Вештачка интелигенција

Домашна задача 1

Љубица Дамјановиќ - 221173

Проблем 1 – Проблем со пребарување на простор на состојби (Децата на Пакман)

Елементи:

1. Големина на таблата – M*N
2. Торка од k (број на деца на Пакман) торки од тековните координати на секое дете на Пакман
3. Позицијата на таткото Пакман – g
4. Торка од r (број на пелетки) торки од координати на преостанатите пелети кои треба да се изедат
5. Горе, Долу, Лево и Десно – акции кои може да ги преземе секое дете на Пакман
6. Ѕидови – полиња на таблата на кои не смее да има Пакман/пелета

А) Во initial влегуваат сите елементи кои се менливи. Во минималната состојба (initial) би се чувало:

initial = (pacmans_coordinates, pellets_coordinates)
pacmans_coordinates – торка од k торки за позициите на секое дете на Пакман
pellets_coordinates – торка од r boolean вредности за позициите на секоја пелета (ако има пелета на одредена позиција – True, во спротивно False)
Големината на таблата, координатите на ѕидовите и позицијата на таткото Пакман не влегуваат во минималната/иницијалната состојба поради тоа што тие не се менуваат – статични се со секоја преземена акција

Б) $(M * N)^k * 2^r$

$M * N$ – големината на таблата
k – број на деца на Пакман
r – број на пелети
$(M * N)^k$ – број на можни позиции за секој пакман на степен бројот на пакмани
2^r – можни вредности за dot boolean векторот основата е 2 поради 2 можни вредности True или False
Првиот степенуван дел ке биде всушност е добиен од првата торка од initial која има облик $((x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_k, y_k)$, каде домените се: x – M и y – N и бидејќи се k деца на Пакманот затоа и овој израз е на степен k.
Вториот степенуван дел е всушност од облик $(1_1, 1_2, 0_3, \ldots, 1_r)$

В) 4^k – бидејќи има к деца на Пакман и можни акции за секое од нив е Горе, Долу, Лево и Десно и во секоја состојба сите k деца на Пакманот преземаат некоја акција.

Г) Почетна состојба:

(((0, 2), (3, 0), (2, 5), (8, 2)), (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1))

Целна состојба: Сите торки од pacmans_coordinates да бидат еднакви на позицијата на таткото Пакман (g) и сите елементи во dot boolean векторот да бидат нули (False)

Д) Во секоја акција, сите деца на таткото Пакман преземаат по една акција (move) – (Горе, Долу, Лево и Десно). Легални акции во секоја состојба (state) се оние каде:

1. Ниту едно дете не излегува надвор од лавиринтот
2. Ниту едно дете не стапнува на ѕид
3. Ниту едно дете да не стапне на исто поле како друго дете освен на полето g каде се наоѓа таткото Пакман

Ѓ)

$$h_a: \frac{\sum_{i=1}^{k} MH(p_i, g)}{k}$$

- Оваа евристика ни го дава збирот од сите Менхетен растојанија од секоја пелета до позицијата на таткото Пакман (g) делено со бројот на деца на Пакман (просечното Менхетен растојание). Оваа евристика не би била допустлива ради фактот што нема смисла – не ги води децата до пелетите.

$$h_b: \max_{1 \le i \le k} MH(p_i, g)$$

- Оваа евристика ни го дава максималното Менхетен растојание од кое било дете на Пакман до позицијата на таткото Пакман (g). Оваа евристика е релативно допустлива поради фактот што барем едно дете ќе мора да го помине ова растојание за да стигне до целта и претставува долна граница на вистинската цена (ова би било доколку во целната состојба не се зема предвид дали сите пелети се собрани). Бидејќи целта исто така надовестува дека сите пелети треба да се соберат, тогаш ниту оваа евристика не би била допустлива поради фактот што нема смисла, односно не се земени во предвид пелетите коишто треба да се соберат.

$$h_c: \max_{1 \le i \le k} \left[\max_{f \in F} MH(p_i, f) \right]$$

- Оваа евристика ни го дава максималното Менхетен растојание од секое дете до најоддалечената пелета за секое дете соодветно. Оваа хевристика е допустлива поради тоа што барем едно дете ќе мора да го помине ова растојание за да ги собере сите пелети и претставува долна граница на вистинската цена.

