ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΪΚΑ
ΜΑΘΗΜΑΤΑ

ΑΛΓΟΡΙΘΜΟΙ

Ενότητα 9: Άπληστοι Αλγόριθμοι

Ιωάννης Μανωλόπουλος, Καθηγητής
Αναστάσιος Γούναρης, Επίκουρος Καθηγητής
Τμήμα Πληροφορικής ΑΠΘ

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΪΚΑ
ΜΑΘΗΜΑΤΑ

Άπληστοι Αλγόριθμοι

Αλγόριθμος Prim, Αλγόριθμος Kruskal, Αλγόριθμος Dijkstra, Κώδικας Huffman

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άπληστοι αλγόριθμοι

Προβλήματα βελτιστοποίησης λύνονται με μια σειρά επιλογών που είναι:

- εφικτές
- τοπικά βέλτιστες
- αμετάκλητες

ΔΕΝ λύνονται όλα τα προβλήματα βελτιστο-ποίησης με αυτόν τον τρόπο

Εφαρμογή της άπληστης στρατηγικής

- Βέλτιστες λύσεις:
  - Υπολογισμός για ρέστα
  - Ελάχιστα ζευγνύοντα δένδρα (MST)
  - Ελάχιστα μονοπάτια (one-to-all)
  - Απλά προβλήματα δρομολόγησης
  - Κώδικες Huffman
- Προσεγγίσεις:
  - Πρόβλημα περιοδεύοντας πωλητή (TSP)
  - Πρόβλημα σάκου
  - Άλλα προβλήματα συνδυαστικής βελτιστοποίησης

Υπολογισμός για ρέστα

Έστω ότι υπάρχει «άπειρη διαθεσιμότητα»κερμάτων με αξία $d_1 > \ldots > d_m$, και το ζητούμενο είναι να δώσουμε ρέστα αξίας n με το μικρότερο αριθμό κερμάτων.

Παράδειγμα: d_1 = 25c, d_2 =10c, d_3 = 5c, d_4 = 1c and n = 48c

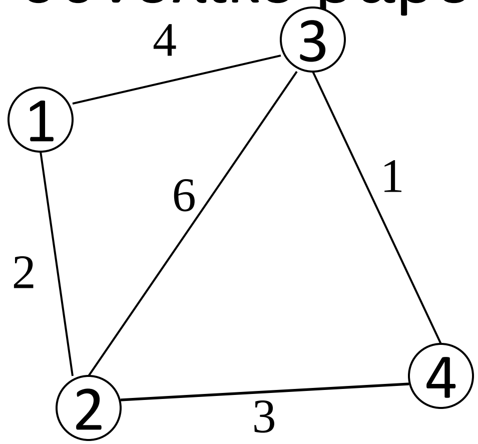
Άπληστη μέθοδος:

Η άπληστη μέθοδος

- Είναι βέλτιστη για κάθε n και φυσιολογικές τιμές υποδιαρέσαιων.
- Στη γενική περίπτωση, δεν είναι βέλτιστη (παράδειγμα δυναμικού προγραμματισμού).

Ελάχιστα ζευγνύοντα δένδρα (MST)

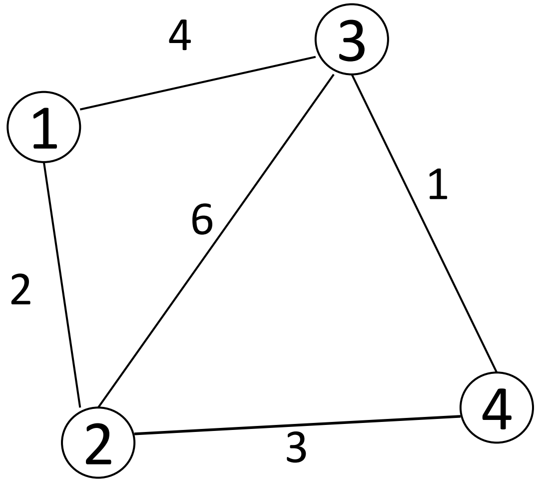
- Ζευγνύον δένδρο ενός συνεκτικού γράφου *G* είναι ένας συνεκτικός άκυκλος υπογράφος του *G* που αποτελείται από όλες τις κορυφές του *G*
- Ελάχιστο ζευγνύον δένδρο ενός ζυγισμένου συνεκτικού γράφου *G* είναι το ζευγνύον δένδρο του *G* με το ελάχιστο συνολικό βάρος
- Παράδειγμα:

