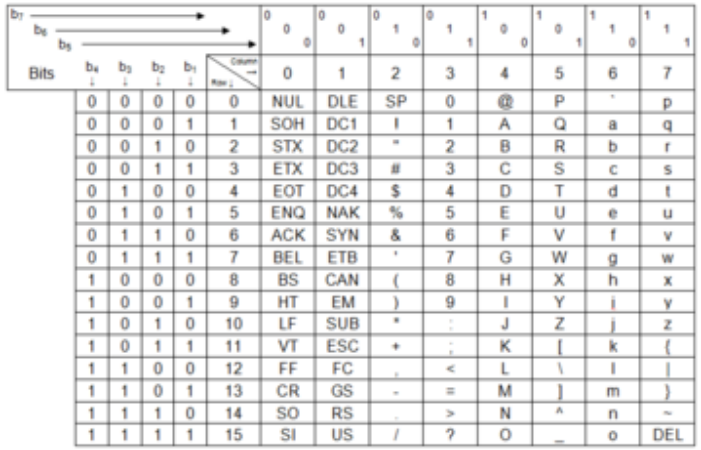
Lecture

Huffman code

ASCII Example



Bits b_4 b_3 b_2 b_1	Column 0 (000)	Column 1 (001)	Column 2 (010)	Column 3 (011)	Column 4 (100)	Column 5 (101)	Column 6 (110)	Column 7 (111)
0 0 0 0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	STX	DC2	"	2	B	R	b	r
0 0 1 1	ETX	DC3	#	3	C	S	c	s
0 1 0 0	EOT	DC4	$	4	D	T	d	t
0 1 0 1	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	ACK	SYN	&	6	F	V	f	v
0 1 1 1	BEL	ETB	'	7	G	W	g	w
1 0 0 0	BS	CAN	(	8	H	X	h	x
1 0 0 1	HT	EM	)	9	I	Y	i	y
1 0 1 0	LF	SUB	*	:	J	Z	j	z
1 0 1 1	VT	ESC	+	;	K	[	k	{
1 1 0 0	FF	FC	,	<	L	\	l	|
1 1 0 1	CR	GS	-	=	M	]	m	}
1 1 1 0	SO	RS	.	>	N	^	n	~
1 1 1 1	SI	US	/	?	O	_	o	DEL

BCAA

B C A A

1000010 1000011 1000001 1000001

Fixed Length Code : Space Used

Assume an m bit fixed length code.

For a file of n characters

Need nm bits.

Variable Length Code : Space Used

Idea: use less bits for frequent characters and more bits for rare characters.

Example: suppose alphabet of 3 symbols:
{ X, Y, Z }.
suppose in file: 1000000
characters.
Need 2 bits for a fixed length code for a total of
2,000000 bits.

Variable Length Code : Space Used

Idea: use less bits for frequent characters and more bits for rare characters.

Suppose the frequency distribution of the characters is:

X	Y	Z
999,000	500	500

Encode:

X	Y	Z
0	10	11

Variable Length Code VS Fixed Length Code : Space Used

Fixed code: 1,000,000 x 2 = 2,000,000

Varable code: 999,000 x 1
\+ 500 x 2
500 x 2

1,001,000

Fixed Length : Decoding

Example: X = 00
Y = 01
Z= 10

000010 is XXZ
100001 is ZXY

Variable Length : Decoding

Idea : Prefix coding

Prefix code, where no code is a prefix of another.

Example: X = 0

Y = 10

Z = 11

None of the above codes is a prefix of another.

Variable Length : Decoding

Example: X = 0
Y = 10
Z = 11

So, for the string:
X X X Y Y Z Z Z the encoding:

