15-780: Graduate AI
Homework Assignment #2 Solutions

Out: February 12, 2015
Due: February 25, 2015

Collaboration Policy: You may discuss the problems with others, but you must write all code and your writeup independently.

Turning In: Please email your assignment by the due date to `shayand@cs.cmu.edu` and `vdperera@cs.cmu.edu`. Make sure your solution to each problem is on a separate page. If your solutions are handwritten, then please take photos or scan them and make sure they are legible and clear. Please submit your code in separate files so we can easily run them

1 Cryptoarithmethic

Solve the cryptarithmethic problem shown in Fig. 1 by hand, using the strategy of backtracking with forward checking and the MRV and least-constraining-value heuristic.

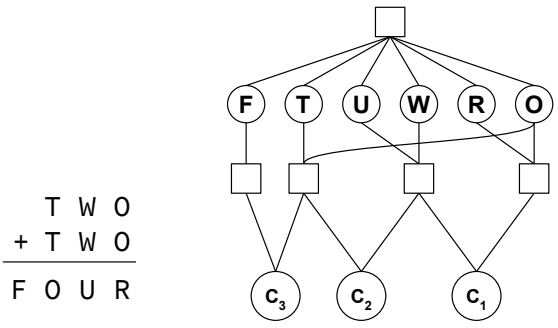


Figure 1: (a) The cryptarithmethic problem. (b) The constraint hypergraph.

In a cryptarithmethic problem each letter stands for a distinct digit; the aim is to find a substitution of digits for letters such that the resulting sum is arithmetically correct, with the added restriction that no leading zeros are allowed.

To help you out in Fig.1 you can see the constraint hypergraph for the cryptoarithmethic problem, showing the *Aldiff* constraint (square box on top) as well as the column addition

constraints (four square box in the middle). The variables C_1,C_2 and C_3 represent the carry digits for the three column.

An *Aldiff* constraint is a global constraint which says that all of the variable involved in the constraint must have different value.

The following solution was provided by Revanth Bhattaram (with slight modifications):

For this problem,we use the Minimum Remaining Value Heuristic to choose a variable and the Least Constraining Value heuristic to assign values to the chosen variables.
The constraints are :

$$O + O = R + 10 * C_1$$
$$C_1 + W + W = U + 10 * C_2$$
$$C_2 + T + T = O + 10 * C_3$$
$$F = C_3$$

The main variables here are F, T, U, W, R, O and they all must have different assigned values. In addition, C_1, C_2, C_3 represent the carries that depend on the assigned values. The carries can take the values {0,1}.
Another constraint for this problem is that the number don't have any leading zeros. This implies that the variables F, T can't take the value 0. The backtracking search algorithm runs as follows :

- A quick look at the constraints shows that the variable F can only take the value 1, since C_3 can be 0 or 1 and $F = C_3$ and F can't be equal to 0. Thus, we set $F = 1$ and consequently $C_3 = 1$.

  $$\text{F} = 1$$

- After this point, the domain of values for the variables U, W, R, O is {0,2,3....9} and the domain of values for the variable T is {2,3....9}. Thus, using the MRV heuristic, we'll now be assigning a value to T.
  Consider the constraints at this state of time : $C_2+2T = O+10 \implies O = C_2+2T-10$. Assigning the values 2,3,4 to T results in O having no possible values. Assigning the value 5 leaves out just a single value for $O = 0$. Setting the value T = {6,7,8} results in O having two possible values. Thus, using the least constraining value heuristic, we set $T = 6$.

  F=1
  |
  T=6

- At this point, one of the constraints becomes $C_2 + 12 = O + 10 \implies O = C_2 + 2$. O's domain is now {2,3} which is smaller than any of the other variables and thus the MRV heuristic directs us to choose to assign a value for O.
  The least constraining value heuristic doesn't help us too much over here and so we proceed in assigning values in order. We now set $O = 2$.

F=1
|
T=6
|
O=2

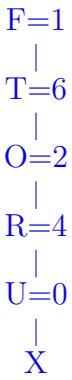
- Setting $O = 2 \implies C_2 = 0$. The constraints are now $C_1 + 2W = U, R + 10C_1 = 4$. Now, observe the variable R. We have $R = 4 - 10C_1$. The possible values for R are {4} and thus the MRV heuristic directs us to use assign this variable. We set $R = 4$.

