

# Tugas 4

Kelompok 5

# Kelompok 5

FIhriz Ilham  
Rabbany

5025211040

Syomeron  
Ansell Widjaya

5025211250

Muhammad  
Ahyun Irsyad

5025211251

# 8-QUEENS PROBLEM DENGAN MENGGUNAKAN GA

# GENETIC ALGORITHM

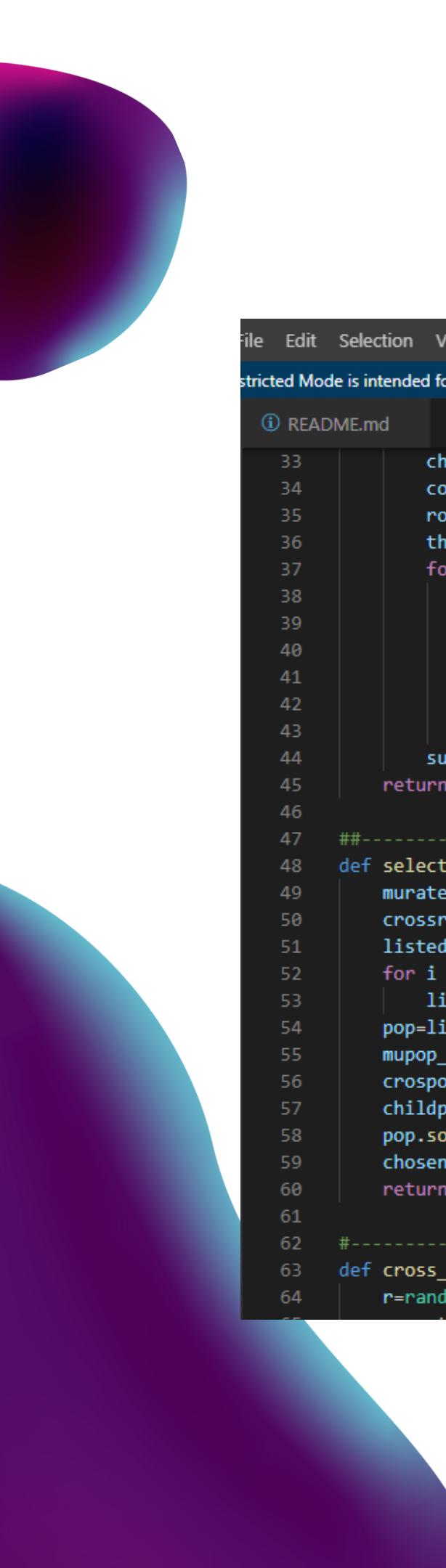
Genetic Algorithm (GA) adalah bagian dari Evolutionary Algorithm yaitu suatu algoritma yang mencontoh proses evolusi alami dimana konsep utamanya adalah individu-individu yang paling unggul akan bertahan hidup, sedangkan individu-individu yang lemah akan punah. Keunggulan individu-individu ini diuji melalui suatu fungsi yang dikenal sebagai fitness function. Fitness dalam GA didefinisikan sebagai gambaran kelayakan suatu solusi terhadap suatu permasalahan. Fitness Function akan menghasilkan suatu nilai fitness value yang akan menjadi referensi untuk proses GA selanjutnya.



```
File Edit Selection View Go Run Terminal Help • import numpy as np • Untitled-2 - Visual Studio Code  
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More  
① README.md import numpy as np Untitled-2 1 ● from random import randint Untitled-1 ●  
2 1 import numpy as np  
2 import random as random  
3 import math as math  
4  
5 popsize=10  
6 generation=100  
7  
8 #init pop-----  
9 def init_pop():  
10     pop=np.zeros((popsize,8))  
11     for i in range(popsize):  
12         pop[i]=np.random.permutation([1,2,3,4,5,6,7,8])  
13     return pop  
14 #-----  
15  
16 #-----  
17 #-----  
18 def printchess(chrom):  
19     print('-----')  
20     for i in range(8):  
21         print('|',end='')  
22         for j in range(8):  
23             if 1+j==chrom[i]:  
24                 print(' Q |',end='')  
25             else:  
26                 print("   |",end='')  
27         print()  
28         print('-----')  
29 #-----  
30 def fitness(chrom):  
31     sum=0  
32     for index in range(len(chrom)):
```

