PS6

Graded

Student

Chetan Hiremath

Total Points

110 / 120 pts

Question 1

(no title) **10** / 10 pts

✔ **+ 10 pts** Correct

Question 2

(no title) **10** / 10 pts

✔ **+ 10 pts** correct

Question 3

(no title) **10** / 10 pts

✔ **+ 10 pts** Correct

Question 4

(no title) **15** / 15 pts

✔ **+ 15 pts** Correct

Question 5

(no title) Resolved **18** / 20 pts

✔ **+ 20 pts** Correct

✔ **– 5 pts** Questionable DPLL implementation

+ 3 pts Point adjustment

Regrade Request **Submitted on: Mar 04**

What do you mean that the DPLL implementation is questionable? My answer is correct since I have explained it and used backtracking for the truth value assignments that must be completely true. Will you regrade Question 5?

your proof is not clear/ readable.

Reviewed on: Mar 05

Question 6

(no title) Resolved **32** / 35 pts

✔ **+ 35 pts** Correct

✔ **− 3 pts** Incorrect game value (should be 0)/missing

Regrade Request **Submitted on: Mar 04**

How the full game value of X is 0 even though it should be 1? Will you regrade Question 6?

You may want to discuss this with Prof. Branicky.

Reviewed on: Mar 05

Question 7

EXTRA CREDIT Resolved **15** / 20 pts

✔ **+ 20 pts** Correct

✔ **− 5 pts** Mostly

Regrade Request **Submitted on: Mar 04**

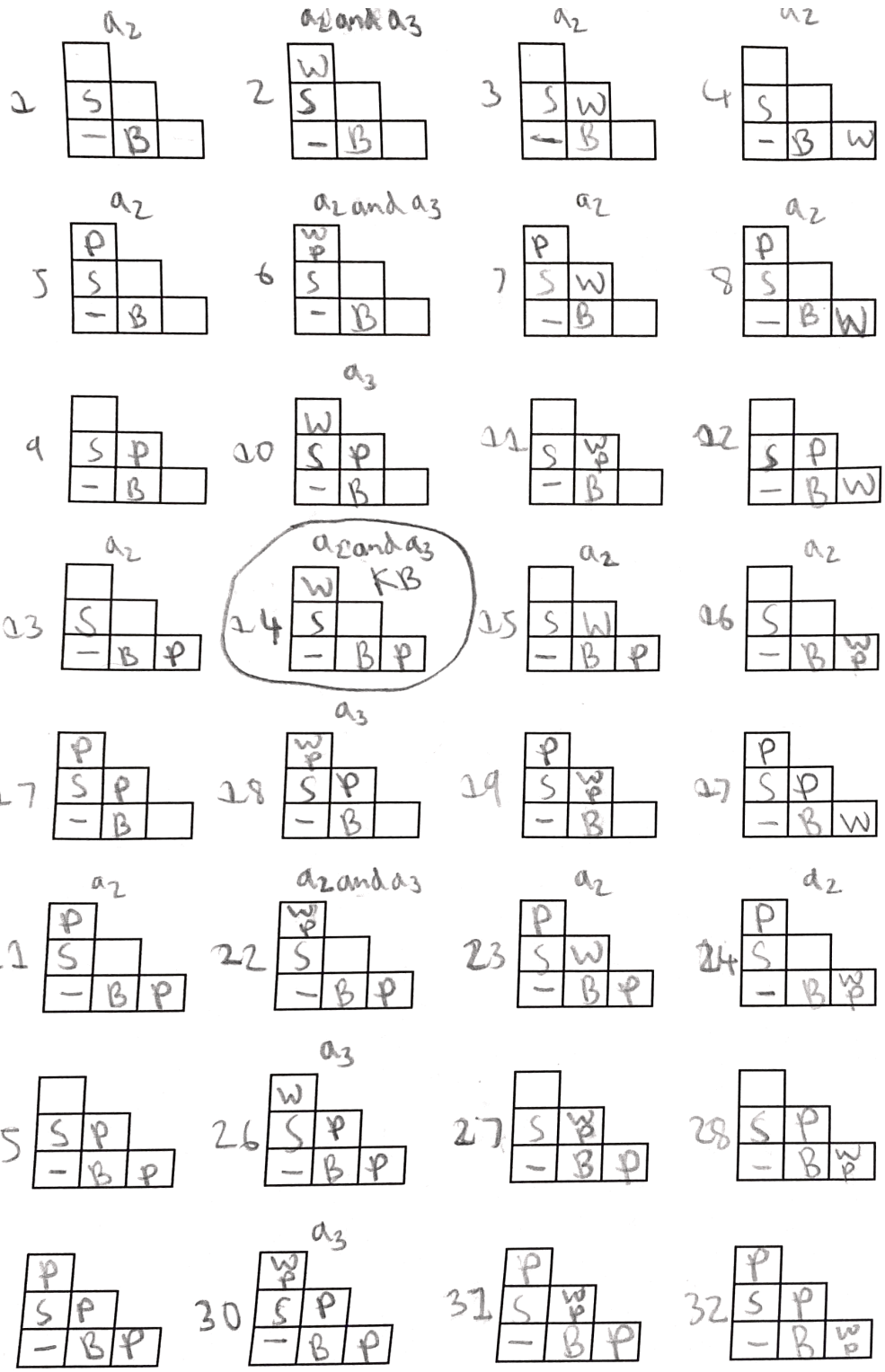
How I have lost 5 points when I have used the cache-ing function and shown the final results? Will you regrade the extra credit problem?

