Problem 0 | Austin Ray | CS 2211 | HW 6

P.0.a

Given
- n light bulbs
- All light bulbs initially off
- m buttons
- Know subset of buttons that each button controls (T_j)
- Pressing button j toggles all lights in T_j set

Goals
- CSP w/ m vars + n constraints
- n-ary constraints allowed

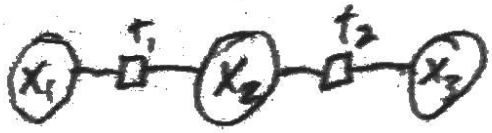
- m variables, 1 for each button
  - ↳ X_j where $j \in \{1, \ldots, m\}$
- Domain of each X_j is $\{0, 1, 2, \ldots\}$ where the number assigned is the number of times button j is pressed
- n constraints - 1 for each light bulb
  - ↳ $f_i(\vec{x})$ where $i \in \{1, \ldots, n\}$

$$f_i(\vec{x}) = \left[\sum_{j=1}^{m} \left[X_j \text{ if } \left(i \in T_j \text{ and } X_j \in \vec{x} \right) \text{ else } 0 \right] \right] = \text{odd \#}$$

Reasoning
- If each lightbulb toggled an odd # of times, its final state will be ON considering its initial state was OFF

Austin Ray CS 221 HW6

P.0.b



Given

- $X_1, X_2, X_3 \in \{0,1\}$
- $t_1(X) = x_1 \oplus x_2 = (x_1 \vee x_2) \wedge \neg(x_1 \wedge x_2)$
- $t_2(X) = x_2 \oplus x_3 = (x_2 \vee x_3) \wedge \neg(x_2 \wedge x_3)$

i

	x_1	x_2	x_3	t_1	t_2
	0	0	0	0	0
	0	0	1	0	1
*	0	1	0	1	1
	1	0	0	1	0
	0	1	1	1	1
*	1	0	1	1	1
	1	1	0	0	1
	1	1	1	0	0

There are [2] consistent assignments (starred above)

ii

PROBLEM 0 b (ii)

1. Root node
    a. initial call to **backtrack**() with x={} and w=1
2. Parent = 1
    a. x = {}
    b. w = 1
    c. X1 = {0, 1}
    d. X3 = {0, 1}
    e. X2 = {0, 1}
    f. delta can't be calculated for any value assignments to X1 -> recurse **backtrack**() with X1=0 first then recurse **backtrack**() with X1=1
3. Parent = 2
    a. x = {X1: 0}
    b. w = 1
    c. X1 = {0}
    d. X3 = {0, 1}
    e. X2 = {0, 1}
    f. delta can't be calculated for any value assignments to X3 -> recurse **backtrack**() with X3=0 first then recurse **backtrack**() with X3=1
4. Parent = 3
    a. x = {X1: 0, X3: 0}
    b. w = 1
    c. X1 = {0}
    d. X3 = {0}
    e. X2 = {0, 1}
    f. delta is 0 for X2=0. delta is 1 for X2=1. -> recurse **backtrack**() with X2=1 since it is the only value where delta != 0
5. Parent = 4
    a. x = {X1: 0, X3: 0, X2: 1}
    b. w = 1
    c. X1 = {0}
    d. X3 = {0}
    e. X2 = {1}
    f. Complete assignment for x found. Update best and return answer.
6. Parent = 3
    a. x = {X1: 0, X3: 1}
    b. w = 1
    c. X1 = {0}
    d. X3 = {1}
    e. X2 = {0, 1}
    f. delta is 0 for both assignments to X2. Do not update best. Return.
7. Parent = 2
    a. x = {X1: 1}

    b. w = 1
    c. X1 = {1}
    d. X3 = {0, 1}
    e. X2 = {0, 1}
    f. delta can't be calculated for any value assignments to X3 -> recurse **backtrack**() with X3=0 first then recurse **backtrack**() with X3=1
8. Parent = 7
    a. x = {X1: 1, X3: 0}
    b. w = 1
    c. X1 = {1}
    d. X3 = {0}
    e. X2 = {0, 1}
    f. delta is 0 for both X2=0 and X2=1. Do not update best. Return.
9. Parent = 7
    a. x = {X1: 1, X3: 1}
    b. w = 1
    c. X1 = {1}
    d. X3 = {1}
    e. X2 = {0, 1}
    f. delta is 0 for X2=1 and 1 for X2=0. -> recurse **backtrack**() with X2=0
10. Parent = 9
    a. x = {X1: 1, X3: 1, X2: 0}
    b. w = 1
    c. X1 = {1}
    d. X3 = {1}
    e. X2 = {0}
    f. Complete assignment for x found. Update best and return answer.

backtrack() is called a total of **9 times**.

