Projekt 2. Kodowanie Huffmana

I. Opis problemu*

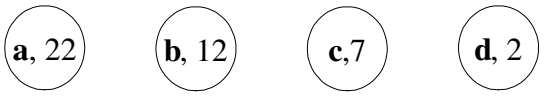
Kodowanie Huffmana polega na zastępowaniu symboli występujących w strumieniu wejściowym specjalnymi sekwencjami bitów stanowiących tzw. słowa kodowe. Słowa kodowe dobiera się w taki sposób, by najkrótsze z nich przyporządkować symbolom najczęściej występującym w strumieniu wejściowym, a najdłuższe symbolom o najniższych częstotliwościach wystąpień. Ponadto żadna sekwencja bitów nie może być przedłużeniem innej, co zapewnia jednoznaczność pomiędzy reprezentacją naturalną a zbiorem danych przedstawionym w postaci zakodowanej. Dzięki swej konstrukcji kody Huffmana znajdują szerokie zastosowanie w bezstratnej kompresji danych, w szczególności tam, gdzie występują znaczne różnice pomiędzy częstotliwościami wystąpień symboli, a słaba korelacja pomiędzy sąsiednimi elementami wyklucza użycie metod słownikowych. Stąd kody Huffmana stosuje się powszechnie w formatach kompresji obrazu i dźwięku, np. JPEG, MPEG oraz MP3.

Klasyczny algorytm wyznaczania słów kodowych bazuje na drzewie binarnym. Załóżmy, że w strumieniu wejściowym występują 8-bitowe liczby całkowite, tzn. dane typu „**unsigned char**”. Niech przykładowy ciąg symboli do zakodowania przyjmie następującą postać:

aaaabbccaaaaccbbaddaaaaaaabbbbcccaaaaaabbbb

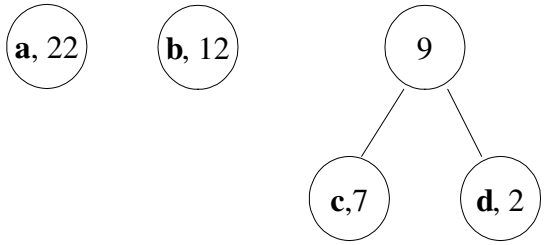
Wówczas:

1. Zliczamy częstotliwości wystąpień poszczególnych symboli tworząc tzw. las, w którym każde drzewo (tutaj posiadające tylko jeden węzeł) zawiera jeden symbol oraz wagę będącą liczbą wystąpień danego symbolu. Ponieważ w naszym przykładzie litera **a** wystąpiła 22 razy, **b** 12 razy, **c** 7 razy oraz **d** 2 razy, to wspomniany las przyjmie postać:

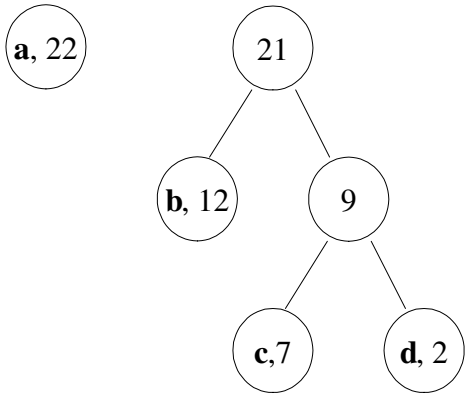


2. Następnie z lasu utworzonego w punkcie 1 wybieramy dwa drzewa o najniższych wagach i łączymy je, tworząc nowe drzewo o wadze będącej sumą wag drzew składowych. Operację tę powtarzamy aż do uzyskania jednego drzewa. Dla naszego przykładu mamy:

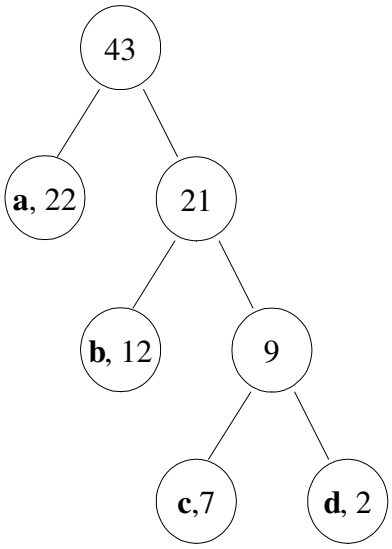
Krok 1.



Krok 2.

