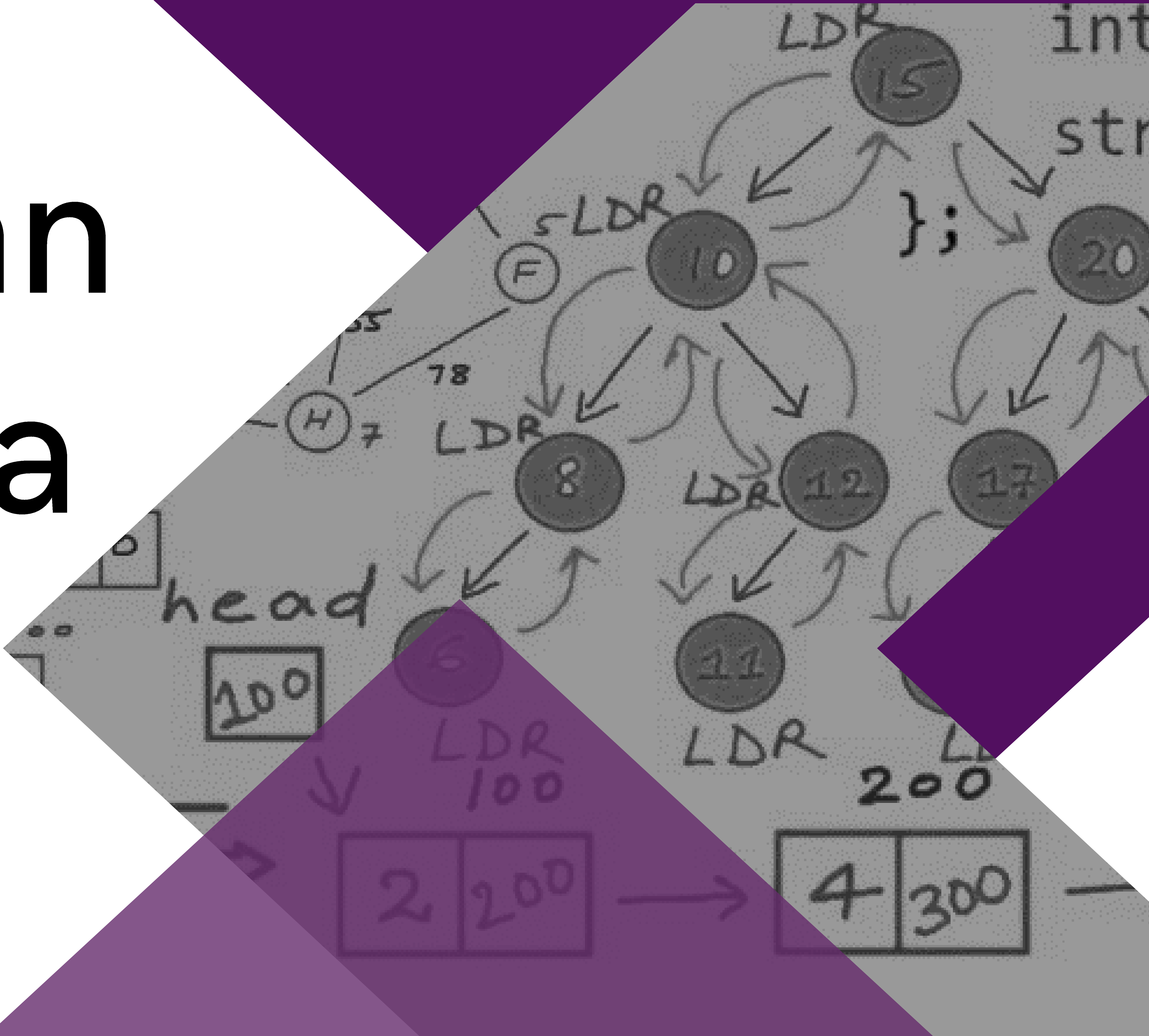
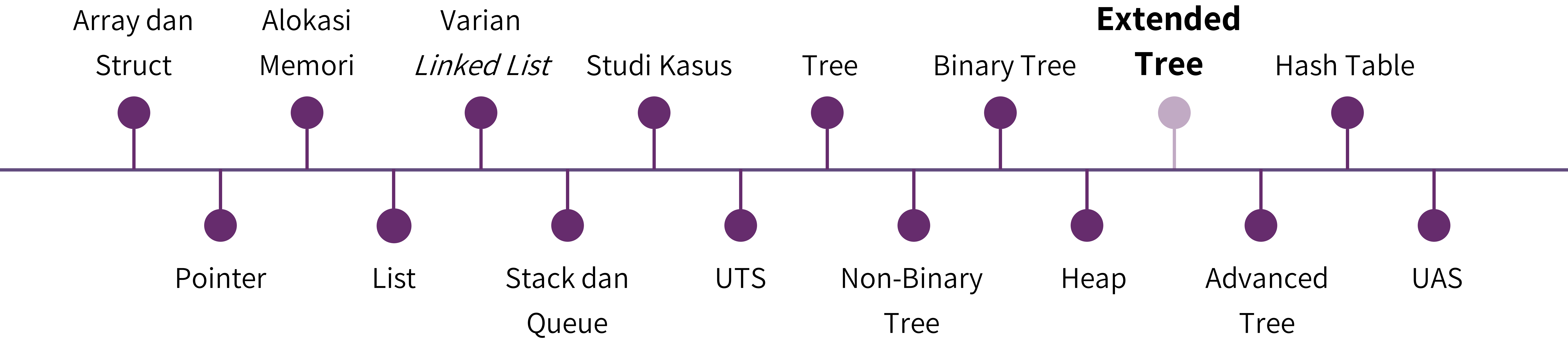
Algoritma dan Struktur Data

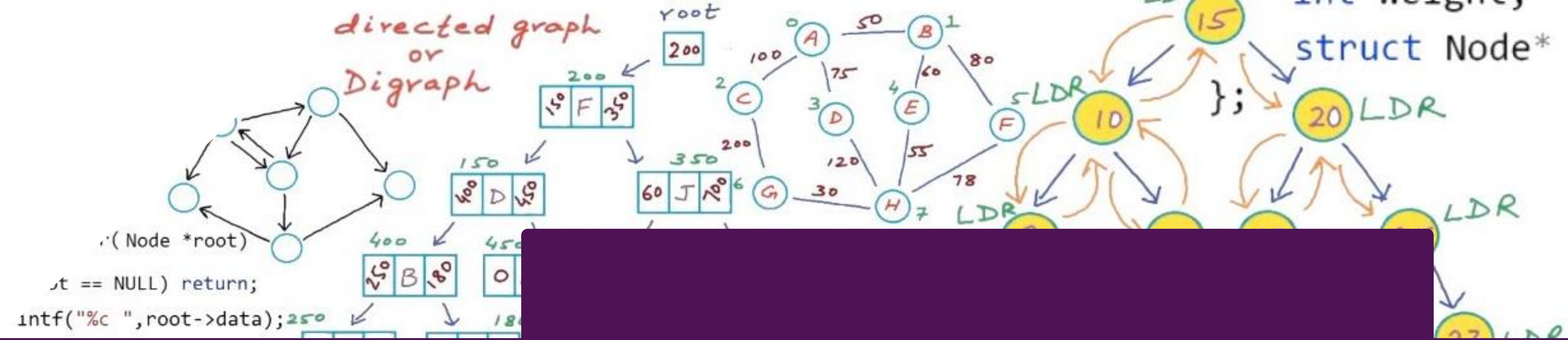


Pekan 13



Array dan Struct
Pointer
Alokasi Memori
List
Varian *Linked List*
Stack dan Queue
Studi Kasus
UTS
Tree
Non-Binary Tree
Binary Tree
Heap
Extended Tree
Advanced Tree
Hash Table
UAS

Tujuan



1	Mahasiswa memahami fitur struktural dari Huffman tree
2	Mahasiswa mampu membedakan BST, heap, Huffman tree
3	Mahasiswa mampu membuat Huffman tree, Huffman code, dan melakukan pencarian pada Huffman tree

Text Compression

ASCII Coding Scheme

Letter	ASCII Code	Binary	Letter	ASCII Code	Binary
a	097	01100001	A	065	01000001
b	098	01100010	B	066	01000010
c	099	01100011	C	067	01000011
d	100	01100100	D	068	01000100
e	101	01100101	E	069	01000101
f	102	01100110	F	070	01000110
g	103	01100111	G	071	01000111
h	104	01101000	H	072	01001000
i	105	01101001	I	073	01001001
j	106	01101010	J	074	01001010
k	107	01101011	K	075	01001011
l	108	01101100	L	076	01001100
m	109	01101101	M	077	01001101
n	110	01101110	N	078	01001110
o	111	01101111	O	079	01001111
p	112	01110000	P	080	01010000
q	113	01110001	Q	081	01010001
r	114	01110010	R	082	01010010
s	115	01110011	S	083	01010011
t	116	01110100	T	084	01010100
u	117	01110101	U	085	01010101
v	118	01110110	V	086	01010110
w	119	01110111	W	087	01010111
x	120	01111000	X	088	01011000
y	121	01111001	Y	089	01011001
z	122	01111010	Z	090	01011010

The standard ASCII coding scheme assigns a unique eight-bit value to each character.

ASCII Coding Scheme

Letter	ASCII Code	Binary	Letter	ASCII Code	Binary
a	097	01100001	A	065	01000001
b	098	01100010	B	066	01000010
c	099	01100011	C	067	01000011
d	100	01100100	D	068	01000100
e	101	01100101	E	069	01000101
f	102	01100110	F	070	01000110
g	103	01100111	G	071	01000111
h	104	01101000	H	072	01001000
i	105	01101001	I	073	01001001
j	106	01101010	J	074	01001010
k	107	01101011	K	075	01001011
l	108	01101100	L	076	01001100
m	109	01101101	M	077	01001101
n	110	01101110	N	078	01001110
o	111	01101111	O	079	01001111
p	112	01110000	P	080	01010000
q	113	01110001	Q	081	01010001
r	114	01110010	R	082	01010010
s	115	01110011	S	083	01010011
t	116	01110100	T	084	01010100
u	117	01110101	U	085	01010101
v	118	01110110	V	086	01010110
w	119	01110111	W	087	01010111
x	120	01111000	X	088	01011000
y	121	01111001	Y	089	01011001
z	122	01111010	Z	090	01011010

It takes a certain minimum number of bits to provide unique codes for each character.

$\log_2 128$= 7 bits to provide the 128 unique codes

The ASCII Standard is 8 bits, the 8th bit is used either to check for transmission errors, or to support “extended” ASCII codes

ASCII Coding Scheme

Letter	ASCII Code	Binary	Letter	ASCII Code	Binary
a	097	01100001	A	065	01000001
b	098	01100010	B	066	01000010
c	099	01100011	C	067	01000011
d	100	01100100	D	068	01000100
e	101	01100101	E	069	01000101
f	102	01100110	F	070	01000110
g	103	01100111	G	071	01000111
h	104	01101000	H	072	01001000
i	105	01101001	I	073	01001001
j	106	01101010	J	074	01001010
k	107	01101011	K	075	01001011
l	108	01101100	L	076	01001100
m	109	01101101	M	077	01001101
n	110	01101110	N	078	01001110
o	111	01101111	O	079	01001111
p	112	01110000	P	080	01010000
q	113	01110001	Q	081	01010001
r	114	01110010	R	082	01010010
s	115	01110011	S	083	01010011
t	116	01110100	T	084	01010100
u	117	01110101	U	085	01010101
v	118	01110110	V	086	01010110
w	119	01110111	W	087	01010111
x	120	01111000	X	088	01011000
y	121	01111001	Y	089	01011001
z	122	01111010	Z	090	01011010

It takes a certain minimum number of bits to provide unique codes for each character.

