ELEMENTE INTRODUCTIVE PRIVIND INFORMAȚIA

*1.*PROBLEME PRIVIND MĂSURAREA INFORMAȚIEI.

1.1 Într-o urnă sunt plasate bilete cu numele a 5 femei şi 3 bărbați. Dacă se comunică faptul că a fost extras un bilet cu numele unui bărbat, câtă informație a fost transmisă?

1.2 Dispunând de un pachet de 52 de cărți, plasate în ordine aleatoare, cu fețele în jos, câți biți de informație se vor obține la extragerea:

a. primei cărți,
b. celei de-a cincea cărți,
c. a ultimei cărți.

1.3 Fie **X** un număr binar de **n** biți (**n** > 3). Se comunică faptul că primii 3 biți ai numărului **X** sunt 001. Câți biți de informație despre numărul **X** au fost transmişi ?

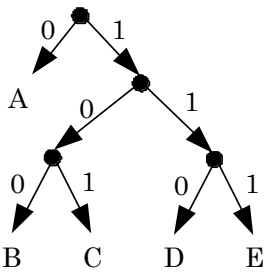
2 PROBLEME PRIVIND DETECTAREA ŞI CORECTAREA ERORILOR, COMPRESIA

2.1 Diferitele scheme de codificare, inclusiv metoda Huffman, dedică mai mulți biți pentru codificarea:

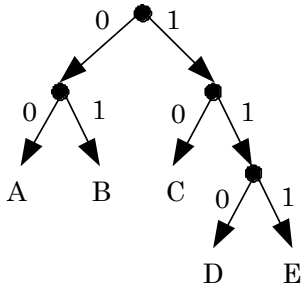
- a simbolurilor care transportă mai multă informație,
- a simbolurilor care transportă cea mai puțină informație,
- a simbolurilor care par a se repeta consecutiv,
- a simbolurilor care conțin informație redundantă ?

2.2 Se dau următorii arbori de decodificarea Huffman, pentru codurile de lungime variabilă corespunzătoare următoarelor 5 simboluri: A, B, C, D şi E:

Arborele 1



Arborele 2



Utilizând **Arborele 1** să se decodifice următorul mesaj: "01000111101".

2.3 Se presupune că are loc codificarea de mesaje cu următoarele probabilități de apariție pentru fiecare dintre cele 5 simboluri: p(A) = 0,5; p(B) = p(C) = p(D) = p(E) = 0,125. Care dintre cele două posibilități de codificare de mai sus (**Arborele 1** sau **Arborele 2**) va genera cele mai scurte mesaje codificate?

2.4 Utilizând probabilitățile pentru A, B, C, D şi E date mai sus, să se construiască un arbore binar de decodificare pentru coduri de lungime variabilă folosind un algoritm simplu "greedy", după cum urmează:

- Se începe cu setul de S simboluri, care trebuie codificate, împreună cu probabilitățile P(x) pentru fiecare simbol x. Suma tuturor probabilităților este 1. Probabilitățile măsoară frecvențele de apariție ale fiecărui simbol în şirul de intrare.
- Se repetă următorii paşi până când în S rămâne un singur simbol:

Se selectează doi membri din S, care au asociate cele mai mici probabilități. În exemplul dat, D şi E pot fi primele noduri alese.

Se înlătură simbolurile selectate din S şi se creează un nou nod în arborele de decodificare ai cărui copii (subnoduri) sunt simbolurile înlăturate. Se marchează ramura din dreapta cu "0" şi ramura din stânga cu "1". Astfel, la prima iterație, se va forma primul nod intern, de pe cel mai jos nivel (care conduce la D şi E).

Se adaugă la S noul simbol (de exemplu "DE"), care va reprezenta un nou nod. Se atribuie acestui nou simbol o probabilitate egală cu suma probabilităților simbolurilor pe care le înlocuieşte.

2.5. Codificarea Huffman este utilizată pentru a codifica compact o serie de păsări, dintr-o rezervație. Daca 50% dintre păsări sunt grauri, iar restul îl reprezintă alte specii, câți biți se vor folosi pentru a codifica graurii?

