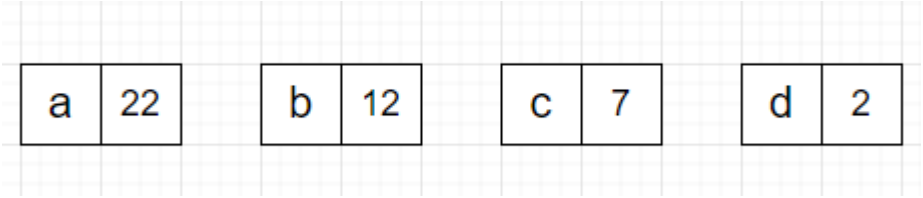
Kodowanie Huffmana – kodowanie polegające na zastępowaniu symboli w łańcuchu wejściowych specjalnymi sekwencjami bitów. Cała idea kodowania Hoffmana polega na tym by znakom występującym najczęściej przypisać najkrótszą sekwencję. Dodatkowo żadna sekwencja bitów nie może być przedłużeniem innej co zapewnie jednoznaczność uzyskanego kodowania.

Przebieg algorytmu:

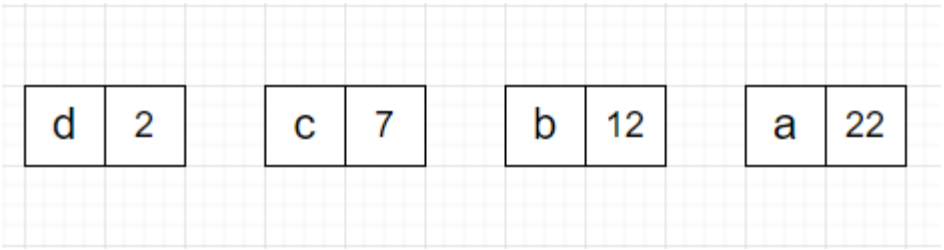
1. Zliczamy wystąpienia każdego ze znaków:

   aaaabbccaaaaccbbaddaaaaaaabbbbcccaaaaaabbbb

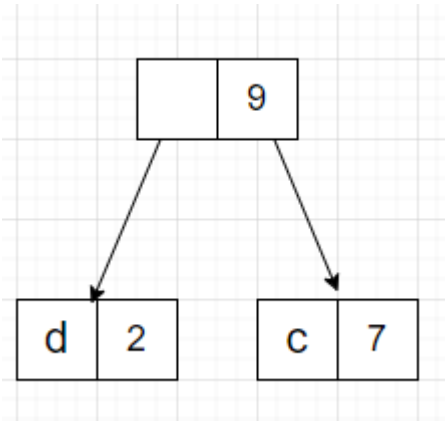
2. Tworzymy z każdego wpisu węzeł drzewa.



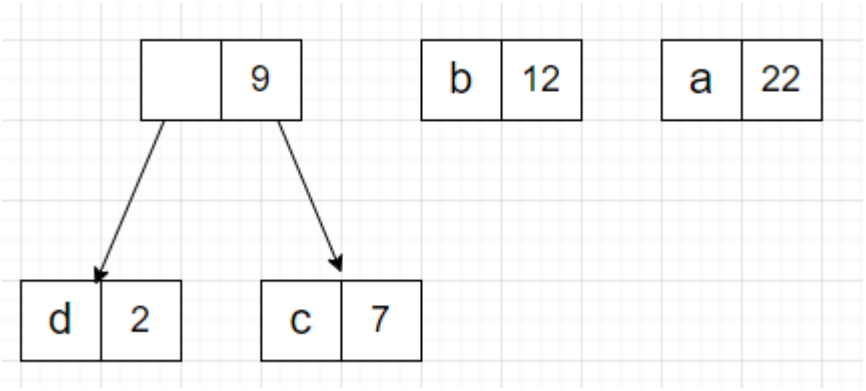
3. Sortujemy węzły.



