

## Program: 3-SAT Solver using DPLL algorithm

```
import sys

def get_input(filename):
    # Input follows the DIMACS format:
    # https://www.cs.utexas.edu/users/moore/acl2/manuals/current/manual/index-
    # seo.php/SATLINK___DIMACS
    f = open(filename, 'r')
    clauses = []
    for line in f:
        if line[0] in ['c','0','%']:
            continue
        elif line[0] == 'p':
            words = line.split()
            num_clauses = int(words[-1])
            num_variables = int(words[-2])
            symbols = list(range(1, num_variables+1))
        else:
            clause = [int(n) for n in line.split()]
            if(len(clause) != 4):
                continue
            if clause[-1] != 0:
                print('Error: Terminal number of one or more clauses is not
0!')
                return (None, None)
            clause = clause[:-1]
            if any(abs(n) > num_variables or abs(n) < 1 for n in clause):
                print('Error: Total number of variables exceeds limit!')
                return (None, None)
            clauses.append(clause)
    f.close()
    if len(clauses) != num_clauses:
```

```

        print('Error: Total number of clauses not equal to specification!')
        return (None, None)
    return (clauses, symbols)

def dpll_satisfiable(filename):
    clauses, symbols = get_input(filename)
    if clauses != None:
        return dpll(clauses, symbols, {})

def dpll(clauses, symbols, model):

    # if every clause in clauses is True in model then return True
    if check_all_clauses_true(clauses, model):
        print(f'\nModel = {model}\n')
        return True

    # if some clause in clauses is False in model then return False
    if check_some_clauses_false(clauses, model):
        return False

    # check for pure symbol
    P, value = find_pure_symbol(clauses, symbols, model)
    if P != None:
        new_symbols = symbols[:]
        new_symbols.remove(P)
        model[P] = value
        return dpll(clauses, new_symbols, model)

    # check for unit clause
    P, value = find_unit_clause(clauses, symbols, model)
    if P != None:

```

```

        new_symbols = symbols[:]
        new_symbols.remove(P)
        model[P] = value
        return dpll(clauses, new_symbols, model)

# branch
P, rest = symbols[0], symbols[1:]
left_model, right_model = model.copy(), model.copy()
left_model[P], right_model[P] = True, False
return dpll(clauses, rest, left_model) or dpll(clauses, rest,
right_model)

def check_all_clauses_true(clauses, model):
    for clause in clauses:
        if not any(abs(literal) in model.keys() and evaluate_literal(literal,
model) == True for literal in clause):
            return False
    return True

def check_some_clauses_false(clauses, model):
    for clause in clauses:
        if all(abs(literal) in model.keys() and evaluate_literal(literal,
model) == False for literal in clause):
            return True
    return False

def evaluate_literal(literal, model):
    if literal < 0:
        return not model[abs(literal)]
    else:
        return model[literal]

```

```

def find_pure_symbol(clauses, symbols, model):
    pure_symbols = symbols[:]
    visited_literals = []
    for clause in clauses:
        if any(abs(literal) in model.keys() and evaluate_literal(literal,
model) == True for literal in clause):
            continue
        for literal in clause:
            if abs(literal) in pure_symbols:
                if -literal in visited_literals:
                    pure_symbols.remove(abs(literal))
                    visited_literals.remove(-literal)
                elif literal not in visited_literals:
                    visited_literals.append(literal)
    if len(pure_symbols) == 0:
        return (None, None)
    P = pure_symbols[0]
    value = P in visited_literals
    return (P, value)

def find_unit_clause(clauses, symbols, model):
    for clause in clauses:
        if any(abs(literal) in model.keys() and evaluate_literal(literal,
model) == True for literal in clause):
            continue
        vars_in_clause = [literal for literal in clause if abs(literal) in
symbols]
        if len(vars_in_clause) == 1:
            return (abs(vars_in_clause[0]), vars_in_clause[0] in symbols)
    return (None, None)

```

```
if len(sys.argv) != 2:
    print('Error: Incorrect number of command line arguments!')
else:
    filename = sys.argv[1]
    if dpll_satisfiable(filename):
        print('Satisfiable!\n')
    else:
        print('Not satisfiable!\n')
```