$\log_2 128$= 7 bits to provide the 128 unique codes

The ASCII Standard is 8 bits, the 8th bit is used either to check for transmission errors, or to support “extended” ASCII codes

Fixed-Length Coding Scheme

$\log_2 n$ bits to represent n unique code values assumes that all codes will be the **same length**

If all characters were used equally often, then a fixed-length coding scheme is the most space efficient method.

Variable-Length Codes

- Not all characters are used equally often in many applications.
- Assign shorter codes for some characters that are used more frequently than others.
- Basic concept of file compression techniques in common use today.

Huffman Coding Tree

Huffman Coding Tree

Huffman coding assigns codes to characters such that the length of the code depends on **the relative frequency** or **weight** of the corresponding character.

If the estimated frequencies for letters match the actual frequency found in an encoded message, then the length of that message will typically be less than if a fixed-length code had been used.

Huffman Coding Tree

- The Huffman code for each letter is derived from a **full binary tree** called the **Huffman coding tree**, or simply the **Huffman tree**.
- Each leaf of the Huffman tree corresponds to a letter
- The weight of the leaf node is the weight (frequency) of its associated letter.
- The goal is to build a tree with **the minimum external path weight**.
- Define **the weighted path length** of a leaf to be its weight times its depth.
- The binary tree with minimum external path weight is the one with the **minimum sum of weighted path lengths** for the given set of leaves.

Building Huffman Coding Tree

Building Huffman Tree

1. Create a collection of n initial Huffman trees, each of which is a single leaf node containing one of the letters.
2. Put the n partial trees onto a priority queue organized by weight (frequency).
3. Remove the first two trees (the ones with lowest weight) from the priority queue.
4. Join these two trees together to create a new tree whose root has the two trees as children, and whose weight is the sum of the weights of the two trees.
5. Put this new tree back into the priority queue.
6. This process is repeated until all of the partial Huffman trees have been combined into one.