F=1
|
T=6
|
O=2
|
R=4

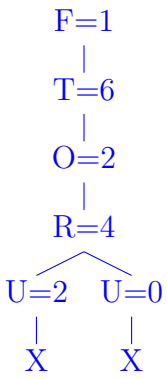
- The constraint now is $2W = U$. The domain of U is the set of remaining even values = {0,8} and has a smaller domain than W. Thus, we now choose to assign a value to U. The least constraining value heuristic value doesn't help narrow down between the two values (they're both bad).

F=1
|
T=6
|
O=2
|
R=4
|
U=0

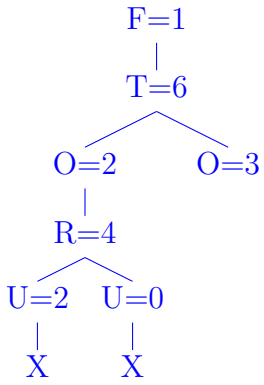
- Setting $U = 0 \implies W = 0$ which is a contradiction and so we backtrack.



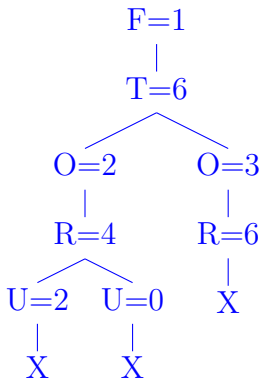
- Similarly, setting $U = 2$ doesn't work as well since 4 has already been assigned to R and so we backtrack.



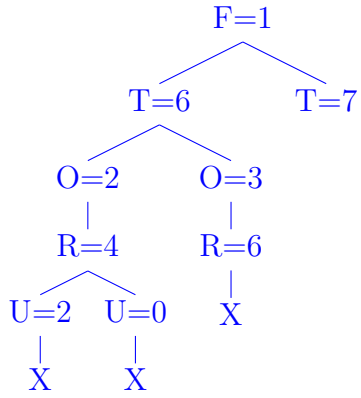
- We now backtrack all the way up to O and set the value of O to be 3.



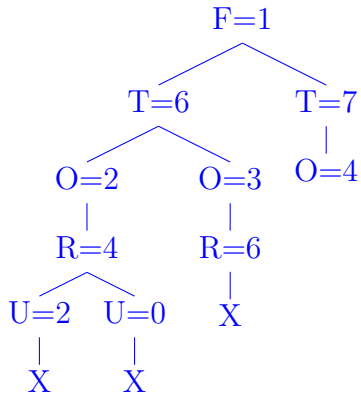
- Setting $O = 3 \implies C_2 = 1$. The constraints are now $C_1 + 2W = U + 10, R + 10C_1 = 6$. Similar to before, R can only take one value - 6. However, this value has already been assigned and so we now backtrack to T.



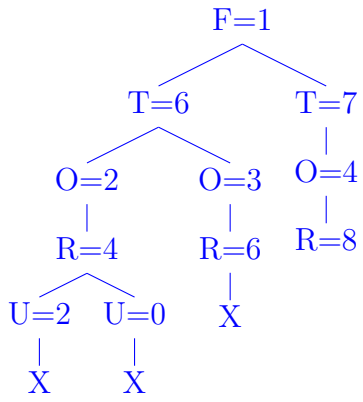
- We now set $T = 7$ - the next possible value for T.



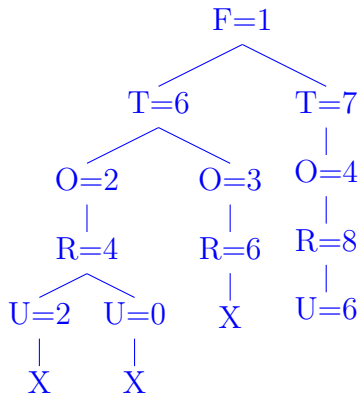
- Now that $T = 7$, the constraints are now $C_2 + 14 = O + 10 \implies O = C_2 + 4$. O can now only take two values - {4,5}. Thus, MRV directs us to choose a value for O. The LCV heuristic doesn't help much here and so we set $O = 4$.



