Tema 3 – Structuri de date (CB)
Decodificare

Responsabili temă	Emanuela Haller, Oana Bălan , Rareș Mihai Taerel
Data publicării	**4.05.2016**
Termenul de predare	**19.05.2016, ora 23:55** **Se acceptă teme, cu penalizare de 10 puncte/zi, până la data de 22.05.2016 ora 23:55**

1. Introducere

Criptarea presupune transformarea informației obișnuite (text în clar) într-un text neinteligibil (text cifrat). Decriptarea este procesul invers, ce presupune trecerea de la textul cifrat, neinteligibil, la textul clar. În contexul curent, vom presupune o variantă simplificată în care criptarea presupune o codificare simplă a fiecărui caracter conform unui cod predefinit. Mai exact, textul criptat va fi obținut prin înlocuirea fiecărui caracter al textului clar, cu un cod prestabilit.

Spre exemplu, am putea folosi codul ASCII, care ar presupune că fiecare caracter din textul clar va fi înlocuit cu un număr, codul ASCII al caracterului respectiv.

Un alt exemplu este codul Morse, care oferă o modalitate de transmitere a informației folosind secvențe standardizate de semne sau pulsații scurte și lungi. Principala diferență dintre codul ASCII și codul Morse o reprezintă dimensiunea variabilă a codificărilor.

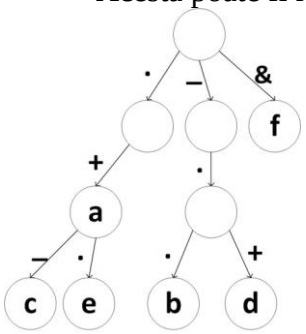
În continuare vom considera codificări în cadrul cărora fiecare caracter are asociat un cod format din unu sau mai multe caractere, codurile având lungimi variabile.

Un astfel de cod poate fi reprezentat cu ajutorul unui arbore, unde fiecare avans în arbore se face conform codificării. Astfel, decodificarea unui anumit caracter poate fi efectuată prin parcurgerea arborelui conform codificării furnizate, până la terminarea șirului furnizat, iar caracterul asociat nodului final va fi caracterul original.

Considerăm următorul cod:

A	.+
B	_..
C	.+_
D	_.+
E	.+.
F	&

Acesta poate fi reprezentat, cu ajutorul unui arbore, astfel:



De asemenea, considerăm că în textul codificat, codurile caracterelor textului original vor fi separate prin caracterul '^'. Astfel, codificarea cuvântului "cade" ar fi ".+_^.+^_.+^.+.".

2. Cerință

Să se decodifice un text, fiind furnizată modalitatea de codificare.

Se va furniza codificarea folosită, specificându-se codul asociat fiecărui caracter, precum și un caracter special folosit ca și separator, între două coduri succesive.

Se garantează furnizarea codificării pentru toate caracterele necesare, inclusiv pentru semnele de punctuație. Nu vor fi codificate caracterele whitespace.

Codul furnizat va fi stocat cu ajutorul unui arbore, decodificarea unui anumit caracter fiind efectuată prin parcurgerea arborelui până la nodul asociat caracterului respectiv.

Codificarea va fi furnizată în doi pași. Inițial se precizează un set de caractere și codificările asociate lor. Acestea vor genera arborele inițial. Ulterior, se va specifica și un set de operații ce trebuiesc efectuate asupra arborelui înainte de decodificarea textului. Operațiile sunt de două tipuri: 'add' și 'delete'.

Operația 'delete' va șterge din arbore codificarea asociată unui caracter. În urma acestei operații se vor șterge toate nodurile necesare astfel încât în arbore să nu existe căi care să conducă la o codificare invalidă (neasociată unui caracter).

Operația 'add' va adăuga în arbore codificarea asociată unui caracter. Dacă în arbore există o codificare asociată acestui caracter, diferită de noua codificare, se va păstra doar noua codificare, cea anterioară fiind ștearsă. În cazul în care noua codificare coincide cu cea deja existentă, arborele nu va fi modificat.

3. Implementare

3.1. Rulare

Programul va fi rulat astfel:

./tema3 cod.in text.in arbore.out text.out

Unde:

cod.in	- numele fișierului în care este specificat codul
text.in	- numele fișierului ce conține textul codificat
arbore.out	- numele fișierului ce conține informații despre arbore
text.out	- numele fișierului ce va conține textul în clar

3.2. Formatul datelor de intrare / ieșire

3.2.1 Formatul fișierului ce conține codul

Fișierul va fi organizat în 2 secțiuni. În prima parte se specifică codificările a N caractere și un caracter suplimentar, care va fi utilizat, în textul codificat, pentru a marca sfârșitul codului

unui caracter (se garantează că acest caracter nu face parte din mulțimea caracterelor utilizate pentru codificare). Aceste N codificări vor forma arborele inițial.

