Laborator PS
Algoritmi de compresie a datelor

Lucrarea 1: Introducere în Compresia Datelor

Lucrarea 2: Algoritmul Shannon-Fano

Lucrarea 3: Algoritmul Huffman static

Lucrarea 4: Algoritmul Huffman dinamic

Prof. Dan Ştefănoiu <dan.stefanoiu@acse.pub.ro>

Ş.l. Alexandru Dumitraşcu <alexandru.dumitrascu@acse.pub.ro>

A. Obiective

Implementarea Algoritmului de compresie Shannon-Fano şi testarea performanței acestuia, prin comparație cu cele ale programelor de uz general **WinZIP** şi **WinRAR**.

B. Suport teoretic

Ideea de bază a minimizării redundanței prin tehnica Shannon-Fano este aceea a construcției unui arbore binar asociat setului de date $\mathcal{D}$, care va permite recodificarea simbolilor. Acesta se construieşte parcurgând paşii Algoritmului 2.1.

Algoritmul 2.1. *Construcția alfabetului de ordin 0 şi a arborelui binar pentru compresia Shannon-Fano.*

1. Se parcurge setul de date $\mathcal{D}$ în mod secvențial, citind fiecare simbol în parte. Odată cu această operație:
   - se construieşte alfabetul de ordin 0 ($\mathcal{A}^0$), format numai din simboli distincți;
   - se contorizează numărul aparițiilor fiecărui simbol $s \in \mathcal{A}^0$, asociindu-i un contor $\mathcal{N}(s) \in \mathbb{N}^*$.
2. În mod normal, ar trebui evaluată frecvența de apariție a fiecărui simbol ca raportul dintre valoarea contorului şi volumul setului de date:

$$\nu(s) = \frac{\mathcal{N}(s)}{\#\mathcal{D}},$$

unde semnul „#” indică aici operația de luare a cardinalului mulțimii. Deoarece volumul setului de date este constant (şi egal cu suma tuturor contoarelor), adică:

$$\#\mathcal{D} = \sum_{s \in \mathcal{A}^0} \mathcal{N}(s),$$

se poate renunța la împărțire, luând în considerare numai valorile contoarelor.

3. Se aranjează simbolii alfabetului $\mathcal{A}^0$ în ordinea descrescătoare a valorilor contoarelor; în caz de egalitate, ei se ordonează lexicografic. Aşadar, $\mathcal{A}^0 = \{...,s,t...\}$, cu $\mathcal{N}(s) \geq \mathcal{N}(t)$.
4. Se divizează alfabetul ordonat $\mathcal{A}^0$ în două *subalfabete* situate în partea stângă (*Left*) $\mathcal{A}^{0L}$, respectiv partea dreaptă (*Right*) $\mathcal{A}^{0R}$. Se urmăreşte satisfacerea a două condiții de bază:
   - ordinea simbolilor trebuie conservată şi în cadrul subalfabetelor;
   - *ponderile* subalfabetelor (adică sumele contoarelor simbolilor constituenți) trebuie să fie cât mai apropiate.

   Astfel, cele două subalfabete nu au în mod necesar acelaşi număr de simboli. Pentru a rezuma, subalfabelete verifică realțiile:

$$\begin{cases} \mathcal{A}^0 = \mathcal{A}^{0L} \cup \mathcal{A}^{0R}; \\ \mathcal{A}^{0L} \cap \mathcal{A}^{0R} = \varnothing \ ; \\ \mathcal{N}\left(\mathcal{A}^{0L}\right) = \sum_{s \in \mathcal{A}^{0L}} \mathcal{N}(s) \cong \mathcal{N}\left(\mathcal{A}^{0R}\right) = \sum_{s \in \mathcal{A}^{0R}} \mathcal{N}(s). \end{cases}$$

