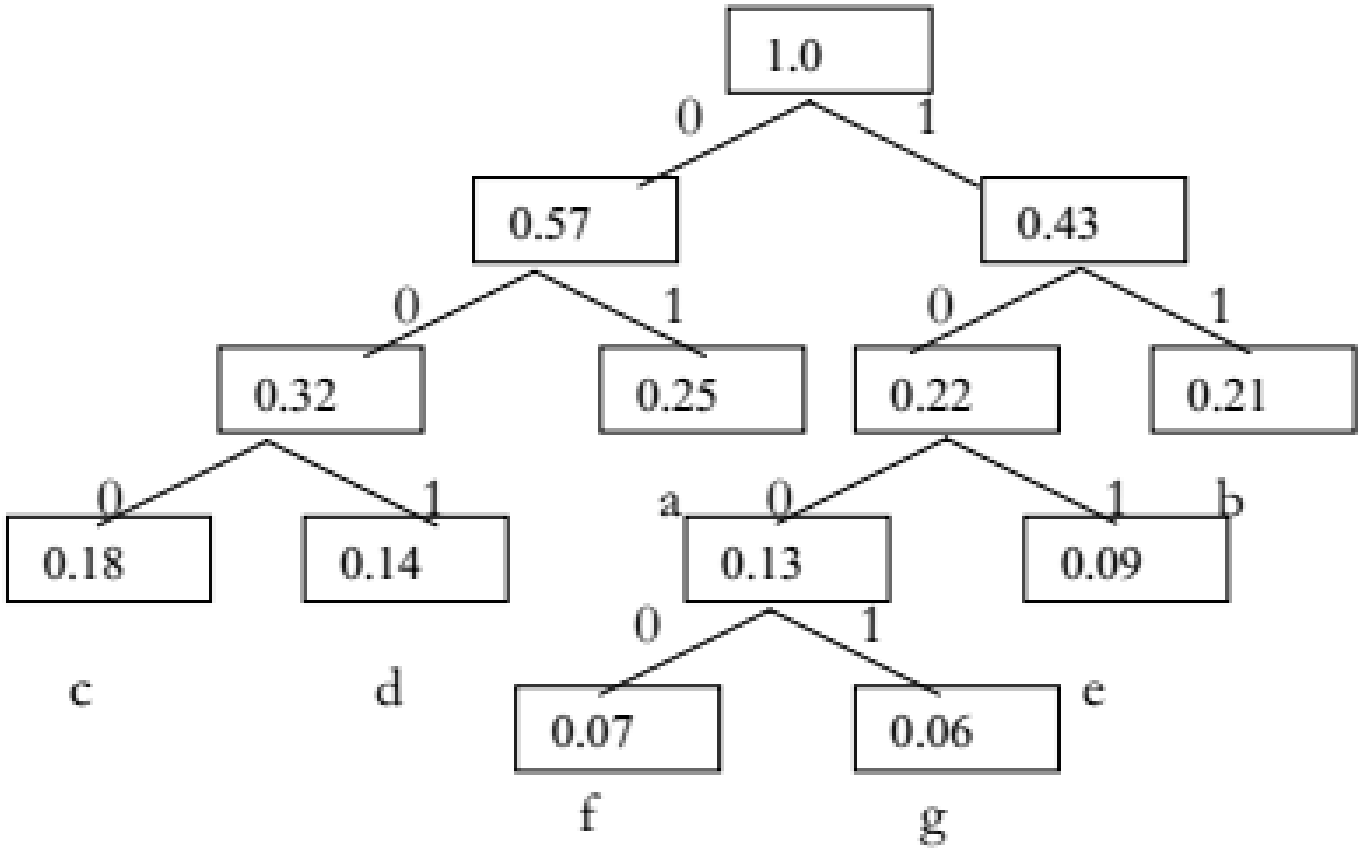
Huffman codes

Περιεχόμενα

Κώδικες

Μια από τις εφαρμογές της θεωρίας κωδίκων είναι και η συμπύκνωση κειμένου για οικονομία χώρου κατά την αποθήκευση. Αντί να χρησιμοποιούνται δυαδικές συμβολοσειρές ίδιου μεγέθους για κάθε χαρακτήρα (π.χ. όπως η κωδικοποίηση ASCII), στην κωδικοποίηση Huffman γίνεται χρήση της γνώσης της συχνότητας εμφάνισης κάθε γράμματος (στατιστικά) σε ένα συνηθισμένο κείμενο. Συμφέρει π.χ. ο κωδικός των χαρακτήρων a, e, h, i, n, o, s, t, space να είναι πιο σύντομος από τον κωδικό των j, k, q, v, x, z. Αν για παράδειγμα θέλουμε να κωδικοποιήσουμε κείμενο που αποτελείται μόνο a, b, c, d, e, f, g με γνωστές συχνότητες εμφάνισης δηλαδή: 0.25, 0.21, 0.18, 0.14, 0.09, 0.07, 0.06 αντίστοιχα, τότε η οικονομικότερη κωδικοποίηση είναι: 01, 11, 000, 001, 101, 1000, 1001 αντίστοιχα που μπορεί να αναπαρασταθεί ως εξής:



Αυτό το δέντρο Huffman κατασκευάζεται ως εξής: Αρχίζουμε με n δεντράκια ενός κόμβου ταξινομημένα με συχνότητα εμφάνισης σε φθίνουσα σειρά. Κατόπιν συγχωνεύουμε σταδιακά έτσι ώστε τελικά να κατασκευαστεί το ανωτέρω δέντρο.

Κώδικες

Ορισμοί

- Αλφάβητο: Σ: Πεπερασμένο σύνολο συμβόλων