$$h_d: \max_{1 \le i \le k} \left[\min_{f \in F} MH(p_i, f) \right]$$

- Оваа евристика ни го дава максималното Менхетен растојание од секое дете до најблиската пелета за секое дете соодветно. Оваа хевристика не е допустлива поради тоа што може во некоја мера да се потцени вистинската цена, тоа би било пример кога сите пелети се распрснати и потребни се повеќе чекори за да се соберат истите.

$$h_e: \min_{1 \le i \le k} \left[\min_{f \in F} MH(p_i, f) \right]$$

- Оваа евристика ни го дава минималното Менхетен растојание од секое дете до најблиската пелета за секое дете соодветно. Оваа хевристика не е допустлива поради истите причини од претходната – може да ја потцени вистинската цена.

$$h_f: \min_{1 \le i \le k} \left[\max_{f \in F} MH(p_i, f) \right]$$

- Оваа евристика ни го дава минималното Менхетен растојание од секое дете до најдалечната пелета за секое дете соодветно. Оваа хеврситки е релативно допустлива поради тоа што барем едно дете на Пакман ќе треба да го помине ова растојание за да ги собере пелетите и претставува долна граница на вистинската цена. Оваа хевристика е малце помалку информативна од евристиката с каде се бара максималното растојание.

Е) Нека h(g) е евристиката со состојба g (моменталната состојба на таблата – state) и нека МН(p_i, p_j) е Менхетен растојанието од p_i до p_j. Предлог хевристика:

$$h(g) = \min_{1 \le x \le k} [\sum_{f \in F} MH\,(p_i, f)]$$

Оваа евристика нам всушност ни дава минималната сума од сумите на растојанијата на сите деца до секоја пелета и ни ја дава долната граница на купното растојание што треба да го поминат сите деца на Пакман за да ги соберат сите преостанати пелетки. Оваа евристика не е тривијална поради тоа што само кога ќе се соберат сите пелетки h(g)=0 и тоа ќе премине од A* во UCS при самиот крај, односно не во секој можен чекор евристиката враќа константа.

Доказ дека оваа евристика е допустлива:

За да се докаже дека оваа евристика е допустлива, треба да докажеме дека таа никогаш нема да ја надмине вистинската цена до целната состојба (нема да ја прецени). Нека z(g) е оптималната цена за да се стигне до целната состојба g. Во сите целни состојби, сите пелетки се веќе собрани, што значи дека секое дете на Пакман треба да го помине барем растојанието кое е сумата на сите растојанија од неговата почетна позиција до сите пелетки. Односно за секое дете i:

$$z(g) \ge \sum_{f \in F} MH\,(p_i, f)$$

Сега, бидејќи предлог евристиката h(g) ја зема минималната сума, тогаш следува дека h(g) ≤ z(g) и според ова евристиката никогаш нема да ја прецени вистинската цена до целта, така што следува дека таа е допустлива.

Ж)

1. DFS (Depth First Search)
   Овој алгоритам ги истражува патиштата до максималната длабочина пред да направи backtracking и ги истражи останатите. Со оглед на тоа овој алгоритам не гарантира најоптимално решение, особено во овој проблем каде има повеќе можни патеки. DFS може лесно да заглави во длабоки, непродктивни патишта и тоа го прави несоодветен за решавање на овој проблем.
2. UCS (Uniform Cost Search)
   Овој алгоритам ги истражува патиштата по растечки редослед според нивниот трошок (цена). Овој алгоритам релативно гарантира наоѓање на оптимално решение кога сите чекори имаат позитивна цена. Во овој проблем, UCS би можел да ги распределува децата на Пакман кон различни пелетки, при тоа секогаш избирајќи го најефективниот

пат, но бидејќи во овој проблем имаме голем број на можни патеки и состојби, користењето на овој алгоритам не би дошло во предвид поради фактот што би бил многу захтевен, односно потребно е повеќе време и меморија за да се најде најоптималното решение.

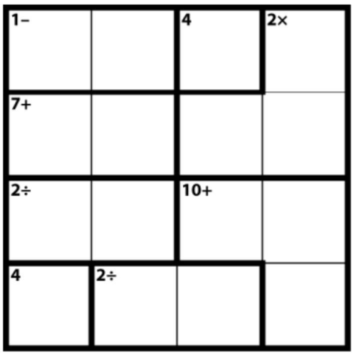
3. BFS (Breadth First Search)

   Овој алгоритам ги истрашува сите нивоа од просторот на состојби едно по едно и гарантира изнаоѓање на оптимално решение.

