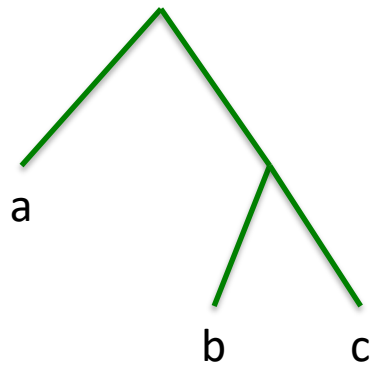
Huffman Codes

algorithms

optimal prefix codes

- binary codes for a character
- no code is the prefix of another
- example: a: 0, b:10, c:11
- simplifies decoding, scan left to right



characters are external nodes

based on frequency

- each character has a frequency
- **greedy choice**: merge together the two least frequent characters
- add their frequencies together
- this creates a new object whose frequency is the sum of the two previous frequencies

example

	a	b	c	d	e	f
frequency (000's)	45	13	12	16	9	5
fixed length word	000	001	010	011	100	101
variable-length	0	101	100	111	1101	1100

on p 429

- file has 100,000 characters
- fixed length encoding has 300,000 bits
- variable length uses 224,000

the famous algorithm

```
Huffman(C)

    priority queue Q = C

    for i = 1 to n-1
        create new node z
        z.left = x = Q.extractMin
        z.right = y = Q.extractMin
        z.freq = x.freq + y.freq
        Q.insert(z)

return Q.extractMin
```

O(nlgn) time using binary heap