Sztuczna inteligencja
Pracownia 3
Zajęcia w tygodniu 9-15 kwietnia

Można używać dowolnego języka programowania. Modelowe rozwiązania pisane były w Pythonie. Dostępna jest sprawdzaczka dla zadań 1 i 2, we wtorek po Świętach powinna być również sprawdzaczka dla pozostałych zadań (mniej pracochłonnych).

Zadanie 1. (6p) W zadaniu tym powinieneś dopisać moduł wnioskujący do obrazków logicznych (czyli procedurę, która analizując specyfikację np. wiersza jest w stanie dedukować informacje o zawartości poszczególnych pikseli, a następnie korzystając z tej wiedzy dedukować wartość innych pikseli rozważając specyfikacje kolumn, i tak dalej).

Testy do tego zadania są w większości powtórzeniem testów z zadania P2.1, taka sama jest też specyfikacja danych wejściowych i oczekiwanego wyniku. Inne limity czasowe. Możesz założyć, że w tym zadaniu samo wnioskowanie jest w stanie wydedukować wartość każdego piksela.

Zadanie 2. (5p) W tym zadaniu nadal rozwiązujesz obrazki logiczne, ale w rozwiązaniu powinien być użyty backtracking (testy do tego zadania będą dobrane w ten sposób, że najprawdopodobniej program z poprzedniego zadania nie będzie w stanie wypełnić wszystkich pikseli). Rozwiązanie modelowe, mieszczące się w limitach czasowych, przeplatało wnioskowanie i backtracking.

Zadanie 3. (1p, $\star$) Zgłoś na SKOS pewną liczbę przypadków testowych dla obrazków logicznych, w formacie znanym Ci ze sprawdzaczki. Obowiązują następujące zasady:

a) Niedozwolone są przypadki testowe będące częścią którejś wersji sprawdzaczki na SKOSie.

b) Twój program powinien przechodzić każdy z tych przypadków testowych.

c) Limit czasowy powinien być dwukrotnością czasu działania Twojego programu.

d) Obrazek powinien mieć co najmniej 300 pikseli.

Twój program może mieć heurystyki w jakiś sposób dostosowane do tych przypadków, ale nie może mieć wpisanych do kodu żadnych fragmentów rozwiązań. Możesz korzystać z obrazków opublikowanych, na przykład w Internecie.

Zadanie 4. (0-4p, $\star$) To zadanie będzie oceniane na **kolejnej** liście pracowniowej. Liczba punktów zależy liniowo od liczby przypadków testowych, które program przejdzie w limicie czasu (można wykonać testy 5-krotnie i wpisać maksimum). Przypadki testowe będą pochodziły ze zgłoszeń Studentów (i ewentualnie od Prowadzących).

Zadanie 5. (1p) To zadanie jest łatwym przygotowaniem do zadania kolejnego. Rozważamy w nim problem Sudoku. Na stronie znajdziesz (prawie kompletny) program sudoku.py, który produkuje, dla konkretnej instancji łamigłówki sudoku, program w Prologu sudoku.pl, rozwiązujący tę łamigłówkę. Uzupełnij program sudoku.py (instrukcje znajdziesz w komentarzach w kodzie), tak aby przeszedł testy.
Opis formatu danych: opis łamigłówki składa się z 9 wierszy, każdy po 9 znaków. Znakami są cyfry (oznacza to, że w łamigłówce na tym miejscu jest wpisana cyfra), oraz kropki. Przykładowy opis łamigłówki (podobno trudnej):

```
3.......1
4..386...
.....1.4.
6.924..3.
..3......
......719
........6
2.7...3..
```

Zadanie 6. (4p) W tym zadaniu powinieneś napisać program, który rozwiązuje łamigłowkę burze (opis łamigłówki znajdziesz na slajdach z wykładu, W5, s21-29). Twój program, dla każdej instancji

zadania, powinien wypisywać program w SWI-Prologu, który rozwiązuje tę instancję (czyli wypisuje jedno rozwiązanie, jako listę kolejnych zer i jedynek).
Opis formatu danych: Pojedynczy przypadek testowy zwiera:

- Opis wierszy (czyli k liczb, w jednym wierszu)
- Opis kolumn (czyli m liczb, w jednym wierszu)
- Pewną liczbę deklaracji o wypełnionych polach, po jednej w wierszu. Deklaracje mają postać:

```
<nr_wiersza> <nr_kolumny> <0_lub_1>
```

Na stronie znajduje się również program storms.py[1], który (w trywialny sposób) rozwiązuje jeden przypadek testowy, co powinno wyjaśnić wątpliwości związane z formatem.

[1] Choć przykład jest w Pythonie, rozwiązując zadanie możesz wybrać dowolny język.