# Source Code

# Source Code



```
File Edit Selection View Go Run Terminal Help • import numpy as np • Untitled-2 - Visual Studio Code
strict Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
① README.md ② import numpy as np Untitled-2 1 ● ③ from random import randint Untitled-1 ●
```

```
33 chrom=list(chrom)
34 col=index
35 row=chrom[index]
36 threat=0
37 for i in range(len(chrom)):
38     if i==col:
39         continue
40     if chrom[i]<row and chrom[i] + math.fabs(col-i)==row:
41         threat=threat+1
42     elif chrom[i]>row and chrom[i]-math.fabs(col-i)==row:
43         threat=threat+1
44     sum=sum+threat
45 return sum
46
47 #-----
48 def selection(pop):
49     murate=0.1
50     crossrate=.3
51     listedpop=[]
52     for i in range(len(pop)):
53         listedpop.append(list(pop[i]))
54     pop=listedpop
55     mupop_size=int(np.ceil(murate*popsiz))
56     crospop_size=(2*np.ceil(crossrate*popsiz))
57     childpop_size=mupop_size+crospop_size
58     pop.sort(key=finess)
59     chosen=pop[0:int(childpop_size)]
60     return chosen
61
62 #-----
63 def cross_over(p1, p2):#PMX
64     r=random.sample(range(1,8),2)
```

```
File Edit Selection View Go Run Terminal Help • import numpy as np • Untitled-2 - Visual Studio Code
strict Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
① README.md ② import numpy as np Untitled-2 1 ● ③ from random import randint Untitled-1 ●
```

```
65 r.sort()
66 r1,r2=r[0],r[1]
67 p2=list(p2)
68 p1=list(p1)
69
70 middle1=p1[r1:r2]
71 middle2=p2[r1:r2]
72
73 child1=np.concatenate((r1*[0], middle1 ,(len(p1)-r2)*[0]),axis=0)
74 child2=np.concatenate((r1*[0],middle2 ,(len(p2)-r2)*[0]),axis=0)
75
76 for i in range(len(middle1)):
77     if middle2[i] not in middle1:
78         temp1=middle1[i]
79         temp2=middle2[i]
80         index=p2.index(temp1)
81         while(p1[index] in middle1):
82             temp1=p1[index]
83             temp2=p2[index]
84             index=p2.index(temp1)
85             child1[index]=middle2[i]
86         for i in range(len(child1)):
87             if child1[i]==0:
88                 child1[i]=p2[i]
89         for i in range(len(middle2)):
90             if middle1[i] not in middle2:
91                 temp2=middle2[i]
92                 temp1=middle1[i]
93                 index=p1.index(temp2)
94                 while(p2[index] in middle2):
95                     temp2=p2[index]
96                     temp1=p1[index]
```

