#Ameeth Kanawaday 4430

#Assignment A4

import sys

from xml.dom import minidom

from mpi4py import MPI

import numpy

import matplotlib.pyplot as plt; plt.rcdefaults()

import numpy as np

import matplotlib.pyplot as plt

class NQueenProblem:

def \_\_init\_\_(self,boardSize):

self.size = boardSize

self.columns = []

self.placesChecked = 0

self.numBacktracks = 0

def chessBoard(self):

for row in range(0,8):

for col in range(0,8):

queenPos = queen.columns.index(row)

if col == queenPos:

print'| Q |-',

else:

print'| |-',

print('\n')

def place(self,startRow = 0):

if len(self.columns) == self.size:

print('Solution found!\nSize of Chessboard :: '+str(self.size))

print('Total number of positions checked :: '+str(self.placesChecked))

print('Total number of backtracks :: '+str(self.numBacktracks))

print('\n:: :: Row Positions :: ::')

print(self.columns)

else:

for row in range(startRow,self.size):

if self.checkPosition(len(self.columns),row) is True:

self.columns.append(row)

self.placesChecked += 1

return self.place()

self.numBacktracks += 1

return self.place(self.columns.pop()+1)

def checkPosition(self,col,row):

for positionedRow in self.columns:

positionedCol = self.columns.index(positionedRow)

if row == positionedRow or col == positionedCol:

return False

elif row - col == positionedRow - positionedCol or row + col == positionedRow + positionedCol:

return False

return True

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

rank = MPI.COMM\_WORLD.Get\_rank()

n = 8

Buffer = []

queen = NQueenProblem(n)

doc = minidom.parse('queenPosition.xml')

#Sending initial configuration

if rank == 0:

start = MPI.Wtime()

#Slave1

position = doc.getElementsByTagName("slave1")[0]

pos = int(position.firstChild.data)

print('Slave\'1 Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=100)

position = doc.getElementsByTagName("slave2")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=200)

position = doc.getElementsByTagName("slave3")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=300)

position = doc.getElementsByTagName("slave4")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=400)

position = doc.getElementsByTagName("slave5")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=500)

position = doc.getElementsByTagName("slave6")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=600)

position = doc.getElementsByTagName("slave7")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=700)

position = doc.getElementsByTagName("slave8")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=800)

MPI.COMM\_WORLD.Barrier()

#Parallel Configuration

if rank == 1:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=100)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=200)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=300)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=400)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=500)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=600)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=700)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=800)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

MPI.COMM\_WORLD.Barrier()

#Gathering and printing output

if rank == 0:

end = MPI.Wtime()

timeMaster = end-start

print('Time :: '+str(end-start))

target = open("TimeAnalysis", 'a')

target.write(str(end-start))

target.write("\n")

target.close()

//eightQueenPlot\_3.py

import sys

from xml.dom import minidom

from mpi4py import MPI

import numpy

import matplotlib.pyplot as plt; plt.rcdefaults()

import numpy as np

import matplotlib.pyplot as plt

class NQueenProblem:

def \_\_init\_\_(self,boardSize):

self.size = boardSize

self.columns = []

self.placesChecked = 0

self.numBacktracks = 0

def chessBoard(self):

for row in range(0,8):

for col in range(0,8):

queenPos = queen.columns.index(row)

if col == queenPos:

print'| Q |-',

else:

print'| |-',

print('\n')

def place(self,startRow = 0):

if len(self.columns) == self.size:

print('Solution found!\nSize of Chessboard :: '+str(self.size))

print('Total number of positions checked :: '+str(self.placesChecked))

print('Total number of backtracks :: '+str(self.numBacktracks))

print('\n:: :: Row Positions :: ::')

print(self.columns)

else:

for row in range(startRow,self.size):

if self.checkPosition(len(self.columns),row) is True:

self.columns.append(row)

self.placesChecked += 1

return self.place()

self.numBacktracks += 1

return self.place(self.columns.pop()+1)

def checkPosition(self,col,row):

for positionedRow in self.columns:

positionedCol = self.columns.index(positionedRow)

if row == positionedRow or col == positionedCol:

return False

elif row - col == positionedRow - positionedCol or row + col == positionedRow + positionedCol:

return False

return True

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

rank = MPI.COMM\_WORLD.Get\_rank()

n = 8

Buffer = []

queen = NQueenProblem(n)

doc = minidom.parse('queenPosition.xml')

#Sending initial configuration

if rank == 0:

start = MPI.Wtime()