Building Huffman Tree

Letter	C	D	E	K	L	M	U	Z
Frequency	32	42	120	7	42	24	37	2

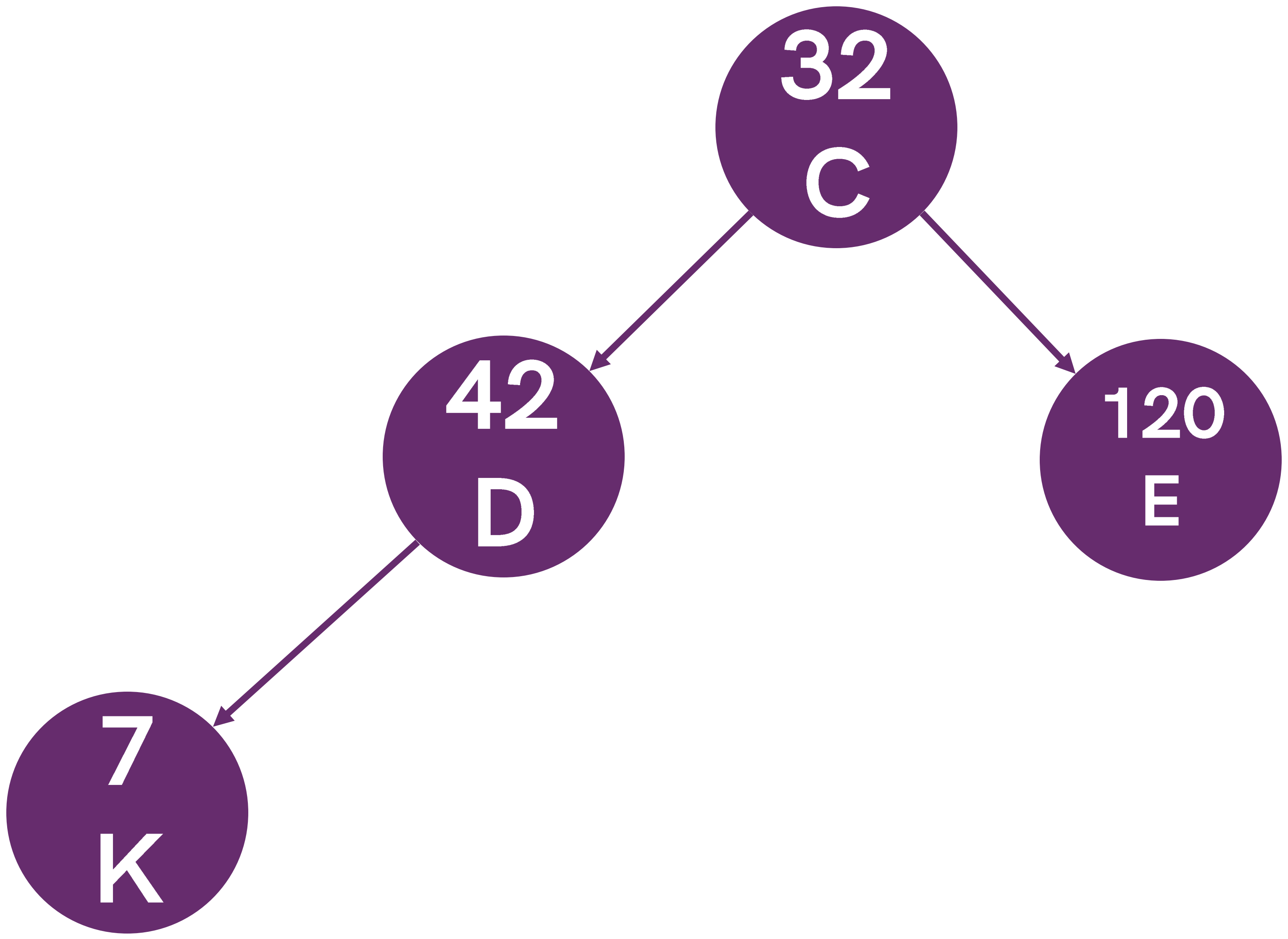
Building Huffman Tree

Step 1

32	42	120	7	42	24	37	2
C	D	E	K	L	M	U	Z

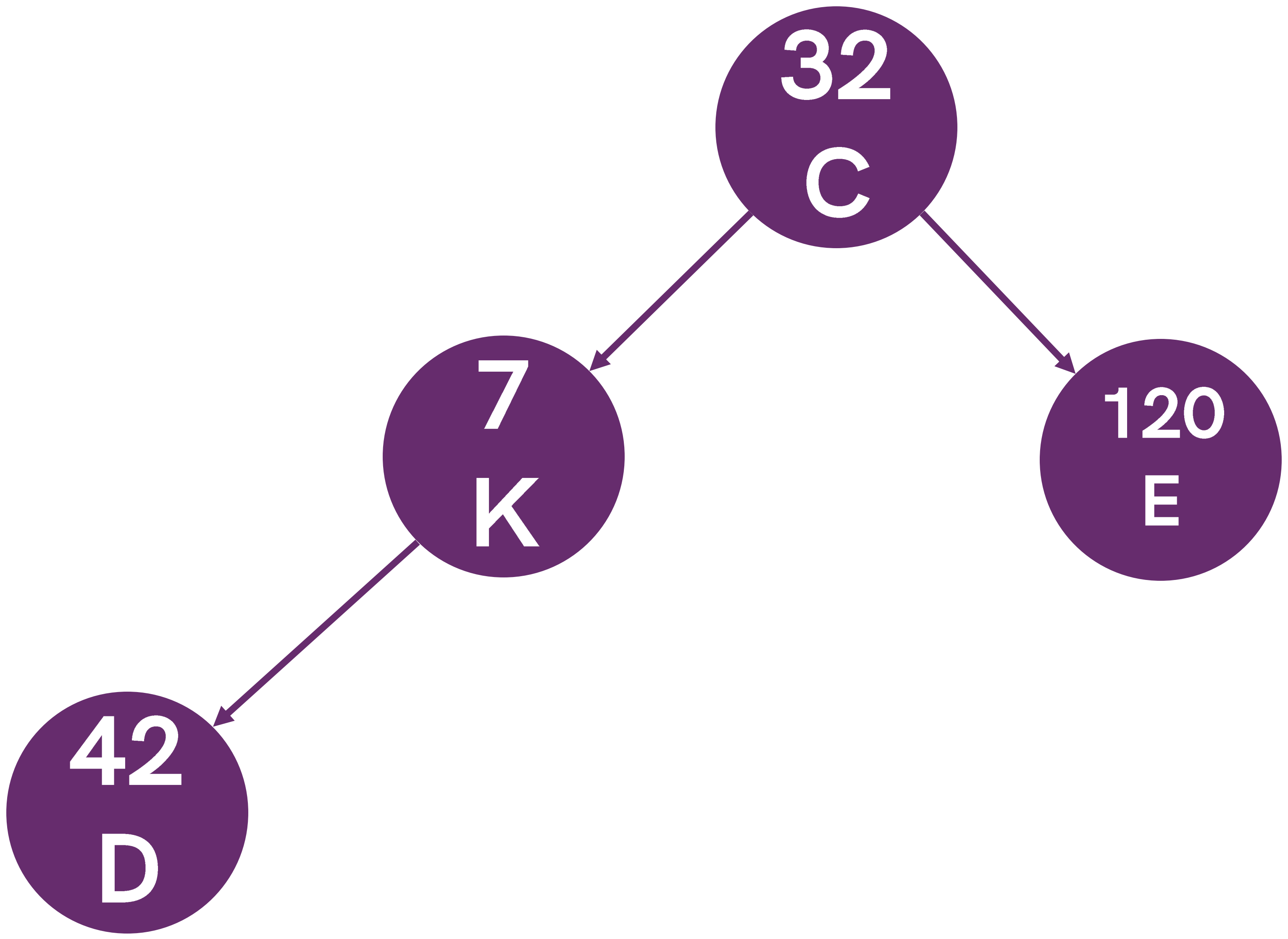
Building Huffman Tree

Step 2



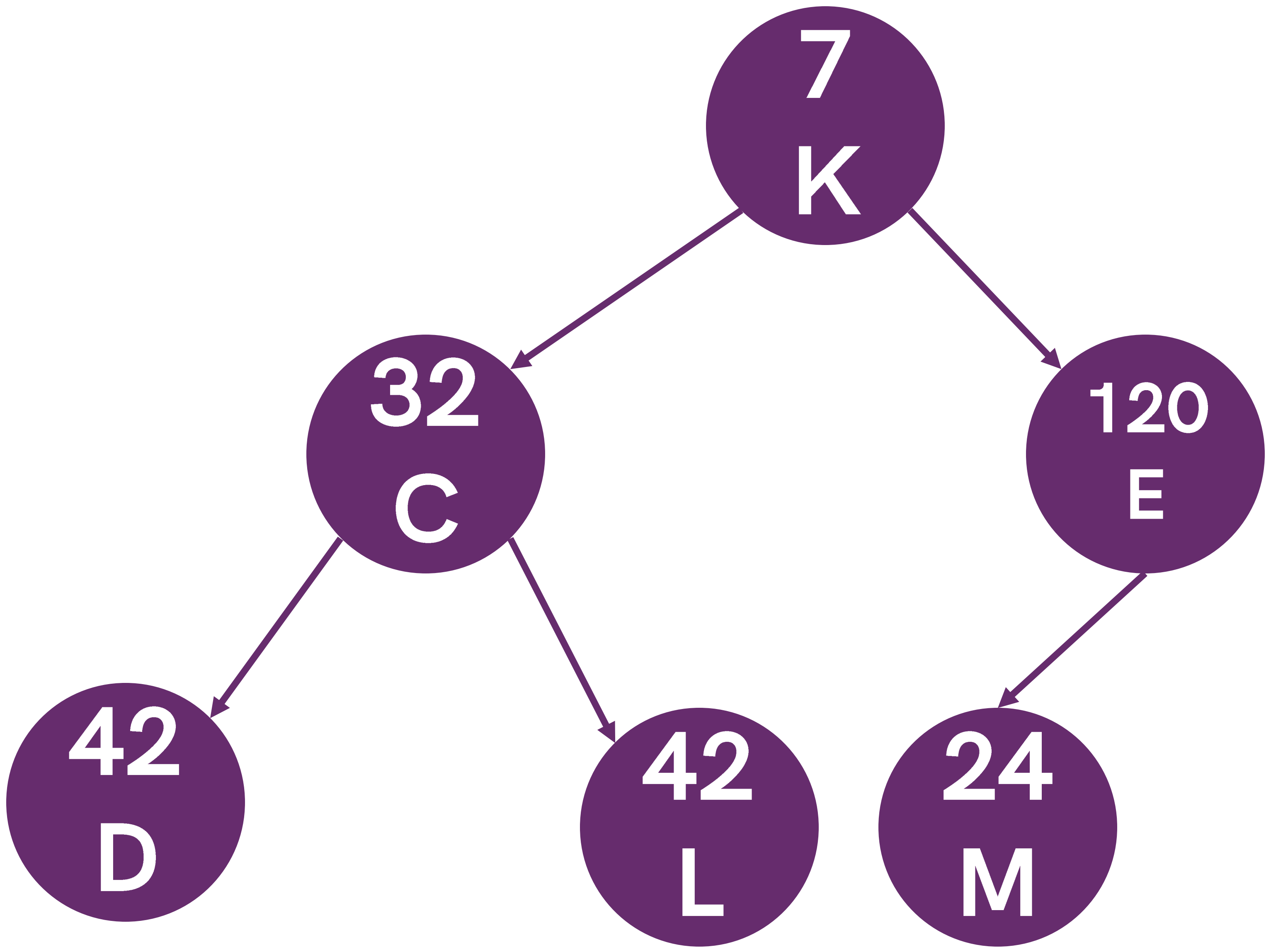