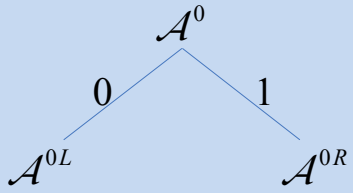
Notă

În construcția subalfabetelor $\mathcal{A}^{0L}$ şi $\mathcal{A}^{0R}$, se va pleca simultan de la cele două „capete” ale alfabetului $\mathcal{A}^0$. Se adaugă simboli în $\mathcal{A}^{0L}$ sau în $\mathcal{A}^{0R}$, mergând spre centru şi comparând mereu ponderile parțiale ale acestora, până se parcurg toate elementele lui $\mathcal{A}^0$. Acest mecanism este ilustrat mai jos:

$$\mathcal{A}^0 = \{s_1, s_2, ..., s_k, ..., s_{N-1}, s_N\}$$
$$\longrightarrow \qquad \longleftarrow$$
$$\mathcal{A}^{0L} \qquad \mathcal{A}^{0R}$$

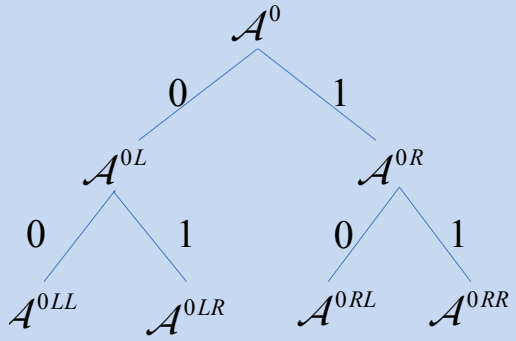
De exemplu, inițial, $\mathcal{A}^{0L}$ îl conține doar pe s_1 şi $\mathcal{A}^{0R}$ doar pe s_N. Evident, $\mathcal{N}(s_1) \geq \mathcal{N}(s_N)$. Se adaugă apoi s_{N-1} la $\mathcal{A}^{0R}$. Dacă $\mathcal{N}(s_1) \geq \mathcal{N}(s_N) + \mathcal{N}(s_{N-1})$, atunci se adaugă şi s_{N-2} la $\mathcal{A}^{0R}$; altfel, se adaugă s_2 la $\mathcal{A}^{0L}$. Se compară din nou ponderile parțiale şi se decide ce simbol va fi adăugat unuia din cele două subalfabete, până când se epuizează toți simbolii lui $\mathcal{A}^0$.

5. Arborele binar este inițializat astfel: $\mathcal{A}^0$ reprezintă nodul rădăcină, $\mathcal{A}^{0L}$ reprezintă ramura stângă *împachetată* a rădăcinii, iar $\mathcal{A}^{0R}$ reprezintă ramura dreaptă *împachetată* a rădăcinii. Arcul stâng va primi codul 0, iar arcul drept codul 1, ca în figura de mai jos:



Termenul de *împachetată* este sugerat de faptul că ramura respectivă conține germenii altor ramuri care se vor naşte în aceeaşi manieră, până cînd se va ajunge la nodurile terminale ale arborelui, numite şi *frunze*.

6. Se reiterează paşii 4 şi 5 pentru fiecare nod terminal al arborelui (subalfabet), care conține mai mult de un simbol. De exemplu, dacă $\mathcal{A}^{0L}$ şi $\mathcal{A}^{0R}$ conțin cel puțin câte 2 simboli fiecare, atunci structura arborelui la pasul următor va fi cea din figura următoare:



Procesul de construcție a arborelui se opreşte când toate subalfabelete frunzelor se reduc la câte un singur simbol. Noul cod asociat unui simbol se obține parcurgând arborele de la rădăcină pînă la frunza ocupată de acel simbol şi concatenând valorile arcelor prin care trece.

Aşa cum se poate constata cu uşurinţă din acest algoritm, condiţia esenţială în construcţia arborelui binar este cea a echilibrării ponderilor subalfabetelor (exprimată la pasul 4). Aceasta determină lungimea noilor coduri, în funcţie de contoarele asociate. Dacă un simbol este foarte frecvent în setul de date, el va face parte dintr-un subalfabet cu un număr mic de elemente, a cărui defalcare pe subalfabete se va încheia rapid, în apropierea rădăcinii. Codul corespondent va fi, deci, scurt. Simbolii rari vor avea ponderi mici şi vor face parte din subalfabete cu un număr mare de elemente, fapt care le trimite departe de rădăcină în arborele binar. În consecinţă, codul lor va avea o lungime mai mare.

