Sprawozdanie z zajęć 14.03.2023

Konrad Bik

Marzec 2023

1 Używane biblioteki

Program został napisany w Pythonie w wersji 3.10.5 przy użyciu bibliotek math i time.

2 Omówienie kodu

```
def decBin(rest, x):
    binary = str(bin(rest)[2:])
    return binary.rjust(x, '0')
```

Funkcja decBin(rest, x) konwertuje liczbę całkowitą rest na jej reprezentację binarną, a następnie doprowadza ją do długości x poprzez dodanie zer na początku lub usunięcie początkowych zer. Wynik jest zwracany jako ciąg znaków.

```
def getFromFile():
    with open('do_kompresji.txt', 'r', encoding='utf-8') as file:
        data = ''.join(file.read())
    return data
```

Funkcja getFromFile() odczytuje zawartość pliku 'do_kompresji.txt' i zwraca ją jako ciąg znaków.

```
if __name__ == '__main__':
    dataBin = ''
    data = getFromFile()
    dataDict = sorted(list(set(data)))
```

Następnie, w sekcji if __name__ == '__main__':, następuje wykonanie programu. Zmienna dataBin jest inicjalizowana jako pusty ciąg znaków. Następnie odczytywana jest zawartość pliku 'do_kompresji.txt' za pomocą funkcji getFromFile(). Kolejnym krokiem jest stworzenie listy dataDict, która zawiera wszystkie unikalne znaki występujące w tekście, posortowane w kolejności alfabetycznej.

```
    X = len(dataDict)
    N = math.ceil(math.log2(X))
    R = (8 - (3 + N * len(data))%8)%8
```

Zmienna X zawiera liczbę unikalnych znaków, a N to minimalna liczba bitów potrzebna do zakodowania każdego znaku. Zmienna R to liczba bitów, które zostaną dodane na koniec zakodowanego tekstu, aby liczba bitów była podzielna przez 8. Wartość R wynosi 0, jeśli N*len(data) jest już podzielna przez 8, w przeciwnym razie wartość ta wynosi 8 minus reszta z dzielenia (3 + N * len(data)) przez 8.

```
    with open('skompresowany2.txt', 'wb') as comp:
        res = bytearray()
        res.append(X)

        for j in dataDict:
            res.append(ord(j))

        dataBin = decBin(R, 3)
        binSecond = ''
```

```
        for char in data:
            binSecond += decBin(dataDict.index(char), N)

        binSecond += str(1) * R
        dataBin += binSecond

        for i in range(0, len(dataBin), 8):
            swToChar = chr(int(dataBin[i:(i+8)], 2))
            res.append(ord(swToChar))

        comp.write(bytes(res))
```

Następnie, otwierany jest plik 'skompresowany2.txt' za pomocą funkcji open() w trybie binarnym. Tworzona jest lista res, która będzie przechowywała skompresowany tekst jako ciąg bajtów. Pierwszy bajt to wartość X, czyli liczba unikalnych znaków. Następnie dla każdego unikalnego znaku dodawana jest jego wartość ASCII do listy res. Następnie wywoływana jest funkcja decBin() z argumentami R i 3, a wynik jest przypisywany do zmiennej dataBin.
Kolejnym krokiem jest zakodowanie tekstu. Dla każdego znaku w tekście, funkcja decBin() jest wywoływana z argumentami dataDict.index(char) i N, a wynik jest dodawany do ciągu znaków binSecond. Na końcu do binSecond dodawane są R jedynki. Następnie łączone są ciągi.

```
    endTime = time.time()
    print(f'Elapsed time {endTime - startTime}')
```

Na końcu wypisywane są wartości X, N i R, a także mierzony jest czas wykonania od tego momentu za pomocą funkcji time.time().