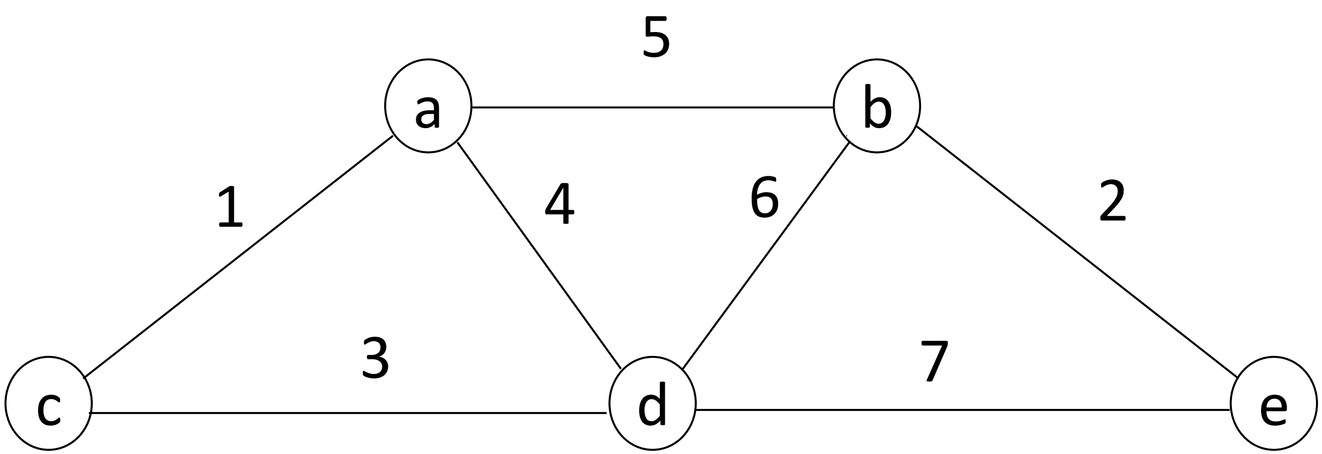


Αλγόριθμος του Prim

- Αρχικά το δένδρο αποτελείται από ένα κόμβο
- Το δένδρο μεγαλώνει κατά μια κορυφή/ακμή κάθε φορά για να προκύψει το MST
  - Δημιουργείται μια σειρά μεγαλύτερων δένδρων T_1, T_2, …
- Σε κάθε στάδιο το δένδρο T_{i+1} κατασκευάζεται από το T_i: προστίθεται η ακμή με το ελάχιστο βάρος, η οποία συνδέει μια κορυφή του δένδρου T_i με κάποια που δεν ανήκει στο T_i
  - Επιλέγουμε από τις ακμές του κρασπέδου (fringe) – αυτό είναι το άπληστο βήμα
- Ο αλγόριθμος σταματά όταν έχουν επιλεγεί όλες οι κορυφές

Παραδείγματα σε Prim:





Κώδικας αλγορίθμου Prim

Algorithm Prim(G)

$V_T \leftarrow \{v_0\}$

$E_T \leftarrow \{\}$

for $i \leftarrow 1$ **to** $|V|-1$ **do**

  find a minimum weight edge $e^*=(v^*,u^*)$ **among the edges** (v,u) **such that** $v \in V_T$ **and** $u \in (V-V_T)$

  $V_T \leftarrow V_T \cup \{u^*\}$

  $E_T \leftarrow E_T \cup \{e^*\}$

return E_T

Σχετικά με τον αλγόριθμο του Prim

- Πρέπει να αποδειχθεί ότι πράγματι ο αλγόριθμος δίνει το MST
- Χρειάζεται μια ουρά προτεραιότητας για να βρούμε την ακμή με το ελάχιστο βάρος: χρησιμοποιούμε ένα min-heap
- Αποτελεσματικότητα: για γράφο με *n* κορυφές και *m* ακμές:
- $(\underbrace{n-1}+m)\ \log n$

insertion/deletion from min-heap

number of stages
(min-heap deletions)