#Slave1

position = doc.getElementsByTagName("slave1")[0]

pos = int(position.firstChild.data)

print('Slave\'1 Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=100)

position = doc.getElementsByTagName("slave2")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=200)

position = doc.getElementsByTagName("slave3")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=300)

position = doc.getElementsByTagName("slave4")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=400)

position = doc.getElementsByTagName("slave5")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=2,tag=500)

position = doc.getElementsByTagName("slave6")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=2,tag=600)

position = doc.getElementsByTagName("slave7")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=2,tag=700)

position = doc.getElementsByTagName("slave8")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=2,tag=800)

MPI.COMM\_WORLD.Barrier()

#Parallel Configuration

if rank == 1:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=100)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=200)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=300)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=400)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 2:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=500)

print('\n\n:: :: Rank 2 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=600)

print('\n\n:: :: Rank 2 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=700)

print('\n\n:: :: Rank 2 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=800)

print('\n\n:: :: Rank 2 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

MPI.COMM\_WORLD.Barrier()

#Gathering and printing output

if rank == 0:

end = MPI.Wtime()

timeMaster = end-start

print('Time :: '+str(end-start))

target = open("TimeAnalysis", 'a')

target.write(str(end-start))

target.write("\n")

target.close()

//eightQueenPlot\_4.py

import sys

from xml.dom import minidom

from mpi4py import MPI

import numpy

import matplotlib.pyplot as plt; plt.rcdefaults()

import numpy as np

import matplotlib.pyplot as plt

class NQueenProblem:

def \_\_init\_\_(self,boardSize):

self.size = boardSize

self.columns = []

self.placesChecked = 0

self.numBacktracks = 0

def chessBoard(self):

for row in range(0,8):

for col in range(0,8):

queenPos = queen.columns.index(row)

if col == queenPos:

print'| Q |-',

else:

print'| |-',

print('\n')

def place(self,startRow = 0):

if len(self.columns) == self.size:

print('Solution found!\nSize of Chessboard :: '+str(self.size))

print('Total number of positions checked :: '+str(self.placesChecked))

print('Total number of backtracks :: '+str(self.numBacktracks))

print('\n:: :: Row Positions :: ::')

print(self.columns)

else:

for row in range(startRow,self.size):

if self.checkPosition(len(self.columns),row) is True:

self.columns.append(row)

self.placesChecked += 1

return self.place()

self.numBacktracks += 1

return self.place(self.columns.pop()+1)

def checkPosition(self,col,row):

for positionedRow in self.columns:

positionedCol = self.columns.index(positionedRow)

if row == positionedRow or col == positionedCol:

return False

elif row - col == positionedRow - positionedCol or row + col == positionedRow + positionedCol:

return False

return True

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

rank = MPI.COMM\_WORLD.Get\_rank()

n = 8

Buffer = []

queen = NQueenProblem(n)

doc = minidom.parse('queenPosition.xml')

#Sending initial configuration

if rank == 0:

start = MPI.Wtime()

#Slave1

position = doc.getElementsByTagName("slave1")[0]

pos = int(position.firstChild.data)

print('Slave\'1 Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=100)

position = doc.getElementsByTagName("slave2")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=2,tag=200)

position = doc.getElementsByTagName("slave3")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=3,tag=300)

position = doc.getElementsByTagName("slave4")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=400)

position = doc.getElementsByTagName("slave5")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=2,tag=500)

position = doc.getElementsByTagName("slave6")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=3,tag=600)

position = doc.getElementsByTagName("slave7")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=700)

position = doc.getElementsByTagName("slave8")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=2,tag=800)

MPI.COMM\_WORLD.Barrier()

#Parallel Configuration

if rank == 1:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=100)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=400)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=700)

print('\n\n:: :: Rank 2 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 2:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=200)

print('\n\n:: :: Rank 2 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=500)

print('\n\n:: :: Rank 2 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=800)

print('\n\n:: :: Rank 2 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 3:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=300)

print('\n\n:: :: Rank 3 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=600)

print('\n\n:: :: Rank 3 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

MPI.COMM\_WORLD.Barrier()

#Gathering and printing output

if rank == 0:

end = MPI.Wtime()

timeMaster = end-start

print('Time :: '+str(end-start))

target = open("TimeAnalysis", 'a')

target.write(str(end-start))

target.write("\n")

target.close()

