ПРВА ДОМАШНА ЗАДАЧА ПО ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА – ГРУПА 1

ПРОБЛЕМИ СО ПРЕБАРУВАЊЕ НА ПРОСТОРОТ НА СОСТОЈБИ И ЗАДОВОЛУВАЊЕ НА УСЛОВИ

1. Проблем со пребарување на простор на состојби – Пакман на ролери

а) Минималната репрезентација на состојбата на проблемот може да се дефинира како K двојки на вредности ((x, y)), при што секоја таква двојка соодветствува на координатите на еден пакман редоследно. Оваа минимална репрезентација е соодветна бидејќи гледајќи ја квадратната површина на која се наоѓаат пакманите како координатен систем со почеток во 1, доволно е да ја знаеме позицијата на секој од пакманите за да можеме истите да ги придвижуваме кон својата целна состојба.

б) Во просторот на состојби за горенаведената дефиницја, максималниот број на состојби ќе изнесува $(NxN)^K$, каде што NxN претставува големината на квадратната површина, а K е бројот на пакманите на таблата. Односо, со оглед на тоа дека немаме одредени придвижувачки духови или пак насоки кон кои е моментално свртен секој од пакманите, за најголемиот број на можни сосјтојби ги користиме само големината на таблата и бројот на пакманите.

в) Со оглед на тоа дека за секој пакман има 5 можни акции (горе, долу, лево, десно и стоп), за K Пакмани максималната вредност на факторот на разгранување изнесува 5^K - 1. Од бројот на пермутации се одзема 1 бидејќи во единствениот случај каде сите Пакмани стојат, состојбата ќе остане иста, т.е фринџот нема да се прошири.

г) Според наведената дефиниција за состојбата на проблемот, иницијалната и целната состојба дадени на сликите од конкретниот примерот се следните:

- Почетна состојба – ((1, 1), (1, 2), (1, 3))
- Целна состојба – ((3, 3), (3, 2), (3, 1))

Во општ случај тие би биле:

- Почетна состојба – ((1, 1), (1, 2), … , (1, N))
- Крајна состојба – ((N, N), (N, N – 1), … , (N, 1))

д) Во продолжение е даден поедноставен псевдокод за тоа како би изгледале можните акции од произволна состојба на проблемот во која се земени 3 Пакмани (секако, во програмски код би искористил циклус кој би ги изминал сите N Пакмани).

```
directions = ("Горе", "Долу", "Лево", "Десно", "Стоп")
for pacman1 in range (5):
	for pacman2 in range(5):
		for pacman3 in range(5):
			new_pacman1 = move_pacman_to_pos(directions[pacman1])
			new_pacman2 = move_pacman_to_pos(directions[pacman2])
			new_pacman3 = move_pacman_to_pos(directions[pacman3])
			if new_state_is_valid( new_pacman1,  new_pacman2,  new_pacman3):
				successors = (new_pacman1,  new_pacman2,  new_pacman3)
```

Бидејќи сите Пакмани се придвижуваат истовремено, не би можело редоследно да се проверат акциите за секој Пакман (како до сега), па затоа за секој Пакман се земаат во предвид сите акции и доколку се валидни по поместувањето се менува successor функцијата. Се користат помошни функции за поместување на секој Пакман каде се проверува доделената насока и се поместува Пакманот (воедно се проверува ако состојбата на еден Пакман е стоп друг да може да го прескокне). На крајот се проверува валидноста на новата состојба од сите Пакмани со is_valid функција.

ѓ) За поедноставена верзија на овој проблем, во која имаме само еден Пакман на ролери на квадратната површина, чија што цел е да стигне од произволна почетна состојба (1, i) во најдолниот ред, во инвертираната состојба (N, N – i + 1) во горниот ред, соодветна можна евристика е Менхетен растојанието од почетната до целната состојба. Оваа евристика е допустлива (оптимистичка во секоја состојба), бидејќи истата секогаш ќе го дава точниот број на чекори за пакманот да стигне до целта, со оглед на тоа дека пакманот нема да може да прескокнува пречки и да се движи дијагонално, односно ќе може секогаш да се придвижува по точно едно поле во четирите насоки. Можна допустлива хевристика во овој поедноставен случај би била и Евклидовото растојание, но Менхетен растојанието е пооптимално бидејќи добиената вредност од евристиката ќе биде поблиску до вистинскиот број на чекори до целта.

