Kauno technologijos universitetas

Matematikos ir gamtos mokslų fakultetas

Grafų teorija ir tinklų analizė

(P110B001)

Individualus darbas

Informacijos suspaudimas taikant Huffmano kodą

Tauras Gaulia (MGTM-9/1)

Studentas

Doc. Mindaugas Šnipas

Dėstytojas

Kaunas, 2021

Turinys

1. Darbo užduotis

Pasirinktą informaciją (tekstą) suspausime taikant Huffmano kodą. Duotą tekstą pakeisime užšifruotu.

2. Metodologija

Šiais laikais įvairūs šaltiniai, grafiniai objektai bei tekstai yra suspaudžiami juos užkoduojant. Siekiant informaciją užkoduoti kuo efektyviau yra taikomi keli algoritmai. Vienas iš jų yra Huffmano algoritmas. Šiuo metodu informacijos simboliai pakeičiami dvejetainės sistemos kodais. Kodavimo pradžioje turėtų būti žinoma simbolių atsiradimo tikimybė. Iš jų yra sudaromas galutinis pranešimas. Dažniausiai naudojami simboliai pakeičiami trumpesniais kodais, o mažiau paplitę – ilgesniais. Šis metodas leidžia sumažinti kodo ilgį kiekvienam originalo pranešimo simboliui. Remiantis šiais duomenimis, statomas Huffmano kodo medis.

Grafų teorijoje medis yra laikomas jungiuoju acikliniu grafu $G = (V; E)$, o jo nusvirusios viršūnės vadinamos lapais. Siekiant iliustruoti algoritmą, braižomas medis su norimais užšifruoti simboliais vietoj lapų. Prieš sudarant medį sudaromas skirtingų simbolių sąrašas. Kiekvieno šiame sąraše esančio simbolio svoris turėtų atitikti to paties simbolio tikimybę tekste. Sudarant simbolių poras pasirenkamas mažiausias svoris. Jei minimalūs rodikliai yra pastebimi keliuose simboliuose, galima laisvai pasirinkti bet kurią porą. Taip sudaromos poros, kol visi simboliai yra sujungiami. Kiekvienam poros elementui priskiriamas skaičius „0“ arba „1“. Sudarius medį simbolių kodai gaunami surašant dvejetainius skaitmenius iš viršaus į apačią, jei medžio lapai yra nusvirę į apačią.

Kartais procesą galima paspartinti vietoj tikimybės skaičiavimo surasti simbolių dažnumą, nes tikimybės yra tiesiogiai proporcingos dažniams. Taip išvengiama dalybos operacija. Šios užduoties sprendimui taip ir padarysime.

Informacijos suspaudimas Huffmano metodu atliktas Python programavimo kalba.

3. Duomenys

Python kodą vaizduosime „Jupyter Notebook“ aplinkoje.

Pasirinktas tekstas suspaudimui:

„This is an example of a Huffman code.“

4. Sprendimas

Importuojame biblioteką ir nuskaitome tekstą:

```
In [1]: import numpy as np
```

Duomenų nuskaitymas

```
In [2]: f = open("Huffman test.txt", "r")
        duom = f.read()
        print(duom)

        This is an example of a Huffman code.
```

Sukuriame medžio atšakų klasę bei sudarome Huffmano kodo funkciją:

Medžio atšakų kūrimas

In [3]:
```
class NodeTree(object):

    def __init__(self, left=None, right=None):
        self.left = left
        self.right = right

    def children(self):
        return (self.left, self.right)

    def nodes(self):
        return (self.left, self.right)

    def __str__(self):
        return '%s_%s' % (self.left, self.right)
```

Huffmano kodo funkcija

In [5]:
```
def huffman_code_tree(node, left=True, binString=''):
    if type(node) is str:
        return {node: binString}
    (l, r) = node.children()
    d = dict()
    d.update(huffman_code_tree(l, True, binString + '0'))
    d.update(huffman_code_tree(r, False, binString + '1'))
    return d
```

Skaičiuojame simbolių svorius (pagal dažnį):

Simbolių svorių skaičiavimas

In [6]:
```
svoris = {}
for c in duom:
    if c in svoris:
        svoris[c] += 1
    else:
        svoris[c] = 1

svoris = sorted(svoris.items(), key=lambda x: x[1], reverse=True)

nodes = svoris

while len(nodes) > 1:
    (key1, c1) = nodes[-1]
    (key2, c2) = nodes[-2]
    nodes = nodes[:-2]
    node = NodeTree(key1, key2)
    nodes.append((node, c1 + c2))

    nodes = sorted(nodes, key=lambda x: x[1], reverse=True)

huffmanCode = huffman_code_tree(nodes[0][0])
```

