**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**

**Інститут прикладної математики і фундаментальних наук**

**Кафедра прикладної математики**

**ЗВІТ**

про виконання лаборатнорної роботи №1

із дисципліни **“Математичні основи штучного інтелекту”**

Виконав: студент групи ПМ-32, Шеремета Данило  
Прийняв: доц. каф. Пабирівський В.В.

**Львів — 2023**

Лабораторна робота №1

***Тема.***Алгоритм відпалу.

**Постановка задачі**

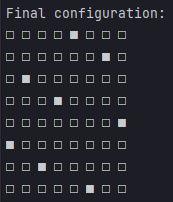
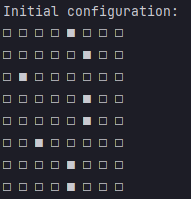
Розв’язати із використанням алгоритму відпалу задачу розстановки N шахових ферзів на шаховій дошці розміру N×N таким чином, аби жоден ферзь не загрожував будь-якому іншому.

**Код програми**

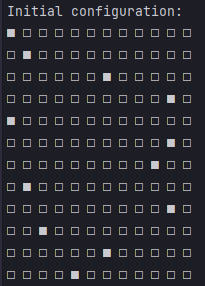
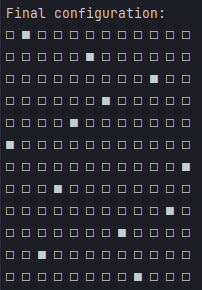
import random  
import math  
  
  
class Chessboard:  
 def \_\_init\_\_(self**,** n**,** board=None):  
 self.n = n  
 if board:  
 self.board = board  
 else:  
 self.board = self.random\_board()  
  
 def random\_board(self):  
 return [random.randint(0**,** self.n-1) for \_ in range(self.n)]  
  
 def print\_board(self):  
 for row in self.board:  
 line = ""  
 for col in range(self.n):  
 if col == row:  
 line += "■ "  
 else:  
 line += "□ "  
 print(line)  
 print("\n")  
  
 def attacks(self):  
 count = 0  
 for i in range(self.n):  
 for j in range(i + 1**,** self.n):  
 if self.board[i] == self.board[j] or abs(self.board[i] - self.board[j]) == abs(i - j):  
 count += 1  
 return count  
  
 def energy(self):  
 return self.attacks()  
  
 def neighbor(self):  
 new\_board = list(self.board)  
 i = random.randint(0**,** self.n-1)  
 j = random.randint(0**,** self.n-1)  
 while new\_board[i] == j:  
 j = random.randint(0**,** self.n-1)  
 new\_board[i] = j  
 return new\_board  
  
  
class SimulatedAnnealing:  
 def \_\_init\_\_(self**,** temperature=10.0**,** cooling\_rate=0.9999):  
 self.temperature = temperature  
 self.cooling\_rate = cooling\_rate  
  
 def acceptance\_probability(self**,** old\_energy**,** new\_energy):  
 if new\_energy < old\_energy:  
 return 1.0  
 else:  
 return math.exp((old\_energy - new\_energy) / self.temperature)  
  
 def anneal(self**,** chessboard):  
 old\_energy = chessboard.energy()  
 while self.temperature > 0.001 and old\_energy > 0:  
 new\_board = chessboard.neighbor()  
 new\_energy = Chessboard(chessboard.n**,** board=new\_board).energy()  
 if self.acceptance\_probability(old\_energy**,** new\_energy) > random.random():  
 chessboard.board = new\_board[:]  
 old\_energy = new\_energy  
 self.temperature \*= self.cooling\_rate  
 return chessboard.board  
  
  
n = 20 # size of chessboard (n x n)  
  
chessboard = Chessboard(n) # generate initial configuration  
  
print("Initial configuration:")  
chessboard.print\_board()  
  
annealer = SimulatedAnnealing()  
board = annealer.anneal(chessboard) # find solution using simulated annealing  
  
print("Final configuration:")  
Chessboard(n**,** board).print\_board()

**Результат виконання програми**

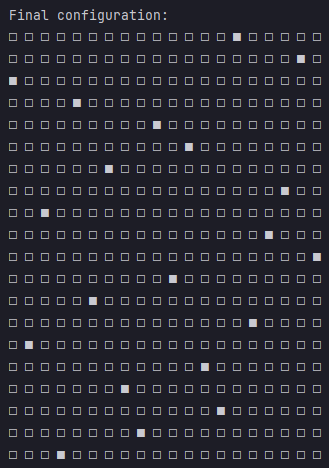
***Для n=8:***



***Для n=12:***

***Для n=20:***



***Для n=30:***

**Висновок**

У результаті виконання даної лабораторної роботи я навчився використовувати алгоритм відпалу. Створив програму яка розв’язує задачу розстановки *N* шахових ферзів на шаховій дошці розміру *N×N* таким чином, аби жоден ферзь не загрожував будь-якому іншому за допомогою алгоритму відпалу.

***Лінк на GitHub:***https://github.com/DanSheremeta/mbai\_labs/tree/main/Lab1