Evaluation Only. Created with Aspose.PDF. Copyright 2002-2019 Aspose Pty Ltd.

Đề 4.

Câu 1:

def BFS(initialState, goal):

frontier = [initialState]

explored = []

while frontier:

state = frontier.pop(0)

explored.append(state)

if state == goal:

return explored

for neighbor in graph[state]:

if neighbor not in frontier + explored:

frontier.append(neighbor)

return False

if \_\_name\_\_ == "\_\_main\_\_":

V = ['S', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']

E = [('S', 'A'), ('S', 'B'), ('S', 'C'),

('A', 'D'), ('A', 'B'),

('B', 'D'), ('B', 'F'), ('B', 'G'),

('C', 'F'),

('D', 'E'),

('E', 'G'), ('E', 'F'),

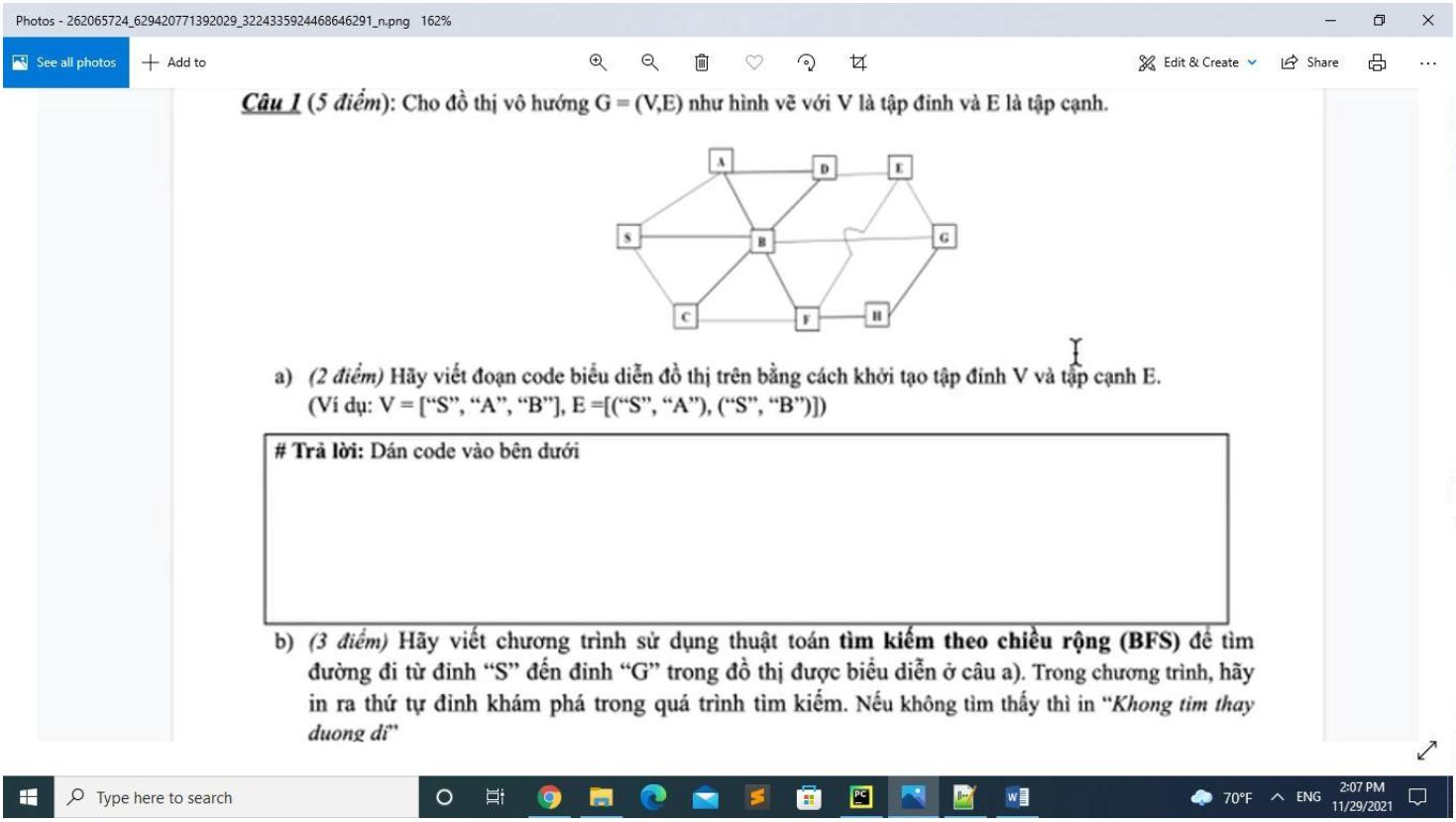
('F', 'H'),

('G', 'H')]

graph = {}

for node in V:

graph[node] = []



Evaluation Only. Created with Aspose.PDF. Copyright 2002-2019 Aspose Pty Ltd.

for edge in E:

firstNode = edge[0]

secondNode = edge[1]

firstList = graph[firstNode]

firstList.append(secondNode)

graph[firstNode] = firstList

secondList = graph[secondNode]

secondList.append(firstNode)

graph[secondNode] = secondList

result = BFS('S', 'G');

if result:

print("explored: ", result)

else:

print('Khong tim thay duong di');

Câu 2.

a. Mô tả thuật toán:

B1: Khởi tạo ngẫu nhiên các trọng tâm

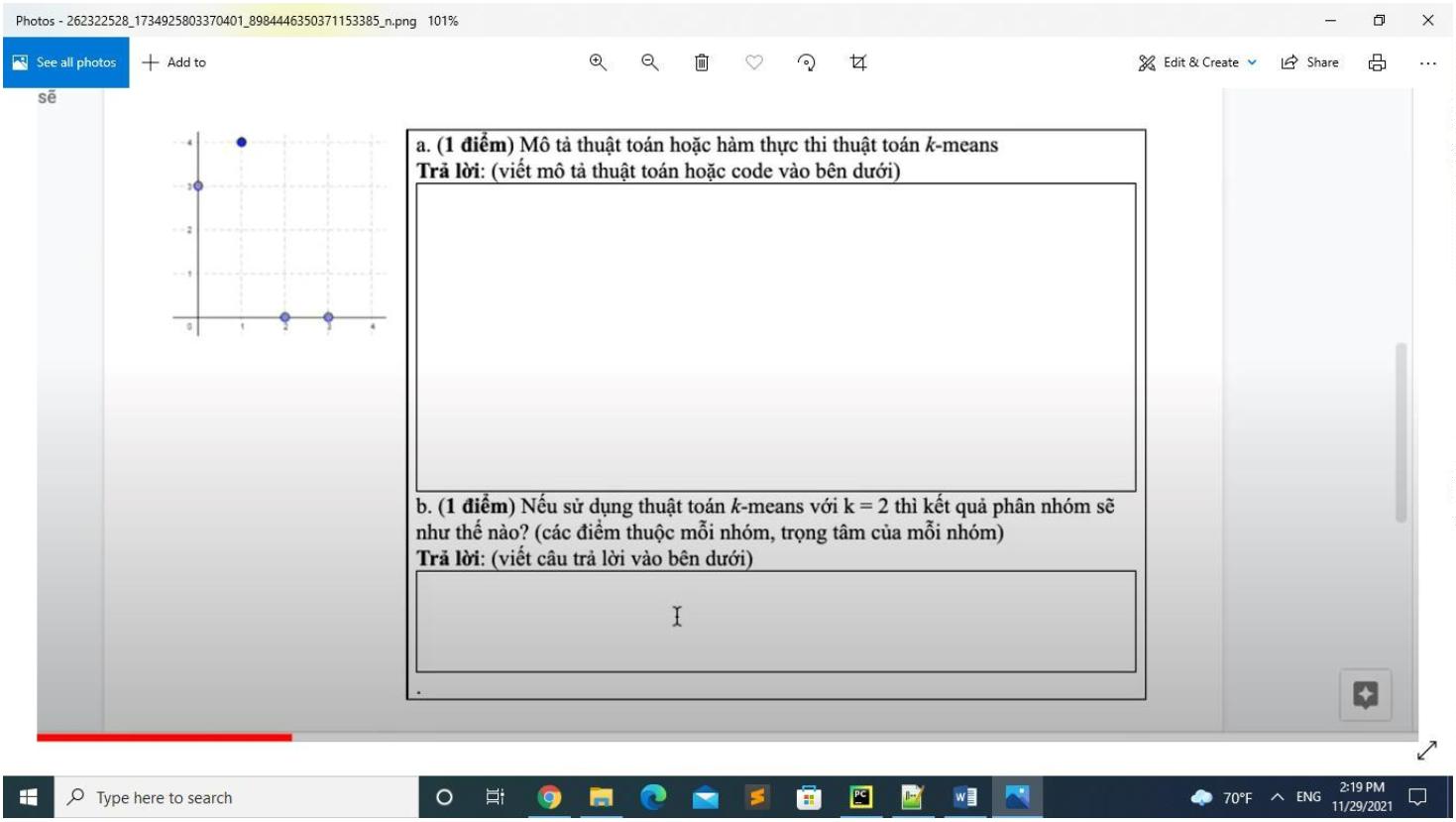
B2: Nhóm dữ liệu: tính khoảng cách giữa các điểm và các tâm. Sau đó xếp chúng vào một cụm sao

cho khoảng cách từ điểm đó tới tâm của cụm là nhỏ nhất.

B3: Cập nhật trọng tâm

Nếu không có sự thay đổi thì dừng thuật toán

B4: Quay lại B2



Evaluation Only. Created with Aspose.PDF. Copyright 2002-2019 Aspose Pty Ltd.

b. Cụm 1: trọng tâm (2.5, 0) gồm hai điểm (2,0) (3,0)

Cụm 2: trọng tâm (0.5, 3.5) gồm hai điểm (0,3) (1,4)

Câu 3:

import math

def grad(x):

return 2 \* (math.exp(2\*x) - 4/math.exp(2\*x))

def cost(x):

temp = math.exp(x) - 2/math.exp(x)

return pow(temp, 2)

def myGD(eta, x0):

x = [x0]

for it in range(100):

x\_new = x[-1] - eta\*grad(x[-1])

if abs(grad(x\_new)) < 1e-3:

break

x.append(x\_new)

return (x, it)

if \_\_name\_\_ == "\_\_main\_\_":

(x1, it1) = myGD(.1, 0.2)

print('Solution x1 = %f, cost = %f, obtained after %d iterations'%(x1[-

1], cost(x1[-1]), it1))