//eightQueenPlot\_5.py

import sys

from xml.dom import minidom

from mpi4py import MPI

import numpy

import matplotlib.pyplot as plt; plt.rcdefaults()

import numpy as np

import matplotlib.pyplot as plt

class NQueenProblem:

def \_\_init\_\_(self,boardSize):

self.size = boardSize

self.columns = []

self.placesChecked = 0

self.numBacktracks = 0

def chessBoard(self):

for row in range(0,8):

for col in range(0,8):

queenPos = queen.columns.index(row)

if col == queenPos:

print'| Q |-',

else:

print'| |-',

print('\n')

def place(self,startRow = 0):

if len(self.columns) == self.size:

print('Solution found!\nSize of Chessboard :: '+str(self.size))

print('Total number of positions checked :: '+str(self.placesChecked))

print('Total number of backtracks :: '+str(self.numBacktracks))

print('\n:: :: Row Positions :: ::')

print(self.columns)

else:

for row in range(startRow,self.size):

if self.checkPosition(len(self.columns),row) is True:

self.columns.append(row)

self.placesChecked += 1

return self.place()

self.numBacktracks += 1

return self.place(self.columns.pop()+1)

def checkPosition(self,col,row):

for positionedRow in self.columns:

positionedCol = self.columns.index(positionedRow)

if row == positionedRow or col == positionedCol:

return False

elif row - col == positionedRow - positionedCol or row + col == positionedRow + positionedCol:

return False

return True

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

rank = MPI.COMM\_WORLD.Get\_rank()

n = 8

Buffer = []

queen = NQueenProblem(n)

doc = minidom.parse('queenPosition.xml')

#Sending initial configuration

if rank == 0:

start = MPI.Wtime()

#Slave1

position = doc.getElementsByTagName("slave1")[0]

pos = int(position.firstChild.data)

print('Slave\'1 Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=100)

position = doc.getElementsByTagName("slave2")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=2,tag=200)

position = doc.getElementsByTagName("slave3")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=3,tag=300)

position = doc.getElementsByTagName("slave4")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=4,tag=400)

position = doc.getElementsByTagName("slave5")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=500)

position = doc.getElementsByTagName("slave6")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=2,tag=600)

position = doc.getElementsByTagName("slave7")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=3,tag=700)

position = doc.getElementsByTagName("slave8")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=4,tag=800)

MPI.COMM\_WORLD.Barrier()

#Parallel Configuration

if rank == 1:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=100)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=500)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 2:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=200)

print('\n\n:: :: Rank 2 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=600)

print('\n\n:: :: Rank 2 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 3:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=300)

print('\n\n:: :: Rank 3 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=700)

print('\n\n:: :: Rank 3 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 4:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=400)

print('\n\n:: :: Rank 4 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=800)

print('\n\n:: :: Rank 4 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

MPI.COMM\_WORLD.Barrier()

#Gathering and printing output

if rank == 0:

end = MPI.Wtime()

timeMaster = end-start

print('Time :: '+str(end-start))

target = open("TimeAnalysis", 'a')

target.write(str(end-start))

target.write("\n")

target.close()

//eightQueenPlot\_6.py

import sys

from xml.dom import minidom

from mpi4py import MPI

import numpy

import matplotlib.pyplot as plt; plt.rcdefaults()

import numpy as np

import matplotlib.pyplot as plt

class NQueenProblem:

def \_\_init\_\_(self,boardSize):

self.size = boardSize

self.columns = []

self.placesChecked = 0

self.numBacktracks = 0

def chessBoard(self):

for row in range(0,8):

for col in range(0,8):

queenPos = queen.columns.index(row)

if col == queenPos:

print'| Q |-',

else:

print'| |-',

print('\n')

def place(self,startRow = 0):

if len(self.columns) == self.size:

print('Solution found!\nSize of Chessboard :: '+str(self.size))

print('Total number of positions checked :: '+str(self.placesChecked))

print('Total number of backtracks :: '+str(self.numBacktracks))

print('\n:: :: Row Positions :: ::')

print(self.columns)

else:

for row in range(startRow,self.size):

if self.checkPosition(len(self.columns),row) is True:

self.columns.append(row)

self.placesChecked += 1

return self.place()

self.numBacktracks += 1

return self.place(self.columns.pop()+1)

def checkPosition(self,col,row):

for positionedRow in self.columns:

positionedCol = self.columns.index(positionedRow)

if row == positionedRow or col == positionedCol:

return False

elif row - col == positionedRow - positionedCol or row + col == positionedRow + positionedCol:

return False

return True

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

rank = MPI.COMM\_WORLD.Get\_rank()

n = 8

Buffer = []

queen = NQueenProblem(n)

doc = minidom.parse('queenPosition.xml')

#Sending initial configuration

if rank == 0:

start = MPI.Wtime()

