Αποδοτικοί Αλγόριθμοι

Μέρος 4ο: Ελάχιστες τομές

Εξάμηνο: 7ο

Κωδικός μαθήματος: 748

Τμήμα Μαθηματικών
Πανεπιστήμιο Ιωαννίνων

Μιχάλης Α. Μπέκος
bekos@uoi.gr

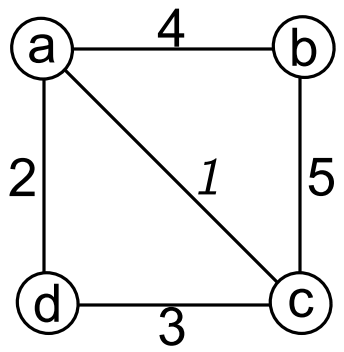
Το πρόβλημα της ελάχιστης τομής

- Είσοδος: Ένα συνδεδεμένο μη κατευθυνόμενο γράφημα $G = (V, E)$
  μια συνάρτηση κόστους $c : E \to \mathbb{R}^+$

- Έξοδος: Μια διαμέριση του V σε δύο σύνολα S και T, έτσι ώστε:
  - $S \cup T = V$ και $S \cap T = \emptyset$
  - $S \neq \emptyset \neq T$
  - Το κόστος $c(S, T) = \sum_{\substack{(u,v)\in E \\ u\in S, v\in T}} c(u, v)$ είναι ελάχιστο

- Παράδειγμα:

S	$\{a\}$	$\{b\}$	$\{c\}$	$\{d\}$	$\{a, b\}$	$\{a, c\}$	$\{a, d\}$	$\{b, c\}$	$\{b, d\}$	$\{c, d\}$
$c(S, V \setminus S)$	7	9	8	5	8	14	8	8	14	8



Το πρόβλημα της ελάχιστης (s,t)-τομής

- Είσοδος: Ένα συνδεδεμένο μη κατευθυνόμενο γράφημα $G = (V, E)$
  μια συνάρτηση κόστους $c : E \to \mathbb{R}^+$
  δύο προκαθορισμένες κορυφές s, t V.

- Έξοδος: Μια διαμέριση του V σε δύο σύνολα S και T, έτσι ώστε:
  - $S \cup T = V$ και $S \cap T = \emptyset$
  - $S \neq \emptyset \neq T$ με $s \in S$ και $t \in T$
  - Το κόστος $c_{st}(S, T) = \sum_{\substack{(u,v)\in E \\ u\in S, v\in T}} c(u, v)$ είναι ελάχιστο

- Παρατήρηση: $c_{st}(S, T) \geq c(S, T)$

  Απόδειξη: Κάθε (s, t)-τομή είναι τομή του G.

Μια απλή τετραγωνική αναγωγή

- Μια απλή $O(n^2)$ αναγωγή:

```
MinimumCut(G)
{
   minimum = +∞;
   foreach vertex s of G
      foreach vertex t of G s.t. t≠s
         minimum = min{minimum, MinimumCut(G,s,t)}
   return minimum;
}
```

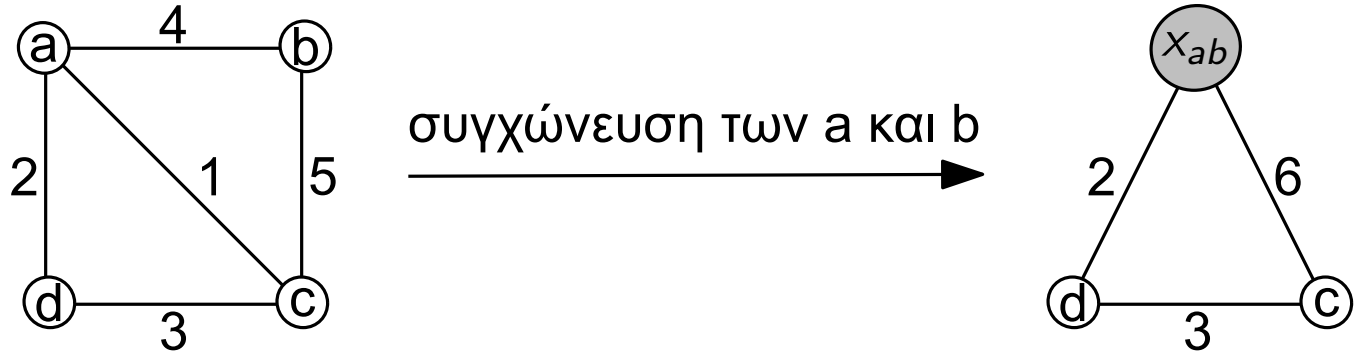
↑ υπολογίζει μια ελάχιστη (s,t)-τομή

- Ορθότητα: Τετριμμένη καθώς εξετάζονται όλα τα ζεύγη
- Ερώτηση: Μπορεί να βελτιωθεί ο τετραγωνικός όρος στην αναγωγή;