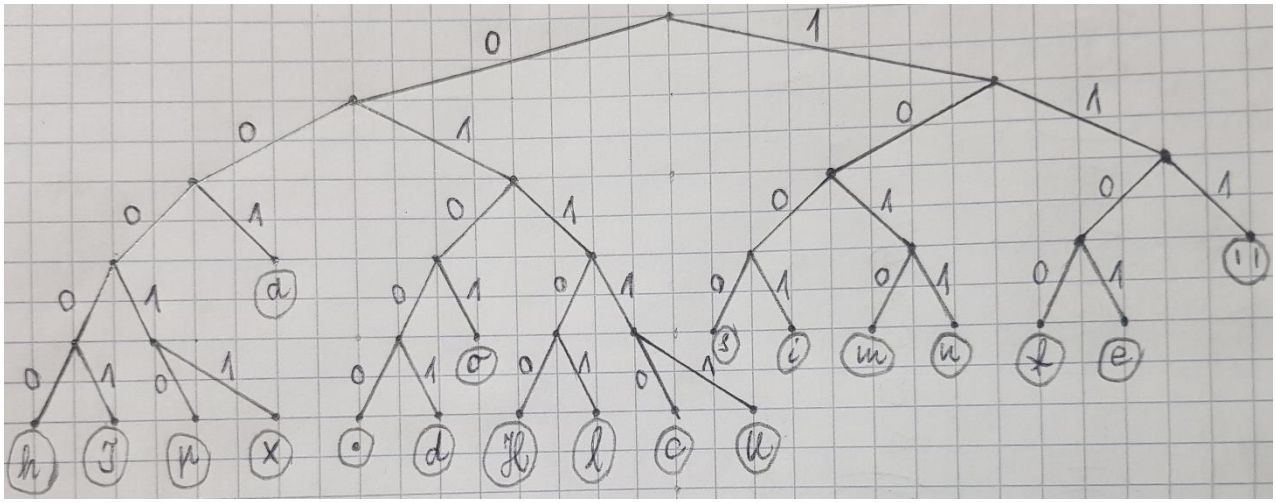
Sudarome simbolių kodų lentelę:

Simbolių kodų lentelė

In [7]:
```
print(' Char | Huffman code ')
print('----------------------')
kodai = []
for (char, frequency) in svoris:
    kodai.append([char,huffmanCode[char]])
    print(' %-4r |%12s' % (char, huffmanCode[char]))
```

```
 Char | Huffman code
----------------------
 ' '  |          111
 'a'  |          001
 'e'  |         1101
 'f'  |         1100
 'i'  |         1001
 's'  |         1000
 'n'  |         1011
 'm'  |         1010
 'o'  |         0101
 'T'  |        00001
 'h'  |        00000
 'x'  |        00011
 'p'  |        00010
 'l'  |        01101
 'H'  |        01100
 'u'  |        01111
 'c'  |        01110
 'd'  |        01001
 '.'  |        01000
```

Nubraižome Huffmano kodo medį:



Apskaičiuojame kiek bitų bus sunaudota tekstui užšifruotam Huffmano kodu:

Bitų kiekis Huffmano kodu

In [8]:
```
bit=0
for (a,b) in svoris:
    bit=bit+(huffmanCode[a].count('')-1)*b
bit
```

Out[8]: 147

Palyginame bitų kiekį šifruojant standartiniu kodavimu:

Bitų kiekis standartiniu kodu

```
In [9]: stbit=0
        stbit_=[]
        for (a,b) in svoris:
            stbit_.append(huffmanCode[a].count('')-1)
        for (a,b) in svoris:
            stbit=stbit+(max(stbit_))*b
        stbit
Out[9]: 185
```

Sutaupomas bitų kiekis

```
In [10]: stbit-bit
Out[10]: 38
```

Naudojant Huffmano kodą sutaupėme 38 bitus. Gauta suspausta informacija:

Užšifruotas tekstas

```
In [11]: sifras = str()
         for c in duom:
             sifras = sifras + huffmanCode[c]
         print(sifras)
```

```
0000100000100110001111001100011100110111111110100011001101000010011011101111010111001110011110110001111110011001010001101111
1011100101010011110101000
```

Patikriname, ar gautas kodas yra teisingas. Iššifruojame tekstą:

Kodo tikrinimas

```
In [12]: atkod = str()
         atkodT = str()
         for c in sifras:
             atkod = atkod + c
             for c1 in range(0,len((kodai[:][:]))):
                 if atkod == kodai[c1][1]:
                     atkodT = atkodT + kodai[c1][0]
                     atkod = ""
         print(atkodT)
```

```
This is an example of a Huffman code.
```

Gavome pradinį tekstą. Vadinasi, kodas yra teisingas.

5. Rezultatai ir išvados

Informaciją suspaudžiant Huffmano kodo algoritmu sunaudojame mažiau bitų nei suspaudžiant standartiniu kodu. Pagal pasirinktą pavyzdį šiame darbe sutaupėme 38 bitus. Jei tekstas būtų ilgesnis, skirtumas būtų dar didesnis. Taigi Huffmano kodas dideliems informacijos kiekiams suspausti yra labai veiksmingas.

```
                    This is an example of a Huffman code.
0000100000100110001111001100011100110111111110100011001101000010011011101111010111001110011110110001111110011001010001101111
1011100101010011110101000
```

Šiai užduočiai pasirinktas tekstas yra nedidelis, nes nubraižyti medį didesniems tekstams būtų gan sudėtinga ir užtruktų nemažai laiko. Vis dėlto, programa dirba ganėtinai greitai ir su ilgesniais tekstais.

6. Literatūra

1. *Kagutech*, [žiūrėta: 2021-12-13]. Prieiga per internetą: https://lit.kagutech.com/4318622-huffman-codes-examples-application