

Huffman Coding

What if we want to send the text **bananabread**?

97	141	61	01100001	a	a		Lowercase a
98	142	62	01100010	b	b		Lowercase b
99	143	63	01100011	c	c		Lowercase c
100	144	64	01100100	d	d		Lowercase d
101	145	65	01100101	e	e		Lowercase e
102	146	66	01100110	f	f		Lowercase f
103	147	67	01100111	g	g		Lowercase g
104	150	68	01101000	h	h		Lowercase h
105	151	69	01101001	i	i		Lowercase i
106	152	6A	01101010	j	j		Lowercase j
107	153	6B	01101011	k	k		Lowercase k
108	154	6C	01101100	l	l		Lowercase l
109	155	6D	01101101	m	m		Lowercase m
110	156	6E	01101110	n	n		Lowercase n
111	157	6F	01101111	o	o		Lowercase o
112	160	70	01110000	p	p		Lowercase p
113	161	71	01110001	q	q		Lowercase q
114	162	72	01110010	r	r		Lowercase r
115	163	73	01110011	s	s		Lowercase s
116	164	74	01110100	t	t		Lowercase t
117	165	75	01110101	u	u		Lowercase u
118	166	76	01110110	v	v		Lowercase v
119	167	77	01110111	w	w		Lowercase w
120	170	78	01111000	x	x		Lowercase x
121	171	79	01111001	y	y		Lowercase y
122	172	7A	01111010	z	z		Lowercase z

<https://www.ascii-code.com/>

Instead, we can use a **variable-length encoding**:
use a different number of bits for each character.

Instead, we can use a **variable-length encoding**:
use a different number of bits for each character.

Huffman coding:

Instead, we can use a **variable-length encoding**:
use a different number of bits for each character.

Huffman coding:

- Encode more frequent characters with fewer bits

Instead, we can use a **variable-length encoding**:
use a different number of bits for each character.

Huffman coding:

- Encode more frequent characters with fewer bits
- Encode less frequent characters with more bits

Instead, we can use a **variable-length encoding**:
use a different number of bits for each character.

Huffman coding:

- Encode more frequent characters with fewer bits
- Encode less frequent characters with more bits
- Uses priority queues!

Huffman Coding Algorithm

1. Count frequency of each character.
2. Insert characters into **priority queue** (lower frequency has higher priority).
3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

Huffman Coding Algorithm (**bananabread**)

Huffman Coding Algorithm (**bananabread**)

1. Count frequency of each character.

Huffman Coding Algorithm (**bananabread**)

1. Count frequency of each character.

D	E	R	B	N	A
1	1	1	2	2	4

Huffman Coding Algorithm (**bananabread**)

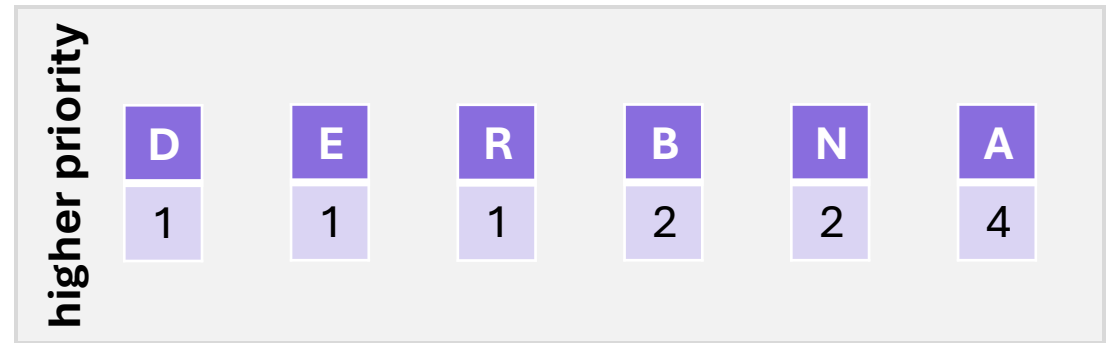
1. Count frequency of each character.
2. Insert characters into **priority queue** (lower frequency has higher priority).

D	E	R	B	N	A
1	1	1	2	2	4

Huffman Coding Algorithm (**bananabread**)

1. Count frequency of each character.
2. Insert characters into **priority queue** (lower frequency has higher priority).

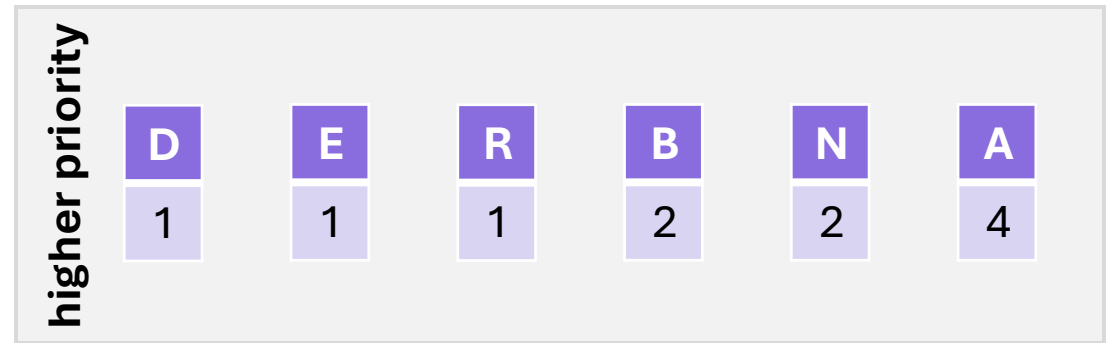
priority queue



Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

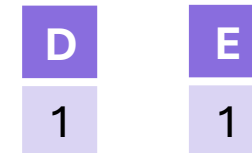
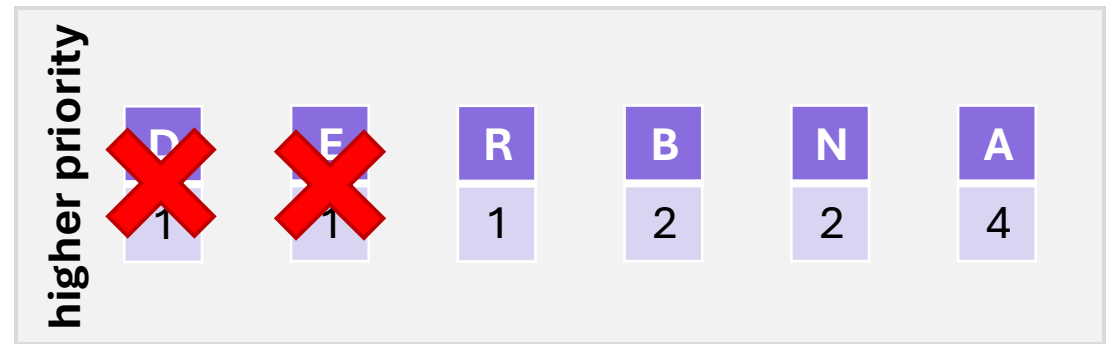
priority queue



Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

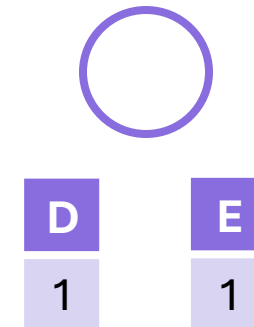
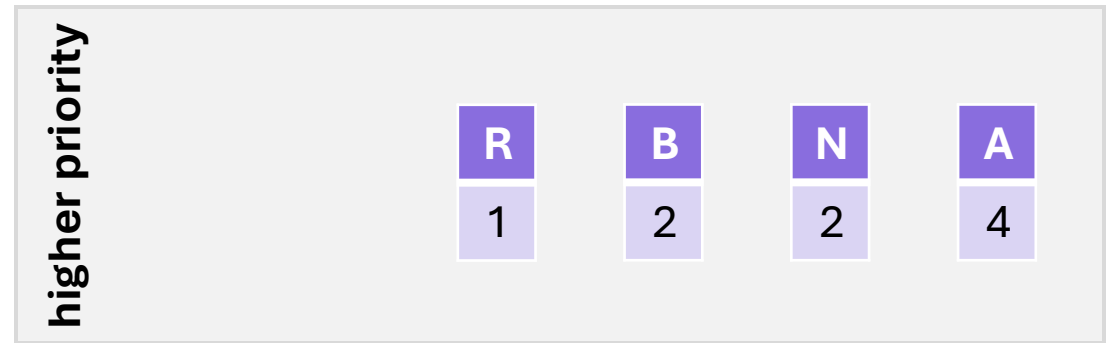
priority queue



Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

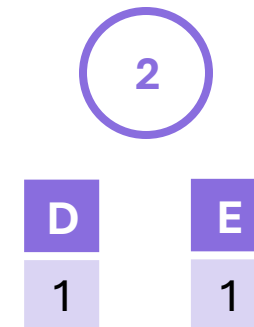
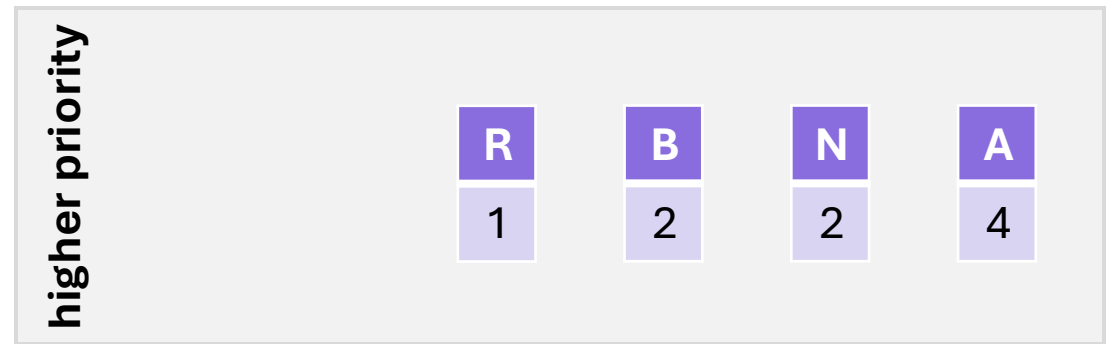
priority queue



Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

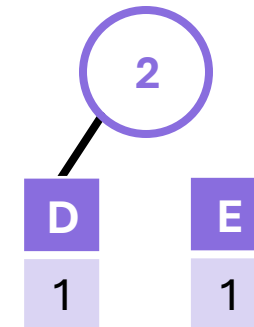
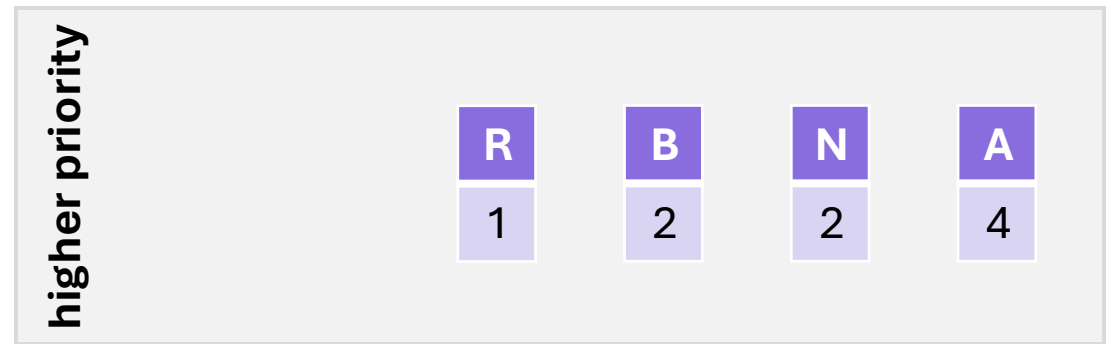
priority queue



Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

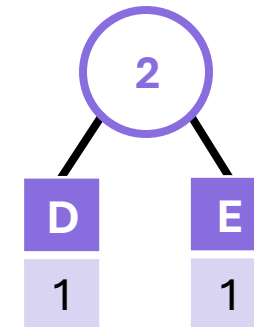
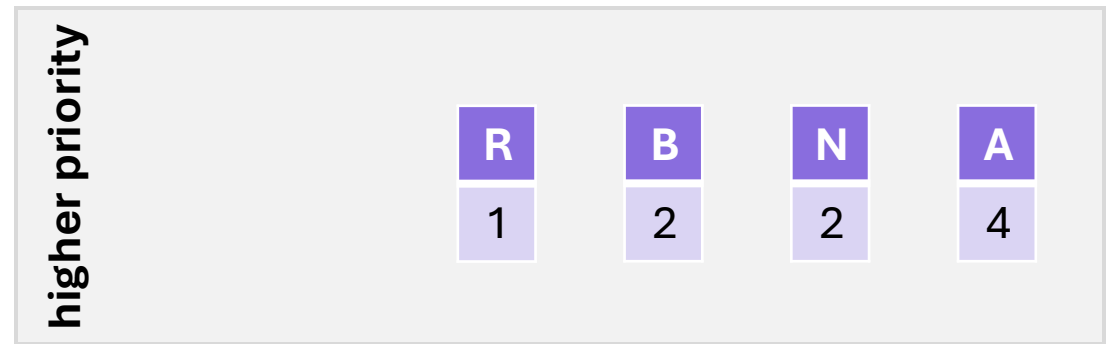
priority queue



Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

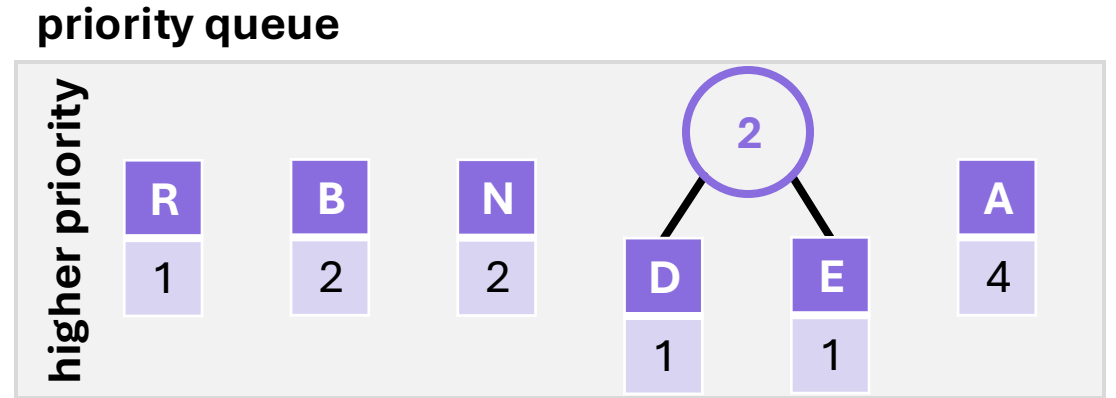
priority queue



Huffman Coding Algorithm (**bananabread**)

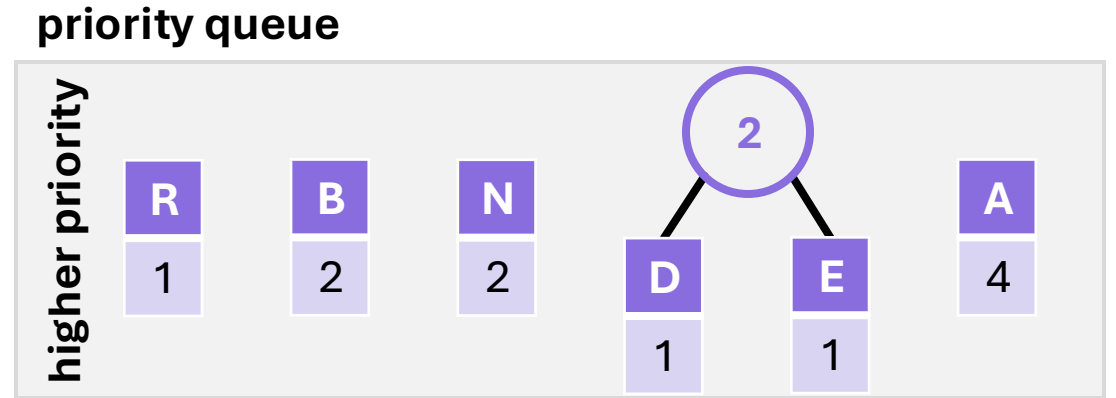
3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

★ For internal consistency in today's lab, give lower priority to internal nodes in ties!



Huffman Coding Algorithm (**bananabread**)

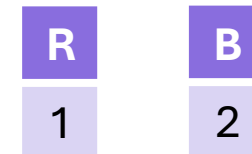
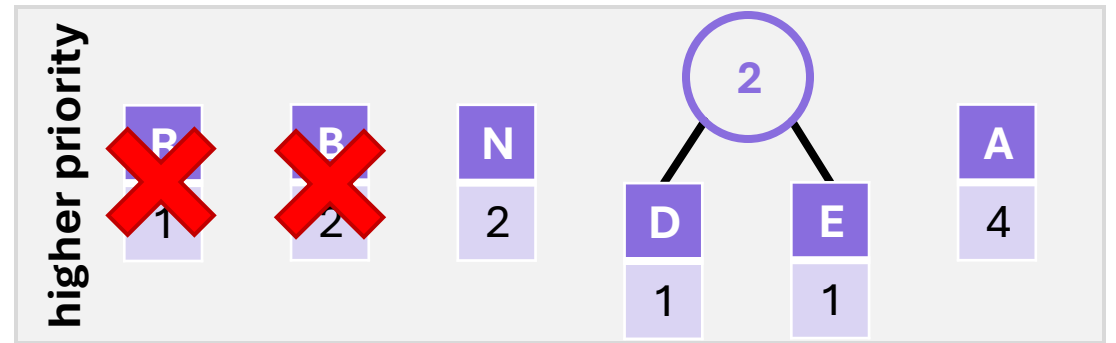
3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.



Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

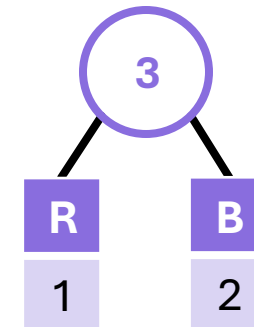
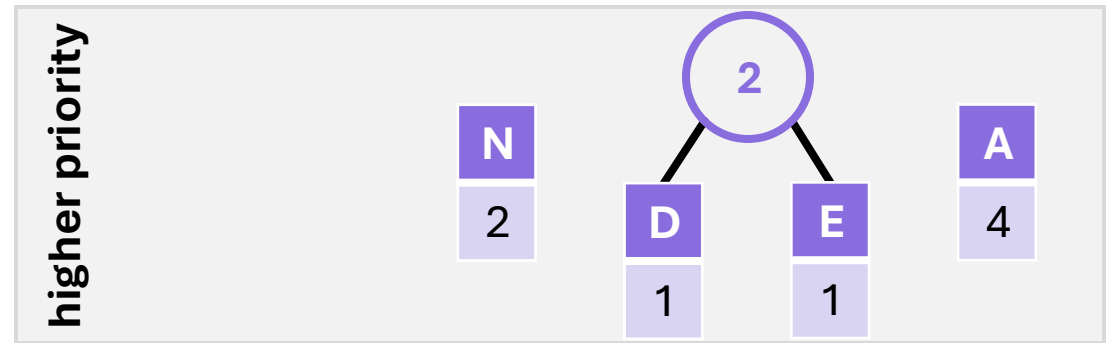
priority queue



Huffman Coding Algorithm (**bananabread**)