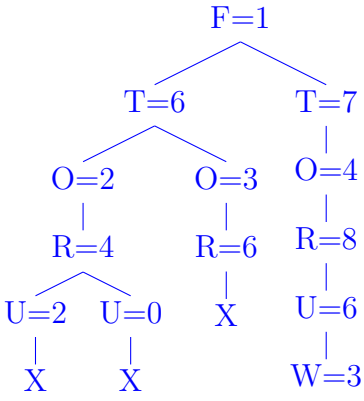
- Setting $O = 4 \implies C_2 = 0$. The constraints are now $C1 + 2W = U, 8 = R + 10C_1$. Observe how the only possible value for R is 8 and thus MRV dictates that we assign that value to R.



- Setting $R = 8 \implies C_1 = 0$. The constraints are now $2W = U$. The possible values for U are {0,2,6} which is a domain smaller than that of W (all remaining values). Thus, we now pick a value for U. The LCV heuristic dictates that we select the value $U = 6$.



- We are now left with only possible value for W. We'll get $W = 3$. Notice that this satisfies all constraints and since we've assigned values for all variables, this is a satisfying assignment.



Thus, the solution to this problem is $F = 1, T = 7, O = 4, R = 8, W = 3, U = 6$.

2 Scheduling Nightmare

As an overworked student, you must complete all of your homework assignments in each of your classes before they are due at the end of the semester (at time $t = T$). Each homework assignment has a specific duration $d \in \mathbb{N}$ representing how long it will take to complete. You must also complete them in order (so the first homework assignment in a class must be completed before the second in that same class); however, homework assignments in one class can be completed in any order relative to assignments in another class.

Since you are a student in computer science, many of your homework assignments require the use of specific machines for experimentation. You have a set of k computers; some homework assignments require one or more specific computers, and occupy them entirely for the entire duration of the homework assignment (i.e., if both homework A and homework B require computer 1, then you cannot work on A and B at the same time).

An Example Semester

Fig. 2 gives an example semester-long homework schedule for a student. The student has four classes, and each class has either three or four homework assignments. These homework assignments have (i) a duration in time periods, (ii) a relative ordering to other homework assignments in the same class, and (iii) an optional resource requirement. We assume the semester is 15 time periods long.

For example, the homework $H_{0,1}$ has duration 4 and requires resource 0. Similarly, homework $H_{1,0}$ has duration 1 and also requires resource 0. As discussed above, the shared resource constraint would prevent the student from overlapping these two homeworks in her schedule.

Please use the example from Fig. 2 in these problems.

1. What are the variables and their domains?

2. Describe how you would write this problem as a CSP with arity $r > 2$, e.g., with at least some constraints having at most r variables.

3. Qualitatively, how would this change if you were limited to arity $r = 2$? Recall from class that any CSP can be written this way.

4. Solve your scheduling CSP—either the $r > 2$ or $r = 2$ version—by hand or with code using (i) a deterministic ordering of variables and values (e.g., alphabetical order) and (ii) the most constrained variable and least constraining value heuristics discussed in class. How much of an effect did these heuristics have on the size of the search tree? Quantify your answer.

5. The problem above is solved relatively easily with simple heuristics and backtracking search. This is not always the case. Come up with a short "worst-case" instantiation of the general scheduling problem. What are some properties of these search problems

that will thwart the most constrained variable heuristic? The least constraining value heuristic?