Note: If backtrack() was designed so that it stopped once it found one consistent assignment to the CSP, backtrack() would only be called 5 times.

PROBLEM 0 b (iii)

1. Root node
    a. initial call to **backtrack**() with x={} and w=1
2. Parent = 1
    a. x = {}
    b. w = 1
    c. X1 = {0, 1}
    d. X3 = {0, 1}
    e. X2 = {0, 1}
    f. Try X1=0
        i. Delta = 1
        ii. After AC-3…
            1. Domains' = {X1: {0}, X2: {1}, X3: {0}}
        iii. Recurse **backtrack**() with X1=0
    g. Try X1=1
        i. Delta = 1
        ii. After AC-3…
            1. Domains' = {X1: {1}, X2: {0}, X3: {1}}
        iii. Recurse **backtrack**() with X1=1
3. Parent = 2
    a. x = {X1: 0}
    b. w = 1
    c. X1 = {0}
    d. X3 = {0}
    e. X2 = {1}
    f. Try X3=0
        i. Delta=1
        ii. After AC-3…
            1. Domains don't change
        iii. Recurse **backtrack**() with X3=0
4. Parent = 3
    a. x = {X1: 0, X3: 0}
    b. w = 1
    c. X1 = {0}
    d. X3 = {0}
    e. X2 = {1}
    f. Try X2=1
        i. Delta=1
        ii. After AC-3…
            1. Domains don't change
        iii. Recurse **backtrack**() with X2=1
5. Parent = 4
    a. x = {X1: 0, X3: 0, X2: 1}

b. w = 1
c. X1 = {0}
d. X3 = {0}
e. X2 = {1}
f. Complete assignment for x found. Update best and return answer.

6. Parent = 2
    a. x = {X1: 1}
    b. w = 1
    c. X1 = {1}
    d. X3 = {1}
    e. X2 = {0}
    f. Try X3=1
        i. Delta=1
        ii. After AC-3…
            1. Domains don't change
        iii. Recurse **backtrack**() with X3=1
7. Parent = 3
    a. x = {X1: 1, X3: 1}
    b. w = 1
    c. X1 = {1}
    d. X3 = {1}
    e. X2 = {0}
    f. Try X2=0
        i. Delta=1
        ii. After AC-3…
            1. Domains don't change
        iii. Recurse **backtrack**() with X2=0
8. Parent = 4
    a. x = {X1: 1, X3: 1, X2: 0}
    b. w = 1
    c. X1 = {1}
    d. X3 = {1}
    e. X2 = {0}
    f. Complete assignment for x found. Update best and return answer.

backtrack() is called a total of **7 times**.

Note: If backtrack() was designed so that it stopped once it found one consistent assignment to the CSP, backtrack() would only be called 4 times.

Austin Ray CS 221 HW 6

2a] old vars: X_1, X_2, X_3

New vars: B_1, B_2, B_3, B_4

Domains:

X_1: $\{0, 1, 2\}$

X_2: $\{0, 1, 2\}$

X_3: $\{0, 1, 2\}$

B_1: {all $[x, y]$ where $0 \leq x \leq K$ and $0 \leq y \leq K$}

B_2: same as B_1's

B_3: same as B_1's

B_4: $\{0, \ldots, K\}$

Constraints/Factors

$B_1[0] = 0$ $\quad B_4 \leq K$

$B_1[1] = B_1[0] + X_1$

$B_2[0] = B_1[1]$

$B_2[1] = B_2[0] + X_2$

$B_3[0] = B_2[1]$

$B_3[1] = B_3[0] + X_3$

$B_4 = B_3[1]$

{8 factors}

After constraints applied:

$B_1 = [0, X_1]$

$B_2 = [X_1, X_1 + X_2]$

$B_3 = [X_1 + X_2, X_1 + X_2 + X_3]$

$B_4 = X_1 + X_2 + X_3$

Problem 3c

I ran run_p3.py on 'profile.txt'

Final output:
Here's the best schedule:

Quarter	Units	Course
Win2016	4	CS228
Win2016	3	CS223A
Win2016	3	CS140
Spr2016	3	CS155
Spr2016	3	CS225A
Spr2016	4	CS161
Aut2016	4	CS145
Aut2016	3	CS144
Aut2016	3	CS229

Analysis:
Yup. That's exactly what I decided to take. Cool stuff! ☺

Program output:

Units: 0-10
Quarter: ['Win2016', 'Spr2016', 'Aut2016']
Taken: set(['CS221', 'CS103', 'STATS116', 'CS110', 'CS107', 'CS106B', 'CME104', 'CME106', 'CS109', 'CME100', 'CME102'])
Requests:
Request{['CS145'] [] [] 4.0}
Request{['CS228'] [] [] 4.0}
Request{['CS223A'] [] [] 4.0}
Request{['CS144'] [] [] 6.0}
Request{['CS140'] [] [] 2.0}
Request{['CS155'] [] ['CS140'] 6.0}
Request{['CS225A'] [] ['CS223A'] 3.0}
Request{['CS161'] ['Spr2016', 'Aut2016'] [] 10.0}
Request{['CS229'] ['Aut2016'] [] 6.0}
Request{['CS246'] [] ['CS145'] 10.0}
Found 24 optimal assignments with weight 829440.000000 in 568611 operations
First assignment took 112 operations
829440.0
('CS161', 'Spr2016') = 4
(Request{['CS246'] [] ['CS145'] 10.0}, 'Spr2016') = None
('CS145', 'Win2016') = 0

(Request{['CS228'] [] [] 4.0}, 'Aut2016') = None
('CS161', 'Win2016') = 0
(Request{['CS161'] ['Spr2016', 'Aut2016'] [] 10.0}, 'Spr2016') = CS161
('CS223A', 'Win2016') = 3
('sum', 'Win2016_units', 1) = (0, 4)
(Request{['CS145'] [] [] 4.0}, 'Win2016') = None
('CS161', 'Aut2016') = 0
('sum', 'Win2016_units', 3) = (7, 7)
('sum', 'Spr2016_units', 10) = 10
('sum', 'Aut2016_units', 4) = (7, 7)
('sum', 'Win2016_units', 5) = (10, 10)
('or', ((Request{['CS246'] [] ['CS145'] 10.0}, 'Spr2016'), 'CS145'), 'aggregated') = False
('CS246', 'Spr2016') = 0
('sum', 'Aut2016_units', 6) = (7, 7)
('sum', 'Win2016_units', 7) = (10, 10)
('or', ((Request{['CS155'] [] ['CS140'] 6.0}, 'Win2016'), 'CS140'), 'aggregated') = False
('sum', 'Aut2016_units', 0) = (0, 4)
('or', ((Request{['CS246'] [] ['CS145'] 10.0}, 'Aut2016'), 'CS145'), 1) = no
('sum', 'Win2016_units', 9) = (10, 10)
('CS223A', 'Spr2016') = 0
('CS140', 'Win2016') = 3
('sum', 'Aut2016_units', 2) = (4, 4)
('or', ((Request{['CS155'] [] ['CS140'] 6.0}, 'Aut2016'), 'CS140'), 'aggregated') = True
('CS155', 'Aut2016') = 0
('CS144', 'Spr2016') = 0
(Request{['CS155'] [] ['CS140'] 6.0}, 'Win2016') = None
('sum', 'Spr2016_units', 8) = (10, 10)
(Request{['CS246'] [] ['CS145'] 10.0}, 'Win2016') = None
('or', ((Request{['CS225A'] [] ['CS223A'] 3.0}, 'Win2016'), 'CS223A'), 'aggregated') = False
('or', ((Request{['CS155'] [] ['CS140'] 6.0}, 'Aut2016'), 'CS140'), 1) = prev
('or', ((Request{['CS246'] [] ['CS145'] 10.0}, 'Aut2016'), 'CS145'), 'aggregated') = False
('sum', 'Aut2016_units', 8) = (7, 10)
('CS140', 'Spr2016') = 0
(Request{['CS140'] [] [] 2.0}, 'Aut2016') = None
('sum', 'Spr2016_units', 4) = (0, 0)
(Request{['CS161'] ['Spr2016', 'Aut2016'] [] 10.0}, 'Aut2016') = None
('CS228', 'Aut2016') = 0
('sum', 'Spr2016_units', 6) = (3, 6)
('or', ((Request{['CS155'] [] ['CS140'] 6.0}, 'Spr2016'), 'CS140'), 0) = equals
('or', ((Request{['CS225A'] [] ['CS223A'] 3.0}, 'Spr2016'), 'CS223A'), 0) = equals
(Request{['CS144'] [] [] 6.0}, 'Win2016') = None
('sum', 'Spr2016_units', 0) = (0, 0)

