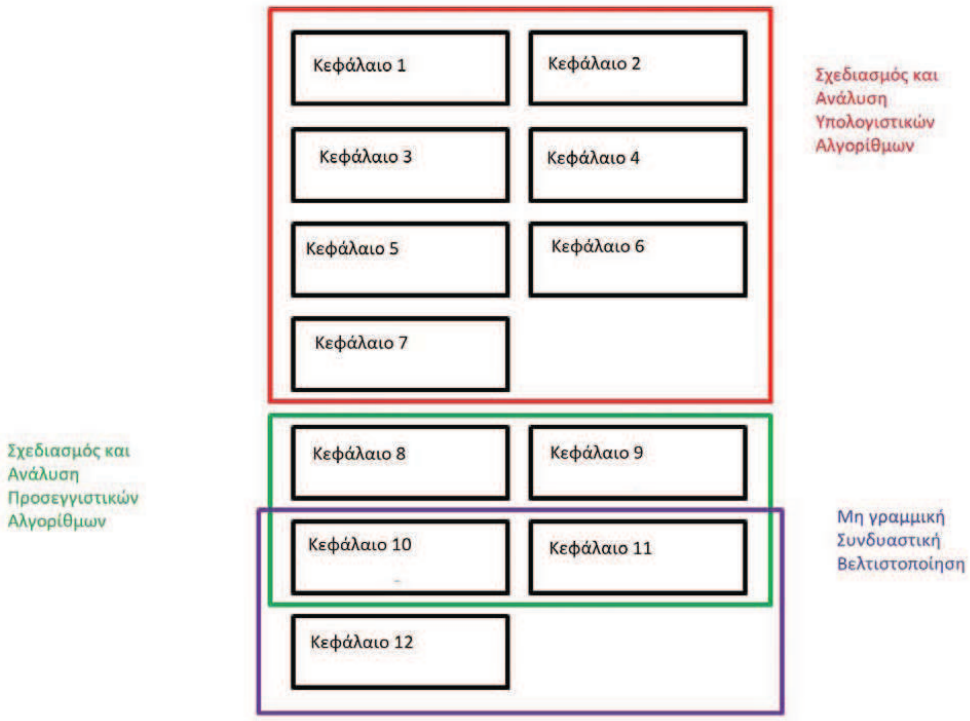


Σχήμα 1.3: Σχεδιασμός και Ανάλυση Υπολογιστικών Αλγορίθμων.

Το πρώτο μέρος αφορά τους αλγορίθμους με αυτοαναγωγιμότητα, όπως ο αλγόριθμος διαίρει και βασίλευε, ο δυναμικός προγραμματισμός, ο αλγόριθμος απληστίας, η τοπική αναζήτηση, η τοπική αναλογία, κ.λπ., τα οποία οργανώνονται στα κεφάλαια 2-4. Το δεύτερο μέρος αφορά την επαυξητική μέθοδο, συμπεριλαμβανομένου του πρωτεύοντος αλγορίθμου, του δυϊκού αλγορίθμου και του πρωτεύοντος-δυϊκού αλγορίθμου, τα οποία παρουσιάζονται επίσης στα κεφάλαια 5-7. Υπάρχει μια τομή μεταξύ αλγορίθμων με αυτοαναγωγιμότητα και των πρωτεύοντων-δυϊκών αλγορίθμων. Στην πραγματικότητα, κατά τη διαδικασία υπολογισμού τις πρώτης ομάδας αλγορίθμων, μια βέλτιστη εφικτή λύση χτίζεται βήμα προς βήμα με βάση ορισμένες τεχνικές, ενώ η δεύτερη ομάδα αλγορίθμων έχει επίσης

μια διαδικασία για τη δημιουργία μια βέλτιστης πρωτεύουσας λύσης με τη χρήση πληροφοριών από τη δυϊκή πλευρά. Επομένως, μπορεί να απεικονιστεί κάποιος αλγόριθμος ως αλγόριθμος με αυτοαναγωγιμότητα, και εν τω μεταξύ μπορεί επίσης να εξηγηθεί ως αλγόριθμος πρωτεύον-δυϊκός.