0 0 0 10 10 11 11 11

Variable Length : Coding

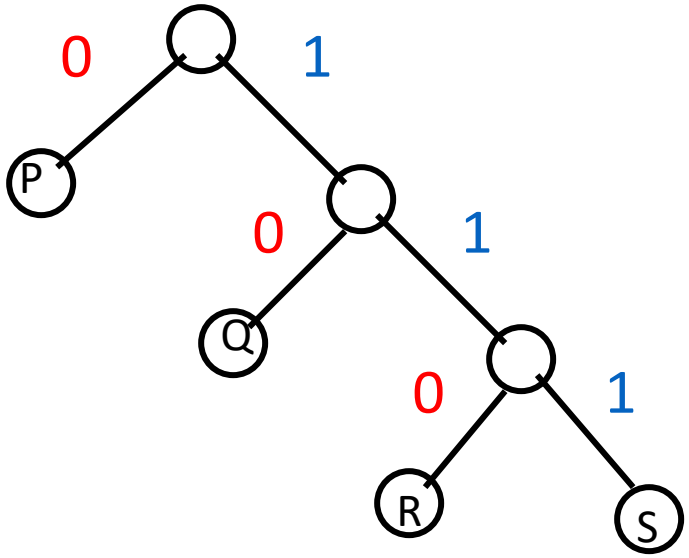
Construct a variable length code for a given file with the following properties:

1. Prefix code.
2. Using shortest possible codes.
3. Efficient.

Variable Length : Coding

Idea :Consider the paths from the root to each of the leaves P, Q, R, S:

P : 0
Q : 10
R : 110
S : 111



Greedy Approach (Variable Length Coding)

1. Consider all pairs: <frequency, symbol>.

2. Choose the two lowest frequencies, and make them brothers, with the root having the combined frequency.

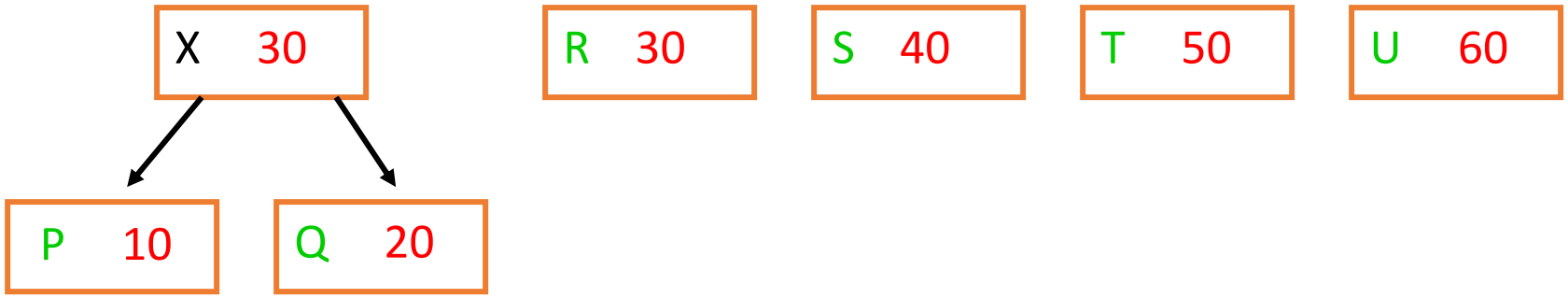
3. Iterate.

