



Tecnológico de Monterrey

Tecnológico de Monterrey - Campus Monterrey
School of Engineering and Sciences
Engineering in Computational Technologies
Analysis and Design of Advanced Algorithms

Class Activity 4: Huffman Codes

Group: 607
Team #3

Luis Salomón Flores Ugalde

Santiago Quintana Moreno A01571222
Miguel Ángel Álvarez Hermida a01722925

FileEditSelectionViewGoRunTerminalHelp←→2.Advanced Algorithms

ClassAct4_TheBells.py ×

HuffmanCodes > ClassAct4_TheBells.py > Node > __init__

```
1 # Analysis and Design of Advanced Algorithms
2 # Group #607
3 # Team 3
4 # Luis Salomón Flores Ugalde
5
6 # Santiago Quintana Moreno A01571222
7 # Miguel Ángel Álvarez Hermida A01722925
8
9 # ----- HUFFMAN CODES -----
10
11 import heapq
12 from collections import defaultdict, Counter
13 import pickle
14 import os
15
16 class Node:
17     def __init__(self, char=None, freq=0, left=None, right=None):
18         self.char = char
19         self.freq = freq
20         self.left = left
21         self.right = right
22
23     def __lt__(self, other):
24         return self.freq < other.freq
25
26 class HuffmanCodes:
27     def __init__(self):
28         self.root = None
29         self.codes = {}
30         self.reverse_codes = {}
31
32     def calculate_probabilities(self, text):
33         """Calculate character frequencies and probabilities"""
34         char_count = Counter(text)
35         total_chars = len(text)
36         probabilities = {char: count/total_chars for char, count in char_count.items()}
37         return char_count, probabilities
38
39     def build_huffman_tree(self, char_freq):
40         """Build Huffman tree from character frequencies"""
41         if not char_freq:
42             return None
```

powershell ×

```
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\User
s\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVER
SIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/HuffmanCodes/ClassAct4_TheBell
s.py"
Files created:
- probabilities.txt (character probabilities)
- huffman_tree.txt (Huffman codes)
- encoded_text.txt (encoded binary text)

--- Huffman Encoder/Decoder Menu ---
1. Encode text
2. Decode binary
3. Exit
Enter your choice (1-3): 1
Enter text to encode: This is encoded Plain Text
Encoded: 00000010111111100011011111000110101000101000100001110000101010001
101111011011111100010101000001010110011011000101001

--- Huffman Encoder/Decoder Menu ---
1. Encode text
2. Decode binary
3. Exit
Enter your choice (1-3): 2
Enter binary to decode: 000000101111111000110111110001101100001010100010000
1110000101010000110111001111101101111100101011001010110010101100001001
Decoded: This is decoded plain text

--- Huffman Encoder/Decoder Menu ---
1. Encode text
2. Decode binary
3. Exit
Enter your choice (1-3): 3
Goodbye!
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
```

main ↺ 🔍 📄 ⚙️ 📶 📶 📶

FileEditSelectionViewGoRunTerminalHelp←→2.Advanced Algorithms

ClassAct4_TheBells.py ×

HuffmanCodes > ClassAct4_TheBells.py > Node > __init__

```
1 # Analysis and Design of Advanced Algorithms
2 # Group #607
3 # Team 3
4 # Luis Salomón Flores Ugalde
5
6 # Santiago Quintana Moreno A01571222
7 # Miguel Ángel Álvarez Hermida A01722925
8
9 # ----- HUFFMAN CODES -----
10
11 import heapq
12 from collections import defaultdict, Counter
13 import pickle
14 import os
15
16 class Node:
17     def __init__(self, char=None, freq=0, left=None, right=None):
18         self.char = char
19         self.freq = freq
20         self.left = left
21         self.right = right
22
23     def __lt__(self, other):
24         return self.freq < other.freq
25
26 class HuffmanCodes:
27     def __init__(self):
28         self.root = None
29         self.codes = {}
30         self.reverse_codes = {}
31
32     def calculate_probabilities(self, text):
33         """Calculate character frequencies and probabilities"""
34         char_count = Counter(text)
35         total_chars = len(text)
36         probabilities = {char: count/total_chars for char, count in char_count.items()}
37         return char_count, probabilities
38
39     def build_huffman_tree(self, char_freq):
40         """Build Huffman tree from character frequencies"""
41         if not char_freq:
42             return None
```