Το δεύτερο σύνολο περιλαμβάνει τα κεφάλαια 8-11, που καλύπτουν τις θεμελιώδεις γνώσεις σχετικά με την υπολογιστική πολυπλοκότητα, συμπεριλαμβανομένης της θεωρίας για την ΝΠ-δυσκολία και τη μη προσεγγισιμότητα, καθώς και βασικές τεχνικές για το σχεδιασμό προσεγγιστικών μεθόδων συμπεριλαμβανομένου των περιορισμών, της άπληστης προσέγγισης και της χαλάρωσης με στρογγυλοποίηση.



Σχήμα 1.4: Δομή του βιβλίου.

Το τρίτο σύνολο περιλαμβάνει τα κεφάλαια 10, 11, 12. Δεδομένου ότι τα κεφάλαια 10-11 εξυπηρετούν τόσο το δεύτερο όσο και το τρίτο σύνολο, τα επιλεγμένα παραδείγματα προέρχονται κυρίως από την υποαρθρωτή βελτιστοποίηση. Στη συνέχεια, το κεφάλαιο 12 συμβάλλει στην μη υποαρθρωτή

βελτιστοποίηση. Η μη υποαρθρωτή βελτιστοποίηση αποτελεί επί του παρόντος ενεργό ερευνητικό πεδίο. Υπάρχουν πολλές πρόσφατες δημοσιεύσεις στη βιβλιογραφία. Πιθανώς, το Κεφάλαιο 12 μπορεί να θεωρηθεί ως εισαγωγή σε αυτή την περιοχή. Για μια πλήρη κάλυψη, ίσως χρειαστούμε ένα νέο βιβλίο.

Τώρα, τοποθετούμε την παραπάνω δομή αυτού του βιβλίου στο Σχήμα 1.4 για μια σαφή επισκόπηση.

1.3 Προεπεξεργασία

Στον αλγόριθμο Kruskal, η πρώτη γραμμή είναι η ταξινόμηση όλων των ακμών σε μια μη φθίνουσα σειρά κόστους. Αυτό απαιτεί μια διαδικασία προεπεξεργασίας για την επίλυση του προβλήματος ταξινόμησης ως εξής.

Πρόβλημα 1.3.1 (Ταξινόμηση)**.** *Δεδομένης μιας ακολουθίας θετικών ακεραίων αριθμών, ταξινομήστε τους σε μη φθίνουσα σειρά.*

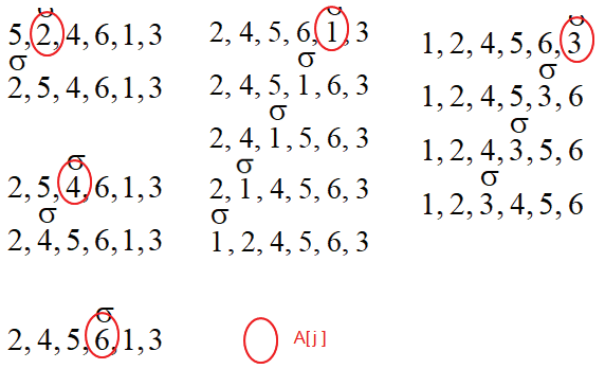
Ακολουθεί ένας απλός αλγόριθμος για την ταξινόμηση.

Ταξινόμηση με Εισαγωγή
input: Ένα διάνυσμα A με μια ακολουθία θετικών ακεραίων αριθμών.
output: Ένα διάνυσμα A με μια ακολουθία θετικών ακεραίων αριθμών σε μη φθίνουσα σειρά.

for $j \leftarrow 2$ **to** length[A]
  do $key \leftarrow A[j]$
    $i \leftarrow j - 1$
    while $i > 0$ and $A[i] > key$
      do $A[i+1] \leftarrow A[i]$
        $i \leftarrow i - 1$
    end-while
    $A[i+1] \leftarrow key$
end-for
return A.

Ένα παράδειγμα χρήσης του αλγορίθμου ταξινόμησης με εισαγωγή φαίνεται στο Σχήμα 1.5.

