Εργασία Ψηφιακές Επικοινωνίες - Parity Check (Lab07)

Ονοματεπώνυμο: Ασημάκης Κύδρος
ΑΕΜ: 3881
asimakis@csd.auth.gr

11 Ιουνίου 2022

0 Εισαγωγή

Θα προσομοιώσουμε την διαδικασία ελέγχου σημάτων με βάση την μέθοδο της ισοτιμίας.

1 Ισοτιμία σε μια διάσταση

Αποφασίζουμε αυθαίρετα πως το άρτιο ή περιττό πλήθος των άσσων σε ένα σήμα χαρακτηρίζει την εγκυρότητά του. Ζητώντας το άρτιο πλήθος λέμε ότι εξετάζουμε την **άρτια** ισοτιμία, διαφορετικά την **περιττή**.

Εδώ θα περιοριστούμε στην άρτια ισοτιμία. Σαρώνουμε το σήμα-string και προσθέτουμε 0 ή 1 στο τέλος του (Parity Bit) ώστε να εξισορροπήσουμε το πλήθος των 1 σε άρτιο. Κατά σύμβαση, ορίζουμε ότι ο έλεγχος ήταν επιτυχής αν το parity bit δεν ήταν 1.

```
bool even_parity_check(string &signal){
    int ones = 0;
    for(char bit : signal) if (bit == '1') ones++;
    signal += ones % 2 == 0 ? '0' : '1';
    return signal[signal.size() - 1] != '1';
}
```

Μπορούμε να ελέγξουμε πλέον την εγκυρότητα. Προσομοιώνουμε το κανάλι θορύβου

```
bool noise_channel(string &signal, const vector<int> &positions){
    for(int index : positions) //scan for out of bounds
        if (index < 0 || index >= (int)signal.size())
            return false;
    for(int index : positions) //flip corresponding bits
        signal[index] = signal[index] == '0' ? '1' : '0';
    return true;
}
```

το οποίο θα αντιστρέψει τα bits με index που ορίζονται στη παράμετρο positions. Με τη μέθοδο της ισοτιμίας, βασιζόμαστε στο ότι ο δέκτης θα μετρήσει με τη σειρά του τους άσσους και ανάλογα το πλήθος τους, θα καταλάβει αν συνέβη λάθος κατά την αποστολή.

Επομένως, σκανάρωντας το σήμα αποκρινόμαστε αν υπέστη βλάβη:
Αποτελέσματα από σκανάρισμα ισοτιμίας:
1) χωρίς λάθη
2) λάθος στο 6ο bit (index 5)
3) λάθη στο 6ο και 12ο bit (indexes 5, 11)

```
Exercise 2:
DATA ACCEPTED
DATA REJECTED
DATA ACCEPTED
```

Βέβαια, αν συμβεί άρτιος αριθμός λαθών στο σήμα, το ένα θα εξουδετερώσει την επίδραση του άλλου και η μέθοδος αυτή θα αποτύγχει. Το φαινόμενο αυτό γίνεται εμφανές παραπάνω, τρέχοντας το παράδειγμα **scan_for_flaws(signal, {5, 11})**.

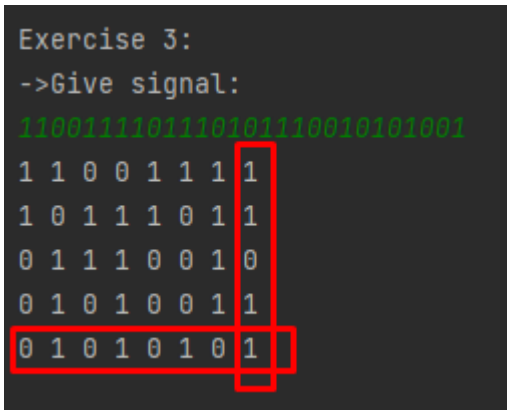
Καταλήγουμε πως η μονοδιάστατη ισοτιμία δεν είναι πολύ αξιόπιστη, αλλά είναι πολύ απλή και φθηνή σε κόστος και πολυπλοκότητα. Η αξιοπιστία της αυξάνεται σημαντικά αν την επεκτείνουμε σε δύο διαστάσεις.

2 Δισδιάστατη ισοτιμία

"Σπάμε" το σήμα και δημιουργούμε με τα bits του έναν πίνακα. Μπορούμε τώρα να θεωρήσουμε κάθε γραμμή και κάθε στήλη ένα καινούριο σήμα, και να κάνουμε μονοδιάστατη ισοτιμία σε αυτά για να καταλείξουμε σε πολλά parity bits:

```
void even_parity_check(matrix &mtrx){
    //check rows and add appropriate parity bit in each one
    for(auto &row : vector<char> &  : mtrx){
        string temp;
        for(char bit : row) temp += bit;
        row[mtrx[0].size() - 1] = even_parity_check( &: temp) ? '0' : '1';
    }
    //do the same for each column
    for(int i = 0, width = (int)mtrx[0].size(); i < width; i++){
        string temp;
        for(auto &row : vector<char> &  : mtrx) temp += row[i];
        mtrx[mtrx.size() - 1][i] = even_parity_check( &: temp) ? '0' : '1';
    }
}
```

Αποτέλεσμα δισδιάστατης ισοτιμίας για το σήμα 11001111011101011100101010001:



Μη ανιχνεύσιμα λάθη είναι πλέον πολύ πιο δύσκολα να συμβούν, καθώς πρέπει να "ξεγελάσουν"όλα τα parity bits:

Επέκταση του καναλιού θορύβου σε δύο διαστάσεις

```
bool noise_channel(matrix &mtrx, const pairs &positions){
    //scan for out of bounds
    for(const auto &[x : const int , y : const int ] : positions)
        if(x < 0 ||  y < 0  ||
           x >= mtrx.size() || y >= mtrx[0].size())
            return false;
    //flip corresponding bits
    for(const auto &[x : const int , y : const int ] : positions)
        mtrx[x][y] = mtrx[x][y] == '0' ? '1' : '0';
    return true;
}
```

Υποσημείωση: ονόματα συντόμευσης για απλότητα

```
using matrix = vector<vector<char>>;
using pairs  = vector<pair<int, int>>;
```

Αποτελέσματα από σκανάρισμα ισοτιμίας:
1) χωρίς λάθη
2) λάθος στο (1, 2)
3) λάθη στο (1, 2) και (1, 4)

```
Exercise 4:
DATA ACCEPTED
DATA REJECTED
DATA REJECTED
```

Μπορούν, όμως, να συμβούν. Η μορφή τους προσεγγίζει ορθογώνιο με πλευρές τα λάθη σε άρτιο πλήθος:

1.


2.


3.


4.


Τρέχοντας το παράδειγμα **scan_for_flaws(mtrx, {{1, 2}, {1, 4}, {2, 2}, {2, 4}})** επιβεβαιώνεται η περίπτωση 2).

Συμπεραίνουμε πως η δισδιάστατη μέθοδος είναι εξαιρετικά cost-effective, καθώς παρέχει μεγάλη αξιοπιστία στα αποτελέσματα για τον μικρό φόρτο που απαιτεί για την υλοποίησή της.