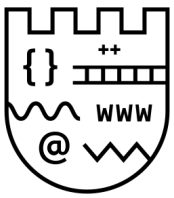


Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Τμήμα Πληροφορικής

Έβδομη εργαστηριακή αναφορά για NCO-04-05

Έλεγχος ισοτιμίας

Αλέξανδρος Κόρκος
alexkork@csd.auth.gr
3870

Θεσσαλονίκη, December 28, 2022

Άσκηση 1

Εκφώνηση: Περιγράψτε την έννοια της ισοτιμίας και δημιουργείστε κώδικα, που να υπολογίζει άρτια ισοτιμία για ένα μήνυμα των 14 bits και να προσαρτά σε αυτό ένα bit ισοτιμίας. Τρέξτε τον κώδικά σας για το μήνυμα 10001101011000.
Απάντηση: Η ισοτιμία είναι μια πολύ απλή μέθοδος, εύρεσης σφαλμάτων σε (δυαδικά μηνύματα) όπου εξετάζεται το πλήθος των bit με τιμή "1" σε ένα μήνυμα. Η διαδικασία έχει ως εξής:

- Ο πομπός και ο δέκτης, θα πρέπει να έχουν συμφωνήσει εάν θα χρησιμοποιηθεί περιττή ή άρτια ισοτιμία.
- Ο πομπός, σε ένα μήνυμα μεγέθους **n** bit, προσθέτει στο τέλος του μηνύματος ένα ακόμα bit όπου η τιμή του εξαρτάται ανάλογα με τι είδους ισοτιμίας έχει συμφωνηθεί (δηλ. εάν έχουμε άρτια ισοτιμία, και το μήνυμα είναι 1010110 θα προστεθεί 0 ενώ για 1010100 θα προστεθεί 1).
- Εάν έχει συμφωνηθεί άρτια ισοτιμία, ο δέκτης εξετάζει εάν το πλήθος των "1" του μηνύματος είναι άρτιο ενώ για περιττή εξετάζει για περιττό πλήθος "1".
- Εάν το πλήθος είναι το συμφωνημένο, τότε το μήνυμα γίνεται αποδεκτό διαφορετικά απορρίπτεται.

Για τον υπολογισμό της άρτιας ισοτιμίας, υλοποιήθηκε η παρακάτω συνάρτηση (`computeParity`):

```
function msg = computeParity(x)
  N = length(x);
  countOnes = 0;
  for i = 1:N
    if (x(i))
      countOnes = countOnes + 1;
    endif
  endfor
  if (mod(countOnes, 2) ~= 0)
    msg = [x 1];
  else
    msg = [x 0];
  endif
endfunction
```

Κώδικας 1: Υλοποίηση αλγορίθμου άρτιας ισοτιμίας (στο αρχείο `computeParity.m`)

Παρακάτω ακολουθεί η υλοποίηση της πρώτης άσκησης:

```
msg = input("Give binary number (14-bit): ");
N = length(msg);

while (N ~= 14)
  msg = input("Give binary number (14-bit): ");
  N = length(msg);
endwhile

msg = computeParity(msg);

disp("Original message + parity bit: "), disp(msg)
```

Κώδικας 2: Υλοποίηση της πρώτης άσκησης

Δίνοντας στον κώδικά 2 το δυαδικό μήνυμα `10001101011000` [1] επιστρέφεται ένα το μήνυμα μαζί με το bit ισοτιμίας, το οποίο είναι: `100011010110000`.

Άσκηση 2

Εκφώνηση: Δημιουργείστε κώδικα που να ελέγχει αν υπάρχει σφάλμα σε ένα bit για το μήνυμα των 14 bits που συνοδεύεται από ένα bit άρτιας ισοτιμίας. Τρέξτε τον κώδικά σας για να ελέγξετε το αν έχει φτάσει χωρίς σφάλμα το μήνυμα 10001101011000 του προηγούμενου σκέλους της άσκησης (και το bit ισοτιμίας που το συνοδεύει) στις εξής περιπτώσεις:

1. Άφιξη στον αποδέκτη χωρίς σφάλμα.
2. Άφιξη στον αποδέκτη με ένα σφάλμα (στο 6ο bit, με αρίθμηση των bit μετρώντας από αριστερά).
3. Άφιξη στον αποδέκτη με δύο σφάλματα (στο 6ο bit και στο 12ο bit, με αρίθμηση των bit μετρώντας από αριστερά).

