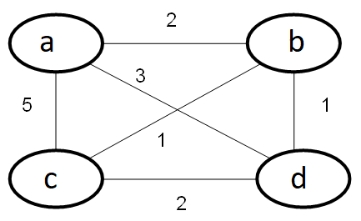
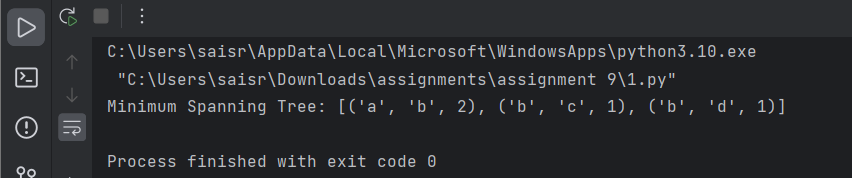
ASSIGNMENT 24.06.24

1. Apply Prim’s algorithm to solve the minimum spanning tree for the given graph. Also compute the total cost of all edges.

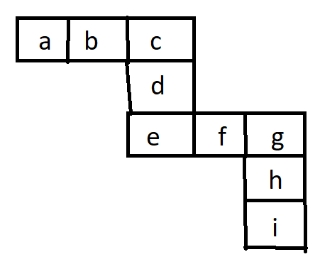


import heapq  
  
  
def prim(graph, start\_node):  
   
 mst = []  
 visited = set([start\_node])  
  
 edges = [(cost, start\_node, to) for to, cost in graph[start\_node].items()]  
 heapq.heapify(edges)  
  
   
 while edges:  
 cost, frm, to = heapq.heappop(edges)  
  
 if to in visited:  
 continue  
  
   
 mst.append((frm, to, cost))  
 visited.add(to)  
  
   
 for to\_next, cost\_next in graph[to].items():  
 if to\_next not in visited:  
 heapq.heappush(edges, (cost\_next, to, to\_next))  
  
 return mst  
  
  
graph = {  
 'a': {'c': 5, 'b': 2},  
 'b': {'a': 3, 'c': 1, 'd': 1},  
 'c': {'a': 5, 'b': 1},  
 'd': {'b': 1}  
}  
  
  
start\_node = 'a'  
  
  
mst = prim(graph, start\_node)  
print("Minimum Spanning Tree:", mst)



1. To Compute the sum of Subsets for the following graph and then satisfy the given constraints.

Set S { } =(a, b, c, d, e, f, g, h, i) Values used are V{i}=(1, 2, 3……….9)

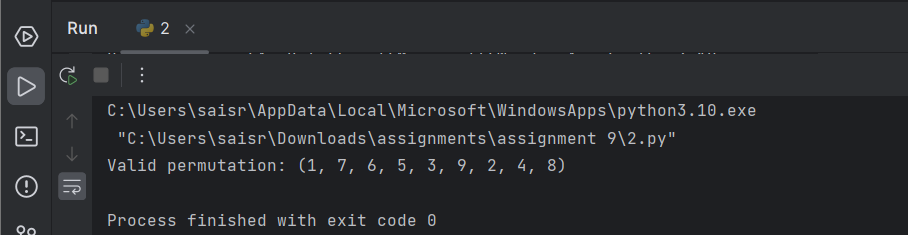


Used all values Only one time

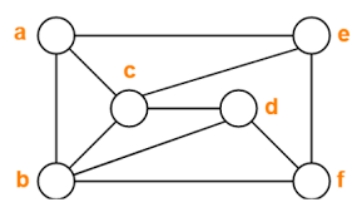
Constraints hold such as

a + b + c = c +d + e = e + f + g = g + h + i

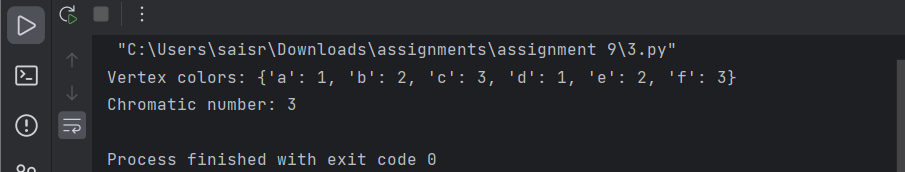
import itertools  
  
def check\_constraints(permutation):  
 a, b, c, d, e, f, g, h, i = permutation  
 sum1 = a + b + c  
 sum2 = c + d + e  
 sum3 = e + f + g  
 sum4 = g + h + i  
 return sum1 == sum2 == sum3 == sum4  
  
  
values = [1, 2, 3, 4, 5, 6, 7, 8, 9]  
  
  
for perm in itertools.permutations(values):  
 if check\_constraints(perm):  
 print(f"Valid permutation: {perm}")  
 break  
else:  
 print("No valid permutation found.")



1. Calculate the chromatic no for the following Graph coloring.



def greedy\_coloring(graph):  
   
 result = {}  
  
 for vertex in graph:  
   
 adjacent\_colors = set(result.get(neighbor) for neighbor in graph[vertex])  
  
   
 color = 1  
 while color in adjacent\_colors:  
 color += 1  
  
   
 result[vertex] = color  
  
 return result  
  
  
  
graph = {  
 'a': ['b', 'c', 'e'],  
 'b': ['a', 'c', 'd'],  
 'c': ['a', 'b', 'd', 'e'],  
 'd': ['b', 'c', 'e', 'f'],  
 'e': ['a', 'c', 'd', 'f'],  
 'f': ['d', 'e']  
}  
  
  
coloring = greedy\_coloring(graph)  
  
  
print("Vertex colors:", coloring)  
  
  
chromatic\_number = max(coloring.values())  
print("Chromatic number:", chromatic\_number)



4.Consider a set S = ( 5, 10, 12, 13, 15, 18) and d=30. Solve it for obtaining a sum of subset.

def subset\_sum(S, d):  
 def backtrack(start, current\_subset, current\_sum):  
 if current\_sum == d:  
 result.append(current\_subset[:])  
 return  
 if current\_sum > d:  
 return  
  
 for i in range(start, len(S)):  
 current\_subset.append(S[i])  
 backtrack(i + 1, current\_subset, current\_sum + S[i])  
 current\_subset.pop()  
  
 result = []  
 S.sort()   
 backtrack(0, [], 0)  
 return result  
  
  
  
S = [5, 10, 12, 13, 15, 18]  
d = 30  
  
  
subsets = subset\_sum(S, d)  
print("Subsets that sum to", d, ":", subsets)