Συγχώνευση δύο κορυφών

- Ορισμός: Δοθέντος ενός γραφήματος $G = (V, E, c)$, το γράφημα $G|\{s, t\} = (V', E', c')$ που προκύπτει από τη συγχώνευση των $s, t \in V$ ορίζεται ως εξής:
  - $V' = V \setminus \{s, t\} \cup x_{st}$
  - $E' = E \setminus \{(u, v) \in E : u = s \text{ ή } v = t\} \cup \{(x_{st}, v) : (s, v) \in E \text{ ή } (v, t) \in E \text{ με } v \neq s, t\}$
  - $$c'(x_{uv}, w) = \begin{cases} c(u, w) & \text{αν } (u, w) \in E, (v, w) \notin E, \\ c(v, w) & \text{αν } (v, w) \in E, (v, w) \notin E, \\ c(u, w) + c(v, w) & \text{αν } (u, w) \in E, (v, w) \in E. \end{cases}$$

- Παράδειγμα:



Συγχώνευση δύο κορυφών

- Θεώρημα: Έστω $G = (V, E, c)$ ένα γραφήμα και έστω $G|\{s, t\}$ το γραφήμα που προκύπτει από τη συγχώνευση των $s, t \in V$. Τότε:

  `MinimumCut(G) = min{MinimumCut(G|{s,t}), MinimumCut(G, s, t)}`

  Απόδειξη:

  Έστω (S, T) μια ελάχιστη τομή του G. Τότε:

  - ...είτε $s, t \in S \Rightarrow$ `MinimumCut(G) = MinimumCut(G|{s,t})`
  - ...είτε $s \in S$ και $t \in T \Rightarrow$ `MinimumCut(G) = MinimumCut(G, s, t)`

Μια πιο δύσκολη γραμμική αναγωγή

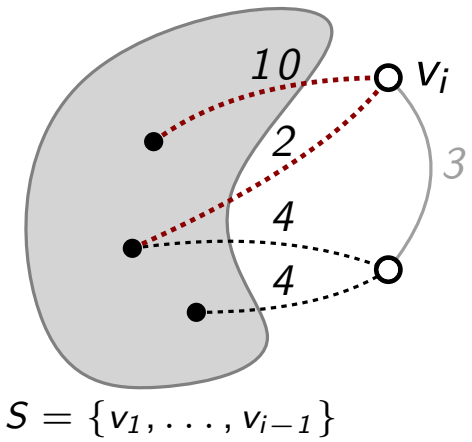
- Μια $O(n)$ αναγωγή:

```
MinimumCut(G) {
   if |V(G)| = 2 then
      return (v1, v2);
   else {
      s,t <- any two vertices of G;
      (S1,T1) <- MinimumCut(G, s, t);   ⟶ ελάχιστη (s,t)-τομή
      (S2,T2) <- MinimumCut(G|{s,t});   ⟶ αναδρομική κλήση
      return the lighter of (S1,T1) and (S2,T2);
   }
}
```

- Ορθότητα: Έπεται από το προηγούμενο θεώρημα.
- Πολυπλοκότητα: O(n) · O(min (s,t)-τομή).

Ο αλγόριθμος των Stoer και Wagner

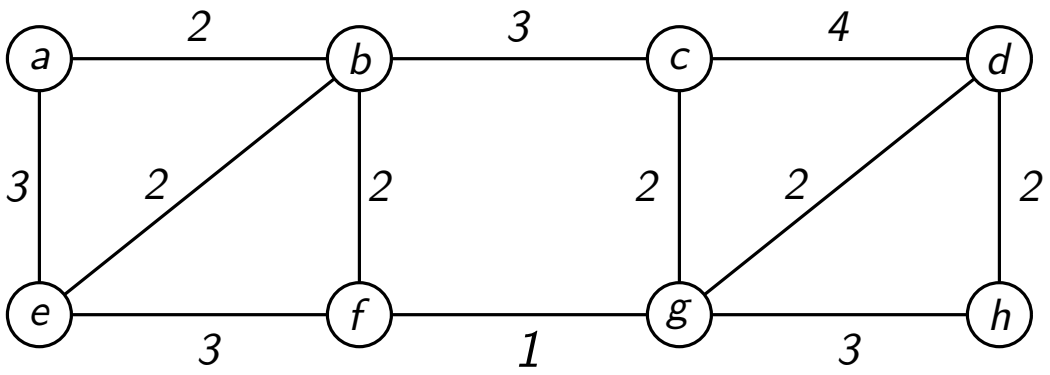
- Διατηρεί ένα σύνολο S επιλεγμένων κορυφών
- Το S αρχικοποιείται σε μία οποιαδήποτε κορυφή του G, έστω v_1
- Δημιουργεί μια διάταξη $v_1, \ldots, v_n$ προσθέτοντας στο S την κορυφή μέγιστου βάρους προς το S