...

powershell ×

```
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\User
s\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVER
SIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/HuffmanCodes/ClassAct4_TheBell
s.py"
Files created:
- probabilities.txt (character probabilities)
- huffman_tree.txt (Huffman codes)
- encoded_text.txt (encoded binary text)

--- Huffman Encoder/Decoder Menu ---
1. Encode text
2. Decode binary
3. Exit
Enter your choice (1-3): 1
Enter text to encode: This is encoded Plain Text
Encoded: 00000010111111100011011111000110101000101000100001110000101010001
101111011011111100010101000001010110011011000101001

--- Huffman Encoder/Decoder Menu ---
1. Encode text
2. Decode binary
3. Exit
Enter your choice (1-3): 2
Enter binary to decode: 000000101111111000110111110001101100001010100010000
1110000101010000110111001111101101111100101011001010110010101100001001
Decoded: This is decoded plain text

--- Huffman Encoder/Decoder Menu ---
1. Encode text
2. Decode binary
3. Exit
Enter your choice (1-3): 3
Goodbye!
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
```

main ↺ 🔍 🔄 ⏏ ⏮ ⏭ ⏪ ⏩ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹ ⏺ ⏻ ⏼ ⏽ ⏾ ⏿ ⏰ ⏱ ⏲ ⏳ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹ ⏺ ⏻ ⏼ ⏽ ⏾ ⏿ ⏰ ⏱ ⏲ ⏳

Go Live

FileEditSelectionViewGoRunTerminalHelp←→2.Advanced Algorithms

ClassAct4_TheBells.py ×

HuffmanCodes > ClassAct4_TheBells.py > Node > __init__

```
26 class HuffmanCodes:
71     def decode(self, binary_text):
81
82         if current.char: # Leaf node
83             decoded.append(current.char)
84             current = self.root
85
86     return ''.join(decoded)
87
88 def save_tree(self, filename):
89     """Save Huffman tree to file"""
90     # Get the directory where this script is located
91     script_dir = os.path.dirname(os.path.abspath(__file__))
92     file_path = os.path.join(script_dir, filename)
93
94     with open(file_path, 'w', encoding='utf-8') as f:
95         f.write("Huffman Codes:\n")
96         for char, code in sorted(self.codes.items()):
97             if char == '\n':
98                 f.write(f"'\n': {code}\n")
99             elif char == ' ':
100                 f.write(f"'space': {code}\n")
101             else:
102                 f.write(f'{char}: {code}\n')
103
104 def main():
105     huffman = HuffmanCodes()
106
107     # Read the text file
108     try:
109         # Get the directory where this script is located
110         script_dir = os.path.dirname(os.path.abspath(__file__))
111         file_path = os.path.join(script_dir, "TheBells_EAP.txt")
112
113         with open(file_path, 'r', encoding='utf-8') as f:
114             text = f.read()
115     except FileNotFoundError:
116         print("Error: TheBells_EAP.txt not found")
117         return
118
119     # Calculate probabilities
120     char_freq, probabilities = huffman.calculate_probabilities(text)
```

powershell ×

```
PS D:\SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\User
s\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVER
SIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/HuffmanCodes/ClassAct4_TheBell
s.py"
Files created:
- probabilities.txt (character probabilities)
- huffman_tree.txt (Huffman codes)
- encoded_text.txt (encoded binary text)

--- Huffman Encoder/Decoder Menu ---
1. Encode text
2. Decode binary
3. Exit
Enter your choice (1-3): 1
Enter text to encode: This is encoded Plain Text
Encoded: 0000001011111111000110111111000110101000101000100001110001010100001
101111011011111110010101000001010110011010100001001

--- Huffman Encoder/Decoder Menu ---
1. Encode text
2. Decode binary
3. Exit
Enter your choice (1-3): 2
Enter binary to decode: 00000010111111110001101111110001101100001010100010000
111000010101000011011100111110110111111001010100101011001010100001001
Decoded: This is decoded plain text

--- Huffman Encoder/Decoder Menu ---
1. Encode text
2. Decode binary
3. Exit
Enter your choice (1-3): 3
Goodbye!
PS D:\SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
```

main ↺ 0 0 0

Go Live

FileEditSelectionViewGoRunTerminalHelp←→2.Advanced Algorithms

ClassAct4_TheBells.py X

104 def main():