2.6 Studentul **Bitescu**, de la Calculatoare, a inventat un nou limbaj, AEI,care constă numai în 5 vocale: "A", "E", "I", "O" şi "U". Acestea apar în mesaje cu următoarele probabilități:

Litera	Probabilitatea de apariție
A	p(A) = 0,15
E	p(E) = 0,4
I	p(I) = 0,15
O	p(O) = 0,15
U	p(U) = 0,15

2.6.1 Dacă se afirmă că prima litera a unui mesaj este “A”, să se stabilească expresia pentru numărul de biți de informație furnizați.

2.6.2 Bitescu încearcă să găsească un cod de lungime fixă pentru AEI, care să-i permită să detecteze şi să corecteze erorile la nivelul unui singur bit. Să se prezinte restricțiile pe care le va întâmpina Bitescu la codificarea fiecărei litere, în vederea realizării dezideratelor propuse.

2.6.3 Bitescu este interesat ca să transmită mesaje AEI cu cât mai puțini biți posibil. Se consideră că fiecare literă este codificată separat. Ajutați-l să creeze un cod de lungime variabilă care să minimizeze numărul mediu de biți transmişi pentru fiecare literă a mesajului.

2.7 Pentru protecția informației stocate sau transmise se pot adăuga datelor biți de verificare, care vor facilita detectarea şi corectarea erorilor. Astfel, o schema pentru detectarea erorilor singulare de 1 bit constă în adăugarea unui bit de paritate **p**:

$$b_0\, b_1\, b_2\, b_{N-1}\, p$$

Când se utilizează paritatea pară, **p** se alege astfel încât numărul de biți egali cu “1“, inclusiv în bitul de paritate să fie par; la paritatea impară numărul de biți va fi impar. În continuare se va considera paritatea pară.

Care dintre şirurile de biți următoare, prevăzute cu bit de paritate au erori detectabile?

(1) 11101101111011011

(2) 11011110101011011

(3) 10111110111011110

(4) 00000000000000000

2.8 Pentru a construi un cod cu corectarea erorii (ECC) se vor adăuga biți suplimentari, pentru a depista pozițiile biților eronați. Există, în acest sens, numeroase scheme. Una dintre acestea, pentru detectarea şi corectarea unei erori singulare, consta în organizarea datelor pe linii şi coloane, adăugând apoi biții de paritate pară pentru fiecare linie şi fiecare coloană. Următoarea organizare va asigura protecția a 9 biți de date:

b0,0 b0,1 b0,2 plinie0

b1,0 b1,1 b1,2 plinie1

b2,0 b2,1 b2,2 plinie2

pcol0 pcol1 pcol2

O eroare singulară de 1 bit de date $(b_{I,J})$ va genera două erori de paritate, una în linia **I** şi cealaltă în coloana **J.** O eroare singulară într-unul din biții de paritate va genera numai o singură eroare de paritate, pentru linia sau coloana corespunzătoare. Astfel, după calculul parității, pentru fiecare linie şi coloană, în cazul apariției unei erori pentru linia şi coloana dată, inversând valoarea bitului de la intersecția acestora se va realiza corectarea erorii. Dacă se detectează o singură eroare de paritate, datele sunt corecte iar eroarea a fost într-unul dintre biții de paritate.

Să se furnizeze datele corecte pentru fiecare dintre blocurile de date protejate cu ECC, la nivel de linii şi coloane

(1) 1011 (2) 1100 (3) 000 (4) 0111

0110 0000 111 1001

0011 0101 10 0110

011 100 100

2.9 Liniile şi coloanele ECC pot detecta, de asemenea, erori de câte doi biți (ex., 2 biți de date sunt biți de verificare modificați). Să se evidențieze tipurile de erori la nivelul a 2 biți, care nu se pot detecta.

2.10 În perioada în care se utilizau cartele perforate, cifrele zecimale erau reprezentate prin codurile speciale *2 din 5*. După cum arată şi numele două dintre cele 5 poziții binare erau încărcate cu unități după cum se vede din tabelul de mai jos:

Codul 2 din 5	Zecimal
11000	1
10100	2
10010	3
10001	4
01100	5
01010	6
01001	7
00110	8
00101	9
00011	0

Care este cea mai mică distanță Hamming între două coduri *2 din 5?.*

2.10.a Să se caracterizeze tipurile de erori (ex., de unul sau doi biți), care pot fi sigur/fiabil detectate în *codul 2 din 5*.

