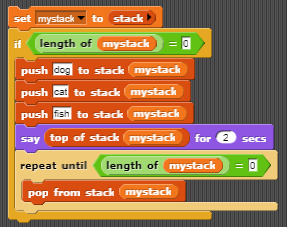
1.

Colours = [“red”, “green”, “blue”] # list of colours

Colours[0] = “purple” # updates first element of list

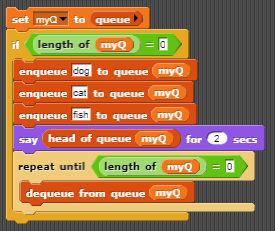
Del Colours[1] # delete second element

2a. S = [“a”, “c”]

2b. Q = [“c”, “d”]

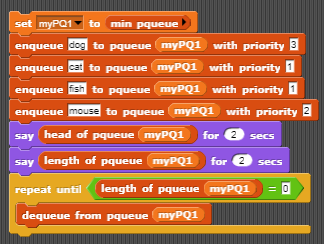
Create a stack call mystack, check if it is empty, add [“dog”, “cat”, “fish”] to the stack.

Look at the top of the stack then empty the stack.



Create a queue call myQ, check if it is empty, enqueue [“dog”, “cat”, “fish”].

Look at the head of the queue and dequeue all elements in the queue.



Create a minimum priority queue called myPQ1, add dog 3 cat 1 fish 1 mouse 2.

Output the head of the queue and length of the queue.

Dequeue all elements from the priority queue.



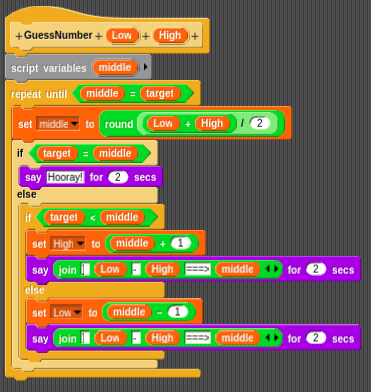
Create a dictionary myD.

Add dog woof, cat meow, lion roar to the dict.

Check the key “cat” is in the dictionary.

Check the key “fish” is in the dictionary.

Page 27:



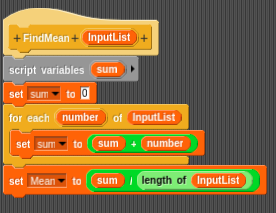
set a middle value to the low value plus the high value divided by 2 rounded.

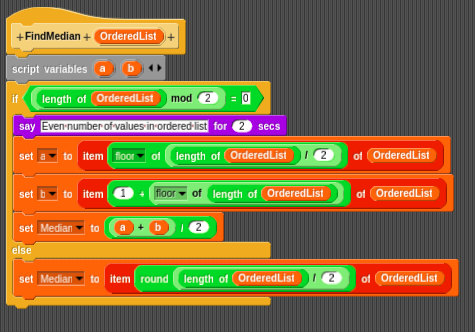
if the middle value is the target value then halt.

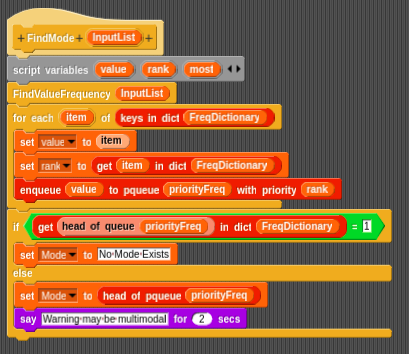
If the middle value is greater than the target value then

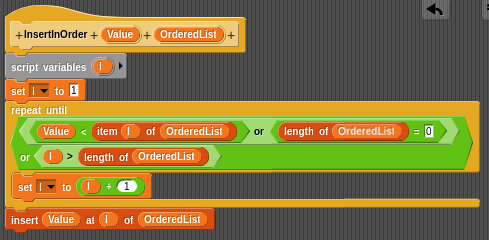


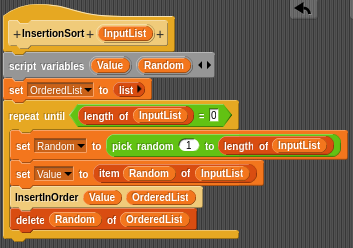
Page 30 & 31:

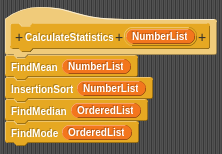




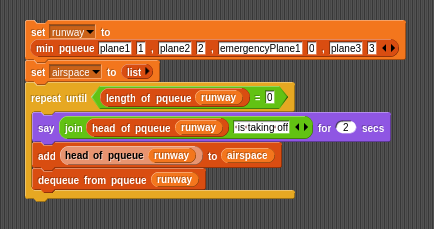








Question 4 page 24



Page 36 & 37:

1. a) 65

b) 9

c) MissScarlet = “Guilty”

1. 