120 char_freq, probabilities = huffman.calculate_probabilities(text)

121

122 # Get the directory where this script is located for saving files

123 script_dir = os.path.dirname(os.path.abspath(__file__))

124

125 # Save probabilities to file

126 prob_file_path = os.path.join(script_dir, "probabilities.txt")

127 with open(prob_file_path, 'w', encoding='utf-8') as f:

128 f.write("Character Probabilities:\n")

129 for char, prob in sorted(probabilities.items()):

130 if char == '\n':

131 f.write(f"'\n': {prob:.6f}\n")

132 elif char == ' ':

133 f.write(f"'space': {prob:.6f}\n")

134 else:

135 f.write(f"'{char}': {prob:.6f}\n")

136

137 # Build Huffman tree

138 huffman.build_huffman_tree(char_freq)

139

140 # Save tree to file

141 huffman.save_tree("huffman_tree.txt")

142

143 # Encode the text

144 encoded_text = huffman.encode(text)

145

146 # Save encoded text

147 encoded_file_path = os.path.join(script_dir, "encoded_text.txt")

148 with open(encoded_file_path, 'w') as f:

149 f.write(encoded_text)

150

151 print("Files created:")

152 print("- probabilities.txt (character probabilities)")

153 print("- huffman_tree.txt (Huffman codes)")

154 print("- encoded_text.txt (encoded binary text)")

155

156 # Menu for encoding/decoding user input

157 while True:

158 print("\n--- Huffman Encoder/Decoder Menu ---")

159 print("1. Encode text")

160 print("2. Decode binary")

powershell X

PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\User\s\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/HuffmanCodes/ClassAct4_TheBell s.py"

Files created:

- probabilities.txt (character probabilities)

- huffman_tree.txt (Huffman codes)

- encoded_text.txt (encoded binary text)

--- Huffman Encoder/Decoder Menu ---

1. Encode text

2. Decode binary

3. Exit

Enter your choice (1-3): 1

Enter text to encode: This is encoded Plain Text

Encoded: 00000010111111100011011111000110101000101000100001110001010100001

10111101101111110001010000010101100110100001001

--- Huffman Encoder/Decoder Menu ---

1. Encode text

2. Decode binary

3. Exit

Enter your choice (1-3): 2

Enter binary to decode: 000000101111111000110111110001101100001010100010000

111000010101000011011100111110110111110010101001010100110100001001

Decoded: This is decoded plain text

--- Huffman Encoder/Decoder Menu ---

1. Encode text

2. Decode binary

3. Exit

Enter your choice (1-3): 3

Goodbye!

PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>

FileEditSelectionViewGoRunTerminalHelp←→2.Advanced Algorithms

ClassAct4_TheBells.py ×

HuffmanCodes > ClassAct4_TheBells.py > Node > __init__

```
104 def main():
160     print("2. Decode binary")
161     print("3. Exit")
162
163     choice = input("Enter your choice (1-3): ")
164
165     if choice == '1':
166         user_text = input("Enter text to encode: ")
167         try:
168             encoded = huffman.encode(user_text)
169             print(f"Encoded: {encoded}")
170         except KeyError as e:
171             print(f"Error: Character {e} not in Huffman tree")
172
173     elif choice == '2':
174         binary_input = input("Enter binary to decode: ")
175         try:
176             decoded = huffman.decode(binary_input)
177             print(f"Decoded: {decoded}")
178         except:
179             print("Error: Invalid binary sequence")
180
181     elif choice == '3':
182         print("Goodbye!")
183         break
184
185     else:
186         print("Invalid choice. Please try again.")
187
188 if __name__ == "__main__":
189     main()
```

powershell ×

PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\User\s\anty\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/HuffmanCodes/ClassAct4_TheBells.py"