#Slave1

position = doc.getElementsByTagName("slave1")[0]

pos = int(position.firstChild.data)

print('Slave\'1 Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=100)

position = doc.getElementsByTagName("slave2")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=2,tag=200)

position = doc.getElementsByTagName("slave3")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=3,tag=300)

position = doc.getElementsByTagName("slave4")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=4,tag=400)

position = doc.getElementsByTagName("slave5")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=5,tag=500)

position = doc.getElementsByTagName("slave6")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=600)

position = doc.getElementsByTagName("slave7")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=2,tag=700)

position = doc.getElementsByTagName("slave8")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=3,tag=800)

MPI.COMM\_WORLD.Barrier()

#Parallel Configuration

if rank == 1:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=100)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=600)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 2:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=200)

print('\n\n:: :: Rank 2 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=700)

print('\n\n:: :: Rank 2 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 3:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=300)

print('\n\n:: :: Rank 3 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=800)

print('\n\n:: :: Rank 3 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 4:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=400)

print('\n\n:: :: Rank 4 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 5:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=500)

print('\n\n:: :: Rank 5 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

MPI.COMM\_WORLD.Barrier()

#Gathering and printing output

if rank == 0:

end = MPI.Wtime()

timeMaster = end-start

print('Time :: '+str(end-start))

target = open("TimeAnalysis", 'a')

target.write(str(end-start))

target.write("\n")

target.close()

//eightQueenPlot\_7.py

import sys

from xml.dom import minidom

from mpi4py import MPI

import numpy

import matplotlib.pyplot as plt; plt.rcdefaults()

import numpy as np

import matplotlib.pyplot as plt

class NQueenProblem:

def \_\_init\_\_(self,boardSize):

self.size = boardSize

self.columns = []

self.placesChecked = 0

self.numBacktracks = 0

def chessBoard(self):

for row in range(0,8):

for col in range(0,8):

queenPos = queen.columns.index(row)

if col == queenPos:

print'| Q |-',

else:

print'| |-',

print('\n')

def place(self,startRow = 0):

if len(self.columns) == self.size:

print('Solution found!\nSize of Chessboard :: '+str(self.size))

print('Total number of positions checked :: '+str(self.placesChecked))

print('Total number of backtracks :: '+str(self.numBacktracks))

print('\n:: :: Row Positions :: ::')

print(self.columns)

else:

for row in range(startRow,self.size):

if self.checkPosition(len(self.columns),row) is True:

self.columns.append(row)

self.placesChecked += 1

return self.place()

self.numBacktracks += 1

return self.place(self.columns.pop()+1)

def checkPosition(self,col,row):

for positionedRow in self.columns:

positionedCol = self.columns.index(positionedRow)

if row == positionedRow or col == positionedCol:

return False

elif row - col == positionedRow - positionedCol or row + col == positionedRow + positionedCol:

return False

return True

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

rank = MPI.COMM\_WORLD.Get\_rank()

n = 8

Buffer = []

queen = NQueenProblem(n)

doc = minidom.parse('queenPosition.xml')

#Sending initial configuration

if rank == 0:

start = MPI.Wtime()

#Slave1

position = doc.getElementsByTagName("slave1")[0]

pos = int(position.firstChild.data)

print('Slave\'1 Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=100)

position = doc.getElementsByTagName("slave2")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=2,tag=200)

position = doc.getElementsByTagName("slave3")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=3,tag=300)

position = doc.getElementsByTagName("slave4")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=4,tag=400)

position = doc.getElementsByTagName("slave5")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=5,tag=500)

position = doc.getElementsByTagName("slave6")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=6,tag=600)

position = doc.getElementsByTagName("slave7")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=700)

position = doc.getElementsByTagName("slave8")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=2,tag=800)

MPI.COMM\_WORLD.Barrier()

#Parallel Configuration

if rank == 1:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=100)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=700)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 2:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=200)

print('\n\n:: :: Rank 2 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=800)

print('\n\n:: :: Rank 2 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 3:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=300)

print('\n\n:: :: Rank 3 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 4:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=400)

print('\n\n:: :: Rank 4 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 5:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=500)

print('\n\n:: :: Rank 5 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 6:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=600)

print('\n\n:: :: Rank 6 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

MPI.COMM\_WORLD.Barrier()

#Gathering and printing output

if rank == 0:

end = MPI.Wtime()

timeMaster = end-start

print('Time :: '+str(end-start))

target = open("TimeAnalysis", 'a')

target.write(str(end-start))

target.write("\n")

target.close()