- Φυσικό Κείμενο: S - Κωδικοποιημένο Κείμενο: O
- Σύμβολο: c - Κωδικός: c', $|c'| =$ μήκος κωδικού c'
- Συχνότητα εμφάνισης συμβόλου c σε δεδομένο κείμενο S: $f_S(c)$

Κώδικες

Κωδικοποίηση $O = C(S)$

Δεδομένου αλφαβήτου Σ και κειμένου S, αντικατάστησε κάθε σύμβολο με τον αντίστοιχο κωδικό.

Σύμβολα: $\Sigma = \{a, b, c, d\}$
Κώδικας: πεπερασμένες δυαδικές ακολουθίες

$$C(a) = 0, C(b) = 110, C(c) = 10, C(d) = 111$$

$$C(bad) = 1100111$$

Κώδικες

Αποκωδικοποίηση $S = D(O)$

Δεδομένου αλφαβήτου Σ και κωδικοποιημένου κειμένου O, αντικατάστησε κάθε κωδικό με το αντίστοιχο σύμβολο.

$$D(00) = a, D(01) = b, D(10) = c, D(11) = d$$

$$D(010011) = bad$$

Κώδικες

Για να είναι ένας κώδικας μονοσήμαντος θα πρέπει η κωδικοποίηση να είναι 1-1, άρα η αποκωδικοποίηση να δίνεται από μια συνάρτηση, δηλαδή να υπάρχει **Μοναδικότητα Αποκωδικοποίησης**.
Διαλέγουμε C, D τέτοιες ώστε $\forall c \in \Sigma : c = D(C(c))$

Κώδικες

Ορισμός: Prefix Code

Ένας κώδικας είναι Prefix Code αν κανένας κωδικός δεν είναι prefix ενός άλλου.

Παράδειγμα: $C(a, b, c, d) = (0, 110, 10, 111)$ είναι Prefix Code

Πρόταση: Μοναδικότητα Αποκωδικοποίησης

Οποιοσδήποτε Prefix Code ικανοποιεί την Μοναδικότητα Αποκωδικοποίησης.