Algoritmul 2.1 este inclus în cele două proceduri de compresie, respectiv decompresie prezentate în Algoritmii 2.2 şi 2.3.

Algoritmul 2.2. *Compresia Shannon-Fano.*

1. Se construieşte arborele binar asociat setului de date $\mathcal{D}$ cu ajutorul Algoritmului 2.1.
2. Se construieşte setul de date comprimat format din:
   a) Alfabetul $\mathcal{A}^0 = \{s_1, ..., s_N\}$ (în ordinea descrescătoare a valorilor contoarelor). Fiecare simbol este reprezentat de codul său original (pe 8 biţi).
   b) Numărul de simboli ai alfabetului $\mathcal{A}^0$, notat cu N (ca mai sus), mai puţin o unitate (adică $N-1$).
   c) Numărul $N-1$ este reprezentat tot pe 8 biţi (deoarece $N < 256$).
   d) Tabela contoarelor asociate simbolilor alfabetului: $\{\mathcal{N}(s_n)\}_{n\in\overline{1,N}}$. Fiecare contor va fi reprezentat pe maxim 32 de biţi (4 octeţi).
   e) Şirul noilor coduri, în ordinea în care se succed simbolii în setul original de date. Fiecare nou cod este produs cu ajutorul arborelui binar, aşa cum s-a menţionat mai sus.

Se constată că, potrivit acestui algoritm, datele transmise pe fluxul de ieşire sunt de două tipuri: utile (şirul noilor coduri) şi auxiliare (celelalte date referitoare la alfabet şi contoare). Dacă setul original de date nu este suficient de mare, informația auxiliară degradează sensibil rata de compresie, putând forța chiar apariția fenomenului de expandare a datelor. Acest algoritm de compresie este eficient numai dacă lungimea informației auxiliare este sensibil inferioară lungimii informației utile sau a setului de date original. Informația auxiliară adăugată aici este, însă, strict necesară în faza de decompresie, chiar dacă, din cauza ei, rata de compresie va fi inferioară celei ideale calculate folosind alfabetul de ordin 0.

Al doilea algoritm descrie maniera de decompresie.

Algoritmul 2.3. *Decompresia Shannon-Fano.*

1. Se citeşte informația auxiliară într-o ordine prestabilită (de exemplu: $N-1$, $\mathcal{A}^0$, $\{\mathcal{N}(s_n)\}_{n\in\overline{1,N}}$).
2. Se construieşte arborele binar plecînd de la informația auxiliară. În acest scop, se apelează la Algoritmul 2.1 (identic cu cel din faza de compresie).
3. Se decriptează şirul noilor coduri (setul de date comprimat) citind informația utilă bit cu bit (adică nu la nivel de octet!). Simultan, se parcurge arborele binar avansând de la rădăcină către frunze, pe un drum indicat de valorile biților citiți. În acest fel, deîndată ce s-a atins o frunză, este emis codul original (pe 8 biți) al simbolului asociat acesteia şi se reîncepe decriptarea din rădăcină. Procesul de decriptare continuă până la epuizarea biților de pe fluxul de intrare.

Este important de observat că algoritmul de decompresie nu necesită cunoaşterea în avans a lungimilor codurilor emise în faza de compresie, tocmai datorită structurii arborescente a modelului statistic. Acesta este un mare avantaj al metodei, deoarece noile coduri pot fi depuse pe fluxul de ieşire în faza de compresie, *fără a mai fi separate între ele*, deci fără a adăuga alte coduri auxiliare. Decriptarea este exactă, datorită faptului că modelul arborescent este construit cu acelaşi algoritm în ambele faze: compresie şi decompresie.

Exemplu

Se doreşte comprimarea-decomprimarea următorului text prin Metoda Shannon-Fano:

$\mathcal{D}$: IT␣IS␣BETTER␣LATER␣THAN␣NEVER.

