Αποδοτικοί Αλγόριθμοι

Μέρος 3ο: Ελάχιστα Διασυνδετικά Δέντρα

Εξάμηνο: 7ο

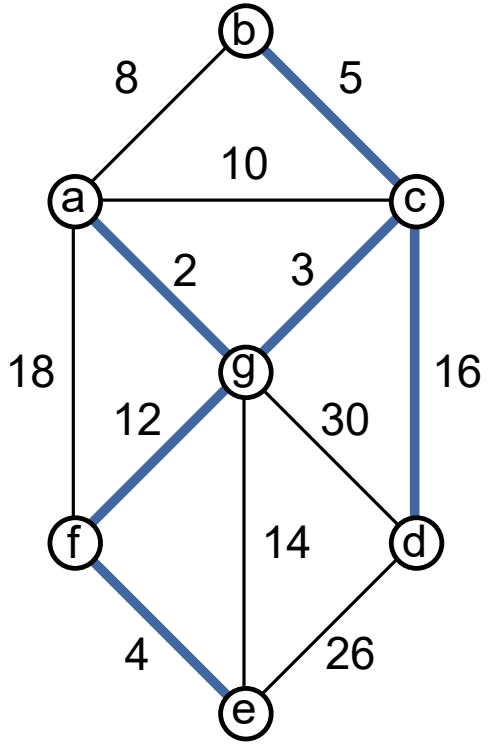
Κωδικός μαθήματος: 748

Τμήμα Μαθηματικών
Πανεπιστήμιο Ιωαννίνων

Μιχάλης Α. Μπέκος
bekos@uoi.gr

Ελάχιστα Διασυνδετικά Δέντρα

- Είσοδος: Ένα συνδεδεμένο, μη κατευθυνόμενο γράφημα $G = (V, E)$
  μια συνάρτηση κόστους $c : E \to \mathbb{R}^+$
- Έξοδος: Ενα ελάχιστο διασυνδετικό δέντρο T του G:
  - $T \leftarrow$ συνδεδεμένο και ακυκλικό
  - $V[T] = V$ και $E[T] \subseteq E$
  - Για κάθε διασυνδετικό δέντρο T^* του G:
    $c(T) = \sum_{e \in E[T]} c(e) \leq c(T^*) = \sum_{e \in E[T^*]} c(e)$



- Άπληστες προσεγγίσεις:
  - ένα ελάχιστο διασυνδετικό δέντρο κατασκευάζεται σε βήματα
  - κάθε βήμα είναι τοπικά βέλτιστο
  - κανένα βήμα δεν επαναφέρεται

Ένας άπληστος αλγόριθμος

- Σε κάθε βήμα, αποδίδουμε ένα χρώμα σε μια ακμή (μπλε ή κόκκινο) του G, έτσι ώστε:

  "υπάρχει ελάχιστο διασυνδετικό δέντρο που περιέχει όλες τις μπλε αλλά καμία κόκκινη ακμή"

- Επόμενα βήματα:
  - ...πρέπει να ορίσουμε τους κανόνες χρωματισμού
  - ...και να αποδείξουμε ότι η παραπάνω συνθήκη ισχύει για αυτούς τους κανόνες.

- Ορισμοί:
  - Μια τομή του G είναι μια διαμέριση του V σε δύο σύνολα X και $V \setminus X$ έτσι ώστε $X \neq \emptyset \neq V \setminus X$
  - Μια ακμή (u, v) διαπερνά την τομή εάν $u \in X$ και $v \in V \setminus X$

Οι κανόνες χρωματισμού

- Ο μπλε κανόνας:

  Έστω $(X, V \setminus X)$ μια τομή που δεν διαπερνάται από μπλε ακμές (αν υπάρχει).

