CS 221 HW 6 - Scheduling

Name: Vinod Kumar Senthil Kumar
Sunet ID: vinodkum
Contributors: Xun Fang, Joe Fan

0a) The CSP will have 'm' variables representing 'm' buttons.
It will have 'n' potentials representing 'n' light bulbs.

The domain of each variable would be $\{0, 1\}$

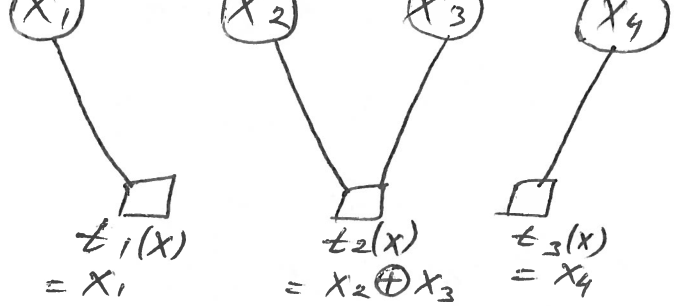
0 - button ~~if~~ is off
1 - button is on

Each variable would be connected to a subset of the potentials based on the lightbulbs the button represented by the variable control.

Eg: if $T_1 = \{1, 2, 4\}$
then the first variable will be connected to potentials 1, 2 & 4.

The potential function is the XOR of values all its connected variables.

Eg: $i = 3$, $j = 4$, $T_1 = \{1\}$, $T_2 = \{2\}$, $T_3 = \{2\}$, $T_4 = \{3\}$



$X_1, X_2, X_3, X_4 \in \{0, 1\}$

The intution is that the light bulbs are connected to the buttons that control the bulb and the bulb will be on ~~(i.e = 1)~~ ~~when~~ (i.e potential = 1) when an odd number of buttons are pressed on. The XOR function gives 1 when odd number of buttons are on & gives 0 ~~when even number of buttons are~~ otherwise

0b) i)

$X_1, X_2, X_3 \in \{0,1\}$

$t_1(x) = x_1 \oplus x_2, \quad t_2(x) = x_2 \oplus x_3$

X_1	X_2	X_3	t_1	t_2	~~bad~~ π weight	
0	0	0	0	0	0	
0	0	1	0	1	0	
0	1	0	1	1	1	✓
0	1	1	1	0	0	
1	0	0	1	0	0	
1	0	1	1	1	1	✓
1	1	0	0	1	0	
1	1	1	0	0	0	

There are 2 consistent assignments

ii) X_1, X_2, X_3 :-

$\{\}$
$X_1, X_2, X_3 \in \{0,1\}$

$\swarrow$

$\{X_1 = 0\}$
$X_2, X_3 \in \{0,1\}$
$\downarrow$
$\{X_1 = 0, X_2 = 1\}$
$X_3 \in \{0,1\}$
$\downarrow$
$\{X_1 = 0, X_2 = 1, X_3 = 0\}$

$\searrow$

$\{X_1 = 1\}$
$X_2, X_3 \in \{0,1\}$
$\downarrow$
$\{X_1 = 1, X_2 = 0\}$
$X_3 \in \{0,1\}$
$\downarrow$
$\{X_1 = 1, X_2 = 0, X_3 = 1\}$

backtrack() is called 7 times

0b iii) x_1, x_3, x_2 with AC-3 :-

$\{\ \}$

$x_1, x_3, x_2 \in \{0,1\}$

$\{x_1 = 0\}$

$x_2 \in \{1\}$

$x_3 \in \{0\}$

$\{x_1 = 0, x_3 = 0\}$

$x_2 = \{1\}$

$\{x_1 = 0, x_3 = 0, x_2 = 1\}$

$\{x_1 = 1\}$

$x_3 \in \{1\}$

$x_2 \in \{0\}$

$\{x_1 = 1, x_3 = 1\}$

$x_2 \in \{0\}$

$\{x_1 = 1, x_3 = 1, x_2 = 0\}$

backtrack is called 7 times.

x_1, x_3, x_2 :-

$\{\}$

$x_1, x_3, x_2 \in \{0, 1\}$

$\{x_1 = 0\}$

$x_3, x_2 \in \{0, 1\}$

$\{x_1 = 1\}$

$x_3, x_2 \in \{0, 1\}$

$\{x_1 = 0, x_3 = 0\}$

$x_2 \in \{0, 1\}$

$\{x_1 = 0, x_3 = 1\}$

$x_2 \in \{0, 1\}$

$\{x_1 = 1, x_3 = 0\}$

$x_2 \in \{0, 1\}$

$\{x_1 = 1, x_3 = 1\}$

$x_2 \in \{0, 1\}$

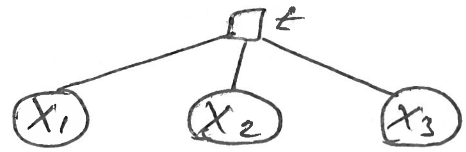
$\{x_1 = 0, x_3 = 0, x_2 = 1\}$

$\{x_1 = 1, x_3 = 1, x_2 = 0\}$

back track is called 9 times.