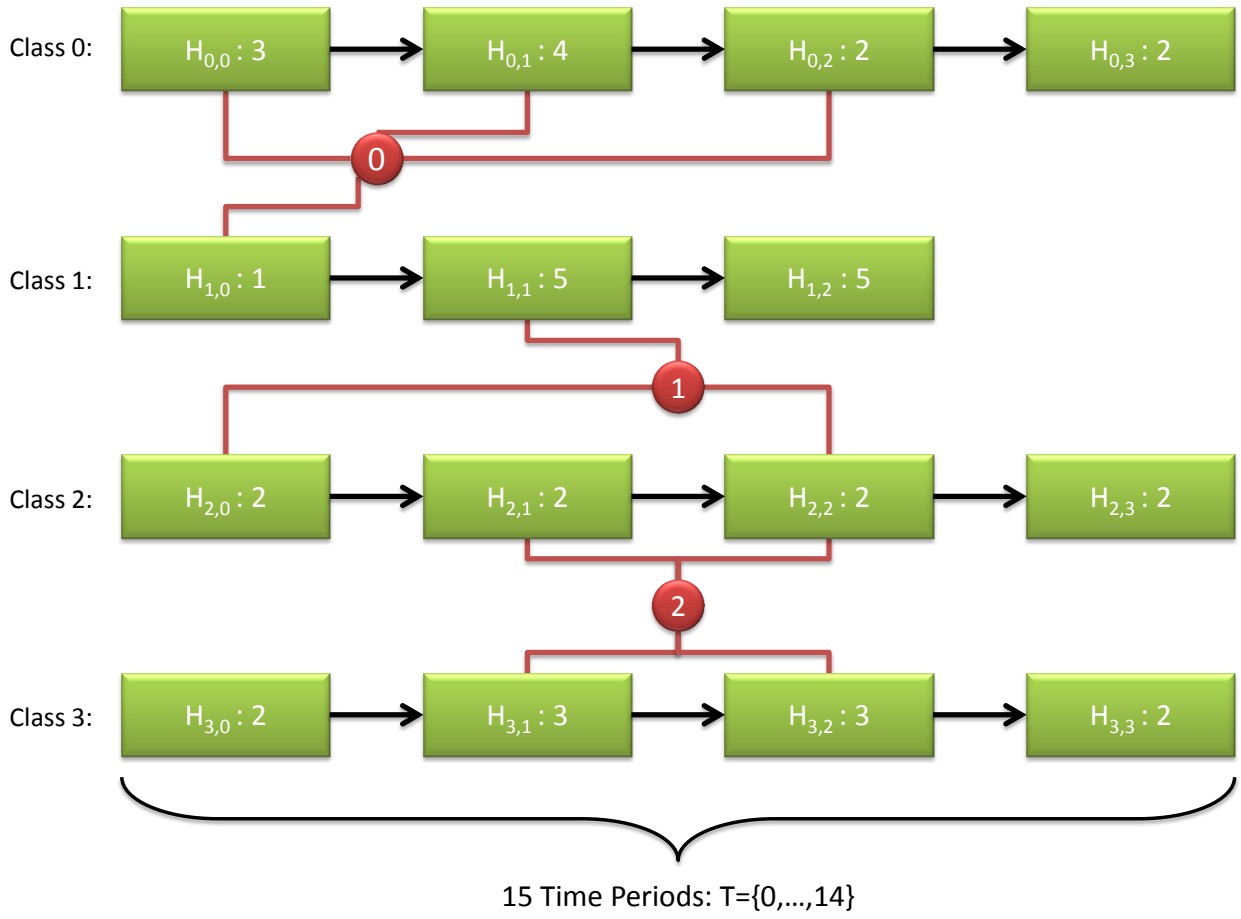


Figure 2: A semester's worth of homeworks for four classes. Each homework has a relative ordering (e.g., $H_{0,0}$ must be completed before $H_{0,1}$), an integral duration, and (possibly) a resource constraint. The semester is assumed to be 15 time periods in length.

The following solution was provided by Revanth Bhattaram (with slight modifications):

(a) We will denote the variables as $S_{i,j}$, which represents the start time of $H_{i,j}$ (the jth homework of the ith course). $S_{i,j}$ can take values from 0 to T.

(b) There are essentially three types of constraints for this problem :

(i) **Completion Constraints :** These constraints basically dictate that the $S_{i,j}$ values must be such that the task will be able to finish before the semester ends. This can be written in mathematical form as :

$$\forall (i,j), S_{i,j} + H_{i,j}.duration \leq T$$

(ii) **Precedence Constraints :** These are the constraints that enforce the requirement that each homework in a course can be started only after the previous homework (in the same course) has been completed. This can be represented as :

$$\forall i, InOrder(S_{i,0}, S_{i,1}....S_{i,j-1}, S_{i,j})$$

Where the function $InOrder$ checks if each homework starts after the previous one has been completed.

