Univerzitet u Sarajevu Elektrotehnički fakultet

Predmet: Multimedijalni Sistemi

Laboratorijska vježba 04: Lempel-Ziv-Welch (LZW) algoritam kompresije

Za izradu laboratorijske vježbe treba koristiti odgovarajuću Jupyter Notebook datoteku. Urađenu vježbu je potrebno konvertirati u PDF format, a zatim je PDF datoteku potrebno predati do postavljenog roka koristeći platformu Zamger.

Ime i prezime studenta, broj indeksa:

```
Amar Hasečić, 2116/18673
```

Datum izrade izvještaja:

```
02.04.2024
```

Zadatak 1.

Potrebno je realizirati funkciju lzw koja implementira **LZW algoritam kompresije**. Funkcija kao parametar prima neku tekstualnu poruku, a kao rezultat vraća listu simbola, listu svih kodova dodijeljenih pojedinačnim simbolima i kodiranu poruku. Kodirana (komprimirana) poruka treba biti u obliku liste cijelih brojeva.

LZW algoritam kodiranja možete implementirati na temelju sljedećeg pseudokoda:

Ulaz: String za enkodiranje, rjecnik
$novi\_string \leftarrow$ ""
while $ch \leftarrow s[i]$!= '#' **do**
    if $novi\_string + ch$ **in** rjecnik **then**
        $novi\_string = novi\_string + ch$
    else
        enkodirati $novi\_string$ u izlaz
        dodati $novi\_string + ch$ u rjecnik
        $novi\_string \leftarrow ch$
    end if
end while
ispisati $novi\_string$ u izlaz

Opis algoritma možete pronaći i u Poglavlju 2 u PDF materijalima na str. 33-36.

Za početni rječnik možete uzeti fiksni rječnik koji se sastoji od sljedećih riječi dužine 1: A, B i C. Znak koji označava kraj ulaznog niza simbola je #.

Rješenje:

```
def pronadjiVrijednost(lst, kljuc):
    for item in lst:
        if item[0] == kljuc:
            return item[1]
    return None


def lzw(poruka):

    kod = []
    rjecnik = [("A",1), ("B",2), ("C",3)]

    s = poruka[0]
    if s == '#':
        return

    indeks = 1
    while poruka[indeks]!='#':
        c = poruka[indeks]

        if pronadjiVrijednost(rjecnik,s+c):
            s = s+c

        else:

            kod.append(pronadjiVrijednost(rjecnik,s))
            rjecnik.append((s+c, len(rjecnik)+1))
            s = c

        indeks+=1

    kod.append(pronadjiVrijednost(rjecnik,s))

    simboli = [item[0] for item in rjecnik]
    kodovi = [item[1] for item in rjecnik]

    return(simboli, kodovi, kod)
```