Files created:
- probabilities.txt (character probabilities)
- huffman_tree.txt (Huffman codes)
- encoded_text.txt (encoded binary text)

--- Huffman Encoder/Decoder Menu ---
1. Encode text
2. Decode binary
3. Exit
Enter your choice (1-3): 1
Enter text to encode: This is encoded Plain Text
Encoded: 0000001011111110001101111110001101010001010001000011100001010100001
1011110110111111100101000000101011001101100001001

--- Huffman Encoder/Decoder Menu ---
1. Encode text
2. Decode binary
3. Exit
Enter your choice (1-3): 2
Enter binary to decode: 0000001011111110001101111110001101100001010100010000
11100001010100001101110011111011011111100101011001101101101100001001
Decoded: This is decoded plain text

--- Huffman Encoder/Decoder Menu ---
1. Encode text
2. Decode binary
3. Exit
Enter your choice (1-3): 3
Goodbye!
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>

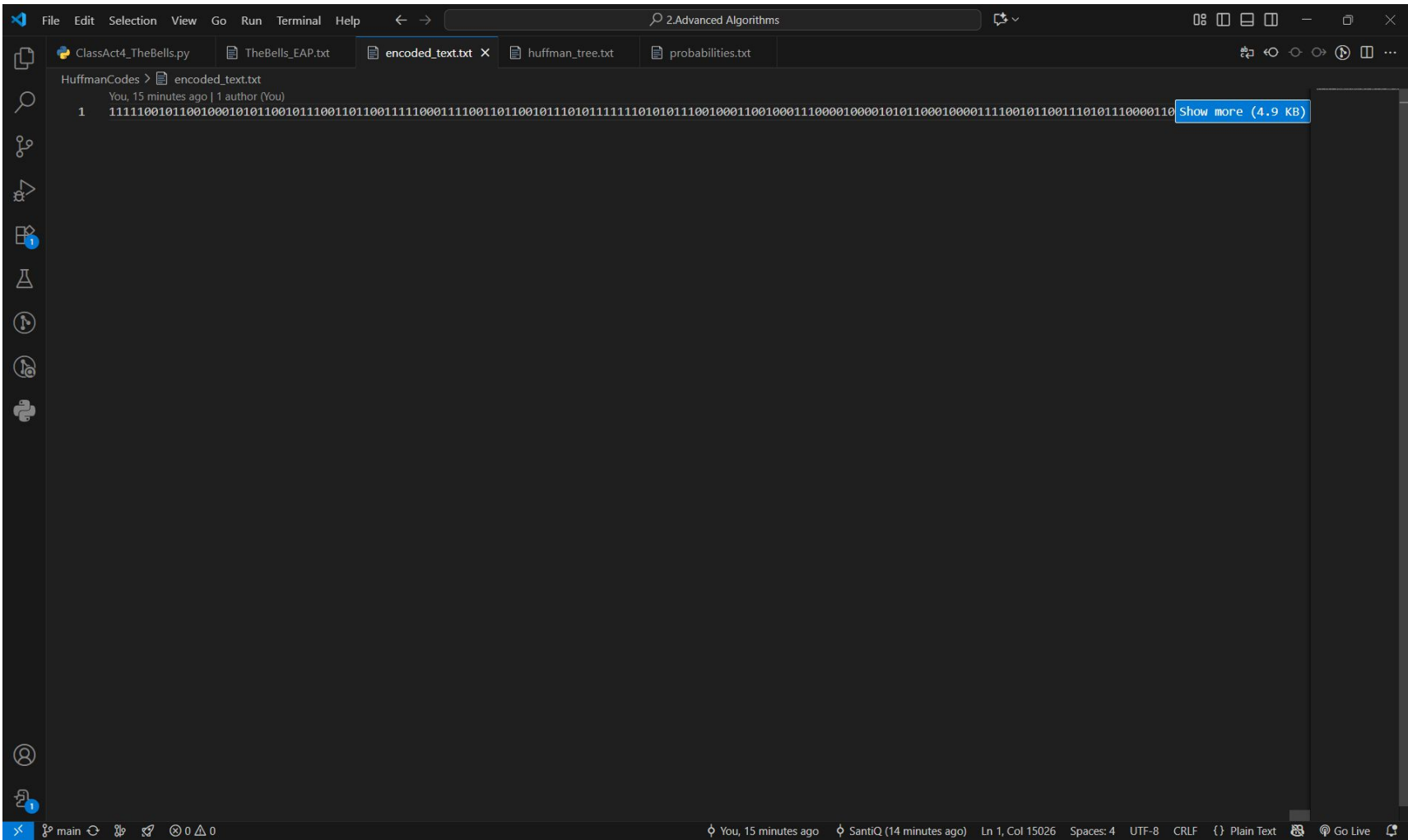
main ↺ 🔍 🔄 0 ⚠ 0


HuffmanCodes > TheBells_EAP.txt

I.
Hear the sledges with the bells—
Silver bells!
What a world of merriment their melody foretells!
How they tinkle, tinkle, tinkle,
In the icy air of night!
While the stars that oversprinkle
All the heavens, seem to twinkle
With a crystalline delight;
Keeping time, time, time,
In a sort of Runic rhyme,
To the tintinnabulation that so musically wells
From the bells, bells, bells, bells,
Bells, bells, bells—
From the jingling and the tinkling of the bells.

II.
Hear the mellow wedding bells,
Golden bells!
What a world of happiness their harmony foretells!
Through the balmy air of night
How they ring out their delight!
From the molten-golden notes,
And all in tune,
What a liquid ditty floats