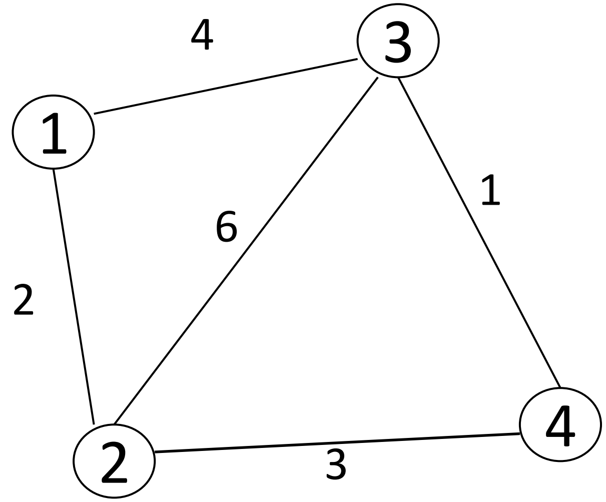
number of edges considered
(min-heap insertions)

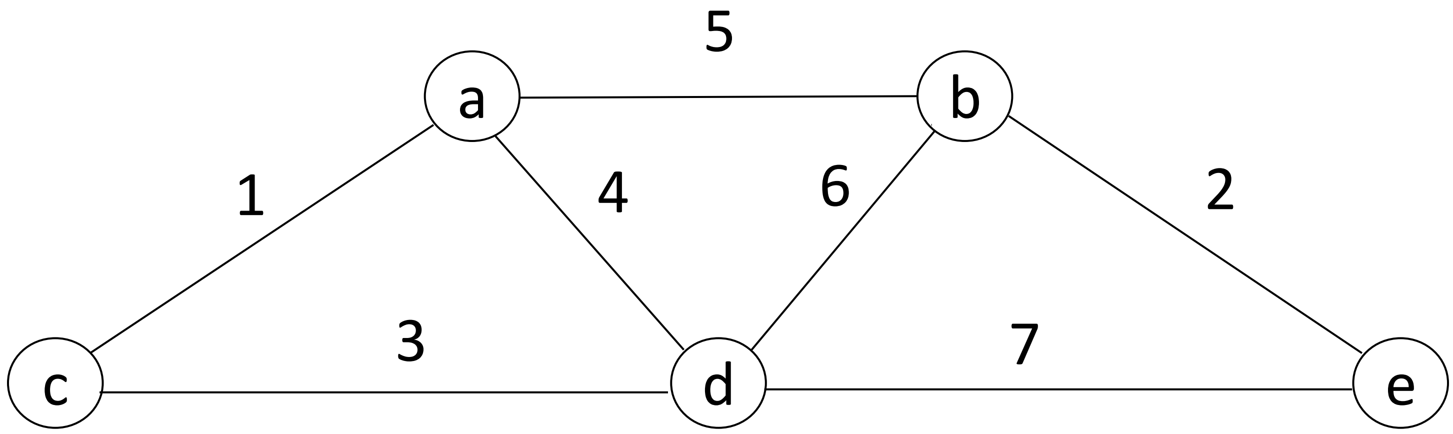
- Συνεπώς: $\Theta(m \log n)$
  - Υποθέτουμε αναπαράσταση με λίστες γειτνίασης και χρήση σωρού.
- Αν δεν χρησιμοποιηθεί λίστα γειτνίασης και σωρός, αλλά απλός αδόμητος πίνακας: $\Theta(n^2)$

Αλγόριθμος του Kruskal

- Αρχίζουμε με ένα κενό δάσος από δένδρα
- Μεγαλώνουμε το MST με μια ακμή κάθε φορά
  - στα ενδιάμεσα στάδια συνήθως έχουμε ένα δάσος από δένδρα (μη συνδεδεμένα)
- Σε κάθε στάδιο προσθέτουμε μια ακμή με ελάχιστο βάρος από αυτές που δεν έχουν χρησιμοποιηθεί και δεν δημιουργούν κύκλο
  - Αρχικά οι κορυφές ταξινομούνται κατά αύξον βάρος
  - Σε κάθε στάδιο η ακμή μπορεί να:
    - Να επεκτείνει ένα υπάρχον δένδρο
    - Να συνδυάσει δυο δένδρα σε ένα
    - Να δημιουργήσει ένα νέο δένδρο
  - Χρειάζεται ένας αποτελεσματικός τρόπος διαπίστωσης κύκλων
- Ο αλγόριθμος τερματίζει όταν επιλεγούν όλες οι κορυφές

Παραδείγματα σε Kruskal:





Κώδικας αλγορίθμου του Kruskal

Algorithm Kruskal(G)

```
Sort E in nondecreasing order of the
  weights
E_T←{ }; k←0; ecounter←0
while ecounter < |V|-1
     k ← k+1
     if E_T U {e_ik} is acyclic
          E_T ← E_T U {e_ik}
          ecounter ← ecounter+1
return E_T
```