Building Huffman Tree

Step 2

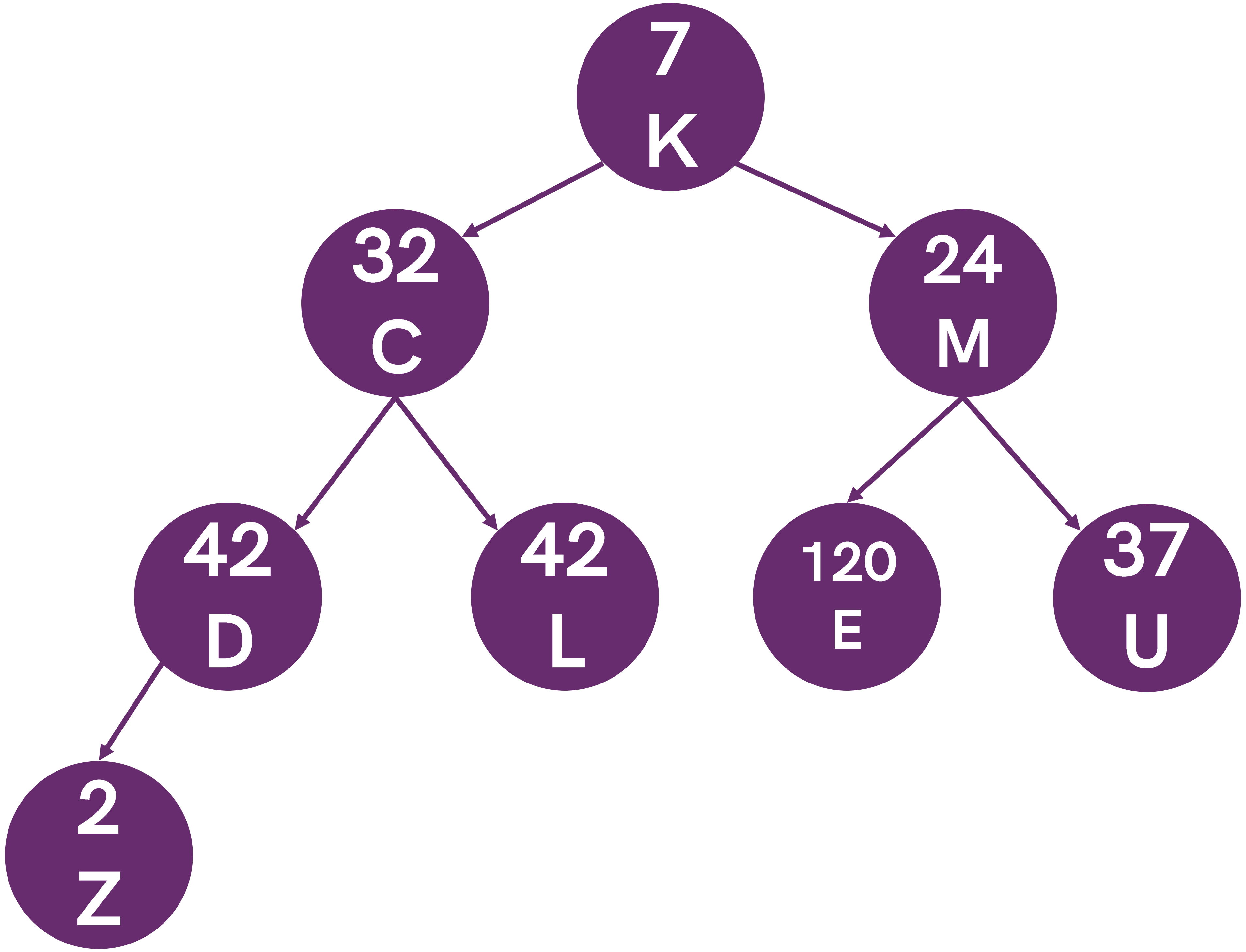


Building Huffman Tree

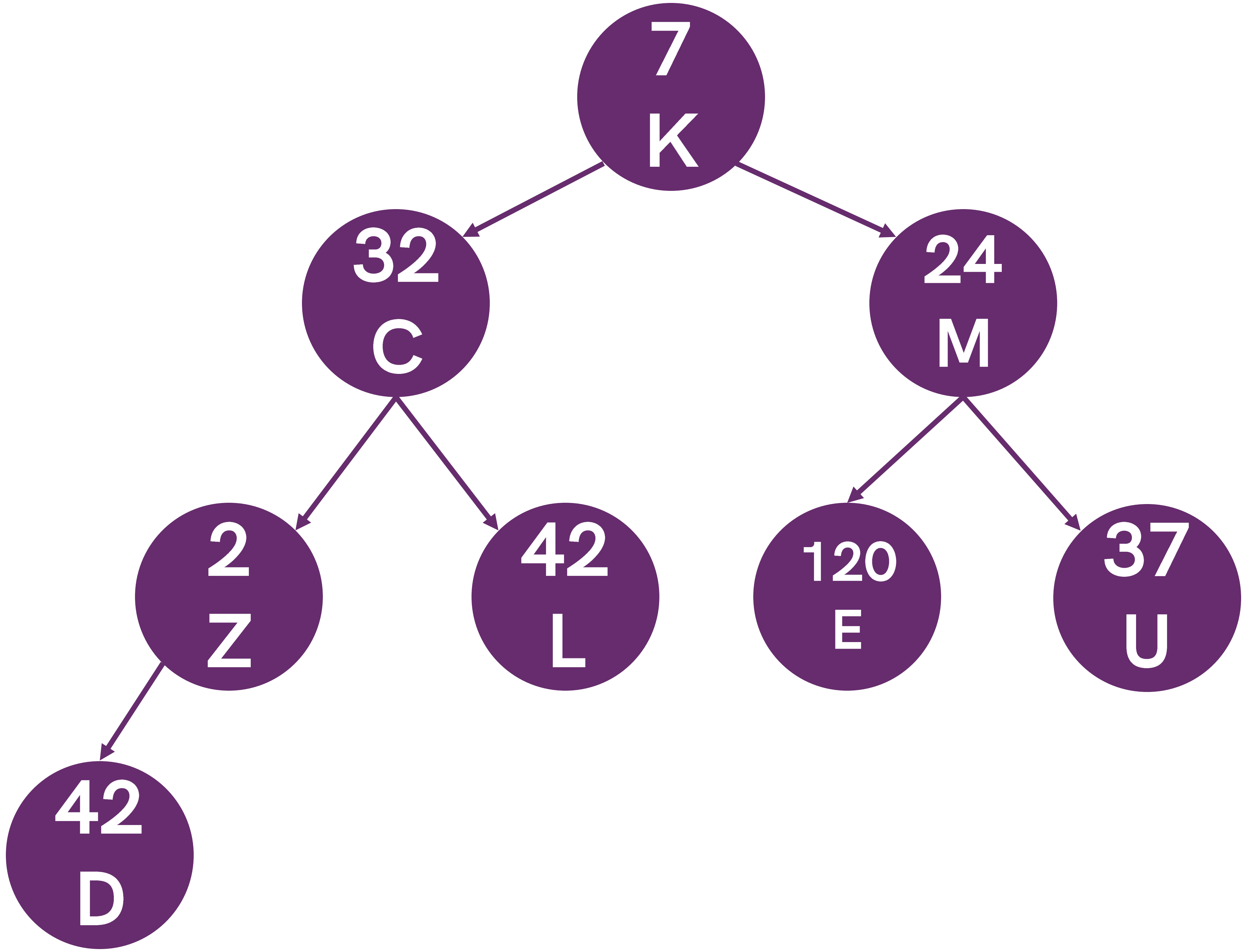
Step 2



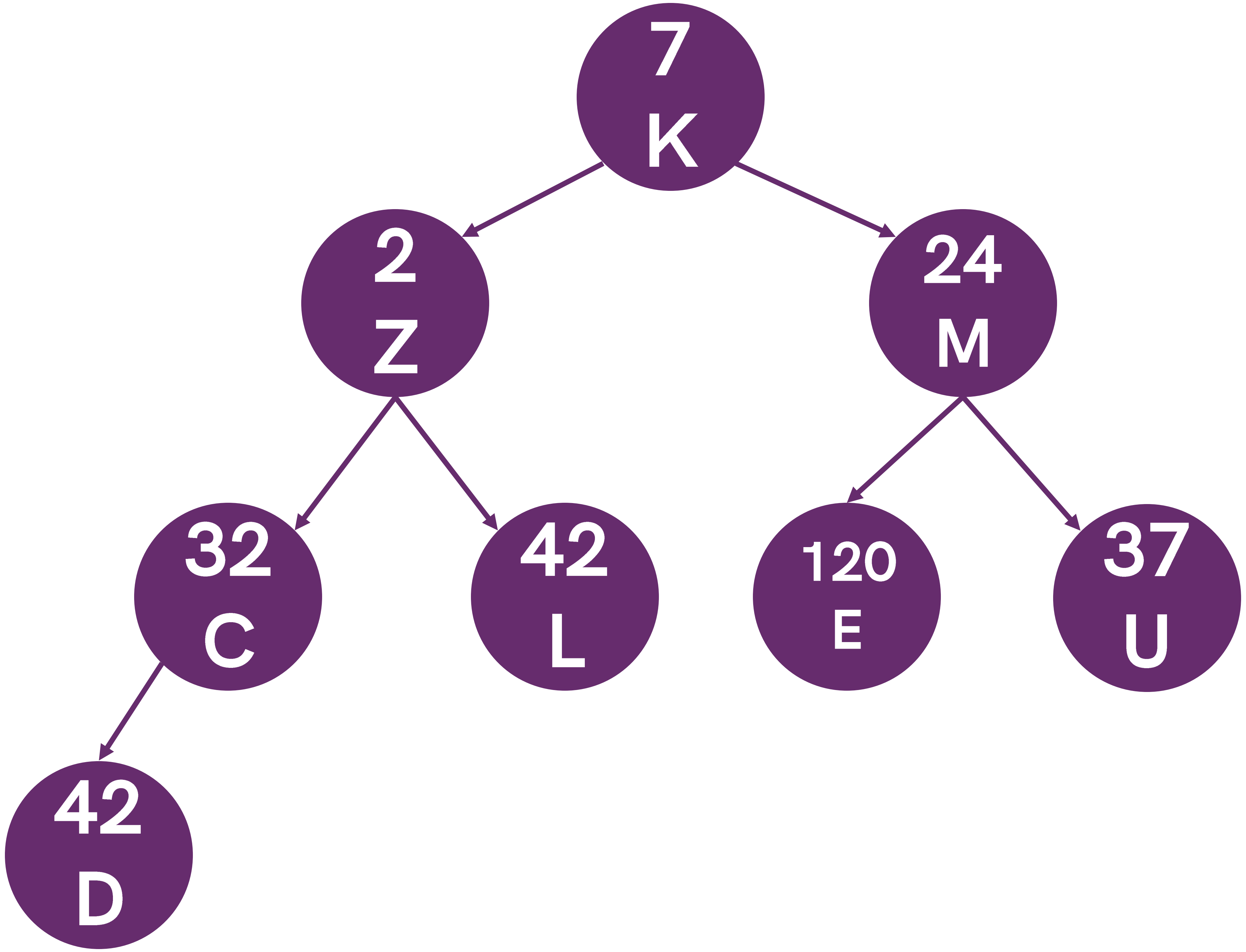
Building Huffman Tree
Step 2



Building Huffman Tree
Step 2

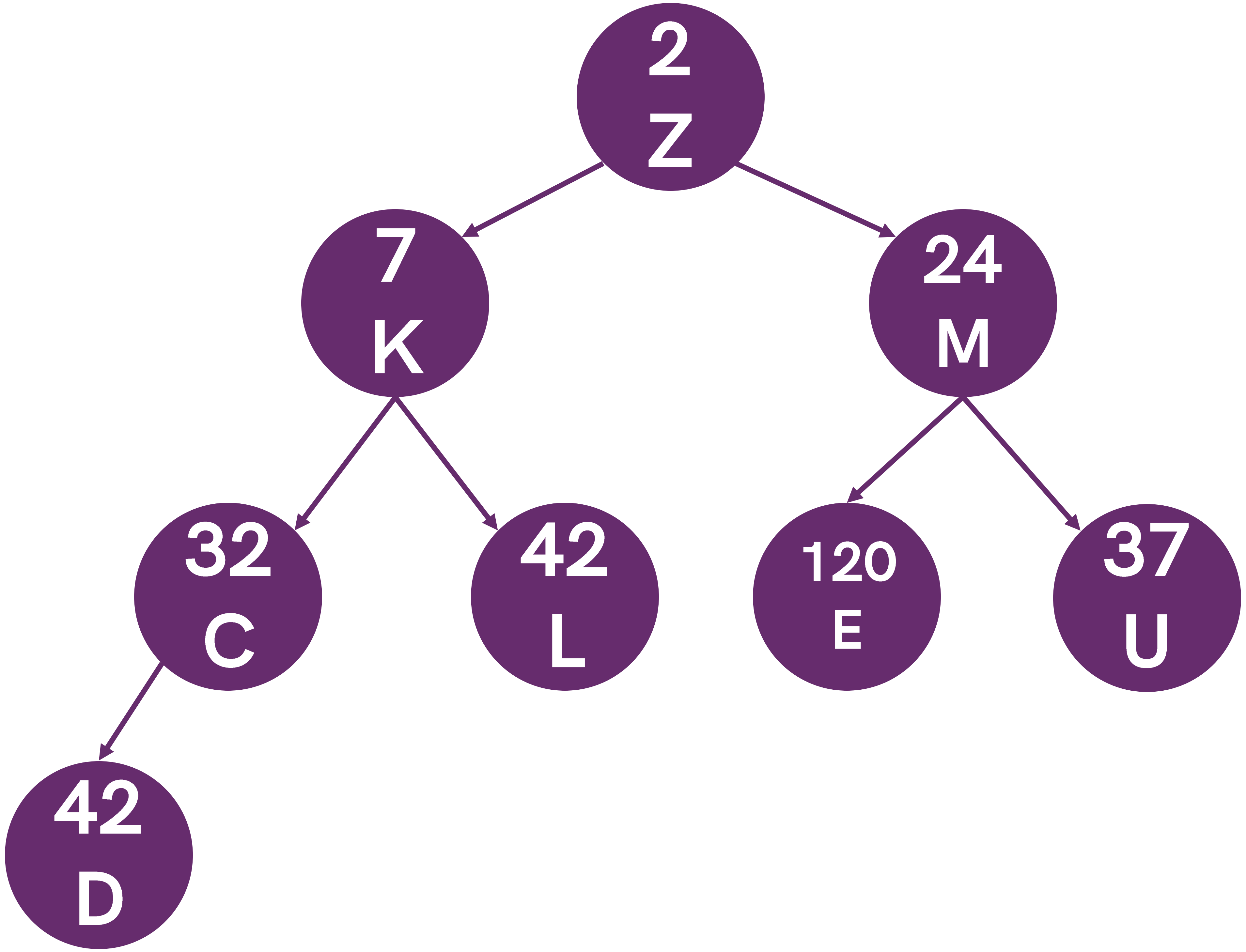


Building Huffman Tree
Step 2