Nakon implementacije funkcije, potrebno je biti moguće izvršiti programski kod ispod tako da se dobije prikazani ispis. Osim toga, potrebno je pored poruke "BABAABBACABABB#" dodati još pet primjera poruka kako bi se implementacija testirala. Dodatnih pet primjera poruka treba obavezno biti iz sljedećeg skupa {"A#", "ABBCBAABCCACBBABBCBBC#", "ABBABBAABBABBAA#", "BBBCABBCABBCAB#", "ABBCAABAAAAAAACCCBCC#"}.

```
poruka = "BABAABBACABABB#"

print("Izvorna poruka je:")
print(poruka)

(simboli, kodovi, kod) = lzw(poruka)

print("\nRječnik nakon kompresije sadrži sljedeće riječi:")
print(", ".join(map(str, simboli)))

print("\nKodovi za gore navedene riječi u respektivnom poretku su:")
print(", ".join(map(str, kodovi)))

print("\nKomprimirana poruka je:")
print(kod)
```

```
Izvorna poruka je:
BABAABBACABABB#

Rječnik nakon kompresije sadrži sljedeće riječi:
A, B, C, BA, AB, BAA, ABB, BAC, CA, ABA

Kodovi za gore navedene riječi u respektivnom poretku su:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Komprimirana poruka je:
[2, 1, 4, 5, 4, 3, 5, 7]
```

```
#Test 1

poruka = "A#"

print("Izvorna poruka je:")
print(poruka)

(simboli, kodovi, kod) = lzw(poruka)

print("\nRječnik nakon kompresije sadrži sljedeće riječi:")
print(", ".join(map(str, simboli)))

print("\nKodovi za gore navedene riječi u respektivnom poretku su:")
print(", ".join(map(str, kodovi)))

print("\nKomprimirana poruka je:")
print(kod)
```

```
Izvorna poruka je:
A#

Rječnik nakon kompresije sadrži sljedeće riječi:
A, B, C
```

```
Kodovi za gore navedene riječi u respektivnom poretku su:
1, 2, 3

Komprimirana poruka je:
[1]
```

```
#Test 2

poruka = "ABBCBAABCCACBBABBCBBC#"

print("Izvorna poruka je:")
print(poruka)

(simboli, kodovi, kod) = lzw(poruka)

print("\nRječnik nakon kompresije sadrži sljedeće riječi:")
print(", ".join(map(str, simboli)))

print("\nKodovi za gore navedene riječi u respektivnom poretku su:")
print(", ".join(map(str, kodovi)))

print("\nKomprimirana poruka je:")
print(kod)
```

```
Izvorna poruka je:
ABBCBAABCCACBBABBCBBC#

Rječnik nakon kompresije sadrži sljedeće riječi:
A, B, C, AB, BB, BC, CB, BA, AA, ABC, CC, CA, AC, CBB, BAB, BBC, CBBC

Kodovi za gore navedene riječi u respektivnom poretku su:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17

Komprimirana poruka je:
[1, 2, 2, 3, 2, 1, 4, 3, 3, 1, 7, 8, 5, 14, 3]
```

```
#Test 3

poruka = "ABBABBAABBABBAA#"

print("Izvorna poruka je:")
print(poruka)

(simboli, kodovi, kod) = lzw(poruka)

print("\nRječnik nakon kompresije sadrži sljedeće riječi:")
print(", ".join(map(str, simboli)))

print("\nKodovi za gore navedene riječi u respektivnom poretku su:")
print(", ".join(map(str, kodovi)))
```

```
print("\nKomprimirana poruka je:")
print(kod)
```

```
Izvorna poruka je:
ABBABBAABBABBAA#

Rječnik nakon kompresije sadrži sljedeće riječi:
A, B, C, AB, BB, BA, ABB, BAA, ABBA, ABBAA

Kodovi za gore navedene riječi u respektivnom poretku su:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Komprimirana poruka je:
[1, 2, 2, 4, 6, 7, 9, 1]
```

```
#Test 4

poruka = "BBBCABBCABBCAB#"

print("Izvorna poruka je:")
print(poruka)

(simboli, kodovi, kod) = lzw(poruka)

print("\nRječnik nakon kompresije sadrži sljedeće riječi:")
print(", ".join(map(str, simboli)))

print("\nKodovi za gore navedene riječi u respektivnom poretku su:")
print(", ".join(map(str, kodovi)))

print("\nKomprimirana poruka je:")
print(kod)
```

```
Izvorna poruka je:
BBBCABBCABBCAB#

Rječnik nakon kompresije sadrži sljedeće riječi:
A, B, C, BB, BBC, CA, AB, BBCA, ABB, BC, CAB

Kodovi za gore navedene riječi u respektivnom poretku su:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

Komprimirana poruka je:
[2, 4, 3, 1, 5, 7, 2, 6, 2]
```

```
#Test 5

poruka = "ABBCAABAAAAAAACCCBCC#"

print("Izvorna poruka je:")
print(poruka)
```

```
(simboli, kodovi, kod) = lzw(poruka)

print("\nRječnik nakon kompresije sadrži sljedeće riječi:")
print(", ".join(map(str, simboli)))

print("\nKodovi za gore navedene riječi u respektivnom poretku su:")
print(", ".join(map(str, kodovi)))

print("\nKomprimirana poruka je:")
print(kod)
```

```
Izvorna poruka je:
ABBCAABAAAAAAACCCBCC#

Rječnik nakon kompresije sadrži sljedeće riječi:
A, B, C, AB, BB, BC, CA, AA, ABA, AAA, AAAA, AAC, CC, CCB, BCC

Kodovi za gore navedene riječi u respektivnom poretku su:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

Komprimirana poruka je:
[1, 2, 2, 3, 1, 4, 8, 10, 8, 3, 13, 6, 3]
```

Zadatak 2.

Potrebno je realizirati **dvije varijante** algoritma za **LZW dekompresiju**. Prvu **pojednostavljenu varijantu** treba realizirati implementacijom funkcije reverse_lzw, dok drugu **modificiranu varijantu** treba realizirati implementacijom funkcije m_reverse_lzw. Obje varijante funkcije kao parametar primaju ulazni kod u obliku liste cijelih brojeva, a kao rezultat vraćaju listu simbola, listu svih kodova dodijeljenih pojedinačnim simbolima i dekodiranu tekstualnu poruku.

