ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

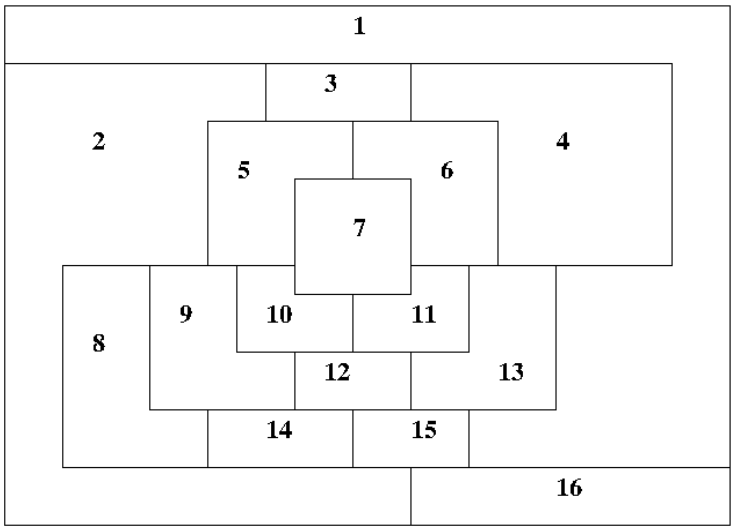
Εργασία Μαθήματος ***ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ ΚΑΙ ΕΜΠΕΙΡΑ ΣΥΣΤΗΜΑΤΑ***

Αριθμός εργασίας – Τίτλος εργασίας	***Bonus***
Όνομα φοιτητή	Καλλίγερος Αναστάσιος
Αρ. Μητρώου	Π19253
Ημερομηνία παράδοσης	21/04/2023

Εκφώνηση εργασίας

<u>Η εργασία είναι ατομική. Παραδοτέο είναι αρχείο pdf με τεκμηριωμένο κώδικα και παραδείγματα εκτέλεσης, περίπου 5 σελίδων. Υποβολή αποκλειστικά μέσω gunet. ΜΗ ΣΤΕΙΛΕΤΕ EMAIL. ΜΗ ΣΤΕΙΛΕΤΕ ΠΗΓΑΙΟ ΚΩΔΙΚΑ.</u>

Αναπτύξτε πρόγραμμα χρωματισμού του παρακάτω γράφου με χρήση γενετικών αλγορίθμων και γλώσσα προγραμματισμού της επιλογής σας. Τα διαθέσιμα χρώματα είναι 4: μπλε, κόκκινο, πράσινο, κίτρινο.



Χρησιμοποιείστε τυχαίο αρχικό πληθυσμό με πλήθος της δικής σας επιλογής. Χρησιμοποιείστε συνάρτηση καταλληλότητας και διαδικασία επιλογής γονέων σας της δικής σας επιλογής, επίσης. Χρησιμοποιείστε αναπαραγωγή με διασταύρωση ενός σημείου. Επιλέξτε αν θέλετε να κάνετε και μερική ανανέωση πληθυσμού σε κάποιο ποσοστό π.χ. 30% και μετάλλαξη ενός ψηφίου π.χ. στο 10% του πληθυσμού.

Παραδοτέα της εργασίας είναι ένα pdf (όχι zip, όχι πηγαίος κώδικας) που να περιλαμβάνει τον κώδικα και να τον εξηγεί, να εξηγεί τον τρόπο δράσης του υπολογιστή σύμφωνα με τον αλγόριθμο επίλυσης και να περιλαμβάνει παραδείγματα εκτέλεσης του προγράμματος που αναπτύξατε.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1 Επεξήγηση Κώδικα

Ο κώδικας υλοποιεί έναν γενετικό αλγόριθμο με διασταύρωση ενός σημείου. Έγινε χρήση ένος πίνακα γειτνίασης(adjacency matrix) για να καταγραφούν τα στοιχεία που συνορεύουν μεταξύ τους. Η συνάρτηση καταλληλόλητας που χρησιμοποιήθηκε υπολογίζει το ποσοστό των στοιχείων που συνορεύουν και έχουν διαφορετικό χρώμα σε σχέση με όλα τα στοιχεία που συνορεύουν. Αξιοσημείωτο αναφοράς είναι ότι όσο μεγαλύτερο το ποσοστό τόσο μεγαλύτερη η ομοιότητα.Η διασταύρωση γίνεται με την τεχνική της απλής ρουλέτας, η οποία αναθέτει τα ποσοστά σε περιοχές της μορφής (α,β] με α<β και με την παραγωγή ενός αριθμού ανάμεσα στο διάστημα [0,1] παίρνουμε το μέλος του πληθυσμού που θα διασταυρωθεί. Για την διασταύρωση όσο μεγαλύτερο το ποσοστό που πήρε το συγκεκριμένο μέλος, τόσο υψηλότερη η πιθανότητα να επιλεχθεί για την διαδικασία παραγωγής. Ο αλγόριθμος τερματίζει μετά τον προκαθορισμένο αριθμό εποχών(epochs) ή εάν δημιουργηθεί κάποιο μέλος με ομοιότητα μεγαλύτερη του προκαθορισμένου κατωφλίου καταλληλόλητας (threshold). Στο παράδειγμα παρουσιάζεται η λειτουργία του αλγορίθμου που κλήθηκε με τον ακόλουθο τρόπο.

