P, NP, NP-Complete Problems

P, N, NP-Complete Problems (1/24)

Λέμε πως ένας αλγόριθμος επιλύει ένα πρόβλημα σε πολυωνυμικό χρόνο όταν στη χειρότερη περίπτωση η πολυπλοκότητά του είναι **O(p(n))** όπου p(n) είναι ένα πολυώνυμο του μεγέθους εισόδου n

Τα προβλήματα που μπορούν να επιλυθούν σε πολυωνυμικό χρόνο ονομάζονται **ανιχνεύσιμα (tractable)**

Τα προβλήματα που δεν μπορούν να επιλυθούν σε πολυωνυμικό χρόνο ονομάζονται **μη ανιχνεύσιμα (intractable)**

Η επίλυση των μη ανιχνεύσιμων προβλημάτων δεν μπορούν να επιλυθούν σε ένα λογικό πλαίσιο χρόνου εκτός αν η έκδοση που θέλουμε να επιλύσουμε είναι μικρού μεγέθους

P, N, NP-Complete Problems (2/24)

Σχετικά με τα πολυώνυμα που περιγράφουν την πολυπλοκότητα των ανιχνεύσιμων προβλημάτων, υπάρχει ένας πολύ μεγάλος αριθμός πολυωνύμων αλλά πρακτικά συναντάμε **μέχρι τρίτου βαθμού**

Συνήθως, τα πολυώνυμα που περιγράφουν χρόνους εκτέλεσης **δεν έχουν υψηλές τιμές στους συντελεστές** των όρων

Τα πολυώνυμα έχουν κάποιες ωφέλιμες ιδιότητες όπως η **σύνθεση ή το άθροισμά τους που είναι πολυώνυμο** επίσης

Η επιλογή των πολυωνύμων οδήγησε στην εξέλιξη **της υπολογιστικής πολυπλοκότητας (computational complexity)** που ψάχνει να ταξινομήσει τα προβλήματα σύμφωνα με τη δυσκολία τους

P, N, NP-Complete Problems (3/24)

Τα περισσότερα προβλήματα που έχουμε δει είναι πολυωνυμικού χρόνου

Τα προβλήματα που μπορεί να επιλυθούν σε πολυωνυμικό χρόνο τα θεωρούμε μέρος ενός συνόλου το οποίο συμβολίζουμε με **P**

Η τάξη P είναι μια τάξη προβλημάτων που μπορεί να επιλυθούν σε πολυωνυμικό χρόνο από ένα (ντετερμινιστικό) αλγόριθμο. Η κλάση αυτή ονομάζεται πολυωνυμική.

P, N, NP-Complete Problems (4/24)

Ο ορισμός στοχεύει κυρίως σε προβλήματα απόφασης (η λύση είναι ένα ναι ή όχι) για τους ακόλουθους λόγους

- Είναι λογικό να αποκλείουμε προβλήματα που δεν επιλύονται σε πολυωνυμικό χρόνο **λόγω του εκθετικού μεγέθους της εξόδου** – παράδειγμα: η δημιουργία υποσυνόλων ενός σύνολου μέσω του υπολογισμού των αναδιατάξεων η αντικειμένων
- Πολλά σημαντικά προβλήματα δεν είναι προβλήματα απόφασης αλλά **μπορεί να αναχθούν σε μια σειρά προβλημάτων απόφασης** που μπορούν να μελετηθούν πιο εύκολα

P, N, NP-Complete Problems (5/24)

Δυστυχώς, όλα τα προβλήματα απόφασης δεν μπορούν να επιλυθούν σε πολυωνυμικό χρόνο

Στην πραγματικότητα κάποια προβλήματα δεν μπορούν να λυθούν από κάποιον αλγόριθμο

Τα προβλήματα απόφασης διακρίνονται σε: **αποφασίσιμα (decidable)** και **μη αποφασίσιμα (undecidable)** ανάλογα με το αν λύνονται από κάποιο αλγόριθμο ή όχι

P, N, NP-Complete Problems (6/24)

Παράδειγμα μη αποφασίσιμου προβλήματος

- The **halting problem by Alan Turing**
- Δοσμένου ενός προγράμματος και μιας εισόδου για αυτό, να καθορίσουμε αν το πρόγραμμα θα σταματήσει κάποτε για τη συγκεκριμένη είσοδο ή θα συνεχίσει να εκτελείται επ' άπειρον
- Ας υποθέσουμε ότι όντως υπάρχει ένας αλγόριθμος που να το λύνει
- Συνεπώς, για κάθε πρόγραμμα P και είσοδο I έχουμε:

$$A(P, I) = \begin{cases} 1, & \text{if program } P \text{ halts on input } I \\ 0, & \text{if program } P \text{ does not halt on input } I \end{cases}$$

P, N, NP-Complete Problems (7/24)

- Μπορούμε να φανταστούμε το πρόγραμμα σαν είσοδο στον εαυτό του και χρησιμοποιούμε την έξοδο του αλγορίθμου A για το ζεύγος (P,P) για να δημιουργήσουμε ένα πρόγραμμα Q τέτοιο ώστε

$$Q(P) = \begin{cases} \text{halts}, & \text{if } A(P, P) = 0, \text{ i.e., if program } P \text{ does not halt on input } P \\ \text{does not halt}, & \text{if } A(P, P) = 1, \text{ i.e., if program } P \text{ halts on input } P \end{cases}$$

- Αν αντικαταστήσουμε το Q στο P θα έχουμε:

$$Q(Q) = \begin{cases} \text{halts}, & \text{if } A(Q, Q) = 0, \text{ i.e., if program } Q \text{ does not halt on input } Q \\ \text{does not halt}, & \text{if } A(Q, Q) = 1, \text{ i.e., if program } Q \text{ halts on input } Q \end{cases}$$

- Όμως, έχουμε φτάσει σε αντίθεση αφού καμία από τις δύο εξόδους για το Q δεν είναι πιθανή

P, N, NP-Complete Problems (8/24)

Το επόμενο ερώτημα που πρέπει να απαντήσουμε είναι: ***υπάρχουν προβλήματα που είναι αποφασίσιμα αλλά όχι ανιχνεύσιμα***;

Η απάντηση είναι, ***ναι αλλά ο αριθμός των παραδειγμάτων είναι μικρός***

Υπάρχουν πολλά προβλήματα για τα οποία δεν έχει βρεθεί αλγόριθμος πολυωνυμικού χρόνου ούτε η μη ύπαρξη τέτοιων αλγορίθμων έχει αποδειχθεί

P, N, NP-Complete Problems (9/24)

Παραδείγματα

- Το πρόβλημα εύρεσης ενός Hamiltonian circuit σε ένα γράφο
  - Εύρεση μονοπατιού που ξεκινά και τελειώνει στον ίδιο κόμβο περνώντας από όλους τους υπόλοιπους κόμβους μόνο μια φορά
- Το πρόβλημα του travelling salesman
  - Εύρεση της συντομότερης διαδρομής σε n πόλεις με γνωστές τις θετικές αποστάσεις μεταξύ τους
- Το πρόβλημα του σακιδίου (knapsack problem)
  - Εύρεση του πιο πολύτιμου υποσυνόλου n αντικειμένων που μπορούν να χωρέσουν σε ένα σακίδιο με δοσμένη χωρητικότητα

P, N, NP-Complete Problems (10/24)

Παραδείγματα (συνέχεια)

- Το πρόβλημα του διαχωρισμού (partition)
  - Δοσμένων n θετικών ακεραίων, να καθορίσουμε αν είναι δυνατόν να τους χωρίσουμε σε δύο αμοιβαία αποκλειόμενα σύνολα με το ίδιο άθροισμα
- The bin packing problem
  - Δοσμένων n αντικειμένων των οποίων το μέγεθος είναι θετικοί αριθμοί όχι μεγαλύτεροι από 1, να τα τοποθετήσουμε στο μικρότερο αριθμό κάδων μεγέθους 1
- Το πρόβλημα του χρωματισμού γράφων
  - Για ένα δοσμένο γράφο, να βρούμε το χρωματικό αριθμό του; Ο μικρότερος αριθμός χρωμάτων που είναι απαραίτητα να ανατεθούν στους κόμβους του γράφου έτσι ώστε οι γειτονικοί κόμβοι να έχουν διαφορετικό χρώμα

P, N, NP-Complete Problems (11/24)

Παραδείγματα (συνέχεια)

- The integer linear programming problem
  - Να βρούμε τη μέγιστο ή την ελάχιστη τιμή μιας γραμμικής συνάρτησης που έχει ένα σύνολο ακεραίων μεταβλητών με δεδομένο ένα πεπερασμένο σύνολο περιορισμών (στη μορφή γραμμικών εξισώσεων και ανισώσεων)

Το κοινό χαρακτηριστικό σε αυτά τα προβλήματα είναι έχουν μια εκθετική αύξηση των επιλογών λύσης ως συνάρτηση του μεγέθους του προβλήματος n