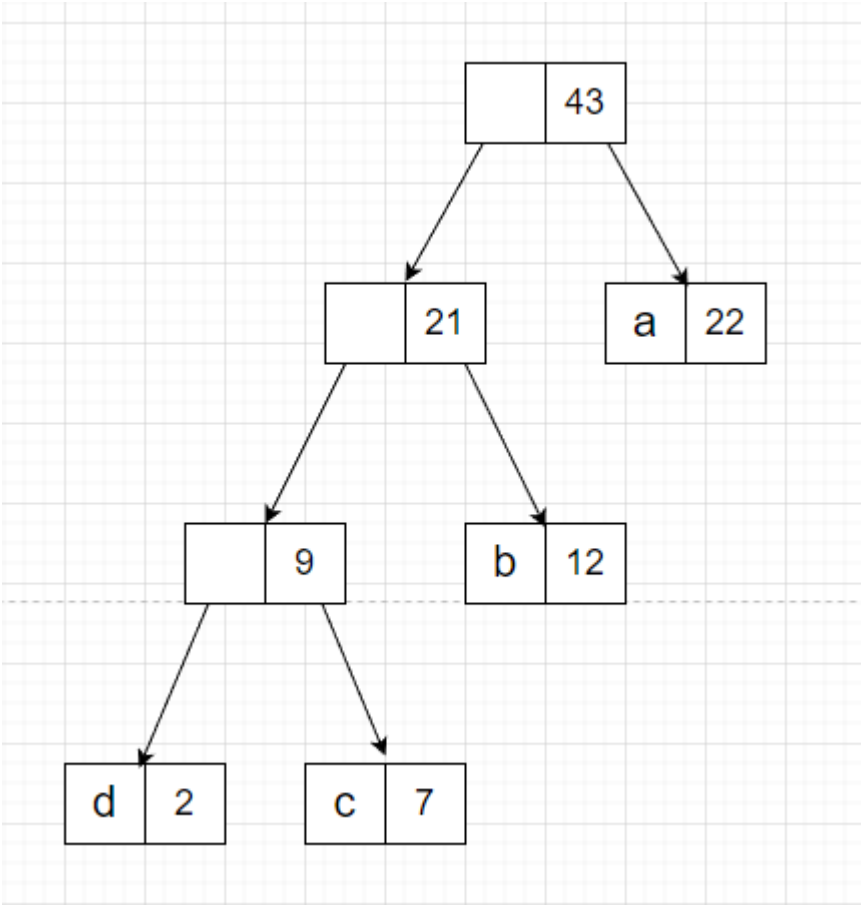
4. Wybieramy dwa najmniejsze węzły i łączymy je w jeden w następujący sposób:



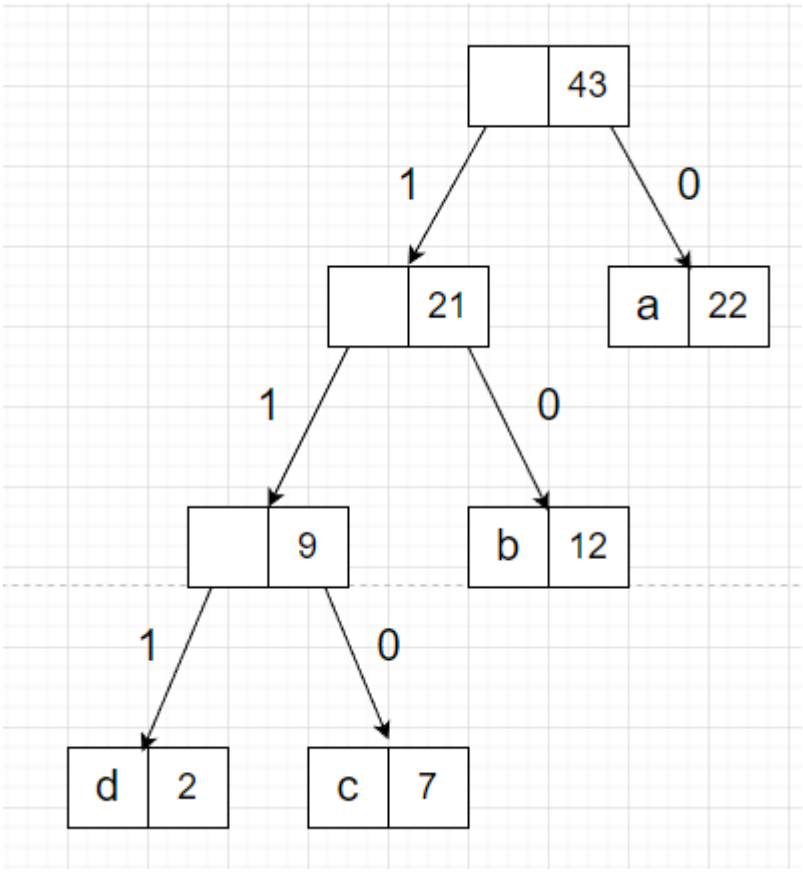
5. Liczba w węźle rodzicu jest sumą ilości wystąpień dzieci. Tak utworzony węzeł dodajemy do listy.