е) За оригиналниот проблем со N Пакмани на ролери, користејќи го Менхетен растојанието како h_i евристиката во следниве примери, (не) допустливи евристики ќе бидат:

- h_a: Збирот на Менхетен растојанијата поделен со бројот на Пакмани ќе биде допустлива евристика, односно, во секој случај ќе дава вредност помала од вистинската цена до целта.
- h_b: Само збирот на сите Менхетен растојанија на сите Пакмани не е допустлива евристика. Доволно е да постои еден случај во којшто евристиката дава поголема цена до целта од вистинската за истата да не е допустлива. Па така, кога сите K пакмани би се наоѓале на позиција веднаш под целната, потребен би бил само 1 чекор за да се стигне до целната состојба, а збирот на Менхетен растојанијата би изнесувал K.
- h_c: Максималното менхетен растојание од сите Пакмани не е допустлива евристика бидејќи одреден Пакман може да прескокне друг во случај кога другиот стои, а исто така можат и да се движат истовремено. Доколку просторот се наоѓа во состојба каде се потребни два чекори до целната состојба и притоа во првиот чекор еден Пакман скокне друг (кој е врз него) кој што е под својата цел и во следниот двата да се

придвижат до целта, максималното менхетен растојание пред првиот чекор ќе изнесува три (поголемо од вистинското – два).

- h_d: Бројот на Пакмани помножен со максималното Менхетен растојание не е допустлива хевристика. Ова следува од заклучокот дека ни максималното Менхетен растојание не е оптимистичко во секој случај.
- h_e: Минимум од Менхетен растојанијата на сите Пакмани е допустлива евристика, бидејќи во секоја можна состојба потребни ќе бидат барем онолку чекори до целта, колку што полиња е оддалечен најблискиот пакман до целта (по Менхетен растојанието).
- h_f: Бројот на Пакмани помножен со минималното Менхетен растојание од сите Пакмани до целта нема да е допустлива евристика од причина што доколку имаме состојба каде што на сите K Пакмани им е потребно едно придвижување нагоре да стигнат до целта, реалната цена до целната состојба би била 1, додека со оваа хевристика би добиле цена K.

ж) Алгоритми за проблемот:

i. DFS е комплетен алгоритам (доколку избегнеме циклуси), но не и оптимален. Односно, одејќи по длабочина, тој секогаш ќе го наѓа “најлевото’’ решение. Во одредени случаи каде што решението се наоѓа најлево во дрвото DFS би бил најбрз, но од друга страна за решение кое се наѓа на спротивната страна ќе трае многу долго.
ii. BFS е комплетен и оптимален (доколку цената на секоја акција е 1). Тој ќе го пронајде оптималното решение до целната состојба, но во многу случаи овој процес ќе трае многу долго бидејќи BFS се разгранува во сите насоки, па ќе посети многу гранки од дрвото.
iii. UCS работи на ист принцип како BFS, со разлика во тоа што UCS прво го проширува нодот со најниска цена. Во случај кога за сите акции имаме цена од 1 (како во овој проблем) UCS ќе го врати истиот резултат како BFS и воедно ќе ги измине истите нодови.
iv. A* алгоритамот за разлика од претходно наведените алгоритми, има информација за локацијата на целта. Овој алгоритам ги комбинира UCS и алчниот алгоритам и во пресметказа за избор на следниот нод ги вклучува и цената до следниот нод и хевристиката. <u>A* е алгоритамот кој би го избрал за оптмално решение и најкратко време за процесирање на овој проблем</u>, бидејќи нема изгубено да се проширува во сите насоки туку ќе води сметка за целната состојба и при секое проширување на фринџот ќе се приближува кон истата.