Επίσης, κάποια από αυτά μπορεί να επιλυθούν σε πολυωνυμικό χρόνο

P, N, NP-Complete Problems (12/24)

Ένα άλλο χαρακτηριστικό των προβλημάτων απόφασης είναι ότι παρά το γεγονός ότι η επίλυσή τους είναι δύσκολη υπολογιστικά, ο έλεγχος της λύσης μπορεί να γίνει σε πολυωνυμικό χρόνο

Για παράδειγμα, είναι εύκολο να ελέγξουμε αν μια προτεινόμενη σειρά κόμβων για ένα Hamiltonian circuit είναι μια αποδεκτή λύση

Το μόνο που χρειάζεται είναι να ελέγξουμε ότι η λύση αποτελείται από n+1 κόμβους, όλοι οι ενδιάμεσοι κόμβοι είναι διακριτοί μεταξύ τους, ο τελευταίος είναι ίδιος με τον πρώτο και όλοι οι κόμβοι συνδέονται με μια ακμή

P, N, NP-Complete Problems (13/24)

Ορισμός

- Ένας **μη ντετερμινιστικός (nondeterministic)** αλγόριθμος είναι μια δυο βημάτων διαδικασία που παίρνει σαν είσοδο ένα στιγμιότυπο I ενός προβλήματος απόφασης και εκτελεί τα ακόλουθα:
  - **Μη ντετερμινιστική φάση (υπόθεση - guessing)**: ένα αλφαριθμητικό S παράγεται ως μια υποψήφια λύση για το στιγμιότυπο I
  - **Ντετερμινιστική φάση (επιβεβαίωση - verification):** Ένας ντετερμινιστικός αλγόριθμος παίρνει τα I και S σαν είσοδο και εξάγει την απάντηση ναι αν το S αναπαριστά μια λύση για το στιγμιότυπο I

Λέμε πως ένας μη ντετερμινιστικός αλγόριθμος λύνει ένα πρόβλημα απόφασης **όταν και μόνο όταν για κάθε στιγμιότυπο που απαιτεί θετική απάντηση, επιστρέφει θετικό αποτέλεσμα**

P, N, NP-Complete Problems (14/24)

Ένας μη ντετερμιστικός αλγόριθμος λέμε ότι είναι **πολυωνυμικός μη ντετερμινιστικός (nondeterministic polynomial)** εφόσον η επίδοση χρόνου της επιβεβαίωσης / επαλήθευσης είναι πολυωνυμικός

Ορισμός

- **Η τάξη / κλάση NP είναι η τάξη των προβλημάτων απόφασης που μπορεί να επιλυθούν από μη ντετερμινιστικούς πολυωνυμικούς αλγορίθμους. Αυτή η τάξη προβλημάτων ονομάζεται μη ντετερμινιστική πολυωνυμική (nondeterministic polynomial)**

P, N, NP-Complete Problems (15/24)

Πολλά προβλήματα ανήκουν στην τάξη NP

Η τάξη NP περιλαμβάνει τα προβλήματα που ανήκουν στην τάξη P

$$P \subseteq NP$$

Αν ένα πρόβλημα ανήκει στην τάξη P, μπορούμε να υιοθετήσουμε ένα ντετερμινιστικό αλγόριθμο που το επιλύει στη φάση της επιβεβαίωσης / επαλήθευσης ενός μη ντετερμινιστικού αλγορίθμου που απλά αγνοεί το S που εξάγεται από τη μη ντετερμινιστική φάση

Η τάξη NP περιλαμβάνει: the Hamiltonian circuit, the partition problem, decision versions of the traveling salesman, the knapsack problem, graph coloring, combinatorial problems

P, N, NP-Complete Problems (16/24)

Ερώτημα προς απάντηση από τους επιστήμονες της Θεωρητικής Πληροφορικής

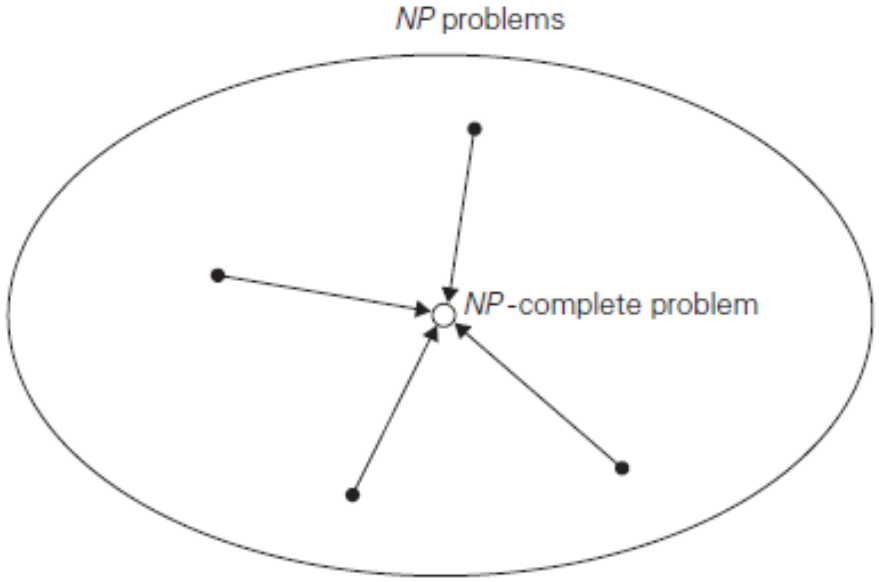
- ***Η τάξη P είναι ένα κατάλληλο υποσύνολο της τάξης NP ή οι δύο τάξεις είναι στην πραγματικότητα οι ίδιες;***

Αν P=NP, τότε κάθε ένα από τα εκατοντάδες δύσκολα προβλήματα συνδυαστικής μπορεί να επιλυθεί σε πολυωνυμικό χρόνο ακόμα και αν οι Επιστήμονες έχουν αποτύχει να βρουν τους αντίστοιχους αλγορίθμους

Επιπρόσθετα, πολλά προβλήματα είναι γνωστό πως είναι NP-complete που προσθέτει αμφιβολίες για τη σχέση P=NP

P, N, NP-Complete Problems (17/24)

Ένα **NP-complete** πρόβλημα, είναι ένα πρόβλημα το οποίο είναι τόσο δύσκολο όσο οποιοδήποτε άλλο πρόβλημα που ανήκει στη τάξη NP επειδή, εξ' ορισμού, οποιοδήποτε άλλο πρόβλημα στην NP μπορεί να 'ελαττωθεί' σε πολυωνυμικό χρόνο



P, N, NP-Complete Problems (18/24)

Ορισμός

- **Ένα πρόβλημα απόφασης D1 λέμε ότι είναι πολυωνυμικά αναγώγιμο (polynomially reducible) σε ένα πρόβλημα D2, αν υπάρχει μια συνάρτηση f που μετασχηματίζει το D1 σε στιγμιότυπα του D2 τέτοια ώστε:**
  - **Η f αντιστοιχίζει όλα τα θετικά στιγμιότυπα (απάντηση: ναι) του D1 στα θετικά στιγμιότυπα του D2 και όλα τα μη δυνατά στιγμιότυπα του D1 σε μη δυνατά στιγμιότυπα του D2**
  - **Η f υπολογίζεται από ένα πολυωνυμικού χρόνου αλγόριθμο**

Ο ορισμός μας υποδεικνύει ότι αν ένα πρόβλημα D1 είναι πολυωνυμικά αναγώγιμο σε ένα πρόβλημα D2 που μπορεί να επιλυθεί σε πολυωνυμικό χρόνο, τότε το ίδιο ισχύει και για το D1

P, N, NP-Complete Problems (19/24)

Ορισμός

- **Ένα πρόβλημα απόφασης D λέμε ότι είναι NP-complete εάν:**
  - **ανήκει στην τάξη NP**
  - **Κάθε πρόβλημα στην NP είναι πολυωνυμικά αναγώγιμο στο D**

Θα αποδείξουμε ότι το πρόβλημα εύρεσης του Hamiltonian circuit είναι πολυωνυμικά αναγώγιμο στην έκδοση απόφασης του traveling salesman problem

Το Hamiltonian circuit μπορεί να οριστεί σαν ένα πρόβλημα ύπαρξης ενός Hamiltonian circuit όχι μεγαλύτερου από ένα θετικό αριθμό m σε ένα δοσμένο γράφο με θετικά βάρη

P, N, NP-Complete Problems (20/24)

Μπορούμε να απεικονίσουμε ένα γράφο G ενός συγκεκριμένου στιγμιότυπου του Hamiltonian circuit προβλήματος σε ένα πλήρες γράφο με βάρη G’ που αναπαριστά ένα στιγμιότυπο του traveling salesman problem

