--Log3430 -- Méthodes de test et de validation du logiciel

Noureddine Kerzazi |
Noureddine.Kerzazi@polymtl.ca

Bram Adams |
bram.adams@polymtl.ca

Test MaDUM

Lab #3

Objectifs : **Test OO**

- Construire le MaDUM pour une classe sous tests.
- Identifier les tranches de données de la classe.
- Générer les séquences de test et les implémenter à l'aide de unittest.

Agenda

- Retour sur le Lab#1 et Lab #2
- Présentation de l'algo Huffman
- Démo du code source (Python)
- Lab #3
- À vous !!!
- Consignes de remise.

Huffman

- Un Algo de compression de données sans perte d'information
- Basé sur des longueurs de codes d'attribution basées sur des fréquences
- Il y a principalement deux parties principales dans Huffman Coding
  - 1) Construire un arbre Huffman à partir des caractères saisis.
  - 2) Parcourez l'arbre Huffman et attribuez des codes aux personnages.

Exemple

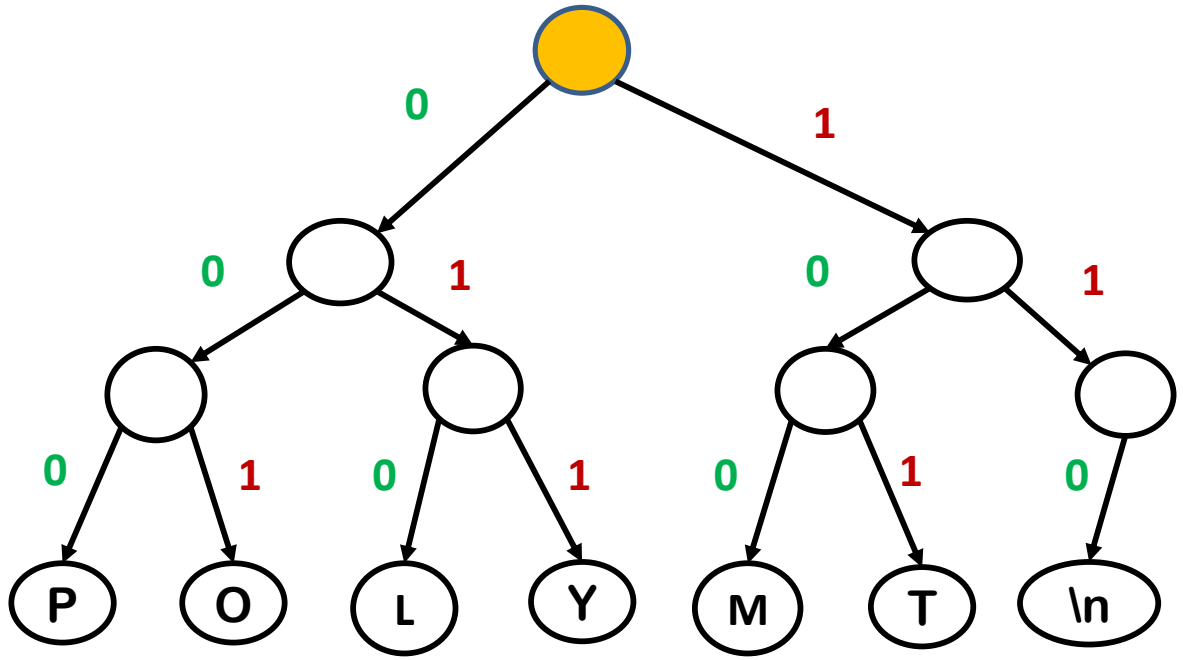
Caractère	Code	Fréquence	Nb Bits
P	000	10	30
O	001	15	45
L	010	12	36
Y	011	3	12
M	100	4	12
T	101	13	39
\n	110	1	3

Total Nb Bits = 174

- **Notre objectif est :**
  - De réduire le nombre total de Bits (174).

