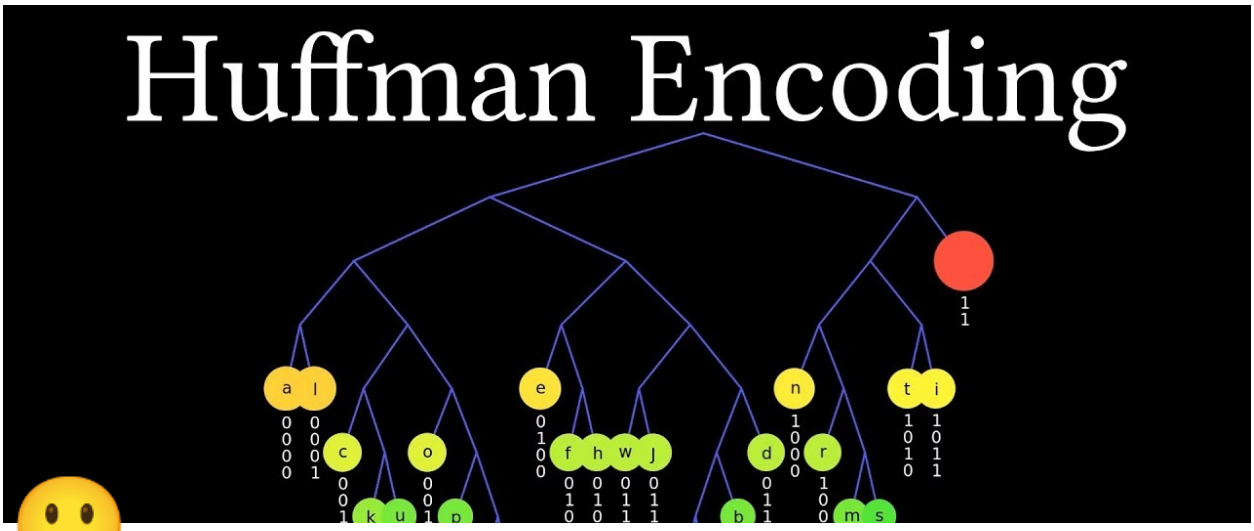


Huffman

# Number	4

👉 Link do wikipedii: https://pl.wikipedia.org/wiki/Kodowanie_Huffmana

👉 Link jak działa drzewo Huffmana: https://www.youtube.com/watch?v=dM6us854Jk0&ab_channel=CSBreakdown

Na czym polega alforytm Huffmana?

Algorytm Huffmana jest używany do **bezstratnej kompresji danych**, czyli zmniejszania rozmiaru plików bez utraty informacji. Polega on na przyporządkowaniu ciągowi znaków **kodów binarnych** o **różnych długościach**, tak aby **najczęściej** używane znaki miały **krótszy kod**, a **rzadziej** używane - **dłuższy**.

Jak działa kodowanie danych algorytmem Huffmana?

1. Bierzemy wiadomość którą chcemy przesłać.
2. Zliczamy wszystkie litery i zapisujemy wyniki w strukturze Node:

```
def __init__(self, freq, symbol, left=None, right=None):
    self.freq = freq  # frequency of the character
    self.symbol = symbol  # the character itself
    self.left = left  # reference to the left child
    self.right = right  # reference to the right child
    self.huff = ''  # the Huffman code for the character
```

   ,gdzie

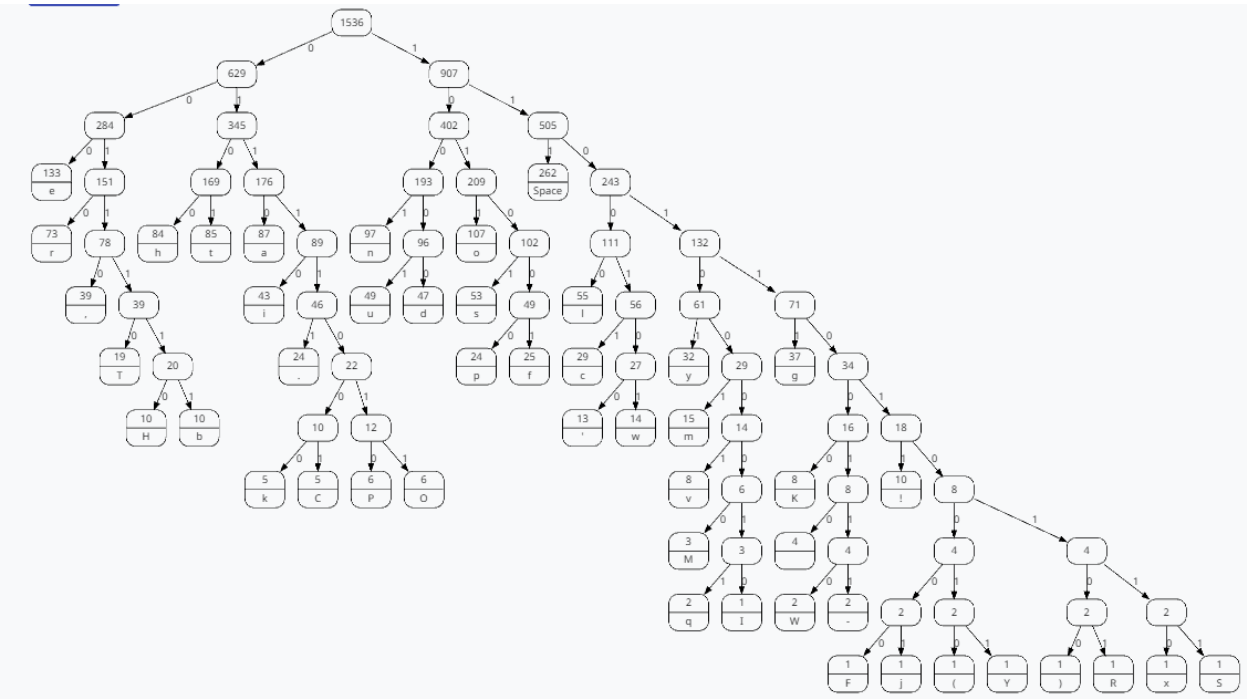
   - `symbol`

```
chars = [
         'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j',
         'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't',
         'u', 'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D',
         'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
         'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X',
         'Y', 'Z', '-', '_', '/', '"', "'", ':', ';', ')',
         '(', '!', '.', '?', ',', chr(10), ' ', '1', '2', '3',
         '4', '5', '6', '7', '8', '9', '0', '=', '@', '*'
        ]
```