4. A*

   Овој проблем е најубаво да се реши со овој алгоритам. Доколку евристиката која ја користиме е тривијална допустлива, тогаш овој алгоритам ви се однесувал како BFS. А доколку евристиката е нетривијална допустлива (пример евристиката која ја предложив во претходната точка) тогаш за овој проблем ќе се најде најоптимално решение во помал број на чекори за разлика од другите алгоритми. Односно овој алгоритам ќе најде решение секогаш кога евристиката е допустлива.

Проблем 2 – Проблеми кои исполнуваат услови (КенКен)



А)

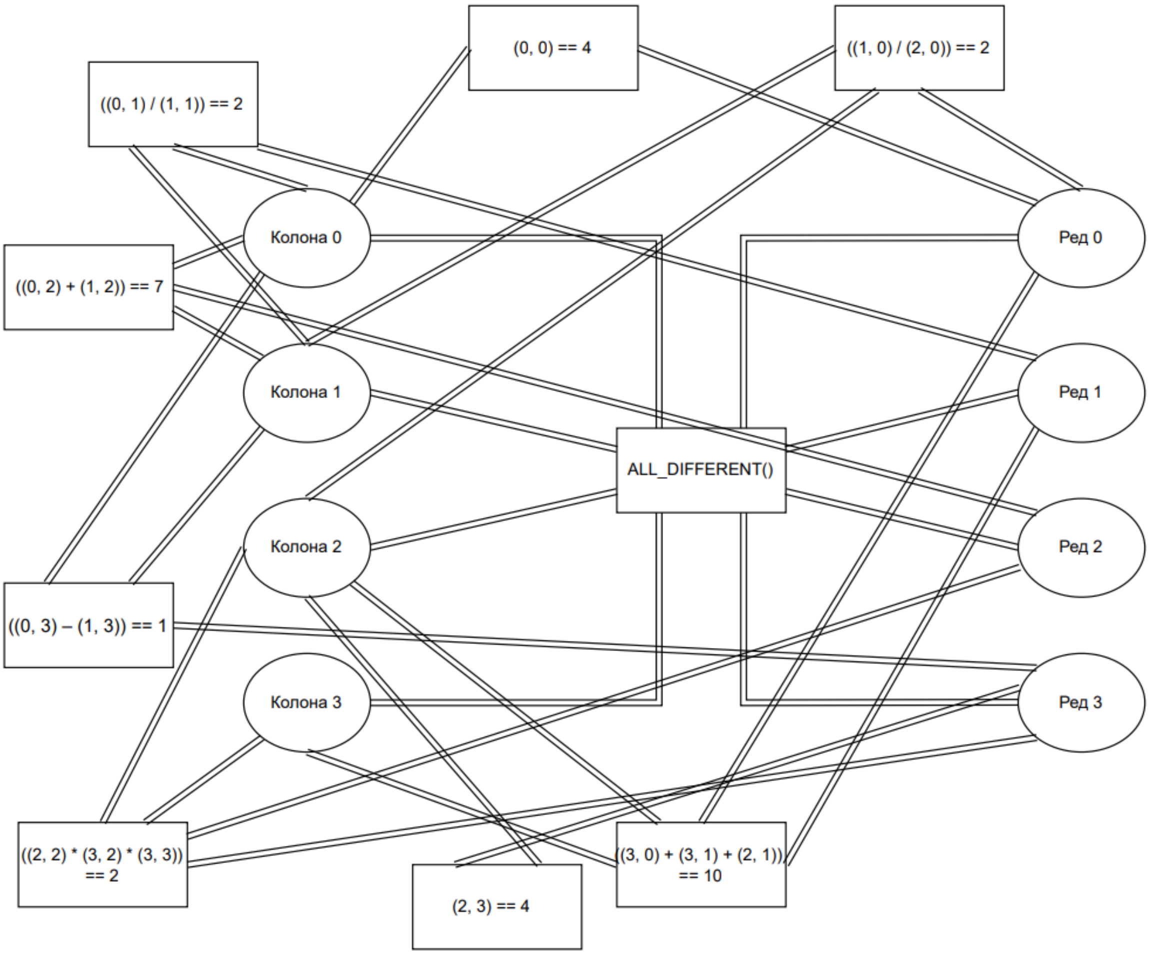
Променливи: сите позиции – (0, 0), (0, 1), (0, 2), (0, 3), (1, 0), (1, 1), (1, 2), (1, 3), (2, 0), (2, 1), (2, 2), (2, 3), (3, 0), (3, 1), (3, 2), (3, 3)