- Ισχυρισμός: Η $(v_1, \ldots, v_{n-1}, v_n)$ είναι μια ελάχιστη (v_{n-1}, v_n)-τομή

συνεπώς, έχουμε έναν αλγόριθμο για τον υπολογισμό ελάχιστων τομών.

Παράδειγμα



Ξεκινάμε με $v_1 = b$. Ακολούθως:

$v_2 = c, v_3 = d, v_4 = g, v_5 = h, v_6 = f, v_7 = e$

Αν ισχύει ο ισχυρισμός, η υπολογισθείσα τομή είναι μια ελάχιστη (a, e)-τομή με κόστος 5.

Ορθότητα

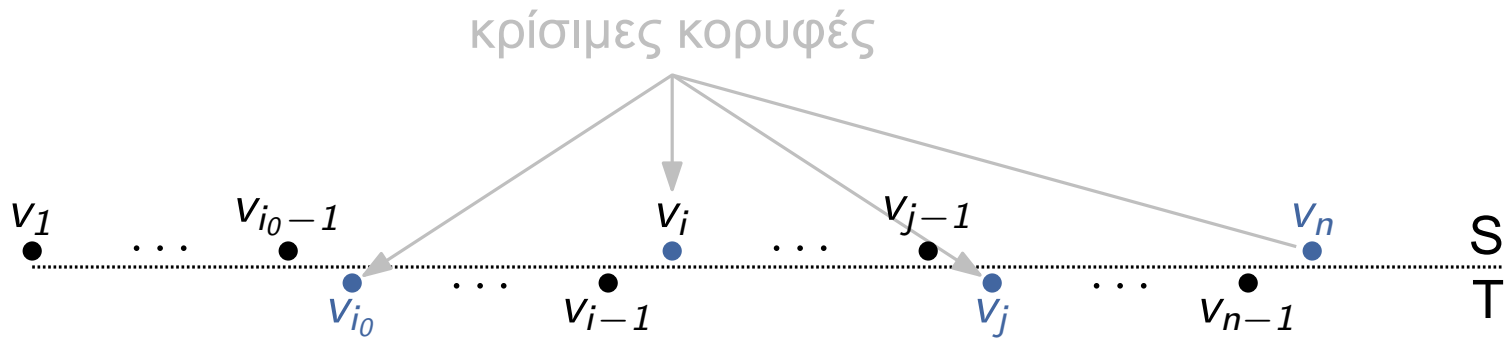
- Ισχυρισμός: Η $(v_1, \ldots, v_{n-1}, v_n)$ είναι μια ελάχιστη (v_{n-1}, v_n)-τομή

  Απόδειξη:

  Έστω (S, T) οποιαδήποτε (v_{n-1}, v_n)-τομή

  Θα αποδείξουμε ότι $c(\{v_1, \ldots, v_{n-1}\}, \{v_n\}) \leq c(S, T)$

  Ορισμός: η v_i είναι κρίσιμη $\iff$ ($v_i \in S$ και $v_{i-1} \in T$) ή ($v_i \in T$ και $v_{i-1} \in S$)



Νέος ισχυρισμός: Για κάθε κρίσιμη κορυφή v_i: → αφού η v_n είναι κρίσιμη, ο νέος ισχυρισμός συνεπάγεται τον παλιό

$$c(\{v_1, v_2, \ldots, v_{i-1}\}, \{v_i\}) \leq c(\{v_1, v_2, \ldots, v_i\} \cap S, \{v_1, v_2, \ldots, v_i\} \cap T)$$

Ορθότητα

- Νέος ισχυρισμός: $c(\{v_1, v_2, \ldots, v_{i-1}\}, \{v_i\}) \leq c(\{v_1, v_2, \ldots, v_i\} \cap S, \{v_1, v_2, \ldots, v_i\} \cap T)$

