Úloha

Majme hľadača pokladov, ktorý sa pohybuje vo svete definovanom dvojrozmernou mriežkou (viď. obrázok) a zbiera poklady, ktoré nájde po ceste. Začína na políčku označenom písmenom S a môže sa pohybovať štyrmi rôznymi smermi: hore - H, dole - D, doprava - P a doľava - L. K dispozícii má konečný počet krokov. Jeho úlohou je nazbierať čo najviac pokladov. Za nájdenie pokladu sa považuje len pozícia, pri ktorej je hľadač aj poklad na tom istom políčku. Susedné políčka sa neberú do úvahy.

Zadanie

Horeuvedenú úlohu riešte prostredníctvom evolučného programovania nad virtuálnym strojom.

Tento špecifický spôsob evolučného programovania využíva spoločnú pamäť pre údaje a inštrukcie. Pamäť je na začiatku vynulovaná a naplnená od prvej bunky inštrukciami. Za programom alebo od určeného miesta sú uložené inicializačné údaje (ak sú nejaké potrebné). Po inicializácii sa začne vykonávať program od prvej pamäťovej bunky (prvou je samozrejme bunka s adresou 000000.) Inštrukcie modifikujú pamäťové bunky, môžu realizovať vetvenie, programové skoky, čítať nejaké údaje zo vstupu a prípadne aj zapisovať na výstup. Program sa končí inštrukciou na zastavenie, po stanovenom počte krokov, pri chybnej inštrukcii, po úplnom alebo nesprávnom výstupe. Kvalita programu sa ohodnotí na základe vyprodukovaného výstupu alebo, keď program nezapisuje na výstup, podľa výsledného stavu určených pamäťových buniek.

Reprezentácia údajov

Mapa – mapu si budem v programe reprezentovať ako 1D int pole, ďalej mapa je trieda, ktorá má uložené pozície pokladov, ktoré sú reprezentované taktiež 1D int poľom, štart, veľkosť mapy a počet pokladov sú reprezentované int hodnotou. Stačí mi veľkosť mapy keďže mapa znázorňuje štvorec a pomocou výpočtu si viem vypočítať, či jedinec bude, respektíve nebude v mape.

Jedinec – je trieda, ktorá bude mať gettery a settery pre gény, fitnes hodnotu, nájdené poklady a cestu, ktorú spravil. Gény reprezentujem ako int pole, ktoré naplním dekadickými číslami, ktoré keď je potrebné, si viem previesť do binárnych čísel pomocou bitových operácií. Fitnes hodnotu reprezentujem int číslom nastaveným na začiatok na hodnotu 0. Ak jedinec vyšiel z mapy tak sa zníži o 1, ale ak nájde poklad, tak sa zvýši o 10. Nájdené poklady sa taktiež reprezentujú int číslom nastaveným na 0 a keď sa nájde poklad, zvýši sa o 1. Cestu reprezentujem StringBuilderom, pomocou funkcie append postupne vyskladám cestu, ktorú jedinec prešiel.

Populácie – populáciu tvoria viacerí títo jedinci. Populáciu reprezentujem arraylistom týchto jedincov. Jedna populácia je v podstate jedna generácia. Z arraylistu sa postupne prechádza každý jedinec a posiela sa do stroja.

Virtuálny stroj – virtuálny stroj je takisto trieda, v ktorej je metóda na vykonávanie inštrukcií jedinca. Táto metóda spracuje jedinca, vykoná inštrukcie podľa zadania a vráti jedinca s fitnes hodnotou, cestou ktorú prešiel, počet pokladov ktoré našiel a upravené gény.

Opis riešenia

Program je naprogramovaný v programovacom jazyku java.

Začínam v triede Main. Na začiatku v tejto triede volám funkciu createPopulation, ktorá mi vytvorí populáciu obsahujúcu 60 jedincov, každému z nich vygenerujem náhodné gény. Jedinec je objekt a tak teda mám arraylist objektov, jedincov. Tento arraylist predstavuje populáciu. Predtým ako z populácie pošlem každého jedinca do stroja, uložím si túto populáciu, arraylist, aby som mal uložené gény jedincov pred vykonaním v stroji a mohol ich neskôr krížiť. Jedinec, ktorý sa vráti zo stroja, tak sa ho hneď spýtam, či našiel všetky poklady, ak áno tak vypíšem v ktorej generácií to bolo, akú cestu prešiel, jeho fitnes hodnotu a aj to že našiel všetkých 5 pokladov následne ukončím vykonávanie programu. Ak nenašiel všetkých 5 pokladov program pokračuje vo vykonávaní. Ak prejdem v cykle všetkých jedincov v populácií a každý z nich prešiel aj strojom, tak som prešiel prvú generáciu a potom môžem prejsť na kríženie génov. Popritom vypisujem najlepšieho jedinca z každej generácie.

