Projeto e Análise de Algoritmos – 2021.1
Segundo trabalho – Questão 1
Matheus Araújo e Rhayane Monteiro

1)

Algoritmo em pseudocódigo

Classe SAT

- Classe que implementa o objeto SAT, lê o arquivo e possui os métodos de verificação *ehsatisfativel()* e *clausulasvdd()*.

```
defina classe Sat(object):

    funcao construtor(self, file):
        self.clausulas = []
        enquanto line in file:
            se line == file[0]:
                meta_data = line.split()
                self.num_variaveis = int(meta_data[0])
                self.num_clausulas = int(meta_data[1])
            senao se line == file[-1]:
                self.statefile = [l.rstrip() enquanto l in line]
                self.statefile.pop()
                self.statefile = [int(i) enquanto i in self.statefile]
            senao:
                clausula = []
                enquanto var in line.split()[:]:
                    clausula.append(int(var))
                se len(clausula) == 3:
                    self.clausulas.append(clausula)

    funcao ehsatisfativel(clausula, bits):
        enquanto x in clausula:
            se (x > 0 and bits[abs(x)-1]) or (x < 0 and not bits[abs(x)-1]):
                return True
        retorne False
```

Classes Solution e State

- Classe Solution implementa o objeto solução herdando SAT, faz chamada dos métodos SAT e implementa o método de avaliação ou *evaluate*. Enquanto State implementa o estado (bits).

```
inclua biblioteca random
inclua biblioteca math
inclua Sat

defina classe State():
    funcao construtor(self, bit_array):
        self.bit_array = bit_array

defina classe Solution(Sat):
    funcao construtor(self, file):
        super(Solution, self).construtor(file)

    funcao get_qtd(self, state):
        new_bit_array = list(state.bit_array)
        retorne self.clausulasvdd(new_bit_array)

    funcao makestate(self, state):
        retorne State(state)
```

Classes File e Main

- A classe File implementa a leitura do arquivo enquanto a main é a classe principal de chamada do programa, invocando o objeto Solution no método *solve()* passando os dados desejados e recebendo os resultados.

```
inclua biblioteca Solution de solution
inclua biblioteca time

defina classe File(list):

    funcao construtor(self, path):
        self = [self.append(x) enquanto x in File.load(path)]

    funcao load(path):
        with open(path) as f:
            retorne f.readlines()

funcao solve(files, ehvetornulo):
    enquanto file in files:
        sa = Solution(File(file))
        inicio = time.time()
        resultqtd, resultehsat = sa.evaluate(ehvetornulo)
        duracao = time.time() - inicio
    return resultqtd, resultehsat, duracao
```

Programa implementado

Classe SAT

- Classe que implementa o objeto SAT, lê o arquivo e possui os métodos de verificação.

```
class Sat(object):

  def __init__(self, file):
    self.clausulas = []
    for line in file:
      if line == file[0]:
        meta_data = line.split()
        self.num_variaveis = int(meta_data[0])
        self.num_clausulas = int(meta_data[1])
      elif line == file[-1]:
        self.statefile = [l.rstrip() for l in line]
        self.statefile.pop()
        self.statefile = [int(i) for i in self.statefile]
      else:
        clausula = []
        for var in line.split()[:]:
          clausula.append(int(var))
        if len(clausula) == 3:
          self.clausulas.append(clausula)

  def clausulasvdd(self, bits):
    num_satisfativeis = 0
    ehsat = True
    for clausula in self.clausulas:
      if Sat.ehsatisfativel(clausula, bits):
        num_satisfativeis += 1
      else:
        ehsat = False

    return num_satisfativeis, ehsat

  @staticmethod
  def ehsatisfativel(clausula, bits):
    for x in clausula:
      if (x > 0 and bits[abs(x)-1]) or (x < 0 and not bits[abs(x)-
1]):
        return True
    return False
```

Classes Solution e State

- Classe Solution implementa o objeto solução herdando SAT, faz chamada dos métodos SAT e implementa o método de avaliação ou *evaluate*. Enquanto State implementa o estado (bits).

```
import random
import math
from sat import Sat

class State():
  def __init__(self, bit_array):
    self.bit_array = bit_array

class Solution(Sat):
  def __init__(self, file):
    super(Solution, self).__init__(file)

  def get_qtd(self, state):
    new_bit_array = list(state.bit_array)
    return self.clausulasvdd(new_bit_array)

  def makestate(self, state):
    return State(state)

  def evaluate(self, ehvetornulo):
    if ehvetornulo:
      statenulo = [0 for i in range(self.num_variaveis)]
      state = self.makestate(statenulo)
      return self.get_qtd(state)
    else:
      state = self.makestate(self.statefile)
      return self.get_qtd(state)
```

Classes File e Main

- A classe File implementa a leitura do arquivo enquanto a main é a classe principal de chamada do programa, invocando o objeto Solution no método *solve()* passando os dados desejados e recebendo os resultados.

```
from solution import Solution
import time

class File(list):
```

```
  def __init__(self, path):
    self = [self.append(x) for x in File.load(path)]

  @staticmethod
  def load(path):
    with open(path) as f:
      return f.readlines()

def solve(files, ehvetornulo):
  for file in files:
    sa = Solution(File(file))
    inicio = time.time()
    resultqtd, resultehsat = sa.evaluate(ehvetornulo)
    duracao = time.time() - inicio
  return resultqtd, resultehsat, duracao

def main():

  arquivocnf = ["SAT1.txt"]
  #arquivocnf = ["SAT2.txt"]
  #arquivocnf = ["SAT3.txt"]

  ehvetornulo = False
  #ehvetornulo = True

  resultqtd, resultehsat, duracao = solve(arquivocnf, ehvetornulo)

  if resultehsat:
    print("F(X) = %d SAT" % resultqtd)
    print("Duracao: %f" % duracao)
  else:
    print("F(X) = %d unSAT" % resultqtd)
    print("Duracao: %f" % duracao)

if __name__ == '__main__':
  main()
```

Resultados

SAT1.txt

F(vetor nulo) = 239 unSAT (Duração: 0.000000)
F(vetor dado) = 275 SAT (Duração: 0.000000)

SAT2.txt

F(vetor nulo) = 436 unSAT (Duração: 0.000000)
F(vetor dado) = 499 unSAT (Duração: 0.000000)

SAT3.txt

F(vetor nulo) = 1320 unSAT (Duração: 0.000000)
F(vetor dado) = 1497 unSAT (Duração: 0.001000)