Απάντηση:

Για να ελεγχθεί εάν ένα μήνυμα έχει φτάσει χωρίς σφάλμα στον δέκτη, δημιουργήθηκε η συνάρτηση `validateParity` όπως φαίνεται και παρακάτω:

[1] Η εισαγωγή του μηνύματος στο περιβάλλον Octave πρέπει να γίνει με αυτή την μορφή: [1 0 0 0 1 1 0 1 0 1 1 0 0 0].

```
function isValid = validateParity(msg)
  N = length(msg);
  countOnes = 0;
  for i = 1:N
    if (msg(i))
      countOnes = countOnes + 1;
    endif
  endfor
  if (mod(countOnes, 2) == 0)
    isValid = 1;
  else
    isValid = 0;
  endif
endfunction
```

Κώδικας 3: Περιεχόμενο του αρχείου `validateParity.m`

Χρησιμοποιώντας της συναρτήσεις 1 και 3 υλοποιείται η δεύτερη άσκηση.

```
msg = input("Give binary number (14-bit): ");

msg = computeParity(msg);

msg = bitManipulation(msg, 6);

if (validateParity(msg))
  disp("Message is OK.");
else
  disp("Message is corrupted.");
endif

msg = bitManipulation(msg, 12);

if (validateParity(msg))
  disp("Message is OK.");
else
  disp("Message is corrupted.");
endif
```

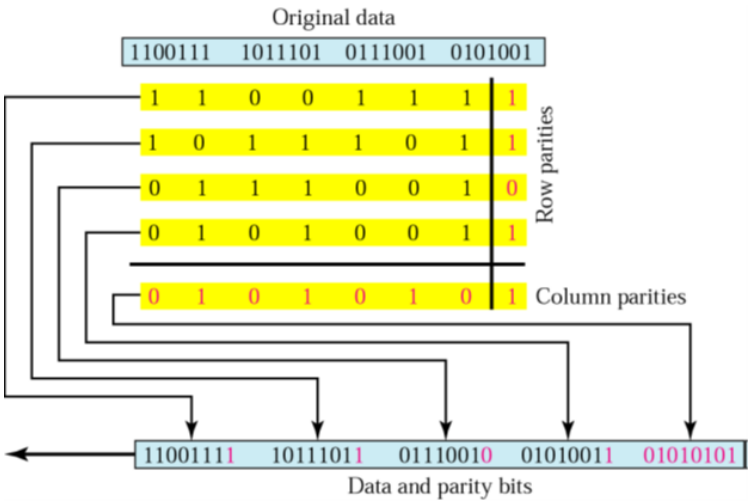
Κώδικας 4: Υλοποίηση δεύτερης άσκησης

Εκτελώντας τον κωδικά 4 για `10001101011000` παρατηρείται πως για την άφιξη με ένα σφάλμα γίνεται αντιληπτό από τον αλγόριθμο ενώ για δυο σφάλματα όχι. Αυτό γίνεται γιατί, ενώ αλλοιώνεται το μήνυμα το πλήθος των bit με τιμή ”1” παραμένει ίδιο.

Άσκηση 3

Εκφώνηση: Δημιουργείστε κώδικα που να διαιρεί ένα μήνυμα των 28 bits σε 4 γραμμές των 7 bits (όπως φαίνεται στο σχήμα), να υπολογίζει τα bits άρτιας

δισδιάστατης ισοτιμίας για κάθε γραμμή και κάθε στήλη και να τα προσαρτά σε αυτές.



Σχήμα 1: Τρίτη άσκηση

Τρέξτε τον κώδικά σας για το συγκεκριμένο μήνυμα που φαίνεται στο παραπάνω σχήμα (original data).