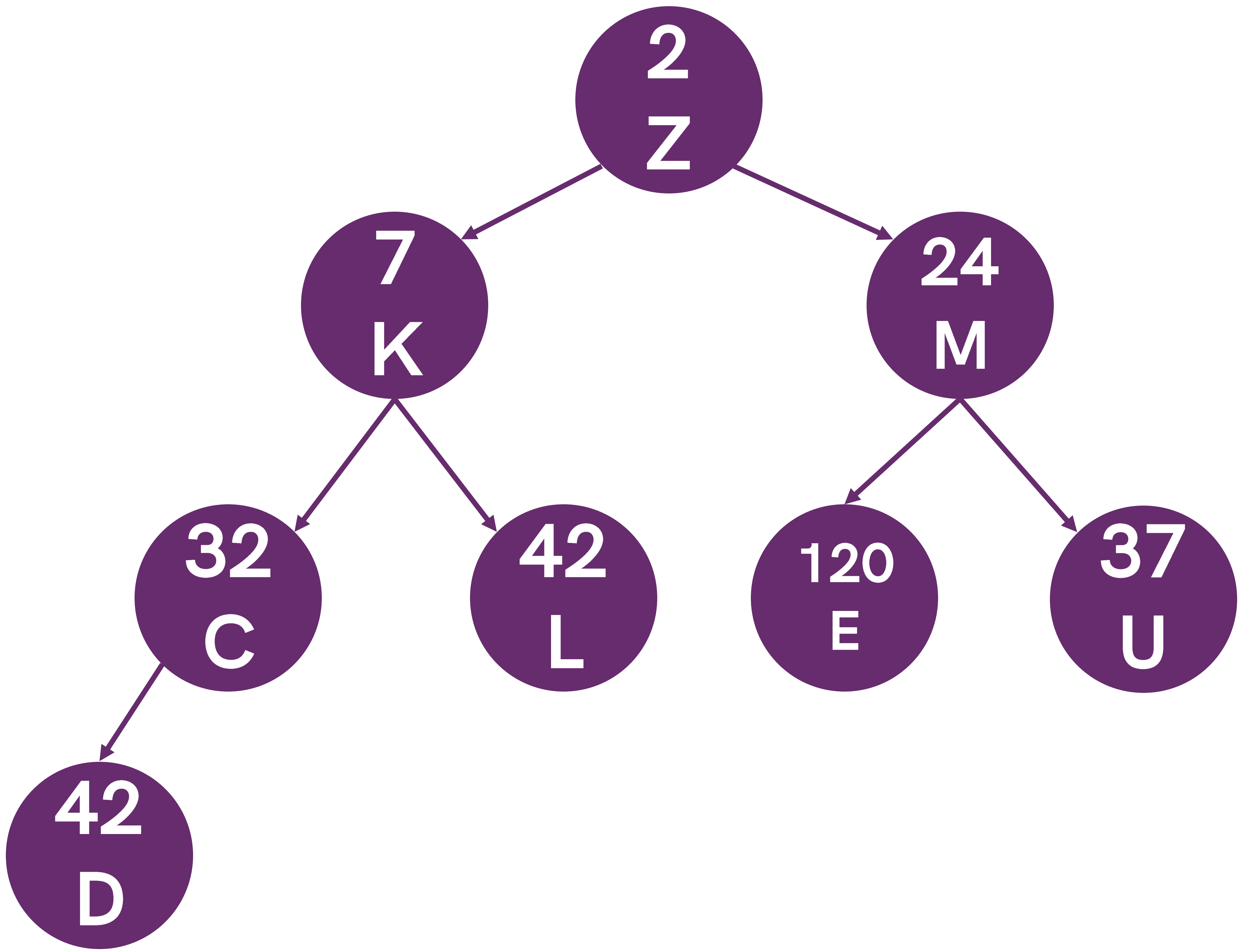


Building Huffman Tree

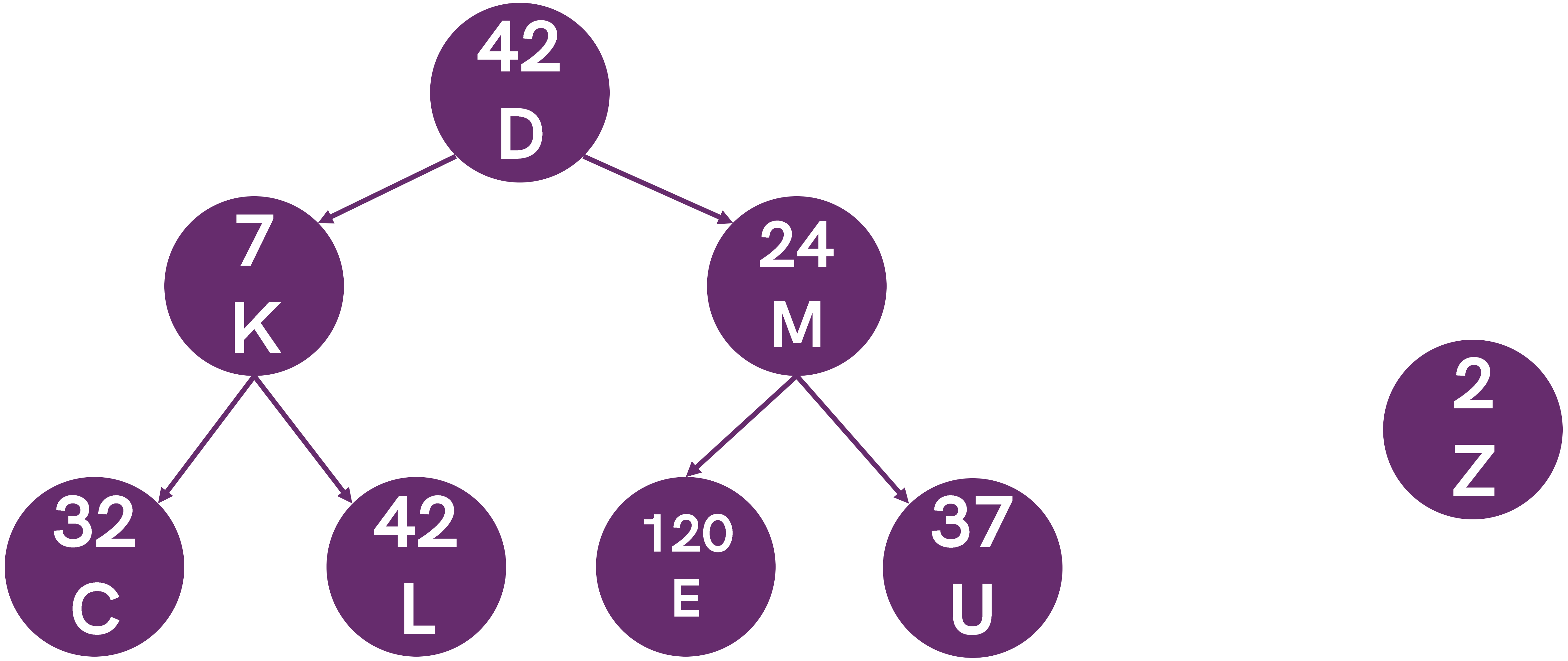
Step 2



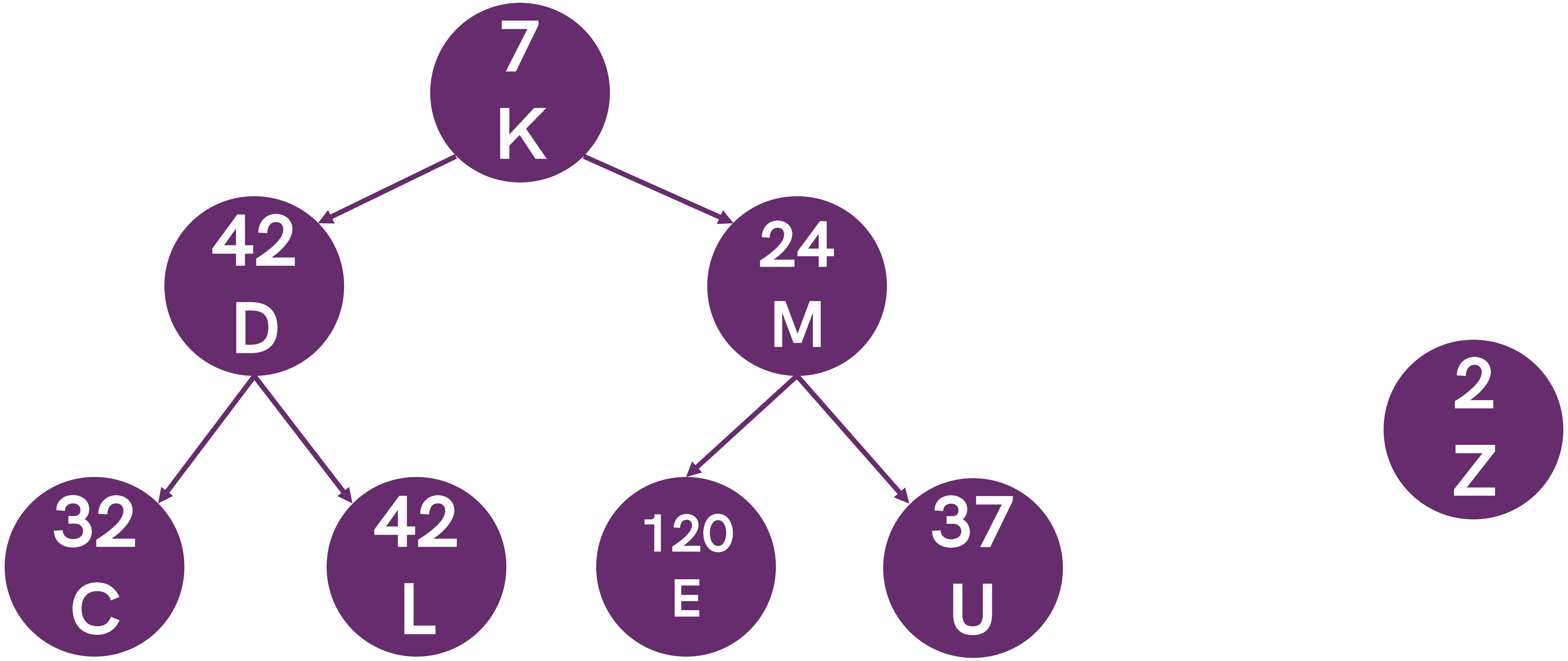
Building Huffman Tree
Step 3



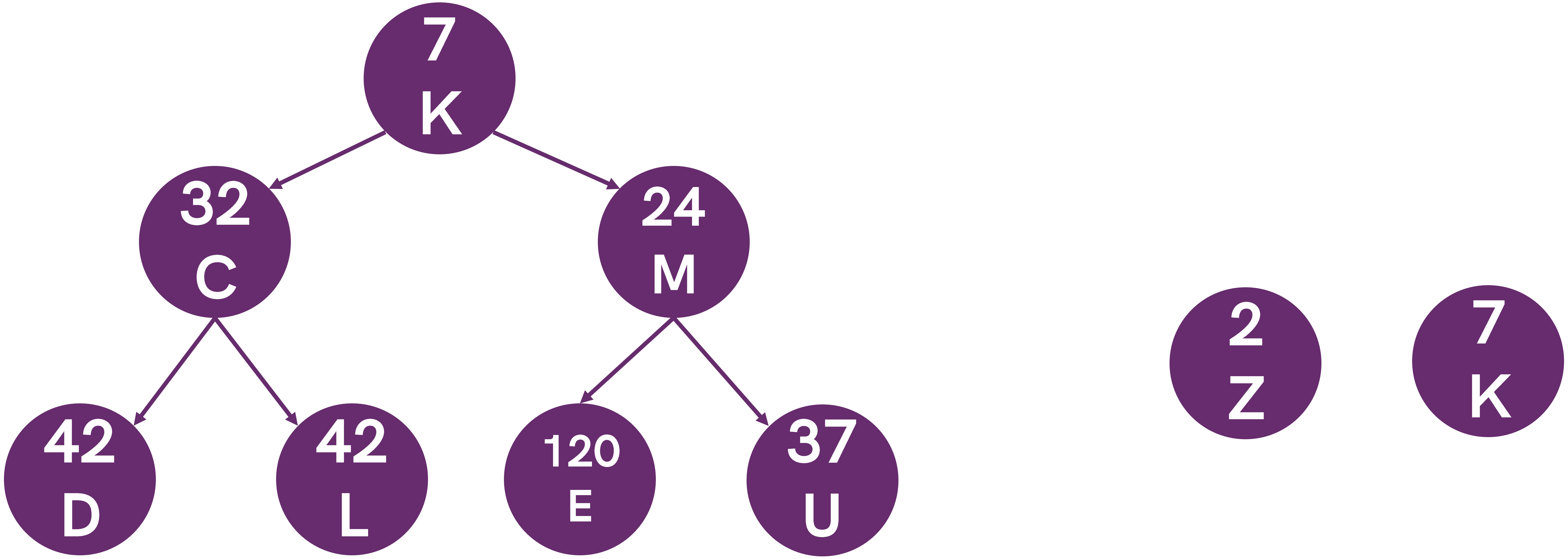
Building Huffman Tree
Step 3



Building Huffman Tree
Step 3

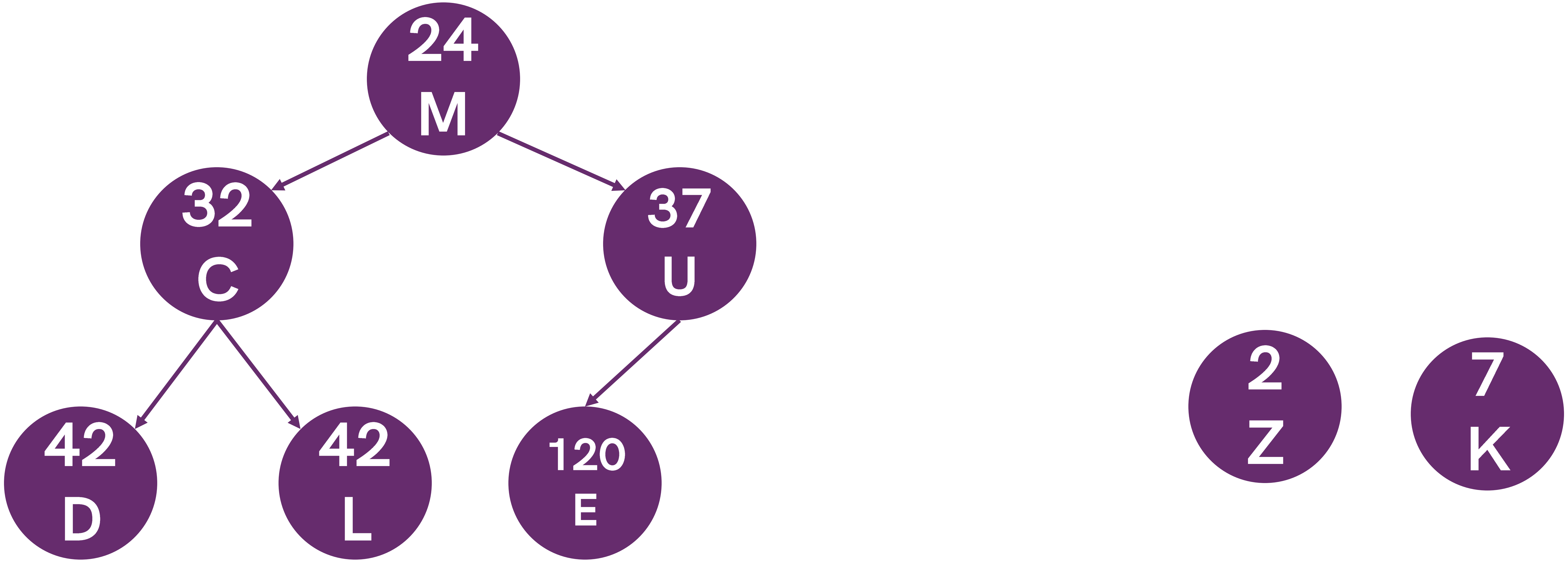


