Ελάχιστα διασυνδετικά δένδρα (Minimum spanning trees)

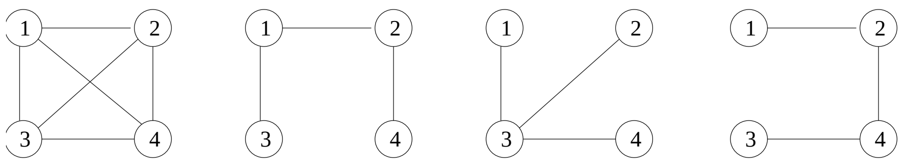
Αφορμή για την εξέτασή τους

• Έχουν άπληστες λύσεις.

• Βρίσκουν εφαρμογή σε πολλά πραγματικά προβλήματα.

Ορισμός Έστω $G = (V, E)$ ένα μη-κατευθυνόμενο συνδεδεμένο γράφημα. Ένα υπογράφημα $T = (V, E')$ του G είναι ένα *διασυνδετικό δένδρο (spanning tree)* του G **εάν και μόνο εάν** το T είναι ένα δένδρο.

Παράδειγμα Ένα πλήρες γράφημα 4 κόμβων και 3 από τα διασυνδετικά δένδρα του.



• Εάν σε κάθε ακμή του γραφήματος προσυνάψουμε ένα βάρος, τότε μας ενδιαφέρει να υπολογίσουμε ένα *διασυνδετικό δένδρο ελαχίστου βάρους (minimum-weight spanning tree)*.

• Το συνολικό βάρος του διασυνδετικού δένδρου T ενός γραφήματος G είναι:

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

όπου e είναι μία ακμή και $w(e)$ είναι το βάρος της.

Εφαρμογή Στο σχεδιασμό ηλεκτρονικών κυκλωμάτων, συχνά μας ενδιαφέρει να καταστήσουμε n ακίδες (pins) ηλεκτρικά ισοδύναμες μέσω της καλωδίωσης τους. Θέλουμε να υπολογίσουμε μία 'καλή' καλωδίωση, δηλαδή, μια καλωδίωση ελάχιστου κόστους Το κόστος της ένωσης δύο οποιονδήποτε ακίδων θεωρείται δεδομένο.

Λύση

• Κάθε ακίδα αναπαριστάται από ένα κόμβο.

• Κάθε πιθανή ένωση αναπαρίσταται από μία ακμή κατάλληλου βάρους.

Τότε, αρκεί να ευρεθεί ένα ελάχιστο διασυνδετικό δένδρο.

Ανάπτυξη ενός ελάχιστου διασυνδετικού δένδρου

Είσοδος: Ένα συνδεδεμένο μη-κατευθυνόμενο γράφημα $G = (V, E)$ και μία συνάρτηση βάρους $w: \; E \rightarrow \mathcal{R}$.

Έξοδος: Ένα ελάχιστο διασυνδετικό δένδρο για το G.

- **Ένας γενικός αλγόριθμος**

  Generic_MST(G, w)
  $A = \emptyset$
  while A does not form a spanning tree **do**
      Find an edge (u, v) that is safe for A.
      $A = A \cup \{(u, v)\}$
  return A

Σταθερά (invariant) του αλγορίθμου ‘Το A είναι πάντοτε ένα υποσύνολο κάποιου ελάχιστου διασυνδετικού δένδρου.’

- Μία ακμή (u, v) είναι μία *ασφαλής ακμή (safe edge)* για το A εάν μπορεί να προστεθεί στο A χωρίς να διαταραχθεί η σταθερά του αλγορίθμου.

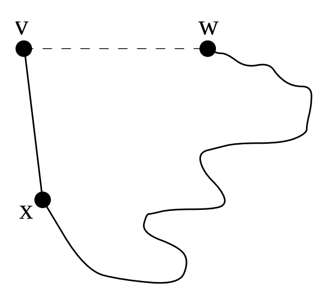
  Εάν το A δεν έχει ήδη σχηματίσει ένα (ελάχιστο) διασυνδετικό δένδρο, τότε πάντοτε μπορούμε να υπολογίσουμε μία ασφαλή για το A ακμή.