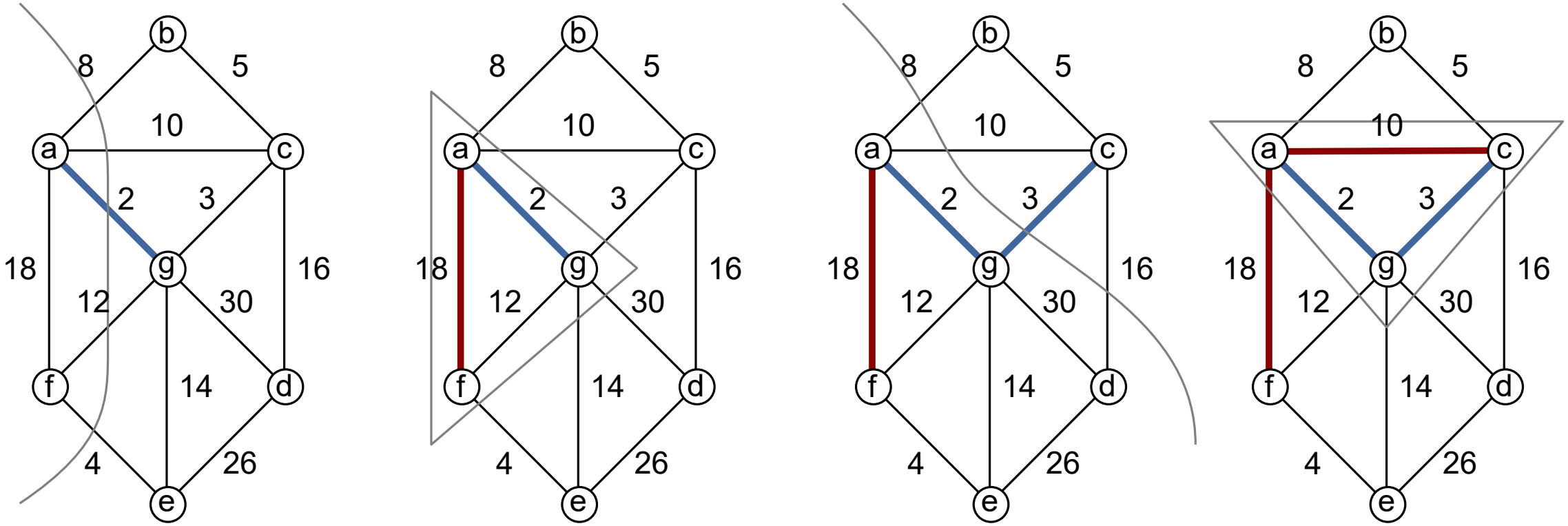
  Χρωμάτισε μπλε την αχρωμάτιστη ακμή ελάχιστου κόστους της τομής $(X, V \setminus X)$

- Ο κόκκινος κανόνας:

  Έστω C ένας απλός κύκλος χωρίς κόκκινες ακμές (αν υπάρχει).

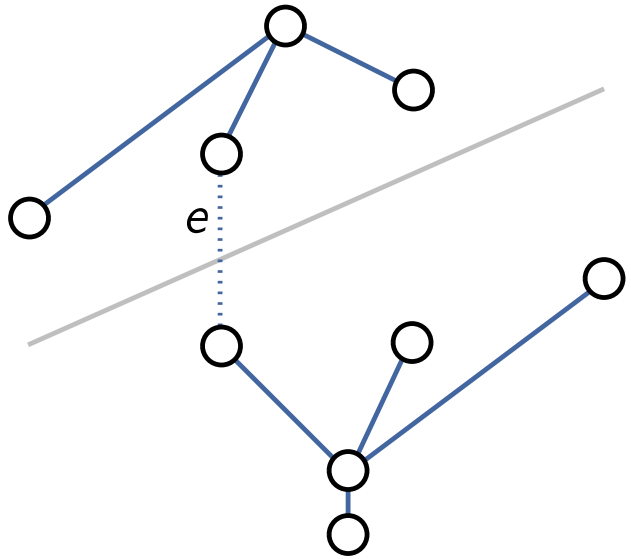
  Χρωμάτισε κόκκινη την αχρωμάτιστη ακμή μέγιστου κόστους του κύκλου C

Πίσω στο παράδειγμα...