L’arbre Huffman

- La traversé vers la gauche à toujours la valeur 0
- Traversé à droite la valeur 1

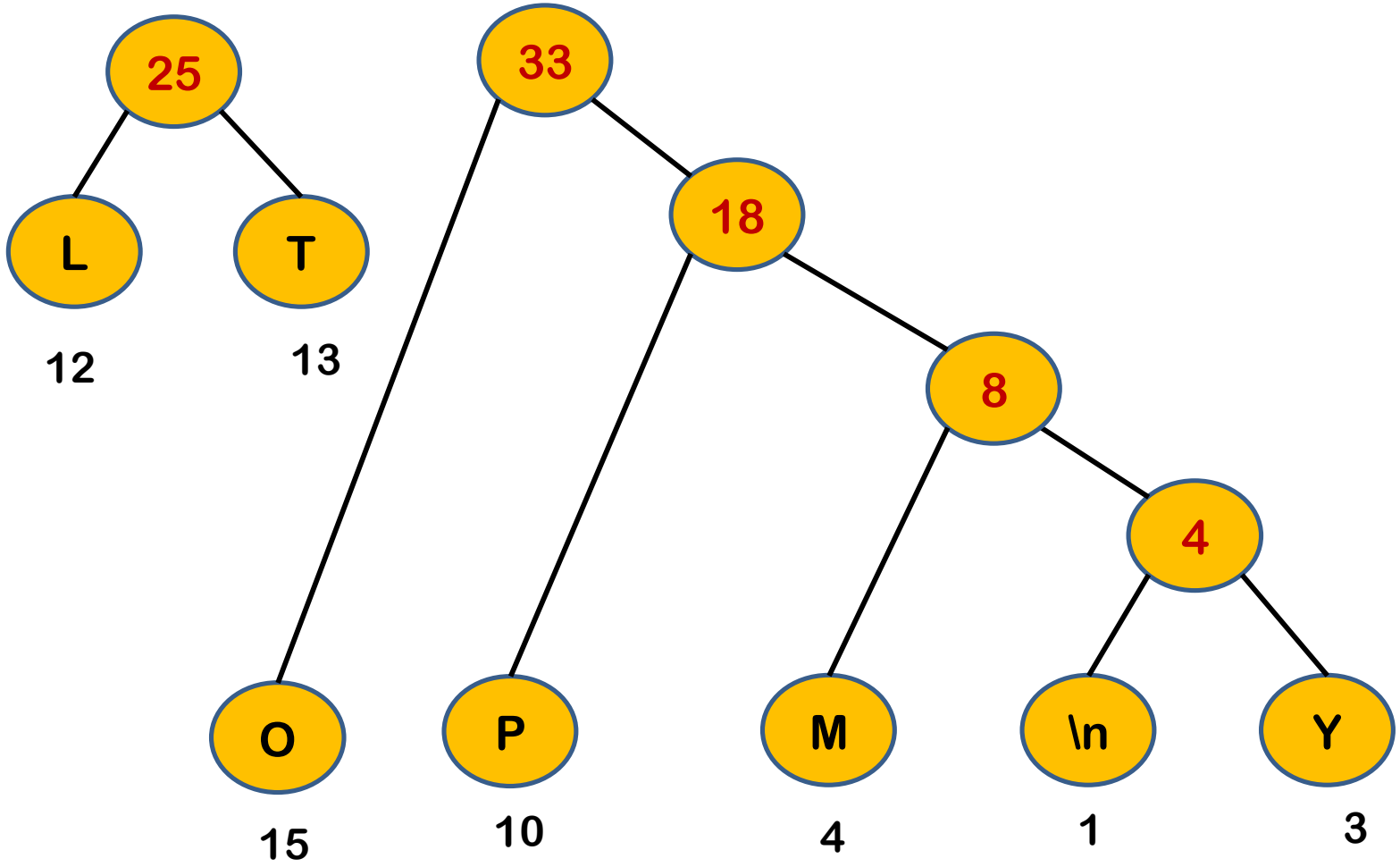


- P 000
- T 101
- \n 110

Nous souhaitons réduire le codage de l’arbre pour les nœuds qui sont très fréquent dans le texte