Στον G’ αναθέτουμε 1 ως βάρος σε κάθε ακμή του G και προσθέτουμε 2 ανάμεσα σε κάθε ζεύγος μη γειτονικών κόμβων του G

Σαν πάνω όριο m του μήκους του Hamiltonian circuit παίρνουμε m=n, όπου n είναι το πλήθος των κόμβων στα G, G’

Ο μετασχηματισμός μπορεί να ολοκληρωθεί σε πολυωνυμικό χρόνο

P, N, NP-Complete Problems (21/24)

Έστω G είναι ένα θετικό στιγμιότυπο του Hamiltonian circuit

Τότε το G έχει ένα Hamiltonian circuit και η εικόνα του G’ θα έχει μήκος n το οποίο κάνει την εικόνα G’ ένα θετικό στιγμιότυπο της απόφασης του traveling salesman problem

Αντίστοιχα, αν έχουμε ένα Hamiltonian circuit μήκους όχι περισσότερο από n στο G’, τότε το μήκος του πρέπει να είναι n και το Hamiltonian circuit πρέπει να αποτελλείται από ακμές του G δημιουργώντας μια αντίστροφη εικόνα των θετικών στιγμιοτύπων του traveling salesman problem που θα είναι ένα θετικό στιγμιότυπο του προβλήματος της εύρεσης του Hamiltonian circuit

P, N, NP-Complete Problems (22/24)

Η έννοια της NP-completeness απαιτεί πολυωνυμική αναγωγή όλων (γνωστών και αγνώστων) των NP προβλημάτων στο πρόβλημα που εξετάζουμε

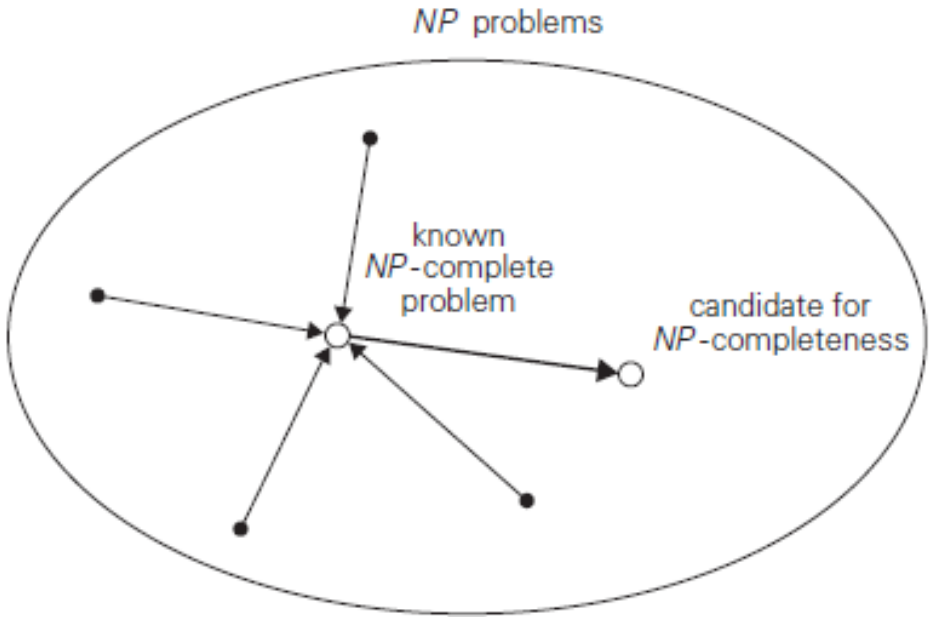
Ένα σύνολο NP-complete προβλημάτων έχουν βρεθεί

Μια δυναμική λίστα τέτοιων προβλημάτων μπορεί κανείς να δει εδώ https://en.wikipedia.org/wiki/List_of_NP-complete_problems

P, N, NP-Complete Problems (23/24)

Για την ανάδειξη ενός προβλήματος ως NP-complete απαιτεί δύο βήματα:

- Πρώτα αποδεικνύουμε ότι το πρόβλημα είναι NP (τυπικά αυτό το βήμα είναι εύκολο)
- Έπειτα αποδεικνύουμε ότι κάθε πρόβλημα στην NP είναι αναγώγιμο στο πρόβλημα που εξετάζουμε σε πολυωνυμικό χρόνο
  - Λόγω της μεταβατικότητας της πολυωνυμικής αναγωγής, το βήμα μπορεί να γίνει αποδεικνύοντας ότι ένα γνωστό NP-complete πρόβλημα μπορεί να αναχθεί σε πολυωνυμικό χρόνο στο πρόβλημα που εξετάζουμε



P, N, NP-Complete Problems (24/24)

Η αναγωγή μέσω ενός γνωστού NP-complete προβλήματος είναι πολύ πιο εύκολη διαδικασία από το να ανάγουμε όλα τα NP προβλήματα στο πρόβλημα που εξετάζουμε

Ο ορισμός της NP-completeness μας δείχνει ότι αν υπάρχει ένας ντετερμινιστικός πολυωνυμικού χρόνου αλγόριθμος για ένα NP-complete πρόβλημα, τότε κάθε NP-complete πρόβλημα μπορεί να λυθεί σε πολυωνυμικό χρόνο μέσω ενός ντετερμινιστικού αλγορίθμου

Χειρισμός των Περιορισμών των Αλγορίθμων

Χειρισμός Περιορισμών (1/32)

Κάποια από τα προβλήματα που είναι δύσκολο να λυθούν αλγοριθμικά, είναι πολύ σημαντικά, συνεπώς θα πρέπει να βρεθεί κάποια λύση

Χρησιμοποιούμε τις τεχνικές **back-tracking** και **branch-and-bound** για να λύσουμε κάποια στιγμιότυπα των προβλημάτων

Και οι δύο τεχνικές βασίζονται στη δημιουργία του **space-state tree**

Οι κόμβοι του δένδρου αντιπροσωπεύουν συγκεκριμένες επιλογές για την επίλυση του προβλήματος

Και οι δύο τεχνικές τερματίζουν ένα κόμβο όταν αυτό δεν μπορεί να εγγυηθεί την επίλυση του προβλήματος

Χειρισμός Περιορισμών (2/32)

Η backtracking χρησιμοποιείται σε προβλήματα που δεν σχετίζονται με βελτιστοποίηση

Η branch-and-bound σχετίζεται και υιοθετείται σε προβλήματα βελτιστοποίησης

Στην backtracking οι κόμβοι του δένδρου κατασκευάζονται σε μια σειρά depth-first

Στην branch-and-bound υιοθετούνται αρκετές τεχνικές μια από τις οποίες είναι ο κανόνας best-first

Χειρισμός Περιορισμών (3/32)

Backtracking

- Η βασική ιδέα είναι να κατασκευάζουμε μια τμηματική λύση τη φορά και να την επαληθεύουμε
- Αν μια τμηματική λύση μπορεί να επεκταθεί περαιτέρω χωρίς να παραβιάζουμε τους διάφορους περιορισμούς, την επιλέγουμε παίρνοντας το επόμενο αποδεκτό στοιχείο της λύσης
- Αν δεν υπάρχει μια αποδεκτή τμηματική επέκταση της λύσης, τότε δεν εξετάζεται καμμία επέκταση
- Σε αυτή την περίπτωση ο αλγόριθμος οπισθοχωρεί ώστε να αντικαταστήσει το τελευταία αποδεκτό στοιχείο της λύσης με την επόμενη επιλογή

Χειρισμός Περιορισμών (4/32)

Backtracking

- Ο αλγόριθμος κατασκευάζει το λεγόμενο **state-space tree**
- Η ρίζα του δένδρου είναι μια αρχική κατάσταση πριν ξεκινήσει η αναζήτηση των τμημάτων της λύσης
- Οι κόμβοι στο πρώτο επίπεδο αντιστοιχούν στις επιλογές που είναι διαθέσιμες για το πρώτο τμήμα της λύσης, οι κόμβοι στο δεύτερο επίπεδο είναι οι επιλογές για το δεύτερο στοιχείο, κ.ο.κ.
- Ένας κόμβος λέμε ότι είναι **υποσχόμενος (promising)** αν αντιστοιχεί σε μια τμηματική λύση που μπορεί να οδηγήσει σε μια πλήρη λύση, αλλιώς καλείται **μη υποσχόμενος (nonpromising)**
- Τα φύλλα αντιστοιχούν είτε σε μη υποσχόμενες λύσεις ή σε αποδεκτές λύσεις

Χειρισμός Περιορισμών (5/32)

Backtracking

- Αν ένας κόμβος είναι υποσχόμενος, τότε κατασκευάζονται τα παιδιά του
- Σε διαφορετική περίπτωση, ο αλγόριθμος οπισθοχωρεί ένα επίπεδο πάνω και για να εξετάσει την επόμενη στη σειρά επιλογή κ.ο.κ.
- Αν ο αλγόριθμος φτάσει σε μια αποδεκτή λύση, είτε σταματά ή συνεχίζει να αναζητά άλλες αποδεκτές λύσεις