A = [2,5,5,5]

B = [2,2,3,5,5,7]

CountA = {}

CountB = {}

Common = []

for i in A:

if i in CountA:

CountA[i] = CountA[i] + 1

else:

CountA[i] = 1

for i in B:

if i in CountB:

CountB[i] = CountB[i] + 1

else:

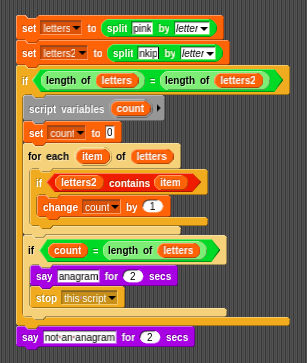
CountB[i] = 1

for i in CountA:

Num = [i] \* min([CountA[i], CountB[i]])

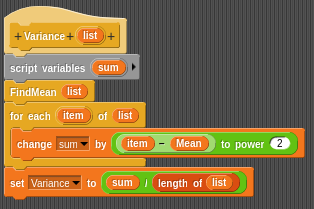
Common = Common + Num

print(Common)

5. 

make a list of letters from both words then compare the lists of both letters

4. a) done above

b) 

Chapter 4

1. a) 8

b) 10

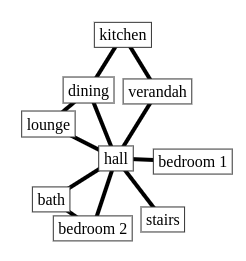
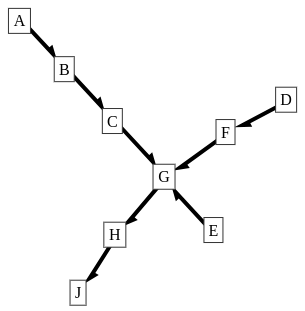
c) {a-c, c-d, d-h}

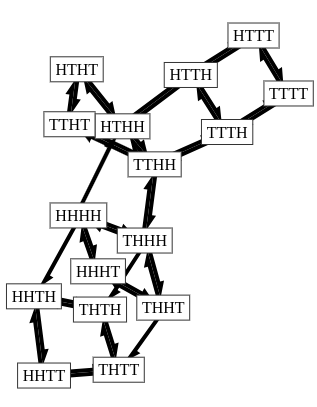
d) {a-b, b-c, c-f, f-a}

e) c & d have a degree of 4

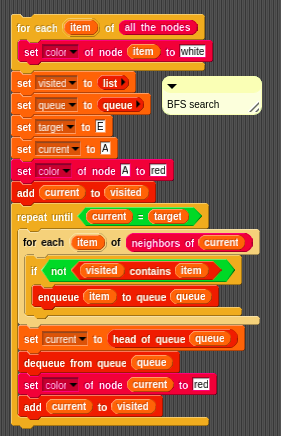
f) c-d

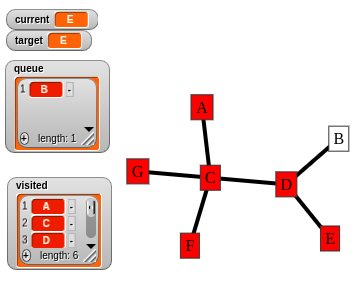
g) c

1. 
2. 
3. The graph needs to be directed since there are states you can advance to but you can’t advance back from.
4. Not sure if this is right

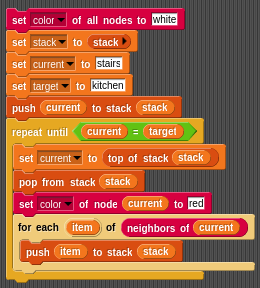


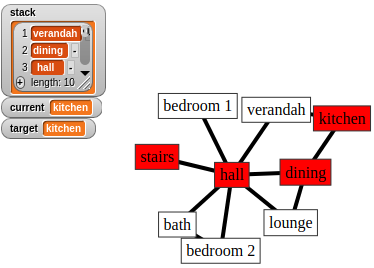
BFS



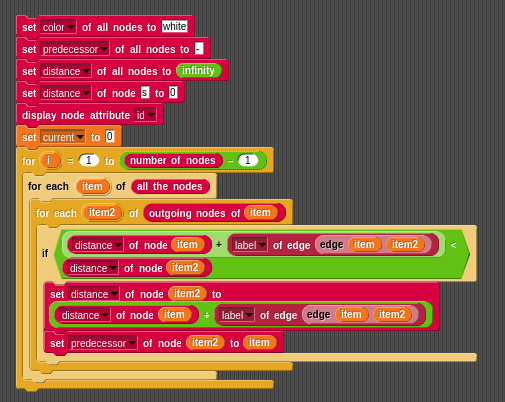


DFS





Bellman-Ford algorithm



Prim’s