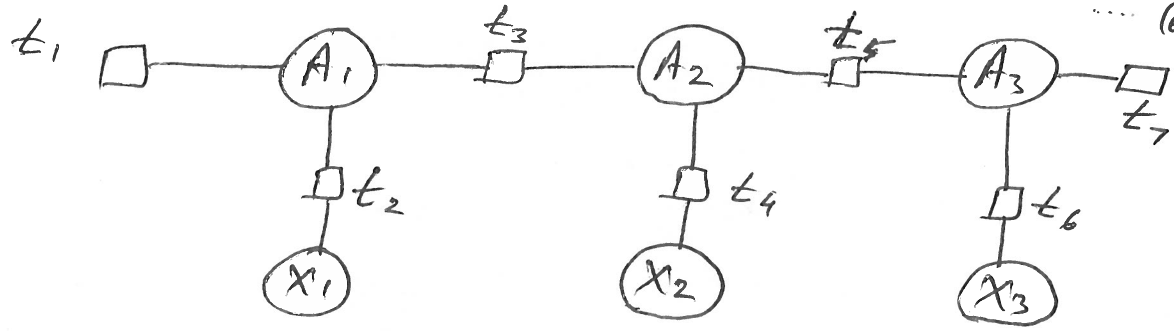
2a) $X_1, X_2, X_3 \in \{0,1,2\}$

potential t:- $X_1 + X_2 + X_3 \le 6$



This can be reduced to unary | binary potentials by introducing auxilary variables that store incremental sum of variables.

$X_1, X_2, X_3 \in \{0,1,2\}$, $A_1, A_2, A_3 \in \{(0,0), (0,1), \ldots (6,6)\}$



$t_1(a) \Rightarrow a[0] == 0$

$t_2(a,x) = t_4(a,x) = t_6(a,x) \Rightarrow a[1] == a[0] + x$

$t_3(a_1, a_2) = t_5(a_1, a_2) \Rightarrow a_1[1] == a_2[0]$

$t_7(a) \Rightarrow a[1] \le 6$

The auxilary variables' value is a tuple of two values from 0 to 6. The first value in the tuple is the sum of values of 'x' variables so far while the second value is the sum including the current

3d) The course scheduler gives a reasonable output for the profile in profile.txt. The best schedule output is:

Quarter	Units	Course
Win2013	4	CS275
Win2013	4	CS231A
Win2013	3	CS240
Spr2014	3	CS231B
Spr2014	3	CS244B
Spr2014	4	CS247

The total units taken for Spr2014 is just 10 when the max units limit is 12. This can be improved by making the add_units_constraint() assign more weights for more units, provided it is within the max limit, instead of a binary weight of 0 or 1.