Pojednostavljenu varijantu LZW algoritma dekompresije možete implementirati prema sljedećem pseudokodu:

Ulaz: String za dekodiranje, rjecnik
1: *novi_string* ← ""
2: pročitati *prethodni_kod* i dekodirati ga
3: **while** *ch* ← *s*[*i*] != '#' **do**
4: **if** *ch* **in** rjecnik **then**
5: *kljuc* ← *rjecnik*[*ch*]
6: dodati *kljuc* u rjecnik
7: *novi_string* ← *kljuc*
8: **end if**
9: **end while**
10: ispisati *novi_string* u izlaz

Opis algoritma LZW dekompresije možete pronaći i u Poglavlju 2 u PDF materijalima na str. 36-38. Prije implementacije modificirane varijante prvo proučite iz PDF materijala (Poglavlje 2, str. 38-41) u kojim slučajevima ta varijanta ne omogućava dekompresiju, a zatim implementirajte modificiranu varijantu tako što ćete već realiziranu pojednostavljenu varijantu modificirati na način kako je to opisano PDF materijalima u Poglavlju 2 na strani 41.

Za početni rječnik možete uzeti fiksni rječnik koji se sastoji od sljedećih riječi dužine 1: A, B i C. Znak koji označava kraj ulaznog niza simbola je #.

Rješenje:

```
def pronadjiKljuc(lst, vrijednost):
    for item in lst:
        if item[1] == vrijednost:
            return item[0]
    return ""

def reverse_lzw(kod):

    poruka =""
    rjecnik = [("A",1), ("B",2), ("C",3)]
    s=""

    for i in range(len(kod)):
        k = kod[i]
        p = pronadjiKljuc(rjecnik,k)
        poruka += p

        if s!="":
            rjecnik.append((s+p[0],len(rjecnik)+1))
        s=p

    simboli = [item[0] for item in rjecnik]
    kodovi = [item[1] for item in rjecnik]

    return(simboli, kodovi, poruka)

def m_reverse_lzw(kod):

    poruka =""
    rjecnik = [("A",1), ("B",2), ("C",3)]
    s=""

    for i in range(len(kod)):
        k = kod[i]
        p = pronadjiKljuc(rjecnik,k)

        #modifikacija
        if p=="":
            p=s+s[0]
```

```
        poruka += p

        if s!="":
            rjecnik.append((s+p[0],len(rjecnik)+1))
        s=p

    simboli = [item[0] for item in rjecnik]
    kodovi = [item[1] for item in rjecnik]

    return(simboli, kodovi, poruka)
```