you're missing the results for boards 7 to 10

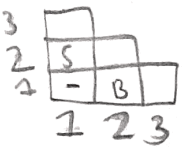
Reviewed on: Mar 05

Question assigned to the following page: 1

1.



S(Stench) = [1,2]
B(Breeze) = [2,1]
-(Nothing) = [1,1]



P - Pit
W - wumpus

a_2 = "There is no pit in [2,2]."

a_3 = "There is a wumpus in [1,3]."

KB = KB⊢a_2 and KB⊢a_3.

The worlds that have no pit in [2,2] are 1, 2, 3, 4, 5, 6, 7, 8, 13, 14, 15, 16, 21, 23, 23, and 24.

The worlds that have wumpus in [1,3] are 2, 6, 10, 14, 18, 22, 26, and 30.

The world that has KB is 14. when KB⊢a_2 and KB⊢a_3.

Therefore, KB⊢a_2 and KB⊢a_3.

Question assigned to the following page: 2

2a. $B \vee C$

A	B	C	D	B ∨ C
F	F	F	F	F
F	F	F	T	F
F	F	T	F	T
F	F	T	T	T
F	T	F	F	T
F	T	F	T	T
F	T	T	F	T
F	T	T	T	T
T	F	F	F	T
T	F	F	T	T
T	F	T	F	T
T	F	T	T	T
T	T	F	F	T
T	T	F	T	T
T	T	T	F	T
T	T	T	T	T

$2^{\text{variables}} = 2^4 = 16$ possible models.

So, there are 12 models in this sentence since 12 models are true.

b. $\neg A \vee \neg B \vee \neg(C \vee \neg D)$

A	B	C	D	¬A ∨ ¬B ∨ ¬(C ∨ ¬D)
F	F	F	F	T
F	F	F	T	T
F	F	T	F	T
F	F	T	T	T
F	T	F	F	T
F	T	F	T	T
F	T	T	F	T
F	T	T	T	T
T	F	F	F	T
T	F	F	T	T
T	F	T	F	T
T	F	T	T	T
T	T	F	F	T
T	T	F	T	T
T	T	T	F	T
T	T	T	T	F

$2^{\text{variables}} = 2^4 = 16$ possible models.

So, there are 15 models in this sentence since 15 models are true.

Question assigned to the following page: 2

c. $(A \Rightarrow B) \wedge A \wedge \neg B \wedge (C \wedge D)$

A	B	C	D	$(A \Rightarrow B) \wedge A \wedge \neg B \wedge (C \wedge D)$
F	F	F	F	F
F	F	F	T	F
F	F	T	F	F
F	F	T	T	F
F	T	F	F	F
F	T	F	T	F
F	T	T	F	F
F	T	T	T	F
T	F	F	F	F
T	F	F	T	F
T	F	T	F	F
T	F	T	T	F
T	T	F	F	F
T	T	F	T	F
T	T	T	F	F
T	T	T	T	F

$2^{\text{variables}} = 2^4 = 16$ possible models.

So, there are 0 models in this sentence since 0 models are true.

Question assigned to the following page: 3

3a. $(a \to B) \equiv (\neg B \to \neg a)$

a	B	¬B	¬a	a → B	¬B → ¬a
F	F	T	T	T	T
F	T	F	T	T	T
T	F	T	F	F	F
T	T	F	F	T	T

So, $(a \to B) \equiv (\neg B \to \neg a)$ is verified since their results are same.

b. $\neg(a \wedge B) \equiv (\neg a \vee \neg B)$

a	B	¬a	¬B	a ∧ B	¬(a ∧ B)	¬a ∨ ¬B
F	T	T	F	F	T	T
F	T	T	F	F	T	T
T	F	F	T	F	T	T
T	T	F	F	T	F	F

So, $\neg(a \wedge B) \equiv (\neg a \vee \neg B)$ is verified since their results are same.

c. $\neg(a \vee B) \equiv (\neg a \wedge \neg B)$

a	B	¬a	¬B	a ∨ B	¬(a ∨ B)	¬a ∧ ¬B
F	F	T	T	F	T	T
F	T	T	F	T	F	F
T	F	F	T	T	F	F
T	T	F	F	T	F	F

So, $\neg(a \vee B) \equiv (\neg a \wedge \neg B)$ is verified since their results are same.