//eightQueenPlot\_8.py

import sys

from xml.dom import minidom

from mpi4py import MPI

import numpy

import matplotlib.pyplot as plt; plt.rcdefaults()

import numpy as np

import matplotlib.pyplot as plt

class NQueenProblem:

def \_\_init\_\_(self,boardSize):

self.size = boardSize

self.columns = []

self.placesChecked = 0

self.numBacktracks = 0

def chessBoard(self):

for row in range(0,8):

for col in range(0,8):

queenPos = queen.columns.index(row)

if col == queenPos:

print'| Q |-',

else:

print'| |-',

print('\n')

def place(self,startRow = 0):

if len(self.columns) == self.size:

print('Solution found!\nSize of Chessboard :: '+str(self.size))

print('Total number of positions checked :: '+str(self.placesChecked))

print('Total number of backtracks :: '+str(self.numBacktracks))

print('\n:: :: Row Positions :: ::')

print(self.columns)

else:

for row in range(startRow,self.size):

if self.checkPosition(len(self.columns),row) is True:

self.columns.append(row)

self.placesChecked += 1

return self.place()

self.numBacktracks += 1

return self.place(self.columns.pop()+1)

def checkPosition(self,col,row):

for positionedRow in self.columns:

positionedCol = self.columns.index(positionedRow)

if row == positionedRow or col == positionedCol:

return False

elif row - col == positionedRow - positionedCol or row + col == positionedRow + positionedCol:

return False

return True

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

rank = MPI.COMM\_WORLD.Get\_rank()

n = 8

Buffer = []

queen = NQueenProblem(n)

doc = minidom.parse('queenPosition.xml')

#Sending initial configuration

if rank == 0:

start = MPI.Wtime()

#Slave1

position = doc.getElementsByTagName("slave1")[0]

pos = int(position.firstChild.data)

print('Slave\'1 Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=100)

position = doc.getElementsByTagName("slave2")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=2,tag=200)

position = doc.getElementsByTagName("slave3")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=3,tag=300)

position = doc.getElementsByTagName("slave4")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=4,tag=400)

position = doc.getElementsByTagName("slave5")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=5,tag=500)

position = doc.getElementsByTagName("slave6")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=6,tag=600)

position = doc.getElementsByTagName("slave7")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=7,tag=700)

position = doc.getElementsByTagName("slave8")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=800)

MPI.COMM\_WORLD.Barrier()

#Parallel Configuration

if rank == 1:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=100)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=800)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 2:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=200)

print('\n\n:: :: Rank 2 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 3:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=300)

print('\n\n:: :: Rank 3 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 4:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=400)

print('\n\n:: :: Rank 4 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 5:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=500)

print('\n\n:: :: Rank 5 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 6:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=600)

print('\n\n:: :: Rank 6 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 7:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=700)

print('\n\n:: :: Rank 7 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

MPI.COMM\_WORLD.Barrier()

#Gathering and printing output

if rank == 0:

end = MPI.Wtime()

timeMaster = end-start

print('Time :: '+str(end-start))

target = open("TimeAnalysis", 'a')

target.write(str(end-start))

target.write("\n")

target.close()

//eightQueenPlot\_9.py

import sys

from xml.dom import minidom

from mpi4py import MPI

import numpy

import matplotlib.pyplot as plt; plt.rcdefaults()

import numpy as np

import matplotlib.pyplot as plt

class NQueenProblem:

def \_\_init\_\_(self,boardSize):

self.size = boardSize

self.columns = []

self.placesChecked = 0

self.numBacktracks = 0

def chessBoard(self):

for row in range(0,8):

for col in range(0,8):

queenPos = queen.columns.index(row)

if col == queenPos:

print'| Q |-',

else:

print'| |-',

print('\n')

def place(self,startRow = 0):

if len(self.columns) == self.size:

print('Solution found!\nSize of Chessboard :: '+str(self.size))

print('Total number of positions checked :: '+str(self.placesChecked))

print('Total number of backtracks :: '+str(self.numBacktracks))

print('\n:: :: Row Positions :: ::')

print(self.columns)

else:

for row in range(startRow,self.size):

if self.checkPosition(len(self.columns),row) is True:

self.columns.append(row)

self.placesChecked += 1

return self.place()

self.numBacktracks += 1

return self.place(self.columns.pop()+1)

def checkPosition(self,col,row):

for positionedRow in self.columns:

positionedCol = self.columns.index(positionedRow)

if row == positionedRow or col == positionedCol:

return False

elif row - col == positionedRow - positionedCol or row + col == positionedRow + positionedCol:

return False

return True

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

rank = MPI.COMM\_WORLD.Get\_rank()

n = 8

Buffer = []

queen = NQueenProblem(n)

doc = minidom.parse('queenPosition.xml')