2 Επίδειξη της λύσης

```
import random

class Color_Areas_GA:
        def __init__(self,population_size=10,epochs=100,threshold=0.9):
self.adj=[[2,3,4,13,15,16],[3,5,8,9,14,15],[1,2,4,5,6],[1,3,6,13],[2,3,6,7,9,10],
[3,4,5,7,11,13],[5,6,10,11],[2,9,14],[2,5,8,10,12,14],[5,7,9,11,12],
[6,7,10,12,13],[9,10,11,13,14,15],[1,4,11,12,15],[2,8,9,12,15],[1,12,13,14,16],
[1,2,15]]
self.colors=['B','R','G','Y']
self.p=population_size
```

```
self.epochs=epochs
self.threshold=threshold
self.total_points=0
for i in self.adj:
for j in i:
self.total_points=self.total_points+1
def generate_initial_population(self):
population=[]
for i in range(0,self.p):
parent=[]
for j in range(0,len(self.adj)):
parent.append(self.colors[random.randint(0,3)])
population.append(parent)
self.population=population
def fitness_function(self,population_member):
color
t=self.total_points
for i in range(0,len(population_member)):
color=population_member[i]
for j in range(0,len(self.adj[i])):
if(population_member[self.adj[i][j]-1]==color):
t=t-1
return t
def roulette_creator(self):
R=[]
sum=0
for i in range(0,len(self.population)):
sum=sum+self.fitness_function(self.population[i])
R.append(self.fitness_function(self.population[i]))
self.fitness=[]
for i in R:
self.fitness.append(round(i/self.total_points,2))
for i in range(0,len(R)):
R[i]=round(R[i]/sum,2)
self.roulette_points=R
b=0
boundaries=[]
for i in R:
b=round(b+i,3)
boundaries.append(b)
self.boundaries=boundaries
def crossover_population(self,K=10):
Pn=[]
for i in range(0,K):
p1=self.get_parent()
p2=self.get_parent()
```

```
point=random.randint(0,15)
pn=p1[:point]+p2[point:]
Pn.append(pn)
self.Pn=Pn
def get_parent(self):
percentage=round(random.uniform(0,1),2)
for i in range(0,len(self.boundaries)):
if(percentage<=self.boundaries[i]):
return self.population[i]
def next_population_round(self,to_be_replaced=4):
nfitness=[]
for i in self.Pn:
nfitness.append(round(self.fitness_function(i)/self.total_points,2))
for i in range(0,to_be_replaced):
min_value=min(self.fitness)
pos_min=self.fitness.index(min_value)
max_value=max(nfitness)
pos_max=nfitness.index(max_value)
self.fitness[pos_min]=nfitness[pos_max]
self.population[pos_min]=self.Pn[pos_max]
def find_solution(self):
self.generate_initial_population()
for i in range(0,self.epochs):
self.roulette_creator()
self.crossover_population()
self.next_population_round()
for i in range(0,len(self.fitness)):
if(self.fitness[i]>self.threshold):
self.print_solution(i)
return
print("After Final Epoch")
best_solution=max(self.fitness)
position=self.fitness.index(best_solution)
self.print_solution(position)
def print_solution(self,i):
p=self.population[i]
print("Solution Found,With a fitness percentage: "+str(self.fitness[i]))
for i in range(0,len(p)):
if(p[i]=='B'):
print("Area: "+str(i+1)+" Color: Blue")
elif(p[i]=='R'):
print("Area: "+str(i+1)+" Color: Red")
elif(p[i]=='G'):
print("Area: "+str(i+1)+" Color: Green")
elif(p[i]=='Y'):
print("Area: "+str(i+1)+" Color: Yello
```