Αν και η Ταξινόμηση με Εισαγωγή είναι απλούστερη, εκτελείται λίγο αργά. Δεδομένου ότι η ταξινόμηση εμφανίζεται πολύ συχνά στο σχεδιασμό



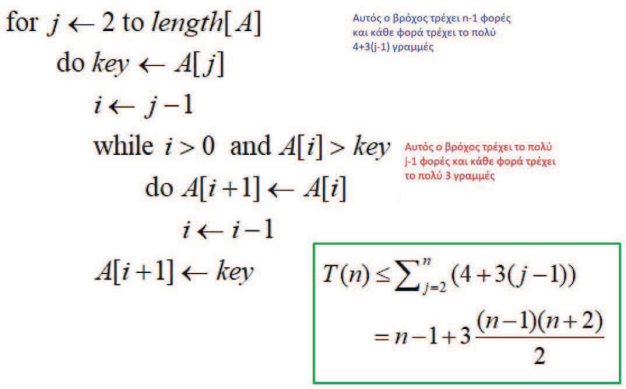
Σχήμα 1.5: Ένα παράδειγμα του αλγορίθμου ταξινόμησης με εισαγωγή. Το σ είναι το key που βρίσκεται εκτός του διανύσματος A.

αλγορίθμων για προβλήματα συνδυαστικής βελτιστοποίησης, πρέπει να αφιερώσουμε λίγο χώρο στο Κεφάλαιο 1 για να παρουσιάσουμε ταχύτερους αλγορίθμους.

1.4 Χρόνος Εκτέλεσης

Το σημαντικότερο μέτρο ποιότητας για τους αλγορίθμους είναι ο χρόνος εκτέλεσης. Ωστόσο, για τον ίδιο αλγόριθμο, μπορεί να χρειαστούν διαφορετικοί χρόνοι όταν τον εκτελούμε σε διαφορετικούς υπολογιστές. Για να δώσουμε ένα ενιαίο πρότυπο, πρέπει να συμφωνήσουμε ότι τρέχουμε τους αλγορίθμους σε ένα θεωρητικό μοντέλο υπολογιστή. Αυτό το μοντέλο είναι η μηχανή Turing πολλαπλών ταινιών που έχει γίνει αποδεκτό από ένα πολύ μεγάλο μέρος του πληθυσμού. Με βάση τη μηχανή Turing, έχει δημιουργηθεί η θεωρία της υπολογιστικής πολυπλοκότητας. Θα αγγίξουμε αυτό το μέρος της θεωρίας στο κεφάλαιο 8.

Όμως, θα χρησιμοποιήσουμε το μοντέλο RAM για να αξιολογήσουμε το χρόνο εκτέλεσης των αλγορίθμων σε όλο το βιβλίο, εκτός από το κεφάλαιο 8. Στο μοντέλο RAM, υποθέστε ότι κάθε γραμμή ψευδοκώδικα απαιτεί σταθερό χρόνο. Για παράδειγμα, ο χρόνος εκτέλεσης της Ταξινόμησης με Εισαγωγή υπολογίζεται στο Σχήμα 1.6.



Σχήμα 1.6: Υπολογισμός χρόνου εκτέλεσης.

Στην πραγματικότητα, το μοντέλο RAM και το μοντέλο μηχανής Turing είναι στενά συνδεδεμένα. Ο χρόνος εκτέλεσης που εκτιμάται με βάση αυτά τα δύο μοντέλα θεωρείται αρκετά κοντινός. Ωστόσο, μερικές φορές διαφέρουν ως προς την εκτίμηση του χρόνου εκτέλεσης. Για παράδειγμα, το παρακάτω είναι ένα κομμάτι ψευδοκώδικα.

for $i = 1$ to n
do ανάθεσε $First(i) \leftarrow i$
end-for

Σύμφωνα με το μοντέλο RAM, ο χρόνος εκτέλεσης αυτού του κομματιού είναι $O(n)$. Ωστόσο, με βάση τη μηχανή Turing, ο χρόνος εκτέλεσης αυτού του κομματιού είναι $O(n \log n)$ επειδή η τιμή που ανατίθεται πρέπει να αναπαρασταθεί από μια συμβολοσειρά με $O(\log n)$ σύμβολα.