Exemple

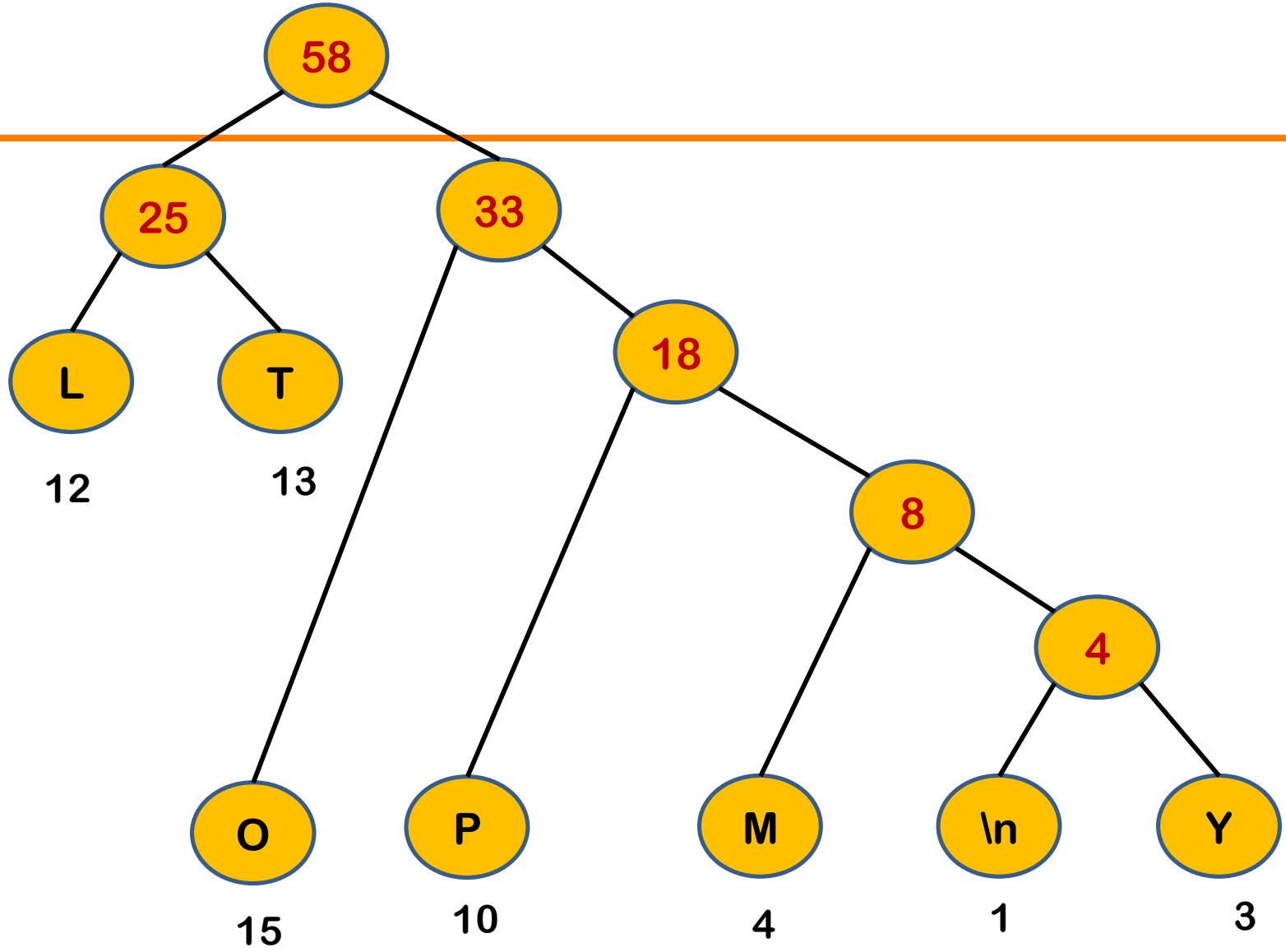
Caractère	Fréquence
P	10
O	15
L	12
Y	3
M	4
T	13
\n	1



