HW5 - Nick Palumbo

Chapter 10, section 6

- **R-10.6**
  Which of the hash table collision-handling schemes could tolerate a load factor above 1 and which could not?

  could handle: Separate Chaining
  could not handle: Open Addressing (e.g linear probing, quadratic probing, double hashing)

- **R-10.9**
  Draw the 11-entry hash table that results from using the hash function, h(i) = (3i+5) mod 11, to hash the keys 12, 44, 13, 88, 23, 94, 11, 39, 20, 16, and 5, assuming collisions are handled by chaining.

Index	0	1	2	3	4	5	6	7	8	9	10
Buckets	13	94				44			12	16	20
		39				88			23	5	
						11					

- **R-10.10**
  What is the result of the previous exercise, assuming collisions are handled by linear probing?

0	1	2	3	4	5	6	7	8	9	10
13	94	39	16	5	44	88	11	12	23	20

- **R-10.12**
  What is the result of Exercise R-10.9 when collisions are handled by double hashing using the secondary hash function h(k) = 7 − (k mod 7)?

0	1	2	3	4	5	6	7	8	9	10
13	94	5	39	88	44	23	11	12	16	20

- **R-10.14**
  Show the result of rehashing the hash table shown in Figure 10.6 into a table of size 19 using the new hash function h(k) = 3k mod 17.

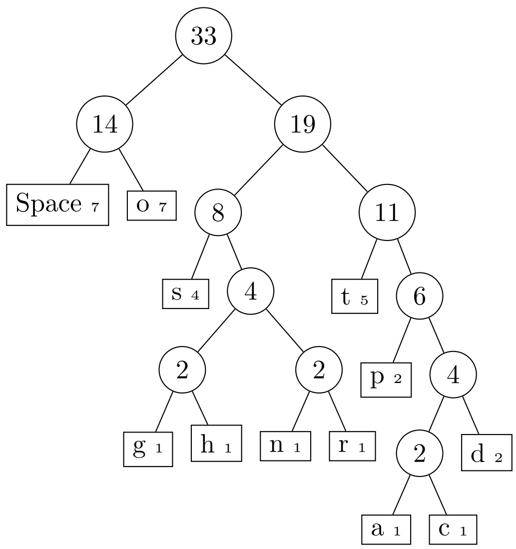
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
		12	18	41		36	25		54			38	10		90	28		

Chapter 13, section 6

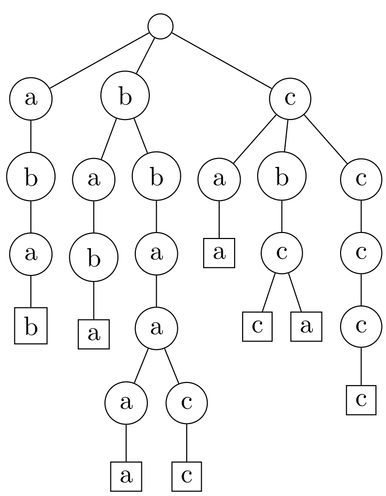
- **R-13.11**
  Draw the frequency array and Huffman tree for the following string:
  `"dogs do not spot hot pots or cats"`

Character		a	c	d	g	h	n	o	p	r	s	t
Frequency	7	1	1	2	1	1	1	7	2	1	4	5

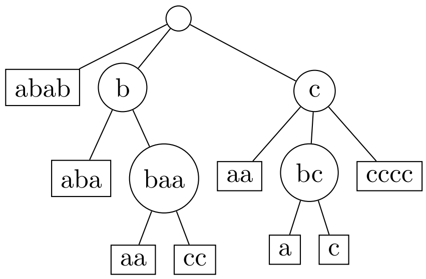
Total
33



- **R-13.12**
  Draw a standard trie for the following set of strings:
  `{ abab, baba, ccccc, bbaaaa, caa, bbaacc, cbcc, cbca }`

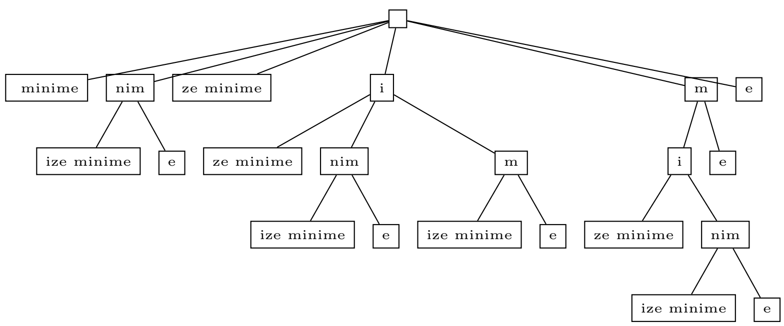


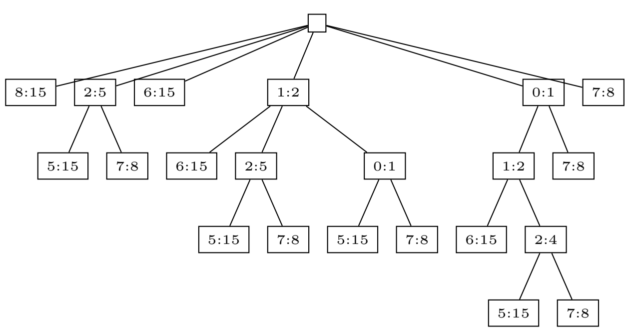
- **R-13.13**
  Draw a compressed trie for the strings given in the previous problem.



- **R-13.14**
  Draw the compact representation of the suffix trie for the string: `"minimize minime"`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
m	i	n	i	m	i	z	e		m	i	n	i	m	e





- **C-13.43**
  Give an efficient algorithm for deleting a string from a standard trie and analyze its running time.

  Input a Trie and Search for the desired string to delete. If the string reaches a leaf then begin to delete each node upwards until hitting a node with more than one child. A node cannot be deleted if it does not reach a leaf which is a termination node because that means it is only a partial string. The running time for deletion of a string would be O(l) where l is the length of the string to be deleted.

Chapter 15, section 5

- **R-15.4**
  For what values of d is the tree T of the previous exercise an order-d B-tree?

  The value for an order-d B-tree 8. A B-Tree of order-d is an (a,b) with a = [d/2] and b = d. For the previous question the (a,b) tree was equal to a = 4 and b = 8. Therefore d = 8 for the order-d tree.

- **R-15.8**
  Draw the result of inserting, into an initially empty order-7 B-tree, entries with keys (4,40,23,50,11,34,62,78,66,22,90,59,25,72,64,77,39,12), in this order.