A doua parte va specifica un set de M operații ce vor fi efectuate asupra arborelui. Operațiile sunt de două tipuri: ‘add’ și ‘delete’.

Notăm:

N	– numărul caracterelor considerate
ch_i	- caracterul i
$code_i$	- codul asociat caracterului i
s	- caracterul folosit ca separator
M	- numărul de operații
op_i	- tipul operației i (add / delete)

Linia $i + 1$ a fișierului ($i \leq N$) este formată din caracterul ch_i, un spațiu și codul $code_i$ (codurile nu vor conține spații sau caracterul ‘\n’).

Linia $j + N + 3$ va conține tipul operației j (op_j), caracterul vizat (ch_j) și în cazul unei operații ‘add’, se specifică și codul asociat caracterului ($code_j$).

N
ch_1 $code_1$
ch_2 $code_2$
…
ch_N $code_N$
s
M
op_1 ch_1 [$code_1$]
op_2 ch_2 [$code_2$]
…
op_M ch_M [$code_M$]

3.2.2 Formatul fișierului ce conține textul codificat

Fișierul conține textul codificat. Mulțimea caracterelor care apar în acest fișier este formată din caracterul s (separatorul), mulțimea caracterelor utilizate pentru codificare și caractere whitespace.

3.2.3 Formatul fișierului ce va conține informații despre arbore

Fișierul va conține M+1 reprezentări ale arborelui, în forma inițială (după inserarea primelor N caractere) și după fiecare din cele M operații.

Pentru afișare, arbore va fi parcurs în lățime specificând pentru fiecare nod caracterul asociat (dacă este cazul) și ultimul element din codificarea asociată nodului. Formatul de afișare este următorul: ‘$(elem: ch)$’, unde elem este ultimul element din codificare și ch caracterul asociat. Dacă elem nu există (nod rădăcină) nu se va afișa nimic în locul său. Similar, dacă ch nu există (nodul nu are asociat un caracter).

Fiecare nivel al arborelui va fi afișat pe o nouă linie. Nodurile unui nivel vor fi afișate unul după altul, fără caractere separatoare.

3.2.4 Formatul fișierului ce va conține textul în clar

Fișierul va conține textul în clar obținut în urma decodificării. Caracterul s (separatorul), nu se va regăsi în acest fișier.

3.3. Exemplu

cod.in

```
9
e .+
m _..
a .+_
T _.+
t .+.
i &
. _.+&
r _..&
q
add u &.
delete a
add a .+__
```

text.in

```
_.+q.+q_..q.+__q q.+.q_..&q.+q&q_.+&q
```

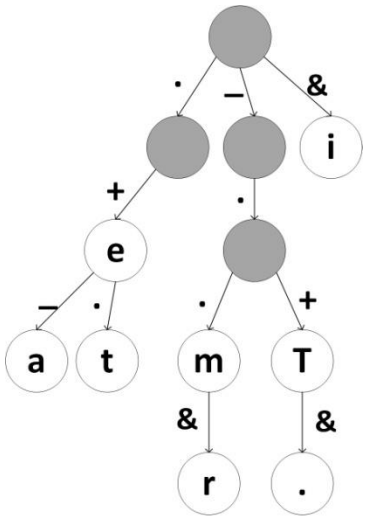
Pentru claritate, în tabelul următor prezentăm conținutul fiecărei linii din fișierul ‘cod.in’ (pentru fiecare linie, se marchează inclusiv caracterul ‘\n’). Fiecare celulă conține un caracter, celulele goale însemnând caracterul spațiu. Se observă că inclusiv caracterul punct are asociată o codificare.

9	\n									
e		.	+	\n						
m		_	.	.	\n					
a		.	+	_	\n					
T		_	.	+	\n					
t		.	+	.	\n					
i		&	\n							
.		_	.	+	&	\n				
r		_	.	.	&	\n				
q	\n									
a	d	d		u		&	.	\n		
d	e	l	e	t	e		a	\n		
a	d	d		a		.	+	_	_	\n

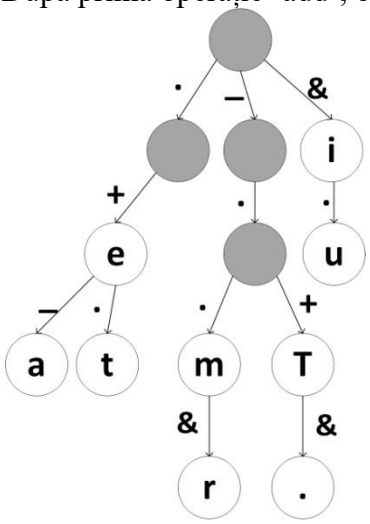
Similar, reprezentăm și textul codificat.