6. Jeżeli ilość węzłów w liście jest większa niż 1 wracamy do punktu 3.
7. W wyniku powyższej pętli powinniśmy uzyskać następujące drzewo:



8. Teraz przydzielamy krawędziom bity 0 i 1. Wszystkim krawędziom wychodzącym w stronę lewego dziecka nadajemy bit 1 a krawędziom wychodzącym w stronę prawego dziecka 0. Nie jest istotne, z której strony znajduje się który bit. Ważne by zawsze z lewej strony był ten sam i analogicznie z prawej. Przykład:



9. Dla każdej literki tworzymy kod czytając wartości przy krawędziach od góry:

   a: 0

   b: 10

   c: 110

   d:111

10. Możemy teraz w oryginalnej wiadomości podmienić literki na odpowiedni sekwencje:

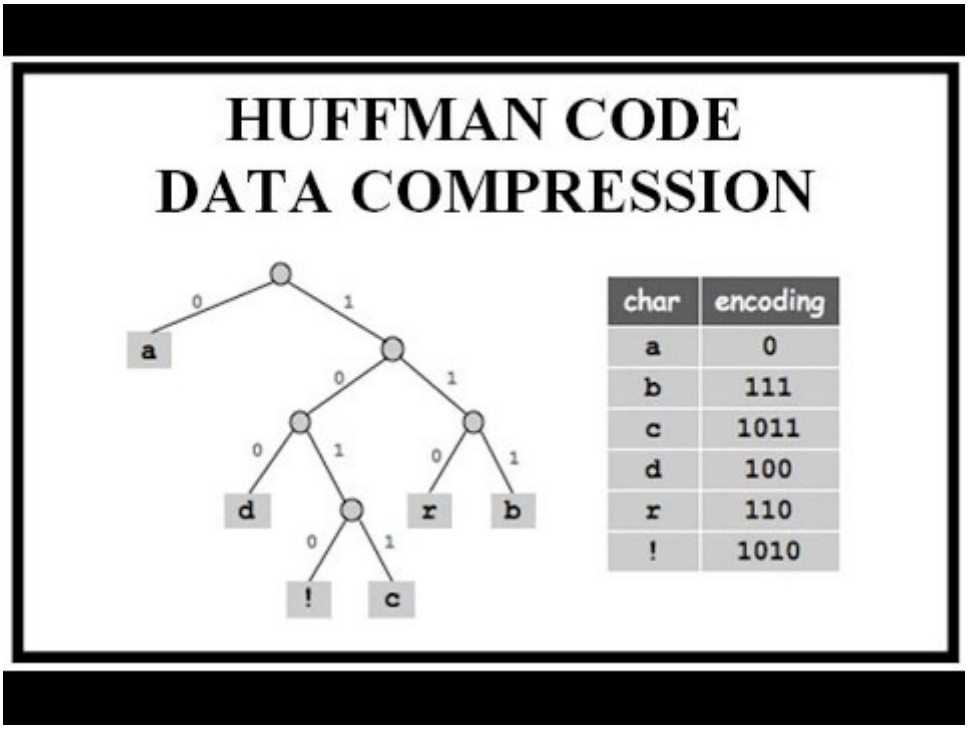
    aaaabbccaaaaccbbaddaaaaaaabbbbcccaaaaaabbbb

    00001010110110000011011010100111111000000010101010110110110000000010101010

Uwaga!

Nie zawsze drzewko będzie się tak ładnie rozchodzić w jednym kierunku. Wszystko zależy od częstotliwości wystąpień danych znaków.

Przykłady:



W przykładzie zaprezentowanym za pomocą listy kroków tworzone węzły zawsze po posortowaniu miały najmniejszą wartość. W przypadku danych z powyższego obrazka nowo tworzone węzły wpadają w środek posortowanej listy dlatego też drzewko przybiera inny kształt.

Zadanie:
W dowolny języku zaimplementuj kodowanie Hoffmana. Użytkownik wpisuje w konsoli ciąg znaków do zakodowania a z programu otrzymuje: informację o ilości wystąpień każdego znaku, kod dla każdego znaku, zakodowaną wiadomość.

Na ocenę dst:
Zlicz i wypisz wystąpienia każdego znaku. Wypisz je bez powtórzeń.

Na ocenę db:
Wygeneruj drzewo Huffmana.

Na ocenę bdb:
Odczytaj i wypisz kod dla każdego znaku. Zakoduj i wypisz wiadomość podaną na starcie programu.

Wskazówka 1:

Przykładowy output programu dla przykładu powyżej:

```
Podaj wiadomość do zakodowania:
aaaabbccaaaaccbbaddaaaaaaabbbbcccaaaaaabbbb
a: 22
b: 12
c: 7
d: 2
-------
d: 111
c: 110
b: 10
a: 0
-------
Zakodowana wiadomość:
000010101101100000110110101001111110000000101010101101101100000010101010
```

Wskazówka 2:

Przykładowy węzeł może wyglądać tak (C#):

```
18 references
class TreeNode
{
    public int value;
    public char character = ' ';
    public char bit;

    public TreeNode parent;
    public TreeNode lChild;
    public TreeNode rChild;

    1 reference
    public override string ToString()
    {
        return $"{character}: {value}";
    }
}
```

Gdzie „value” to ilość wystąpień znaku (później suma), „character” to przechowywany w węźle znak, „bit” to wartość bitu krawędzi wchodzącej do danego węzła.

Pozostałe trzy pola to referencje na dzieci i rodzica. Metoda ToString() jest nieistotna.