(iii) **Resource Sharing Constraints :** Some homeworks require the use of one or more resources. When two homeworks use the same resource, they cannot be worked on at the same time (or overlapping periods). These constraints handle such cases. I'll define a function $NotOverlapping(R_i)$ that checks if the all the homeworks that use resource R_i have non-overlapping time periods.

$$\forall i, NotOverlapping(R_i)$$

For the example for this problem, $T = 14$ and the possible pair of values for (i, j) are $\{(0,0), (0,1), (0,2), (0,3), (1,0), (1,1), (1,2), (2,0), (2,1), (2,2), (2,3), (3,0), (3,1), (3,2), (3,3)\}$

(c) If we were limited to arity $r = 2$, then the constraints have to be restructured in order to satisfy this restriction. To do so, each $r > 2$ arity constraint (such as $InOrder()$ and $NotOverlapping()$) can be replaced by a set of binary constraints. The number of constraints will have to increase for this but the same restrictions would hold. For arity $r = 2$, the constraints are as follows :

(i) **Completion Constraints :** $\forall (i, j), S_{i,j} + H_{i,j}.duration \leq T$

(ii) **Precedence Constraints :** $\forall (i, j), S_{i,j} + H_{i,j}.duration \leq S_{i,(j+1)}$

(iii) **Resource Sharing Constraints :** $\forall (i_1, j_1)$ and (i_2, j_2) that share a constraint,

$$S_{i_1,j_1} + H_{i_1,j_1}.duration \leq S_{i_2,j_2}$$

$$or$$

$$S_{i_2,j_2} + H_{i_2,j_2}.duration \leq S_{i_1,j_1}$$

If you wish to express this as a single inequality, write is as :

$$(S_{i_1,j_1} + H_{i_1,j_1}.duration - S_{i_2,j_2})(S_{i_2,j_2} + H_{i_2,j_2}.duration - S_{i_1,j_1}) \leq 0$$

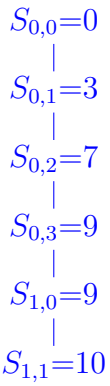
(d) For both parts we will use forward checking to eliminate values after each stage of the search algorithm. For the first part, we will use an alphabetical ordering of variables and increasing order of values for the deterministic ordering.

(i) The search algorithm for this part works as follows :

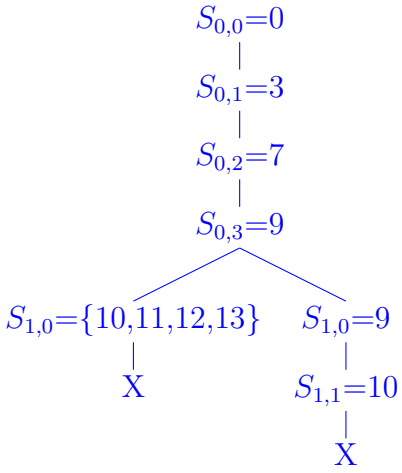
- We first set a value of 0 for $S_{0,0}$.

$$S_{0,0} = 0$$

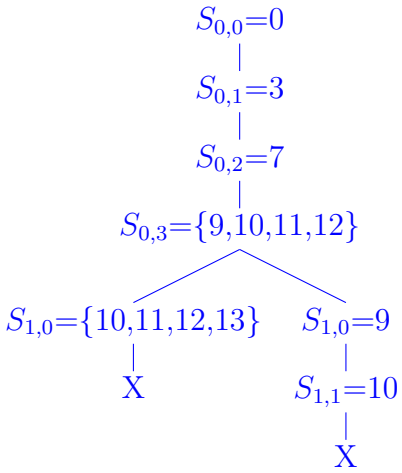
- From here on, we proceed to set values for $S_{0,1}, S_{0,2}, S_{0,3}, S_{1,0}, S_{1,1}$



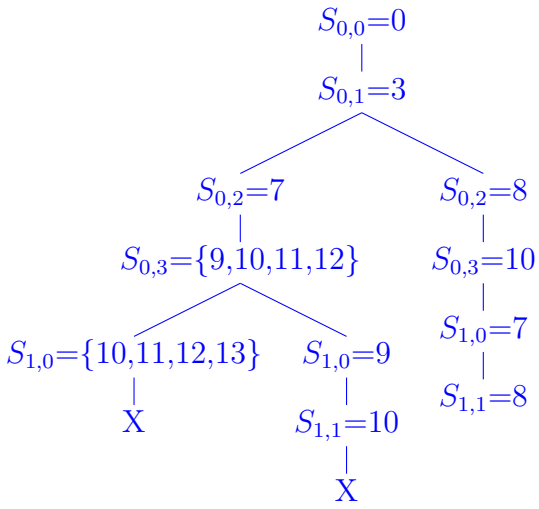
- At this point, we notice that $S_{1,2}$ has no possible values that can be set and so we backtrack. $S_{1,1}$ has no other possible values and so we work up to $S_{1,0}$. Setting $S_{1,0} = 10$ rules out all values for $S_{1,1}$. Same goes for the remaining values of $S_{1,0}$.