Χειρισμός Περιορισμών (6/32)

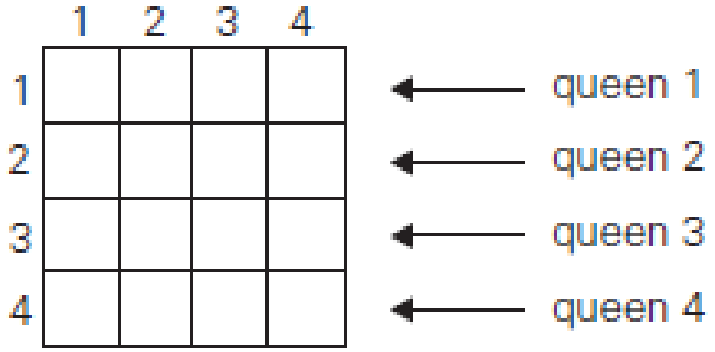
n-Queens Example

- Καλούμαστε να τοποθετήσουμε σε ένα σκάκι nXn, n βασίλισσες, έτσι ώστε να μην μπορούν 2 βασίλισσες να επιτεθούν η μια στην άλλη (ίδια γραμμή, ίδια στήλη, ίδια διαγώνιο)
- Για n=1, η λύση είναι εύκολη
- Επίσης είναι εύκολο να δούμε πως δεν υπάρχει λύση για n=2, n=3
- Ας υποθέσουμε πως έχουμε n=4 και λύνουμε το πρόβλημα με backtracking
- Για n>=4, μια λύση μπορεί να βρεθεί σε γραμμικό χρόνο

Χειρισμός Περιορισμών (7/32)

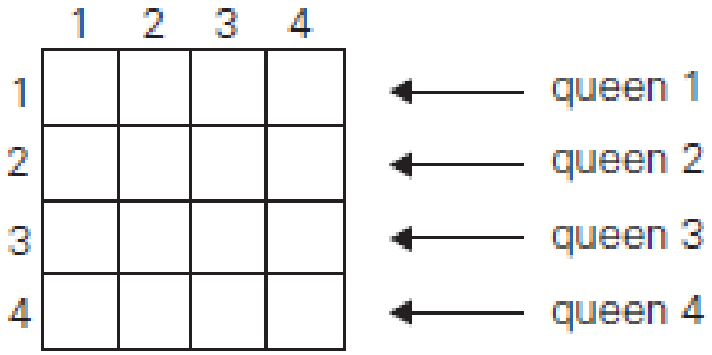
n-Queens Example

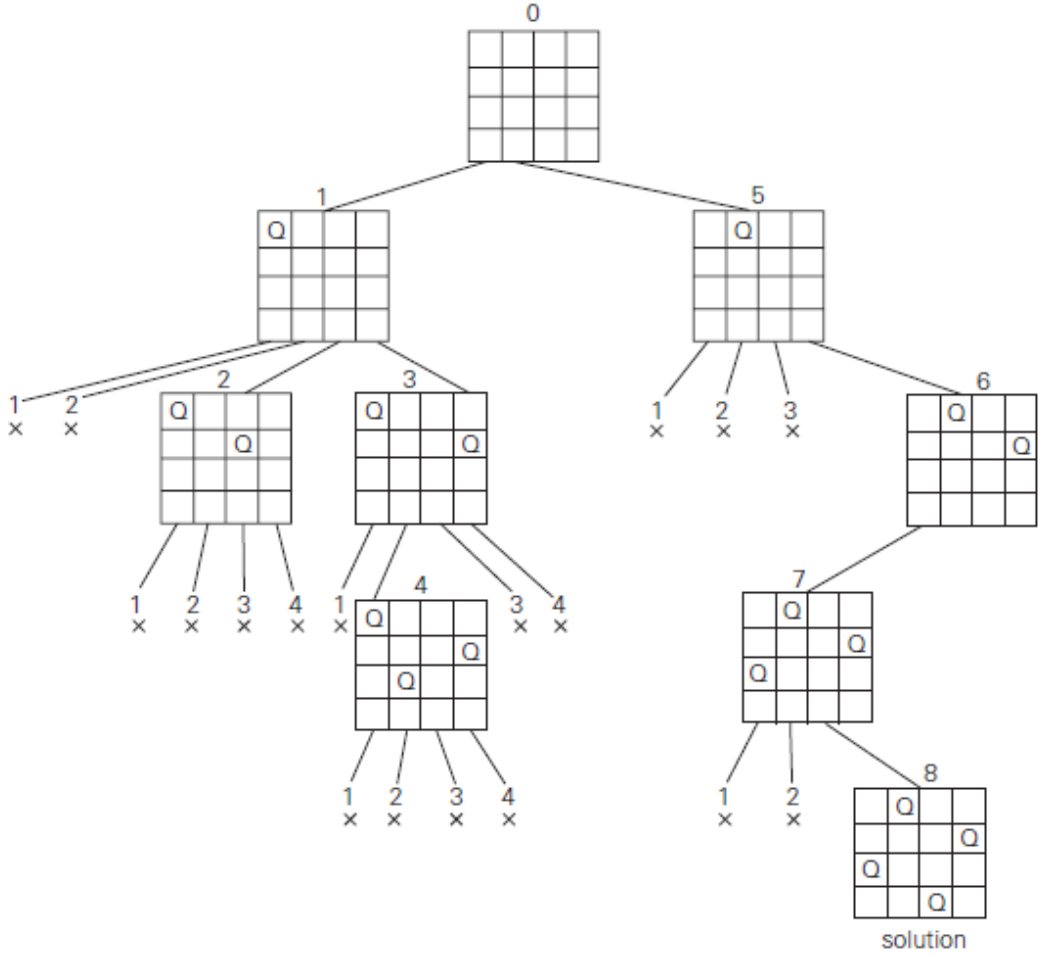
- Ξεκινάμε με άδειο σκάκι και τοποθετούμε την 1η βασίλισσα στην 1η θέση της γραμμής 1.
- Μετά τοποθετούμε τη 2η βασίλισσα (αφού προσπαθήσουμε στις στήλες 1,2) στο κελί (2,3)
- Αποδεικνύεται ότι αυτή η κίνηση δεν είναι αποδεκτή αφού δεν θα μπορεί να τοποθετηθεί η 3η βασίλισσα
- Οπισθοχωρούμε και και βάζουμε τη 2η βασίλισσα στο (2,4)
- Η 3η βασίλισσα θα τοποθετηθεί στο (3,2) το οποίο είναι επίσης αδιέξοδο
- Οπσιθοχωρούμε και βάζουμε την 1η βασίλισσα στο (1,2)
- Η 2η θα πάει στο (2,4), η 3η στο (3,1) και η 4η στο (4,3)



Χειρισμός Περιορισμών (8/32)

n-Queens Example

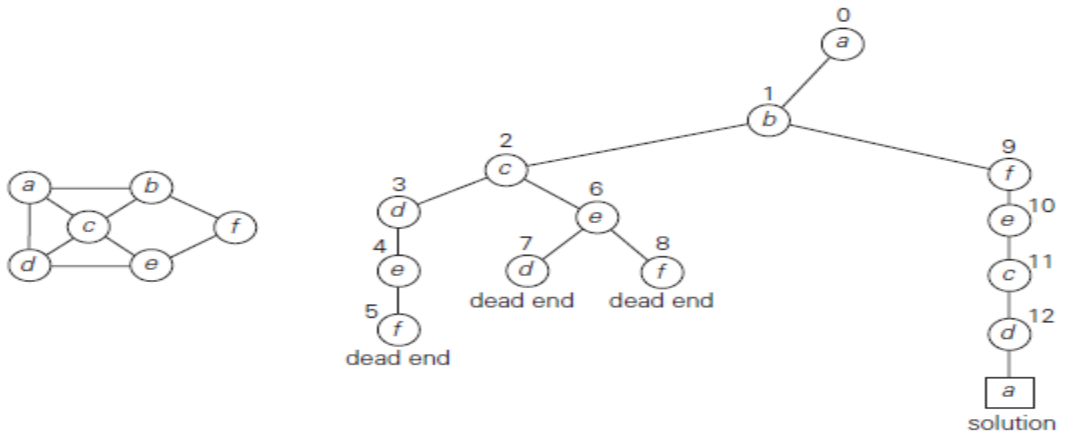




Χειρισμός Περιορισμών (9/32)