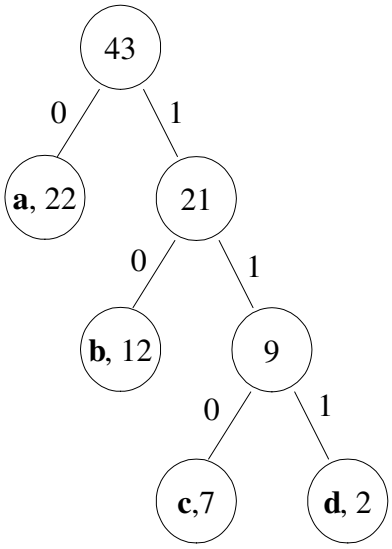


Krok 3.



W rezultacie otrzymujemy jedno drzewo, którego liście zawierają kodowane symbole, tutaj liczby typu „**unsigned char**”.

3. Następnie dla każdego symbolu wyznaczamy słowo kodowe, będące ścieżką od korzenia do liścia zawierającego dany symbol, przy czym przejściu w lewo przyporządkowujemy bit '0' a przejściu w prawo bit '1'. W naszym przykładzie po przyjęciu tej zasady drzewo kodów Huffmana przyjmie postać:



Przechodząc drzewo od korzenia do każdego liścia otrzymamy tablicę kodów Huffmana dla poszczególnych symboli występujących w przykładowym ciągu danych:

Symbol	**Kod**
a	**0**
b	**10**
c	**110**
d	**111**

* Opracowano na podstawie [1].

II. Specyfikacja zadania

Państwa zadanie sprowadza się do wyznaczenia słów kodowych wg. powyższego algorytmu dla symboli występujących w pliku, którego nazwę wprowadza użytkownik programu. Słowa kodowe należy zapisać do drugiego pliku (tzw. pliku słownika) wskazanego przez użytkownika, zgodnie z następującą zasadą:

1. Najpierw zapisujemy symbol jako liczbę typu „**unsigned char**", tzn. jako liczbę całkowitą z przedziału od 0 do 255 w postaci kodów ASCII poszczególnych cyfr.
2. Następnie wstawiamy znak ':'.
3. Dalej podajemy kod Huffmana dla danego symbolu w postaci ciągu zer i jedynek zapisanych również za pomocą kodów ASCII.
4. Na końcu takiego wpisu wstawiamy znak nowej linii, tzn. liczby 13 i 10.

   W naszym przykładzie plik wynikowy z kodami Huffmana wyglądałby w sposób następujący:

   97:0
   98:10
   99:110
   100:111

   oraz jako liczby szesnastkowe (tak nie trzeba zapisywać):

   39 37 3A 30 0D 0A
   39 38 3A 31 30 0D 0A
   39 39 3A 31 31 30 0D 0A
   31 30 30 3A 31 31 31 0D 0A

   Oczywiście kolejność wpisów w pliku nie ma znaczenia. Załączony program demonstracyjny „**gen.exe**" służy do generowania oraz zapisu słów kodowych Huffmana wg. powyższych ustaleń.

Tutaj kończy się obowiązkowy dla Państwa zakres działań.

Następnie należałoby zakodować wejściowy plik (plik_we) przy użyciu wcześniej wyznaczonych kodów (zamieszczonych w pliku słownika plik_slownika) zapisując do pliku wynikowego (plik_wy) zamiast symboli wejściowych, odpowiadające im ciągi bitów, oczywiście już nie jako kody ASCII. Operację tę wykonuje załączony program demonstracyjny „kom.exe".

Dekodowanie pliku umożliwia program „dek.exe". Operację tę realizuje się również w oparciu o drzewo kodów Huffmana przechodząc drzewo od korzenia do liści wg. ścieżek zapisanych w zakodowanym pliku. Dzięki jednoznaczności kodów odtworzenie zbioru danych wejściowych jest możliwe.

W praktyce należałoby oczywiście zapamiętywać słownik w sposób bardziej optymalny niż w postaci kodów ASCII oraz zamieścić nagłówek w zakodowanym pliku zawierający długość oryginalnego zbioru danych, co uniemożliwi doklejanie dodatkowych znaków na końcu pliku podczas dekompresji.

Pomimo długiego opisu projekt drugi nie jest projektem bardzo czasochłonnym. Zakłada znajomość podstawowych pojęć związanych z budową drzew binarnych, w tym przechodzenia wg. kolejności

preorder, inorder oraz postorder, a także umiejętności dynamicznego tworzenia danych. Ze względu na spóźnione zamieszczenie specyfikacji projektu termin zaliczenia zostanie przesunięty.

III. Załączone programy

Generator kodów Huffmana - gen.exe
Program kompresujący dane - kom.exe
Program dekompresujący - dek.exe

Literatura:

[1] pod redakcją W. Skarbka, „MULTIMEDIA. Algorytmy i standardy kompresji", Akademicka Oficyna Wydawnicza PLJ, Warszawa 1998.