Alex Sadler – Homework 5

Problem 1

Key Values	Probe Sequence
43	**0**
23	**6**
1	**3**
0	**1**
15	**7**
31	**2**
4	**9**
7	**5**
11	**5, 6, 7, 8**
3	**7, 8, 9, 10**
5	**0, 1, 2, 3, 4**
9	**10, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9**

	Final Hash Table Contents
0	**43**
1	**0**
2	**31**
3	**1**
4	**5**
5	**7**
6	**23**
7	**15**
8	**11**
9	**4**
10	**3**

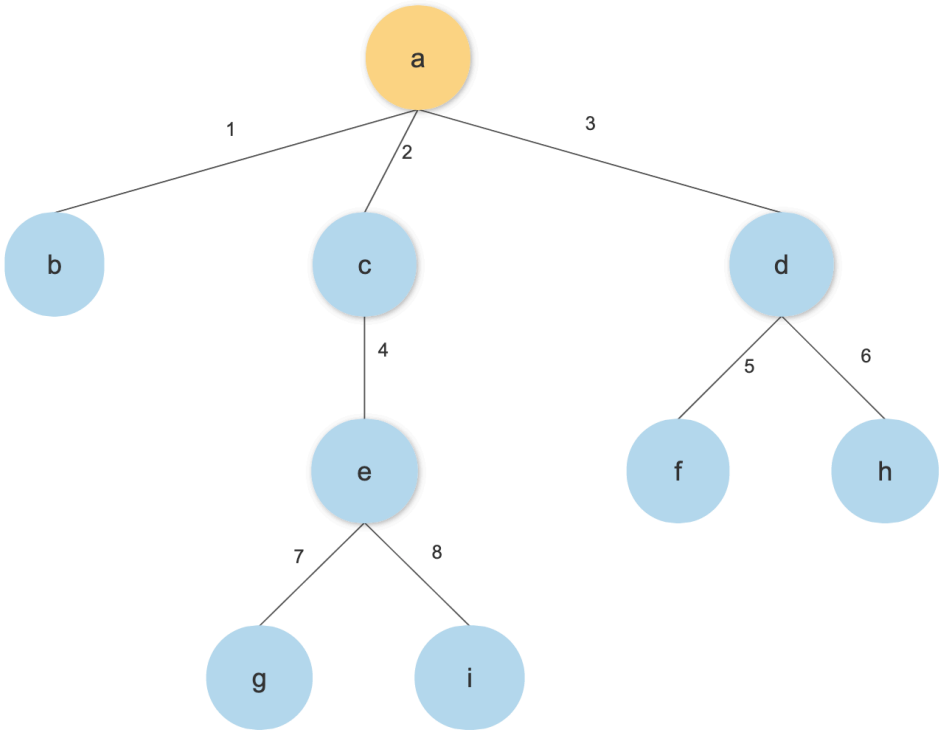
Problem 2

Proof by induction:

- Base case:
  - The children of the root items[0] are located at items[1] = items[2*0 + 1] (left child) and at items[2] = items[2*0 + 2] (right child). The parent of both of these children is located at items[0] = items[⌊(1-1)/2⌋] = items[⌊(2-1)/2⌋]. Thus, the base case is true.
- Assumption:
  - Assume that children of items[k] are items[2k+1] (left child) and items[2k+2] (right child) and that the parent of any child node located at k is located at items[⌊(k-1)/2⌋].
- Inductive step:
  - By assumption, the children of items[k] are items[2k+1] and items[2k+2]. The following elements after these children are located at 2k+3 and 2k+4. For 2k+3, we have 2k+3 = 2(k+1) + 1, and for 2k+4, we have 2k+4=2(k+1) + 2. Thus, for a parent node located at k+1, its children are located at 2(k+1)+1 and 2(k+1)+2. For these children, their parent is located at ⌊((2(k+1)+1)-1)/2⌋ = ⌊((2(k+1)+2)-1)/2⌋ = k+1. Thus, the proof is complete.

