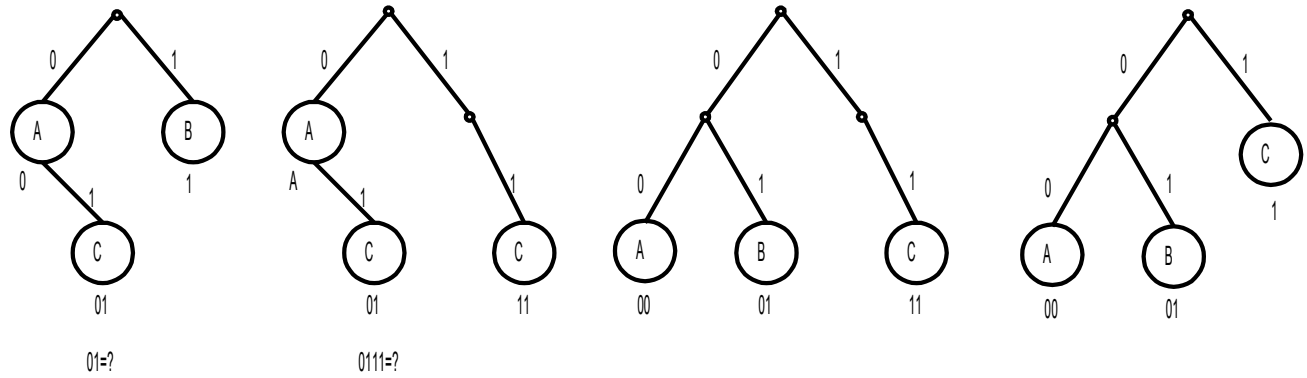
Załóżmy, że mamy źródło *S*, które generuje symbole ze zbioru
$S=\{x_1, x_2, ..., x_N\}$ z prawdopodobieństwem $P=\{p_1, p_2, ..., p_N\}$,
symbolom tym odpowiadają kody $P=\{c_1, c_2, ..., c_N\}$.
Efektywność danego sposobu kodowania można określić jako

$$L_{sr} = \sum_{i=1}^{N} p_i |c_i|$$

Def. Kod jednoznacznie definiowalny
Def. Kod przedrostkowy
Def. Kod optymalny

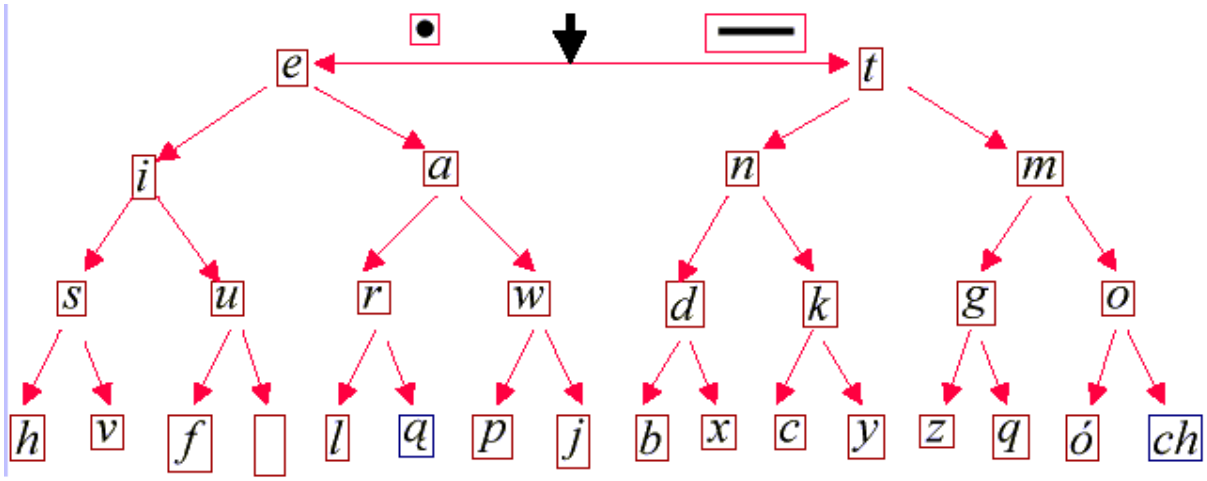
Przykłady kodów



Kody optymalne

Idea:
symbole (litery) występujące częściej powinny otrzymać kody krótsze, a symbole występujące rzadziej dłuższe.