### Outputs

#### Input file 1

c This is a comment

c the following line specifies number of variables and number of clauses respectively

p 4 4

c the following lines are the clauses

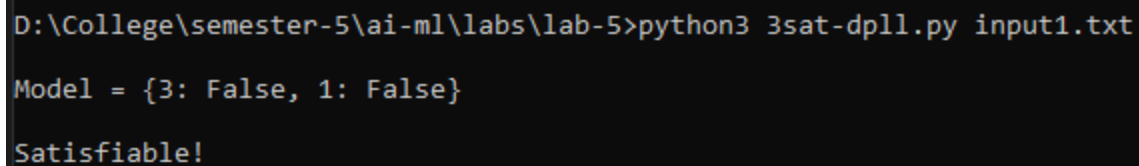
1 2 -3 0

-1 -2 4 0

-1 2 -4 0

2 -3 4 0

#### Output screenshot



```
D:\College\semester-5\ai-ml\labs\lab-5>python3 3sat-dpll.py input1.txt
Model = {3: False, 1: False}
Satisfiable!
```

#### Input file 2

c This is a comment

c the following line specifies number of variables and number of clauses respectively

p 3 8

c the following lines are the clauses

1 2 3 0  
1 2 -3 0  
1 -2 3 0  
1 -2 -3 0  
-1 2 3 0  
-1 2 -3 0  
-1 -2 3 0  
-1 -2 -3 0

Output screenshot

```
D:\College\semester-5\ai-ml\labs\lab-5>python3 3sat-dpll.py input2.txt  
Not satisfiable!
```

Input file 3

c FILE: aim-50-2\_0-yes1-1.cnf

c

c SOURCE: Kazuo Iwama, Eiji Miyano (miyano@cscu.kyushu-u.ac.jp),

c and Yuichi Asahiro

c

c DESCRIPTION: Artifical instances from generator by source. Generators

c and more information in sat/contributed/iwama.

c

c NOTE: Satisfiable

c

p cnf 50 100

-9 17 50 0

17 20 -50 0

17 -20 -50 0

-9 -17 39 0

-9 -17 -39 0  
9 29 43 0  
9 -29 43 0  
9 10 -43 0  
-10 -27 -43 0  
4 -10 -43 0  
-4 -6 -10 0  
-4 11 -16 0  
6 -11 -16 0  
-4 6 26 0  
11 -26 39 0  
6 -11 39 0  
-26 32 38 0  
32 -38 -39 0  
12 -26 -32 0  
-12 25 -39 0  
-13 -25 -32 0  
7 -12 -25 0  
-7 28 49 0  
-7 -25 49 0  
-7 33 -49 0  
8 -33 -49 0  
1 -8 -49 0  
-1 -8 21 0  
-1 5 36 0  
-5 -8 36 0  
-1 -14 -36 0  
-21 -36 -50 0  
14 24 -36 0  
14 -24 -38 0

-23 34 50 0  
-23 -24 -34 0  
23 -24 -34 0  
23 34 -42 0  
28 34 42 0  
-11 -28 42 0  
15 -28 42 0  
23 35 45 0  
-23 -28 45 0  
-15 -35 45 0  
-15 -17 -45 0  
12 -15 30 0  
-12 30 -45 0  
22 -30 -45 0  
-22 -30 -37 0  
-3 -22 -30 0  
3 -22 -47 0  
37 40 44 0  
-31 40 44 0  
4 13 37 0  
13 37 -40 0  
-13 33 -40 0  
-13 -33 44 0  
2 3 -44 0  
-2 -40 -44 0  
27 43 47 0  
-2 16 41 0  
-16 27 47 0  
-27 41 47 0  
41 -44 -47 0



-18 38 -41 0  
-2 -18 -38 0  
40 -41 46 0  
-20 33 -46 0  
-20 -33 -41 0  
18 19 28 0  
14 18 19 0  
-14 18 19 0  
-5 -19 -46 0  
-5 -18 -46 0  
20 21 -35 0  
-19 20 -35 0  
3 35 -48 0  
-3 -19 -48 0  
29 35 48 0  
-29 31 38 0  
27 31 48 0  
-29 31 48 0  
4 12 16 0  
25 26 -42 0  
-6 13 -37 0  
11 25 -37 0  
8 16 -47 0  
1 15 -31 0  
1 10 -21 0  
-14 22 -42 0  
32 -32 36 0  
2 10 -21 0  
-3 5 8 0  
15 21 22 0

5 7 29 0  
26 -27 50 0  
30 -31 -48 0  
7 -34 46 0  
-6 24 49 0  
2 24 46 0

### Output screenshot

```
• $ python3 3sat-dpll.py ./aim/aim-50-2_0-yes1-1.cnf  
  
Model = {1: True, 2: True, 3: False, 4: True, 5: False, 36: True, 14: False, 24: True, 38: False, 6: False, 26: True,  
32: True, 12: True, 7: True, 8: True, 21: True, 50: False, 9: False, 17: False, 10: True, 11: False, 16: False, 39: Tr  
ue, 25: True, 13: False, 49: True, 33: True, 46: True, 41: True, 18: False, 20: False, 19: True, 35: False, 15: True,  
48: False, 29: True, 43: True, 47: False, 27: False, 31: True, 22: True, 23: False, 28: True, 42: False, 34: False, 45  
: True, 30: True, 37: False, 40: False, 44: True}  
  
Satisfiable!
```