Κώδικες

Δεδομένου ενός κειμένου S σε ένα αλφάβητο Σ θέλω να κωδικοποιήσω το κείμενο έτσι ώστε να χρησιμοποιείται ο ελάχιστος αριθμός ψηφίων (εξοικονόμηση μνήμης). Επομένως, τίθεται το παρακάτω πρόβλημα ελαχιστοποίησης:

Πρόβλημα Κωδικοποίησης

Δεδομένου κειμένου S σε ένα αλφάβητο $\Sigma = \{c_1, \ldots, c_n\}$, να βρεθεί συνάρτηση κωδικοποίησης C που ικανοποιεί την **Μοναδικότητα Αποκωδικοποίησης**, έτσι ώστε να ελαχιστοποιείται η παρακάτω συνάρτηση κόστους

$$B(C) = \sum_{i=1}^{n} f_S(c_i) \cdot |C(c_i)|$$

Κώδικες

Ένα prefix code αναπαριστάται με ένα δυαδικό δέντρο T.

- Σύμβολα $\Longleftrightarrow$ Φύλλα
- Αριστερή Ακμή: 0 - Δεξιά Ακμή: 1
- Κωδικός του c $\Longleftrightarrow$
  Δυαδική επιγραφή μονοπατιού από την ρίζα στο φύλλο c
- $B(C) = \sum_{i=1}^{n} f_S(c_i) \cdot |C(c_i)| = \sum_{i=1}^{n} f_S(c_i) \cdot depth(c_i) = B(T)$

Κώδικας Huffman

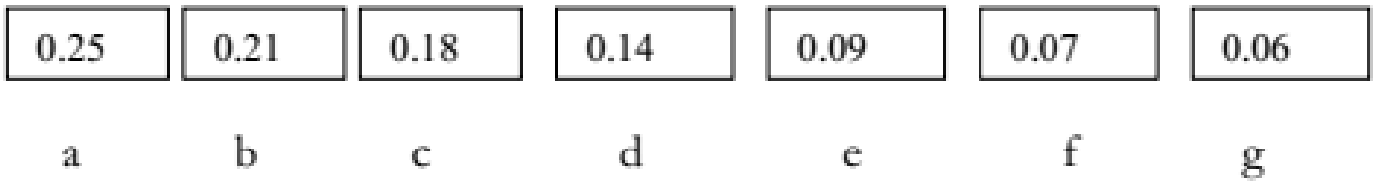
Ο Huffman, στο διδακτορικό του (MIT), ανέπτυξε έναν **Άπληστο** Αλγόριθμο που λύνει το Πρόβλημα Κωδικοποίησης και παράγει έναν Prefix Code. Δημοσιεύτηκε το 1952.

Αλγόριθμος

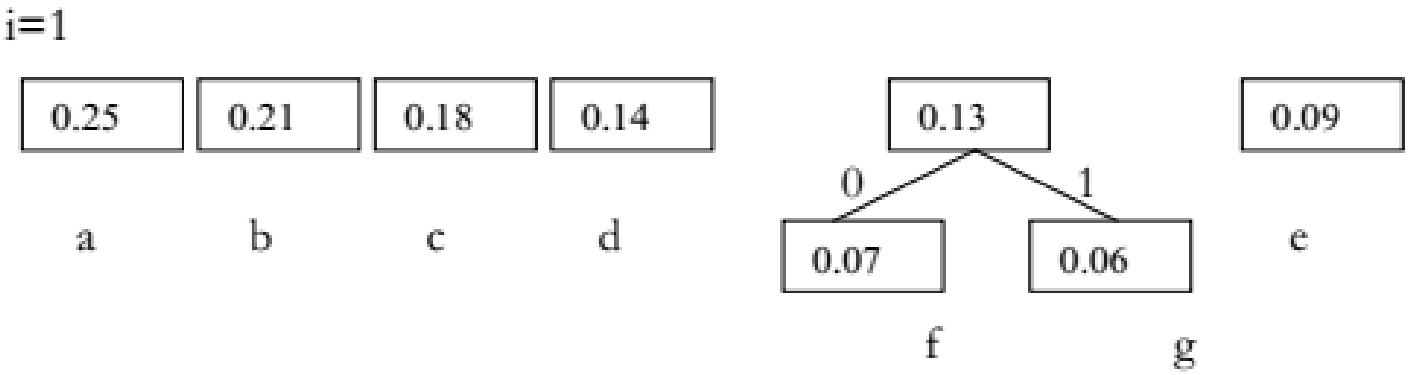
PROC huffman(int n);
{
int i;
FOR (i, 1 TO $n-1$)
{
$\Rightarrow$ Merge last two subtrees;
$\Rightarrow$ Rearrange subtrees in nondecreasing order of root - probability;
}
}