Building Huffman Tree
Step 3



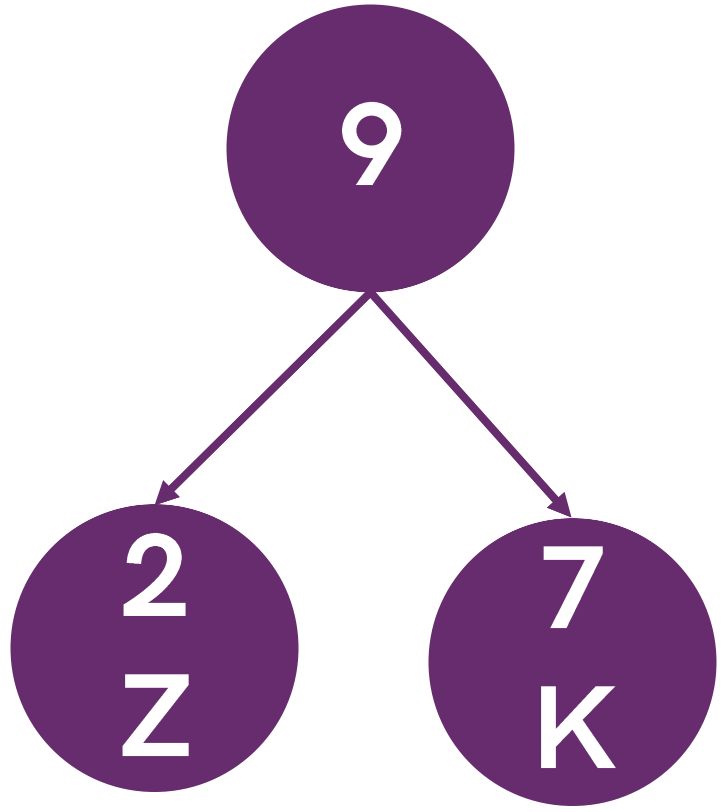
Building Huffman Tree

Step 3



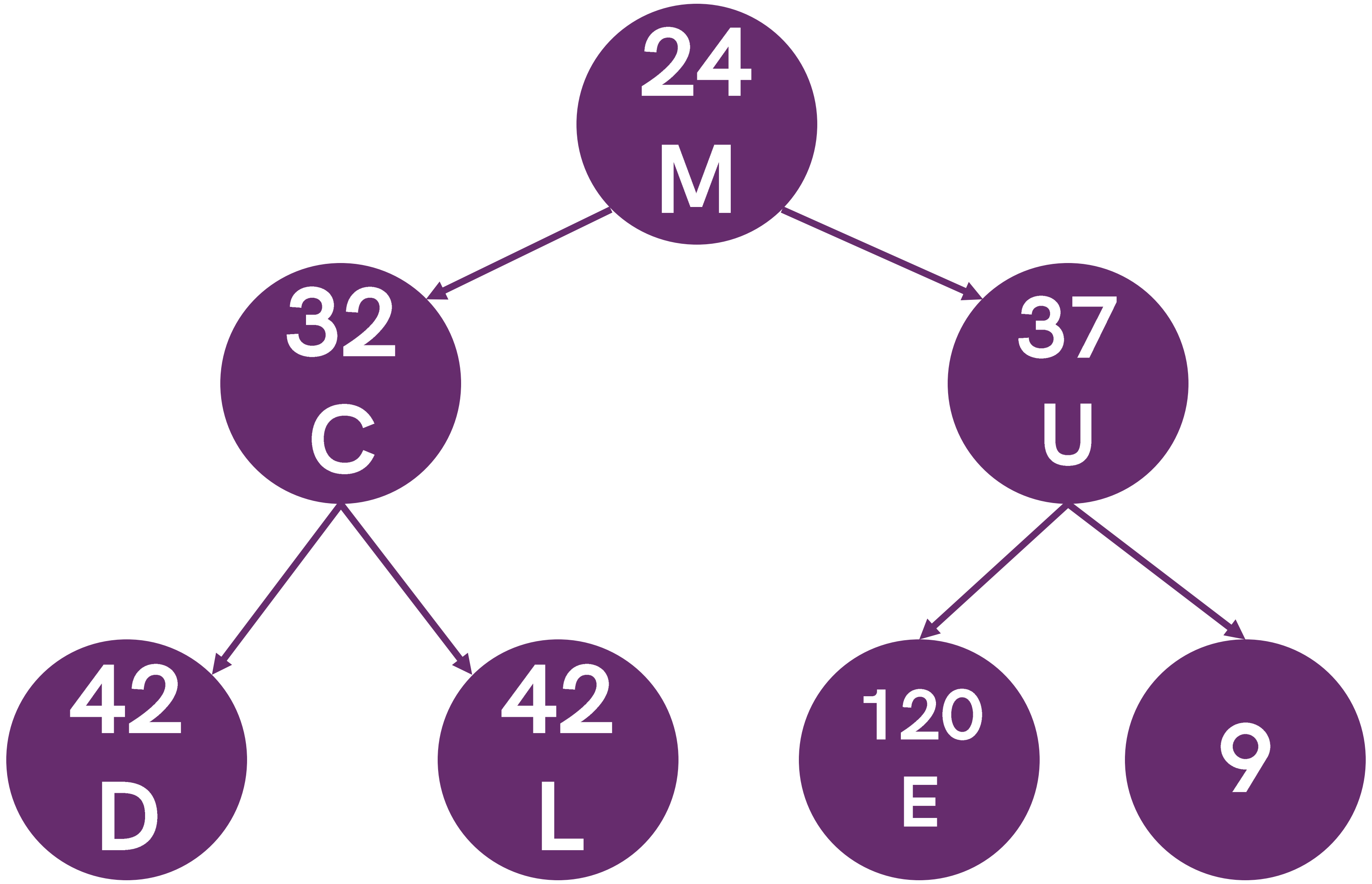
Building Huffman Tree

Step 4



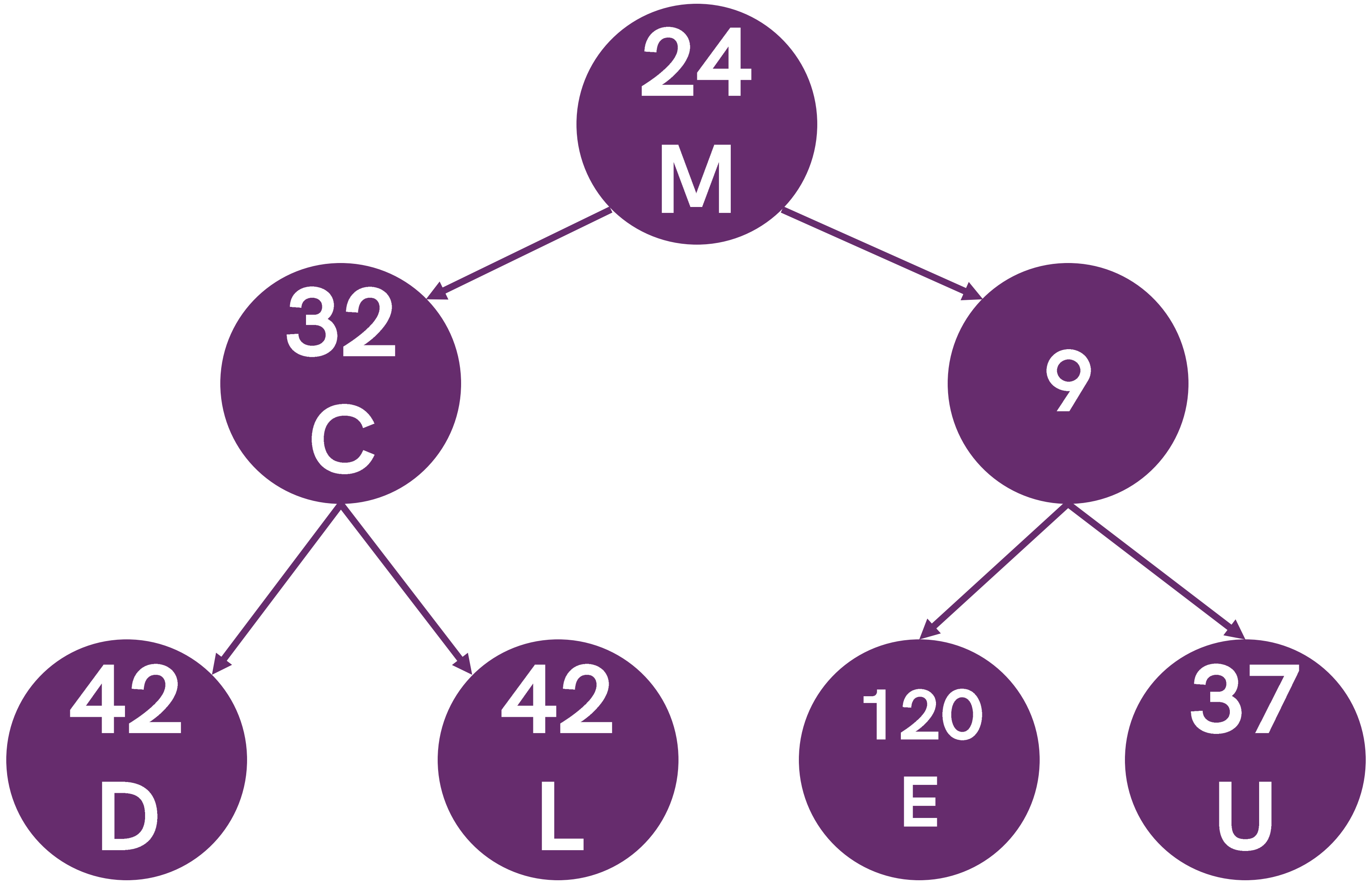
Building Huffman Tree

Step 5

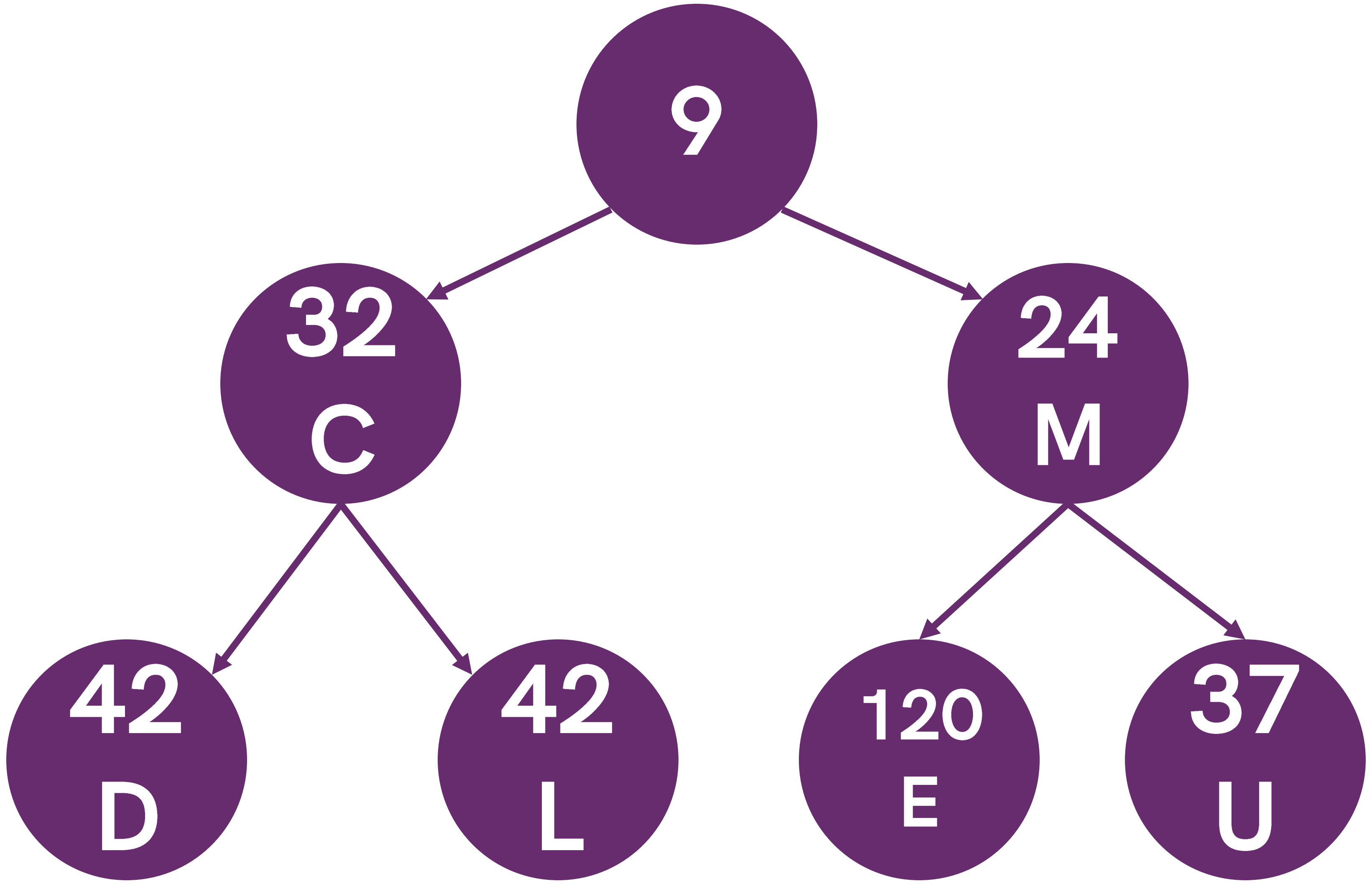


