ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΠΛ 231: Δομές Δεδομένων και Αλγόριθμοι

Εαρινό Εξάμηνο 2013

ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΑΣΚΗΣΗ 2

Διδάσκων Καθηγητής: Παναγιώτης Ανδρέου

Ημερομηνία Υποβολής: 05/03/2013
Ημερομηνία Παράδοσης: 22/03/2013

ΠΕΡΙΓΡΑΦΗ

Σε αυτή την άσκηση καλείστε να υλοποιήσετε το πρόγραμμα HuffmanCompression για την κωδικοποίηση και αποκωδικοποίηση συμβολοσειρών με σκοπό η αποθήκευσή τους στο δίσκο να χρειάζεται μικρότερο χώρο (σε bytes).

ΠΡΟΚΑΤΑΡΚΤΙΚΑ ΓΙΑ ΚΩΔΙΚΟΠΟΙΗΣΗ

Οι περισσότερες γλώσσες προγραμματισμού χρησιμοποιούν την κωδικοποίηση ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange). Στην κωδικοποίηση ASCII, κάθε χαρακτήρας κωδικοποιείται με τον ίδιο αριθμό από bits, συγκεκριμένα 8. Αν θέλετε να δείτε την κωδικοποίηση ASCII τότε μπορείτε να επισκεφθείτε την ιστοσελίδα http://www.asciitable.com/.

Ένας τρόπος να δημιουργήσετε το δικό σας ASCII table είναι να καταλάβετε πως αντιπροσωπεύονται οι χαρακτήρες στην γλώσσα Java. Σκεφθείτε το επόμενο παράδειγμα:

```
char c='a';
System.out.println("Char: "+c +"\tASCII:"+((int) c));
```

Το αποτέλεσμα του πιο πάνω κώδικα είναι το εξής:

```
Char: a         ASCII:97
```

Πως αποθηκεύονται όμως οι πληροφορίες σε ένα αρχείο;

Αφού κάθε χαρακτήρας μπορεί να αναπαρασταθεί από 8 bits τότε σε κάθε αρχείο με N χαρακτήρες χρειαζόμαστε Nx8bits για να το αποθηκεύσουμε.

Παράδειγμα

Σκεφτείτε το αρχείο input.txt το οποίο έχει την εξής συμβολοσειρά μόνο αποθηκευμένη.

abaacbdace

ASCII coding		
Char	ASCII	Binary
a	97	11100001
b	98	11100010
c	99	11100011
d	100	11100100
e	101	11100101

Για να αποθηκεύσουμε αυτή την συμβολοσειρά binary, θα χρειαστούμε να αποθηκεύσουμε το εξής:

abaacbdace

11100001 11100010 11100001 11100001 11100011 11100010 11100100
11100001 11100011 11100101

Συνολικά θα χρειαστούμε 10x8bits = 80 bits.

Υπάρχει καλύτερος τρόπος για να αποθηκεύσουμε αυτή την συμβολοσειρά;

Η απάντηση είναι φυσικά ΝΑΙ και είναι η κωδικοποίηση. Αφού η συμβολοσειρά μας αποτελείται μόνο από 5 χαρακτήρες μπορούμε να δημιουργήσουμε μία δική μας κωδικοποίηση η οποία να περιέχει κώδικες μόνο για τους 5 χαρακτήρες. Για να κωδικοποιήσουμε σε binary τους αριθμούς 0-4 χρειαζόμαστε 3 bits. Εφαρμόζοντας αυτό στο πιο πάνω παράδειγμα έχουμε το εξής αποτέλεσμα:

abaacbdace

ASCII coding		
Char	My Coding	Binary
a	0	000
b	1	001
c	2	010

d	3	011
e	4	100

Σε αυτή την περίπτωση για να αποθηκεύσουμε αυτή την συμβολοσειρά binary, θα χρειαστούμε να αποθηκεύσουμε το εξής:

abaacbdace – 10chars

000 001 000 000 010 001 011 000 010 100

Συνολικά θα χρειαστούμε 10x3bits = 30 bits. Η διαφορά είναι σημαντική, ειδικά για πιο μεγάλα αρχεία!

