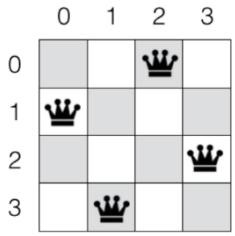
# GENETSKI ALGORITMI: N-QUEEN PROBLEM

ZADATAK: Korištenjem programskog okvira jGAP potrebno je napisati program koji će u najkraćem vremenu i uz najmanji broj evolucija pronaći rješenje raspoređivanja N kraljica na „šahovsku" ploču dimenzija NxN tako da se međusobno ne napadaju. Primjer jednog od rješenja za ploču 4x4:




Zadatak je potrebno riješiti korištenjem sljedećih koraka:

1. S mrežnih stranica „http://jgap.sourceforge.net/" potrebno je preuzeti posljednju verziju programskog okvira jGAP.
2. Unutar razvojnog okruženja Eclipse (ili nekog drugog) potrebno je kreirati Java projekt te i njegov „Build path" dodati datoteku „jgap.jar" koja se nalazi unutar arhive preuzete u prvom koraku.
3. Napisati klasu koja će nasljeđivati klasu „FitnessFunction", te implementirati metodu „evaluate" koja prima objekt klase koja implementira sučelje „IChromosome". Unutar metode „evaluate" potrebno je implementirati algoritam izračunavanja „dobrote" svakog od rješenja.
4. Napisati klasu koja će sadržavati „main" metodu i u njoj je potrebno implementirati sljedeće:
    a. Zadati (ili tražiti od korisnika) željene dimenzije ploče.
    b. Kreirati objekt klase iz trećeg koraka.
    c. Kreirati objekt klase „DefaultConfiguration" koji služi za konfiguriranje genetskog algoritma i njegovih detalja.
    d. U konfiguraciju iz koraka 4.c potrebno je dodati operatore za rekombinaciju (engl. *crossover operators*) i mutaciju (engl. *mutation operators*) pomoću metode „addGeneticOperator".
    e. Definirati parametre konfiguracije iz koraka 4.c kao što su održavanje veličine populacije konstantnom (metodom „setKeepPopulationSizeConstant"), zadržavanje najbolje jedinke u generaciji („setPreservFittestIndividual") itd.
    f. Definirati objekt funkcije dobrote kreiran u trećem koraku u objekt konfiguracije genetskog algoritma kreiranog u koraku 4.c.
    g. Kreirati gene odgovarajućeg tipa u obliku jednodimenzionalnog polja koji će sadržavati vrijednosti u odgovarajućem rasponu.

h. Kreirati objekt klase „Chromosome" koji u konstruktor prima polje gena iz koraka 4.g i objekt konfiguracije kreiranog u koraku 4.c.
i. Pomoću metoda „setSampleChromosome" i „setPopulationSize" u konfiguraciju kreiranu u koraku 4.c postaviti primjer kromosoma kreiranog u koraku 4.h i veličinu populacije koja će se koristiti. Voditi računa o tome da veličina populacije bude smislena s obzirom na ukupan broj mogućih kombinacija.
j. Pomoću naredbe npr.
„`Genotype population = Genotype.randomInitialGenotype(conf);`"
kreirati objekt populacije, pri čemu objekt „conf" predstavlja konfiguraciju kreiranu u koraku 4.c.
k. Unutar petlje koja se izvršava onoliko puta koliko je potrebno pokrenuti novu evoluciju, u svakom koraku je potrebno ispisati podatke o najboljoj jedinki koja se može dohvatiti pomoću naredbe „`population.getFittestChromosome();`", te izvršiti naredbu „`population.evolve();`" kojom se pokreće novi ciklus evolucije. Pretraživanje završava pronalaskom ispravnog rješenja ili istekom zadanog broja evolucija (što znači da rješenje nije pronađeno).

5. Implementirati programski isječak koji će tražiti rješenje na neki od klasičnih (konvencionalnih) metoda pretraživanja, npr. tako da pretražuje sve moguće kombinacije pozicija kraljica. Zasebno izmjeriti vrijeme takvog načina traženja rješenja i usporediti ga s vremenom genetskog algoritma.

6. Napisati izvještaj o rezultatima izvođenja prema zadanom predlošku koji se nalazi na stranicama kolegija. U izvještaju opisati korištenu fitness funkciju i navesti koja vrsta gena je odabrana i zašto.
Potrebno je napisati zasebnu tablicu s rezultatima za svaku korištenu konfiguraciju te napisati zaključak zbog čega određena konfiguracija genetskog algoritma daje najbolje, odnosno najlošije rezultate. **U sklopu zaključka grafički prikazati ovisnosti korištenih parametara o rezultatima**.
Također, potrebno je napraviti usporedbu najboljih dobivenih rezultata za manje i veće ploče, kao i usporedbu vremena potrebnog za pronalazak rješenja konvencionalnom metodom i genetskim algoritmom.

UPUTE:

1. Za bolje razumijevanje korištenja klasa iz programskog okvira „jGAP" koristiti pripadajući javadoc (dolazi zajedno s programskim okvirom) i tutorial koji se može pronaći na sljedećim mrežnim stranicama:
https://karczmarczuk.users.greyc.fr/TEACH/IAD/java/jgap_tutorial.html

2. Za dobivanje boljih rezultata potrebno je proširivati ili mijenjati konfiguraciju genetskog algoritama (korak 4.c) pomoću različitih operatora rekombinacije (engl. *crossover operators*) i operatora mutacije (engl. *mutation operators*). Osim toga je potrebno mijenjati i veličinu populacije i opcije poput elitizma i sl. kako bi se dobili što bolji rezultati.
3. Fitness funkcija za ovakav tip zadatka treba biti definirana tako da pronalazi u potpunosti točno rješenje (a ne samo „dovoljno dobro") jer cijelo vrijeme znamo što točno rješenje je – da se kraljice međusobno ne napadaju. Također, kod evaluacije potencijalnih rješenja, treba paziti na to da se netočna rješenja kažnjavaju (ili „točnija" nagrađuju) ovisno o tome koliko su blizu konačnom rješenju, a ne da sva netočna budu okarakterizirana kao jednako „loša". Nedovoljnom gradacijom takvih netočnih rješenja, u iduću se generaciju ne prenose zbilja najbolja svojstva prethodne generacije.