2.10.b Paritatea pară reprezintă o altă schemă de detectare a erorilor. Daca se trece de la codul *2 din 5* la un cod de 5 biți, care include un bit de paritate pară , câte date pot fi suplimentar codificate?

2.11 Codul Hamming pentru corectarea unei erori singulare necesita aproximativ $\log_2(N)$ biți de verificare, pentru a corecta erori singulare, de un bit. Se începe cu reenumerarea biților de date cu indici care nu sunt puteri ale lui 2:

Indicii pentru16 biți de date = 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21

Ideea este aceea de a calcula biții de verificare pentru subseturile de date, astfel încât o eroare de 1 bit să producă un set de erori de paritate, care în mod unic indică indicele bitului eronat:

p0 = paritate para pentru biții de date 3, 5, 7, 9, 11, 13, 15, 17, 19, 21

p1 = paritate para pentru biții de date 3, 6, 7, 10, 11, 14, 15, 18, 19

p2 = paritate para pentru biții de date 5, 6, 7, 12, 13, 14, 15, 20, 21

p3 = paritate para pentru biții de date 9, 10, 11, 12, 13, 14, 15

p4 = paritate para pentru biții de date 17, 18, 19, 20, 21

De remarcat faptul că fiecare bit de date apare în cel puțin două calcule de bit de paritate, astfel încât o eroare de un singur bit de date va produce cel puțin două erori de paritate. Atunci când se verifică câmpul de date protejat, dacă numărul de erori de paritate este egal cu 0 sau 1, biții de date sunt în regulă (mai exact, o eroare de paritate singulară arată că un bit de paritate este incorect). Dacă sunt detectate două sau mai multe erori de paritate, atunci erorile identifică exact care bit a fost eronat.

2.11.1 Care este relația între indicele unui bit particular de date şi subsetul de verificare în care acesta apare? (Indicație: se consideră reprezentarea binară a indicelui)

2.11.2 Dacă calculele de paritate referitoare la **p0**, **p2** şi **p3** indică erori, considerând o eroare de un bit, care este indicele bitului eronat de date?

2.11.3 SECC Hamming nu detectează toate erorile de câte doi biți. Să se caracterizeze tipurile de erori la nivelul a doi biți, care nu vor fi detectate. Să se sugereze o extensie la SECC Hamming, care permite detectarea tuturor erorilor la nivelul a doi biți.

RĂSPUNSURI

1.1 Cunoscând că, în total sunt **n** = 8 nume şi că extragerea a constatat că este vorba de numele unui bărbat, numărul selecțiilor posibile s-a redus la **m** = 3. Conform formulei $\log_2\left(\frac{n}{m}\right)$ s-au obținut $\log_2\left(\frac{8}{3}\right)$ biți de informație. Probabilitatea de a extrage numele unui bărbat este $p_b = \frac{3}{8}$ astfel cantitatea de informație obținută va fi: $\log_2\left(\frac{1}{p_b}\right) = \log_2\left(\frac{1}{\frac{3}{8}}\right) = \log_2\left(\frac{8}{3}\right)$.

1.2 (a) $\log_2\left(\frac{52}{1}\right)$ biți de informație.

(b) $\log_2\left(\frac{48}{1}\right)$ biți de informație.

(b) $\log_2\left(\frac{1}{1}\right) = 0$ biți de informație.

1.3 Intuitiv, întrucât s-a dat informație despre 3 biți din X, se apreciază că au fost dați 3 biți de informație. Trecând la formule: există 2^n numere binare de câte **n** biți şi 2^{n-3} numere binare care încep cu 001. Astfel au fost transmişi: $\log_2\left(\frac{2^n}{2^{n-3}}\right) = \log_2(2^3) = 3$ biți de informație.

2.1 Pentru a menține mesajele cât mai scurte posibil, simbolurile care apar cu frecvența mare sunt codificate cu mai puțini biți, în timp ce simbolurile care apar rar sunt codificate cu mai mulți biți.

2.2 Pentru decodificare se porneşte de la rădăcina arborelui, parcurgând rangurile pe măsura traversării acestuia pentru a atinge un nod frunză. Se repetă procedura până la epuizarea tuturor rangurilor numărului. Prelucrarea se efectuează de la stânga la dreapta.

