PRÁCTICA 6

ALGORITMOS DE GRAFOS. PARTE I

## Alumno: Robles Martínez Héctor

## Número de Cuenta UNAM: 317356548

## Semestre: Tercer semestre de Ingeniería en Computación (2016)

## Semestre Actual: 2021-1

## Profesor: Jorge A. Solano G.

Objetivo:

Conocer las formas de representar un grafo e identificará las características necesarias

para entender el algoritmo de búsqueda por expansión.

Instrucciones:

• Implementar en lenguaje Python el algoritmo de búsqueda por expansión dentro de

un grafo dirigido y no dirigido, con costo y sin costo, a partir de un nodo fuente s.

• Obtener la complejidad algorítmica del algoritmo búsqueda por expansión.

• Crear las gráficas de la complejidad que tiene el algoritmo de búsqueda por expansión

dentro de un grafo para el mejor caso, el peor caso y el caso promedio.

Resultados obtenidos:

from graph import \*

from expansionFind import \*

def bestCaseTime(data):

size = len(data)

directed = True

numEdges = size

cost = 1

grafo = fromCsvFile(data, directed, 0, cost)

root = 1

iterations = expansionFind(grafo, root)

return iterations

def baseCaseTime(data):

size = len(data)

directed = random.getrandbits(1)

numEdges = random.randint(0, size)

cost = 1

grafo = fromCsvFile(data, directed, numEdges, cost)

root = 1

iterations = expansionFind(grafo, root)

return iterations

def worseCaseTime(data):

size = len(data)

directed = False

numEdges = size

cost = 1

grafo = fromCsvFile(data, directed, size, cost, True)

root = 1

iterations = expansionFind(grafo, root)

return iterations

if \_\_name\_\_ == '\_\_main\_\_':

data = [1, 2, 3, 4, 5]

print("Best case by time: ")

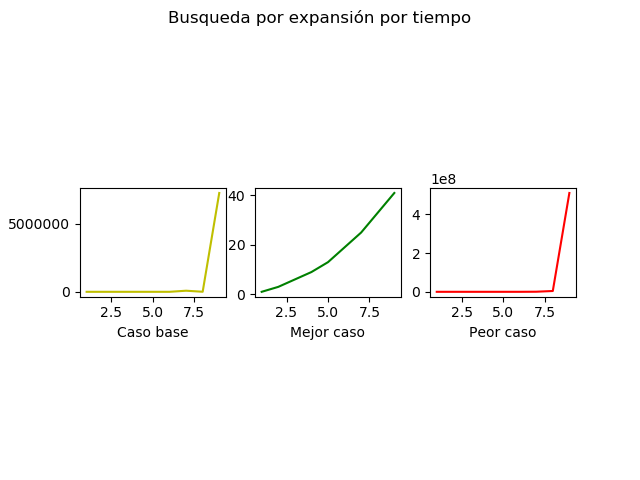
bestCaseTime(data)

print("Base case by time: ")

baseCaseTime(data)

print("worse case by time: ")

worseCaseTime(data)



Conclusiones:

La búsqueda por expansión puede llegar a ser algo compleja de implementar si no se tiene un buen orden