Problem 3

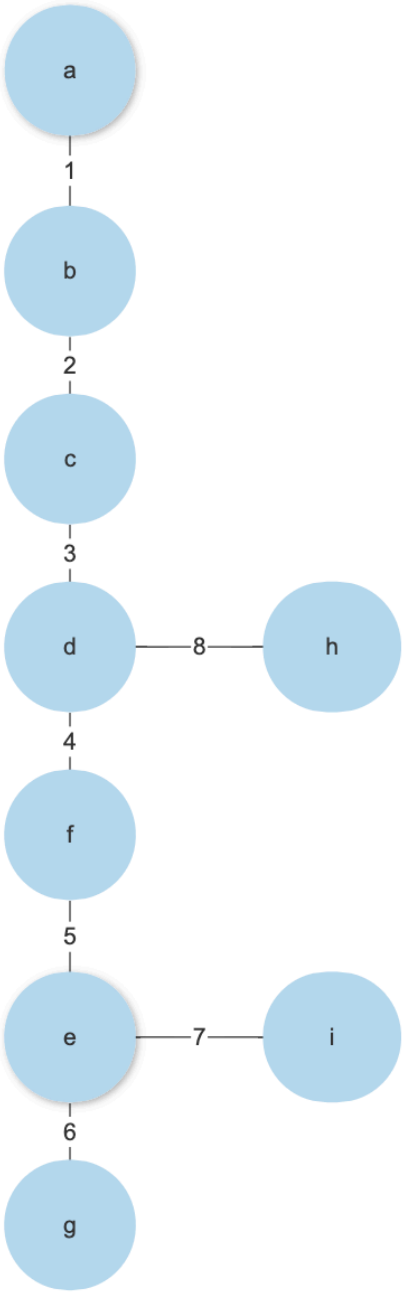
3. BFS traversal tree (with order of search commented)



BFS

Node Visited	Queue (front to back)
a	a
	(empty)
b	b
c	b c
d	b c d
	c d
	d
e	d e
	e
f	e f
h	e f h
	f g
g	h f g
i	h f g i
	f g i
	g i
	i
	(empty)