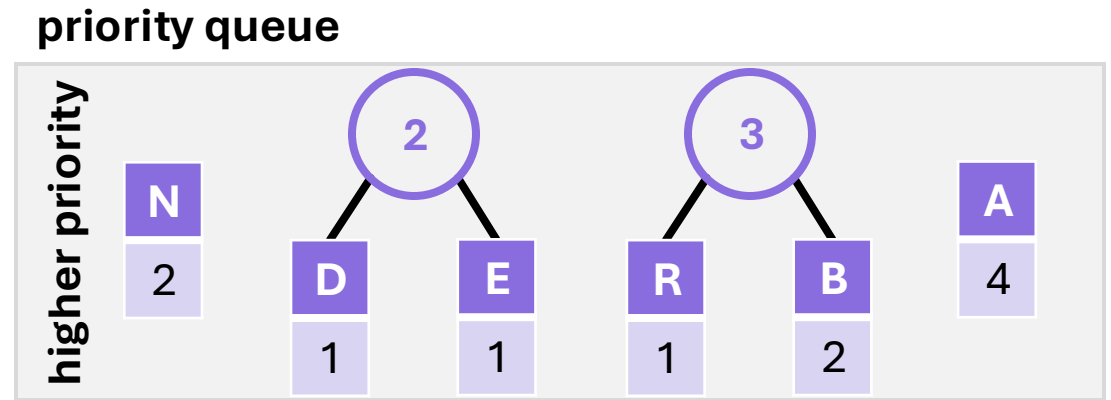
3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

priority queue



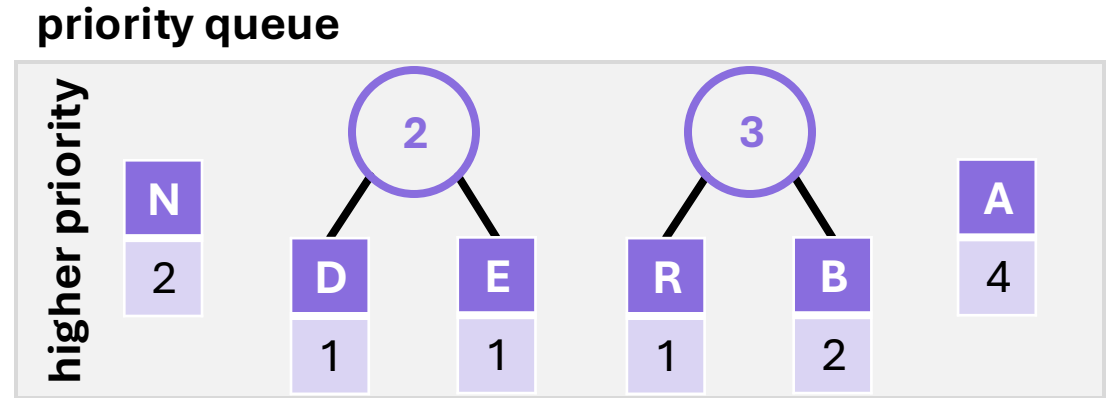
Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.



Huffman Coding Algorithm (**bananabread**)

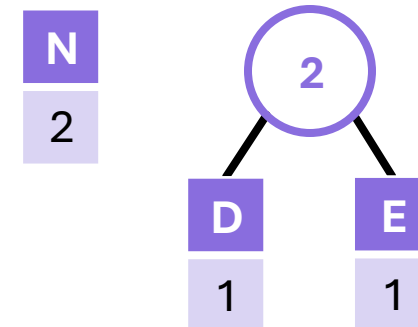
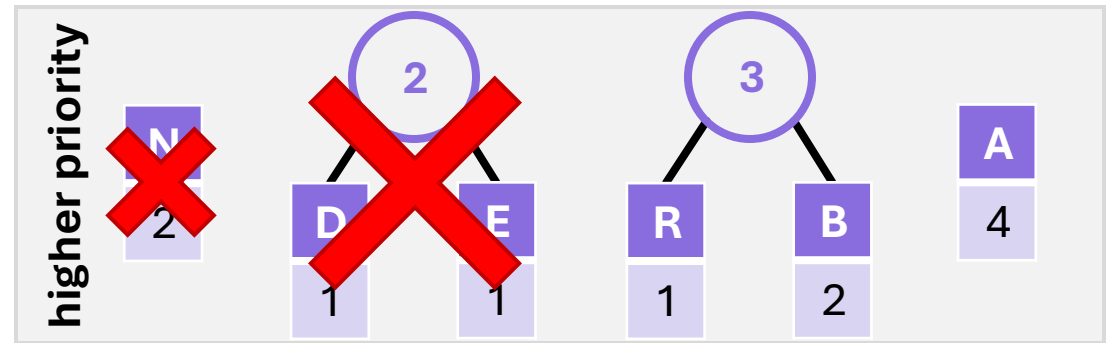
3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.



Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

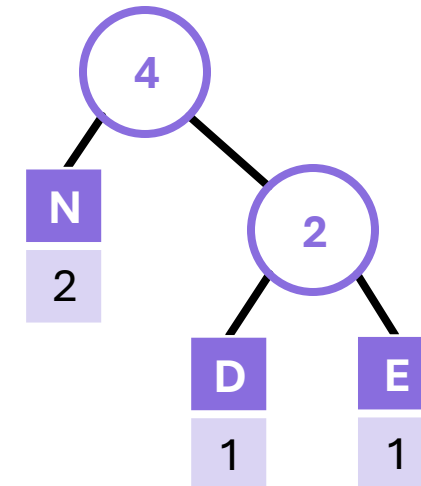
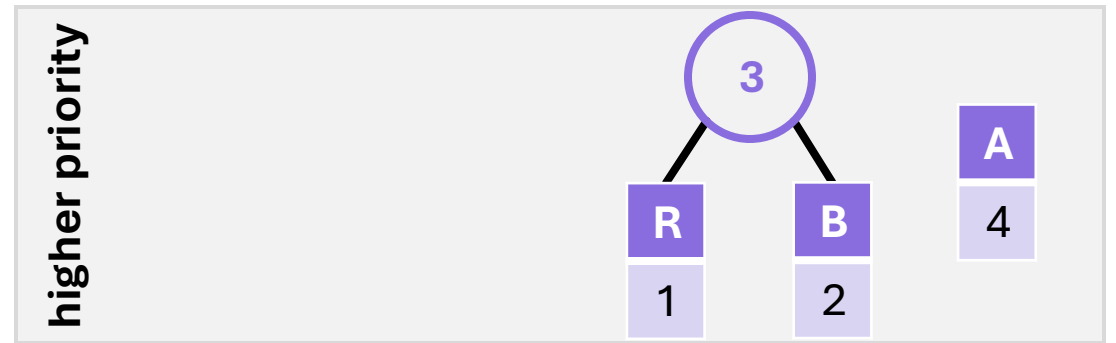
priority queue



Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

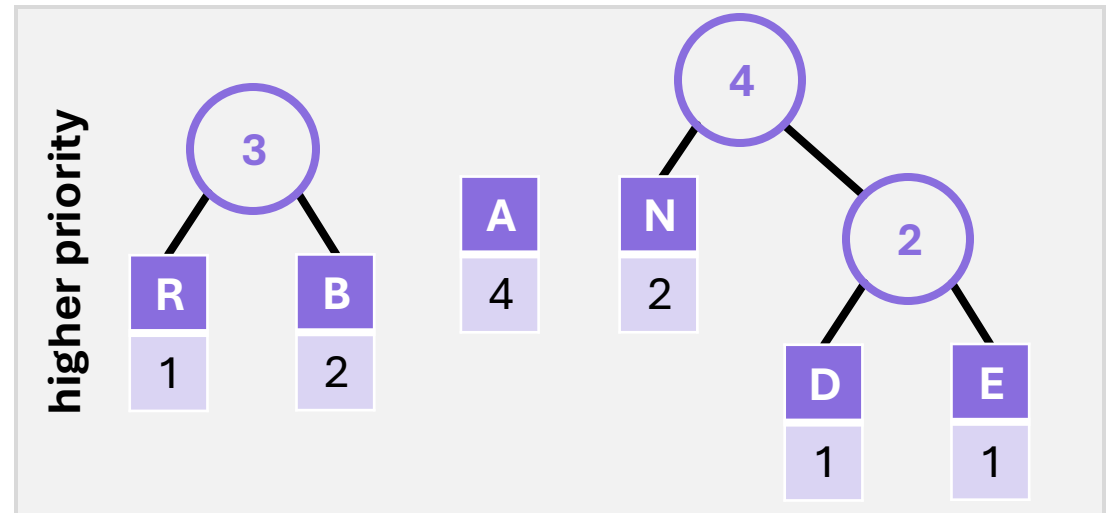
priority queue



Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

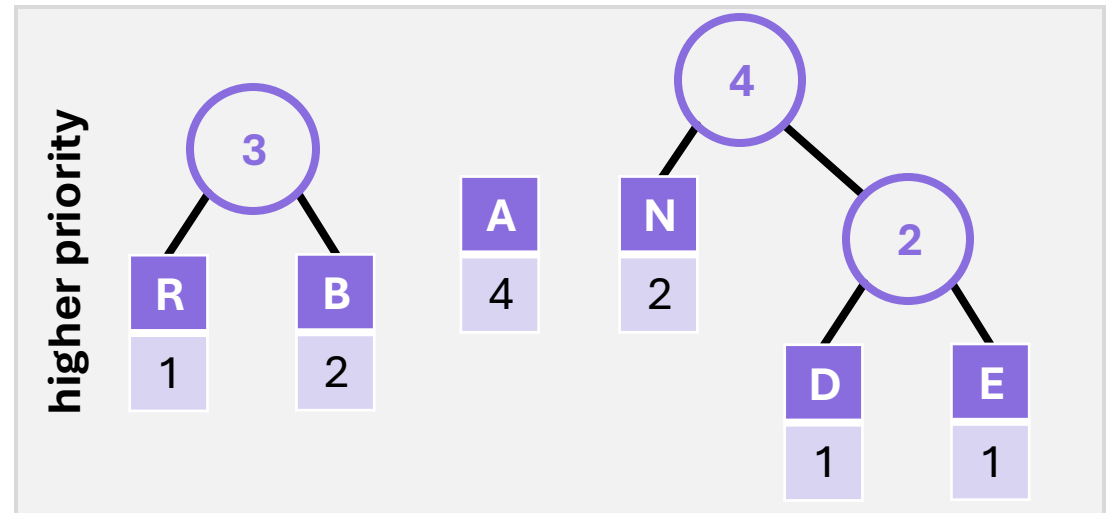
priority queue



Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

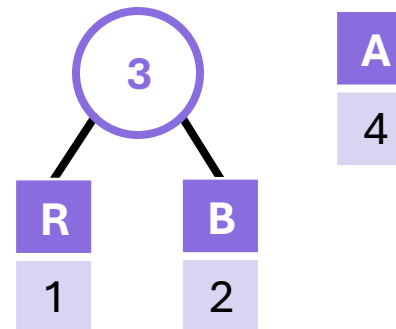
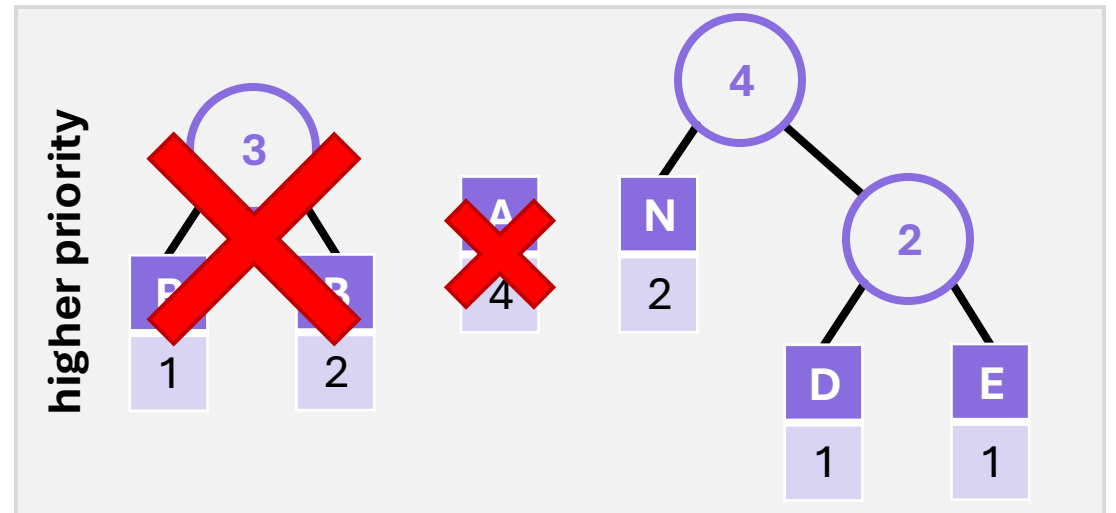
priority queue



Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

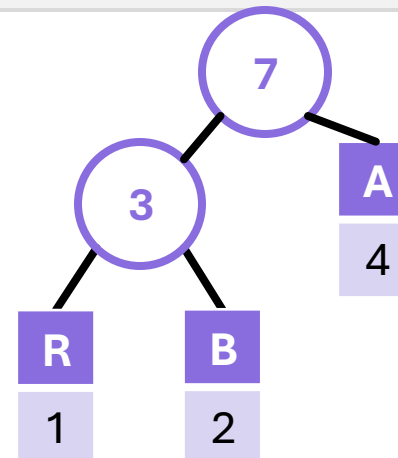
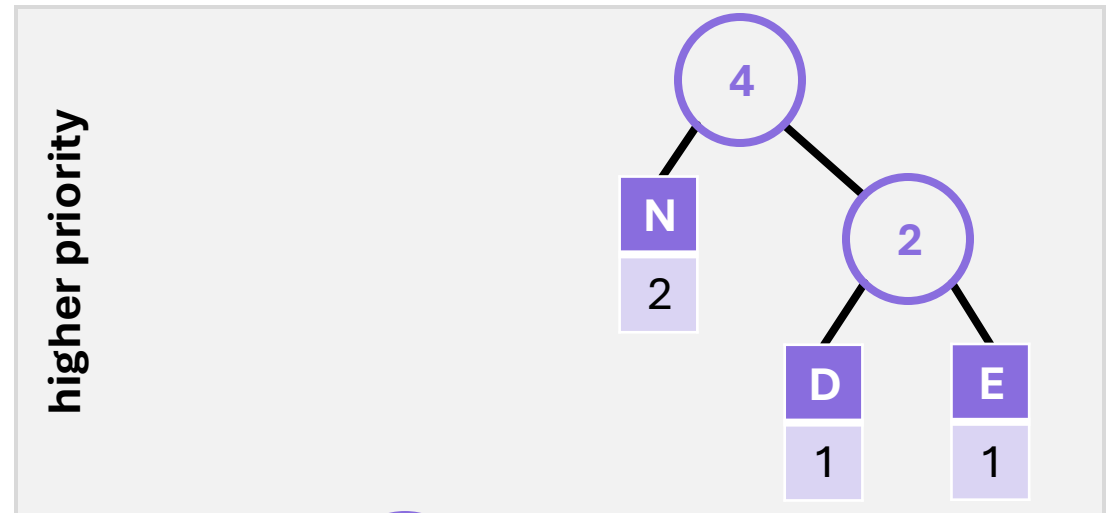
priority queue



Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items:
1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

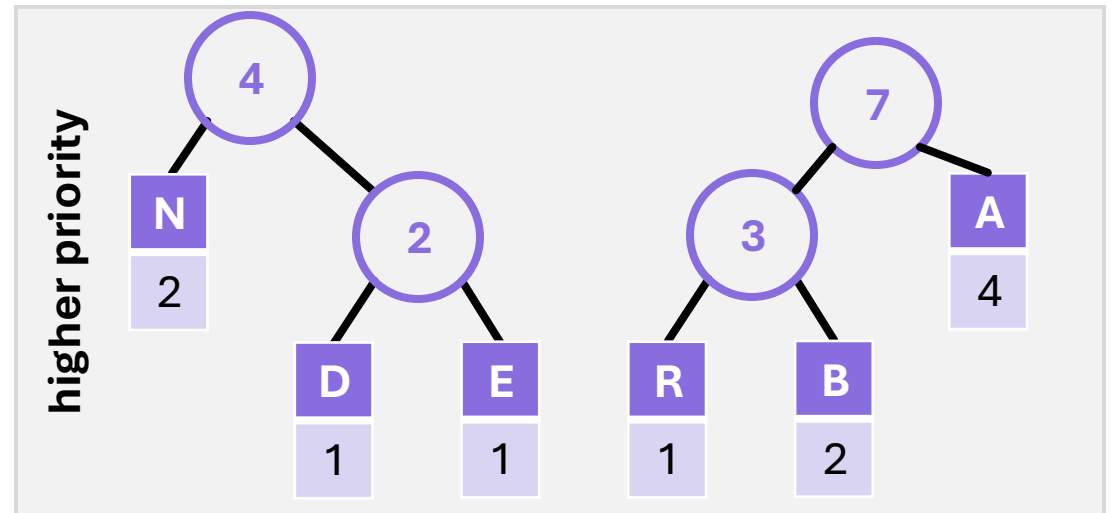
priority queue



Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items:
1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