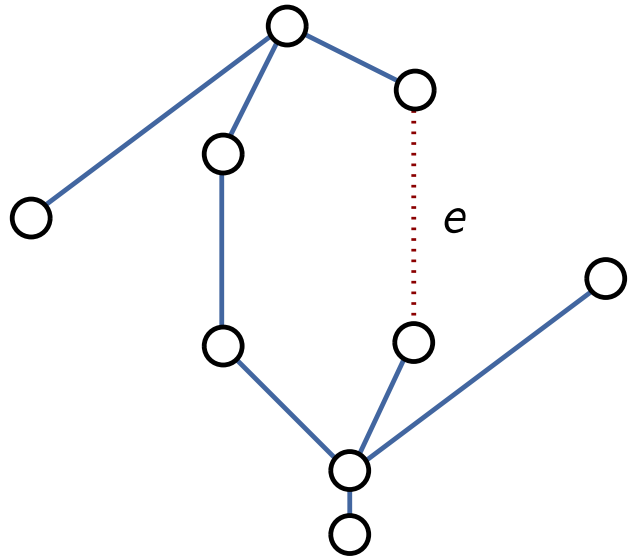
Ο αλγόριθμος μπορεί πάντα να συνεχιστεί

- Έστω *e* μια αχρωμάτιστη ακμή:
  - Αν η *e* συνδέει δύο μπλε υποδέντρα, υπάρχει τομή που δεν διαπερνάται από μπλε ακμή



Εφαρμογή του μπλε κανόνα

  - Διαφορετικά, η *e* κλείνει έναν κύκλο, στον οποίο η *e* είναι η μόνη αχρωμάτιστη ακμή



Εφαρμογή του κόκκινου κανόνα

Ορθότητα

- Θα αποδείξουμε ότι η σταθερά του αλγορίθμου ισχύει, όταν εφαρμόζεται ένας κανόνας:

  “υπάρχει ελάχιστο διασυνδετικό δέντρο που περιέχει όλες τις μπλε αλλά καμία κόκκινη ακμή”

  …με επαγωγή στο πλήθος των χρωματισμένων ακμών.

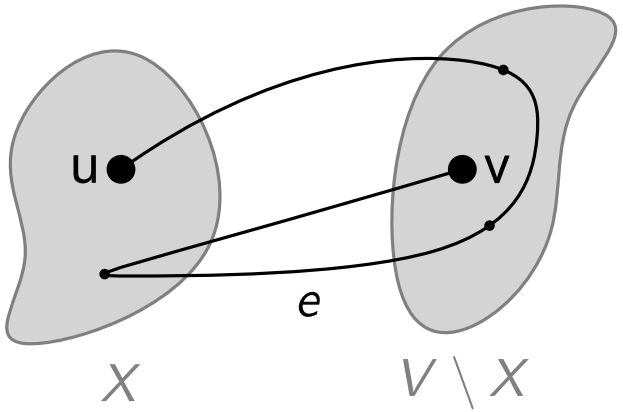
  - Βάση: Αν έχουν χρωματιστεί μηδέν ακμές, τότε η σταθερά τετριμμένα ισχύει.
  - Ε.Β.: $(u, v) \leftarrow$ η επόμενη ακμή που θα χρωματιστεί

    $T \leftarrow$ το δέντρο που ικανοποιεί τη σταθερά πριν το χρωματισμό της (u, v) λόγω Ε.Υ.
  - Περιπτώσεις: Η (u, v) χρωματίζεται είτε μπλε είτε κόκκινη

Ορθότητα: Η (u, v) χρωματίζεται μπλε

…εξαιτίας μιας τομής $(X, V \setminus X)$ με $u \in X$ και $v \in V \setminus X$

- Αν $(u, v) \in T$, τότε τελειώσαμε.
- Έστω ότι $(u, v) \notin T \Rightarrow$ υπάρχει μονοπάτι $P : u \to v$ στο T.
  - Το P έχει μια ακμή e που διαπερνά την τομή $(X, V \setminus X)$
    - η e δεν είναι κόκκινη (το δέντρο T δεν έχει κόκκινες ακμές)
    - η e δεν είναι μπλε (η τομή $(X, V \setminus X)$ δεν διαπερνάται από μπλε ακμές)



  - Η e είναι αχρωμάτιστη $\Rightarrow cost(u, v) \leq cost(e)$.
  - Το T είναι ελάχιστο διασυνδετικό $\Rightarrow cost(u, v) = cost(e)$.
  - Αντάλλαξε την e με την $(u, v) \Rightarrow$ νέο ελάχιστο διαδυνδετικό δέντρο που περιέχει την (u, v)