Υπάρχει όμως ακόμη καλύτερος τρόπος για να αποθηκεύσουμε αυτή την συμβολοσειρά; Προσέξτε ότι χρησιμοποιήσαμε 3 bits για κάθε χαρακτήρα → σταθερό μέγεθος. Αν το μέγεθος είναι μεταβλητό;

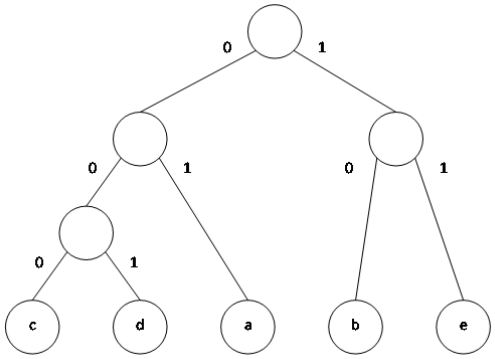
Η απάντηση είναι φυσικά ΝΑΙ, και είναι η κωδικοποίηση Huffman!

<u>Κωδικοποίηση Huffman</u>

Η κωδικοποίηση Huffman βασίζεται σε ένα δέντρο κωδικοποίησης. Το δέντρο κωδικοποίησης είναι ένα δυαδικό δέντρο αναζήτησης όπου αποθηκεύει binary αναπαραστήσεις χαρακτήρων. Χρησιμοποιώντας αυτό το δέντρο μία ακολουθία από χαρακτήρες μπορεί να κωδικοποιηθεί σε bits και αντίστροφα μία ακολουθία από bits μπορεί να αποκωδικοποιηθεί σε χαρακτήρες.

Το πλεονέκτημα της κωδικοποίησης με δέντρο κωδικοποίησης βρίσκεται στο γεγονός ότι δεν χρειάζεται να κωδικοποιούνται οι χαρακτήρες σε ένα σταθερό μέγεθος από bits αλλά σε μεταβλητό.

Σκεφτείτε το εξής τυχαίο δέντρο κωδικοποίησης.



Με αυτό το δέντρο παράγεται η εξής κωδικοποίηση:

Char	Code
c	000
d	001
a	01
b	10
e	11

Σε αυτή την περίπτωση για να αποθηκεύσουμε αυτή την συμβολοσειρά binary, θα χρειαστούμε να αποθηκεύσουμε το εξής:

abaacbdace - 10chars

01 10 01 01 000 10 001 01 000 11

Συνολικά θα χρειαστούμε 7x2bits και 3x3 bits = 23 bits. (<30 bits)

<u>Πως παράγεται όμως ένα δέντρο κωδικοποίησης Huffman;</u>

Η βασική λογική είναι ότι οι χαρακτήρες με μεγάλη συχνότητα θα πρέπει να κωδικοποιούνται με όσο το δυνατό λιγότερα bits ώστε να μειώνεται όσο είναι δυνατό περισσότερο το τελικό αρχείο. Για να το καταφέρουμε αυτό θα πρέπει:

1. να διατρέξουμε το αρχείο μας μία φορά για να βρούμε τις συχνότητες.
2. Ακολούθως να δημιουργήσουμε το Huffman tree. Σε κάθε βήμα επιλέγουμε δύο κόμβους, αυτούς με τις δύο ελάχιστες συνολικές συχνότητες και δημιουργούμε ένα καινούριο κόμβο με το άθροισμα των συχνοτήτων.
3. Να κωδικοποιήσουμε το αρχείο σύμφωνα με την κωδικοποίηση που παράξαμε

