UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI FAKULTET

Katedra za elektroniku

Predmet: Digitalna obrada slike

Izveštaj: 5. domaći zadatak

Rok za predaju: 26.12.2015.

Student:

Ime	Prezime	broj indeksa
Predrag	Kuzmanović	49/2012

1. Kodovanje i dekodovanje pomoću Hafmanovog kodnog postupka

U cilju što bržeg izvršavanja algoritama za kodovanje i dekodovanje napisane su 2 MEX funkcije koje pozivaju odgovarajući C (preciznije: C++) kod: **huffman.m** i **neglect.m**. Odgovarajući C kod se nalazi u istoimenim **cpp** fajlovima. Ideja je bila da se C kod koristi na mestima gde će ušteda biti značajna, a to se pre svega odnosi na izbacivanje rekurzije i izbacivanje potrebe za sortiranjem.

1.1. huffman.m

Ova funkcije generise Hafmanovu kodnu tabelu na osnovu ulaznog histograma slike. Takođe, ona i dekoduje ulazni binarni string na osnovu Hafmanove kodne tabele. Dakle, ova funkcija se koristi i u procesu kodovanja i u procesu dekodovanja, ali sa različitim ulaznim parametrima.

Algoritam kodovanja: Radi brzog izvršenja kodovanja implementirana je struktura podataka pod nazivom **MinHeap**. Ova struktura podataka podržava ubacivanje novog elementa i izbacivanje trenutnog minimalnog elementa u logaritamskom vremenu. Ova struktura podataka je vrlo korisna pri sažimanju 2 simbola pri Hafmanovom kodovanju u novi simbol jer se izbegava rekurzija. Srž algoritma kodovanja čini funcija **encode**. U njoj se najpre gore opisanim postupkom u potrebnom broju iteracija napravi struktura binarnog stabla iz koje se mogu iščitati kodovi odgovarajućih simbola u funkciji **traverse**. Ova funkcija rekurzivno obilazi formirano stablo, tako sto svim levim granama dodeli '0', a svim desnim granama dodeli '1'. Kodovi koji odgovaraju listovima u stablu (“pravi” simboli) odgovaraju putanjama od korena stabla do tih listova.

Algoritam dekodovanja: Najpre, pošto procesi za kodovanje i dekodovanje moraju biti nezavisni, za dekodovanje se ne može koristiti već formirano stablo prilikom kodovanja. Zato algoritam dekodovanja počinje tako što se rekonstruiše Hafmanovo binarno stablo na osnovu ulaznih argumenata (Hafmanova kodna tabela, ustvari). Poševši od praznog stabla, za kod svakog simbola pravimo nove čvorove prateći odgovarajuću putanju, po potrebi. Nakon rekonstrukcije, sledi funkcija **decode** koja je srž ovog algoritma. Ona na osnovu ulaznog binarnog stringa obilazi stablo od korena granu po granu (karakter po karakter), sve dok ne naiđe na list u stablu. Tada se dekoduje odgovarajući simbol i dekodovanje sledećeg počinje, ponovo polazeći od korena stabla.

Sintaksa i format ulaznih i izlaznih podataka detaljno su opisani u kodu. Ovde samo navodimo primer upotrebe ove funkcije za kodovanje:

```
>> p = [0.4 0.3 0.15 0.1 0.05];
>> huffman('encode', p)

ans =

0
10
110
1111
1110
```

1.2. neglect.m

Ova funkcija anulira željeni broj koeficijenata sa najmanjim amplitudama u ulaznoj matrici sa **8x8** = **64** elemenata. Radi brzog traženja **q** elemenata najmanjih po amlitudi, implementirana je struktura podataka pod nazivom **MaxHeap**. Ova struktura podataka podržava ubacivanje novog elementa i izbacivanje trenutnog maksimalnog elementa u logaritamskom vremenu. Srž algoritma anuliranja je u funkciji **KillQSmallest**. Algoritam se sastoji u sledećem: Heap velicine **q** koristimo za čuvanje trenutnih kandidata za anuliranje. Pri obilasku elemenata matrice, prvih **q** elemenata koje obiđemo jednostavno stavimo u heap jer su trenutno svi oni kandidati da budu anulirani. Nakon toga, za svaki sledeći element koji obilazimo, proveravamo da li je po amplitudi manji od trenutno maksimalnog elementa u heap-u. Ako to jeste slučaj, tada se taj element ubacuje u heap, a dosadašnji maksimum se izbacuje iz heap-a. Na kraju obilaska svih elemenata u heap-u nam se nalaze elementi koje treba anulirati. Primenom ovog algoritma gubi se stalna potreba za sortiranjem nizova. Ušteda zbog izbacivanja sortiranja se dobija u slučaju više poziva ove funkcije, a to je slušaj kod kompresije slika upotrebom funkcije **im2dos**.

Napomena: Obilazak elemenata u ulaznoj matrici se radi "od nazad" zbog prirode elemenata ulazne matrice pri primeni ove funkcije, a to su DCT koeficijenti. Naime, za prirodne slike je dominatan DC koeficijent, a kako se udaljavamo od njega, koeficijenti po pravilu postaju manji po amplitudi. Dakle, obilaskom po mogućstvu prvo najmanjih elemenata ostvarujemo uštedu u broju operacija ubacivanja i izbacivanja iz heap-a.