Σχετικά με τον αλγόριθμο του Kruskal

- Ο αλγόριθμος φαίνεται ευκολότερος από του Prim αλλά είναι:
  - Δυσκολότερος στην υλοποίηση (έλεγχος κύκλων)
  - Λιγότερο αποτελεσματικός Θ(m log m)
- Έλεγχος κύκλων: ένας κύκλος υφίσταται αν και μόνο αν μια ακμή συνδέει κορυφές της ίδιας συνιστώσας
- Αλγόριθμοι ένωσης-εύρεσης (union-find)

Συντομότερα μονοπάτια - Dijkstra

- Συντομότερα μονοπάτια (one-to-all) : Δεδομένου ενός ζυγισμένου γράφου *G*, να βρεθούν τα συντομότερα μονοπάτια από μια κορυφή προς όλες τις άλλες
- Δεν δουλεύει με αρνητικά βάρη
- Εφαρμόζεται σε κατευθυνόμενους και μη γράφους

Αλγόριθμος του Dijkstra

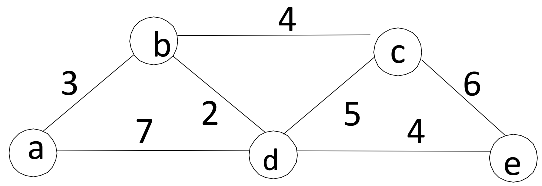
Ο αλγόριθμος του Dijkstra είναι παρόμοιος με του Prim, με την εξής διαφορά:

- Αρχικά το δένδρο έχει μια κορυφή
- Το δένδρο μεγαλώνει με μια κορυφή/ακμή κάθε φορά
- Κρατούμε το συντομότερο μονοπάτι από την πηγή προς τις κορυφές του δένδρου T_i
- Σε κάθε στάδιο κατασκευάζουμε το T_{i+1} από το T_i:

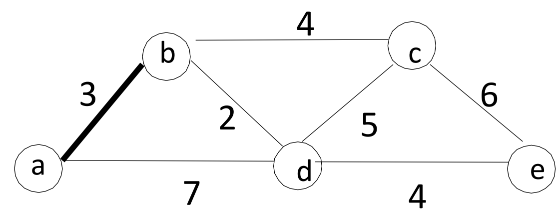
  προσθέτουμε την ακμή (v,w) με το ελάχιστο $d(s,v)+d(v,w)$, όπου $v \in T_i$, αλλά δεν ισχύει $w \in V_i$
- Επιλέγουμε από τις ακμές του κρασπέδου (fringe) – αυτό είναι το άπληστο βήμα
- Ο αλγόριθμος τερματίζει όταν επιλεγούν όλες οι κορυφές

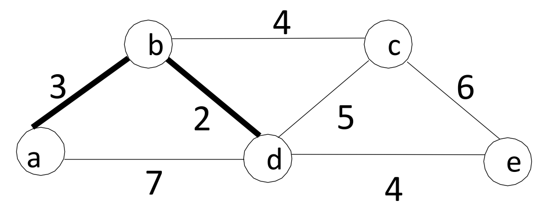
Παράδειγμα

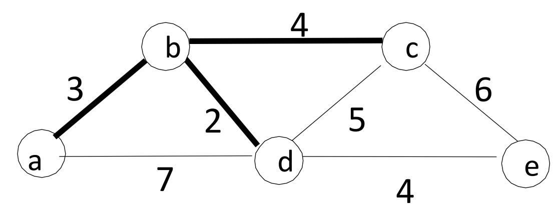
Αρχικό Γράφημα:

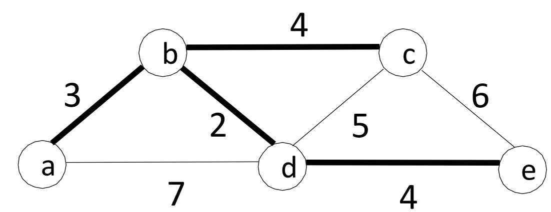


Κορυφές δέντρου	Εναπομείνασες κορυφές
a(-,0)	b(a,3) c(-,∞) d(a,7) e(-,∞)
b(a,3)	c(b,3+4) d(b,3+2) e(-,∞)
d(b,5)	c(b,7) e(d,5+4)
c(b,7)	e(d,9)
e(d,9)	