Question assigned to the following page: 4

4a. Smoke → Smoke

Smoke	Smoke → Smoke
F	T
T	T

So, it's valid since the results are true only.

b. Smoke → Fire

Smoke	Fire	Smoke → Fair
F	F	T
F	T	T
T	F	F
T	T	T

So, it's neither valid nor unsatisfiable since both results are true and false.

Question assigned to the following page: 4

c. (Smoke → Fire) → (¬Smoke → ¬Fire)

Smoke	Fair	¬Smoke	¬Fire	Smoke → Fire	¬Smoke → ¬Fair	(Smoke → Fire) → (¬Smoke → ¬Fire)
F	T	T	T	T	T	T
F	F	T	F	T	F	F
T	T	F	T	F	T	T
T	F	F	F	T	T	T

So, it's neither valid or unsatisfiable since both results are true and false.

d. Smoke ∨ Fire ∨ ¬Fire

Smoke	Fire	¬Fire	Smoke ∨ Fire ∨ ¬Fire
F	F	T	T
F	T	F	T
T	F	T	T
T	T	F	T

So, it's valid since the results are true only.

Question assigned to the following page: 4

e. $((Smoke \wedge Heat) \rightarrow Fire) \leftrightarrow ((Smoke \rightarrow Fire) \vee (Heat \rightarrow Fire))$

Smoke	Heat	Fire	Smoke ∧ Heat	Smoke → Fire	Heat → Fire	(Smoke ∧ Heat) → Fire
F	F	F	F	T	T	T
F	F	T	F	T	T	T
F	T	F	F	T	F	T
F	T	T	F	T	T	T
T	F	F	F	F	T	T
T	F	T	F	T	T	T
T	T	F	T	F	T	F
T	T	T	T	T	T	T

(Smoke → Fire) ∨ (Heat → Fire)	((Smoke ∧ Heat) → Fire) ↔ ((Smoke → Fire) ∨ (Heat → Fire))
T	T
T	T
T	T
T	T
T	T
T	T
F	T
T	T

So, it's valid since the results are true only.

Question assigned to the following page: 4

f. (Smoke → Fire) → ((Smoke ∧ Heat) → Fire)

Smoke	Heat	Fire	Smoke ∧ Heat	(Smoke ∧ Heat) → Fire
F	F	F	F	T
F	F	T	F	T
F	T	F	F	T
F	T	T	F	T
T	F	F	F	T
T	F	T	F	T
T	T	F	T	F
T	T	T	T	T

Smoke → Fire	(Smoke → Fire) → ((Smoke ∧ Heat) → Fire)
T	T
T	T
T	T
F	T
T	T
T	T

So, it's valid since the results are true only.

Question assigned to the following page: 5

5. $P \rightarrow Q$

$L \wedge M \rightarrow P$

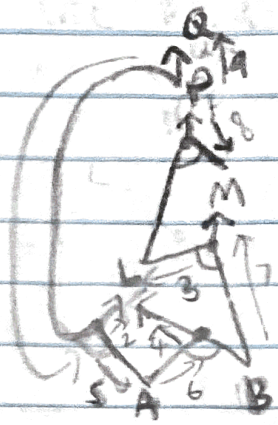
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



$\neg P \vee Q$ A=T

$\neg L \vee \neg M \vee P$ B=T

$\neg B \vee \neg L \vee M$ L=T

$\neg A \vee \neg P \vee L$ M=T

$\neg A \vee \neg B \vee L$ P=T

A Q=T

B

I have used DPLL to backtrack and continue the process of exploring possible truth value assignments until they are true. The DPLL behavior and forward-chaining behavior are analyzed, and DPLL is more efficient than forward chaining since DPLL proves assignments in propositional logic and checks if they are true. The satisfiability works if the assignments are true, so DPLL is efficient at backtracking.