2. Проблеми кои исполнуваат услови – Задача со подготовка на колоквиум

а) Формално дефинирање на проблемот како проблем на исполнување услови:

Променливи:

- D – домашната задача
- S – самостојна проверка
- L – лабараториска вежба
- T – теоретски дел
- P – програмски дел

Домен:

- 1 – Понеделник
- 2 – Вторник
- 3 – Среда
- 4 – Четврток
- 5 – Петок
- 6 – Сабота
- 7 – Недела

Доделување на услови редоследно:

- Секоја активност треба да е доделена за изработка во различен ден (претставено преку бинарни услови:
    - (lambda a, b: a != b, ("D", "S"))
    - (lambda a, b: a != b, ("D", "L"))
    - (lambda a, b: a != b, ("D", "T"))
    - (lambda a, b: a != b, ("D", "P"))
    - (lambda a, b: a != b, ("S", "L"))
    - (lambda a, b: a != b, ("S", "T"))
    - (lambda a, b: a != b, ("S", "P"))
    - (lambda a, b: a != b, ("L", "T"))
    - (lambda a, b: a != b, ("L", "P"))
    - (lambda a, b: a != b, ("T", "P"))

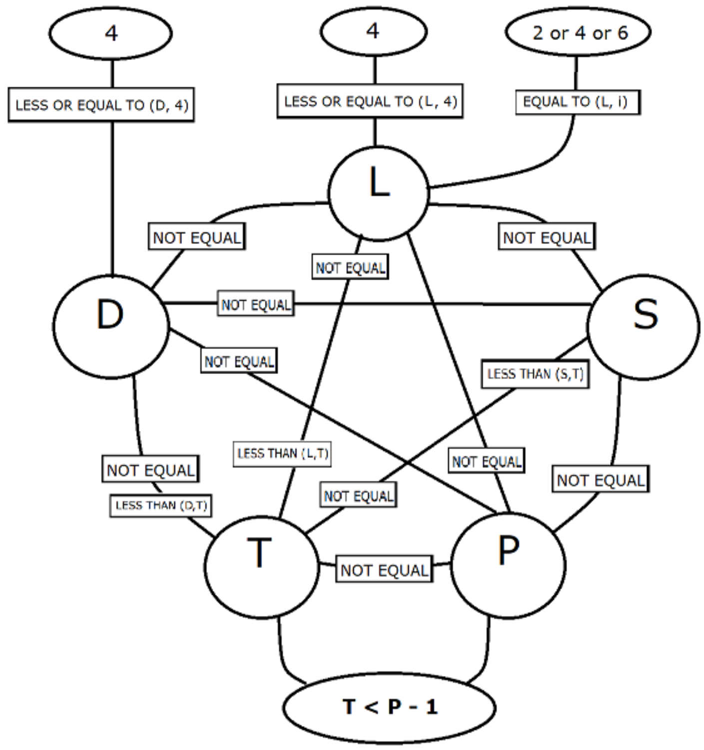
- Домашната задача и Лабораториската вежба имаат краен рок за 4 дена, следствено треба да бидат изработени навремено во првите 4 дена од седмицата:
    - (lambda a, b: a <=4 and b <= 4, ("D", "L"))
    - може да се одвои и во два унарни услови

- Повторувањето на теоретскиот дел треба заврши пред да започне повторување на програмскиот дел и препораката е двете повторувања да не бидат во два последователни дена:
    - (lambda a, b: a < b – 1, ("T", "P"))

- Студентот мора да ги заврши и предаде сите активности за проверка на знаењето пред да започне со повторувањето на теоретскиот дел од материјалот за колоквиум
    - (lambda a, b: a < b, ("D", "T"))
    - (lambda a, b: a < b, ("S", "T"))
    - (lambda a, b: a < b, ("L", "T"))

- Лабораториската вежба може да се предаде и оцени само во деновите вторник, четврток и сабота:
    - (lambda a: a == 2 or a == 4 or a == 6, ("L"))

б) Граф на ограничувања (услови) за проблемот:



в) Вредности кои преостануваат по примена на сите унарни услови:

- D – (1, 2, 3, 4)
- S – (1, 2, 3, 4, 5, 6, 7)
- L – (2, 4)
- T – (1, 2, 3, 4, 5, 6, 7)
- P – (1, 2, 3, 4, 5, 6, 7)

Во овој проблем, ограничувања од унарните услови имаат само лабараториската вежба и домашната задача.