Hamiltonian circuit

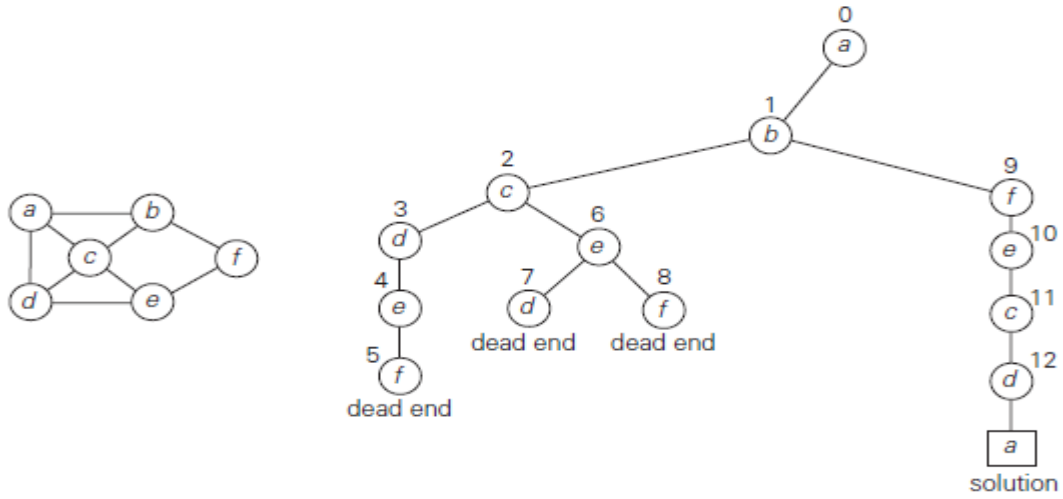
- Χωρίς βλάβη της γενικότητας θεωρούμε πως η διαδρομή ξεκινά από τον κόμβο a
- Θεωρούμε τον a ως τη ρίζα του δένδρου
- Το πρώτο στοιχείο της λύσης μας, εφόσον υπάρχει είναι ο πρώτος άμεσος κόμβος του Hamiltonian circuit



Χειρισμός Περιορισμών (10/32)

Hamiltonian circuit

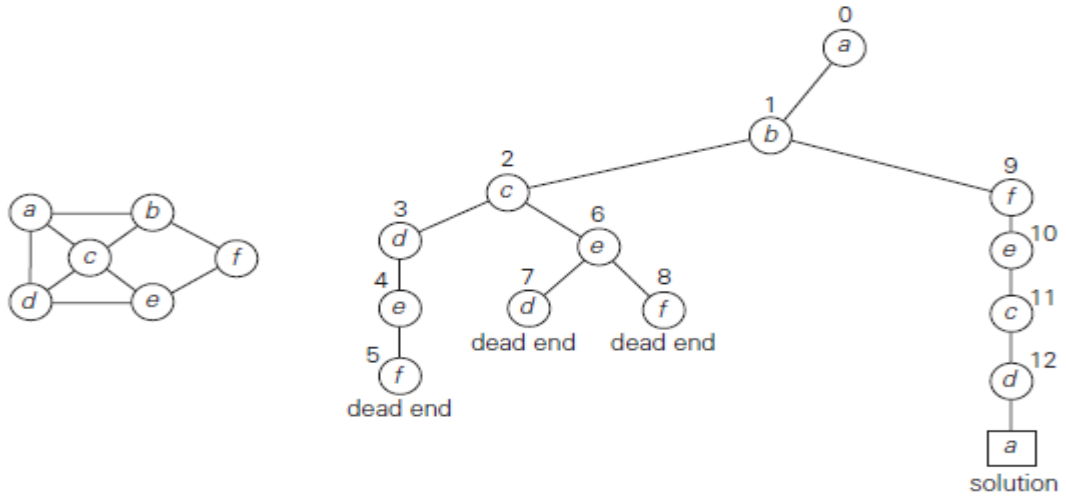
- Αν θεωρήσουμε την αλφαβητική σειρά για τους άμεσους γειτονικούς κόμβους, τότε προχωράμε στον επόμενο κόμβο της διαδρομής που είναι ο b
- Από τον b προχωράμε στον c



Χειρισμός Περιορισμών (11/32)

Hamiltonian circuit

- Μετά πάμε στον d, στον e και τελικά στον f που αποδεικνύεται ότι είναι αδιέξοδο
- Οπισθοχωρούμε και από τον f, στον e, στον d, στον c, κ.ο.κ.



Χειρισμός Περιορισμών (12/32)

Backtracking

- Γενικά η έξοδος του αλγορίθμου είναι ένα διάνυσμα (x_1, x_2, …, x_n) όπου οι συντεταγμένες x_i είναι ένα στοιχείο ενός πεπερασμένου συνόλου S_i
- Εξαρτάται από το πρόβλημα το αν όλα τα διανύσματα θα έχουν το ίδιο μήκος

ALGORITHM *Backtrack*$(X[1..i])$

```
//Gives a template of a generic backtracking algorithm
//Input: X[1..i] specifies first i promising components of a solution
//Output: All the tuples representing the problem's solutions
if X[1..i] is a solution write X[1..i]
else      //see Problem 9 in this section's exercises
    for each element x ∈ S_{i+1} consistent with X[1..i] and the constraints do
        X[i + 1] ← x
        Backtrack(X[1..i + 1])
```

Χειρισμός Περιορισμών (13/32)

Backtracking

- Η τεχνική δεν είναι αποδοτική, γενικά
- Στη χειρότερη περίπτωση θα πρέπει να δημιουργήσουμε όλες τις πιθανές λύσεις
- Ελπίζουμε στο να μην ληφθούν υπόψιν αρκετές διαδρομές προς μη εφικτές λύσεις

ALGORITHM *Backtrack*$(X[1..i])$

```
//Gives a template of a generic backtracking algorithm
//Input: X[1..i] specifies first i promising components of a solution
//Output: All the tuples representing the problem's solutions
if X[1..i] is a solution write X[1..i]
else      //see Problem 9 in this section's exercises
    for each element x ∈ S_{i+1} consistent with X[1..i] and the constraints do
        X[i + 1] ← x
        Backtrack(X[1..i + 1])
```

Χειρισμός Περιορισμών (14/32)

Backtracking

- Υπάρχει ένα σύνολο μεθόδων για να μειώσουμε το μέγεθος του δένδρου
- Ένας τρόπος είναι να εξερευνήσουμε τη συμμετρία που υπάρχει συχνά σε προβλήματα συνδυαστικά
- Η παρατήρηση αυτή μπορεί να μειώσει στο μισό περίπου το πλήθος πιθανών λύσεων
- Για παράδειγμα, η $1^{η}$ βασίλισσα δεν χρειάζεται να τοποθετηθεί στις τελευταίες n/2 στήλες

Χειρισμός Περιορισμών (15/32)

Backtracking

- Άλλος τρόπος είναι να προ-αναθέσουμε τιμές σε ένα ή περισσότερα στοιχεία μιας λύσης
- Για παράδειγμα μπορούμε να προ-ταξινομήσουμε κάποια δεδομένα

Χειρισμός Περιορισμών (16/32)

Backtracking

- Είναι δύσκολο να εκτιμήσουμε το μέγεθος του δένδρου
- Ο Knuth πρότεινε να δημιουργήσουμε ένα τυχαίο μονοπάτι από τη ρίζα μέχρι ένα φύλλο και χρησιμοποιώντας την πληροφορία σχετικά με το πλήθος των επιλογών που είναι διαθέσιμες όταν δημιουργούμε το μονοπάτι να εκτιμήσουμε το μέγεθος του δένδρου
- Έστω c_1 το πλήθος των τιμών για το πρώτο στοιχείο
- Επιλέγουμε τυχαία (πιθανότητα $1/c_1$) μια από αυτές
- Επαναλαμβάνουμε τη διαδικασία για τις τιμές c_2, κ.ο.κ.

Χειρισμός Περιορισμών (17/32)

Backtracking

- Υποθέτωντας πως στο επίπεδο i το πλήθος των κόμβων είναι c_i κατά μέσο όρο, η εκτίμηση του πλήθους των κόμβων είναι: $1+c_1+c_1c_2+\ldots+c_1c_2..c_n$
- Αν δημιουργήσουμε πολλές τυχαίες ακολουθίες και πάρουμε το μέσο όρο του μεγέθους, θα έχουμε μια εκτίμηση για το τελικό μέγεθος του δένδρου

Χειρισμός Περιορισμών (18/32)

Branch-and-bound

- Ένα πρόβλημα βελτιστοποίησης αναζητά τη μεγιστοποίηση ή ελαχιστοποίηση κάποιας συνάρτησης με βάση κάποιους περιορισμούς
- Σε σχέση με το backtracking η μέθοδος branch-and-bound απαιτεί δύο επιπρόσθετα βήματα:
  - Ένα τρόπο να καθορίσει για κάθε κόμβο του δένδρου ένα όριο της βέλτιστης τιμής της συνάρτησης βελτιστοποίησης σε σχέση με οποιαδήποτε λύση που μπορεί να προκύψει από το συγκεκριμένο κόμβο
  - Την τιμή της βέλτιστης λύσης μέχρι στιγμής
- Αν αυτές οι πληροφορίες είναι διαθέσιμες, μπορούμε να συγκρίνουμε το όριο του κάθε κόμβου με τη βέλτιστη λύση μέχρι στιγμής