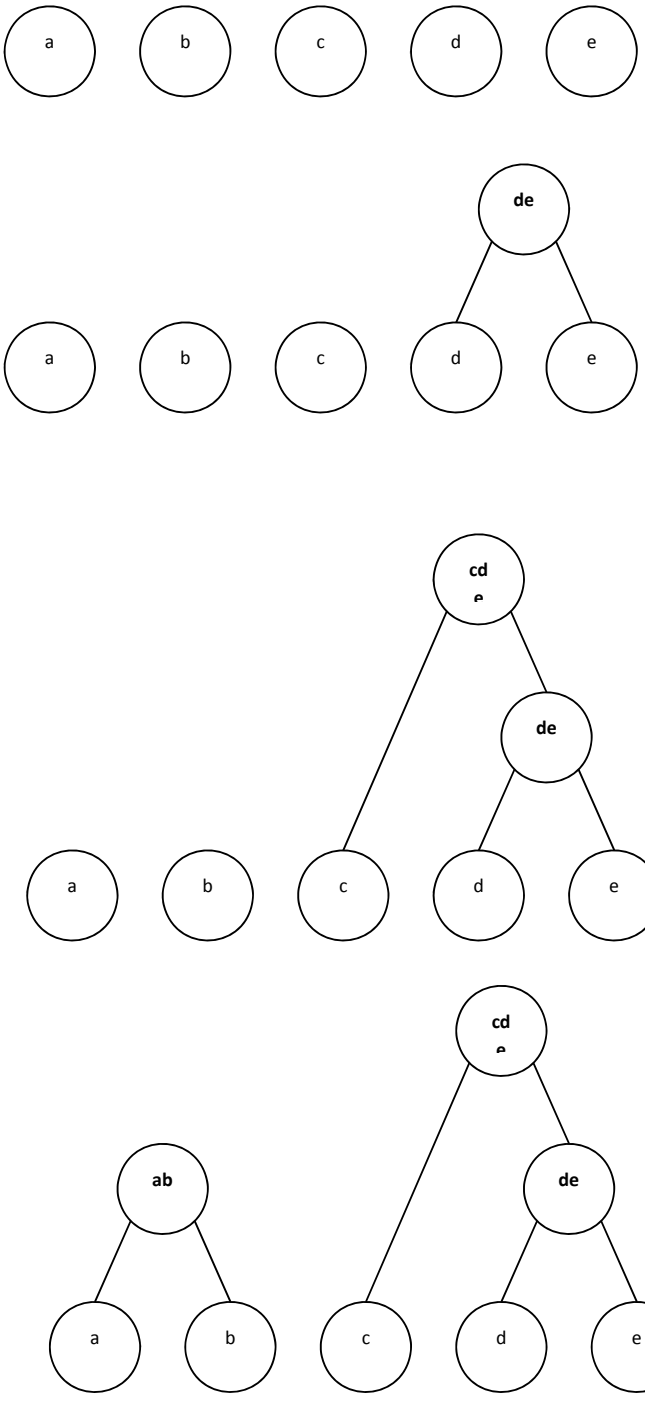
Για παράδειγμα:

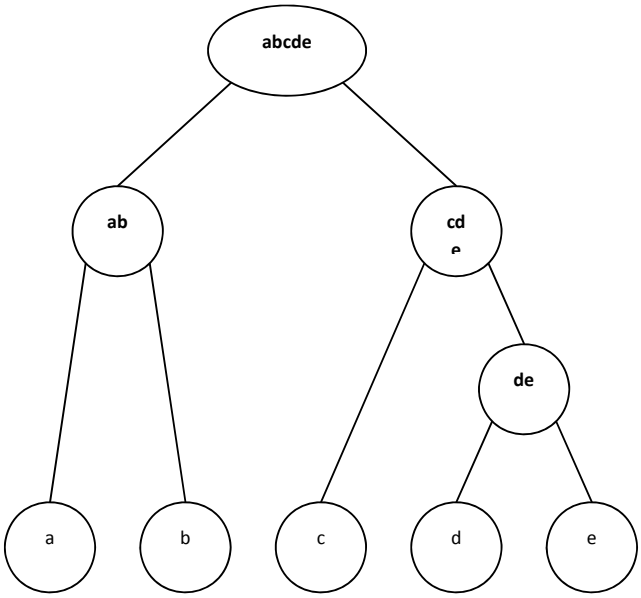
abaacbdacebd – 12 chars

1.

Char	**Συχνότητα**
a	4/12
b	3/12
c	2/12
d	2/12
e	1/12

2.





Από το πιο πάνω δέντρο βγαίνει η πιο κάτω κωδικοποίηση:

Char	Code
a	00
b	01
c	10
d	110
e	111

Για να αποθηκεύσουμε αυτή την συμβολοσειρά binary, θα χρειαστούμε να αποθηκεύσουμε το εξής:

abaacbdacebd – 12 chars

00 01 00 00 10 01 110 00 10 111 01 110

Συνολικά θα χρειαστούμε (9x2bits + 3x3bits)=27bits.

ΖΗΤΟΥΜΕΝΟ

Το πρόγραμμά σας με την εκκίνηση θα παρουσιάζει το εξής menu στον χρήστη.