Θεωρητικά, ένας σταθερός παράγοντας συχνά αγνοείται. Για παράδειγμα, συνήθως λέμε ότι ο χρόνος εκτέλεσης της Ταξινόμησης με Εισαγωγή είναι $O(n^2)$ αντί να δώσουμε τη συγκεκριμένη τετραγωνική συνάρτηση ως προς n. Εδώ το $f(n) = O(g(n))$ σημαίνει ότι υπάρχουν σταθερές $c > 0$ και $n_0 > 0$ τέτοιες ώστε

$$f(n) \leq c \cdot g(n) \text{ για } n \geq n_0$$

Υπάρχουν δύο ακόμη συμβολισμοί που εμφανίζονται πολύ συχνά στην αναπαράσταση του χρόνου εκτέλεσης. Το $f(n) = \Omega(g(n))$ σημαίνει ότι υπάρχει σταθερά $c > 0$ και $n_0 > 0$ τέτοια ώστε

$$0 \leq c \cdot g(n) \leq f(n) \text{ για } n \geq n_0.$$

Το $f(n) = \Theta(g(n))$ σημαίνει ότι υπάρχουν σταθερές $c_1 > 0$, $c_2 > 0$ και $n_0 > 0$ τέτοιες ώστε

$$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \text{ για } n \geq n_0.$$

Τέλος, ας κάνουμε μια παρατήρηση σχετικά με τους αριθμούς εισόδου.

Στο πρόβλημα του ελάχιστου τανύοντος δέντρου, κάθε τόξο έχει ένα μη αρνητικό βάρος, το οποίο είναι ένας αριθμός εισόδου. Στον ορισμό του προβλήματος, υποθέσαμε ότι πρόκειται για πραγματικό αριθμό. Ωστόσο, ένας πραγματικός αριθμός δεν μπορεί να εισαχθεί ακριβώς σε έναν υπολογιστή. Στην πραγματικότητα, ο υπολογισμός ενός πραγματικού αριθμού είναι μερικές φορές ένα πολύ δύσκολο πρόβλημα στη θεωρία της υπολογιστικής πολυπλοκότητας [260]. Επομένως, πρέπει να αντιμετωπίσουμε κάθε πραγματικό αριθμό με ένα χρησμό που μπορεί να παρέχει έναν ορθολογικό αριθμό με αναμενόμενη ακρίβεια, χωρίς υπολογιστικό κόστος. Με άλλα λόγια, αγνοούμε το πρόβλημα υπολογισμού του πραγματικού αριθμού.

Ωστόσο, όταν η ανάλυση του χρόνου εκτέλεσης πρέπει να λαμβάνει υπόψη το μέγεθος των αριθμών εισόδου, π.χ., στην ανάλυση αλγορίθμων ασθενώς πολυωνυμικού χρόνου, πρέπει να αλλάξουμε τη ρύθμιση από πραγματικό αριθμό σε ακέραιο αριθμό.

1.5 Δομές Δεδομένων

Μια δομή δεδομένων είναι μια μορφή αποθήκευσης δεδομένων η οποία οργανώνεται και διαχειρίζεται για να έχει αποτελεσματική πρόσβαση και τροποποίηση. Κάθε δομή δεδομένων έχει διάφορες τυποποιημένες λειτουργίες. Αποτελούν δομικά στοιχεία για την κατασκευή αλγορίθμων. Η δομή δεδομένων παίζει σημαντικό ρόλο στη βελτίωση της αποτελεσματικότητας των αλγορίθμων. Για παράδειγμα, μπορούμε να εισαγάγουμε μια δομή δεδομένων 'Διαχωριστικών Συνόλων' για να να βελτιώσουμε τον αλγόριθμο Kruskal.

Θεωρήστε μια συλλογή από διαχωριστική σύνολα. Για κάθε σύνολο S, έστω ότι το $First(S)$ είναι ο πρώτος κόμβος στο σύνολο S. Για κάθε στοιχείο x του συνόλου S, συμβολίζουμε $First(x) = First(S)$. Ορίστε τρεις πράξεις ως εξής.

$Make - Set(x)$ δημιουργεί ένα καινούριο σύνολο που περιέχει μόνο το x.