Домен: {1, 2, 3, 4}

Услови:

- AllDifferentConstraint за сите променливи што се наоѓаат во секоја редица и во секоја колона соодветно
- (0, 0) == 4
- ((1, 0) / (2, 0)) == 2
- ((3, 0) + (3, 1) + (2, 1)) == 10
- ((0, 1) / (1, 1)) == 2
- ((0, 2) + (1, 2)) == 7
- ((0, 3) – (1, 3)) == 1
- (2, 3) == 4
- ((2, 2) * (3, 2) * (3, 3)) == 2

Б)



В)

Чекор	(0, 0)	(0, 1)	(0, 2)	(0, 3)	(1, 0)	(1, 1)	(1, 2)	(1, 3)
1	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
2	4	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
3	4	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3

(2, 0)	(2, 1)	(2, 2)	(2, 3)	(3, 0)	(3, 1)	(3, 2)	(3, 3)	Услов
1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	~
1, 2, 3	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	(0, 0) == 4
1, 2, 3	1, 2, 3	1, 2, 3	4	1, 2, 3	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3	(2, 3) == 4

Г)

Реброто ((0, 0), (0, 1)) е конзистентно затоа што се можна е вредноста {4}, додека пак вредностите {1, 2, 3} не се дозволени.

Реброто ((0, 1), (0, 0)) е конзистентно затоа што се можни се е вредностите {1, 2, 3}, додека пак вредноста {4} не е дозволена.

Реброто ((0, 0), (0, 2)) е конзистентно затоа што се можна е вредноста {4}, додека пак вредностите {1, 2, 3} не се дозволени.

Реброто ((0, 2), (0, 0)) е конзистентно затоа што се можни се е вредностите {1, 2, 3}, додека пак вредноста {4} не е дозволена.

Реброто ((0, 0), (0, 3)) е конзистентно затоа што се можна е вредноста {4}, додека пак вредностите {1, 2, 3} не се дозволени.

Реброто ((0, 3), (0, 0)) е конзистентно затоа што се можни се е вредностите {1, 2, 3}, додека пак вредноста {4} не е дозволена.

Реброто ((0, 0), (1, 0)) е конзистентно затоа што се можна е вредноста {4}, додека пак вредностите {1, 2, 3} не се дозволени.

Реброто ((1, 0), (0, 0)) е конзистентно затоа што се можни се е вредностите {1, 2, 3}, додека пак вредноста {4} не е дозволена.

Реброто ((0, 0), (2, 0)) е конзистентно затоа што се можна е вредноста {4}, додека пак вредностите {1, 2, 3} не се дозволени.

Реброто ((2, 0), (0, 0)) е конзистентно затоа што се можни се е вредностите {1, 2, 3}, додека пак вредноста {4} не е дозволена.

Реброто ((0, 0), (3, 0)) е конзистентно затоа што се можна е вредноста {4}, додека пак вредностите {1, 2, 3} не се дозволени.

Реброто ((3, 0), (0, 0)) е конзистентно затоа што се можни се е вредностите {1, 2, 3}, додека пак вредноста {4} не е дозволена.

Реброто ((2, 3), (3, 3)) е конзистентно затоа што се можна е вредноста {4}, додека пак вредностите {1, 2, 3} не се дозволени.

Реброто ((3, 3), (2, 3)) е конзистентно затоа што се можни се е вредностите {1, 2, 3}, додека пак вредноста {4} не е дозволена.

Реброто ((2, 3), (1, 3)) е конзистентно затоа што се можна е вредноста {4}, додека пак вредностите {1, 2, 3} не се дозволени.

Реброто ((1, 3), (2, 3)) е конзистентно затоа што се можни се е вредностите {1, 2, 3}, додека пак вредноста {4} не е дозволена.