д) Едно решение за проблемот од примерот:
Можни вредности за променливите по примена на унарните услови:

	1	2	3	4	5	6	7
D	✓	✓	✓	✓			
S	✓	✓	✓	✓	✓	✓	✓
L		✓		✓			
T	✓	✓	✓	✓	✓	✓	✓
P	✓	✓	✓	✓	✓	✓	✓

Со оглед на тоа дека трите активности за проверка на знаењето треба да се завршат пред повторувањето на материјалот од теорија (од ова произлегува T >= 4, потребни се барем три дена за завршување на проверка на знаењето) и повторувањето на практичниот дел треба да е барем два дена по завршувањето на материјалот за теорија, по употреба на forward checking и отстранување на вредностите кои ги прекршуваат условите, можни вредности на променливите се:

	1	2	3	4	5	6	7
D	✓	✓	✓	✓			
S	✓	✓	✓	✓			
L		✓		✓			
T				✓	✓		
P						✓	✓

1. Користејќи ја евристиката за најмал број на преостанати вредности (MRV), можни променливи се L, T и P
   - Бидејќи имаат ист број на преостанати вредности, со Degree heuristics за tie break, ја избираме вредноста која што е вклучена во најголем број на услови – L
   - За избирање на вредност за доделување може да се употреби евристиката за најмалку ограничувачка вредност (LCV), во случајов 2
   - Со forward checking ги отстрануваме вредностите кои би предизвикале прекршување на условите (D != 2 и S != 2, All Different constraint)

по чекор 1:

	1	2	3	4	5	6	7
D	✓		✓	✓			
S	✓		✓	✓			
L		2					
T				✓	✓		
P						✓	✓

2. Повторно со MRV ја избираме променливата со најмал број на преостанати вредности, сега можни избори се T и P
   - Бидејќи имаат ист број на преостанати вредности, со Degree heuristics за tie break, ја избираме вредноста која што е вклучена во најголем број на услови – T
   - Според евристиката LCV, ја ибираме вредноста 5, бидејќи ограничува најмал број на останати вредности
   - Со forward checking ги отстрануваме вредностите кои би предизвикале прекршување на условите (P != 6, да не е последователно со T)

по чекор 2:

	1	2	3	4	5	6	7
D	✓		✓	✓			
S	✓		✓	✓			
L		2					
T					5		
P							✓

3. Користејќи го MRV, ја избираме променливата P за која што имаме само една дозволена вредност – 7
    - Во овој чекор forward checking нема да отстрани ниедна од останатите вредности

по чекор 3:

	1	2	3	4	5	6	7
D	✓		✓	✓			
S	✓		✓	✓			
L		2					
T					5		
P							7

4. Во овој случај MRV ќе даде иста вредност и за домашната задача и за тестот за самостојна проверка
    - Со употреба на Degree heuristics за tie break, ја избираме променливата вклучена во најголем број на услови – D
    - LCV евристиката дава ист резултат за сите дозволени вредности па можеме да избереме по растечки редослед – 1
    - Со fast forward ги отстрануваме недозволените вредности

по чекор 4:

	1	2	3	4	5	6	7
D	1						
S			✓	✓			
L		2					
T					5		
P							7

5. Единствена преостаната вредност ни е тестот за самостојна проверка
    - LCV евристиката повторно дава ист резултат за сите дозволени вредности па бирајќи по растечки редослед ја земаме вредноста 3

по чекор 5:

	1	2	3	4	5	6	7
D	1						
S			3				
L		2					
T					5		
P							7

Едно решение на проблемот за подготовка на колоквиум