Χειρισμός Περιορισμών (19/32)

Branch-and-bound

- Αν το όριο δεν είναι καλύτερο από τη βέλτιστη τιμή, ο κόμβος είναι μη υποσχόμενος και μπορεί να τερματιστεί
- Τερματίζουμε την αναζήτηση σε ένα κόμβο όταν:
  - Η τιμή του ορίου του **κόμβου δεν είναι καλύτερη από την τιμή της βέλτιστης λύσης** μέχρι στιγμής
  - Ο κόμβος αναπαριστά μια **μη εφικτή λύση** επειδή οι περιορισμοί του προβλήματος έχουν ήδη παραβιαστεί
  - Το υποσύνολο των εφικτών λύσεων που απεικονίζονται από τον κόμβο **είναι ένα απλό σημείο** – συγκρίνουμε την τιμή της συνάρτησης για αυτή την εφικτή λύση με τη βέλτιστη λύση μέχρι στιγμής

Χειρισμός Περιορισμών (20/32)

The Assignment Problem

- Εφαρμόζουμε την branch-and-bound τεχνική στο πρόβλημα της ανάθεσης
- Θέλουμε να αναθέσουμε σε n άτομα, n εργασίες
- Υιοθετούμε τον nXn πίνακα κόστους
- Το κόστος οποιαδήποτε λύσης δεν μπορεί να είναι μικρότερο από το άθροισμα των μικρότερων στοιχείων σε κάθε γραμμή του πίνακα, 2+3+1+4=10
- Αυτό το κόστος μπορεί να μην αντιστοιχεί σε μια αποδεκτή λύση

	job 1	job 2	job 3	job 4	
$C =$	9	2	7	8	person *a*
	6	4	3	7	person *b*
	5	8	1	8	person *c*
	7	6	9	4	person *d*

Χειρισμός Περιορισμών (21/32)

The Assignment Problem

- Εφαρμόζουμε την ίδια λογική σε κάθε τμηματική λύση
- Για παράδειγμα, επιλέγοντας την ανάθεση (α,1) θα προκύψει ένα ελάχιστο κόστος 9+3+1+4=17
- Σε αντίθεση με την backtracking, δημιουργούμε όλα τα παιδιά της πιο υποσχόμενης λύσης ανάμεσα στους μη τερματικούς κόμβους
- Συγκρίνουμε τα ελάχιστα όρια κάθε κόμβου
- Το καλύτερο όριο αντιστοιχεί στον πιο πολλά υποσχόμενο κόμβο

$$C = \begin{array}{c} \begin{array}{cccc} \text{job 1} & \text{job 2} & \text{job 3} & \text{job 4} \end{array} \\ \begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix} \end{array} \begin{array}{l} \text{person } a \\ \text{person } b \\ \text{person } c \\ \text{person } d \end{array}$$

Χειρισμός Περιορισμών (22/32)

The Assignment Problem

- Η συγκεκριμένη στρατηγική ονομάζεται **best-first branch-and-bound**

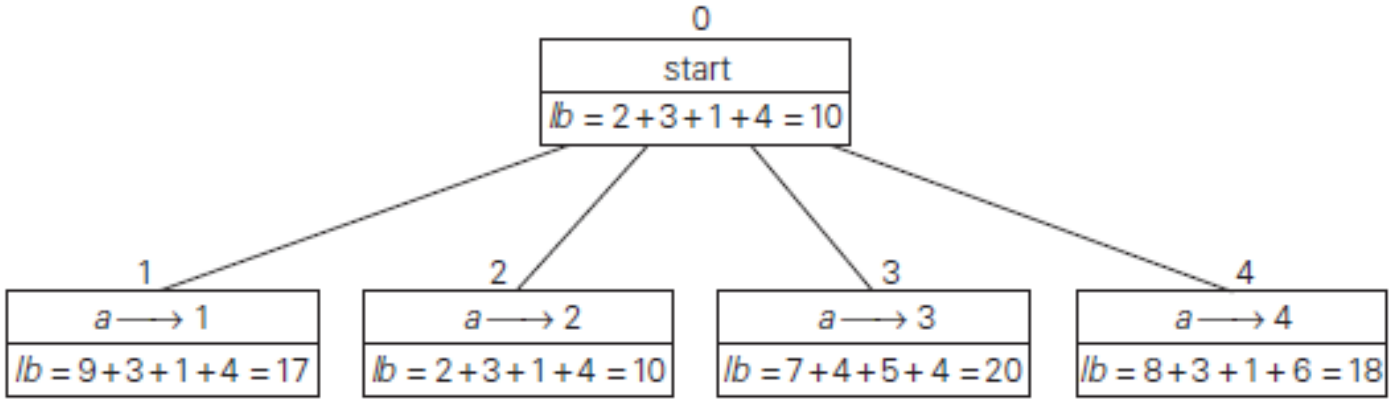
$$C = \begin{array}{c} \begin{matrix} \text{job 1} & \text{job 2} & \text{job 3} & \text{job 4} \end{matrix} \\ \begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix} \end{array} \begin{matrix} \\ \text{person } a \\ \text{person } b \\ \text{person } c \\ \text{person } d \end{matrix}$$

Χειρισμός Περιορισμών (23/32)

The Assignment Problem

- Ξεκινάμε από τη ρίζα που δεν περιλαμβάνει κάποιο στοιχείο από τον πίνακα κόστους
- Το ελάχιστο κόστος είναι 10
- Δημιουργούμε 4 φύλλα: κόμβοι 1-4

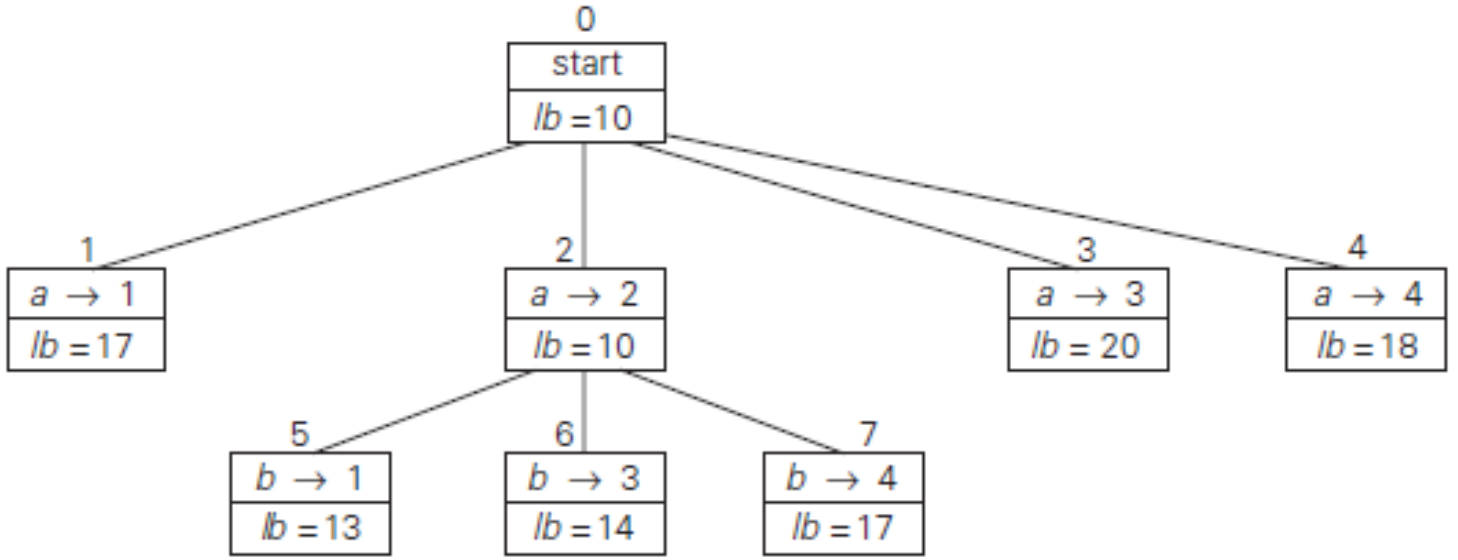
$$C = \begin{array}{c} \begin{matrix} \text{job 1} & \text{job 2} & \text{job 3} & \text{job 4} \end{matrix} \\ \begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix} \end{array} \begin{matrix} \\ \text{person } a \\ \text{person } b \\ \text{person } c \\ \text{person } d \end{matrix}$$



Χειρισμός Περιορισμών (24/32)