$Union(x, y)$ ενώνει τα σύνολα S_x και S_y που περιέχουν τα στοιχεία x και y, αντίστοιχα, στο $S_x \cup S_y$, Επιπλέον, θέτουμε

$$First(S_x \cup S_y) = \begin{cases} First(S_x) & \text{εάν } |S_x| \geq |S_y|, \\ First(S_y) & \text{αλλιώς.} \end{cases}$$

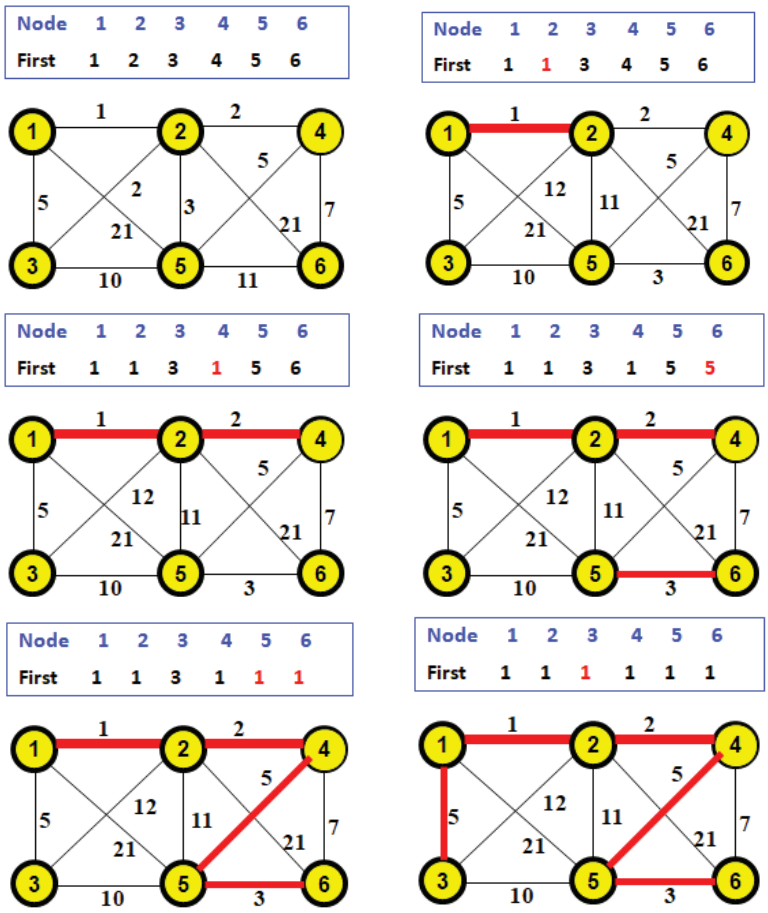
$Find - Set(x)$ επιστρέφει το $First(S_x)$ όπου S_x είναι ένα σύνολο που περιέχει το x.

Με αυτή τη δομή δεδομένων, ο αλγόριθμος Kruskal μπορεί να τροποποιηθεί ως εξής.

Αλγόριθμος Kruskal
input: Ένα συνεκτικό γράφημα $G = (V, E)$ με μη αρνητικά βάρη στα τόξα $c : E \to R_+$.
output: Ένα ελάχιστο τανύον δένδρο T.
Ταξινόμησε όλα τα τόξα $e_1, e_2, ..., e_m$ σε μη φθίνουσα σειρά βάση των βαρών, π.χ., $c(e_1) \leq c(e_2) \leq \cdots \leq c(e_m)$;
$T \leftarrow \emptyset$;
for κάθε κόμβο $v \in V$ **do**
    $Make - Set(v)$;
end-for
for $i \leftarrow 1$ **to** m **do**
    if $Find - Set(x) \neq Find - Set(y)$ όπου $e_i = (x, y)$
        then $T \leftarrow T \cup e_i$
            και $Union(x, y)$;
end-for
return T.

Ένα παράδειγμα εκτέλεσης αυτού του αλγορίθμου φαίνεται στο Σχήμα 1.7.

Συμβολίζουμε με $m = |E|$ και με $n = |V|$. Ας εκτιμήσουμε τον χρόνο εκτέλεσης του αλγορίθμου Kruskal.