# Source Code

```
File Edit Selection View Go Run Terminal Help • import numpy as np • Untitled-2 - Visual Studio Code
stricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
① README.md ② import numpy as np Untitled-2 1 ● ③ from random import randint Untitled-1 ●
96     temp2=p2[index]
97     temp1=p1[index]
98     index=p1.index(temp2)
99     child2[index]=middle1[i]
100    for i in range(len(child2)):
101        if child2[i]==0:
102            child2[i]=p1[i]
103    return [child1,child2]
104 #####
105 def mutation(childs): #swap
106     muindex=(random.sample(range(0,len(childs)),1))#choose child
107     chrom=childs[muindex[0]]
108     position1 = random.randint(0, len(chrom)-1)
109     position2 = random.randint(0, len(chrom)-1)
110     #####chrom = list(chrom)
111     temp = chrom[position1]
112     chrom[position1] = chrom[position2]
113     chrom[position2] = temp
114     #####chrom = list(chrom)
115     childs[muindex[0]]=chrom
116     return childs
117 #####
118 def survival_selection(population, childs):
119     population=list(population)
120     population.sort(key=finess)
121     childs.sort(key=finess)
122     population[-1] = childs[0]
123     population[-2] = childs[1]
124     population[-3] = childs[2]
125     population[-4] = childs[3]
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
temp2=p2[index]
temp1=p1[index]
index=p1.index(temp2)
child2[index]=middle1[i]
for i in range(len(child2)):
    if child2[i]==0:
        child2[i]=p1[i]
return [child1,child2]
#####
def mutation(childs): #swap
    muindex=(random.sample(range(0,len(childs)),1))#choose child
    chrom=childs[muindex[0]]
    position1 = random.randint(0, len(chrom)-1)
    position2 = random.randint(0, len(chrom)-1)
    #####chrom = list(chrom)
    temp = chrom[position1]
    chrom[position1] = chrom[position2]
    chrom[position2] = temp
    #####chrom = list(chrom)
    childs[muindex[0]]=chrom
    return childs
#####
def survival_selection(population, childs):
    population=list(population)
    population.sort(key=finess)
    childs.sort(key=finess)
    population[-1] = childs[0]
    population[-2] = childs[1]
    population[-3] = childs[2]
    population[-4] = childs[3]
population[-1] = child1
population[-2] = child2
population[-3] = child3
population[-4] = child4
fit=[]
for i in range(len(population)):
    fit.append(finess(pop[i]))
return population,fit
#genetic main loop-
pop=init_pop()
mainpopsize=popsiz
crossrate=.5
crospop_size=(2*np.ceil(crossrate*popsiz))/2
for iteration in range(generation):
    child1,child2=cross_over(selected[i],selected[i+1])
    child1.append(list(child1))
    child1.append(list(child2))
    child1 = mutation(child1)
    newpop , bestfitness = survival_selection(pop, child1)
    pop=newpop
    print('best fitness:',bestfitness[0])
    if bestfitness[0]==0:
        printchess(pop[0])
        break
print('best soulution',newpop[0])
```

# Penjelasan

Pada awalnya, populasi awal dibuat dengan memilih secara acak N angka dari 1 sampai N dan menempatkannya pada setiap baris di papan catur. Kemudian, untuk setiap individu pada populasi, fitness-nya dihitung berdasarkan jumlah ancaman yang dialami oleh setiap ratu. Jika ada dua ratu yang saling menyerang, maka nilai fitness-nya akan lebih besar.

Selanjutnya, pada setiap iterasi, individu pada populasi dipilih berdasarkan nilai fitness-nya, kemudian dilakukan operasi crossover dan mutasi untuk menghasilkan anak baru. Setelah itu, dilakukan seleksi untuk menentukan individu mana yang akan ditambahkan ke populasi berikutnya. Proses iterasi tersebut akan terus dilakukan hingga ditemukan solusi yang memenuhi kriteria atau telah mencapai jumlah iterasi yang telah ditentukan sebelumnya.

Dalam kode ini, operasi crossover yang digunakan adalah Partially Matched Crossover (PMX), sedangkan operasi mutasi yang digunakan adalah Swap. Seleksi orang tua dilakukan dengan metode elitism, yaitu memilih individu dengan nilai fitness terbaik dari populasi saat ini.

# OUTPUT

The image shows two side-by-side terminal windows from an "Interactive Python Course". Both windows have a dark theme with light-colored text.

**Left Terminal Window:**

- Header: "Interactive Python Course" and "Shell".
- Text output:
  - generation 0
  - best fitness: 10
  - generation 1
  - best fitness: 4
  - generation 2
  - best fitness: 2
  - generation 3
  - best fitness: 2
  - generation 4
  - best fitness: 2
  - generation 5
  - best fitness: 2
  - generation 6
  - best fitness: 2
  - generation 7
  - best fitness: 2
  - generation 8
  - best fitness: 0
- Below the generations, there are two rows of binary-like strings separated by dashed lines:
  - | | | | | Q | | | |
  - | | | | | | | Q | |

**Right Terminal Window:**

- Header: "Interactive Python Course" and "Shell".
- Text output:
  - generation 7
  - best fitness: 2
  - generation 8
  - best fitness: 0
- Below the generations, there are several rows of binary-like strings separated by dashed lines:
  - | | | | | | Q | | | |
  - | | | | | | | | Q | |
  - | | Q | | | | | | | |
  - | | | | | | | | Q | |
  - | | | | Q | | | | | |
  - | | | | | | | | | | Q |
  - | | | | | | | | | | |
- Text at the bottom: "best soulution [5.0, 7.0, 2.0, 6.0, 3.0, 1.0, 8.0, 4.0]".

TERIMAKASIH