Alfabet Morse'a (połowa XIX)



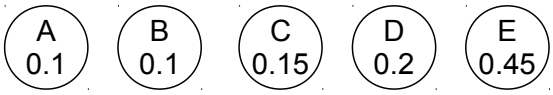
Kodowanie Huffmana

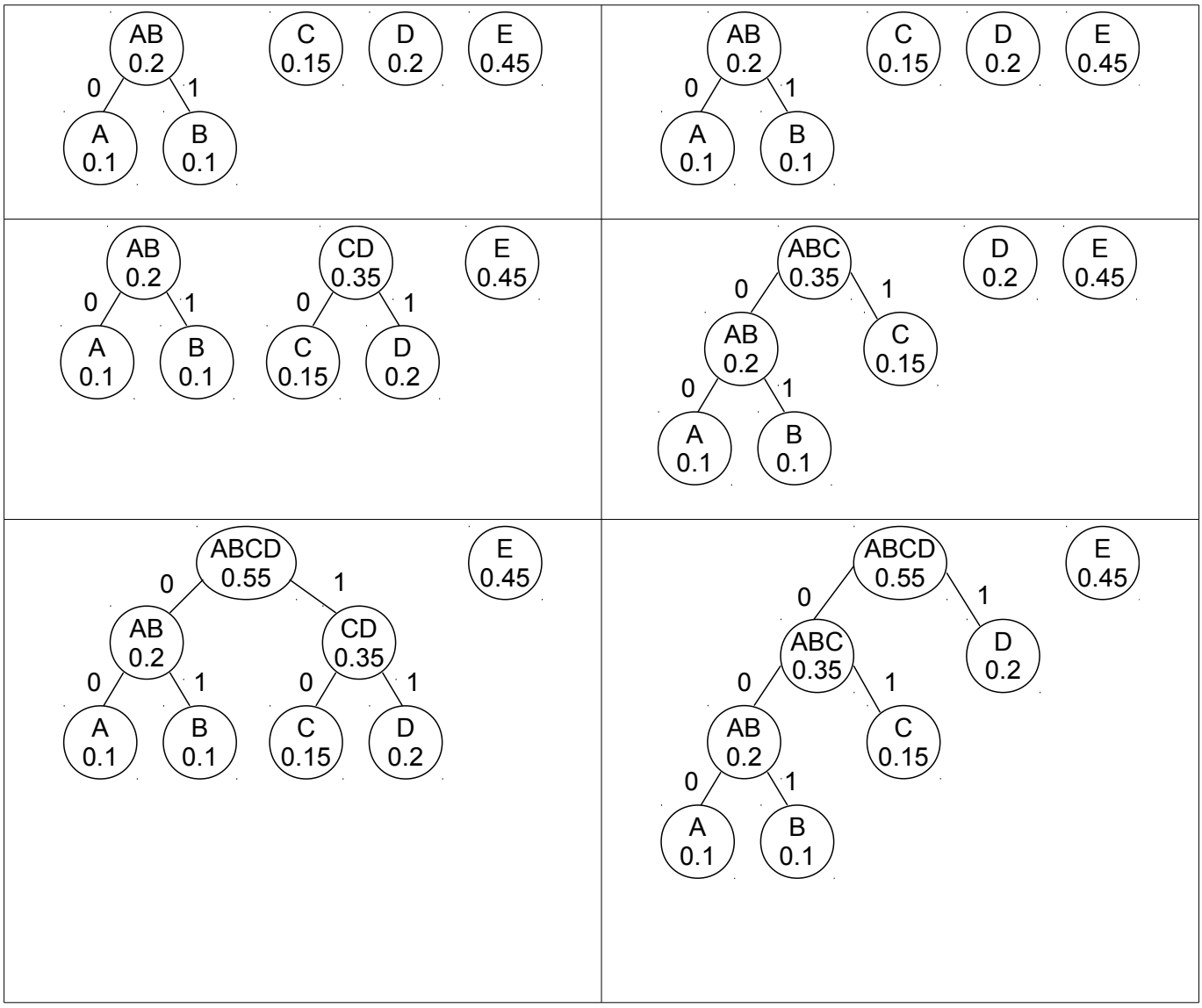
Algorytm konstrukcji drzewa Huffmana

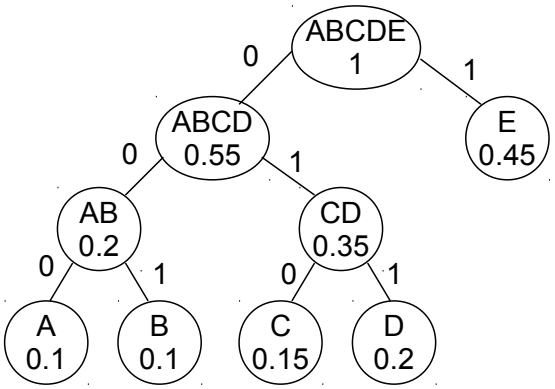
1. dla każdej litery utwórz reprezentujące ją drzewo - złożone tylko z jednego węzła (korzenia). W korzeniu tego drzewa umieść nazwę litery i prawdopodobieństwie jej wystąpienia.
2. znajdź dwa drzewa t_1 i t_2 z najmniejszą wartością wskaźnika prawdopodobieństwa.
3. połącz te drzewa w jedno drzewo – w korzeniu nowego drzewa wpisz sumę prawdopodobieństw p_1 i p_2. Lewym poddrzewem nowego drzewa jest drzewo t_1, prawym t_2. Nadaj etykietę 0 lewej gałęzi łączącej korzeń nowego drzewa z korzeniem drzewa t_1, nadaj etykietę 1 prawej gałęzi łączącej korzeń nowego drzewa z korzeniem drzewa t_2.
4. jeśli istnieją przynajmniej dwa drzewa wróć do punktu 2.
5. w tym momencie istnieje tylko jedno drzewo, a wszystkie symbole umieszczone są w jego liściach. Utwórz kod dla każdego symbolu przechodząc z korzenia do liścia i łącząc etykiety gałęzi, po których przechodzisz.

Przykład:

Dane są symbole i częstości ich wystąpień. Należy zaprojektować drzewo Huffmana i skonstruować kod.







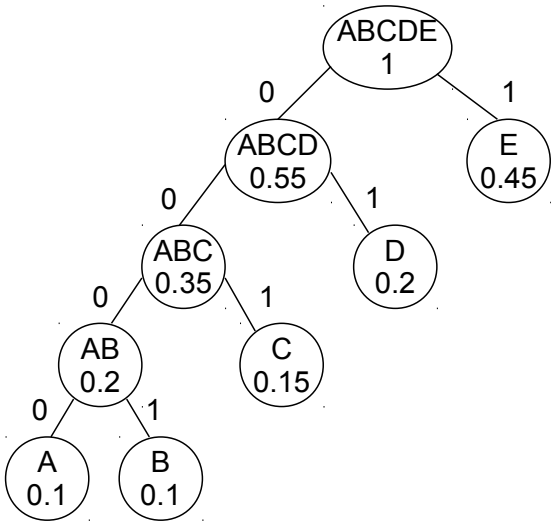
zaprojektowane słowa kodowe:

A 000
B 001
C 010
D 011
E 1

Średnia długość kodu

$L_{śr}$=

3× 0.1+3× 0.1+3× 0.15+3× 0.20+1× 0.45 =2.1



zaprojektowane słowa kodowe:

A 0000
B 0001
C 001
D 01
E 1

Średnia długość kodu

$L_{śr}$=

4× 0.1+4× 0.1+3× 0.15+2× 0.20+1× 0.45 =2.1