- Prendre les deux caractères avec la fréquence minimale et créer deux feuilles

Exemple

Caractère	Fréquence
P	10
O	15
L	12
Y	3
M	4
T	13
\n	1

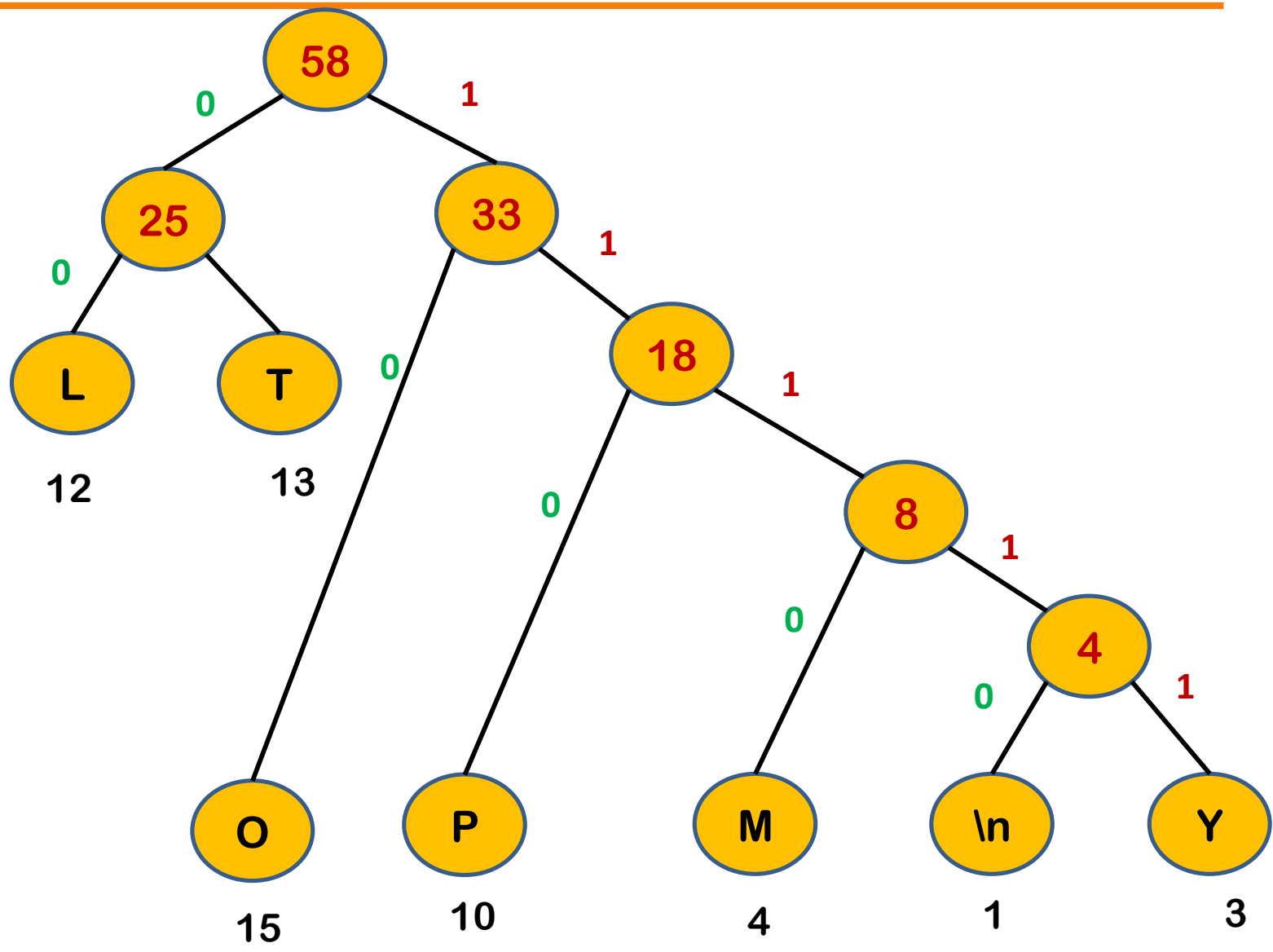


- Prendre les deux caractères avec la fréquence minimale et créer deux feuilles (ici y et \n)