  Απόδειξη: με επαγωγή στο πλήθος των κρίσιμων κορυφών

  - Για την πρώτη κρίσιμη κορυφή, ο ισχυρισμός ισχύει ως ισότητα.
  - Ας υποθέσουμε ότι ο ισχυρισμός ισχύει για την κρίσιμη κορυφή v_i
  - Έστω v_j η επόμενη κρίσιμη κορυφή. Τότε:

$$
\begin{aligned}
c(\{v_1, \ldots v_{j-1}\}, \{v_j\}) &= c(\{v_1, \ldots, v_{i-1}\}, \{v_j\}) + c(\{v_i, \ldots, v_{j-1}\}, \{v_j\}) \\
&\downarrow \text{ λόγω του τρόπου με τον οποίο ο αλγόριθμος διατάσσει τις κορυφές} \\
&\leq c(\{v_1, \ldots, v_{i-1}\}, \{v_i\}) + c(\{v_i, \ldots, v_{j-1}\}, \{v_j\}) \\
&\downarrow \text{ λόγω της επαγωγικής υπόθεσης} \\
&\leq c(\{v_1, \ldots, v_i\} \cap S, \{v_1, \ldots, v_i\} \cap T) + c(\{v_i, \ldots, v_{j-1}\}, \{v_j\}) \\
&\downarrow \text{ λόγω του ότι η } v_j \text{ είναι η τελευταία κρίσιμη κορυφή} \\
&\leq c(\{v_1, \ldots, v_j\} \cap S, \{v_1, \ldots, v_j\} \cap T)
\end{aligned}
$$

Ψευδοκώδικας

```
MinimumCut(G) {
    if |V(G)| = 2 then
        return (v1, v2);
    else {
        ({v_1,...,v_{n-1}}, {v_n}) <- MinimumCutPhase(G);   → επόμενη διαφάνεια
        (S,T) <- MinimumCut(G|{v_{n-1}, v_n});              → αναδρομική κλήση
        return the lighter of (S,T) and ({v_1,...,v_{n-1}}, {v_n});
    }
}
```

Ψευδοκώδικας

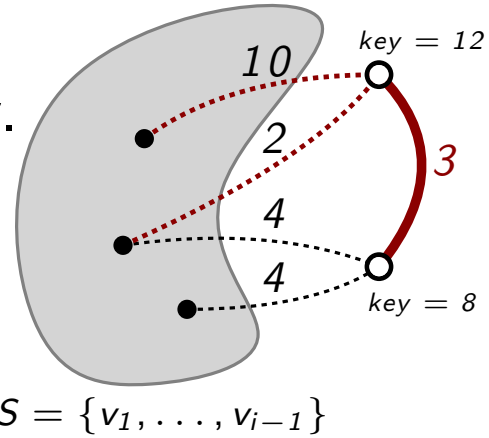
```
MinimumCutPhase(G) {
    n <- |V(G)|;
    v_1 <- any vertex of G;
    S <- {v_1};
    for i=2 to n do {
        v_i <- the vertex of V(G)\S s.t. c(S, {v}) is maximum
               overall vertices v of V(G)\S;
        S <- S ∪ {v_i};
    }
    return ({v_1,...,v_{n-1}}, {v_n});
}
```

Ανάλυση πολυπλοκότητας

- Δομή δεδομένων ουρά προτεραιότητας:
  - `insert(x)`: εισάγει το στοιχείο `x` στη δομή δεδομένων
  - `delete_max()`: διαγράφει το μέγιστο στοιχείο από τη δομή δεδομένων
  - `increase_key(x,k)`: αυξάνει το κλειδί του στοιχείου `x` σε `k`.

  Εάν η δομή δεδομένων υλοποιείται ως σωρός Fibonacci, τότε κάθε μέθοδος μπορεί να υλοποιηθεί σε χρόνο $O(n \log n)$, ενώ η τελευταία σε επιμερισμένο χρόνο O(1)

- Ιδέα για την υλοποίηση: Έστω για κάθε κορυφή $v \in V(G) \setminus S$: $\boxed{key(v) = \sum_{(v,s)\in E, s\in S} c(v,s)}$
  - Η επόμενη κορυφή v_i του S είναι αυτή με το μεγαλύτερο κλειδί.
  - Όταν η v_i προστεθεί στο S, τα κλειδιά των $V(G) \setminus S$ πρέπει να αυξηθούν.
  - Συνολικά, έχουμε το πολύ $O(m)$ αυξήσεις.



- MinimumCutPhase: $O(n \log n + m)$ χρόνο $\Rightarrow$ Συνολικά: $O(n^2 \log n + nm)$

Επιπλέον υλικό

- Κεφάλαιο 4 από τις σημειώσεις του μαθήματος.
  Διαθέσιμες στη σελίδα του μαθήματος στο ecourse.