Nakon implementacije funkcija, potrebno je biti moguće izvršiti programski kod ispod tako da se dobije prikazani ispis. Osim toga, potrebno je pored koda1 kojeg čini niz [2,1,4,5,4,3,5,7] dodati još dva primjera za ulazni kod1 kako bi se implementacija testirala. Dodatna dva primjera za kod1 trebaju obavezno biti iz sljedećeg skupa {[1],[1,2,2,3,2,1,4,3,3,1,7,8,5,14,3]}. Nadalje, potrebno je pored koda2 kojeg čini niz [1,2,2,4,6,7,9,1] dodati još dva primjera za kod2. Dodatna dva primjera za kod2 trebaju obavezno biti iz sljedećeg skupa {[2,4,3,1,5,7,2,6,2], [1,2,2,3,1,4,8,10,8,3,13,6,3]}.

```
kod1 = [2,1,4,5,4,3,5,7]
kod2 = [1,2,2,4,6,7,9,1]

print("Ulazni kod1 je:")
print(kod1)

(simboli, kodovi, poruka1) = reverse_lzw(kod1)

print("\nRječnik nakon dekompresije je:")
print(", ".join(map(str, simboli)))

print("\nKodovi za gore navedene riječi u respektivnom poretku su:")
print(", ".join(map(str, kodovi)))

print("\nDekomprimirana poruka1 je:")
print(poruka1)

print("\n\nUlazni kod2 je:")
print(kod2)

(simboli, kodovi, poruka2) = m_reverse_lzw(kod2)

print("\nRječnik nakon dekompresije je:")
print(", ".join(map(str, simboli)))

print("\n\nKodovi za gore navedene riječi u respektivnom poretku su:")
print(", ".join(map(str, kodovi)))

print("\nDekomprimirana poruka2 je:")
print(poruka2)
```

```
Ulazni kod1 je:
[2, 1, 4, 5, 4, 3, 5, 7]

Rječnik nakon dekompresije je:
A, B, C, BA, AB, BAA, ABB, BAC, CA, ABA

Kodovi za gore navedene riječi u respektivnom poretku su:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Dekomprimirana poruka1 je:
BABAABBACABABB


Ulazni kod2 je:
[1, 2, 2, 4, 6, 7, 9, 1]

Rječnik nakon dekompresije je:
A, B, C, AB, BB, BA, ABB, BAA, ABBA, ABBAA


Kodovi za gore navedene riječi u respektivnom poretku su:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Dekomprimirana poruka2 je:
ABBABBAABBABBAA
```

```
#Test 1
kod1 = [1]
kod2 = [2,4,3,1,5,7,2,6,2]

print("Ulazni kod1 je:")
print(kod1)

(simboli, kodovi, poruka1) = reverse_lzw(kod1)

print("\nRječnik nakon dekompresije je:")
print(", ".join(map(str, simboli)))

print("\nKodovi za gore navedene riječi u respektivnom poretku su:")
print(", ".join(map(str, kodovi)))

print("\nDekomprimirana poruka1 je:")
print(poruka1)

print("\n\nUlazni kod2 je:")
print(kod2)

(simboli, kodovi, poruka2) = m_reverse_lzw(kod2)

print("\nRječnik nakon dekompresije je:")
print(", ".join(map(str, simboli)))
```

```
print("\n\nKodovi za gore navedene riječi u respektivnom poretku su:")
print(", ".join(map(str, kodovi)))

print("\nDekomprimirana poruka2 je:")
print(poruka2)
```

```
Ulazni kod1 je:
[1]

Rječnik nakon dekompresije je:
A, B, C

Kodovi za gore navedene riječi u respektivnom poretku su:
1, 2, 3

Dekomprimirana poruka1 je:
A


Ulazni kod2 je:
[2, 4, 3, 1, 5, 7, 2, 6, 2]

Rječnik nakon dekompresije je:
A, B, C, BB, BBC, CA, AB, BBCA, ABB, BC, CAB


Kodovi za gore navedene riječi u respektivnom poretku su:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

Dekomprimirana poruka2 je:
BBBCABBCABBCAB
```

```
#Test 2
kod1 = [1,2,2,3,2,1,4,3,3,1,7,8,5,14,3]
kod2 = [1,2,2,3,1,4,8,10,8,3,13,6,3]

print("Ulazni kod1 je:")
print(kod1)

(simboli, kodovi, poruka1) = reverse_lzw(kod1)

print("\nRječnik nakon dekompresije je:")
print(", ".join(map(str, simboli)))

print("\nKodovi za gore navedene riječi u respektivnom poretku su:")
print(", ".join(map(str, kodovi)))

print("\nDekomprimirana poruka1 je:")
print(poruka1)
```

```
print("\n\nUlazni kod2 je:")
print(kod2)

(simboli, kodovi, poruka2) = m_reverse_lzw(kod2)

print("\nRječnik nakon dekompresije je:")
print(", ".join(map(str, simboli)))

print("\n\nKodovi za gore navedene riječi u respektivnom poretku su:")
print(", ".join(map(str, kodovi)))

print("\nDekomprimirana poruka2 je:")
print(poruka2)
```

```
Ulazni kod1 je:
[1, 2, 2, 3, 2, 1, 4, 3, 3, 1, 7, 8, 5, 14, 3]

Rječnik nakon dekompresije je:
A, B, C, AB, BB, BC, CB, BA, AA, ABC, CC, CA, AC, CBB, BAB, BBC, CBBC

Kodovi za gore navedene riječi u respektivnom poretku su:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17

Dekomprimirana poruka1 je:
ABBCBAABCCACBBABBCBBC


Ulazni kod2 je:
[1, 2, 2, 3, 1, 4, 8, 10, 8, 3, 13, 6, 3]

Rječnik nakon dekompresije je:
A, B, C, AB, BB, BC, CA, AA, ABA, AAA, AAAA, AAC, CC, CCB, BCC


Kodovi za gore navedene riječi u respektivnom poretku su:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

Dekomprimirana poruka2 je:
ABBCAABAAAAAAACCCBCC
```