Exemple

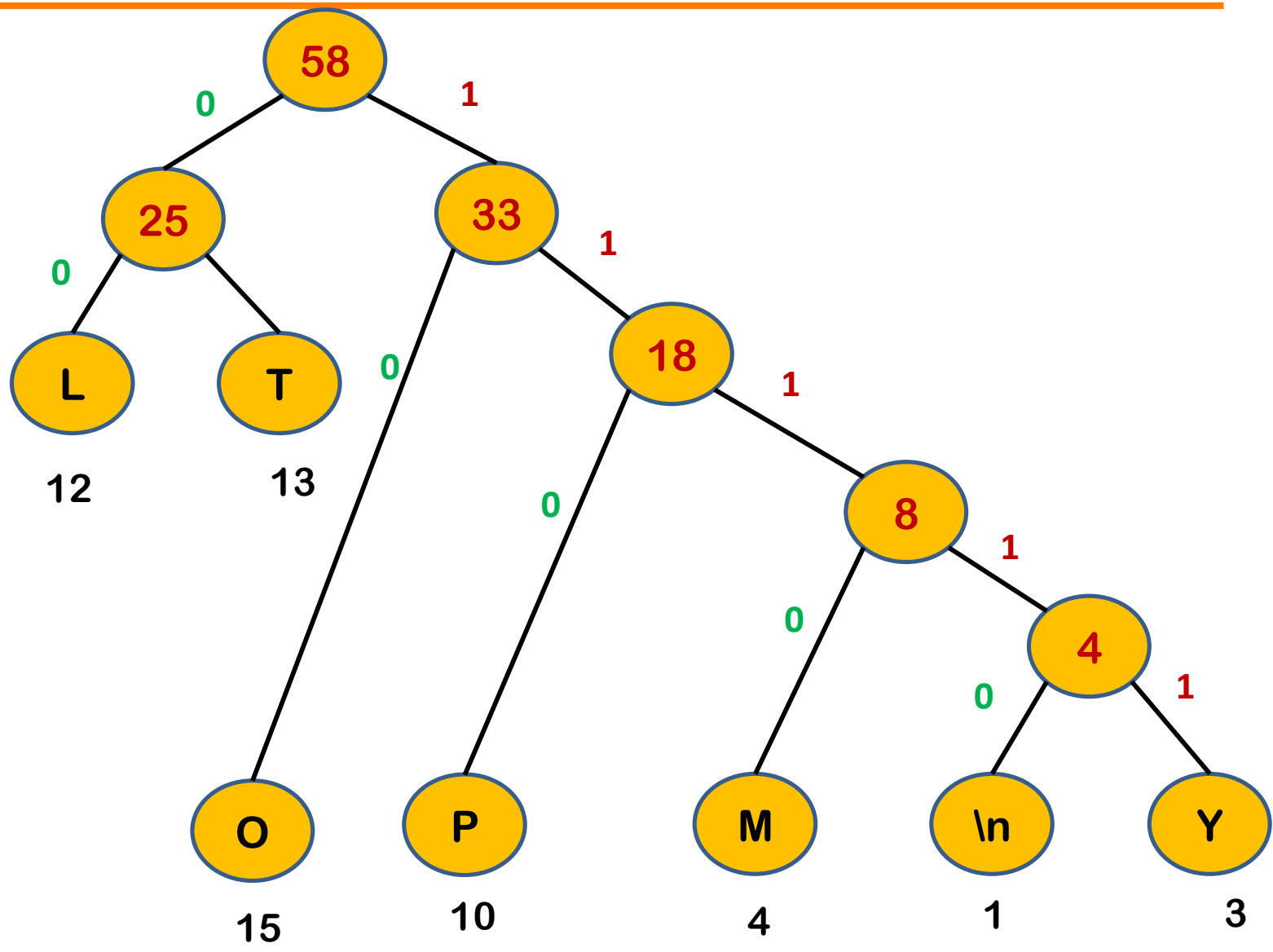
Caractère	Fréquence
P	10
O	15
L	12
Y	3
M	4
T	13
\n	1