Ορθότητα: Η (u, v) χρωματίζεται κόκκινη

...εξαιτίας ενός κύκλου C

- Αν $(u, v) \notin T$, τότε τελειώσαμε.
- Έστω ότι $(u, v) \in T$
  - Εφόσον το T είναι ακυκλικό, $\exists e \in C : e \notin T$.
    - η e δεν είναι κόκκινη, καθώς η e ανήκει στον C και ο C δεν έχει κόκκινες ακμές.
    - η e δεν είναι μπλε, καθώς η e δεν ανήκει στο T.
  - Η e είναι αχρωμάτιστη $\Rightarrow cost(u, v) \geq cost(e)$.
  - Το T είναι ελάχιστο διασυνδετικό $\Rightarrow cost(u, v) = cost(e)$.
  - Αντάλλαξε την e με την $(u, v) \Rightarrow$ νέο ελάχιστο διαδυνδετικό δέντρο που δεν περιέχει την (u, v)

Ο άπληστος αλγόριθμος του Kruskal

- ```
  Sort the edges by increasing cost
  T ← ∅
  For each edge e in increasing cost order do {
      If adding e to T does not form a cycle {
          add e to T and color it blue
      }
      else {
          color it red
      }
  }
  ```
  διατηρεί την ακυλικότητα
- Ορθότητα:
  - Εάν η e κλείνει έναν κύκλο, τότε λόγω της ταξινόμησης, η e έχει μέγιστο κόστος. Εφαρμογή του κόκκινου κανόνα.
  - Διαφορετικά, η e συνδέει δύο συνιστώσες $\Rightarrow$ η e έχει ελάχιστο κόστος σε αυτή την τομή. Εφαρμογή του μπλε κανόνα.
  ```

Ο άπληστος αλγόριθμος του Kruskal: Ανάλυση πολυπλοκότητας

- Η δομή δεδομένων union/find:
  - `union(A,B)`: ενώνει δύο ξένα σύνολα A και B
  - `find(x)`: επιστρέφει το όνομα του συνόλου που περιέχει το x

  και οι δύο μπορούν να υλοποιηθούν σε χρόνο $O(a(m, n))$, όπου $a(m, n)$ είναι η συνάρτηση Ackermann - σχεδόν σταθερά

- ```
  Sort the edges by increasing cost
  T ← ∅
  For each edge (u,v) in increasing cost order do {
     If find(u) ≠ find(v)
     {
        add (u,v) to T;
        Union(find(u), find(v));
     }
  }
  ```

  $O(m \log m)$

  $O(a(m, n) \cdot m)$
  ```

Ο άπληστος αλγόριθμος του Prim

- ```
  Initialize T to a vertex of G
  While T is not spanning do
  {
      Add to T the cheapest uncolored edge incident to T
      and color it blue ⟶ διατηρεί τη συνδεσιμότητα
  }
  ```

- Ορθότητα:
  - Εκτελεί επανειλημμένα τον μπλε κανόνα.
  ```

Ο άπληστος αλγόριθμος του Prim: Ανάλυση πολυπλοκότητας

- Δομή δεδομένων ουρά προτεραιότητας:
  - `insert(x)`: εισάγει το στοιχείο `x` στη δομή δεδομένων
  - `delete_min()`: διαγράφει το ελάχιστο στοιχείο από τη δομή δεδομένων
  - `decrease_key(x,k)`: μειώνει το κλειδί του στοιχείου `x` σε `k`.

Εάν η δομή δεδομένων υλοποιείται ως δυαδικός σωρός,
τότε κάθε μέθοδος υλοποιείται σε χρόνο $O(n \log n)$.

- Ιδέα για την υλοποίηση:
  - Διατηρεί ένα σύνολο D με κορυφές που πρέπει να διασυνδεθούν.
  - $V \setminus D \leftarrow$ κορυφές που έχουν διασυνδεθεί.
  - Το κλειδί κάθε κορυφής του D είναι το βάρος της φθηνότερης ακμής μεταξύ αυτής και του $V \setminus D$.

Ο άπληστος αλγόριθμος του Prim: Ανάλυση πολυπλοκότητας

```
D ← ∅; // D is a binary heap
Foreach vertex v do {
    key(v) = +∞;
    π(v)=null;
    D.insert(v);
}
D.decrease_key(s, 0); // s is some vertex of G

While !D.isEmpty() do {
    u = D.delete_min();
    Foreach neighbor v of u do {
        if (key(v) > c(u,v)) {
            key(v) = c(u,v);
            π(v) = u;
            D.decrease_key(v, key(v);
        }
    }
}
```

$O(n \log n)$

$O(n \log n)$

$O(m \log n)$

Επιπλέον υλικό

- Κεφάλαιο 3 από τις σημειώσεις του μαθήματος.
  Διαθέσιμες στη σελίδα του μαθήματος στο ecourse.