HW 6 SUNID: BENMA

0 a. Construct:

$X=(X_1, \ldots, X_n)$, where $X_i \in$ Domain$[0, 1]$ for switch off and on

$T_i(X \in [X_1 \cdots X_n]) =$ Sum(X_i) if $i \in T_i = (1, \ldots, n)$ for j in range$(1, m+1)$

Function:

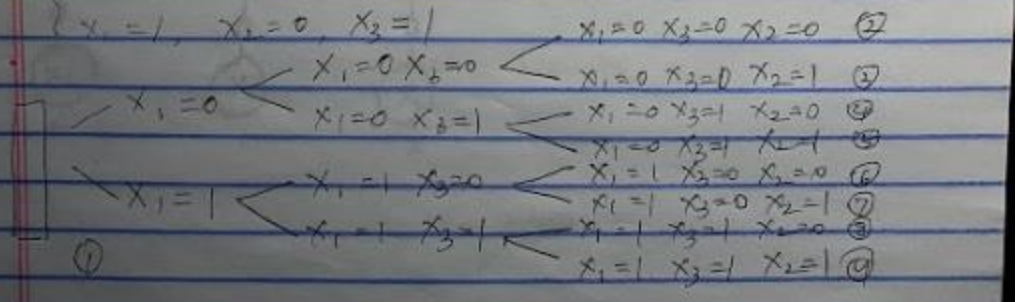
```
def fi(Xi):
    Total = 0
    for j in range(i, m+1):
        if i in Tj:
            Total = Xi
    if Total %2 == 1:
        return True
    else
        return False
```

0 b. i Found 2 assignments in 9 operations

$\{X_1 = 0, X_2 = 1, X_3 = 0$

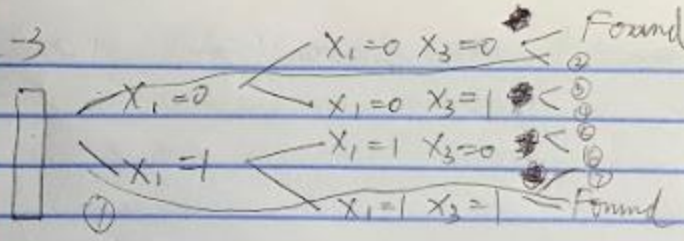
$\{X_1 = 1, X_2 = 0, X_3 = 1$

ii



So call backtrack() 9 times to get all consistent assignment

iii AC-3



AC-3 call backtrack() 7 times to get all consistent assignments

2a. Initialization: $A_i[A_i(1)=0]$

Processing: $[A_i[2]=A_i[1]+X_2]$ $i=1,2,3$. $X_1=0$, $X_2=1$, $X_3=2$

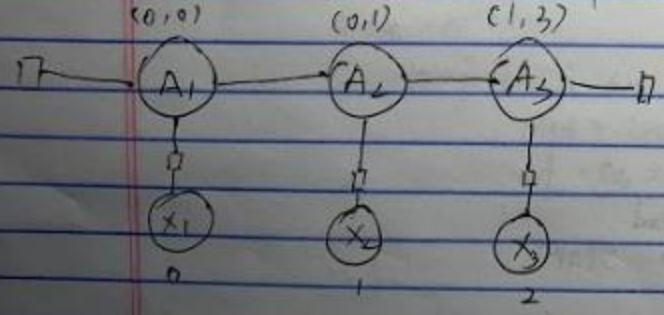
$[A_1[2]=A_2[1]+X_2]$

$[A_3[2]=A_3[1]+X_3]$

Final Output: $[A_3[2] \le K]$

Consistency: $[A_i[2]=A_{i+1}[1]]$

Aux Domain: $A_i=\{(x,y) \mid 0 \le x \le 3,\ 0 \le y \le 3^{x_i} \in Z\}$



3C

```
# Unit limit per quarter.
minUnits 3
maxUnits 15

# These are the quarters that I need to fill. It is assumed that
# the quarters are sorted in chronological order.
register Aut2017
register Win2018
register Aut2019
# Courses I've already taken
taken CS140
taken CS221
taken MATH51
taken CS145
taken CS124
taken CS106B
taken CS107
taken CS109

# Courses that I'm requesting
request CS103
request CS161
request CS109
request CS148
request CS246
request CS149
request CS231A

Here's the best schedule:
Quarter        Units   Course
  Aut2017       4       CS148
  Win2018       4       CS231A
  Aut2019       5       CS103
```

4a. Worst case: Only 1 notable pattern of length n eliminate any 1 variables, so we can have the tree width = $n-1$

4b. Use pattern match searching DP of prefix tree.

Set state = (index$_i$, Value$_i$, State)

(i, V_i, State) → transition to (i+1, K, state_New) K∈(1, K)

and get to the factor of how many patter matched.

Time and space is $O(n^2|P|K)$

Sample Algorithm: def PatternSearch(string, ~~V~~)

```
    transitions = {}
    output = {}
    fail = []
state = 0
~~for~~ for i, char in enumerate(Vi)
    res = transitions.get(state, char) FAIL
    if res == FAIL:
        break
    state = res
state = 0
results = []
for i, char in enumerate(string):
    ~~res~~ = transitions.get(state, char) and FAIL
    if res != FAIL
        state = res
        break
    state = fail[state]
for match in output.get(state, 0):
    pos = i - len(match) + 1
    results.append(pos, match)
```