   - `freq`

```
freq = [
         87, 10, 29, 47,133, 25, 37, 84, 43, 1 ,
         5 , 55, 15, 97,107, 24, 2 , 73, 53, 85,
         49, 8 , 14, 1 , 32, 0 , 0 , 0 , 5 , 0 ,
         0 , 1 , 0 , 10, 1 , 0 , 8 , 0 , 3 , 0 ,
         6 , 6 , 0 , 1 , 1 , 19, 0 , 0 , 2 , 0 ,
         1 , 0 , 2 , 0 , 0 , 0 , 13, 0 , 0 , 1 ,
         1 , 10, 24, 0 , 39, 0 ,265, 0 , 0 , 0 ,
         0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
        ]
```

   Czyli każda litera naszej wiadomości ma przypisaną swoją ilość wystąpień.
3. Na podstawie symboli i ich ilości wystąpień tworzymy **drzewo Huffmana (**link do filmiku na górze**)**:

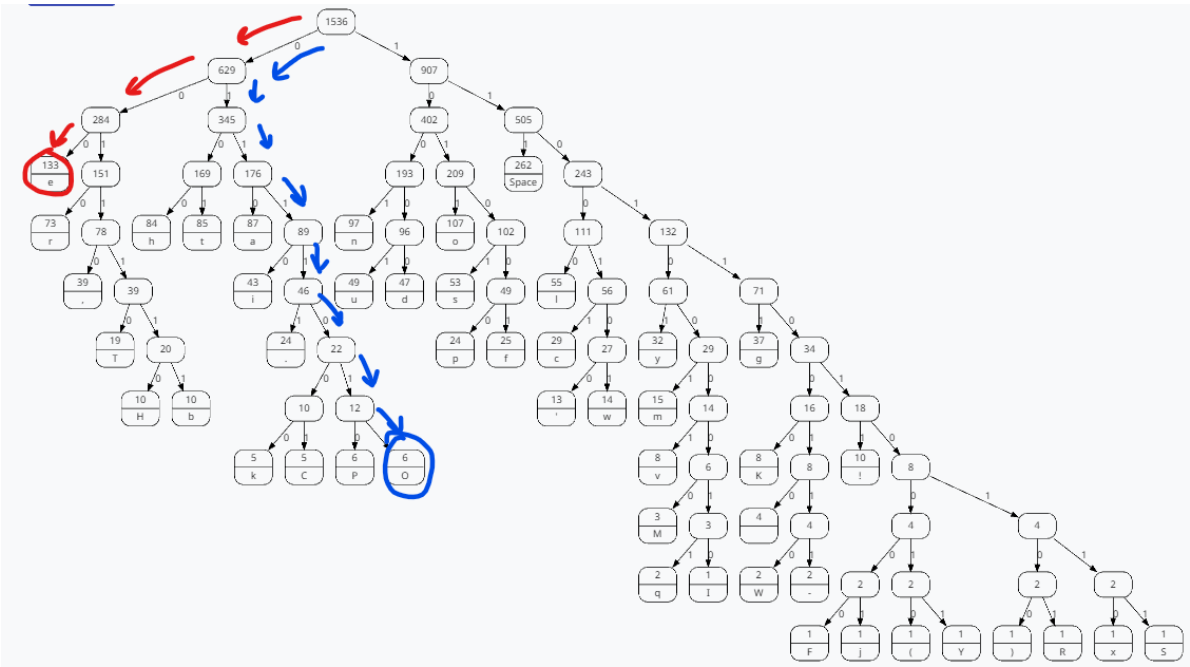


4. Długość kodu danego symbolu zależy od tego przez ile gałęzi trzba przejść aby dostać się do jakiejś litery. Dodatkowo jeżeli “idziemy” w danym kroku w lewo to wartość bitu ma wartość 0, a jeżeli prawo → 1 **np.**

   - e = 000 (dużo wystąpień)
   - O = 01111011 (mało wystąpień)

   Im dany symbol jest częściej używany, tym długość jego kodu binarnego jest krótsza.

   Im dany symbol jest rzadziej używany, tym długość jego kodu binarnego jest dłuższa.



5. każdy znak w wiadomości jest zastępowany kodem binarnym odpowiadającym temu znakowi z drzewa Huffmana, jest to nasza skompresowana wiadomość.

Jak działa odkodowywanie danych algorytmem Huffmana?

1. Odczytaj zakodowaną wiadomość bitową.
2. Przejdź przez drzewo Huffmana, zaczynając od korzenia.
3. Jeżeli bit jest równy 0, przejdź w lewo, jeśli 1, przejdź w prawo.
4. Gdy osiągniesz liść, odczytaj odpowiadający mu symbol.
5. Powtarzaj kroki 2-4, aż do osiągnięcia końca ciągu bitów.
6. Dzięki temu uzyskaliśmy pierwotną wiadomość.