3 Παράδειγμα

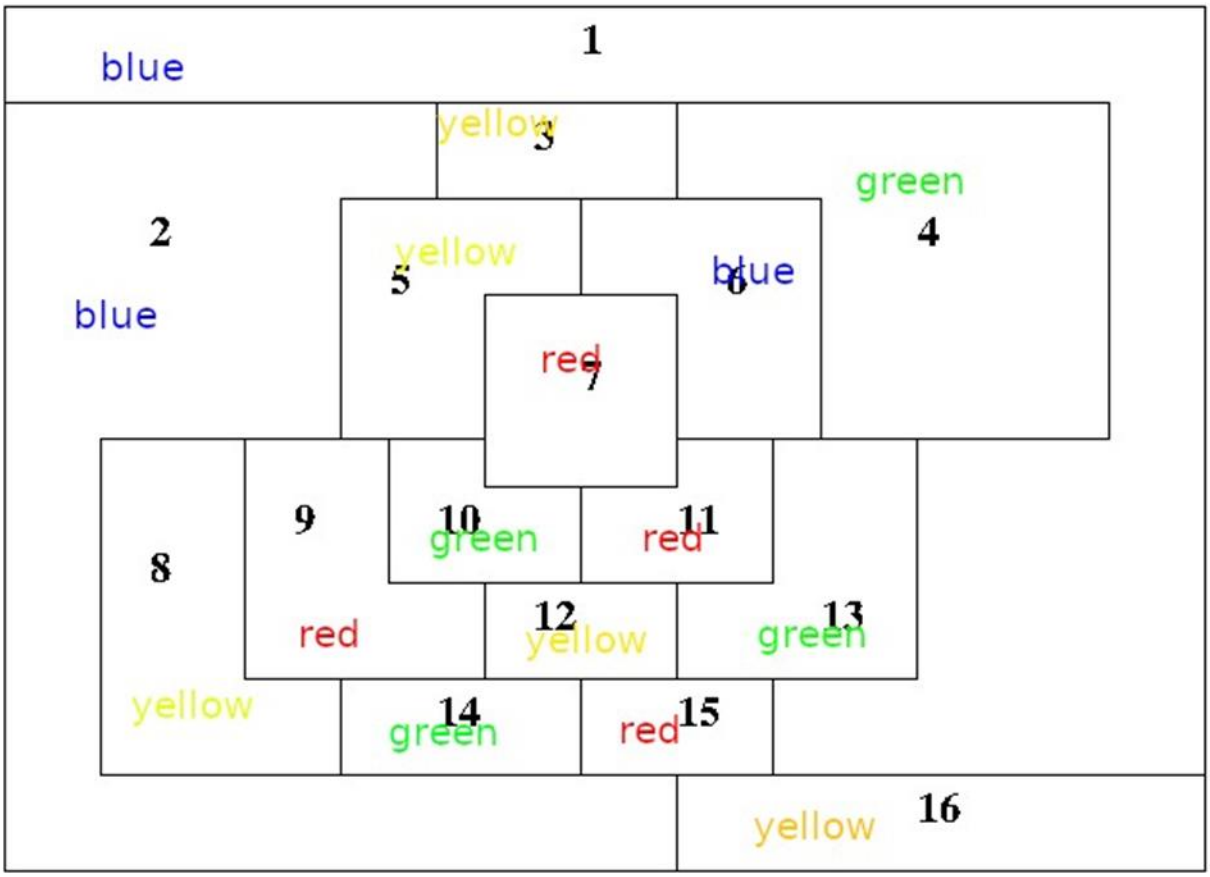
```
from genetic_coloring import Color_Areas_GA

ga=Color_Areas_GA(population_size=10,epochs=10,threshold=0.9)
ga.find_solution()
```

(10 δείγματα πλυθισμού,10 εποχές κατώφλι καταλληλότητας ίσο με 90% ομοιότητα με την βέλτιστη λύση)

```
Solution Found,With a fitness percentage: 0.91
Area: 1 Color: Blue
Area: 2 Color: Blue
Area: 3 Color: Yellow
Area: 4 Color: Green
Area: 5 Color: Yellow
Area: 6 Color: Blue
Area: 7 Color: Red
Area: 8 Color: Yellow
Area: 9 Color: Red
Area: 10 Color: Green
Area: 11 Color: Red
Area: 12 Color: Yellow
Area: 13 Color: Green
Area: 14 Color: Green
Area: 15 Color: Red
Area: 16 Color: Yellow
```

Η παραπάνω λύση δίνει αποτέλεσμα όμοιο κατά 91% με το βέλτιστο και οπτικά έχει την παρακάτω μορφή.


1
blue
yellow
3
green
2
4
yellow
5
6
blue
blue
red
7
9
10
11
8
green
red
12
13
red
yellow
green
yellow
14
15
green
red
16
yellow