The complete output is as follows:

```
Units: 9-12
Quarter: ['Win2013', 'Spr2014']
Taken: set(['BIOMEDIN210', 'BIOMEDIN214', 'BIOMEDIN217', 'STATS116', 'CS103', 'CS140', 'CS109',
'CS124', 'CS221', 'CS148', 'CS147', 'CS144', 'CS249A', 'CS229', 'CS228', 'CS106A', 'CS224N'])
Requests:
  Request{['CS224W'] [] [] 1}
  Request{['CS248'] [] [] 1}
  Request{['CS275'] [] [] 1}
  Request{['CS231A'] [] [] 1}
  Request{['CS240'] [] [] 1}
  Request{['CS161'] [] [] 1}
  Request{['CS224S'] [] [] 1}
  Request{['CS227B'] [] [] 1}
  Request{['CS231B'] [] ['CS231A'] 2.0}
  Request{['CS244B'] [] [] 1}
  Request{['CS247'] ['Spr2014'] [] 1}
  Request{['CS247L'] ['Spr2014'] [] 1}
Found 1146 optimal assignments with weight 2.000000 in 35113 operations
First assignment took 84 operations
2.0
('or', ((Request{['CS231B'] [] ['CS231A'] 2.0}, 'Spr2014'), 'CS231A'), 'aggregated') = True
('CS224S', 'Spr2014') = 0
(Request{['CS275'] [] [] 1}, 'Win2013') = CS275
(Request{['CS224W'] [] [] 1}, 'Win2013') = None
('sum', 'Win2013', 5) = (4, 4)
(Request{['CS247L'] ['Spr2014'] [] 1}, 'Win2013') = None
(Request{['CS231B'] [] ['CS231A'] 2.0}, 'Spr2014') = CS231B
('CS227B', 'Spr2014') = 0
('sum', 'Win2013', 6) = (4, 7)
('CS224S', 'Win2013') = 0
(Request{['CS275'] [] [] 1}, 'Spr2014') = None
('sum', 'Spr2014', 0) = (0, 0)
```

```
(Request{['CS248'] [] [] 1}, 'Spr2014') = None
('sum', 'Win2013', 7) = (7, 7)
(Request{['CS224W'] [] [] 1}, 'Spr2014') = None
('sum', 'Spr2014', 5) = (3, 7)
('CS275', 'Spr2014') = 0
(Request{['CS224S'] [] [] 1}, 'Spr2014') = None
(Request{['CS231B'] [] ['CS231A'] 2.0}, 'Win2013') = None
(Request{['CS161'] [] [] 1}, 'Spr2014') = None
('CS240', 'Win2013') = 3
(Request{['CS231A'] [] [] 1}, 'Spr2014') = None
('sum', 'Spr2014', 'aggregated') = 10
('CS224W', 'Spr2014') = 0
('sum', 'Win2013', 0) = (0, 0)
('sum', 'Spr2014', 6) = (7, 7)
(Request{['CS247'] ['Spr2014'] [] 1}, 'Spr2014') = CS247
('or', ((Request{['CS231B'] [] ['CS231A'] 2.0}, 'Spr2014'), 'CS231A'), 0) = equals
('sum', 'Win2013', 2) = (0, 0)
('sum', 'Win2013', 10) = (11, 11)
(Request{['CS161'] [] [] 1}, 'Win2013') = None
('CS240', 'Spr2014') = 0
('CS231B', 'Win2013') = 0
('sum', 'Win2013', 3) = (0, 4)
('CS224W', 'Win2013') = 0
(Request{['CS227B'] [] [] 1}, 'Spr2014') = None
('sum', 'Spr2014', 10) = (10, 10)
(Request{['CS247'] ['Spr2014'] [] 1}, 'Win2013') = None
('or', ((Request{['CS231B'] [] ['CS231A'] 2.0}, 'Win2013'), 'CS231A'), 'aggregated') = False
('CS231A', 'Win2013') = 4
('sum', 'Win2013', 4) = (4, 4)
('sum', 'Spr2014', 1) = (0, 0)
(Request{['CS244B'] [] [] 1}, 'Spr2014') = CS244B
('CS244B', 'Win2013') = 0
('sum', 'Spr2014', 3) = (3, 3)
('sum', 'Spr2014', 9) = (10, 10)
('CS247L', 'Spr2014') = 0
('CS275', 'Win2013') = 4
('CS161', 'Spr2014') = 0
(Request{['CS227B'] [] [] 1}, 'Win2013') = None
(Request{['CS244B'] [] [] 1}, 'Win2013') = None
('CS244B', 'Spr2014') = 3
('sum', 'Win2013', 8) = (7, 11)
('CS248', 'Spr2014') = 0
(Request{['CS247L'] ['Spr2014'] [] 1}, 'Spr2014') = None
('sum', 'Win2013', 9) = (11, 11)
(Request{['CS240'] [] [] 1}, 'Spr2014') = None
('sum', 'Win2013', 1) = (0, 0)
(Request{['CS231A'] [] [] 1}, 'Win2013') = CS231A
('CS231B', 'Spr2014') = 3
```

```
('CS247', 'Spr2014') = 4
('sum', 'Spr2014', 8) = (10, 10)
(Request{['CS248'] [] [] 1}, 'Win2013') = None
('CS248', 'Win2013') = 0
('sum', 'Spr2014', 2) = (0, 3)
('sum', 'Win2013', 11) = (11, 11)
(Request{['CS224S'] [] [] 1}, 'Win2013') = None
('CS231A', 'Spr2014') = 0
(Request{['CS240'] [] [] 1}, 'Win2013') = CS240
('CS227B', 'Win2013') = 0
('sum', 'Win2013', 'aggregated') = 11
('sum', 'Spr2014', 4) = (3, 3)
('CS247', 'Win2013') = 0
('CS247L', 'Win2013') = 0
('sum', 'Spr2014', 7) = (7, 10)
('sum', 'Spr2014', 11) = (10, 10)
('CS161', 'Win2013') = 0
```

Here's the best schedule:

Quarter	Units	Course
Win2013	4	CS275
Win2013	4	CS231A
Win2013	3	CS240
Spr2014	3	CS231B
Spr2014	3	CS244B
Spr2014	4	CS247

~~3d~~ 3) Extra Credit:

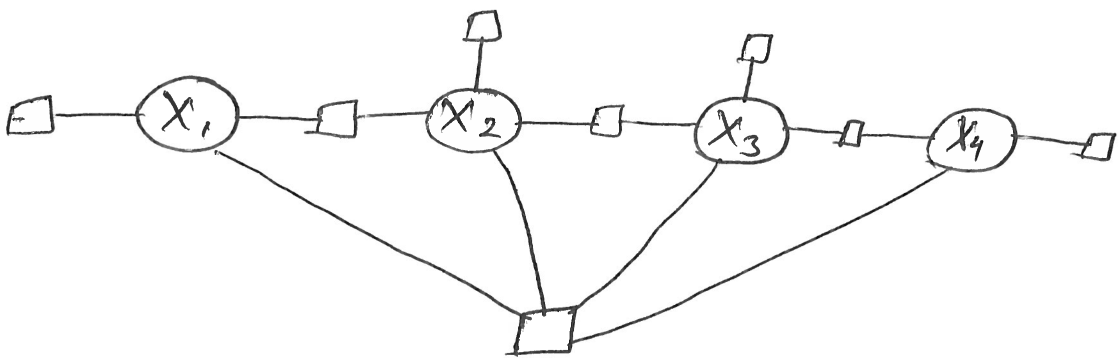
a) Tree width, by definition, is the maximum arity of the potential introduced by variable elimination.

Let there be n variables $X_1, \ldots, X_n$.
Let the longest notable pattern in P be $m \leq n$.
If this longest pattern is introduced as a potential and when one removes a variable from the set of variables connected to the potential, a new potential of arity $m-1$ is created.
This would be the maximum arity potential that would be created. Thus tree width is ~~Let n = 4, m = 3~~ length of longest notable pattern minus one.

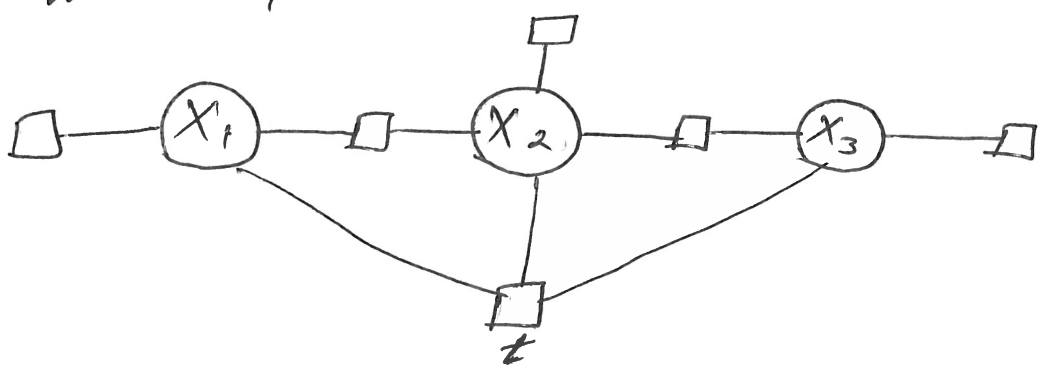
Worst case treewidth = length of longest possible notable pattern − 1

$= n - 1$

Let $n = 4$.



When X_4 is removed,



tree width = arity of 't' = 3

EC b) Let there be 'n' variables $X_1, \dots X_n$ with domain $= \{1, \dots, k\}$. Let P be the notable pattern ~~tat~~ set. Let m be the length of longest pattern

$\Rightarrow m \leq n$.

Let the potentials b/w consecutive variables be $f_i(x) = g(x_i, x_{i+1})$ for $i = 1, \dots, n-1$

Let γ be the weight given for an occurrance of a notable pattern.

Algorithm :-

i) Create a dict D with keys as the notable pattern and values as γ.

ii) Sort $|P|$ in ascending order of length.

iii) for $i = 1$ to $|P|$:

for $j = i+1$ to $|P|$:

Do KMP string matching of P_i & P_j

Let C = no. of matches of P_i in P_j

eg: if $P_i = \{a, b\}$, $P_j = \{a, b, a, b\}$

$C = 2$

Update D as $D[P_j] *= \gamma ** C$

~~Incorporate weight from f(x) potentials for each pair of elements in P~~

iv) for $i=1$ to $|P|$:

Incorporate weights form $P(x)$ for each pair of elements in P_i into dict D.

eg: if $P_i = \{a, b, c\}$

$$D[P_i] \mathrel{*}= g(a,b) * g(b,c)$$

v) ~~Pick the roo~~ Sort D on decreasing order of values. Break ties by preferring shorter keys (or) patterns.

vi) foreach pattern in D:

generate a n-length string with maximum overlap.

eg: if pattern = "aba" & n = 5

result = "ababa"

Update D[pattern] with new weight based on the n-length string

vi) Pick the ~~string with~~ pattern with highest weight in D & return its n-string values as the maximum weight assignment.

Space = $O(n+m)$ (from KMP string matching)

Time = $O(|P|^2)$