- P: 110
- O: 10
- Y: 11111
- \n: 11110



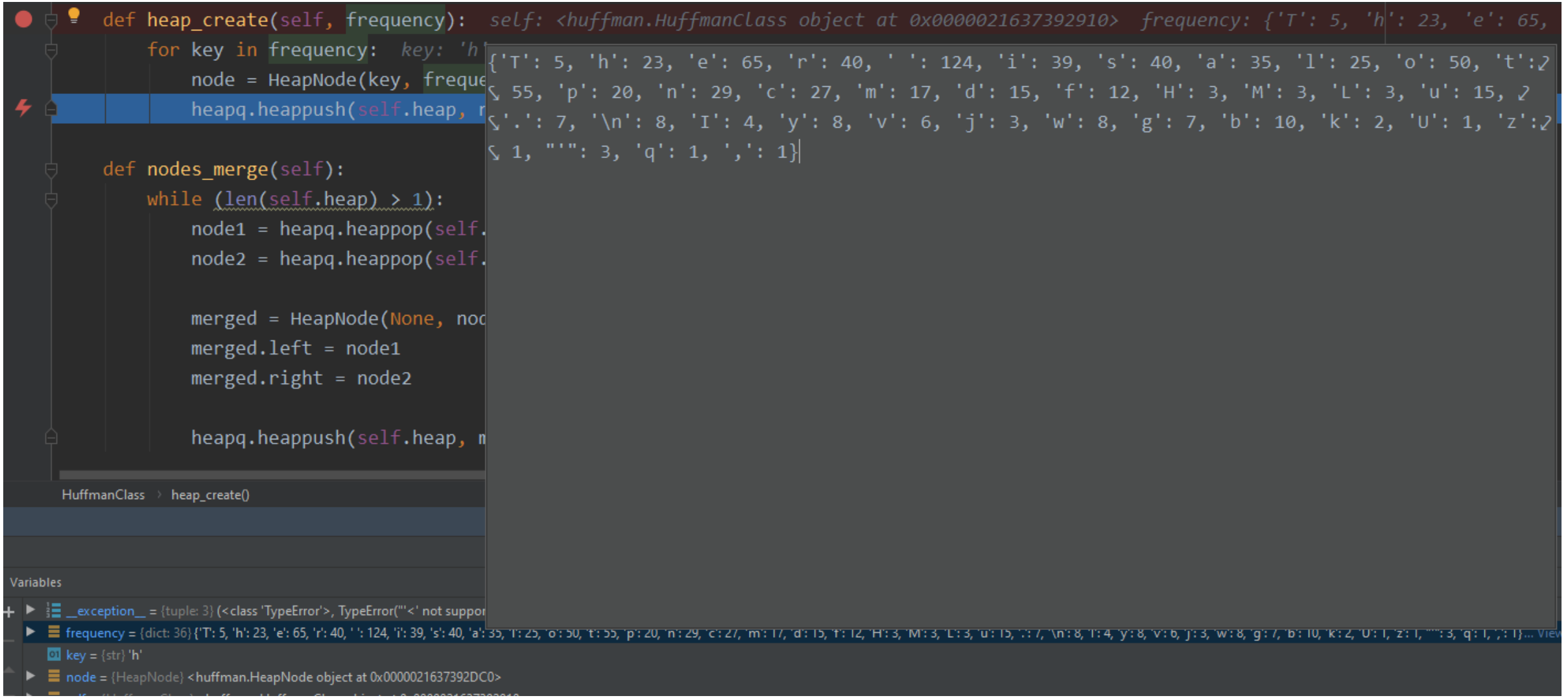
Exemple

Caractère	Code	Fréquence	Nb Bits
P	110	10	30
O	10	15	30
L	00	12	24
Y	1111	3	15
M	1110	4	16
T	01	13	26
\n	11110	1	5