#Sending initial configuration

if rank == 0:

start = MPI.Wtime()

#Slave1

position = doc.getElementsByTagName("slave1")[0]

pos = int(position.firstChild.data)

print('Slave\'1 Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=1,tag=100)

position = doc.getElementsByTagName("slave2")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=2,tag=200)

position = doc.getElementsByTagName("slave3")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=3,tag=300)

position = doc.getElementsByTagName("slave4")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=4,tag=400)

position = doc.getElementsByTagName("slave5")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=5,tag=500)

position = doc.getElementsByTagName("slave6")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=6,tag=600)

position = doc.getElementsByTagName("slave7")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=7,tag=700)

position = doc.getElementsByTagName("slave8")[0]

pos = int(position.firstChild.data)

print('Slave1\'s Position :: '+str(pos))

Buffer.append(pos)

MPI.COMM\_WORLD.send(Buffer,dest=8,tag=800)

MPI.COMM\_WORLD.Barrier()

#Parallel Configuration

if rank == 1:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=100)

print('\n\n:: :: Rank 1 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 2:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=200)

print('\n\n:: :: Rank 2 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 3:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=300)

print('\n\n:: :: Rank 3 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 4:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=400)

print('\n\n:: :: Rank 4 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 5:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=500)

print('\n\n:: :: Rank 5 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 6:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=600)

print('\n\n:: :: Rank 6 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 7:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=700)

print('\n\n:: :: Rank 7 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

if rank == 8:

Buffer = MPI.COMM\_WORLD.recv(source=0,tag=800)

print('\n\n:: :: Rank 8 (Calculation):: ::\n')

queen.place(Buffer[0])

queen.chessBoard()

MPI.COMM\_WORLD.Barrier()

#Gathering and printing output

if rank == 0:

end = MPI.Wtime()

timeMaster = end-start

print('Time :: '+str(end-start))

target = open("TimeAnalysis", 'a')

target.write(str(end-start))

target.write("\n")

target.close()

//queenPosition.xml

<chessboard>

<slave1>0</slave1>

<slave2>1</slave2>

<slave3>2</slave3>

<slave4>3</slave4>

<slave5>4</slave5>

<slave6>5</slave6>

<slave7>6</slave7>

<slave8>7</slave8>

</chessboard>

//script.py

import os

import matplotlib.pyplot as plt; plt.rcdefaults()

import numpy as np

import matplotlib.pyplot as plt

target = open("TimeAnalysis", 'w')

target.close()

os.system("mpiexec -np 2 python eightQueenPlot\_2.py")

os.system("mpiexec -np 3 python eightQueenPlot\_3.py")

os.system("mpiexec -np 4 python eightQueenPlot\_4.py")

os.system("mpiexec -np 5 python eightQueenPlot\_5.py")

os.system("mpiexec -np 6 python eightQueenPlot\_6.py")

os.system("mpiexec -np 7 python eightQueenPlot\_7.py")

os.system("mpiexec -np 8 python eightQueenPlot\_8.py")

os.system("mpiexec -np 9 python eightQueenPlot\_9.py")

# --> , '2nd\nConfig', '3rd\nConfig', '4th\nConfig', '5th\nConfig', '6th\nConfig', '7th\nConfig', '8th\nConfig

objects = ('1st\nConfig','2nd\nConfig', '3rd\nConfig', '4th\nConfig', '5th\nConfig', '6th\nConfig', '7th\nConfig', '8th\nConfig')

y\_pos = np.arange(len(objects))

target = open("TimeAnalysis", 'r')

performance = []

performance.append(target.readline())

performance.append(target.readline())

performance.append(target.readline())

performance.append(target.readline())

performance.append(target.readline())

performance.append(target.readline())

performance.append(target.readline())

performance.append(target.readline())

plt.bar(y\_pos, performance, align='center', alpha=0.3)

plt.xticks(y\_pos, objects)

plt.ylabel('Time for Execution')

plt.title('Loaded and Unloaded Cluster')

plt.show()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LOADING AND UNLOADING OF CLUSTER\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

pict@localhost~$ cd workspace/CL4/Final\_codes/Unloading/

pict@localhost~/workspace/CL4/Final\_codes/Unloading$ python script.py

Slave1 Position :: 0

Slave2 Position :: 1

Slave3 Position :: 2

Slave4 Position :: 3

Slave5 Position :: 4

Slave6 Position :: 5

Slave7 Position :: 6

Slave8 Position :: 7

:: :: Rank 1 (Calculation):: ::

Solution found!