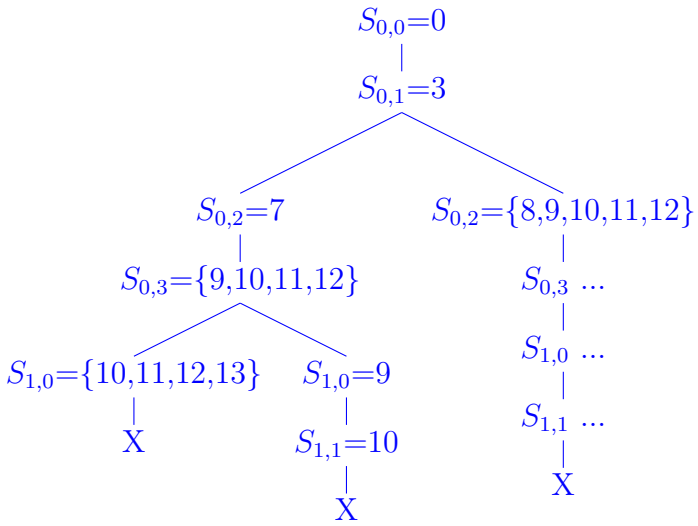
- You now backtrack to $S_{0,3}$ and try the remaining values. Again, this doesn't make a difference and results in the same assignments and disappointments.



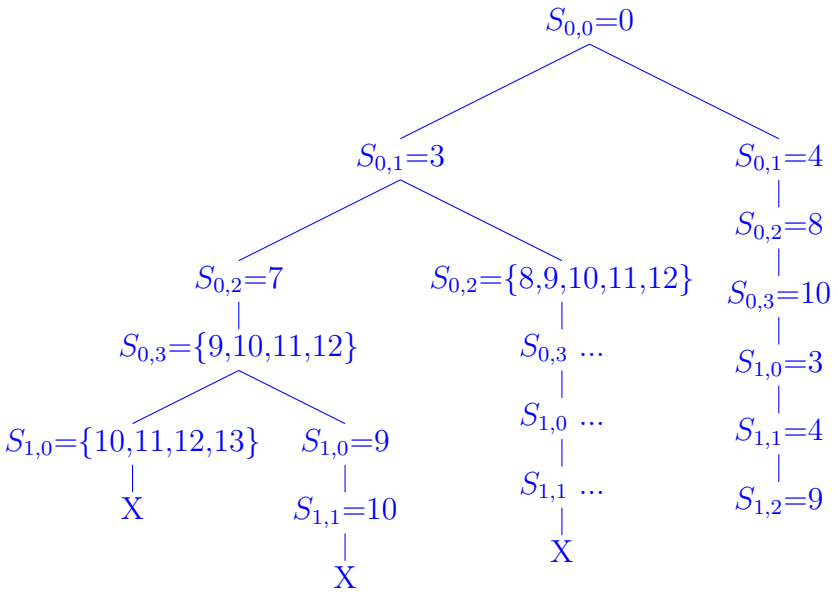
- We now set $S_{0,2} = 8$ and consequently set the values for $S_{0,3}, S_{1,0}, S_{1,1}$