- **Total = 146**
- Ancien total = 174

Code source

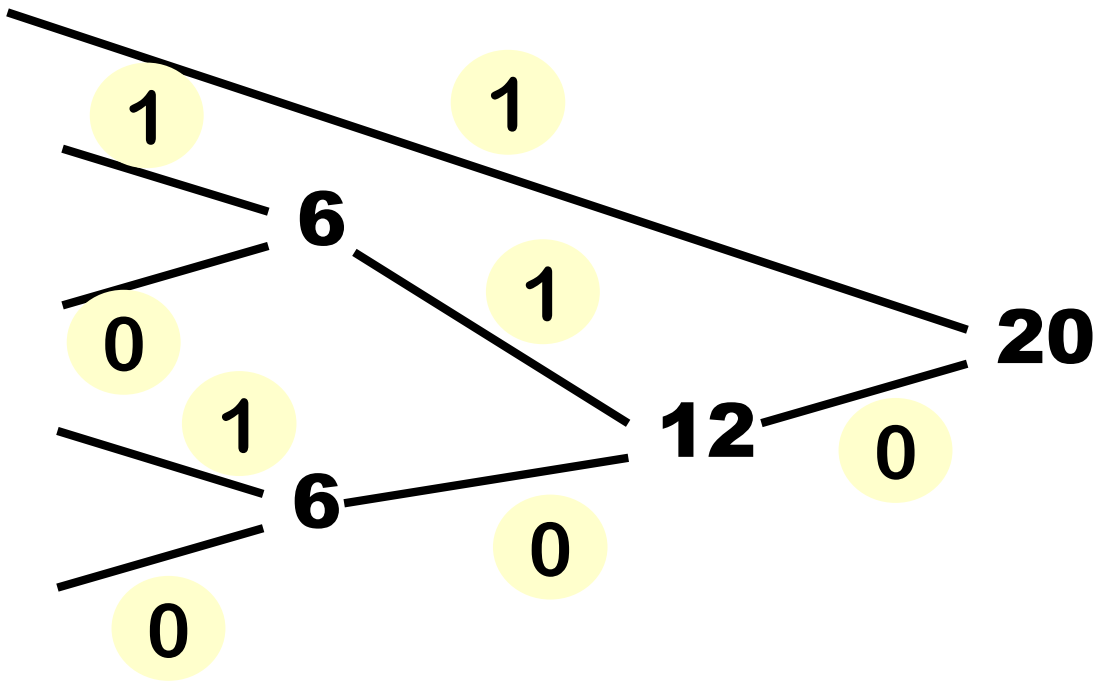

def heap_create(self, frequency):
for key in frequency:
node = HeapNode(key, freque
heapq.heappush(self.heap, r
def nodes_merge(self):
while (len(self.heap) > 1):
node1 = heapq.heappop(self.
node2 = heapq.heappop(self.
merged = HeapNode(None, noc
merged.left = node1
merged.right = node2
heapq.heappush(self.heap, m
HuffmanClass
heap_create()
Variables
__exception__ = {tuple: 3} (<class 'TypeError'>, TypeError("'<' not suppor
frequency = {dict: 36} {'T': 5, 'h': 23, 'e': 65, 'r': 40, ' ': 124, 'i': 39, 's': 40, 'a': 35, 'l': 25, 'o': 50, 't': 55, 'p': 20, 'n': 29, 'c': 27, 'm': 17, 'd': 15, 'f': 12, 'H': 3, 'M': 3, 'L': 3, 'u': 15, '.': 7, '\n': 8, 'I': 4, 'y': 8, 'v': 6, 'j': 3, 'w': 8, 'g': 7, 'b': 10, 'k': 2, 'U': 1, 'z': 1, "'": 3, 'q': 1, ',': 1}
key = {str} 'h'
node = {HeapNode} <huffman.HeapNode object at 0x0000021637392DC0>
self: <huffman.HuffmanClass object at 0x0000021637392910>
frequency: {'T': 5, 'h': 23, 'e': 65,
{'T': 5, 'h': 23, 'e': 65, 'r': 40, ' ': 124, 'i': 39, 's': 40, 'a': 35, 'l': 25, 'o': 50, 't': 55, 'p': 20, 'n': 29, 'c': 27, 'm': 17, 'd': 15, 'f': 12, 'H': 3, 'M': 3, 'L': 3, 'u': 15, '.': 7, '\n': 8, 'I': 4, 'y': 8, 'v': 6, 'j': 3, 'w': 8, 'g': 7, 'b': 10, 'k': 2, 'U': 1, 'z': 1, "'": 3, 'q': 1, ',': 1}

Exemple Codage

- Ex. Polmmt Polmmt Polmmmmt
- M: 8
- P: 3
- O: 3
- L: 3
- T: 3

Exemple Codage

- Ex. Polmmt Polmmt Polmmmmt
- M: 8
- P: 3
- O: 3
- L: 3
- T: 3



- M: 8 -- » 1
- P: 3 -- » 011
- O: 3 -- » 010
- L: 3 -- » 001
- T: 3 -- » 000

Optimisation

[Polmmt Polmmt Polmmmmmt]

- M: 8 -- » 1
- P: 3 -- » 011
- O: 3 -- » 010
- L: 3 -- » 001
- T: 3 -- » 000

- Performance du code Huffman
- Initial : 8 * 1 + 4 * 3 * 3 = 44 bits
- La longueur de la chaine = 20 chars

Performance = 44 / 20 = 2.2

À vous !!!

C’est fini !!!

Merci !

Questions?

- noureddine.kerzazi@polymtl.ca

Remise .Zip bien commenté d’ici 2 semaines