Simbolul '␣' este utilizat pentru a indica spațiul liber dintre cuvinte (cunoscut sub denumirea englezească de *blank* sau *(white) space*). În figurile care urmează, el va fi identificat de asemenea prin '☐'.

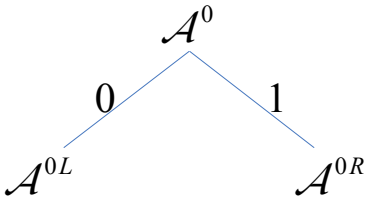
Înainte de a construi arborele binar necesar recodificării simbolilor, trebuie constituit alfabetul asociat setului de date $\mathcal{D}$:

$\mathcal{A}^0$	␣	E	T	R	A	I	N	.	B	H	L	S	V
$\mathcal{N}(s)$	5	5	5	3	2	2	2	1	1	1	1	1	1

Rezultă că ponderea acestui alfabet este: $\mathcal{N}\left(\mathcal{A}^0\right)=30$.

Arborele binar se construieşte, în acest caz, după 5 iterații de divizare în subalfabete. Vom ilustra aceste divizări împreună cu arborii parțiali aferenți.

- Prima iterație:

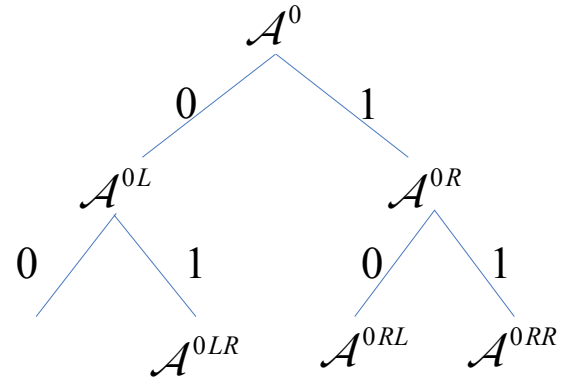


$\mathcal{A}^{0L}$	␣	E	T
$\mathcal{N}(s)$	5	5	5

$\mathcal{A}^{0R}$	R	A	I	N	.	B	H	L	S	V
$\mathcal{N}(s)$	3	2	2	2	1	1	1	1	1	1

Se constată că, întâmplător, ponderile celor două subalfabete coincid în acest caz, fiind egale cu 15. Simbolii subalfabetului $\mathcal{A}^{0L}$, care au toți ponderi egale, au fost aranjați în ordine lexicografică. La fel şi simbolii cu ponderi egale ai subalfabetului $\mathcal{A}^{0R}$.

- A doua iterație:



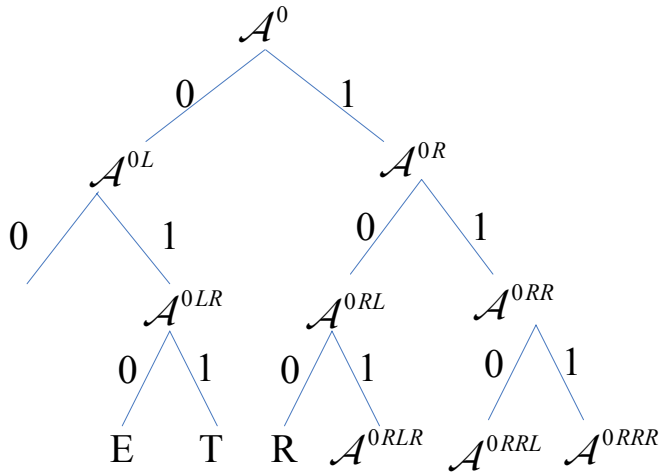
$\mathcal{A}^{0LR}$	E	T
$\mathcal{N}(s)$	5	5

$\mathcal{A}^{0RL}$	R	A	I
$\mathcal{N}(s)$	3	2	2

$\mathcal{A}^{0RR}$	N	.	B	H	L	S	V
$\mathcal{N}(s)$	2	1	1	1	1	1	1

Simbolul '␣', care este unul din cei mai frecvenți ai alfabetului, a şi atins o frunză, fiind recodificat pe numai 2 biți: 00 (în loc de codul original pe 8 biți, care este 0010 0000, adică 0x20=32).