Απάντηση:

Για την υλοποίηση αυτής της άσκησης δημιουργήθηκαν δυο συναρτήσεις.

```
function msgMatrix = createMatrix(msg, M, len)
  i = 1;
  j = 1;

  for k = 1 : len
    msgMatrix(i, j) = msg(k);
    if (mod(k, M) == 0)
      i = i + 1;
      j = 0;
    endif
    j = j + 1;
  endfor
endfunction
```

Κώδικας 5: Συνάρτηση δημιουργίας του πίνακα

Το παραπάνω απόσπασμα κώδικα 5 υλοποιεί μια απλή δημιουργία ενός πίνακα για την αποθήκευση του αρχικού μηνύματος σε ”σπασμένα” μέρη.

Για να ελεγχθεί εάν ένα μήνυμα έχει φτάσει χωρίς σφάλμα στον δέκτη, δημιουργήθηκε η συνάρτηση `validateParity` όπως φαίνεται και παρακάτω:

```
function msg = compute2dParity(msg, N, M)
  for i = 1 : N
  countOnes = 0;
  for j = 1 : M
    if (msg(i ,j))
      countOnes = countOnes + 1;
    endif
  endfor
  if (mod(countOnes, 2) == 0)
    msg(i, M + 1) = 0;
  else
    msg(i, M + 1) = 1;
  endif
endfor

for i = 1 : M + 1
  countOnes = 0;
  for j = 1 : N
    if (msg(j, i))
      countOnes = countOnes + 1;
    endif
  endfor
  if (mod(countOnes, 2) == 0)
    msg(N + 1, i) = 0;
  else
    msg(N + 1, i) = 1;
  endif
endfor
endfunction
```

Κώδικας 6: Υπολογισμός 2 διαστάσεων ισοτιμίας

Τέλος, οι παραπάνω συναρτήσεις (5 και 6) καλούνται στο αρχείο `exe3.m`.

```
msg = input("Give binary number (28-bit): ");

N = 4;
M = 7;
len = 28;

msgMatrix = zeros(N + 1, M + 1);

msgMatrix = createMatrix(msg, M, len);

msgMatrix = compute2dParity(msgMatrix, N, M);

disp(msgMatrix);
```

Κώδικας 7: Υλοποίηση δεύτερης άσκησης

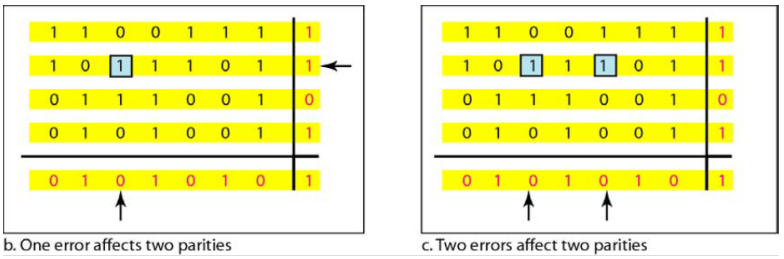
Όπου για είσοδο 11001111011101011100101010 01 [2] επιστρέφεται:

$$\begin{Bmatrix} 11001111 \\ 10111011 \\ 01110010 \\ 01010011 \\ 01010101 \end{Bmatrix}$$

Άσκηση 4

Εκφώνηση: Δημιουργείστε κώδικα που να ανιχνεύει σφάλματα στον πίνακα δύο διαστάσεων (αποτελούμενο από bits μηνύματος και bits ισοτιμίας) που παράγεται από τον κώδικα του 3ου σκέλους της άσκησης.
Τρέξτε τον κώδικά σας για το συγκεκριμένο μήνυμα του 3ου σκέλους της άσκησης και για τις εξής περιπτώσεις:

1. Άφιξη στον αποδέκτη χωρίς σφάλμα.
2. Ο πίνακας φτάνει με σφάλματα σε ένα bit (στο συγκεκριμένο bit που φαίνεται με γαλάζιο χρώμα στο σχήμα 2b).
3. Ο πίνακας φτάνει με σφάλμα σε δύο bits (στα συγκεκριμένα bits που φαίνονται με γαλάζιο χρώμα στο σχήμα 2c).