Size of Chessboard :: 8

Total number of positions checked :: 113

Total number of backtracks :: 105

:: :: Row Positions :: ::

[0, 4, 7, 5, 2, 6, 1, 3]

| Q |- | |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | Q |- | |-

| |- | |- | |- | |- | Q |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | |- | Q |-

| |- | Q |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | Q |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | Q |- | |- | |-

| |- | |- | Q |- | |- | |- | |- | |- | |-

:: :: Rank 1 (Calculation):: ::

Solution found!

Size of Chessboard :: 8

Total number of positions checked :: 113

Total number of backtracks :: 105

:: :: Row Positions :: ::

[0, 4, 7, 5, 2, 6, 1, 3]

| Q |- | |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | Q |- | |-

| |- | |- | |- | |- | Q |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | |- | Q |-

| |- | Q |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | Q |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | Q |- | |- | |-

| |- | |- | Q |- | |- | |- | |- | |- | |-

:: :: Rank 1 (Calculation):: ::

Solution found!

Size of Chessboard :: 8

Total number of positions checked :: 113

Total number of backtracks :: 105

:: :: Row Positions :: ::

[0, 4, 7, 5, 2, 6, 1, 3]

| Q |- | |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | Q |- | |-

| |- | |- | |- | |- | Q |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | |- | Q |-

| |- | Q |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | Q |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | Q |- | |- | |-

| |- | |- | Q |- | |- | |- | |- | |- | |-

:: :: Rank 1 (Calculation):: ::

Solution found!

Size of Chessboard :: 8

Total number of positions checked :: 113

Total number of backtracks :: 105

:: :: Row Positions :: ::

[0, 4, 7, 5, 2, 6, 1, 3]

| Q |- | |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | Q |- | |-

| |- | |- | |- | |- | Q |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | |- | Q |-

| |- | Q |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | Q |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | Q |- | |- | |-

| |- | |- | Q |- | |- | |- | |- | |- | |-

:: :: Rank 1 (Calculation):: ::

Solution found!

Size of Chessboard :: 8

Total number of positions checked :: 113

Total number of backtracks :: 105

:: :: Row Positions :: ::

[0, 4, 7, 5, 2, 6, 1, 3]

| Q |- | |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | Q |- | |-

| |- | |- | |- | |- | Q |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | |- | Q |-

| |- | Q |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | Q |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | Q |- | |- | |-

| |- | |- | Q |- | |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | Q |- | |- | |-

| |- | |- | Q |- | |- | |- | |- | |- | |-

:: :: Rank 2 (Calculation):: ::

Solution found!

Size of Chessboard :: 8

Total number of positions checked :: 113

Total number of backtracks :: 105

:: :: Row Positions :: ::

[0, 4, 7, 5, 2, 6, 1, 3]

| Q |- | |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | Q |- | |-

| |- | |- | |- | |- | Q |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | |- | Q |-

| |- | Q |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | Q |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | Q |- | |- | |-

| |- | |- | Q |- | |- | |- | |- | |- | |-

Time :: 0.143836021423

Slave1 Position :: 0

Slave2 Position :: 1

Slave3 Position :: 2

Slave4 Position :: 3

Slave5 Position :: 4

Slave6 Position :: 5

Slave7 Position :: 6

Slave8 Position :: 7

:: :: Rank 1 (Calculation):: ::

Solution found!

Size of Chessboard :: 8

Total number of positions checked :: 113

Total number of backtracks :: 105

:: :: Row Positions :: ::

[0, 4, 7, 5, 2, 6, 1, 3]

| Q |- | |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | Q |- | |-

| |- | |- | |- | |- | Q |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | |- | Q |-