Rzecz ciekawa dwa różne drzewa dają taką samą średnią długość kodu !

Przykład: kodowanie i dekodowanie:

Zakodować ciąg danych *EDBCB* używając kodu z lewej części powyższej tablicy

Kod(*EDBCB*) = 1,011,001,010, 001

Przecinki postawiono tylko dla zwiększenia czytelności

Zdekodować ciąg:

1011001010001

Procedurę dekodowania rozpoczynamy od korzenia, czytamy pierwszy bit (0) i dochodzimy do liścia zawierającego symbol *E*. Czytamy drugi bit i znowu rozpoczynając od korzenia idziemy w prawo, potem trzeci bit prowadzi nas do liścia z symbolem *D*, itd.

Implementacja tablicowa

1. utwórz tablicę o 2*N*-1 pozycjach, gdzie *N* jest liczbą symboli
2. znajdź dwa symbole s_i i s_j o najmniejszej wadze i zerowej wartości pola *Link*
3. w drugiej części tablicy w pierwszej wolnej komórce utwórz nowy symbol s_{ij} i zapisz sumę wag symboli s_i i s_j.
4. w polu *Link* zapisz numer nowo utworzonego symbolu
5. w polu *P* symbolu s_i wpisz 0, a w pole *P* symbolu s_j wpisz 1.
6. jeśli są co najmniej dwa symbole z pustym polem *Link* wróć do punktu 2
7. rozpoczynając od symbolu (pozycje: 1:*N*) idziemy do korzenia (pozycja 2*N*-1) korzystając z pola *Link* i zapisując wartości z pola *P* – to daje ścieżkę od symbolu do korzenia.
8. utworzone ciągi zapisujemy w odwróconej kolejności otrzymując zapis ścieżki od korzenia do symbolu.

Przykład:
Stosując algorytm tablicowy konstrukcji drzewa Huffmana zbudować optymalny kod dla symboli występujących z częstością jak w poprzednim przykładzie.

Etap 1.