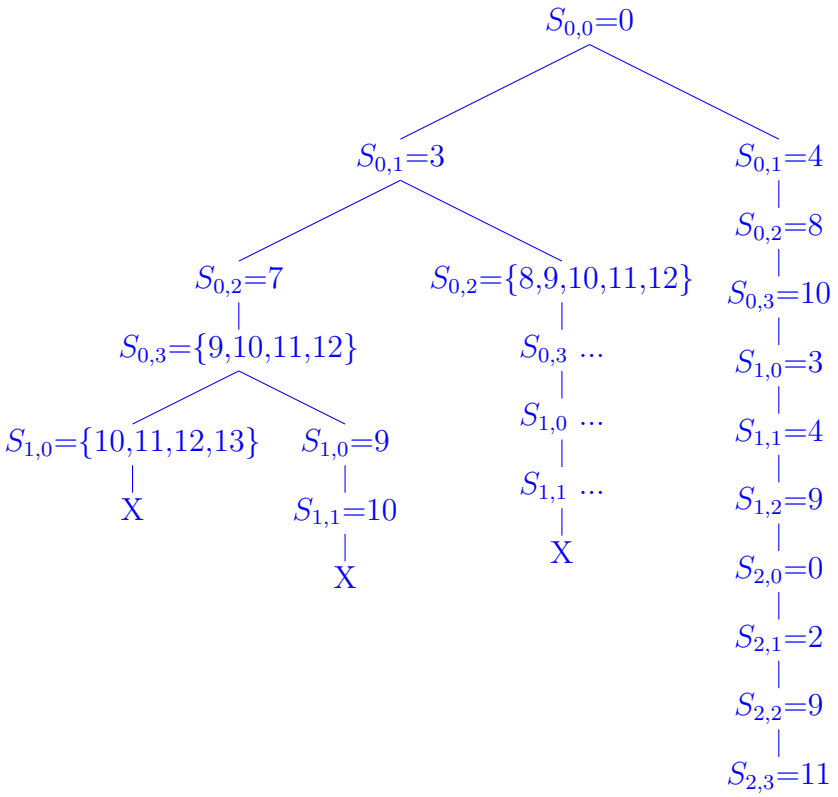
- Now, when we try to set a value for $S_{1,2}$, we see that we have no possible values and so we backtrack. We notice that going back to $S_{0,2}$ and try all possible values doesn't help much and always results in disappointment. The search tree (in a compressed form is as follows) :



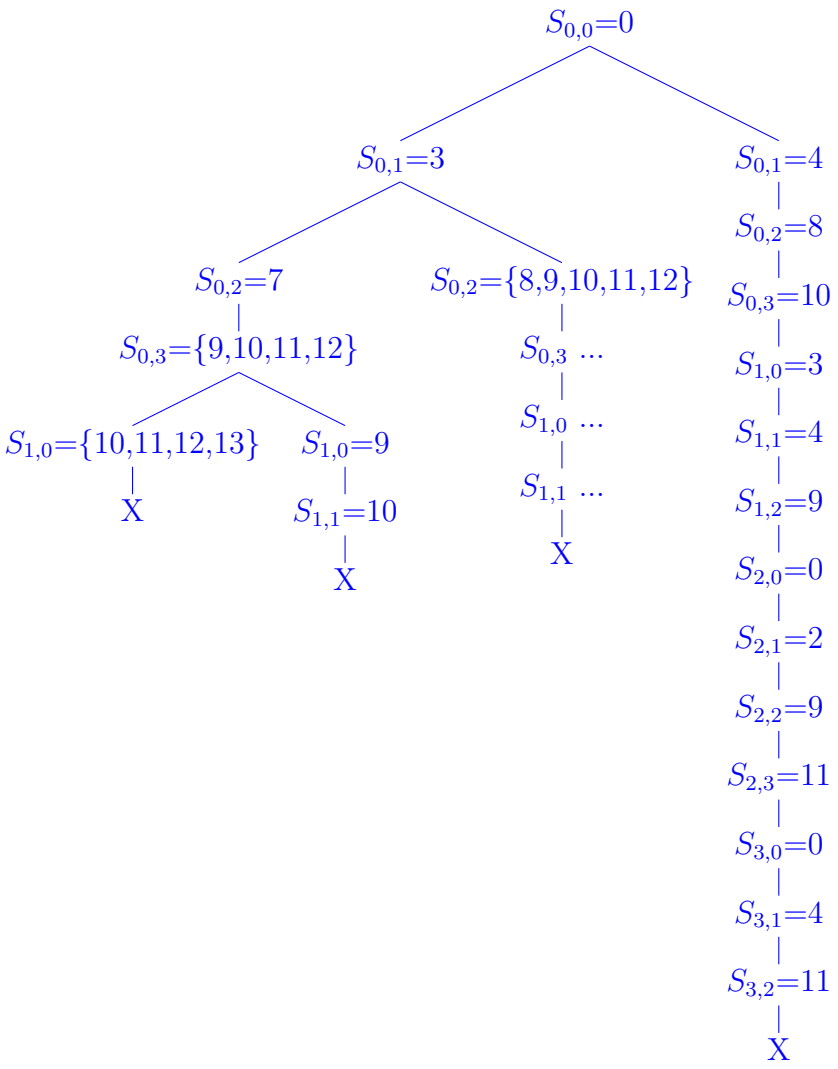
- We now go all the way back to $S_{0,1}$ and set it to be 4. Consequently, values for $S_{0,2}, S_{0,3}, S_{1,0}, S_{1,1}, S_{1,2}$. As it cab be seen, we finally have a valid value for $S_{1,2}$.