Πολυπλοκότητα: $O(n \log n)$ (αν έχει υλοποιηθεί η λίστα των δέντρων σε σωρό – heap)

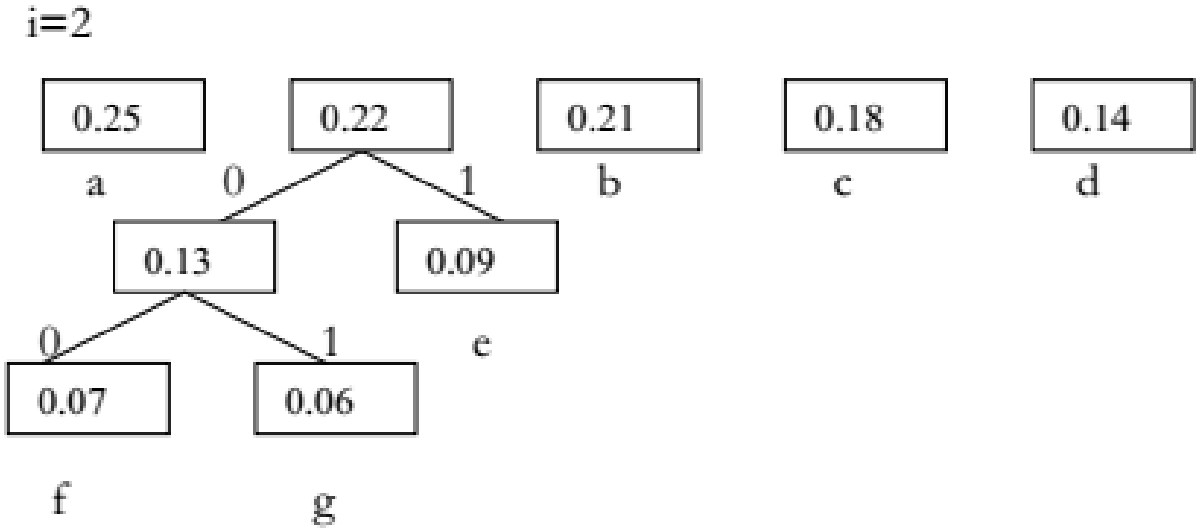
Παράδειγμα αλγορίθμου



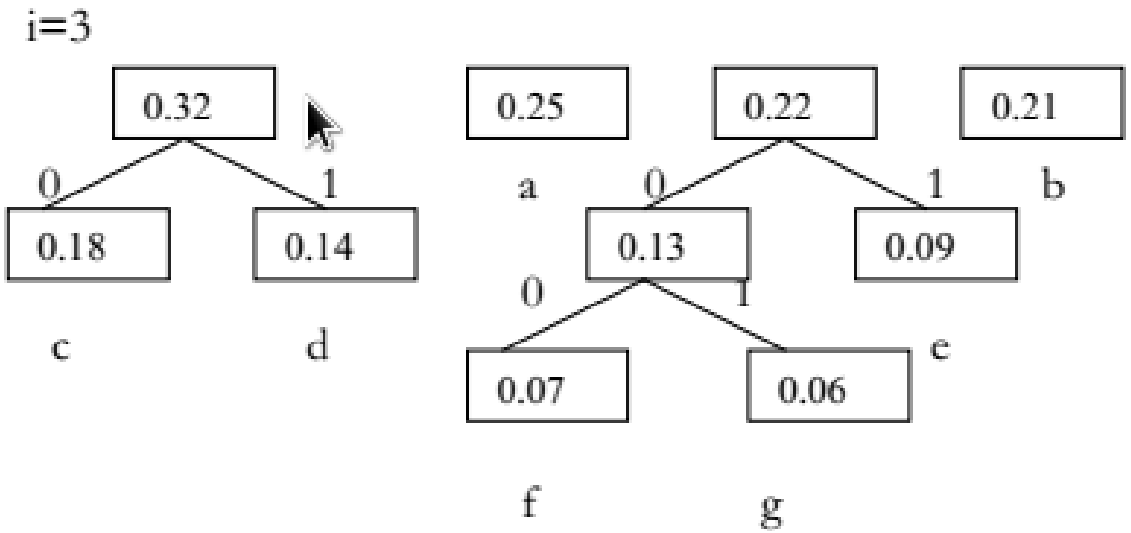
Παράδειγμα αλγορίθμου



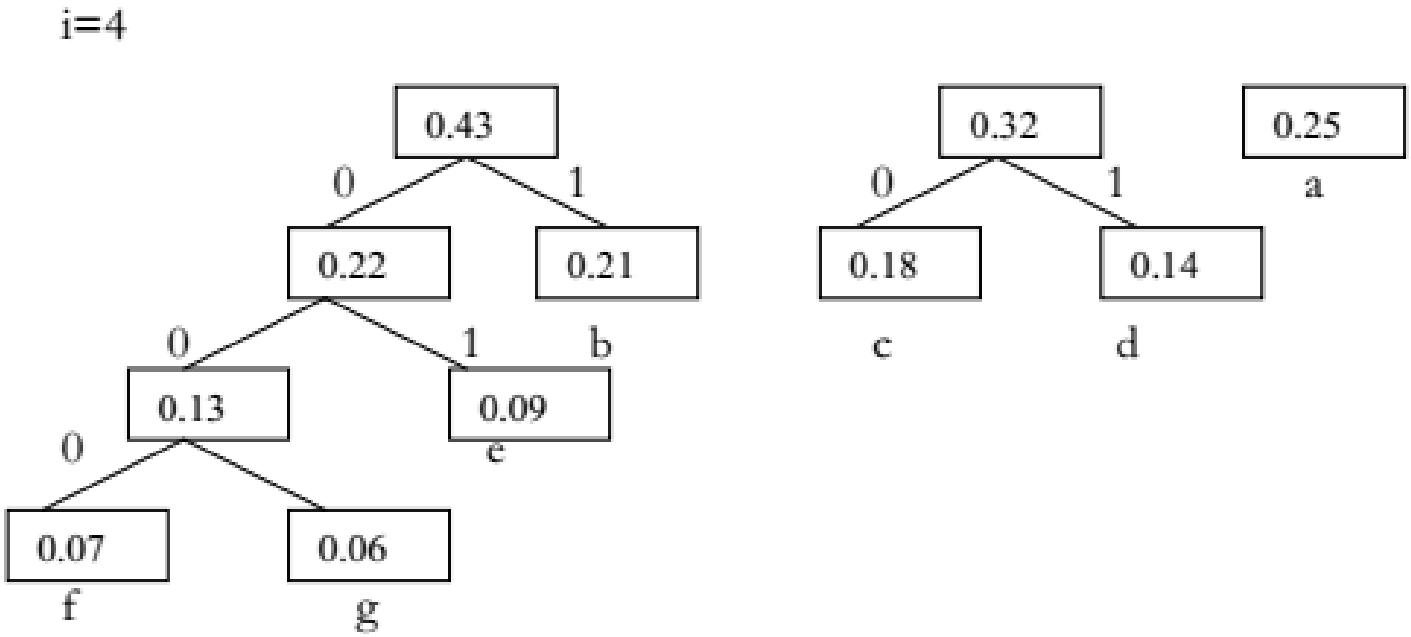
Παράδειγμα αλγορίθμου



Παράδειγμα αλγορίθμου

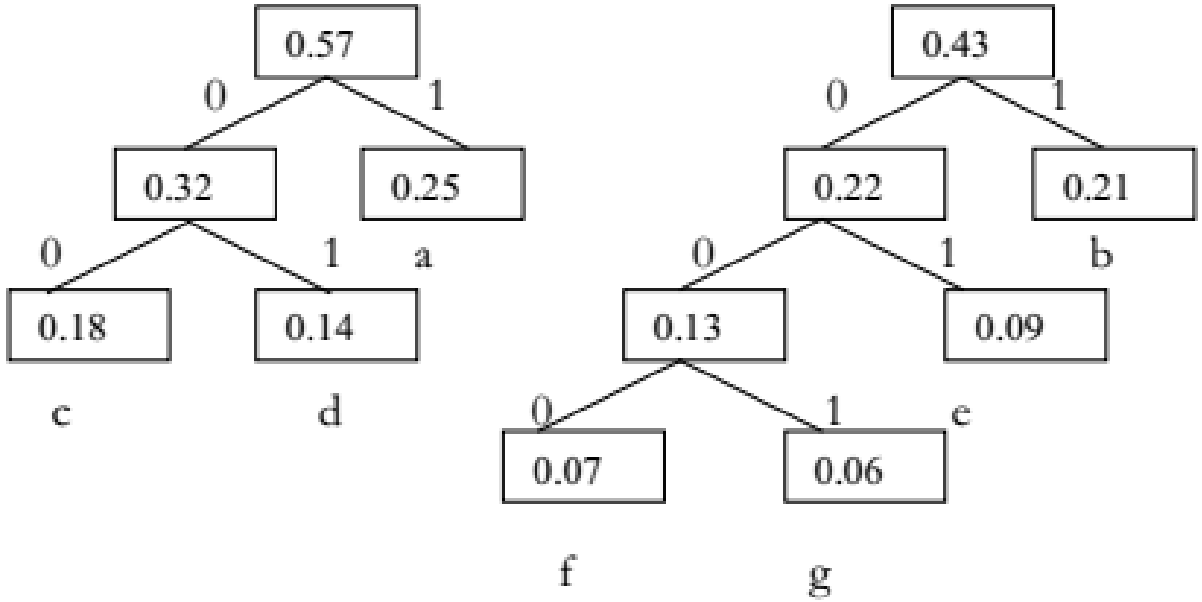


Παράδειγμα αλγορίθμου