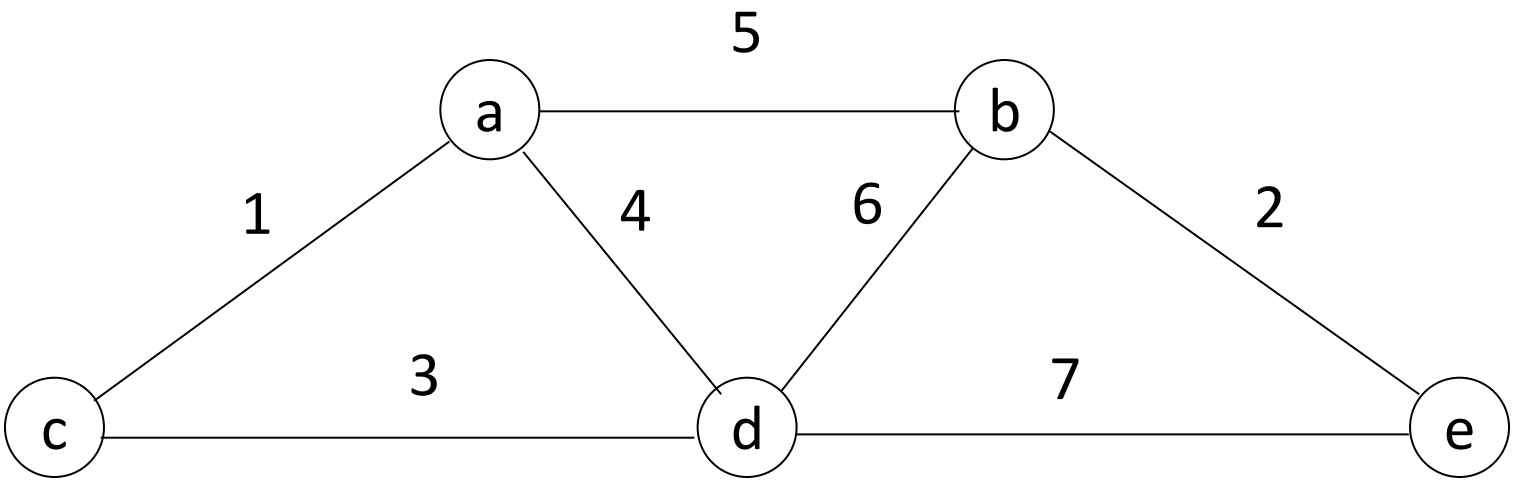








Παράδειγμα σε Dijkstra:



Κώδικας αλγορίθμου του Dijkstra

Algorithm Dijkstra(G,s)

```
Initialize(Q)                    //Q: priority queue
for every vertex v ∈ V do
  dv←∞; pv←null; //p: προτελευταία κορυφή στο
                    μονοπάτι προς v
  Insert(Q,v, dv)
ds←0; Decrease(Q,s,ds); VT←0 //ενημέρωση αφετηρίας
for i←0 to |V|-1 do
  u*←DeleteMin(Q); VT ← VTU{u*}
  for every vertex u in V-VT that is    adjacent
  to u* do
  if du*+w(u*,u)<du
      du←du+w(u*,u); pu←u*; Decrease(Q,u,du)
```

- Αποτελεσματικότητα;
  - Παρόμοια με αυτή του Prim

Πρόβλημα Κωδικοποίησης

Κωδικές Λέξεις: ακολουθία ορισμένων bits για κάθε χαρακτήρα του αλφαβήτου

Δύο τύποι κωδικοποίησης:

- *Σταθερού μήκους (fixed-length encoding)* (π.χ., ASCII)
- *Μεταβλητού μήκους (variable-length encoding)* (π.χ., κώδικας Morse)
- *Prefix-free codes* -Κώδικες προθέματος (κανονικά πρέπει να λέγονται κώδικες άνευ προθέματος):

  καμία κωδική λέξη δεν είναι πρόθεμα μιας άλλης
- Πρόβλημα: Αν είναι γνωστές οι συχνότητες της εμφάνισης των χαρακτήρων, ποιος είναι ο βέλτιστος δυαδικός κώδικας προθέματος;

Κώδικας Huffman

- Κάθε δυαδικό δένδρο με ετικέτες στις ακμές με) και 1 δίνει ένα κώδικα προθέματος για τους χαρακτήρες στα φύλλα.
- Ένα βέλτιστο δυαδικό δένδρο που ελαχιστοποιεί το σταθμισμένο μήκος μίας κωδικής λέξης παράγεται από τον αλγόριθμο Huffman.