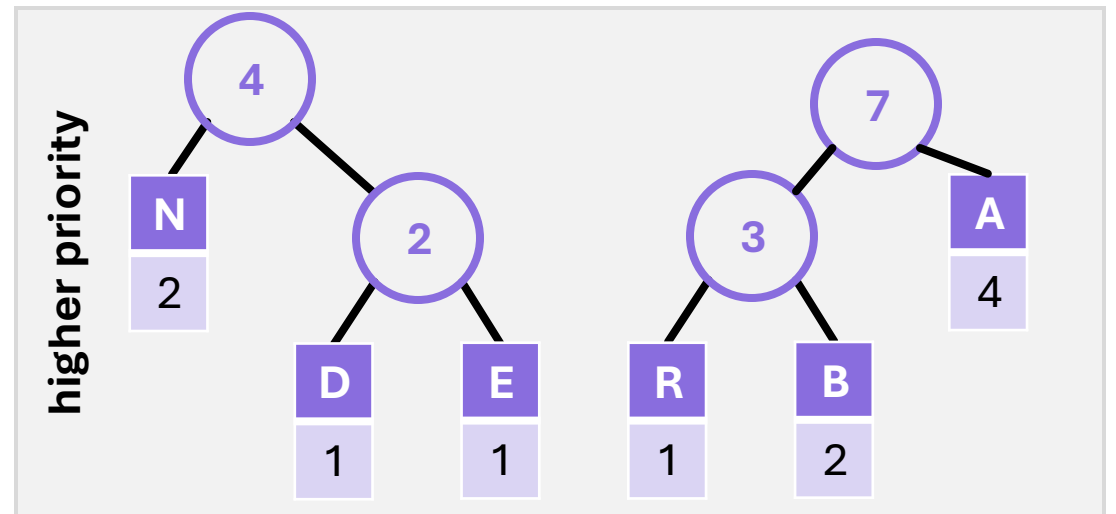
priority queue



Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

priority queue

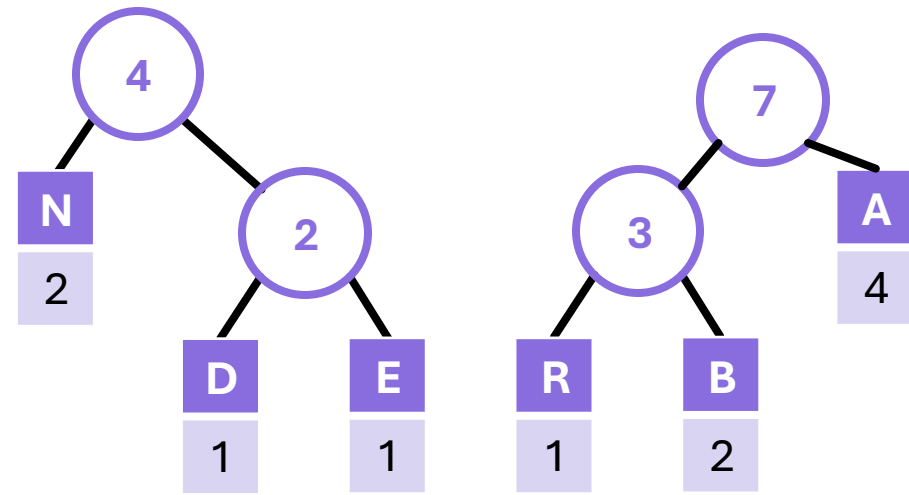


Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

priority queue

higher
priority

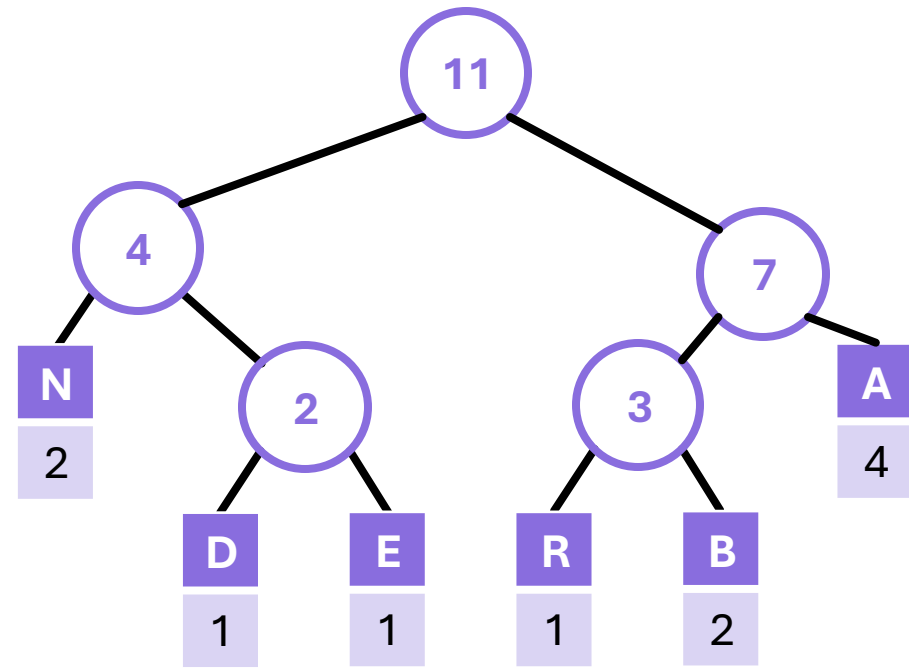


Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

priority queue

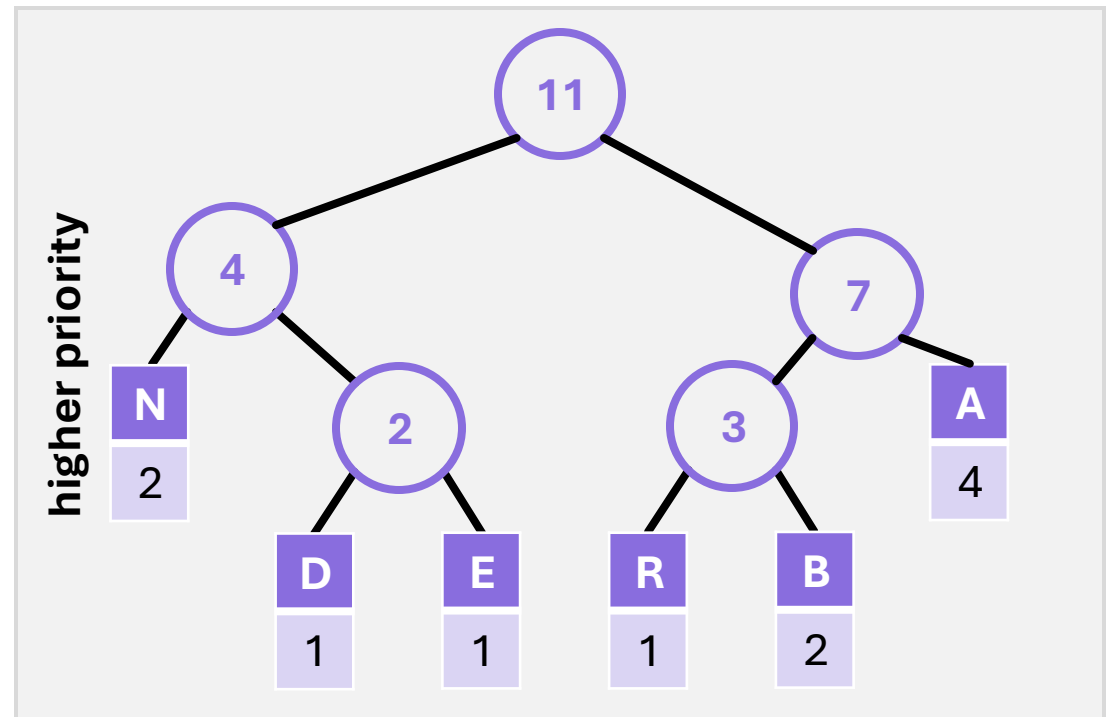
higher
priority



Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items:
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