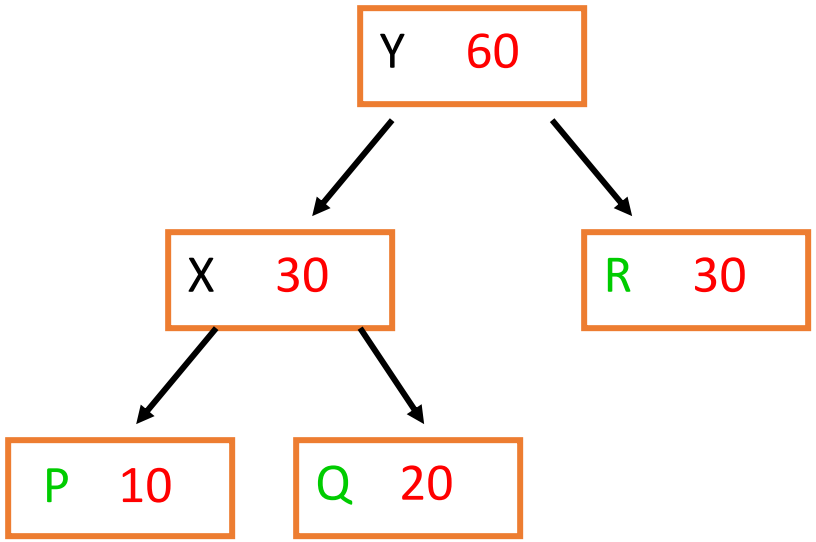
Greedy Approach : Example

P	Q	R	S	T	U
10	20	30	40	50	60

Greedy Approach : Example

P 10	Q 20	R 30	S 40	T 50	U 60

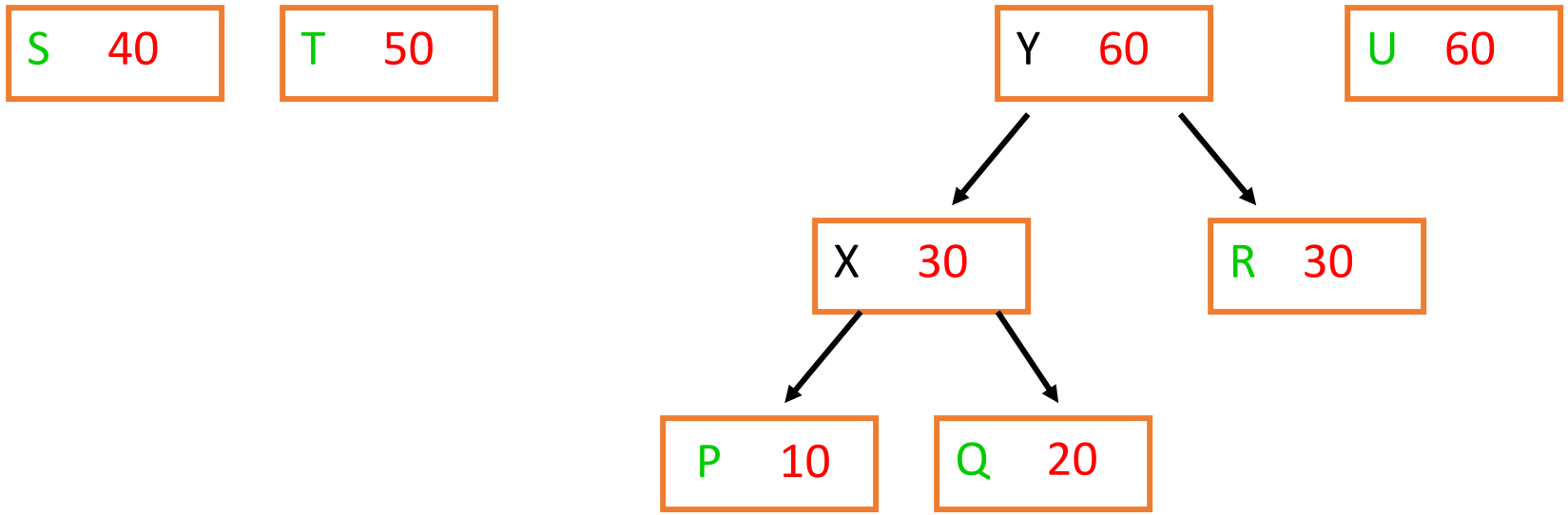

X 30
R 30
S 40
T 50
U 60
P 10
Q 20

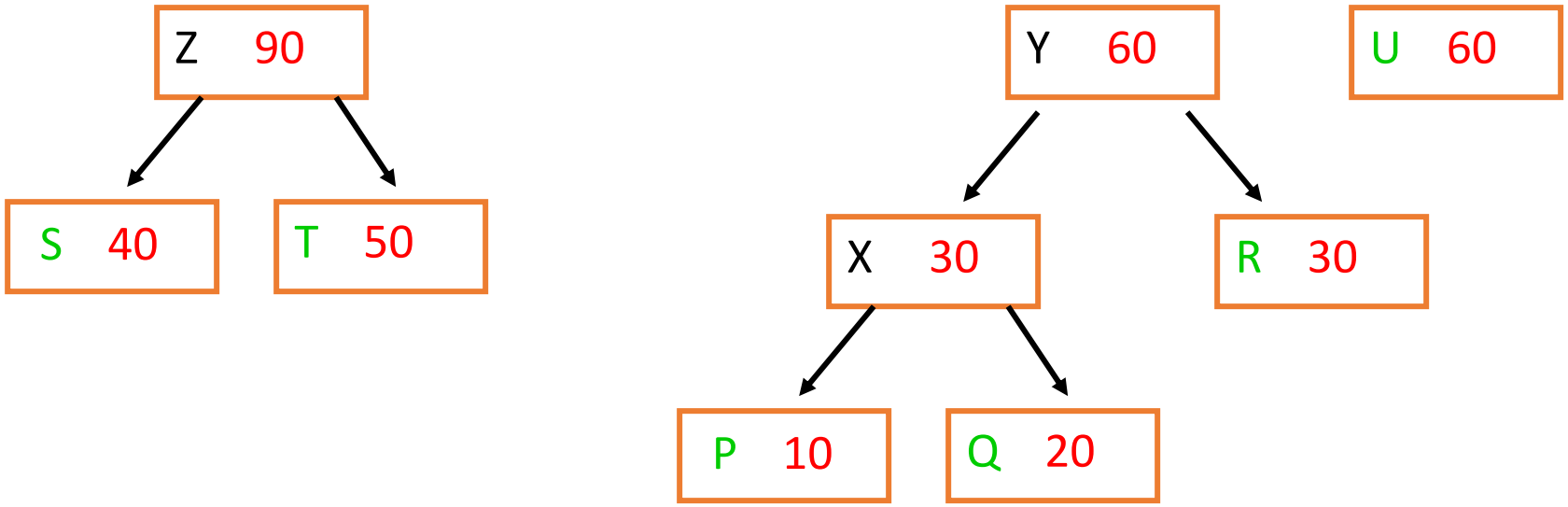


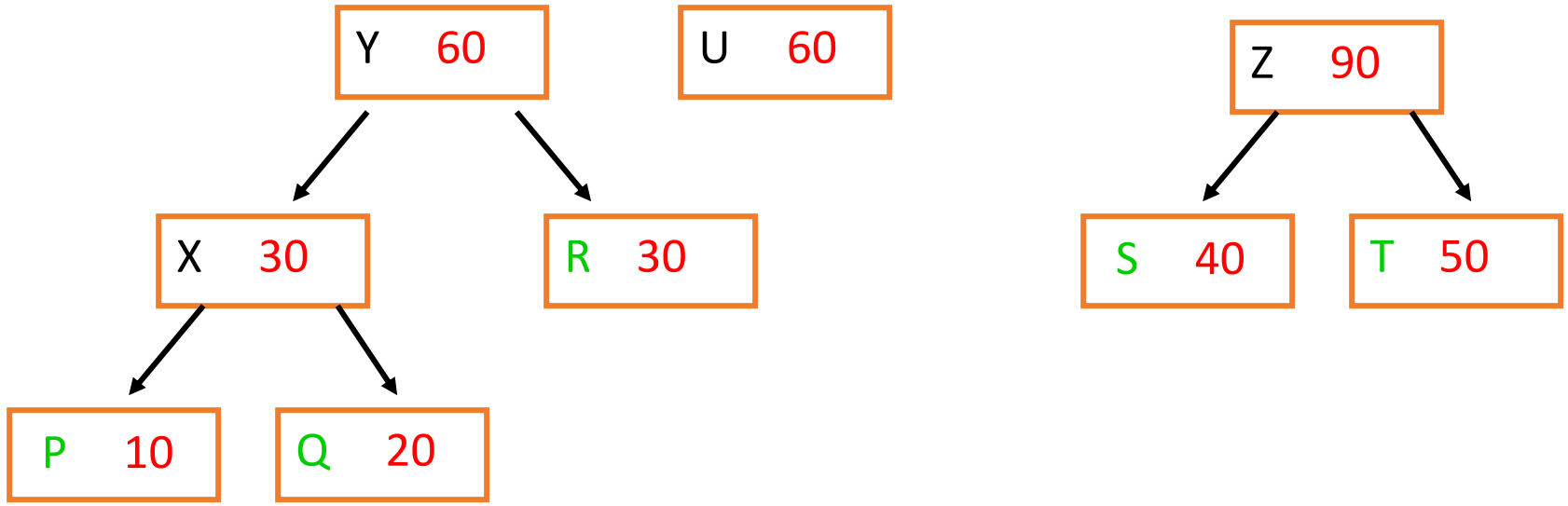
S 40

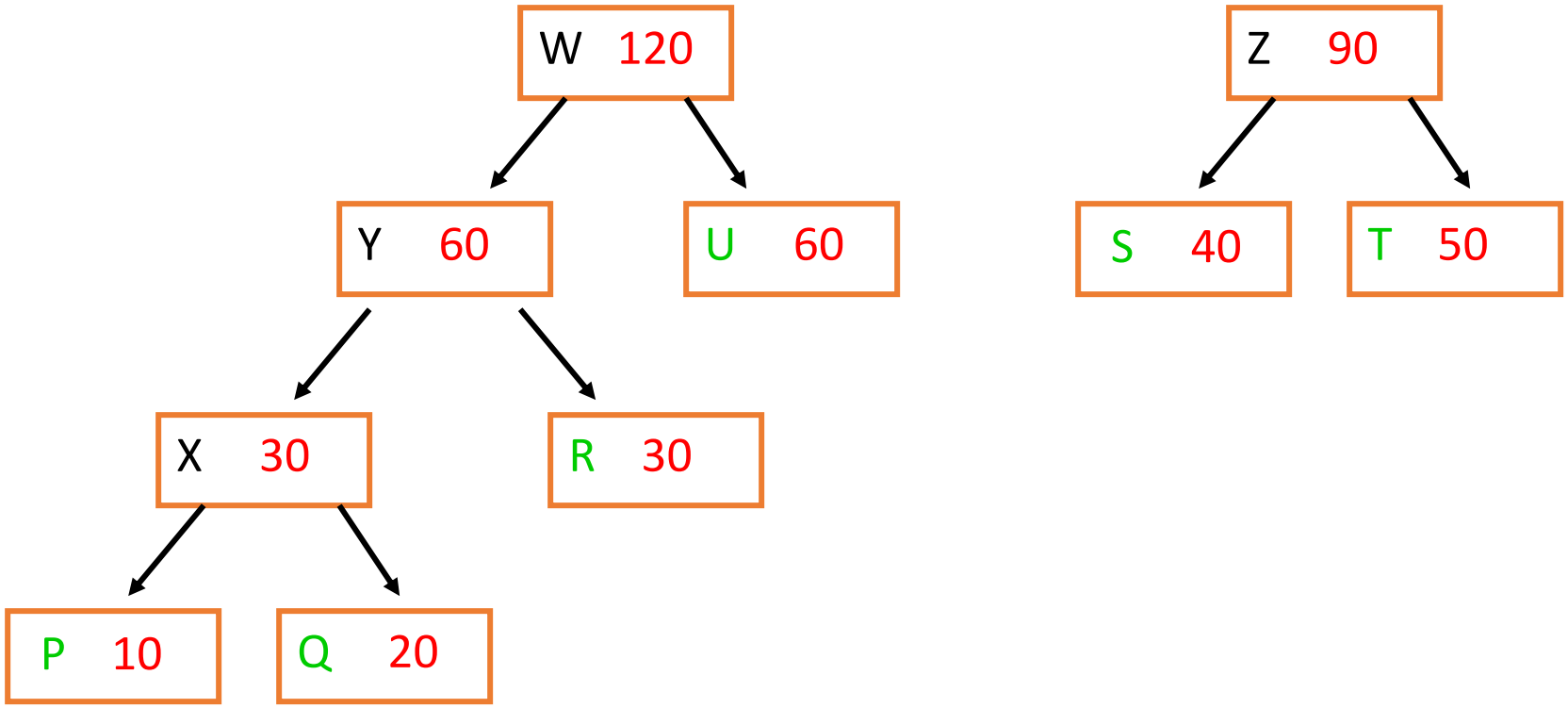
T 50

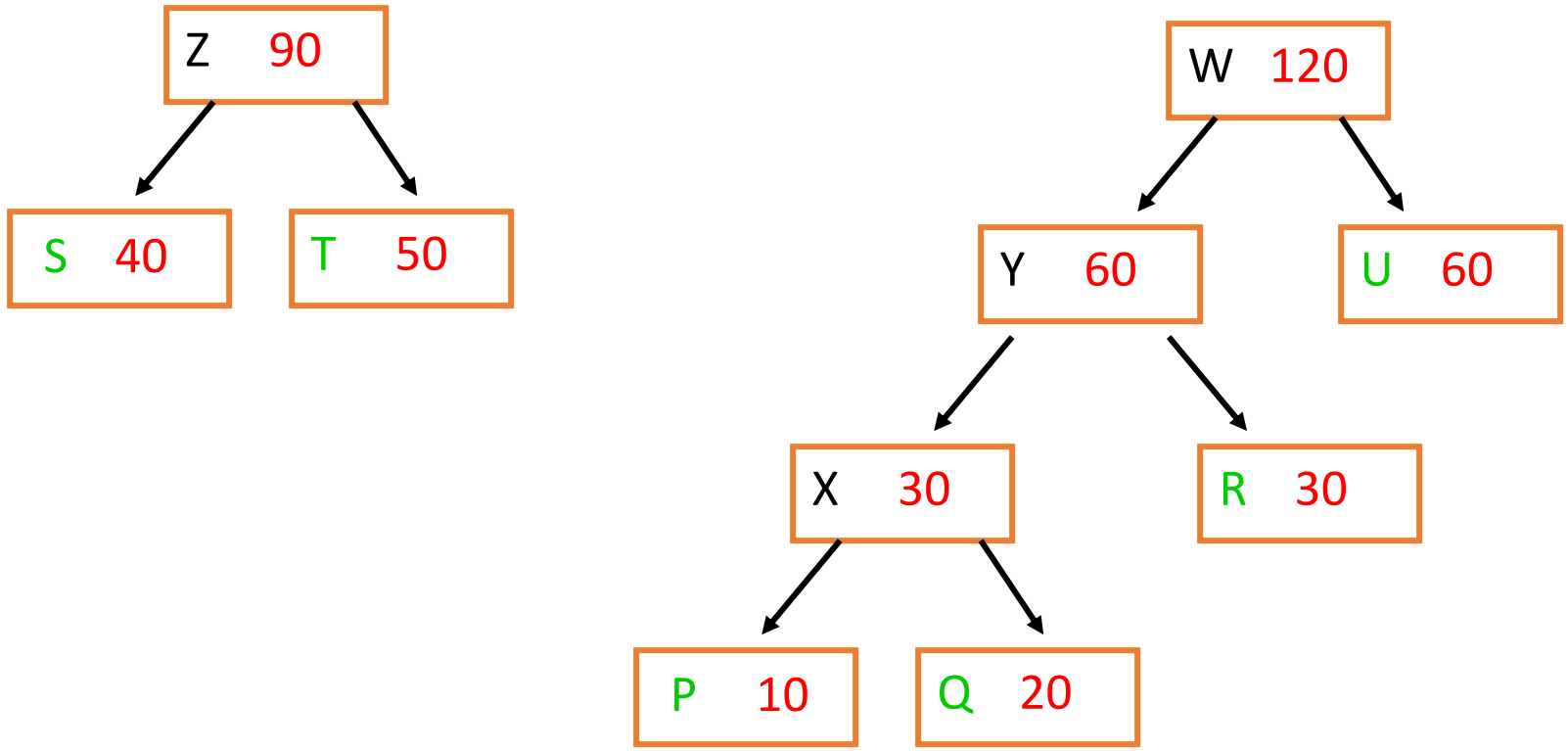