### Input file 4

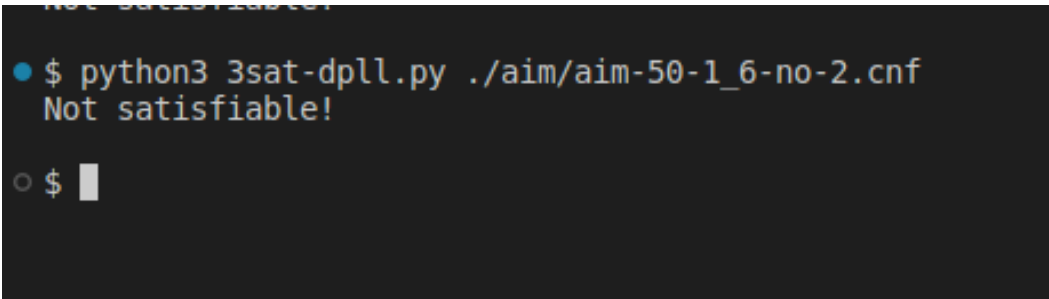
```
c FILE: aim-50-1_6-no-2.cnf  
  
c  
  
c SOURCE: Kazuo Iwama, Eiji Miyano (miyano@cscu.kyushu-u.ac.jp),  
c         and Yuichi Asahiro  
  
c  
  
c DESCRIPTION: Artifical instances from generator by source.  Generators  
c               and more information in sat/contributed/iwama.  
  
c  
  
c NOTE: Not Satisfiable  
  
c  
  
p cnf 50 80  
  
5 17 37 0  
  
24 28 37 0  
  
24 -28 40 0  
  
4 -28 -40 0  
  
4 -24 29 0
```

13 -24 -29 0  
-13 -24 -29 0  
-4 10 -17 0  
-4 -10 -17 0  
26 33 -37 0  
5 -26 34 0  
33 -34 48 0  
33 -37 -48 0  
5 -33 -37 0  
2 -5 10 0  
2 -5 -10 0  
-2 15 47 0  
15 30 -47 0  
-2 -15 30 0  
20 -30 42 0  
-2 20 -30 0  
13 -20 29 0  
13 16 -20 0  
-13 -20 31 0  
-13 16 -31 0  
-16 23 38 0  
-16 19 -38 0  
-19 23 -38 0  
14 -23 34 0  
1 14 -34 0  
-1 9 14 0  
-1 -9 -23 0  
-14 21 -23 0  
-14 -16 -21 0  
25 -35 41 0

-25 41 50 0  
-35 49 -50 0  
-25 -49 -50 0  
-19 -48 -49 0  
3 -39 44 0  
1 3 -44 0  
9 35 44 0  
-9 -31 44 0  
22 25 -44 0  
-12 -43 46 0  
-12 -28 -46 0  
6 35 48 0  
11 18 -48 0  
22 38 -42 0  
22 -35 -42 0  
-3 11 41 0  
27 28 -43 0  
-15 -21 31 0  
-33 39 50 0  
-8 -22 -47 0  
-22 -40 -47 0  
39 44 -46 0  
-25 -26 47 0  
38 43 45 0  
-6 -14 -45 0  
-7 12 36 0  
8 -11 45 0  
27 -38 -50 0  
7 -11 -36 0  
-7 -41 42 0

7 21 23 0  
-18 32 46 0  
8 19 -36 0  
-32 -45 -50 0  
7 17 21 0  
6 18 43 0  
-6 24 -27 0  
40 -41 49 0  
-11 12 26 0  
-3 32 -36 0  
-6 36 -44 0  
-3 36 42 0  
-8 -11 -32 0  
-18 -27 -38 0  
-18 -27 -39 0

Output screenshot

A terminal window with a dark background. The first line shows a command prompt with a blue dot, followed by the command 'python3 3sat-dpll.py ./aim/aim-50-1\_6-no-2.cnf'. The output of the command is 'Not satisfiable!'. The second line shows a new command prompt with a grey circle, followed by a dollar sign and a cursor.

```
● $ python3 3sat-dpll.py ./aim/aim-50-1_6-no-2.cnf
Not satisfiable!
○ $ █
```