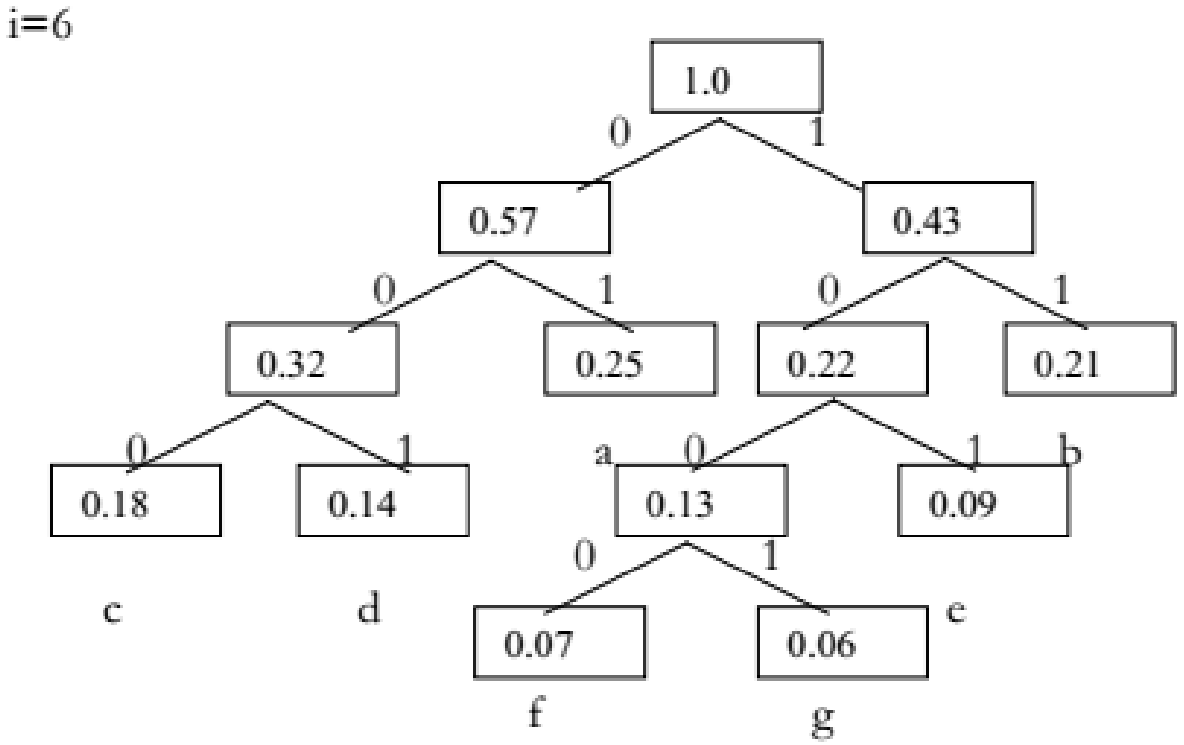


Παράδειγμα αλγορίθμου

i=5



Παράδειγμα αλγορίθμου



Άσκηση

Σχεδιάστε ένα πρόγραμμα Pazcal που υλοποιεί τον παραπάνω αλγόριθμο κατασκευής δέντρου Huffman. Χρησιμοποιείστε για σύμβολα μόνο γράμματα του Λατινικού αλφαβήτου και το κενό, και για συχνότητες αυτές που δίνονται στον παρακάτω πίνακα (μέσος όρος εμφάνισης σε αγγλικό κείμενο):

a	b	c	d	e	f	g	h	i	j	k
.065	.013	.022	.032	.104	.021	.015	.047	.058	.001	.005
l	m	n	o	p	q	r	s	t	u	v
.032	.032	.058	.064	.015	.001	.049	.056	.081	.023	.008
w	x	y	z	<space>						
.018	.001	.017	.001	.172						

Σχεδιάστε διαλογικό πρόγραμμα Pazcal που να έχει την δυνατότητα να κωδικοποιήσει και να αποκωδικοποιήσει αγγλικό κείμενο (αγνοώντας σημεία στίξης κλπ) χρησιμοποιώντας μόνο το προηγούμενο πρόγραμμα.

Βελτιστότητα

Λήμμα 1

Το δέντρο ενός βέλτιστου prefix code πρέπει να είναι πλήρες, δηλαδή κάθε εσωτερικός κόμβος πρέπει να έχει ακριβώς δύο παιδιά.

Βελτιστότητα

Λήμμα 1

Το δέντρο ενός βέλτιστου prefix code πρέπει να είναι πλήρες, δηλαδή κάθε εσωτερικός κόμβος πρέπει να έχει ακριβώς δύο παιδιά.

Απόδειξη.