_	.	+	q	.	+	q	_	.	.	q	.	+	_	_	q		q	.	+	.	q	_	.	.	&	q	.	+	q	&	q	_	.	+	&	q

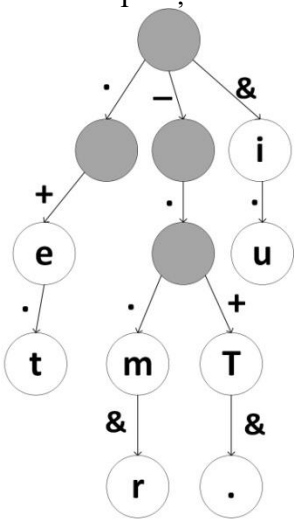
În continuare prezentăm arborele asociat codificării inițiale (după inserția codificărilor celor 9 caractere). Nodurile ce nu au atașat un caracter au fost marcate cu gri.



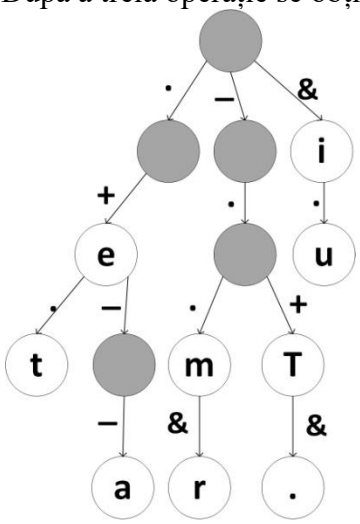
După prima operație ‘add’, obținem următorul arbore:



În urma operației ‘delete’, obținem următorul arbore:



După a treia operație se obține următorul arbore:



În urma decodificării se obține textul: “Tema trei.”.

arbore.out

```
(:)
(.:)(_:)(&:i)
(+:e)(.:)
(_:a)(.:t)(.:m)(+:T)
(&:r)(&:.)
(:)
(.:)(_:)(&:i)
(+:e)(.:)(.:u)
(_:a)(.:t)(.:m)(+:T)
(&:r)(&:.)
(:)
(.:)(_:)(&:i)
(+:e)(.:)(.:u)
(.:t)(.:m)(+:T)
(&:r)(&:.)
(:)
(.:)(_:)(&:i)
(+:e)(.:)(.:u)
(.:t)(_:)(.:m)(+:T)
(_:a)(&:r)(&:.)
```

text.out

```
Tema trei.
```

3.4. Detalii de implementare

Inserarea codificărilor în arbore se va face conform ordinii din fișierul ‘cod.in’. Copiii unui nod se vor completa întotdeauna de la stânga la dreapta.

4. Notare

- **85 de puncte** obținute conform testelor de pe vmchecker
- **10 puncte**: coding style, codul trebuie să fie comentat, consistent și ușor de citit. De exemplu, tema nu trebuie să conțină:
  - Warning-uri la compilare
  - Linii mai lungi de 80 de caractere
  - Denumire neadecvată a funcțiilor sau a variabilelor
  - Inconsistențe în indentare
- **5 puncte** README + comentarii
  Este necesar ca fișierul README să conțină o descriere a structurilor de date utilizate, precizând și modul în care ați construit arborele. De asemenea, se va explica implementarea algoritmul de decodificare.
- Bonus **20 de puncte** pentru soluțiile ce nu au memory leak-uri (bonusul se va calcula proporțional cu numărul de teste trecute)
- **O temă care nu va compila se va nota automat cu 0**
- **Se vor depuncta soluțiile ce alocă mai multă memorie decât este necesar.**
  De exemplu, se vor depuncta soluțiile care alocă și alte zone de memorie pentru stocarea codificărilor, în afară de arborele construit.

5. Reguli de trimitere a temelor

A se vedea 'Regulament SD' [1].

Temele vor fi încărcate atât pe vmchecker (în secțiunea Structuri de Date (CB): SD-CB), cât și pe cs.curs.pub.ro, în secțiunea aferentă temei 3.

Arhiva cu rezolvarea temei trebuie să fie .zip și să conțină:
- Fișiere sursă; fiecare fișier va trebui să înceapă cu un comentariu de forma:
  /* NUME Prenume – grupa */
- Fișier README, care să conțină detalii despre implementarea temei
- Fișier Makefile (numele fișierului este obligatoriu ''Makefile'), care să conțină următoarele reguli:
  - build - cara va compila sursele și va obține executabilul cu numele 'tema3'
  - clean – care va șterge executabilul și eventualele fișiere obiect generate

Arhiva nu trebuie să conțină fișiere executabile sau obiect.

Dacă arhiva nu respectă specificațiile de mai sus nu va fi acceptată la upload și tema nu va fi luată în considerare.

Referințe

[1] http://cs.curs.pub.ro/2015/mod/page/view.php?id=4322