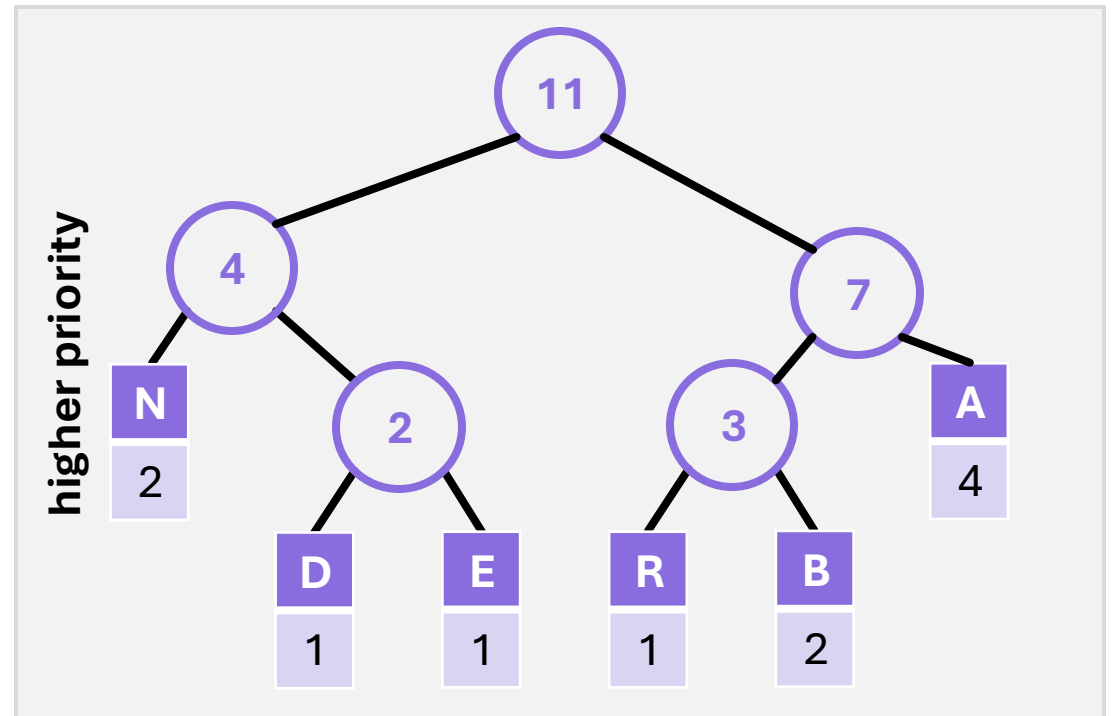
priority queue



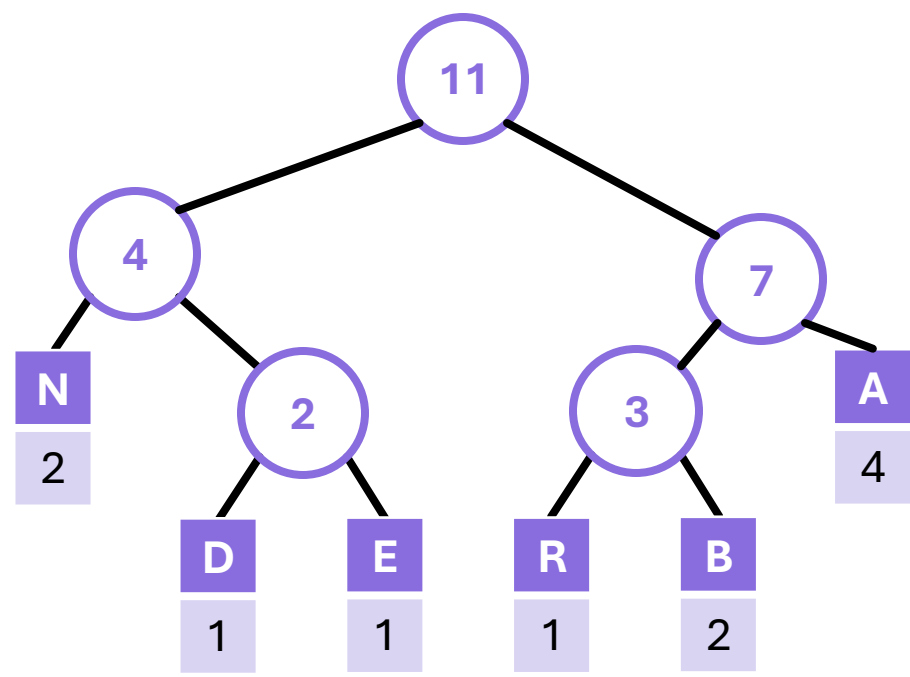
Huffman Coding Algorithm (**bananabread**)

3. **while** priority queue has at least two items: **false**
 1. Extract (and remove) top **two** items from priority queue.
 2. Create a new internal node.
 3. Add value (frequency) of two items and assign to internal node.
 4. Make the left child of the new node the first (lower) item.
 5. Make the right child of the new node the second (higher) item.
 6. Insert new node into priority queue.

priority queue

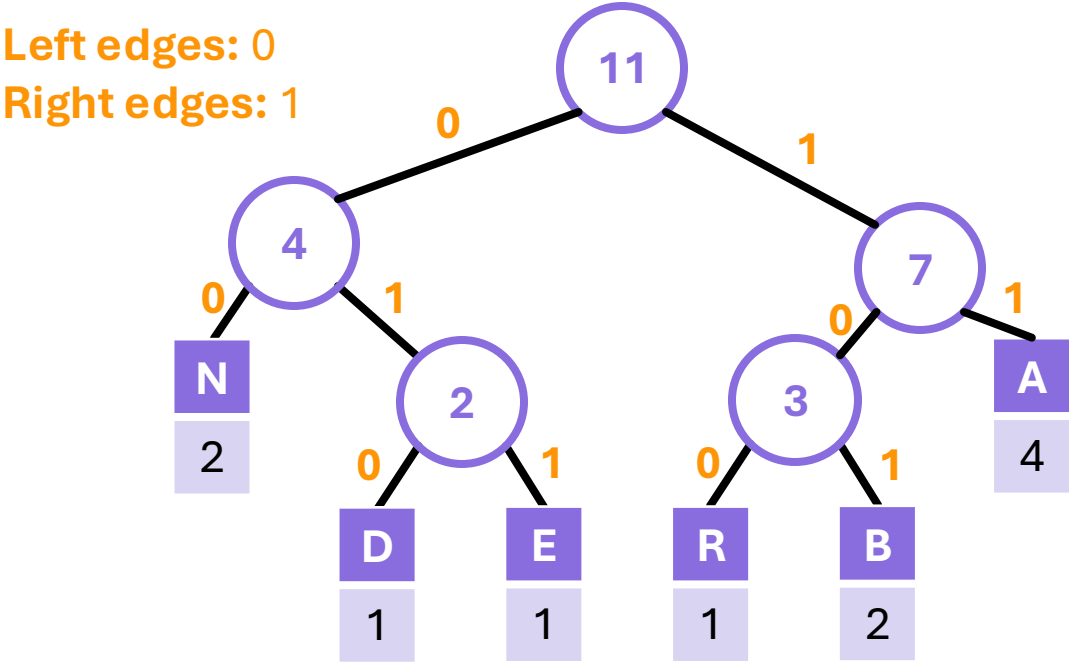


Tree to Codes



Letter	Frequency	Code
A	4	
B	2	
N	2	
D	1	
E	1	
R	1	

Tree to Codes



To **decode**, traverse tree until leaf to retrieve characters

1100010
a n d

10001111010
r e a d

Letter	Frequency	Code
A	4	11
B	2	101
N	2	00
D	1	010
E	1	011
R	1	100

How many bits to encode **bananabread**?

B	A	N	A	N	A
101	11	00	11	00	11

B	R	E	A	D
101	100	011	11	010



Only 27 bits!

Letter	Frequency	Code
A	4	11
B	2	101
N	2	00
D	1	010
E	1	011
R	1	100

Today's Lab Assignment

1. Complete the worksheet with your partner
You will want to work on the board to draw out your Huffman tree!
Remember to give internal nodes lower priorities when there are ties
2. Complete the **canvas quiz** on your own to submit and confirm some of your answers
Infinite attempts are allowed
3. If you finish, you can work on the word search, the programming exam problems, and/or the homework