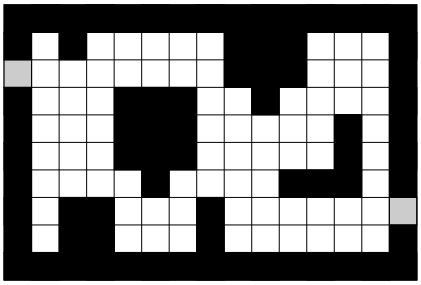
Zadanie 1.
Algorytmy genetyczne w poszukiwaniu drogi w labiryncie

1 Problem

Napisać program implementujący prosty algorytm genetyczny przeznaczony do poszukiwania drogi w labiryncie.

2 Reprezentacja problemu

Obszar poszukiwań pokrywamy kwadratową kratą. Liczba oczek kraty zależy od stopnia złożoności pejzażu, w którym porusza się agent (np. robot). Przykładowy labirynt przedstawia poniższy rysunek:



Rysunek 1: Przykładowy labirynt. Wejście i wyjście wskazują szare kwadraty. Przeszkody oznaczono czarnymi kwadratami.

Pokrywamy go siatką w kwadratów dobierając odpowiednio rozmiar oczka. W naszym przypadku potrzebujemy 10×15 kwadratów. Do reprezentacji uzyskanej w ten sposób przestrzeni poszukiwań zastosujemy tablicę `int[][]` o wymiarze 10×15. Tablicę tę wypełniamy następująco:

```
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
 2, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1,
 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1,
 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1,
 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 4,
 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
```

Poszczególne wartości mają poniższą interpretację:

`0` – biały kwadrat (pusta przestrzeń)
`1` – czarny kwadrat (przeszkoda)
`2` – szary kwadrat, WEJŚCIE
`4` – szary kwadrat, WYJŚCIE

3 Reprezentacja rozwiązania

Załóżmy, że agent może poruszać się w jednym z czterech kierunków: `Lewo`, `Prawo`, `Góra`, `Dół`. Będziemy je reprezentować w postaci liczb binarnych z zakresu $\{0, \ldots, 3\}$, np:

`00` – `Lewo`
`01` – `Prawo`
`10` – `G`
`11` – `Dół`

W takim przypadku przykładowy ciąg bitów (tzn. chromosom)

```
111110011011101110010100
```

reprezentuje sekwencję 12 ruchów

```
11 11 10 01 10 11 10 11 10 01 01 00
```

której odpowiada 12 rzeczywistych ruchów agenta:

```
D D G P G D G D G P P L
```

Aby zrealizować daną sekwencję, ustawiamy agenta w punkcie startowym i wykonujemy kolejne ruchy. Jeżeli dany ruch sprawia, że agent trafia w przeszkodę, ignorujemy go i wykonujemy kolejny w danej sekwencji ruch. Postępowanie to jest kontynuowane albo do chwili wykonania ostatniego genu (pozycji w chromosomie), albo do osiągnięcia wyjścia z labiryntu (zależy, które z tych zdarzeń zajdzie jako pierwsze).

4 Ocena rozwiązania

Do oceny chromosomu stosujemy odległość Manhattan między punktem (X_i, Y_i), w którym dany agent (o numerze i) zatrzymał się po wykonaniu instrukcji zawartych w tym chromosomie, a punktem (X_g, Y_g) reprezentującym współrzędne wyjścia. Jakość f_i i-tego chromosomu jest odwrotnie proporcjonalna do tejże odległości, tzn.

$$f_i = \frac{1}{|X_i - X_g| + |Y_i - Y_g| + 1}$$

5 Parametry

Przyjąć następujące parametry:

- długość chromosomu $l = 70$, tzn. agent może wykonać maksymalnie 35 ruchów;
- rozmiar populacji $m = 150$;
- prawdopodobieństwo krzyżowania $p_c = 0.7$
- prawdopodobieństwo mutacji $p_m = 0.7$

6 Pytania

(a) Sprawdzić jak szybko program znajduje rozwiązania dla podanego wyżej zestawu parametrów. Wykonać wykres obrazujący wartość dopasowania najlepszego osobnika w populacji w kolejnych iteracjach. Czy zmniejszenie rozmiaru populacji do 50 osobników wpływa na szybkość znalezienia rozwiązania? W celu podania miarodajnej odpowiedzi wykonać 30 przebiegów algorytmu genetycznego i podać uśrednioną liczbę iteracji koniecznych do znalezienia rozwiązania.

(b) Porównać selekcję proporcjonalną z selekcją turniejową. W selekcji proporcjonalnej prawdopodobieństwo p_i wybrania i-tego osobnika jest równe

$$p_i = \frac{f_i}{\sum_{j=1}^{m} f_j}$$

W przypadku selekcji turniejowej aby wybrać pierwszego rodzica losujemy (bez powtórzeń) $k \geqslant 2$ osobników i zwracamy najlepszego (o największej wartości funkcji f) z nich. Do wyboru drugiego rodzica stosujemy analogiczną metodę. Sprawdzić wpływ rozmiaru turnieju k na jakość rozwiązania.

(c) Porównać efektywność algorytmu genetycznego z algorytmem A^*.

UWAGA: Program, najlepiej w języku `Java`, powinien być zaopatrzony w komentarze dotyczące roli ewentualnych klas, używanych w nich poszczególnych metod, oraz przeznaczenia użytych zmiennych.