('CS246', 'Aut2016') = 0
('or', ((Request{['CS225A'] [] ['CS223A'] 3.0}, 'Aut2016'), 'CS223A'), 1) = prev
('CS144', 'Aut2016') = 3
('sum', 'Spr2016_units', 2) = (0, 0)
(Request{['CS145'] [] [] 4.0}, 'Spr2016') = None
(Request{['CS225A'] [] ['CS223A'] 3.0}, 'Win2016') = None
('CS145', 'Spr2016') = 0
(Request{['CS140'] [] [] 2.0}, 'Win2016') = CS140
('CS223A', 'Aut2016') = 0
('or', ((Request{['CS246'] [] ['CS145'] 10.0}, 'Aut2016'), 'CS145'), 0) = no
(Request{['CS228'] [] [] 4.0}, 'Spr2016') = None
('sum', 'Aut2016_units', 10) = 10
('CS228', 'Spr2016') = 0
(Request{['CS225A'] [] ['CS223A'] 3.0}, 'Spr2016') = CS225A
(Request{['CS155'] [] ['CS140'] 6.0}, 'Spr2016') = CS155
('CS144', 'Win2016') = 0
(Request{['CS223A'] [] [] 4.0}, 'Aut2016') = None
('CS225A', 'Win2016') = 0
(Request{['CS144'] [] [] 6.0}, 'Spr2016') = None
(Request{['CS161'] ['Spr2016', 'Aut2016'] [] 10.0}, 'Win2016') = None
(Request{['CS140'] [] [] 2.0}, 'Spr2016') = None
('or', ((Request{['CS155'] [] ['CS140'] 6.0}, 'Spr2016'), 'CS140'), 'aggregated') = True
(Request{['CS223A'] [] [] 4.0}, 'Win2016') = CS223A
('sum', 'Win2016_units', 0) = (0, 0)
('sum', 'Aut2016_units', 5) = (7, 7)
('sum', 'Win2016_units', 2) = (4, 7)
('or', ((Request{['CS225A'] [] ['CS223A'] 3.0}, 'Spr2016'), 'CS223A'), 'aggregated') =
True
('sum', 'Aut2016_units', 7) = (7, 7)
('sum', 'Win2016_units', 4) = (7, 10)
(Request{['CS145'] [] [] 4.0}, 'Aut2016') = CS145
('sum', 'Aut2016_units', 1) = (4, 4)
('sum', 'Win2016_units', 6) = (10, 10)
('CS246', 'Win2016') = 0
(Request{['CS229'] ['Aut2016'] [] 6.0}, 'Spr2016') = None
('sum', 'Aut2016_units', 3) = (4, 7)
('CS229', 'Win2016') = 0
('sum', 'Win2016_units', 8) = (10, 10)
('CS155', 'Win2016') = 0
('sum', 'Win2016_units', 10) = 10
('CS140', 'Aut2016') = 0
(Request{['CS144'] [] [] 6.0}, 'Aut2016') = CS144
(Request{['CS223A'] [] [] 4.0}, 'Spr2016') = None
('CS228', 'Win2016') = 4
('or', ((Request{['CS155'] [] ['CS140'] 6.0}, 'Aut2016'), 'CS140'), 0) = equals
('sum', 'Spr2016_units', 9) = (10, 10)

('sum', 'Aut2016_units', 9) = (10, 10)
('CS155', 'Spr2016') = 3
('CS229', 'Spr2016') = 0
('or', ((Request{['CS246'] [] ['CS145'] 10.0}, 'Win2016'), 'CS145'), 'aggregated') = False
(Request{['CS229'] ['Aut2016'] [] 6.0}, 'Win2016') = None
('CS229', 'Aut2016') = 3
('sum', 'Spr2016_units', 5) = (0, 3)
(Request{['CS246'] [] ['CS145'] 10.0}, 'Aut2016') = None
(Request{['CS225A'] [] ['CS223A'] 3.0}, 'Aut2016') = None
('sum', 'Spr2016_units', 7) = (6, 10)
(Request{['CS155'] [] ['CS140'] 6.0}, 'Aut2016') = None
(Request{['CS228'] [] [] 4.0}, 'Win2016') = CS228
('sum', 'Spr2016_units', 1) = (0, 0)
('CS225A', 'Spr2016') = 3
('or', ((Request{['CS225A'] [] ['CS223A'] 3.0}, 'Aut2016'), 'CS223A'), 0) = equals
('CS225A', 'Aut2016') = 0
('CS145', 'Aut2016') = 4
('sum', 'Spr2016_units', 3) = (0, 0)
('or', ((Request{['CS225A'] [] ['CS223A'] 3.0}, 'Aut2016'), 'CS223A'), 'aggregated') = True
(Request{['CS229'] ['Aut2016'] [] 6.0}, 'Aut2016') = CS229
('or', ((Request{['CS246'] [] ['CS145'] 10.0}, 'Spr2016'), 'CS145'), 0) = no
Here's the best schedule:

Quarter	Units	Course
Win2016	4	CS228
Win2016	3	CS223A
Win2016	3	CS140
Spr2016	3	CS155
Spr2016	3	CS225A
Spr2016	4	CS161
Aut2016	4	CS145
Aut2016	3	CS144
Aut2016	3	CS229