Id	**1**	**2**	3	4	5	6	7	8	9
P	**0.1**	**0.1**	0.15	0.20	0.45				
Link									

Etap 2: symbole 1 i 2, utworzono 6

Id	1	2	**3**	4	5	**6**	7	8	9
P	0	1	**0.15**	0.20	0.45	**0.2**			
Link	6	6							

Etap 2: symbole 3 i 6, utworzono 7

Id	1	2	3	**4**	5	6	7	8	9
P	0	1	0	**0.20**	0.45	1	**0.35**		
Link	6	6	7			7			

Etap 2: symbole 4 i 7, utworzono 8

Id	1	2	3	4	**5**	6	**7**	**8**	9
P	0	1	0	0	**0.45**	1	1	**0.55**	
Link	6	6	7	8		7	8		

Etap 2: symbole 5 i 8, utworzono 9 – koniec konstrukcji drzewa

Id	1	2	3	4	5	6	7	8	9
P	0	1	0	0	0	1	1	1	
Link	6	6	7	8	9	7	8	9	

Słowa kodu otrzymane w wyniku wykonania powyższego algorytmu

Symbol	Utworzony ciąg	Kod
A	0111	1110
B	1111	1111
C	011	110
D	01	10
E	0	0

Szybki algorytm dekodowania (tablicowy)

Tworzymy tablicę o rozmiarze 2^N, gdzie N jest długością najdłuższego kodu Huffmana. Z konstrukcji (przez drzewo H) wynika, że kod H jest kodem przedrostkowym, tzn. kod żadnego symbolu nie jest prefiksem kodu innego symbolu. Z tego faktu wynika możliwość szybkiego dekodowania przy użyciu odpowiednio wypełnionej tablicy:

indeks	wartość	symbol	długość
0	0000	E	1
1	0001	E	1
2	0010	E	1
3	0011	E	1
4	0100	E	1
5	0101	E	1
6	0110	E	1
7	0111	E	1
8	1000	D	2
9	1001	D	2
10	1010	D	2
11	1011	D	2
12	1100	C	3
13	1101	C	3
14	1110	A	4
15	1111	B	4

Symbol	Kod
A	1110
B	1111
C	110
D	10
E	0

Kod(*EDACB*) = 01011111101111

Algorytm budowy tablicy dekodera

Dla każdego symbolu s_i o kodzie c_i o długości l_i tworzymy $2^{(N-l_i)}$ liczb Idx(k), które traktujemy jako indeksy Idx(k) do tablicy T. Wartości tych indeksów wynoszą Idx(k)=(c_i << (N-l_i)) + k, k=0: $2^{(N-l_i)}$-1, gdzie << oznacza bitowe przesunięcie w lewo (równoważnie: Idx(k)= $2^{(N-l_i)} \times c_i + k$, k=0: $2^{(N-l_i)}$-1). W pierwszą kolumnę wiersza Idx(k) tablicy T wpisujemy nazwę symbolu (s_i), a w drugiej kolumnie długość kodu tego symbolu (l_i). W powyższym przykładzie N=4, dla symbolu „D" o kodzie 10_b otrzymujemy Idx(k) = 10_b <<2 + k = $1000_b + k$, k=0:3, Idx(0)= 1000_b, Idx(1)= 1001_b, Idx(2)= 1010_b, Idx(2)= 1011_b, oraz

T[1000_b, 1]=D, T[1000_b, 2]=2
T[1001_b, 1]=D, T[1001_b, 2]=2
T[1010_b, 1]=D, T[1010_b, 2]=2
T[1011_b, 1]=D, T[1011_b, 2]=2

Proces dekodowania polega na utrzymywaniu bufora zawierającego N bitów i traktowaniu go jako indeksu do tablicy. Po odczytaniu symbolu i jego długości przesuwamy zawartość bufora w lewo o tyle pozycji jak była odczytana długość kodu, na zwolnione miejsce wprowadzamy nowe bity ze strumienia wejściowego.

Laboratorium

1. Dla źródła S generującego symbole z prawdopodobieństwem P = [0.01, 0.02, 0.07, 0.02, 0.04, 0.14, 0.07, 0.14, 0.49]; zaprojektować kod optymalny Huffmana stosując algorytm tablicowy. Sprawdzić, czy otrzymany kod jest kodem przedrostkowym.
2. Dla ciągu współczynników lena.cof uzyskanych w wyniku transformacji DCT i kwantyzacji obrazu lena.bmp wyznaczyć optymalny kod stosując metodę Huffmana. Następnie zakodować ten ciąg w formie binarnej. Przed przystąpieniem do projektowania kodu należy wyznaczyć częstotliwości występowania poszczególnych współczynników. Proponowaną metodą jest założenie tablicy hist w której będziemy zliczać liczbę wystąpień danego symbolu. Na początku tablica jest pusta, po znalezieniu symbolu nie należącego do tablicy należy go do niej wstawić.

Przydatne funkcje Matlaba, to:
dec2bin, bin2dec, bitand, bitor, bitshift, bitxor, bitget.