Σχήμα 2: Τέταρτη άσκηση

Απάντηση: Για τον εντοπισμό εάν έχει υποστεί κάποια αλλοίωσή το μήνυμα χρησιμοποιήθηκε η παρακάτω συνάρτηση:

[2] Η εισαγωγή του μηνύματος στο περιβάλλον Octave πρέπει να γίνει με αυτή την μορφή: [1 1 0 0 1 1 1 1 0 1 1 1 0 1 0 1 1 1 0 0 1 0 1 0 1 0 0 1].

```
function isValid = validate2dParity (msg, N, M)
isValid1 = true; isValid2 = true;
for i = 1 : N + 1
  countOnes = 0;
  for j = 1 : M + 1
    if (msg(i ,j))
      countOnes = countOnes + 1;
    endif
  endfor
  if (mod(countOnes, 2) == 0 && isValid1)
    isValid1 = true;
  else
    isValid1 = false;
  endif
endfor
for i = 1 : M + 1
  countOnes = 0;
  for j = 1 : N + 1
    if (msg(j, i))
      countOnes = countOnes + 1;
    endif
  endfor
  if (mod(countOnes, 2) == 0 && isValid2)
    isValid2 = true;
  else
    isValid2 = false;
  endif
endfor
isValid = isValid1 && isValid2;
endfunction
```

Κώδικας 8: Περιεχόμενο του αρχείου `validate2dParity.m`

Χρησιμοποιώντας τις συναρτήσεις 5, 6 και 8 υλοποιείται και η τέταρτη άσκηση (`exe4.m`).

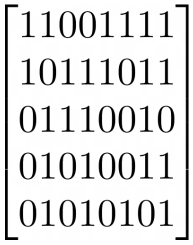
```
msg = input("Give binary number (28-bit): ");

N = 4;
M = 7;
len = 28;

msgMatrix = zeros(N + 1, M + 1);

msgMatrix = createMatrix(msg, M, len);

msgMatrix = compute2dParity(msgMatrix, N, M);

msgMatrix = bitManipulation(msgMatrix, 2, 3);

if (validate2dParity(msgMatrix, N, M))
  disp("Message is OK.");
else
  disp("Message is corrupted.");
endif

msgMatrix = bitManipulation(msgMatrix, 2, 5);

if (validate2dParity(msgMatrix, N, M))
  disp("Message is OK.");
else
  disp("Message is corrupted.");
endif
```

Κώδικας 9: Υλοποίηση τέταρτης άσκησης

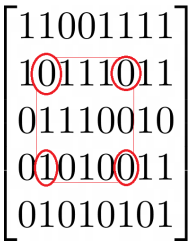
Και στις δυο περιπτώσεις (που προέκυψαν σφάλματα) ο αλγόριθμος -άρτιας- διδιάστατης ισοτιμίας εντοπίζει πως έχει υποστεί το μήνυμα κάποια αλλοίωση.

Άσκηση 5

Ερώτηση: Υπάρχουν κάποια πρότυπα σφάλματος που δεν ανιχνεύονται στη δισδιάστατη ισοτιμία; Αν ναι, ποια είναι αυτά;
Απάντηση: Η δισδιάστατη ισότιμα φαίνεται να είναι ιδιαίτερα αποδοτική στον εντοπισμό διάσπαρτων σφαλμάτων. Ωστόσο, στην περίπτωσή όμως που σχηματιστεί ένα τετράγωνο από σφάλματα, όπως φαίνεται στο σχήμα 4, τότε αδυνατεί να εντοπίσει την αλλοίωση των δεδομένων.



Σχήμα 3: Πριν την αλλοίωση



Σχήμα 4: Αφού έχουν υποστεί ”τετραγωνική” αλλοίωση

Έστω ότι στο παραπάνω σχήμα οι αριθμοί που βρίσκονται στους κύκλους αλλάζουν στην αντίθετη τιμή τους, τότε ο αλγόριθμος δεν θα είναι σε θέση να εντοπίσει κάποιο σφάλμα και θα θεωρήσει πως τα δεδομένα είναι ορθά.

Πηγές

[1] Stallings W., *Επικοινωνίες Υπολογιστών και Δεδομένων*, Εκδόσεις Τζιόλα, 2011.