Kod Shannona

Część 1

```
In[6]:= ClearAll["Global`*"]
        ShannonAlgorithm[BaseWithCharsAndProbability_] :=
         Module[{X = BaseWithCharsAndProbability, SumOfPreviousProbabilities, SortedX },
        SortedX = SortBy[X, Last, Greater];

        probability = Table[SortedX[[i, 2]], {i, 1, Length[X]}];

        SumOfPreviousProbabilities := {0};
        For[i = 1, i < Length[X], i++,
        AppendTo[SumOfPreviousProbabilities, Sum[probability[[j]], {j, 1, i}]]];
        ShannonLength = Table[ Ceiling[Log[2, 1 / probability[[i]]]] , {i, 1, Length[SortedX]}];

        CodedWords := {};
        For[i = 1, i ≤ Length[SumOfPreviousProbabilities], i++,
        tmp = SumOfPreviousProbabilities[[i]];
        OneCodedWord = {};
        For[j = 1, j ≤ ShannonLength[[i]], j++,
        If[tmp ≥ 1 / 2 ^ j, tmp = tmp - (1 / 2 ^ j);
               AppendTo[OneCodedWord, 1], AppendTo[OneCodedWord, 0]];
        ];

        AppendTo[CodedWords, StringJoin[ToString /@ OneCodedWord]];
        ];
        Base := {};
        For[i = 1, i ≤ Length[X], i++,
        AppendTo[Base, {SortedX[[i, 1]], CodedWords[[i]]}];
        ];
        Return[Base]
        ]
```

```
In[8]:= X = {{x1, 0.2}, {x2, 0.1}, {x3, 0.05}, {x4, 0.05}, {x5, 0.2}, {x6, 0.25}, {x7, 0.15}};
```

```
        Y = ShannonAlgorithm[X]
Out[9]= {{x6, 00}, {x5, 010}, {x1, 011}, {x7, 101}, {x2, 1100}, {x4, 11100}, {x3, 11110}}
```

Kodowanie

```
In[12]:= Coding[BaseOfCode_, WordWeWantToCode_] :=
          Module[{Base = BaseOfCode, Word = WordWeWantToCode},
         CodedWord := {};
         For[i = 1, i ≤ Length[Word], i++,
         For[j = 1, j ≤ Length[Base], j++,
         If[Word⟦i⟧ == Base⟦j, 1⟧, AppendTo[CodedWord, Base⟦j, 2⟧]];
         ];
         ];
         ];
         Return[CodedWord]
         ]
```

```
         Y = ShannonAlgorithm[X]
         M = {x4, x7, x2}
         Coding[Y, M]
Out[13]=
         {{x6, 00}, {x5, 010}, {x1, 011}, {x7, 101}, {x2, 1100}, {x4, 11100}, {x3, 11110}}
Out[14]=
         {x4, x7, x2}
Out[15]=
         {11100, 101, 1100}
```

Dekodowanie

```
In[16]:= Decoding[BaseOfCode_, WordWeWantToDecode_] :=
          Module[{Base = BaseOfCode, Word = WordWeWantToDecode},
         DecodedWord := {};
         For[i = 1, i ≤ Length[Word], i++,
         For[j = 1, j ≤ Length[Base], j++,
         If[Word⟦i⟧ == Base⟦j, 2⟧, AppendTo[DecodedWord, Base⟦j, 1⟧];
         ];
         ];
         ];
         Return[DecodedWord]
         ]
         Y = ShannonAlgorithm[X]
         W = {"010", "00", "011", "11110"}
         Decoding[Y, W]
```

```
Out[17]=
         {{x6, 00}, {x5, 010}, {x1, 011}, {x7, 101}, {x2, 1100}, {x4, 11100}, {x3, 11110}}
Out[18]=
         {010, 00, 011, 11110}
Out[19]=
         {x5, x6, x1, x3}
```

Tworzenie kodu na bazie hasła

In[27]:=

```
BuildingCodeBase[BaseWord_] := Module[{Word = BaseWord},
DividedWord := CharacterCounts[Word];
char := Keys@DividedWord;
counts := Values@DividedWord;
probability := Normalize[counts, Total];
Base := {};
For[i = 1, i ≤ Length[probability], i++,
AppendTo[Base, {char[[i]], probability[[i]]}];
];
Return[Base]
]
Example = "alamakota"
X1 = BuildingCodeBase[Example]
Y1 = ShannonAlgorithm[X1]
M := Characters[Example]
CodedExample = Coding[Y1, M]
StringJoin[ToString /@ CodedExample]
Decoding[Y1, CodedExample]
```

Out[28]=

```
alamakota
```

Out[29]=

$$\left\{\left\{a, \frac{4}{9}\right\}, \left\{t, \frac{1}{9}\right\}, \left\{o, \frac{1}{9}\right\}, \left\{k, \frac{1}{9}\right\}, \left\{m, \frac{1}{9}\right\}, \left\{l, \frac{1}{9}\right\}\right\}$$

Out[30]=

```
{{a, 00}, {t, 0111}, {o, 1000}, {m, 1010}, {l, 1100}, {k, 1110}}
```

Out[32]=

```
{00, 1100, 00, 1010, 00, 1110, 1000, 0111, 00}
```

Out[33]=

```
0011000010100011101000011100
```

Out[34]=

```
{a, l, a, m, a, k, o, t, a}
```