Building Huffman Tree

Step 5

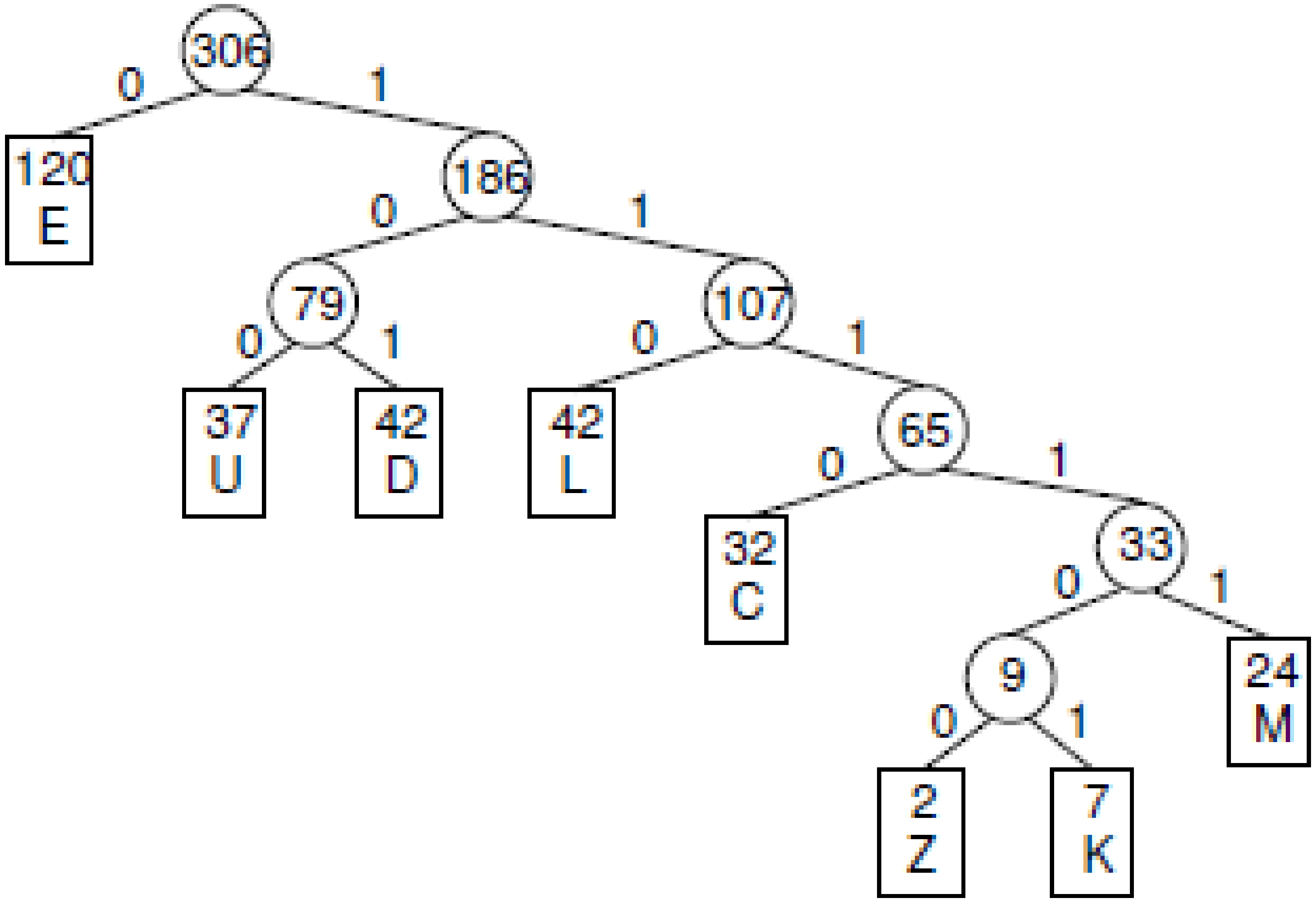


Building Huffman Tree
Step 5



Assigning Huffman Codes

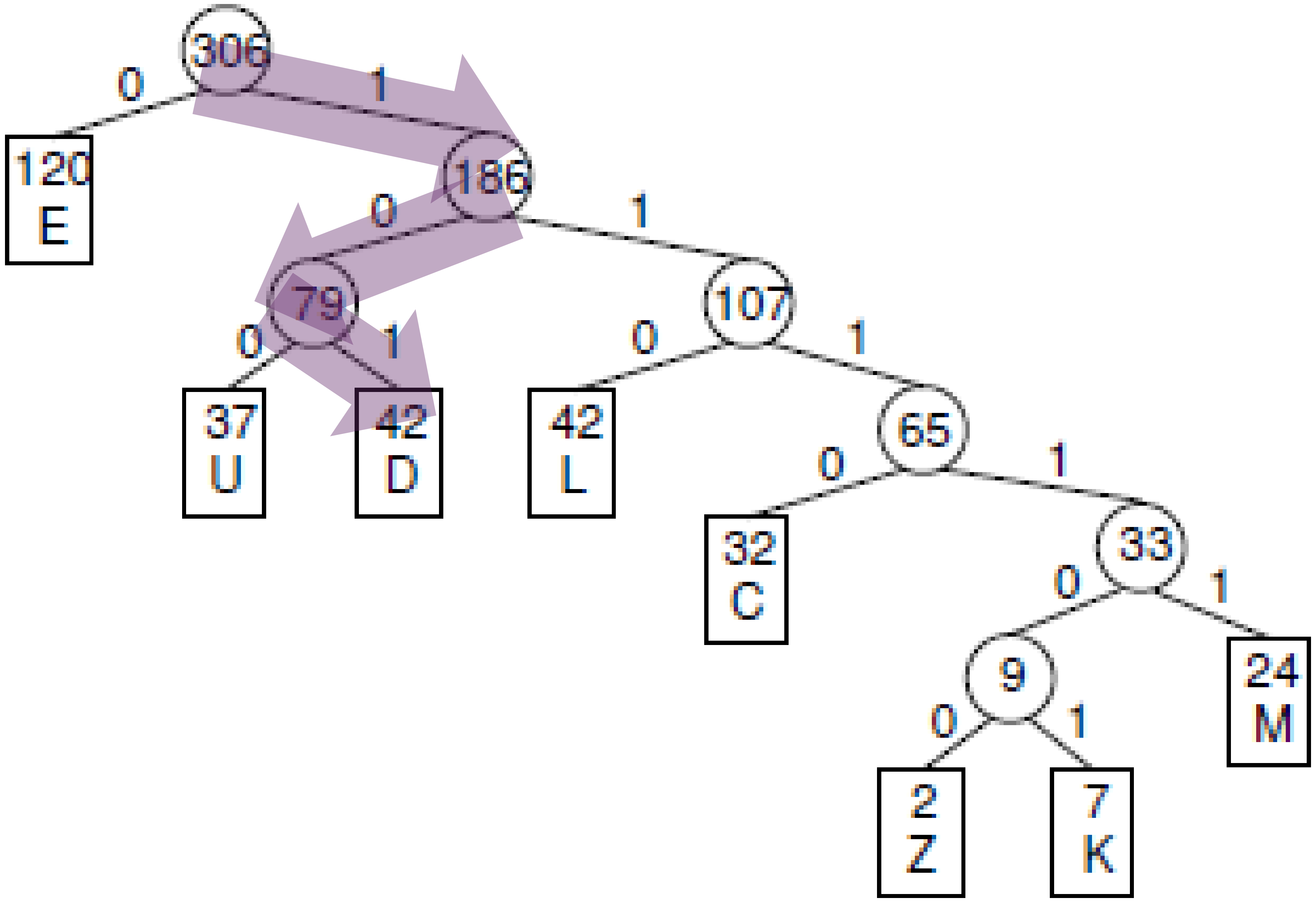
Assigning Huffman Codes



Letter	Freq	Code	Bits
C	32	1110	4
D	42	101	3
E	120	0	1
K	7	111101	6
L	42	110	3
M	24	11111	5
U	37	100	3
Z	2	111100	6