"0" => A; "100" => B; "0" => A; "111" => E; "101" => C

2.3 Folosind **Arborele 1**, lungimea aşteptată a codificării pentru un simbol este:

$$1 * p(A) + 3 * p(B) + 3 * p(C) + 3 * p(D) + 3 * p(E) = 2{,}0$$

Folosind **Arborele 2**, lungimea aşteptată a codificării pentru un simbol este:

$$2 * p(A) + 2 * p(B) + 2 * p(C) + 3 * p(D) + 3 * p(E) = 2{,}25$$

În concluzie, codificarea pe baza Arborelui 1 va conduce, în medie, la cele mai scurte mesaje.

2.4 $S = \left\{ \frac{A}{0,5}, \frac{B}{0,125}, \frac{C}{0,125}, \frac{D}{0,125}, \frac{E}{0,125} \right\}$

Se aleg arbitrar D & E

Se codifică: "0" => D, "1" => E

$$S = \left\{ \frac{A}{0,5}, \frac{B}{0,125}, \frac{C}{0,125}, \frac{DE}{0,25} \right\}$$

Se aleg B & C

Se codifică: "0" => B, "1" => C

$$S = \left\{ \frac{A}{0,5}, \frac{BC}{0,25}, \frac{DE}{0,25} \right\}$$

Se aleg BC & DE

Se codifică: "00" => B, "01" => C, "10" => D, "11" => E

$$S = \left\{ \frac{A}{0,5}, \frac{BCDE}{0,5} \right\}$$

Se aleg A & BCDE

Codificarea: "0" => A, "100" => B, "101" => C, "110" => D, "111" => E

$$S = \left\{ \frac{ABCDE}{1,0} \right\}$$

Acesta este **Arborele 1** de mai sus. Selectarea lui D şi E ca prime simboluri, care se combină, a fost arbitrară. Puteau fi alese oricare alte două simboluri dintre B, C, D şi E. Astfel, există multe codificări plauzibile pentru acest exemplu, date fiind probabilitățile egale ale acestor 4 simboluri.

2.5 Folosind algoritmul de mai sus - 1 bit.

2.6.1 Folosind formula dată în cadrul cursului de va obține: $\log_2 \left(\frac{1}{p(A)} \right) = \log_2 \frac{1}{0,15}$

2.6.2 Fiecare cod trebuie să difere de celelalte coduri prin cel puțin 3 poziții, adică să aibă distanța Hamming ≥ 3. Aceasta va asigura faptul că fiecare cuvânt-cod recepționat (chiar şi acelea cu un singur bit eronat) vor fi asociate cu o sursă dată de cod.

2.6.3. Folosind algoritmul "greedy", descris mai sus se va obține:

$$S = \left\{ \frac{A}{0,15}, \frac{E}{0,4}, \frac{I}{0,15}, \frac{O}{0,15}, \frac{U}{0,15} \right\}$$

Se aleg arbitrar O & U

Se codifică: "0" => O, "1" => U

$$S = \left\{ \frac{A}{0,15}, \frac{E}{0,4}, \frac{I}{0,15}, \frac{OU}{0,3} \right\}$$

Se aleg A & I

Se codifică: "0" => A, "1" => I

$$S = \left\{ \frac{AI}{0,3}, \frac{E}{0,4}, \frac{OU}{0,3} \right\}$$

Se aleg AI & OU

Se codifică: "00" => A, "01" => I, "10" => O, "11" => U

$$S = \left\{ \frac{AIOU}{0,6}, \frac{E}{0,4} \right\}$$

Se aleg E & AIOU

Se codifică: "0" => E, "100" => A, "101" => I, "110" => O, "111" => U

$$S = \left\{ \frac{AEIOU}{0,1} \right\}$$

Se poate observa că atribuirile simbolurilor pentru codificările “0” şi”1” au fost arbitrare şi că, la fiecare nivel, ele au putut fi interschimbate. Astfel, interschimbarea la nivelul ultimului pas ar fi dus la următoarea codificare:

"1" => E, "000" => A, "001" => I, "010" => O, "011" => U,

care conduce la aceiaşi valoare medie de biți/simbol ca şi codificarea anterioară.