➢ A treia iterație:

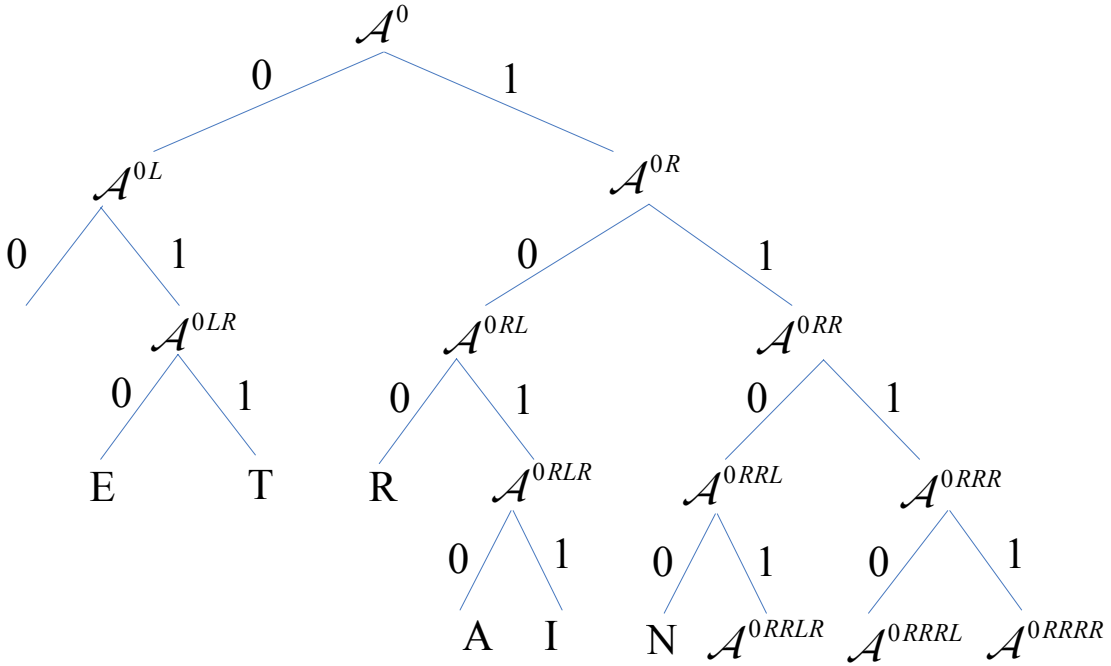


$\mathcal{A}^{0RRR}$	H	L	S	V
$\mathcal{N}(s)$	1	1	1	1

$\mathcal{A}^{0RRL}$	N	.	B
$\mathcal{N}(s)$	2	1	1

$\mathcal{A}^{0RLR}$	A	I
$\mathcal{N}(s)$	2	2

➢ A patra iterație:



$\mathcal{A}^{0RRLR}$	.	B
$\mathcal{N}(s)$	1	1

$\mathcal{A}^{0RRRL}$	S	V
$\mathcal{N}(s)$	1	1

$\mathcal{A}^{0RRRR}$	H	L
$\mathcal{N}(s)$	1	1

□

Numai simbolii foarte rari din textul analizat au mai rămas de recodificat. Cu toate acestea, codurile lor vor avea o lungime de 5 biți, care rămîne inferioară celei originale de 8 biți.

În urma ultimei iterații (a cincea) se obține Figura 2.1. Aceasta ilustrează structura arborescentă a unui model statistic (în speță, alfabet de ordin 0), construită cu ajutorul Algoritmului Shannon-Fano. Ea evidențiază foarte bine principiile recodificării datelor, specificate anterior. Noile coduri nu depăşesc 5 biți în lungime, fapt pe care se sprijină minimizarea redundanței, adică maximizarea ratei de compresie.