4. DFS traversal tree (with order of search commented)

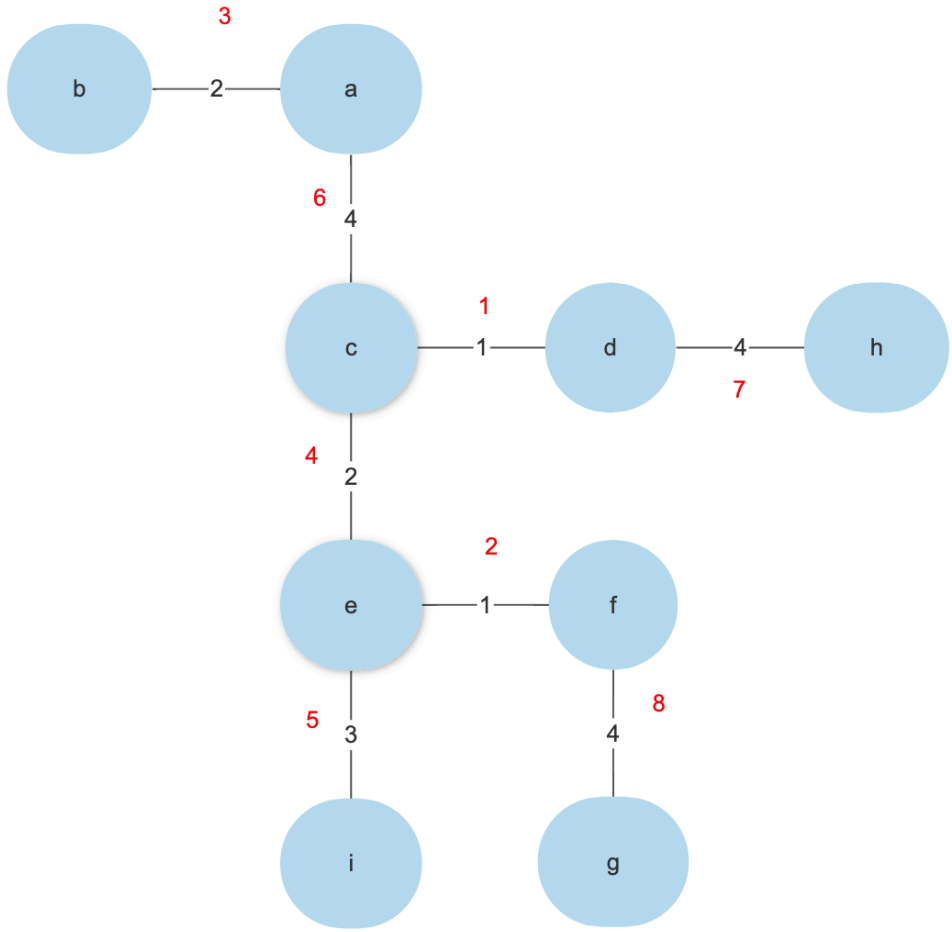


DFS

Node Visited	Stack (bottom to top)
a	a
b	a b
c	a b c
d	a b c d
f	a b c d f
e	a b c d f e
g	a b c d f e g
(backtrack)	a b c d f e
i	a b c d f e i
(backtrack)	a b c d f e
(backtrack)	a b c d f
(backtrack)	a b c d
h	a b c d h
(backtrack)	a b c d
(backtrack)	a b c
(backtrack)	a b
(backtrack)	a

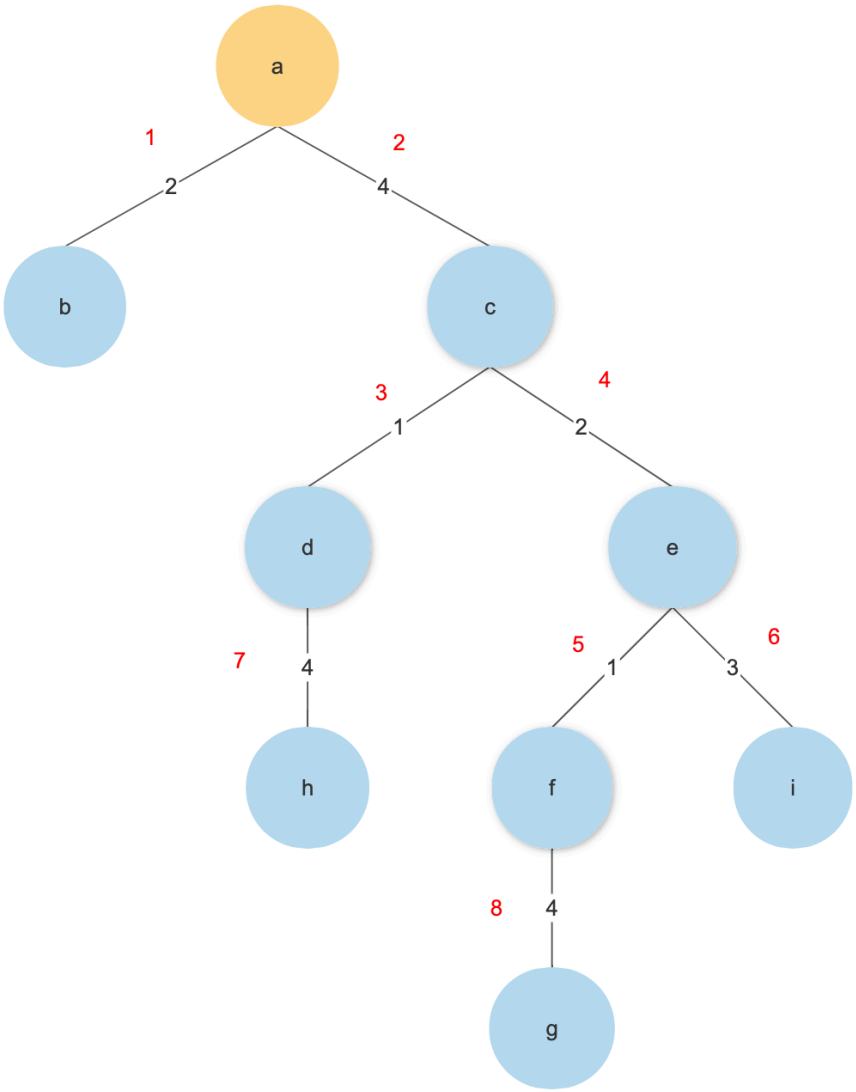
5. Minimum spanning trees (weights are numbers in black, and order of edges established are in red)

Kruskal MST:



Alex Sadler – Homework 5

Prim MST:



Alex Sadler – Homework 5

Extra Credit 1:

The heap sort algorithm has performance comparable to merge and quick sort. It's worst, average, and best case performance are all fairly similar and don't vary largely like the performance of the other algorithms do. Heap does take more moves and comparisons on average than merge and quick sort do, but that is also partly due to having to convert the array to a heap.