U 60

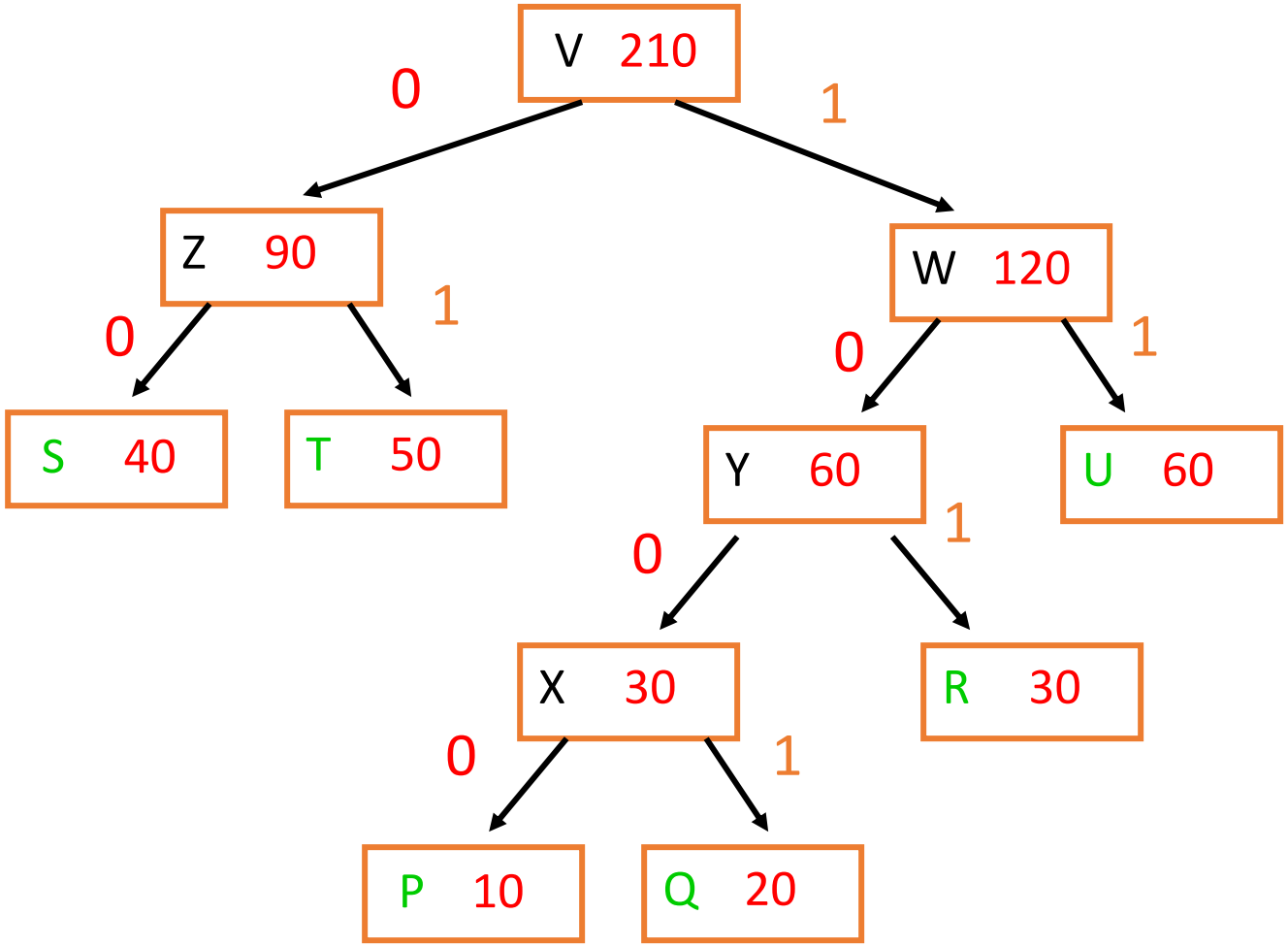

S 40
T 50
Y 60
U 60
X 30
R 30
P 10
Q 20


Z 90
S 40
T 50
Y 60
U 60
X 30
R 30
P 10
Q 20


Y 60
U 60
X 30
R 30
P 10
Q 20
Z 90
S 40
T 50

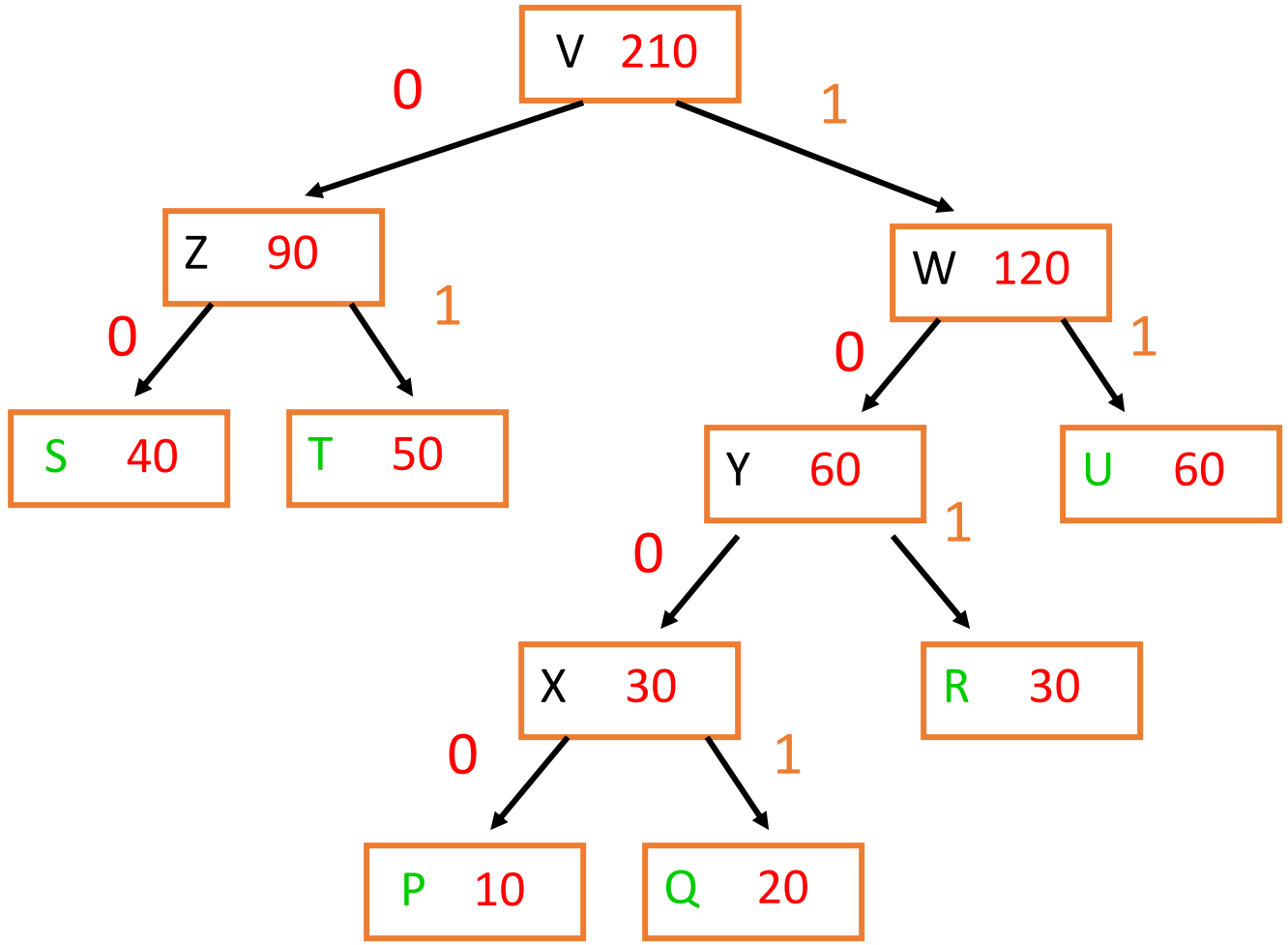

W 120
Y 60
U 60
X 30
R 30
P 10
Q 20
Z 90
S 40
T 50


Z 90
S 40
T 50
W 120
Y 60
U 60
X 30
R 30
P 10
Q 20


V 210
0
1
Z 90
W 120
0
1
0
1
S 40
T 50
Y 60
U 60
0
1
X 30
R 30
0
1
P 10
Q 20