Sintaksa i format ulaznih i izlaznih podataka detaljno su opisani u kodu. Ovde samo navodimo primer upotrebe ove funkcije:

```
>> a = magic(8)

a =

    64     2     3    61    60     6     7    57
     9    55    54    12    13    51    50    16
    17    47    46    20    21    43    42    24
    40    26    27    37    36    30    31    33
    32    34    35    29    28    38    39    25
    41    23    22    44    45    19    18    48
    49    15    14    52    53    11    10    56
     8    58    59     5     4    62    63     1

>> reshape(neglect(a, 3), size(a))

ans =

    64     0     0    61    60     6     7    57
     9    55    54    12    13    51    50    16
    17    47    46    20    21    43    42    24
    40    26    27    37    36    30    31    33
    32    34    35    29    28    38    39    25
    41    23    22    44    45    19    18    48
    49    15    14    52    53    11    10    56
     8    58    59     5     4    62    63     0
```

2. Ostale funkcije

Funkcije **generate_huffman_code.m**, **encode_huffman.m**, **decode_huffman.m**, **im2dos.m** i **dos2im.m** čija impementacija je tražena poštuju zadate interfejse i detaljno su iskomentarisane u kodu.

3. Testiranje

Implementirane funkcije su testirane u glavnom programu, **domaci5_12_049.m**. Rezultati iz komandnog prozora u MATLAB-u prikazani su ispod.

```
F == decoded_F ?
DA
K == decoded_K ?
DA
Vreme kompresije ulazne slike (q = 1): 1.2736s
Stepen kompresije ulazne slike (q = 1): 57.6594
Vreme kompresije ulazne slike (q = 7): 1.2089s
Stepen kompresije ulazne slike (q = 7): 16.0999
Vreme kompresije ulazne slike (q = 15): 1.2157s
Stepen kompresije ulazne slike (q = 15): 13.8395
Vreme kompresije ulazne slike (q = 33): 1.2605s
Stepen kompresije ulazne slike (q = 33): 13.5229
Vreme kompresije ulazne slike (q = 64): 1.2134s
Stepen kompresije ulazne slike (q = 64): 13.5197
Vreme dekompresije kompresovane slike (q = 1): 0.98849s
Vreme dekompresije kompresovane slike (q = 7): 1.0579s
Vreme dekompresije kompresovane slike (q = 15): 1.1237s
Vreme dekompresije kompresovane slike (q = 33): 1.0488s
Vreme dekompresije kompresovane slike (q = 64): 1.0989s
```

Na slici 3.1. prikazana je originalna slika na kojoj je testirana kompresija, **lena.tif**. Na slikama 3.2. do 3.6. redom su prikazane dekodovane slike nakon kompresije, za različite parametre nivoa kvaliteta.

Slika 3.1. Originalna slika, lena.tif

Slika 3.2. Dekompresovana slika, lena_q1.tif

Slika 3.3. Dekompresovana slika, lena_q7.tif

Slika 3.4. Dekompresovana slika, lena_q15.tif

Slika 3.5. Dekompresovana slika, lena_q33.tif

Slika 3.6. Dekompresovana slika, lena_q64.tif

Iz prikazanih rezultata možemo izvući sledeće zaključke. Najpre, stepen kompresije se povećava sa smanjenjem kvaliteta (broja zadržanih DCT koeficijenata u bloku), i kreće su u opsegu od **13.5159** (za najveći mogući kvalitet) do **57.6594** (za najmanji mogući kvalitet).

Takođe, sa slike 3.2. vidmo da se u slučaju zadržavanja samo jednog DCT koeficijenta u bloku dekompresovana slika sastoji od blokova u kojima su potisnute visoke učestanosti. Mada je stepen kompresije ostvaren na ovaj način najveći, kvalitet dekompresovane slike nije zadovoljavajući jer se značajan deo informacija iz originalne slike gubi.

Sa slike 3.3. vidimo da se već sa **7** zadržanih najznačajnijih DCT koeficijenata po bloku dobija dekompresovana slika viskog kvaliteta, uz i dalje visok stepen kompresije koji iznosi **16.0999**. Iako su razlike u kvalitetu originalne i dekompresovane slike uočljive, ovakav kvalitet nam može biti prihvatljiv u nekim primenama.

Već sa kvalitetom **15** i nadalje, razlike između originalne i dekompresovane slike su teško uočljive, čak i za veoma oštro oko.

U cilju dobijanja kompresije sa manjim gubicima predlaže se izbacivanje koraka deljenja (odnosno množenja) DCT koeficijenata kvantizacionom matricom u funkciji **im2dos** (odnosno **dos2im**). Na ovaj način povećava se broj nenultih DCT koeficijenata koji se koduju Hafmanovim kodnim postpkom. Samim tim, fajl kompresovane slike biće veći (znatno manji stepen kompresije), ali će gubitak kvaliteta nakom dekompresije biti manji. Rezultati iz komandnog prozora u MATLAB-u prikazani su ispod.

```
Vreme kompresije ulazne slike (q = 64): 2.0721s
Stepen kompresije ulazne slike (q = 64): 1.8163
Vreme dekompresije kompresovane slike (q = 64): 1.6289s
Minimalno odstupanje vrednosti piksela u originalnoj i
dekompresovanoj slici: 0
Maksimalno odstupanje vrednosti piksela u originalnoj i
dekompresovanoj slici: : 1
Broj piksela (u %) na kojima se javlja odstupanje:
4.0745
```

Dobijena dekompresovana slika prikazana je na slici 3.7. Kao što se vidi iz gore prikazanih rezultata, originalna i dekompresovana slika ni u ovom slučaju nisu potpuno identične. Međutim, razlike su zanemarljivo male: maksimalno odstupanje vrednosti piksela u originalnoj i dekompresovanoj slici je **1**, i to odstupanje se javlja na samo **4.0745%** piksela u slici. To znači da je **95.9255%** piksela u slici idealno rekontruisano! Ovo malo odstupanje javlja se zbog zaokruzivanja DCT koeficijenata na celobrojne vrednosti pre postupka Hafmanovog kodovanja.

Slika 3.7. Dekompresovana slika, lena_q64_no_quantization.tif