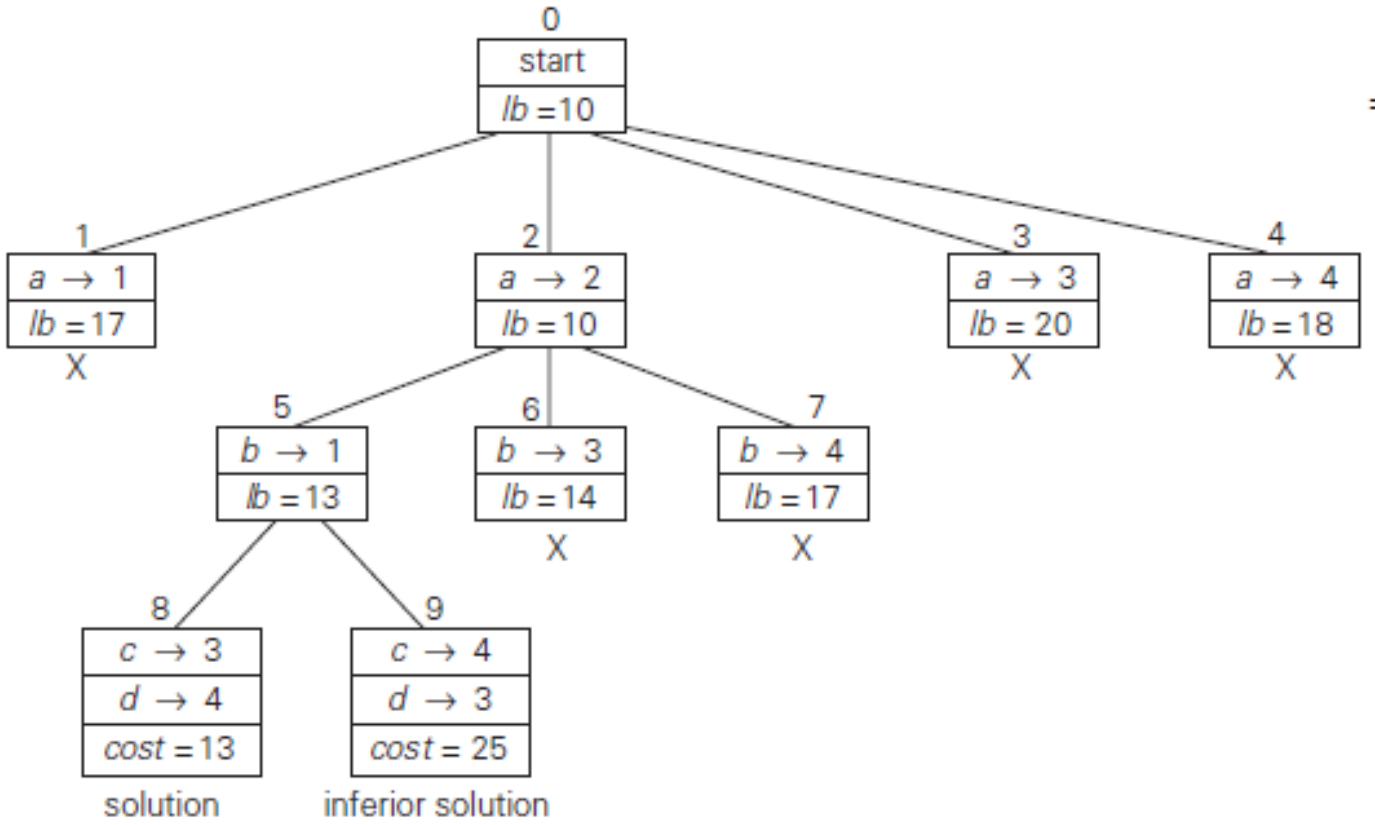
The Assignment Problem

$$C = \begin{array}{c} \begin{array}{cccc} \text{job 1} & \text{job 2} & \text{job 3} & \text{job 4} \end{array} \\ \begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix} \end{array} \begin{array}{l} \text{person } a \\ \text{person } b \\ \text{person } c \\ \text{person } d \end{array}$$



Χειρισμός Περιορισμών (25/32)

The Assignment Problem



$$= \begin{array}{c} \begin{array}{cccc} \text{job 1} & \text{job 2} & \text{job 3} & \text{job 4} \end{array} \\ \begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix} \end{array} \begin{array}{l} \text{person } a \\ \text{person } b \\ \text{person } c \\ \text{person } d \end{array}$$

Χειρισμός Περιορισμών (26/32)

The knapsack problem

- Έχουμε n αντικείμενα βάρους w_i και αξίας v_i και ένα σακίδιο χωρητικότητας W
- Πρέπει να βρούμε τα μεγαλύτερης αξίας αντικείμενα που χωράνε στο σακίδιο
- Είναι βολικό να ταξινομήσουμε τα αντικείμενα σε φθίνουσα σειρά αξίας προς βάρος – οπότε το πρώτο αντικείμενο έχει τον καλύτερο λόγο

$$v_1/w_1 \geq v_2/w_2 \geq \cdots \geq v_n/w_n$$

- Το δένδρο κατασκευάζεται σαν ένα δυαδικό δένδρο αναζήτησης

Χειρισμός Περιορισμών (27/32)

The knapsack problem

- Κάθε κόμβος το i επίπεδο αναπαριστά όλα τα υποσύνολα των αντικειμένων που περιλαμβάνουν μια συγκεκριμένη επιλογή από τα πρώτα i αντικείμενα
- Η συγκεκριμένη επιλογή καθορίζεται μοναδικά μέσω της διαδρομής από τη ρίζα μέχρι ένα κόμβο
- Μια διακλάδωση αριστερά μας δείχνει ότι συμπεριλαμβάνουμε το αντικείμενο ενώ η διασταύρωση προς τα δεξιά μας δείχνει ότι δεν συμπεριλαμβάνουμε το αντικείμενο
- Καταγράφουμε το συνολικό βάρος w και τη συνολική αξία υ μαζί με ένα άνω όριο **ub** της τιμής οποιουδήποτε υποσυνόλου που μπορεί να προκύψει προσθέτωντας μηδέν ή περισσότερα αντικείμενα στην τρέχουσα επιλογή

Χειρισμός Περιορισμών (28/32)

The knapsack problem

- Ένας απλός τρόπος υπολογισμού του ub είναι να προσθέσουμε στο υ τη συνολική τιμή των αντικειμένων που έχουν ήδη επιλεγεί, το γινόμενο της εναπομείνουσας χωρητικότητας του σακιδίου W-w και τον καλύτερο λόγο αξία προς βάρος από τα εναπομείναντα αντικείμενα

$$ub = v + (W - w)(v_{i+1}/w_{i+1})$$

Χειρισμός Περιορισμών (29/32)