The Huffman encoding:

P: 1000
Q: 1001
R: 101
S: 00
T: 01
U: 11



File Size: 10x4 + 20x4 + 30x3 + 40x2 + 50x2 + 60x2 =
40 + 80 + 90 + 80 + 100 + 120 = 510 bits

The Huffman code:

Required 510 bits for the file.

Fixed length code:

Need 3 bits for 6 characters.

File has 210 characters.

Total: 630 bits for the file.

The algorithm:

Initialize trees of a single node each.

Keep the roots of all subtrees in a priority queue.

Iterate until only one tree left:

Merge the two smallest frequency subtrees into a single subtree with two children, and insert into priority queue.

Huffman Code Construction of Tree

```
HUFFMAN(C)
1  n = |C|
2  Q = C
3  for i = 1 to n − 1
4       allocate a new node z
5       z.left = x = EXTRACT-MIN(Q)
6       z.right = y = EXTRACT-MIN(Q)
7       z.freq = x.freq + y.freq
8       INSERT(Q, z)
9  return EXTRACT-MIN(Q)      // return the root of the tree
```

Note:-

- If elements are equal put it any side.
- After constructing the tree, put Left level 0 and Right level 1.
- If answer to any question does not match check it by putting Left level 1 and Right level 0.

Algorithm Time

Each priority queue operation (e.g. heap):

$O(\log n)$

In each iteration: one less subtree.

Initially: n subtrees.

Total: $O(n \log n)$ time.

Steps for Traversing the Huffman Tree

1.Create an auxiliary array

2. Traverse the tree starting from root node

3. Add 0 to array while traversing the left child and add 1 to array while traversing the right child

4. Print the array elements whenever a leaf node is found

Do yourself

Character	a	b	c	d	e	f
Frequency	70	2	5	13	3	7
Huffman Code	1	01100	0111	00	01101	010

Thank You