2.7 Şirurile 1 şi 3 au erori detectabile. De observat că paritatea permite detectarea erorilor de un bit (sau a erorilor de un număr impar de biți).

2.8 (1**)** **0**011 (2) 1100 (3) 000 (4) 011**0**

0110	0000	1**0**1	1001
0011	0101	10	0110
011	100		100

(1) şi (3) au erori de un singur bit, (2) nu are erori detectabile, iar (4) are o eroare în bitul de paritate. Rangurile reprezentate în roşu constituie valorile corecte.

2.9 Dacă biții de paritate detectează o eroare pentru o singură coloană şi o singură linie, aceasta se va interpreta ca o eroare în poziția corespunzătoare a datei. Apariția a două erori nu a fost detectată, şi mai rău, valoarea din poziția corespunzătoare a datei a fost modificată, de la cea corectă la una incorectă.

2.10 Distanța Hamming între două coduri este numărul de poziții de bit în care cele două coduri diferă. În cazul de față distanța Hamming este egală cu 2.

2.10.a Codurile Hamming cu distanța 2 pot detecta o eroare de 1 bit. Codul *2 din 5* poate detecta erori de 3 biți, dar nu erori de 2 biți. În mod normal, când se spune că un cod detectează erori de **n** biți, se consideră că detectează erori de **m** biți, unde **m < n**. Conform acestei convenții, se poate spune că, *codul 2 din 5* detectează erori de 1 bit.

2.10.b La un cod cu 4 biți de date şi 1 bit de paritate există 16 posibilități de a codifica datele, cu 6 mai mult decât în cazul codului *2 din 5*.

2.11.1 Rangul **i** în reprezentarea binară a indicelui bitului de date indică faptul că indicele trebuie să fie inclus în calculul lui **pi.** De exemplu, primul bit de date (al cărui indice este 3 = 00011) va fi utilizat în calculul parității pentru biții de paritate **p0** şi **p1**. În mod asemănător bitul 6 de date (care are indicele 10 = 01010) va fi utilizat pentru calculul biților de paritate **p1** şi **p3.** Întrucât nici unui bit de date nu i-a fost

atribuit un indice, care să fie o putere a lui 2, se poate garanta faptul că fiecare bit de date este utilizat în calculul a cel puțin 2 biți de paritate diferiți.

2.11.2 Trebuie să se determine indicele datei, care apare în calculele pentru **p0**, **p2** şi **p3,** dar şi in calculele pentru **p1** şi **p4**. Indicii 13 şi 15 apar în **p0**, **p2** şi **p3**, dar indicele 15 apare în calculul pentru **p1**. Astfel, indicele bitului de date eronat este 13.
Se poate construi indicele bitului eronat de date direct din calculul de paritate folosind **ei** = 1, dacă **pi** este eronat sau **ei** =0, dacă nu este eronat. Astfel, dacă **p0**, **p2** şi **p3** sunt eronați, iar alți nu sunt, indicele este **e4e3e2e1e0** = 01101 = 13_{10}.

2.11.3 Dacă apare o eroare în doi biți de paritate separați, aceasta se va interpreta greşit ca o eroare la nivelul unui bit singular de date. De asemenea, când anumite perechi de biți de date au erori, eroarea la nivelul celor doi biți se insinuează ca fiind eroare la nivelul unui singur bit într-un alt bit, fără nici o legătură (ex. Biții cu indicii 19 şi 21 sunt eronați)
Aceste situații pot fi detectate prin adăugarea unui bit suplimentar de paritate, **p5**, care este bitul de paritate pentru ceilalți biți de date şi paritate. Dacă un bit singular de date este eronat, acest bit va indica o eroare. Dacă exact doi biți sunt eronați, **p5** nu va indica eroare, dar **p0, p1,…p4** vor indica prezența unor erori (cu toate că indicația se referă la un bit eronat de date). Astfel, dacă examinarea lui **p0, p1, ..., p4** sugerează apariția unei erori, se va verifica **p5**, pentru a fi siguri ca a apărut numai o eroare. Dacă **p5** nu indică o eroare, atunci a apărut o eroare la nivelul a doi biți.