Trieda VirtualMachine v ktorej sa vykonávajú preddefinované inštrukcie pre virtuálny stroj pomocou metódy runMachine. Tá pre jedinca vykoná inštrukcie zo zadania. V tejto metóde si nastavím premenné pre mapu, rozmery a potrebné súradnice pre štart a do poľa si uložím pozície pokladov z triedy Map. Nasleduje cyklus pre vykonanie 500 inštrukcií. Na začiatku tohto cyklu si vyberiem jeden gén z jedinca z pozície 0, túto pozíciu budem zvyšovať o 1 po každej iterácii cyklu, alebo sa zmení na hodnotu, ktorú zmení inštrukcia skok. Ďalej z tohto génu pomocou bitových operácií získam zľava prvé dva bity, ktoré mi predstavujú inštrukciu. Toto int číslo inštrukcie prevediem do stringu, aby som ju mohol porovnávať a zistiť konkrétnu inštrukciu, keďže porovnávam stringy pre zistenie konkrétnych inštrukcií. Určím si ešte adresu, teda na akej pozícií budem meniť gén danému jedincovi. Nasledujú if podmienky pre inštrukcie, preto som inštrukcie prevádzal do stringu a zistím, či mám inkrementovať, dekrementovať, skočiť alebo vypísať. Na ikrementovanie a dekrementovanie mám metódy, ktoré vykonajú požadovanú inštrukciu na danej adrese, pozmenia gén príslušnou inštrukciou. Inštrukcia skoku priradí adresu odkiaľ sa bude čítať nasledujúci gén. Pri inštrukcii na výpis je viacero if podmienok, najprv sa spraví and operácia načítaného génu s číslom 11. Pomocou tohto zistím konkrétne číslo, ktoré mi reprezentuje smer. Ak viem smer, tak predtým, ako vykonám pohyb, zistím, či ho môžem uskutočniť a či po tomto kroku nebudem mimo mapy. Ak budem mimo mapy, tak fitnes znížim o jedna a cyklus breaknem, ale ak budem v mape, tak pridám smer, napr. H do stringu z ktorého postupne vyskladám cestu, kadiaľ jedinec išiel. Následne zistím, či som stúpil na poklad a ak áno, tak fitnes zvýšim o 10 a odstránim poklad z poľa, kde mám uložené poklady, aby som ho nepočítal dvakrát ak naň ešte raz stúpim. Ak prejdem 500 inštrukcií alebo som vyšiel mimo mapy alebo som našiel už všetky poklady, tak sa tento cyklus ukončí a danému jedincovi priradím cestu kadiaľ išiel, fitness hodnotu a počet nájdených pokladov.

Celá populácia prešla virtuálnym strojom a následne je pripravená na kríženie, ktoré prebieha v triede CrossGenes. Táto trieda má metódu na kríženie s názvom evolute a atribúty tejto funkcie sú populácia, ktorá má už fitness hodnoty, cestu, počet pokladov ale aj upravené gény a preto posielam do tejto funkcie aj base_population, ktorá má pôvodné gény týchto jedincov ale nemá fitnes, cestu a počet pokladov. V tejto metóde sa nachádza cyklus, ktorý ide do veľkosti populácie. V každej iterácii tohto cyklu sa vyberú pomocou rulety dvaja rodičia ktorých budem krížiť. Následne ich pošlem aj s base_population do metódy na vykonanie kríženia. V tejto metóde vyberiem z base_population konkrétnych jedincov, tých ktorých určia čísla rodičov. Krížim ich nasledovných spôsobom – cyklus ide do 64 a v každom cykle generujem náhodne číslo od 0 do 1 a ak sa náhodne vygenerované číslo rovná 0, gén sa bude brať od prvého rodiča a ak sa číslo rovná 1, gén príde z druhého rodiča. Po ukončení cyklu vrátim zkrížené gény, pole a priradím ho jedincovi, ktorého následne pridám do novej populácie do arraylistu. Takto prejdem celú populáciu. Po prejdeni celej populácie zmutujem túto novo vytvorenú populáciu. Mutovať budem dvoch náhodne vybraných jedincov a každému z týchto jedincov v cykle, ktorý ide do 64, vygenerujem nové náhodné gény.

Následne sa prejde na druhú generáciu, ktorá prebieha popisovaným spôsobom. Ak sa vygeneruje 500 generácií, program sa spýta, či sa má v evolúcii pokračovať, alebo sa má ukončiť.