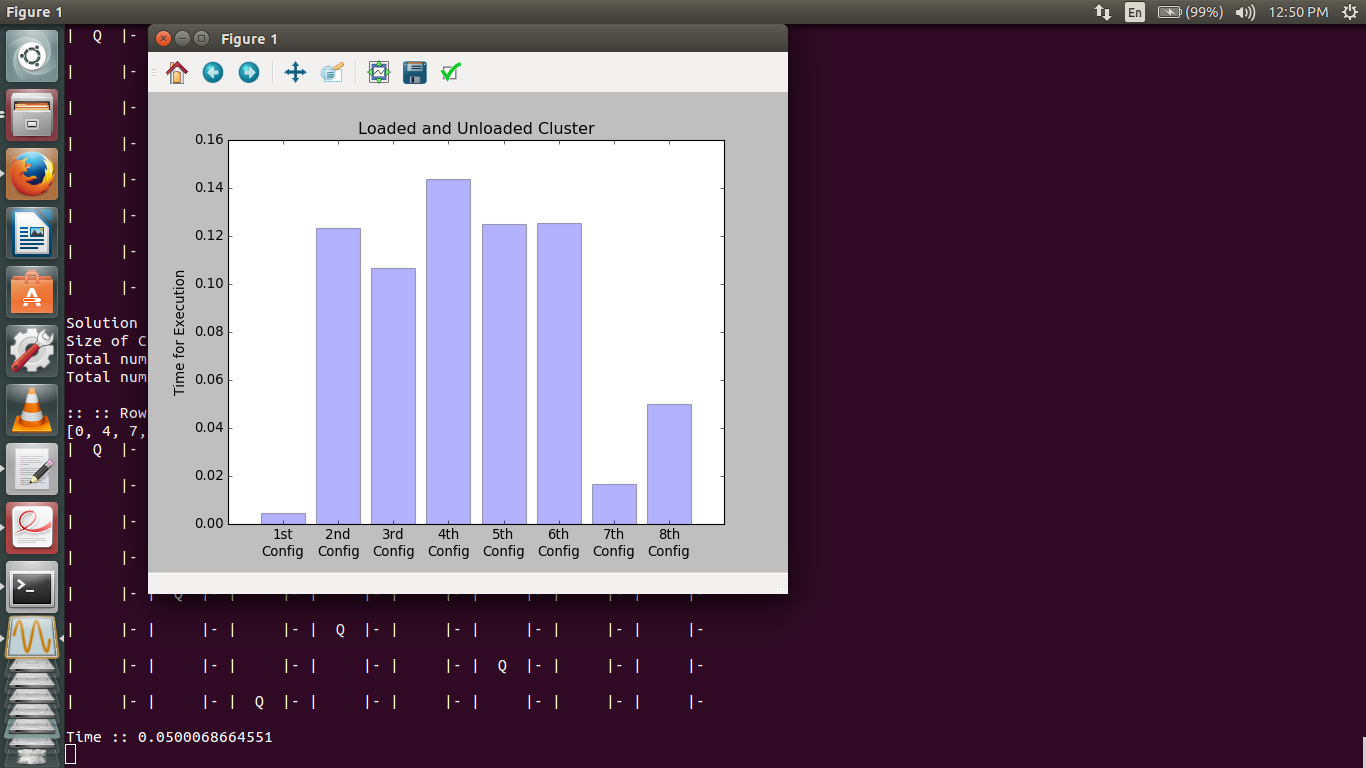
| |- | Q |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | Q |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | Q |- | |- | |-

| |- | |- | Q |- | |- | |- | |- | |- | |-

:: :: Rank 7 (Calculation):: ::

Solution found!

Size of Chessboard :: 8

Total number of positions checked :: 113

Total number of backtracks :: 105

:: :: Row Positions :: ::

[0, 4, 7, 5, 2, 6, 1, 3]

| Q |- | |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | Q |- | |-

| |- | |- | |- | |- | Q |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | |- | Q |-

| |- | Q |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | Q |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | Q |- | |- | |-

| |- | |- | Q |- | |- | |- | |- | |- | |-

Solution found!

Size of Chessboard :: 8

Total number of positions checked :: 113

Total number of backtracks :: 105

:: :: Row Positions :: ::

[0, 4, 7, 5, 2, 6, 1, 3]

| Q |- | |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | Q |- | |-

| |- | |- | |- | |- | Q |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | |- | Q |-

| |- | Q |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | Q |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | Q |- | |- | |-

| |- | |- | Q |- | |- | |- | |- | |- | |-

Solution found!

Size of Chessboard :: 8

Total number of positions checked :: 113

Total number of backtracks :: 105

:: :: Row Positions :: ::

[0, 4, 7, 5, 2, 6, 1, 3]

| Q |- | |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | Q |- | |-

| |- | |- | |- | |- | Q |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | |- | Q |-

| |- | Q |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | Q |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | Q |- | |- | |-

| |- | |- | Q |- | |- | |- | |- | |- | |-

:: :: Row Positions :: ::

[0, 4, 7, 5, 2, 6, 1, 3]

| Q |- | |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | Q |- | |-

| |- | |- | |- | |- | Q |- | |- | |- | |-

| |- | |- | |- | |- | |- | |- | |- | Q |-

| |- | Q |- | |- | |- | |- | |- | |- | |-

| |- | |- | |- | Q |- | |- | |- | |- | |-

| |- | |- | |- | |- | |- | Q |- | |- | |-

| |- | |- | Q |- | |- | |- | |- | |- | |-

Time :: 0.0500068664551