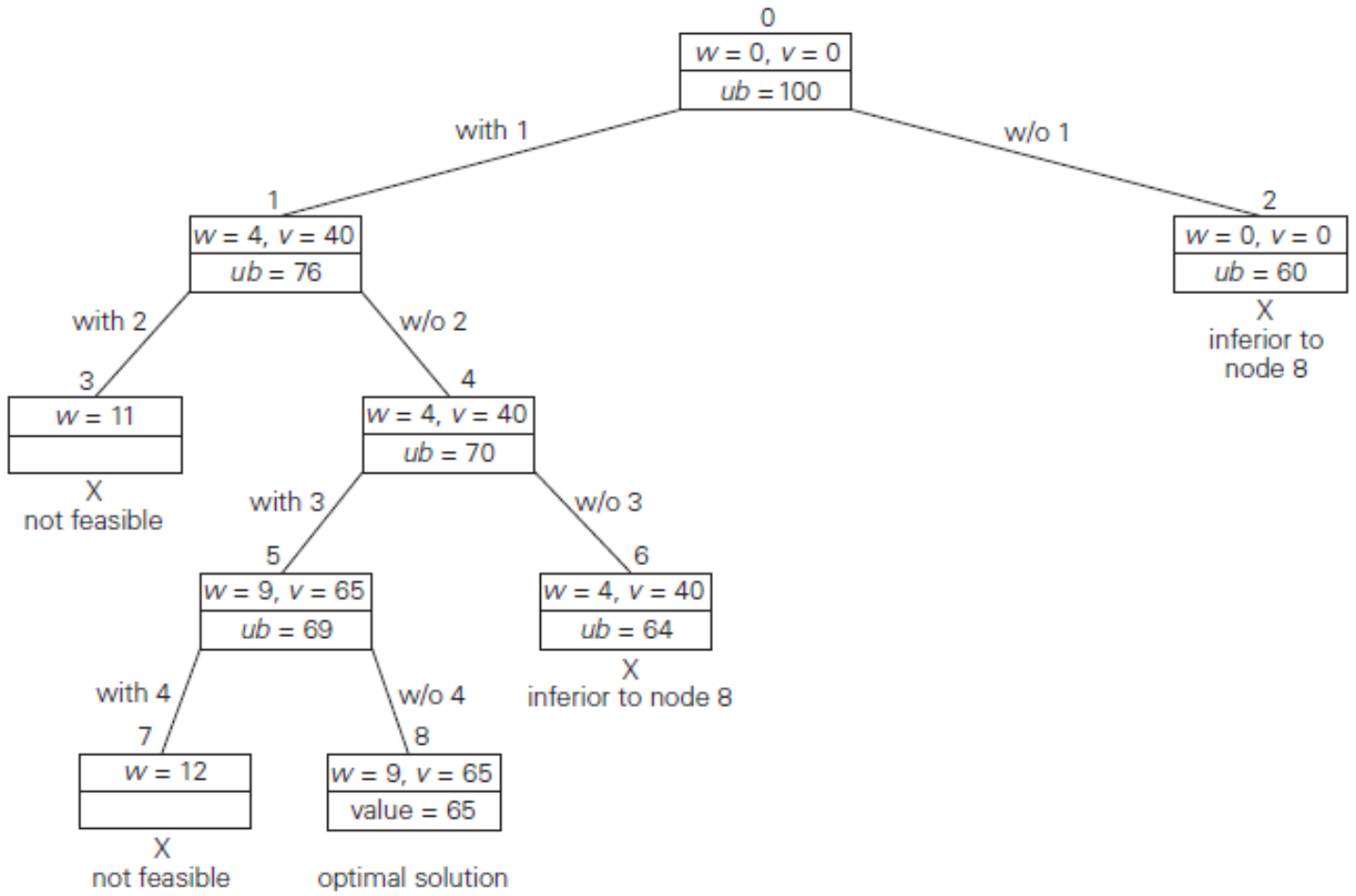
The knapsack problem

- Παράδειγμα:

item	weight	value	value / weight
1	4	$40	10
2	7	$42	6
3	5	$25	5
4	3	$12	4

The knapsack's capacity W is 10

$$ub = v + (W - w)(v_{i+1}/w_{i+1})$$



Χειρισμός Περιορισμών (30/32)

The traveling salesman problem

- Προσπαθούμε να ορίσουμε ένα κάτω όριο για το μήκος της διαδρομής
- Ένας απλός τρόπος είναι να βρούμε τη μικρότερη απόσταση και να την πολλαπλασιάσουμε με το πλήθος των πόλεων n
- Άλλος τρόπος είναι να το υπολογίσουμε ως εξής:
  - Για κάθε πόλη i βρίσκουμε το άθροισμα s_i των αποστάσεων από την πόλη i στις δύο πιο κοντινές πόλεις
  - Αθροίζουμε όλες αυτές τις αποστάσεις (n αριθμοί) και διαιρούμε προς 2

$$lb = \lceil s/2 \rceil$$

Χειρισμός Περιορισμών (31/32)

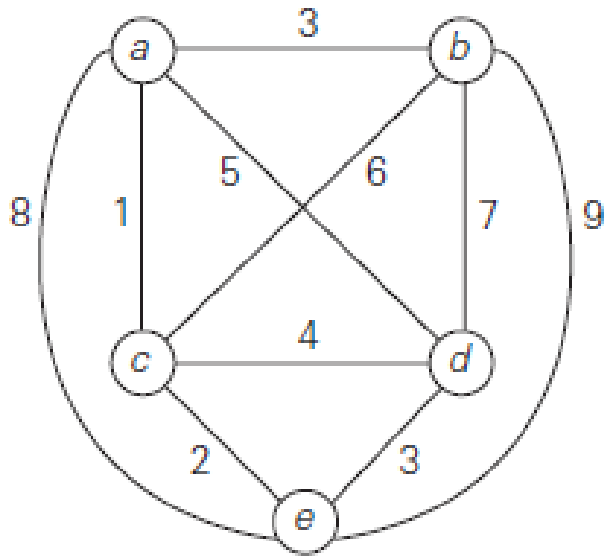
The traveling salesman problem

- Παράδειγμα
  - Για το διπλανό γράφο έχουμε

$$lb = \lceil [(1+3)+(3+6)+(1+2)+(3+4)+(2+3)]/2 \rceil = 14$$

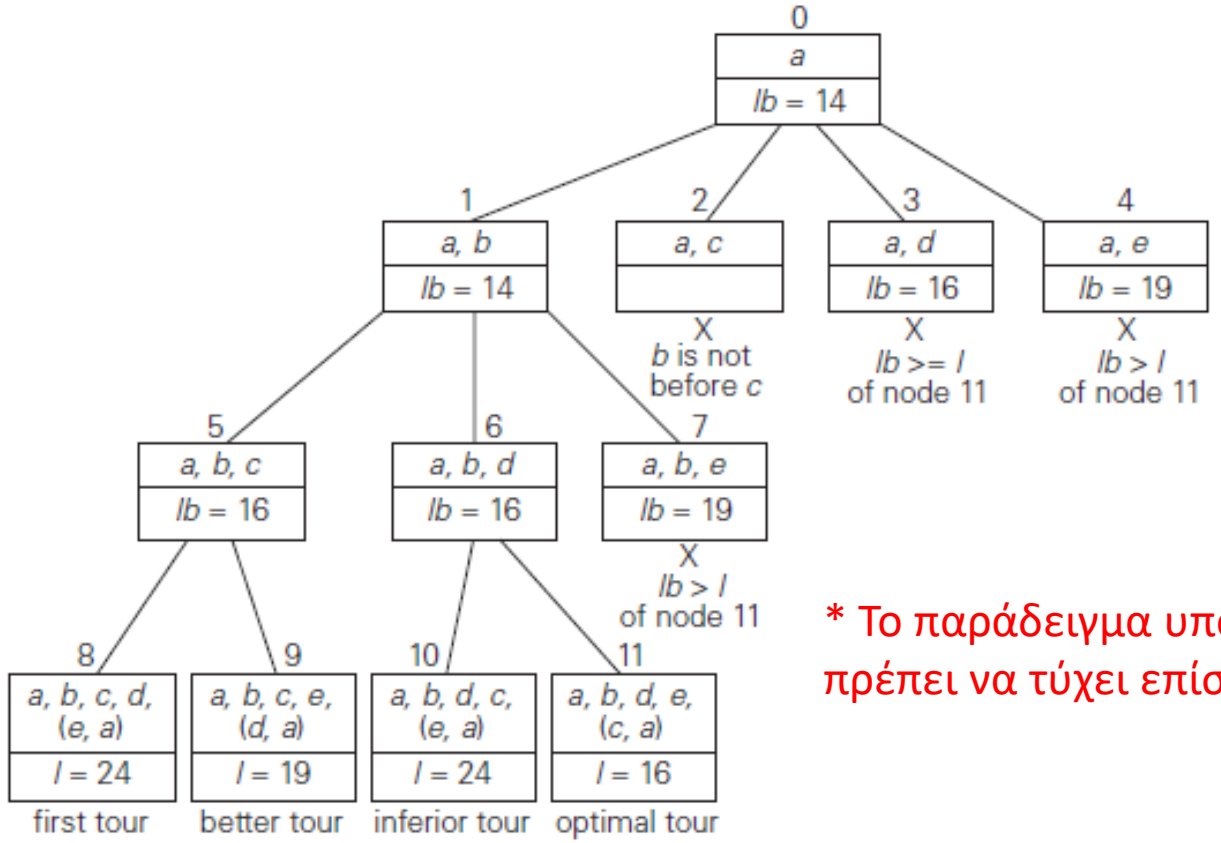
- Τροποιούμε το lb για κάθε υποσύνολο διαδρομών που πρέπει να περιλαμβάνουν μια συγκεκριμένη ακμή
- Παράδειγμα: για όλα τα Hamiltonian circuits που πρέπει να περιλαμβάνουν την ακμή (a,d), παίρνουμε το ακόλουθο lb

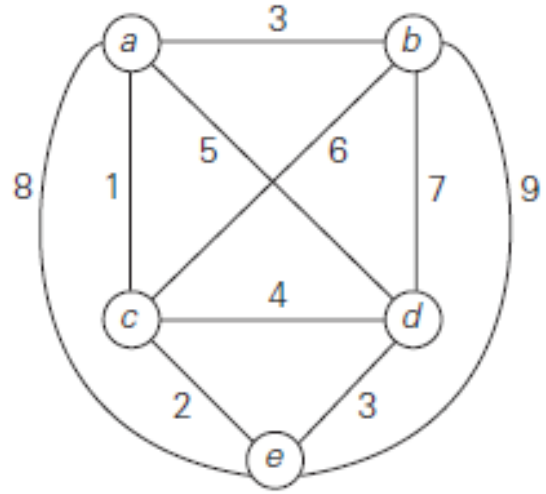
$$\lceil [(1+5)+(3+6)+(1+2)+(3+5)+(2+3)]/2 \rceil = 16$$



Χειρισμός Περιορισμών (32/32)

The traveling salesman problem





* Το παράδειγμα υποθέτει ότι ο κόμβος b πρέπει να τύχει επίσκεψης μετά τον a.