Έστω ότι κάποιος εσωτερικός κόμβος έχει μόνο ένα παιδί τότε θα μπορούσαμε να διαγράψουμε αυτόν κόμβο και να τον αντικαταστήσουμε με το μοναδικό παιδί του. Αυτό θα μείωνε το συνολικό κόστος της κωδικοποίησης. □

Βελτιστότητα

Λήμμα 2

Έστω δύο σύμβολα x και y με τις μικρότερες συχνότητες, τότε υπάρχει βέλτιστο δέντρο κωδικοποίησης στο οποίο τα δύο σύμβολα είναι αδέρφια στο κατώτατο επίπεδο.

Βελτιστότητα

Λήμμα 2

Έστω δύο σύμβολα x και y με τις μικρότερες συχνότητες, τότε υπάρχει βέλτιστο δέντρο κωδικοποίησης στο οποίο τα δύο σύμβολα είναι αδέρφια στο κατώτατο επίπεδο.

Η απόδειξη του λήμματος χρησιμοποιεί το επιχείρημα της ανταλλαγής που είναι κλασσικό στην απόδειξη ορθότητας των άπληστων αλγορίθμων.

Βελτιστότητα

Απόδειξη.

Έστω T βέλτιστο δέντρο και έστω b και c δύο αδέρφια στο κατώτατο επίπεδο του δέντρου. Τέτοια υπάρχουν αφού το T είναι πλήρες. Θεωρούμε χωρίς βλάβη της γενικότητας ότι $f(b) \leq f(c)$ και $f(x) \leq f(y)$. Αφού x, y οι χαρακτήρες με τις μικρότερες συχνότητες τότε $f(x) \leq f(b)$ και $f(y) \leq f(c)$, αφού b και c βρίσκονται στο κατώτατο επίπεδο τότε $depth(b) \geq depth(x)$ και $depth(c) \geq depth(y)$. Αλλάζοντας x με b φτιάχνουμε ένα καινούργιο δέντρο T'. $B(T) \leq B(T') = B(T) - f(x)depth(x) - f(b)depth(b) + f(x)depth(b) + f(b)depth(x) = B(T) + (f(x) - f(b))(depth(b) - depth(x)) \leq B(T)$

Επομένως $B(T') = B(T)$, δηλαδή T' επίσης βέλτιστο δέντρο. Ομοίως αλλάζουμε y με c και παίρνουμε T'' το οποίο είναι επίσης βέλτιστο και έχει την μορφή που ζητάμε. □

Βελτιστότητα

Λήμμα 3

Έστω δυαδικό δέντρο T ένα βέλτιστο δέντρο στο αλφάβητο Σ. Έστω δύο σύμβολα x και y που είναι αδέρφια στο T, και έστω z ο γονέας τους. Το z δεν είναι σύμβολο του Σ. Αν όμως βγάλουμε το x και το y από το Σ και προσθέσουμε σαν νέο σύμβολο το z με συχνότητα εμφάνισης $f(z) = f(x) + f(y)$, τότε το νέο δέντρο με το z ως φύλλο (χωρίς τα x και y) είναι βέλτιστο.

Βελτιστότητα

Λήμμα 3

Έστω δυαδικό δέντρο T ένα βέλτιστο δέντρο στο αλφάβητο Σ. Έστω δύο σύμβολα x και y που είναι αδέρφια στο T, και έστω z ο γονέας τους. Το z δεν είναι σύμβολο του Σ. Αν όμως βγάλουμε το x και το y από το Σ και προσθέσουμε σαν νέο σύμβολο το z με συχνότητα εμφάνισης $f(z) = f(x) + f(y)$, τότε το νέο δέντρο με το z ως φύλλο (χωρίς τα x και y) είναι βέλτιστο.

κ.ο.κ. Απόδειξη με επαγωγή στο μέγεθος του αλφαβήτου.

Διαφημιστικό Μήνυμα

Για περισσότερους άπληστους αλγορίθμους και πολλά άλλα ...
ΑΛΓΟΡΙΘΜΟΙ ΚΑΙ ΠΟΛΥΠΛΟΚΟΤΗΤΑ (7Ο ΕΞΑΜΗΝΟ)