Σχήμα 1.7: Παράδειγμα του αλγορίθμου Kruskal.

- Η ταξινόμηση σε όλα τα τόξα απαιτεί χρόνο $O(m \log n)$.
- Η ανάθεση $First(v)$ για όλα τα $v \in V$ διαρκεί $O(n)$ χρόνο.
- Για κάθε κόμβο v, η τιμή του $First(v)$ μπορεί να αλλάξει το πολύ $O(\log n)$ χρόνο. Αυτό συμβαίνει επειδή η τιμή του $First(v)$ αλλάζει μόνο αν ο v περιλαμβάνεται στην πράξη $Union$. και μετά την πράξη, το σύνολο που περιέχει το v έχει διπλασιαστεί σε μέγεθος.
- Έτσι, ο δεύτερος βρόχος (for-loop) διαρκεί $O(n \log n)$ χρόνο.
- Συνολικά, ο χρόνος εκτέλεσης είναι $O(m \log n) = O(m \log n + n \log n)$.

Ασκήσεις

1. Σε μια πόλη υπάρχουν N σπίτια, καθένα από τα οποία έχει ανάγκη από παροχή νερού. Η κατασκευή ενός πηγαδιού στο σπίτι i κοστίζει W_i δολάρια, και κοστίζει C_{ij} για να κατασκευαστεί ένας σωλήνας μεταξύ των σπιτιών i και j. Ένα σπίτι μπορεί να λάβει νερό εάν είτε υπάρχει ένα πηγάδι που έχει κατασκευαστεί εκεί είτε υπάρχει κάποια διαδρομή σωλήνων προς ένα σπίτι με πηγάδι. Δώστε έναν αλγόριθμο για να βρείτε το ελάχιστο ποσό των χρημάτων που απαιτούνται για να προμηθεύσουμε κάθε σπίτι με νερό.

2. Θεωρήστε ένα συνεκτικό γράφημα G με όλα τα διαφορετικά βάρη ακμών. Δείξτε ότι το ελάχιστο τανύον δένδρο του G είναι μοναδικό.

3. Θεωρήστε ένα συνεκτικό γράφημα $G = (V, E)$ με μη αρνητικό βάρος τόξου $c : E \to R_+$. Ας υποθέσουμε ότι $e_1^*, e_2^*, ..., e_k^*$ είναι τόξα που παράγονται από τον αλγόριθμο Kruskal, και $e_1, e_2, ..., e_k$ είναι τόξα ενός δέντρου με διάταξη $c(e_1) \leq c(e_2) \leq \cdots \leq c(e_k)$. Δείξτε ότι $c(e_i^*) \leq c(e_i)$ για όλα τα $1 \leq i \leq k$.

4. Έστω V ένα σταθερό σύνολο κορυφών n. Θεωρήστε μια ακολουθία m μη προσανατολισμένων τόξων e_1, e_2, ..., e_m. Για $1 \leq i \leq m$, έστω G_i το γράφημα με σύνολο κορυφών V και σύνολο τόξων $E_i = \{e_1, ..., e_i\}$. Έστω c_i συμβολίζει τον αριθμό των συνδεδεμένων συνιστωσών του G_i. Σχεδιάστε έναν αλγόριθμο για τον υπολογισμό του c_i για όλα τα i. Ο αλγόριθμός σας θα πρέπει να είναι ασυμπτωτικά όσο το δυνατόν πιο γρήγορος. Ποιος είναι ο χρόνος εκτέλεσης του αλγορίθμου σας;

5. Υπάρχουν n σημεία που βρίσκονται στο Ευκλείδειο επίπεδο. Δείξτε ότι υπάρχει ένα ελάχιστο τανύον δένδρο σε αυτά τα n σημεία έτσι ώστε κάθε κόμβος (δηλ. σημείο) να έχει βαθμό το πολύ πέντε.

