- Variables: For each button j, $X_j = (x_1, x_2, \cdots, x_n)$, where

$$x_i = \begin{cases} 1, & x_i \in T_j \\ 0, & x_i \notin T_j \end{cases} \tag{1}$$

- Constrains: $X_1 \oplus X_2 \oplus \cdots X_j = (1, 1, \cdots, 1)$

- There are 2 consistent assignments:

X_1	X_2	X3
1	0	1
0	1	0

- backtrack() will be called 9 times to get all consistent assignments if we use the fixed ordering X_1, X_3, X_2 operations:

Backtrack($x = \{\}$,w=1, Domains={0, 1})
Backtrack($x = \{X_1 = 0\}$,w=1, Domains={0, 1})
Backtrack($x = \{X_1 = 0, X_3 = 0\}$,w=1, Domains={0, 1})
Backtrack($x = \{X_1 = 0, X_3 = 0, X_2 = 1\}$,w=1, Domains={0, 1})

Backtrack($x = \{X_1 = 0, X_3 = 1\}$,w=1, Domains={0, 1})

Backtrack($x = \{X_1 = 1\}$,w=1, Domains={0, 1})
Backtrack($x = \{X_1 = 1, X_3 = 0\}$,w=1, Domains={0, 1})

Backtrack($x = \{X_1 = 1, X_3 = 1\}$,w=1, Domains={0, 1})
Backtrack($x = \{X_1 = 1, X_3 = 1, X_2 = 0\}$,w=1, Domains={0, 1})

- backtrack() will be called 7 times to get all consistent assignments if we use the fixed ordering X_1, X_3, X_2 operations:

Backtrack($x = \{\}$,w=1, Domains={0, 1})
Backtrack($x = \{X_1 = 0\}$,w=1, Domains={0, 1})
Backtrack($x = \{X_1 = 0, X_3 = 0\}$,w=1, Domains={0, 1})
Backtrack($x = \{X_1 = 0, X_3 = 0, X_2 = 1\}$,w=1, Domains={0, 1})

Backtrack($x = \{X_1 = 1\}$,w=1, Domains={0, 1})
Backtrack($x = \{X_1 = 1, X_3 = 1\}$,w=1, Domains={0, 1})
Backtrack($x = \{X_1 = 1, X_3 = 1, X_2 = 0\}$,w=1, Domains={0, 1})

We'll introduce an auxiliary variable A_i for $i = 1, 2, \cdots, n$ which represents the sum of variables $X_1, X_2, \ldots X_n$. In this case, $n = 3$. Then we can write a simple recurrence that updates A_i with A_{i-1}. The constraint enforces that the sum of all the variables is less than or equal to K.

- Initialization: $[A_i = 0]$
- Processing: $[A_i = A_{i-1} + X_i]$
- Final output: $[A_4 \leq K]$

Then, we'll need turn the ternary constraint $[A_i = A_{i-1} + X_i]$ into a binary constraint by merging A_{i-1} and A_i into one variable, represented as one variable B_i. The variable B_i will represent a pair of sums, where $B_i[1]$ represents A_{i-1} and $B_i[2]$ represents A_i .

- Initialization: $[B_i = 0]$
- Processing: $[B_i[2] = B_i[1] + X_i]$
- Final output: $[B_4[2] \leq K]$
- Consistency: $[B_i[1] = B_{i-1}[2]]$

```
# Unit limit per quarter. You can ignore this for the first
# few questions in problem 2.
minUnits 3
maxUnits 3

# These are the quarters that I need to fill out.
# It is assumed that the quarters are sorted in chronological order.
register Aut2019
register Win2020
register Spr2020

# Courses I've already taken
taken CS107
taken CS109
taken CS145
taken CS140
taken CS221

# Courses that I'm requesting
request CS229 or CS144 in Aut2019
request CS240 in Win2020,Spr2020 # Can only be taken in Spr2020
request CS228
```

Here's the best schedule, which looks resonable to me:
Quarter Units Course
Aut2019 3 CS229
Win2020 3 CS228
Spr2020 3 CS240

In the worst case, treewidth will be n, when notable patterns have the length of n.