Реброто ((2, 3), (2, 1)) е конзистентно затоа што се можна е вредноста {4}, додека пак вредностите {1, 2, 3} не се дозволени.

Реброто ((2, 1), (2, 3)) е конзистентно затоа што се можни се е вредностите {1, 2, 3}, додека пак вредноста {4} не е дозволена.

Реброто ((2, 3), (2, 2)) е конзистентно затоа што се можна е вредноста {4}, додека пак вредностите {1, 2, 3} не се дозволени.

Реброто ((2, 2), (2, 3)) е конзистентно затоа што се можни се е вредностите {1, 2, 3}, додека пак вредноста {4} не е дозволена.

Ребрата се конзистентни поради тоа што нема вредности на полињата со исти можни вредности по исполнување на унарните услови и може сите полиња да добијат соодветни резултати.

Д)

Услови:

- AllDifferentConstraint за сите променливи што се наоѓаат во секоја редица и во секоја колона соодветно
- (0, 0) == 4
- ((1, 0) / (2, 0)) == 2
- ((3, 0) + (3, 1) + (2, 1)) == 10
- ((0, 1) / (1, 1)) == 2
- ((0, 2) + (1, 2)) == 7
- ((0, 3) – (1, 3)) == 1
- (2, 3) == 4
- ((2, 2) * (3, 2) * (3, 3)) == 2

Доделување на вредности

Чекор	(0, 0)	(0, 1)	(0, 2)	(0, 3)	(1, 0)	(1, 1)	(1, 2)	(1, 3)	(2, 0)	(2, 1)	(2, 2)	(2, 3)	(3, 0)	(3, 1)	(3, 2)	(3, 3)
1	**4**	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
2	4	~	~	~	~	~	~	~	~	~	~	**4**	~	~	~	~
3	4	~	~	~	**1**	~	~	~	~	~	~	4	~	~	~	~
4	4	~	~	~	1	~	~	**3**	~	~	~	4	~	~	~	~
5	4	~	~	**2**	1	~	~	3	~	~	~	4	~	~	~	**1**
6	4	**1**	**3**	2	1	~	~	3	~	~	~	4	~	~	~	1
7	4	1	3	2	1	**2**	**4**	3	**2**	**3**	**1**	4	**3**	**4**	**2**	1

Седмата редица е крајното решение.

Останати можни вредности

Чекор	(0, 0)	(0, 1)	(0, 2)	(0, 3)	(1, 0)	(1, 1)	(1, 2)	(1, 3)
1	~	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
2	~	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3
3	~	1, 2, 3	1, 2, 3	1, 2, 3	~	2, 3, 4	2, 3, 4	2, 3
4	~	1, 2, 3	1, 2, 3	1, 2	~	2, 4	2, 4	~
5	~	1, 3	1, 3	~	~	2, 4	2, 4	~
6	~	~	~	~	~	2, 4	2, 4	~
7	~	~	~	~	~	~	~	~

Чекор	(2, 0)	(2, 1)	(2, 2)	(2, 3)	(3, 0)	(3, 1)	(3, 2)	(3, 3)
1	1, 2, 3	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
2	1, 2, 3	1, 2, 3	1, 2, 3	~	1, 2, 3	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3
3	2, 3	1, 2, 3	1, 2, 3	~	2, 3	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3
4	2, 3	1, 2, 3	1, 2, 3	~	2, 3	1, 2, 3, 4	1, 2, 3, 4	1, 2
5	2, 3	1, 2, 3	1, 2, 3	~	2, 3	2, 3, 4	2, 3, 4	~
6	2, 3	2, 3	1, 2	~	2, 3	2, 3, 4	2, 4	~
7	~	~	~	~	~	~	~	~

Користената евристика во сите чекори е MRV – Minimum Remaining Values и доколку има променливи со ист број на преостанати вредности се одбира лексикографски (пример ако (0,1) и (2,3) имаат ист број на преостанати вредности, се зема (0,1)). Додека пак за избирање на вредноста се користи LCV – Least Constraining Value и доколку има вредности со ист број на попречувачки вредности, тогаш се зема пример помалата и доколку таа не доведува до вистинското решение се навраќаме назад и се одбира другата.