1. Διάβασμα αρχείου και δημιουργία δέντρου Huffman Tree
    a. Είσοδος: όνομα αρχείου που θα διαβαστεί
    b. Έξοδος: μήνυμα ότι το αρχείο έχει διαβαστεί επιτυχώς, πίνακας με συχνότητες χαρακτήρων, πίνακας με κωδικοποιήσεις χαρακτήρων

2. Αποθήκευση αρχείου και Huffman Tree σε binary αρχείο
    a. Είσοδος: όνομα αρχείου που θα αποθηκευτεί
    b. Έξοδος: μήνυμα ότι το αρχείο έχει αποθηκευτεί

3. Διάβασμα και αποκωδικοποίηση δυαδικού αρχείου
    a. Είσοδος: όνομα αρχείου που θα διαβαστεί
    b. Έξοδος: αποκωδικοποίηση αρχείου – τύπωμα στην οθόνη

4. Έξοδος

ΠΕΡΙΟΡΙΣΜΟΙ

- Δεν μπορείτε να υποθέσετε ότι τα αρχεία για αποκωδικοποίηση είναι σωστά.
- Δεν μπορείτε να υποθέσετε ότι οι χαρακτήρες των αρχείων για αποκωδικοποίηση θα βρίσκονται στο δέντρο Huffman.

ΑΞΙΟΛΟΓΗΣΗ

Για την αξιολόγηση του προγράμματος σας, θα ληφθούν οι πιο κάτω παράμετροι:

- **70% Ορθότητα**: ο πρόγραμμα σας πρέπει τρέχει ορθά για οποιαδήποτε είσοδο.
- **5% Κατανοητά σχόλια**: Γράφετε κατανοητά σχόλια που να εξηγούν την λειτουργία της κάθε κλάσης/πεδίου/μεθόδου.
- **5% Αναφορά Προγράμματος**: Θα πρέπει να παραδώσετε μία αναφορά η οποία να περιγράφει συνοπτικά τα διάφορα συστατικά (κλάσεις, μεθόδους, διαπροσωπείες, κτλ.) του προγράμματος σας. Επίσης, η αναφορά θα πρέπει να καταγράφει τυχόν υποθέσεων τις οποίες έχετε κάνει.
- **20% Αποδοτικότητα**: το πρόγραμμα σας θα πρέπει να χρησιμοποιήσει αποδοτικές δομές δεδομένων και αλγόριθμους όπου αρμόζει.
- **10% Επιπρόσθετη Βαθμολογία:** η κατάλληλη μετατροπή του προγράμματος ώστε να διαχειρίζεται χαρακτήρες από διάφορες γλώσσες (δηλ., Unicode).

ΟΔΗΓΙΕΣ

- Στο πάνω μέρος της κάθε κλάσης να γράψετε με σχόλια το ονοματεπώνυμο και τον αριθμό ταυτότητας σας.
- Βεβαιωθείτε ότι τα προγράμματα σας είναι ορθά και τρέχουν.
- Η άσκηση σας θα πρέπει να παραδοθεί στον υπεύθυνο βοηθό μέσω moodle σαν ένα zip αρχείο (που να περιλαμβάνει όλες τις κλάσεις που θα υλοποιήσετε και το έγγραφο τεκμηρίωσης) σε ηλεκτρονική μορφή και σε έντυπη μορφή την ημερομηνία παράδοσης.
- Το zip αρχείο θα πρέπει να περιλαμβάνει τον κώδικα σας που θα γίνει μέσω export από το eclipse σαν archive file και την αναφορά σας.
- Το zip αρχείο θα ονομάζεται με τον εξής τρόπο:
  epl231.pex2.<ταυτότητα>.zip
  Η αναφορά θα πρέπει να ονομάζεται με τον εξής τρόπο:
  epl231.pex2.<ταυτότητα>.[doc|pdf]
- Μη τήρηση των ημερομηνιών παράδοσης των εργασιών συνεπάγεται τις ανάλογες βαθμολογικές επιπτώσεις (μέχρι τον μηδενισμό της εργασίας).
- Οι προγραμματιστικές ασκήσεις θα ελέγχονται από ειδικό πρόγραμμα για την ανίχνευση των αντιγραφών. Οι αντιγραμμένες εργασίες θα μηδενίζονται και για τους αντιγραφείς θα εφαρμόζονται οι κανόνες τού Πανεπιστημίου. Αποφύγετε λοιπόν την αντιγραφή προγραμμάτων από άλλους συναδέλφους σας, διότι έτσι εκτίθετε και αυτούς και τον εαυτό σας στον κίνδυνο μηδενισμού και πειθαρχικής δίωξης.

ΚΑΛΗ ΕΠΙΤΥΧΙΑ!!!