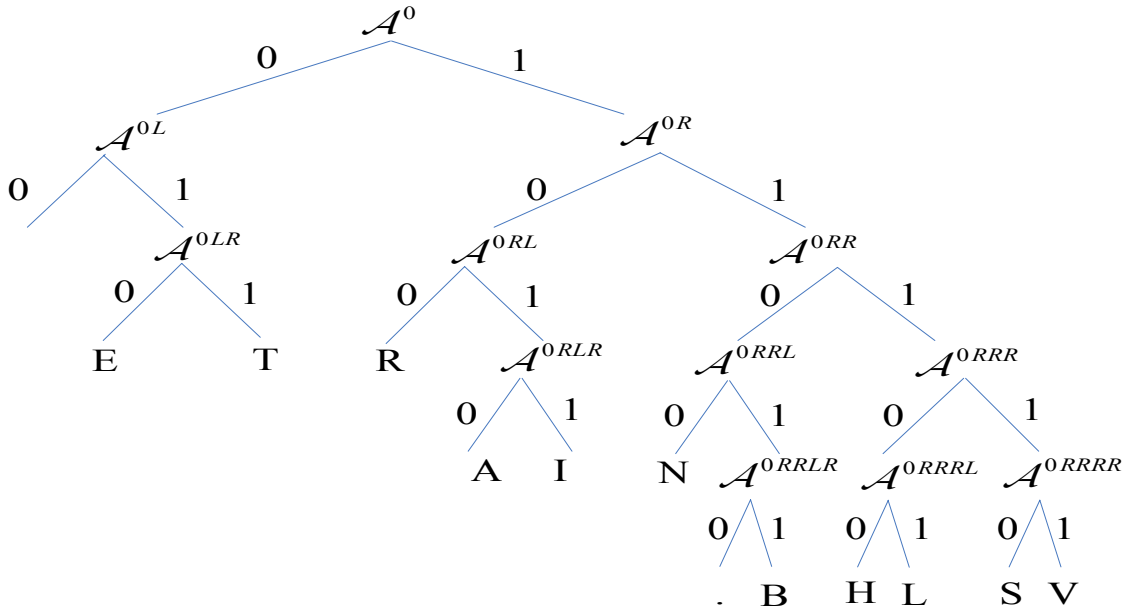


Figura 2.1. *Un arbore binar de tip SHANNON-FANO (alfabet static de ordinul 0).*

După construcția arborelui binar statistic în manieră *statică* (adică după ce întregul set de date a fost parcurs sau citit de pe fluxul de intrare), informația înscrisă pe fluxul de ieşire va avea configurația din Tabelele 2.1 şi 2.2. În Tabelul 2.1, este enumerată informația auxiliară.

Tabelul 2.1. *Informație auxiliară de tip Shannon-Fano.*

Cod	13	32	5	69	5	84	5	82	3	65	2	72	2	78	2	46	1	66	1	72	76	1	83	1	86	1
Info	$\mathcal{N}$	□	$\mathcal{N}$	E	$\mathcal{N}$	T	$\mathcal{N}$	R	$\mathcal{N}$	A	$\mathcal{N}$	I	$\mathcal{N}$	N	$\mathcal{N}$	.	$\mathcal{N}$	B	$\mathcal{N}$	H	L	$\mathcal{N}$	S	$\mathcal{N}$	V	$\mathcal{N}$
Nr. biți	8	8	32	8	32	8	32	8	32	8	32	8	32	8	32	8	32	8	32	8	8	32	8	32	8	32

Codurile numerice au fost exprimate în baza 10, pentru uşurința citirii tabelului. În realitate, aceste coduri sunt binare (şi au cîte 8 biți lungime, pentru simbolii alfabetului). De asemenea, gruparea în perechi a simbolilor cu contoarele corespunzătoare nu este obligatorie. Aranjarea informației auxiliare poate fi realizată şi în altă manieră, dar modul de aranjare trebuie cunoscut şi la decompresia datelor.

Urmează informația utilă (noile coduri de tip Shannon-Fano), ca în Tabelul 2.2.