Question assigned to the following page: 6

6a. I have used the program from TicTacToe.ipynb to test the examples from the lecture notes of Slide 14 of Adversarial Search 1, and I am getting the same results since the program's results and the lecture notes' results are completely same.

b. I have borrowed, used, and modified TicTacToe.ipynb in a separate file to change the board's size from 3 to 4 since I am supposed to convert the 3X3 board to the 4X4 board. I have made many changes on various functions like is_terminal(), utility(), next_state(), and print_out() for the 4X4 board. Then, I have included the boards that are represented as strings in the test_boards variable to print these boards on the terminal. I have taken approximately 10 minutes to modify this program and convert the 3X3 board to the 4X4 board. Here are the test boards that are translated as strings for the test_boards variable, and this variable is very important in this modified program.

```
test_boards = ['XXXXOOO'+BLANK*9,
               'OOOOXXX'+BLANK*9,
               'XXX'+BLANK+'OXXOOXXO'+BLANK+'OOO',
               BLANK+'XX'+BLANK+'OXXOOXXO'+BLANK+'OO'+BLANK,
               BLANK*4+'OXXOOXXO'+BLANK*4,
               'XOOO'+BLANK+'X'+BLANK*4+'X'+BLANK*5]
```

Question assigned to the following page: 6

c. I have compiled the modified program by using the minimax algorithm and its Python code, and terminal shows me that the boards and their respective results are successfully printed on the terminal. The first 6 boards display their results quickly, but the last 4 boards take too much time to display their results due to the time/space complexity's order for the 10 boards chronologically. I have managed to solve the first 6 boards only since their results are quickly displayed, so here are their final results in a table. (I haven't provided the final results of the last 4 boards since it takes several hours for these boards that are not solvable in the reasonable time to display their final results.)

Boards	Game Values	Optimal Move	Examined Nodes	Examined Terminals	Unique States	Time (s)
1	1	[]	0	1	0	8.106231689453125e-05
2	-1	[]	0	1	0	5.340576171875e-05
3	-1	[3, 12]	2	2	2	9.107589721679688e-05
4	0	[12, 15]	52	24	31	0.0006368160247802734
5	-1	[0, 1, 2, 3, 12, 13, 14, 15]	48232	23976	1460	0.49524855613708496
6	1	[15]	4958299	2156185	12500	45.815176486968994

Question assigned to the following page: 6

d. I have compiled the modified program by using the alpha-beta algorithm and including the alpha-beta algorithm's Python code in the modified program, and the terminal shows me that the boards and their respective results are successfully printed on the terminal. The first 6 boards display their results quickly due to the time/space complexity's order for the 10 boards chronologically. I have managed to solve the first 6 boards only since their results are quickly displayed, so here are their final results in a table. (There is no column of the optimal move in this table since the optimal move is used for the minimax algorithm only.)

Boards	Game Values	Examined Nodes	Examined Terminals	Unique States	Time (s)
1	1	0	1	0	7.2479248046875e-05
2	-1	0	1	0	4.38690185546875e-05
3	-1	2	2	2	7.677078247070312e-05
4	0	37	16	27	0.00042128562927246094
5	-1	6116	2374	825	0.05664229393005371
6	1	55504	20093	3336	0.5154292583465576

Question assigned to the following page: 6

e. I have compared the results of the minimax algorithm and the results of the alpha-beta algorithm after I have compiled the modified program. The results show me that the alpha-beta algorithm is more efficient than the minimax algorithm since the alpha-beta algorithm solves the boards by few nodes and short time. The alpha-beta algorithm reduces the number of nodes and the amount of time for computation and search. The examined nodes' count of the alpha-beta algorithm is less than the examined nodes' count of the minimax algorithm since the alpha-beta algorithm reduces the nodes' count, and the boards are solved in few seconds due to the alpha-beta algorithm. I have weakly solved 4X4 Tic-Tac-Toe, and the value of the full game of the first player ("X") is 1.

Question assigned to the following page: 7

7. I have compiled the modified program by including the cache-ing function or lru_cache(maxsize=None) from the functools library that is explained in CacheDemo.ipynb in this program and using the alpha-beta algorithm since this algorithm is more efficient than the minimax algorithm, and the terminal shows me that the boards and their respective results are successfully printed on the terminal. The first 6 boards display their results quickly due to the time/space complexity's order for the 10 boards chronologically. I have managed to solve the first 6 boards only since their results are quickly displayed, so here are their final results in a table. (There is no column of the optimal move in this table since the optimal move is used for the minimax algorithm only.)

Boards	Game Values	Examined Nodes	Examined Terminals	Unique States	Time (s)
1	1	0	1	0	6.985664367675781e-05
2	-1	0	1	0	4.249162292480469e-05
3	-1	2	2	2	7.034706115722656e-05
4	0	37	16	27	0.0004105567932128906
5	-1	6116	2374	825	0.05028914451599121
6	1	55504	20093	3336	0.5133132934570312

The results prove that the cache-ing has increased the speed of computation and enabled the modified program to display the results of the first 6 boards quickly. Because cache-ing has decreased the time that is taken to evaluate the results of these boards if I compare this table and the previous table. The lru_cache() function reduces the time of execution, so the boards have computed their results quickly.