Υπολογισμός μίας ασφαλής ακμής για το A

Λήμμα Θεωρήστε ένα οποιοδήποτε κόμβο v. Έστω (v, w) μία ελαχίστου βάρους, προσκείμενη στον v, ακμή. Τότε, υπάρχει ένα ελάχιστο διασυνδετικό δένδρο το οποίο περιέχει την ακμή (v, w).

Απόδειξη (Με εις άτοπο επαγωγή)

- Υποθέστε ότι ένα τέτοιο δένδρο δεν υπάρχει.
- Θεωρήστε ένα οποιοδήποτε ελάχιστο διασυνδετικό δένδρο T.
  Έστω (v, x) η ακμή του T που είναι προσκείμενη στον κόμβο v.

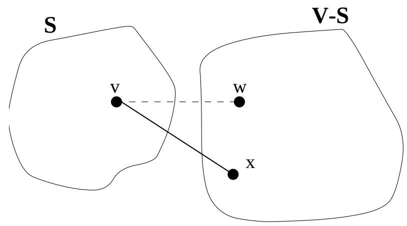


- Προσθέτοντας την (v, w) στο T δημιουργούμε ένα κύκλο. Ο κύκλος αυτός μπορεί να ‘σπάσει’ με την αφαίρεση της (v, x). Έτσι δημιουργείται ένα νέο διασυνδετικό δένδρο $T' = T - \{(v, x)\} \cup \{(v, w)\}$.
- $w(T') \leq w(T)$ διότι $w((v, w)) \leq w((v, x))$. Άρα, το T' περιέχει την (v, w) και επιπλέον δεν έχει βάρος μεγαλύτερο από το ελάχιστο διασυνδετικό δένδρο T. Μία καθαρή αντίφαση. $\square$

$\Rightarrow$ Από το παραπάνω λήμμα, **γνωρίζουμε πώς να εντοπίσουμε μία ασφαλή ακμή.**

Λήμμα Έστω A ένα υποσύνολο των ακμών ενός ελάχιστου διασυνδετικού δένδρου και έστω S το σύνολο των προσκείμενων στις ακμές του A κόμβων. Έστω (v, w) μία ελάχιστου βάρους ακμή τέτοια ώστε $v \in S$ και $w \in V - S$. Τότε, η (v, w) είναι μία ασφαλής ακμή για το A.

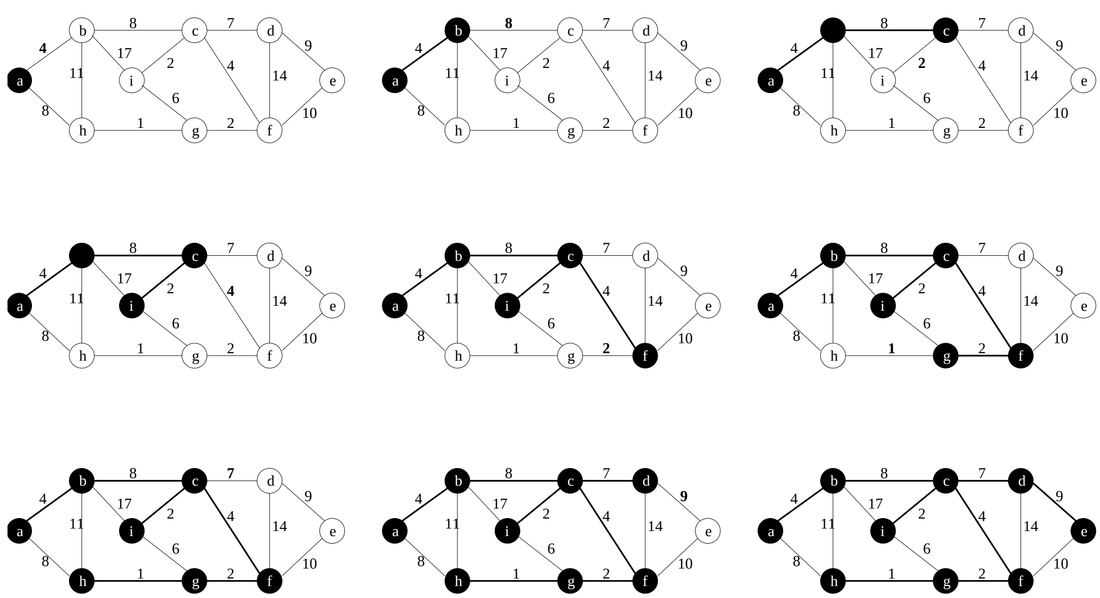
Απόδειξη (Με εις άτοπο επαγωγή)