Tabelul 2.2. *Informație utilă de tip Shannon-Fano.*

Cod	1011	011	00	1011	11110	00	11011	010	011	011	010	100	00	11101	1010
Info	I	T	□	I	S	□	B	E	T	T	E	R	□	L	A
Nr. biți	4	3	2	4	5	2	5	3	3	3	3	3	2	5	4
Cod	011	010	100	00	011	11100	1010	1100	00	1100	010	11111	010	100	11010
Info	T	E	R	□	T	H	A	N	□	N	E	V	E	R	.
Nr. biți	3	3	3	2	3	5	4	4	2	4	3	5	3	3	5

De această dată, noile coduri au fost redate în reprezentare binară, corespunzătoare arborelui statistic din Figura 2.1. Atât codurile informației utile cât şi cele ale informației auxiliare se succed fără separatori între ele, aşa cum ar sugera tabelele anterioare. Între informația auxiliară şi cea utilă ar putea apărea un separator sub forma unui simbol virtual, în absența numărului de simboli ai alfabetului (N). În acest caz, însă, separatorul de informație trebuie să aibă un cod superior lui 255 (eventual, poate avea valoarea 256), ceea ce atrage după sine reprezentarea lui pe 16 biți (în loc de 8, cât ocupă N).

În faza de decompresie, după citirea informației auxiliare (5 octeți consecutivi indică o pereche simbol-contor) şi după construcția arborelui binar asociat, se trece la citirea secvențială, bit cu bit, a informației utile. Fiecare simbol al setului de date este decriptat folosind arborele binar. De exemplu, revenind la Figura 2.1 şi la şirul de coduri binare utile prezentate în Tabelul 2.2, primul bit este 1, ceea ce indică deplasarea din nodul rădăcină $\mathcal{A}^0$ în nodul $\mathcal{A}^{0R}$. Urmează bitul 0, ceea ce ne aduce în nodul $\mathcal{A}^{0RL}$. Un nou bit 1 ne aruncă în nodul $\mathcal{A}^{0RLR}$ şi încă un bit 1 ne permite atingerea frunzei ocupate de simbolul 'I', care va fi înscris pe fluxul de ieşire. Atingerea unei frunze reinițiază căutarea din rădăcina arborelui, astfel că un nou bit citit (aici 0), startează o nouă deplasare spre frunze.

Subliniem încă o dată (aşa cum o ilustrează şi acest exemplu), că nu este necesară cunoaşterea dimensiunilor noilor coduri de compresie, care, în general variază de la un simbol la altul. Acesta constituie un mare avantaj al metodei.

C. Sarcini de lucru

Tema 1 (Implementarea algoritmilor Shannon-Fano)

a. Scrieți cele 3 funcții corespunzătoare Algoritmilor 2.1, 2.2 şi 2.3 .

b. Verificați corectitudinea lor cu ajutorul unui program de test aplicat exemplului din secțiunea precedentă (textul trebuie să fie reconstituit perfect).

c. Evaluați factorul de compresie realizat în acest exemplu (ținând cont şi de informația auxiliară). Este capabilă metoda să realizeze compresia?

Tema 2 (Testarea algoritmilor Shannon-Fano)

a. Scrieți un program care realizează compresia şi decompresia celor 12 fişiere din corpusul de date descris în secțiunea precedentă, cu ajutorul rutinelor de la tema precedentă.

b. Calculați norma diferenței dintre setul inițial de date şi cel decomprimat. Pentru fiecare set de date, indicați factorul de compresie aferent (incluzând informația auxiliară) şi evaluați duratele de compresie, respectiv decompresie (cu ajutorul funcțiilor menționate în lucrarea precedentă). Comentați rezultatul obținut, cu referire la capacitatea de compresie a metodei pentru diferite tipuri de fişiere.

c. Comparați factorii de compresie de la punctul precedent cu cei obținuți prin utilizarea celor două programe de uz general **WinZIP** şi **WinRAR**. (Utilizați funcția **`fread`** pentru a evalua exact numărul de biți ai arhivelor produse de aceste utilitare.) Comentați rezultatele obținute, cu referire la compromisul dintre complexitatea algoritmilor de compresie şi performanțele acestora.