To the turtle-dove that listens, while she gloats
On the moon!
Oh, from out the sounding cells,
What a gush of euphony voluminously wells!
How it swells!
How it dwells
On the Future! how it tells
Of the rapture that impels
To the swinging and the ringing
Of the bells, bells, bells,
Of the bells, bells, bells, bells,
Bells, bells, bells—
To the rhyming and the chiming of the bells!

III.
Hear the loud alarum bells—



 ClassAct4_TheBells.py

probabilities.txt X

HuffmanCodes > probabilities.txt

Character Probabilities:

2	'n': 0.035800
3	'space': 0.146510
4	':': 0.007220
5	',' : 0.028881
6	'-': 0.000903
7	'_': 0.002708
8	':': 0.000602
9	':': 0.001504
10	'A': 0.004513
11	'B': 0.002407
12	'D': 0.000301
13	'F': 0.002106
14	'G': 0.000602
15	'H': 0.003911
16	'I': 0.006318
17	'K': 0.001203
18	'L': 0.000301
19	'N': 0.000301
20	'O': 0.005114
21	'R': 0.001504
22	'S': 0.000301
23	'T': 0.005114
24	'V': 0.000301
25	'W': 0.003610
26	'Y': 0.000602
27	'a': 0.038508
28	'b': 0.021059
29	'c': 0.008424
30	'd': 0.016546
31	'e': 0.101384
32	'f': 0.017750
33	'g': 0.019856
34	'h': 0.055355
35	'i': 0.042419
36	'j': 0.000602
37	'k': 0.004813
38	'l': 0.000024
39	'm': 0.017750
40	'n': 0.049940
41	'o': 0.050241

<https://colab.research.google.com/drive/1GAjJSPmSfVweoDj5X2JqajzWx-SEF1tX?usp=sharing>

REFERENCES

GeeksforGeeks. (2025, July 23). *Huffman Coding | Greedy algo3*. GeeksforGeeks.

<https://www.geeksforgeeks.org/dsa/huffman-coding-greedy-algo-3/>

GeeksforGeeks. (2025, July 23). *Huffman Coding in Python*. GeeksforGeeks.

<https://www.geeksforgeeks.org/dsa/huffman-coding-in-python/>

W3Schools.com. (n.d.). https://www.w3schools.com/dsa/dsa_ref_huffman_coding.php