**Pham Duc Dat**

**ITITIU20184**

Exercise 1:

self.variables = squares  
self.domain = self.getDict(grid)  
self.values = self.getDict(grid)  
  
self.unitlist = ([cross(rows, c) for c in cols] +  
 [cross(r, cols) for r in rows] +  
 [cross(rs, cs) for rs in ('ABC', 'DEF', 'GHI') for cs in ('123', '456', '789')])  
  
self.units = dict((s, [u for u in self.unitlist if s in u]) for s in squares)  
self.peers = dict((s, set(sum(self.units[s],[]))-set([s])) for s in squares)  
self.constraints = {(variable, peer) for variable in self.variables for peer in self.peers[variable]}

Exercise 2:

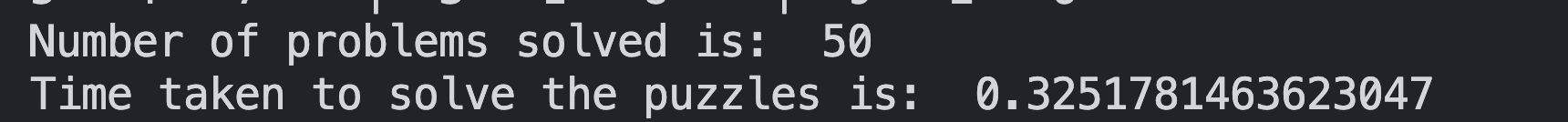
def Backtracking\_Search(csp):  
 return Recursive\_Backtracking({}, csp)  
  
def Recursive\_Backtracking(assignment, csp):  
 if isComplete(assignment):  
 return assignment  
  
 var = Select\_Unassigned\_Variables(assignment, csp)  
 domain = deepcopy(csp.values)  
  
 for value in csp.values[var]:  
 if isConsistent(var, value, assignment, csp):  
 assignment[var] = value  
 inferences = {}  
 inferences = Inference(assignment, inferences, csp, var, value)  
 if inferences!= "FAILURE":  
 result = Recursive\_Backtracking(assignment, csp)  
 if result!="FAILURE":  
 return result  
  
 del assignment[var]  
 csp.values.update(domain)  
 return "FAILURE"

Exercise 3:

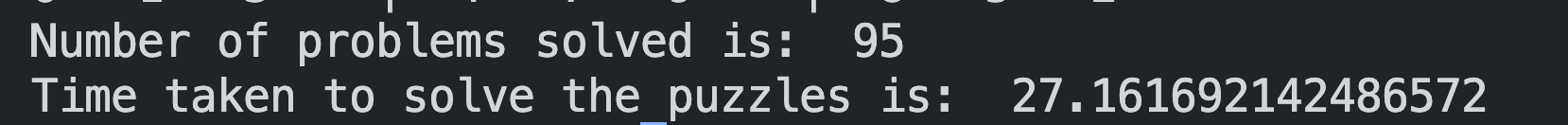
A screen shot of a computer

Description automatically generated with low confidence

What to submit:

1. 

The Euler file which is attached beside this word file. As we can see, those sudoku are quite normal and are not difficult. Therefore, the solving time is fast (around 0.3 second)

1. 

The magictour file is also attached beside this word file. It can be seen clearly that the second one is more complex and taking more time for solving (around 27 seconds)

1. This assignment took my work approximately 1 day, because I need to understand deeply about how it works.