Search in Huffman Trees

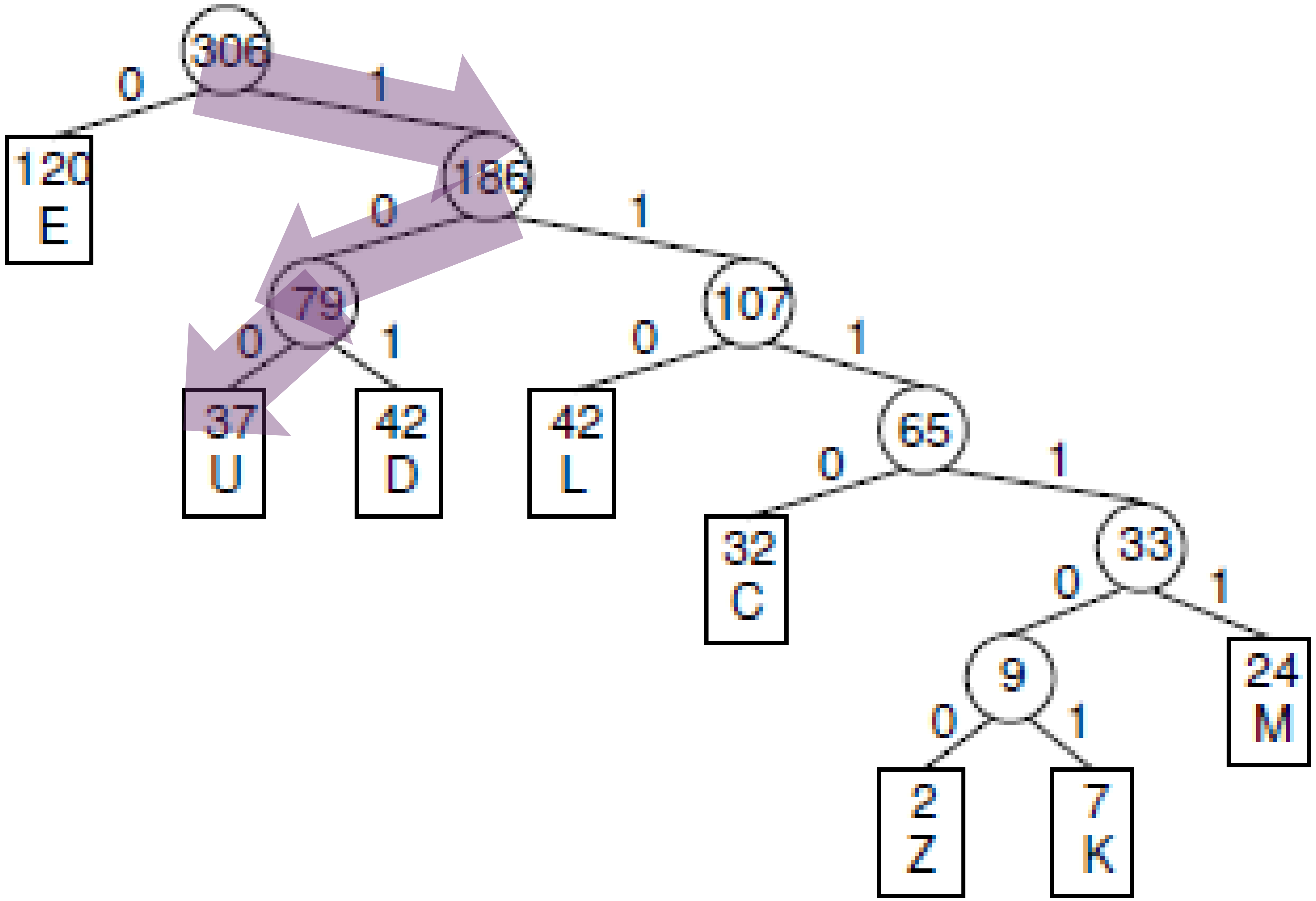
Searching



1011001110111101

D

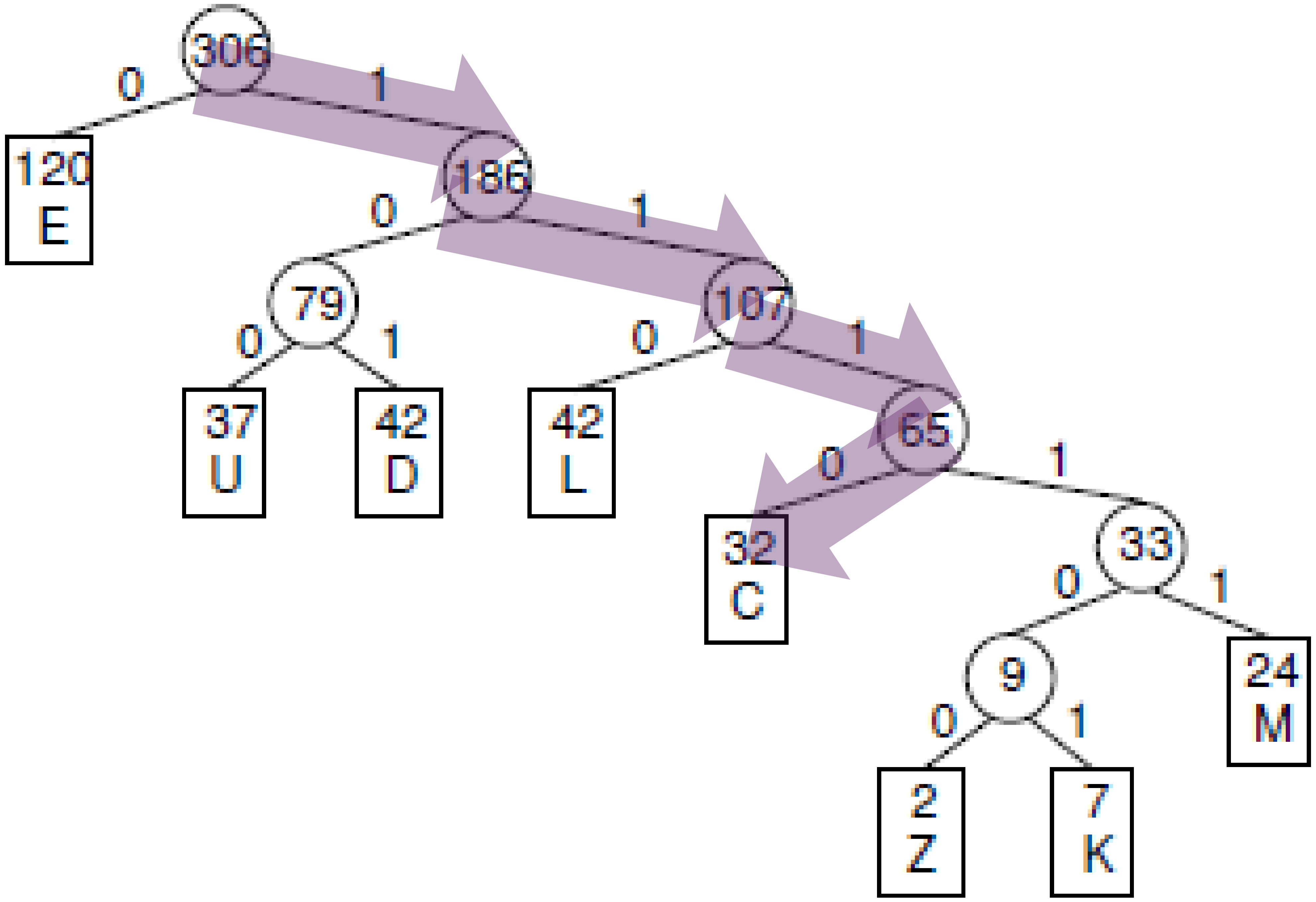
Searching



~~101~~1001110111101

D U

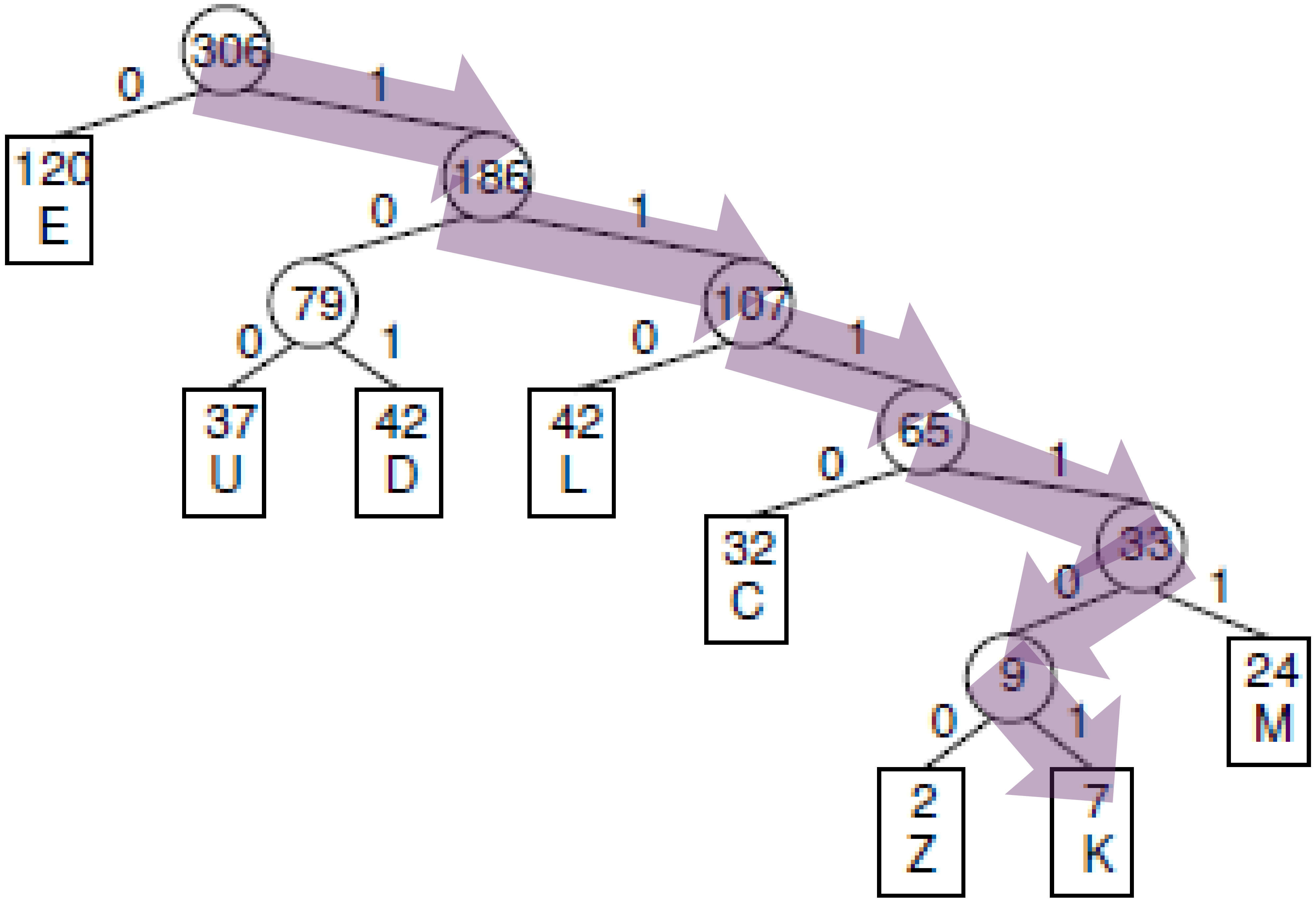
Searching



~~101100~~1110111101

D U C

Searching



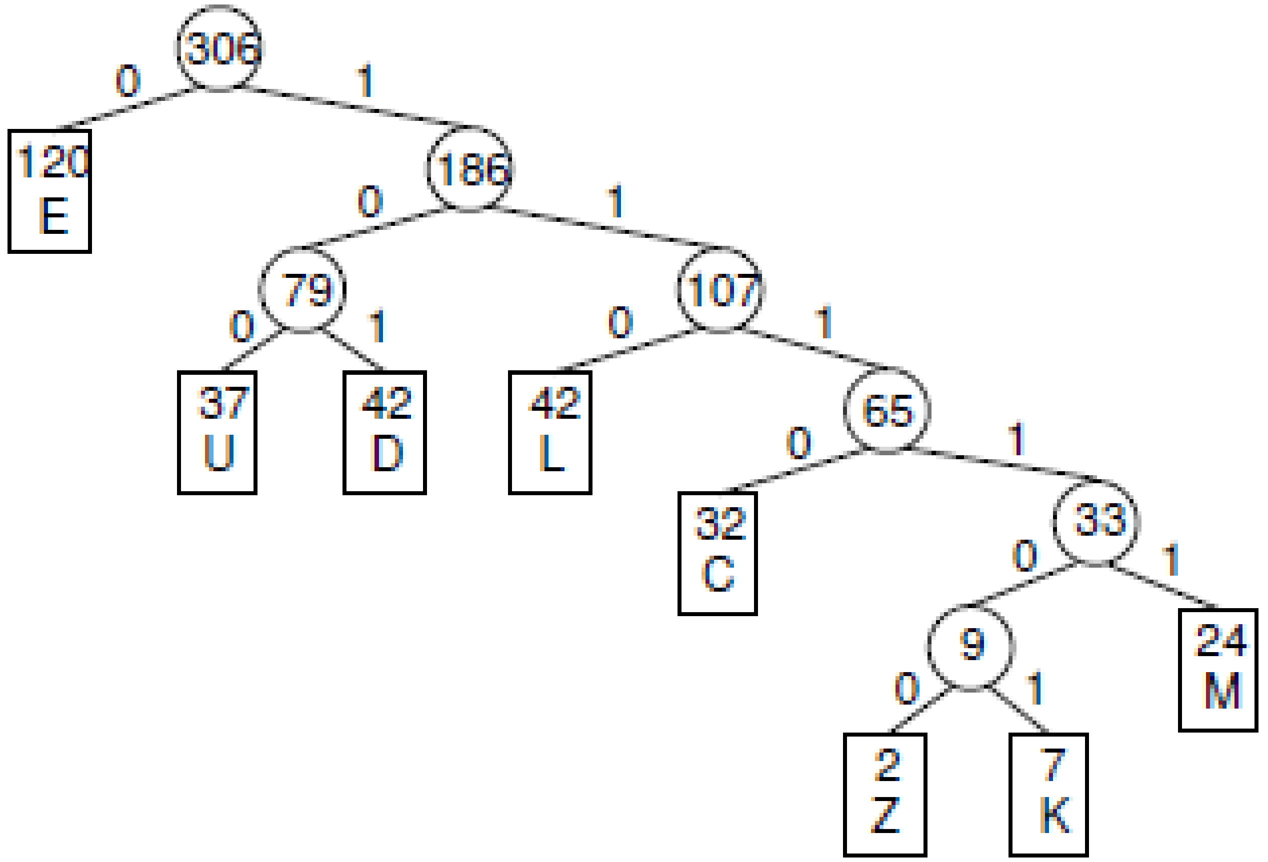
~~1011001110~~111101

D U C K

Prefix Property

- No code in the set is the prefix of another.
- The prefix property guarantees that there will be no ambiguity in how a bit string is decoded.

- Example: 11111 → M



Efficiency

- The true frequencies are known.
- The frequency of a letter is independent of the context of that letter in the message.
- The relative frequencies of the letters.

Efficiency

Example 5.11 Because the sum of the frequencies in Figure 5.31 is 306 and E has frequency 120, we expect it to appear 120 times in a message containing 306 letters. An actual message might or might not meet this expectation. Letters D, L, and U have code lengths of three, and together are expected to appear 121 times in 306 letters. Letter C has a code length of four, and is expected to appear 32 times in 306 letters. Letter M has a code length of five, and is expected to appear 24 times in 306 letters. Finally, letters K and Z have code lengths of six, and together are expected to appear only 9 times in 306 letters. The average expected cost per character is simply the sum of the cost for each character (c_i) times the probability of its occurring (p_i), or

$$c_1p_1 + c_2p_2 + \cdots + c_np_n.$$

This can be reorganized as

$$\frac{c_1f_1 + c_2f_2 + \cdots + c_nf_n}{f_T}$$

where f_i is the (relative) frequency of letter i and f_T is the total for all letter frequencies. For this set of frequencies, the expected cost per letter is

$$[(1\times 120)+(3\times 121)+(4\times 32)+(5\times 24)+(6\times 9)]/306 = 785/306 \approx 2.57$$

A fixed-length code for these eight characters would require $\log 8 = 3$ bits per letter as opposed to about 2.57 bits per letter for Huffman coding. Thus, Huffman coding is expected to save about 14% for this set of letters.