- Έστω ότι το T είναι ένα ελάχιστο διασυνδετικό δένδρο και η ακμή του (v, x) η οποία ενώνει το S με το $V - S$.
- Το $T' = T - \{(v, x)\} \cup \{(v, w)\}$ είναι ένα διασυνδετικό δένδρο, ίσου ή μικρότερου βάρους με το T, το οποίο επιπλέον περιέχει την ακμή (v, w). ία καθαρή αντίφαση.

$\square$

Παράδειγμα



Ο αλγόριθμος του Prim

- Κατά την εκτέλεση του αλγόριθμου, όλοι οι κόμβοι που δεν έχουν εισαχθεί στο δένδρο βρίσκονται σε μία ουρά προτεραιότητας Q με κλειδί το πεδίο key.
- Για κάθε κόμβο $v \in V, key[v]$ είναι το ελάχιστο βάρος της ακμής που ενώνει τον κόμβο v με έναν κόμβο του δένδρου.
- Το πεδίο $\pi[v]$ δηλώνει τον 'πατέρα' του v στο δένδρο.

```
MST_prim(G, w, r)     /* r is the root of the spanning tree */
for each u ∈ V do      key[u] = +∞
key[r] = 0
π[r] = nil
Q = V
while Q ≠ ∅ do
    u = extract_min(Q)
    for each v ∈ adjacent_list(u) do
        if v ∈ Q and w[u, v] < key[v] then
            π[v] = u
            key[v] = w(u, v)
```

Ανάλυση: $O(V \log V + E \log V) = O(E \log V)$ εάν η ουρά προτεραιότητας υλοποιηθεί σαν δυαδικός σωρός.

$O(V \log V + E)$ εάν η ουρά προτεραιότητας υλοποιηθεί σαν σωρός Fibonacci.

Συμπέρασμα

- Πολλά προβλήματα βελτιστοποίησης μπορεί να λυθούν από ένα τετριμμένο αλγόριθμο. Συνήθως, ο αλγόριθμος αυτός **δεν** γρήγορος.

Παραδείγματα

- Ελάχιστα διασυνδετικά δένδρα

  Κατασκεύασε όλα τα διασυνδετικά δένδρα.
  Προσδιόρισε το δένδρο με το ελάχιστο βάρος.

  Ανάλυση: $O\left(C\left(|E|, n-1\right) \cdot n\right)$

- Ελάχιστα μονοπάτια από κοινή αφετηρία (με μη αρνητικά βάρη)

  Για κάθε κόμβο v, κατασκεύασε όλα τα μονοπάτια από την αφετηρία s προς τον κόμβο v.
  Προσδιόρισε το μονοπάτι ελάχιστο βάρους.

  Ανάλυση: $O\left(C\left(|E|, n-1\right) \cdot n \cdot n\right)$

- Η εξεύρεση ιδιοτήτων που διέπουν την βέλτιστη λύση οδηγεί σε γρήγορους αλγόριθμους.