Αλγόριθμος Huffman

- Αρχικοποίησε κάθε χαρακτήρα ως δένδρα μίας κορυφής
  - Η ρίζα κάθε δένδρου έχει αποθηκευμένη τη συχνότητα του χαρακτήρα (βάρος δένδρου)
- Επανέλαβε *n*-1 φορές:
  - Σύνδεσε τα δύο δένδρα με το μικρότερο βάρος σε ένα δένδρο (ως αριστερό και δεξί υποδένδρο αντίστοιχα). Το νέο βάρος είναι το άθροισμα των βαρών των δύο αρχικών δένδρων.
- Βάλε ετικέτες στις ακμές για τα αριστερά και τα δεφιά υποδένδρα 0 και 1 αντίστοιχα.

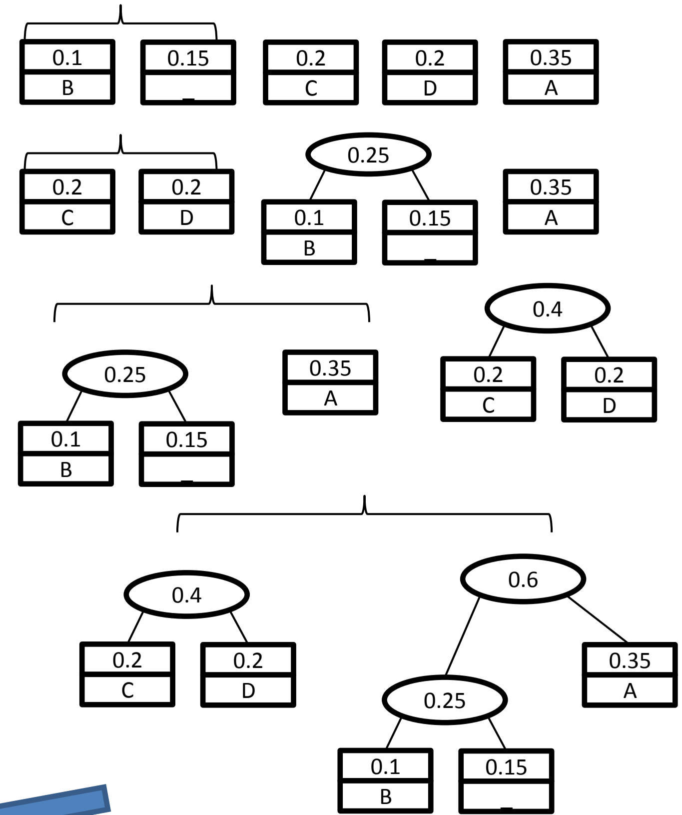
Παράδειγμα

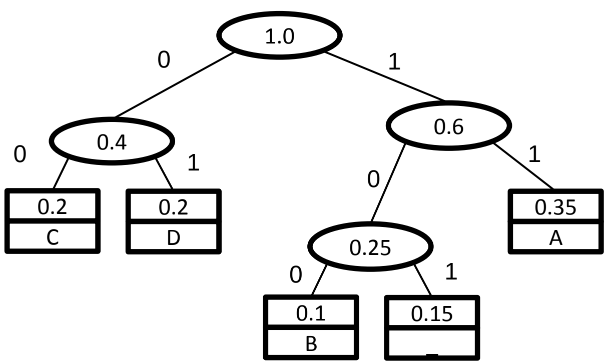
Χαρακτήρας	A	B	C	D	_
Συχνότητα	0.35	0.1	0.2	0.2	0.15
Κωδ. λέξεις	11	100	00	01	101

Μέσο πλήθος bits/χαρακτήρα: 2.25

Κωδ/ση σταθερού μήκους: 3

Λόγος συμπίεσης: (3-2.25)/3*100% = 25%





Σημείωμα Αναφοράς

Copyright Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, Ιωάννης Μανωλόπουλος, Αναστάσιος Γούναρης. «Αλγόριθμοι. ». Έκδοση: 1.0. Θεσσαλονίκη 2015. Διαθέσιμο από τη δικτυακή διεύθυνση: http://eclass.auth.gr/courses/OCRS417/

Σημείωμα Αδειοδότησης

Τέλος ενότητας

Επεξεργασία: Ανδρέας Κοσματόπουλος
Θεσσαλονίκη, Αύγουστος 2015

Σημειώματα

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.00.

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.