6. Μπορείτε να τροποποιήσετε τον αλγόριθμο Kruskal για να υπολογίσετε ένα μέγιστου βάρους τανύον δέντρου;

7. Θεωρήστε ένα συνεκτικό γράφημα $G = (V, E)$ με βάρος τόξου $c : E \to R$, δηλαδή το βάρος είναι ενδεχομένως αρνητικό. Λειτουργεί ο αλγόριθμος Kruskal για τον υπολογισμό ενός ελάχιστου βάρους τανύοντος δέντρου;

8. Θεωρήστε ένα συνεκτικό γράφημα $G = (V, E)$ με μη αρνητικό βάρος τόξου $c : E \to R_+$. Ας υποθέσουμε ότι το τόξο e είναι το μοναδικό μακρύτερο τόξο σε έναν κύκλο. Δείξτε ότι το e δεν μπορεί να συμπεριληφθεί σε οποιοδήποτε ελάχιστο τανύον δένδρο.

9. Θεωρήστε ένα συνεκτικό γράφημα $G = (V, E)$ με μη αρνητικό βάρος τόξου $c : E \to R_+$. Όσο υπάρχει κύκλος, διαγράψτε ένα μακρύτερο τόξο από τον κύκλο. Δείξτε ότι ο υπολογισμός αυτός τελειώνει σε ένα ελάχιστο τανύον δέντρο.

10. Θεωρήστε ένα παίγνιο με δύο παίκτες σε ένα μη προσανατολισμένο δεδομένο γράφημα G. Οι δύο παίκτες παίρνουν σειρά εναλλάξ. Ο πρώτος παίκτης αφαιρεί ένα τόξο της επιλογής του/της. Στη συνέχεια, ο δεύτερος παίκτης προσθέτει ένα τόξο του/της επιλογής του. Ο κανόνας είναι ότι κανένα αφαιρούμενο τόξο δεν μπορεί να προστεθεί ξανά και κανένα προστιθέμενο τόξο δεν μπορεί να αφαιρεθεί. Ο πρώτος παίκτης κερδίζει αν καταφέρει να αποσυνδέσει το γράφημα. Ο δεύτερος παίκτης κερδίζει αν καταφέρει να έχει ένα τανύον δένδρο στο γράφημα. Αποδείξτε ότι ο δεύτερος παίκτης έχει μία νικητήρια στρατηγική αν και μόνο αν το G περιέχει ένα τανύον δέντρο με δύο τόξα που δεν ενώνονται μεταξύ τους.

Ιστορικές Σημειώσεις

Υπάρχουν πολλά βιβλία που έχουν γραφτεί για τη συνδυαστική βελτιστοποίηση [72, 272, 275, 336, 359, 105, 368, 369]. Υπάρχουν επίσης πολλά

βιβλία που έχουν δημοσιευτεί για το σχεδιασμό και την ανάλυση υπολογιστικών αλγορίθμων [73, 280], τα οποία καλύπτουν μεγάλο μέρος των προβλημάτων συνδυαστικής βελτιστοποίησης. Ωστόσο, αυτά τα βιβλία αναφέρονται κυρίως στον υπολογισμό ακριβών βέλτιστων λύσεων και ενδεχομένως ένα μικρό μέρος αναφέρεται σε προσεγγιστικές λύσεις. Για τις προσεγγιστικές λύσεις, ένα μεγάλο μέρος του υλικού καλύπτεται συνήθως σε ξεχωριστά βιβλία [388, 409, 100]. Για θέματα σχετικά με την υπολογιστική πολυπλοκότητα, ο αναγνώστης μπορεί να ανατρέξει στο [260, 99].

Στις πρόσφατες εξελίξεις της τεχνολογίας, η συνδυαστική βελτιστοποίηση αποκτά πολλές νέες εφαρμογές και νέες ερευνητικές κατευθύνσεις [339, 426, 423, 338]. Σε αυτό το βιβλίο, προσπαθούμε να ικανοποιήσουμε αιτήματα από διάφορους τομείς για διδασκαλία, έρευνα, και αναφορές, να συγκεντρώσουμε τρία στοιχεία, το κλασικό μέρος της συνδυαστικής βελτιστοποίησης, την προσέγγιστική θεωρία που αναπτύχθηκε τα τελευταία χρόνια, και τη νεοεμφανιζόμενη μη γραμμική συνδυαστική βελτιστοποίηση.