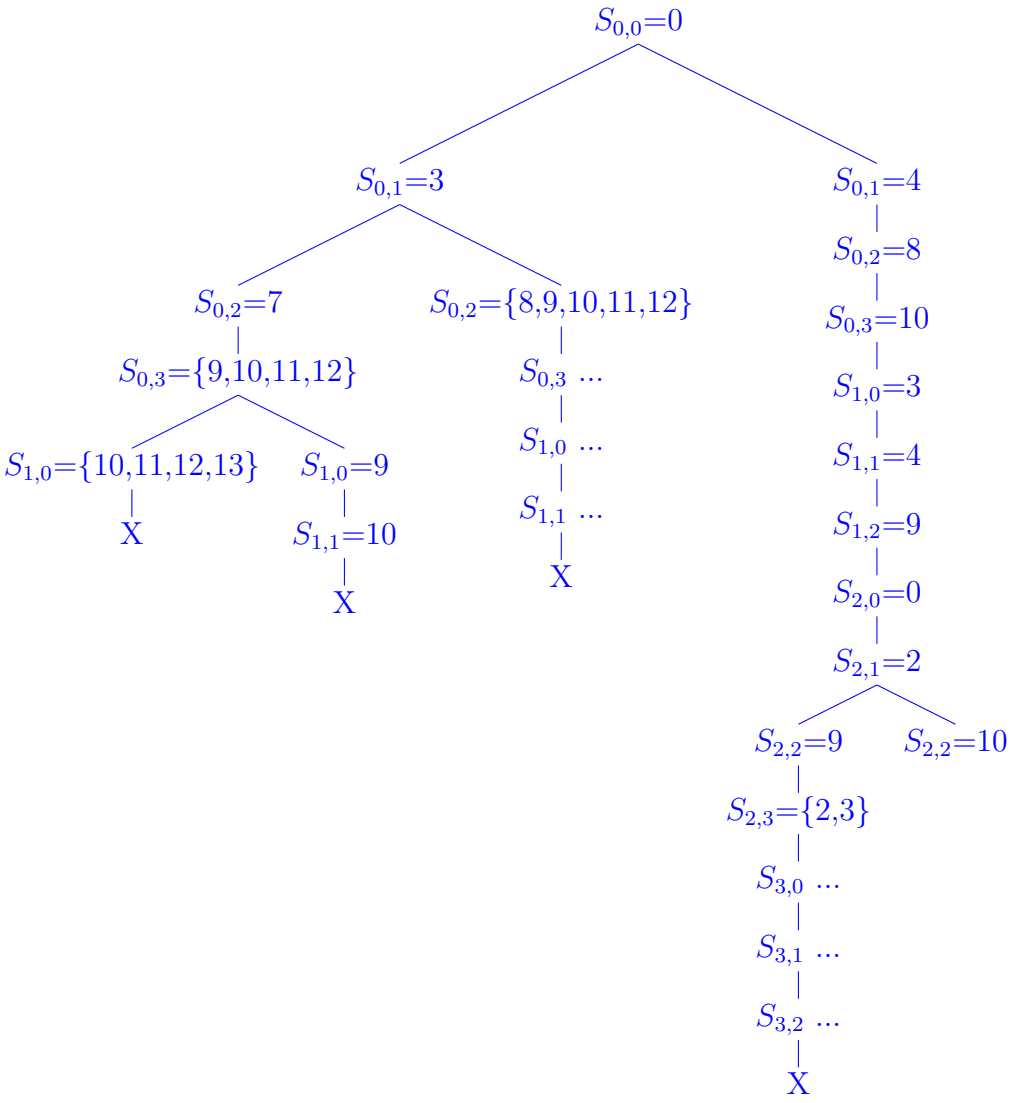
- Continue to set the values of $S_{2,0}, S_{2,1}, S_{2,2}, S_{2,3}$. The only thing to be cleared up here is that we're setting $S_{2,2} = 9$ because of the fact that $H_{1,1}$ that uses the same resource finishes at 9.



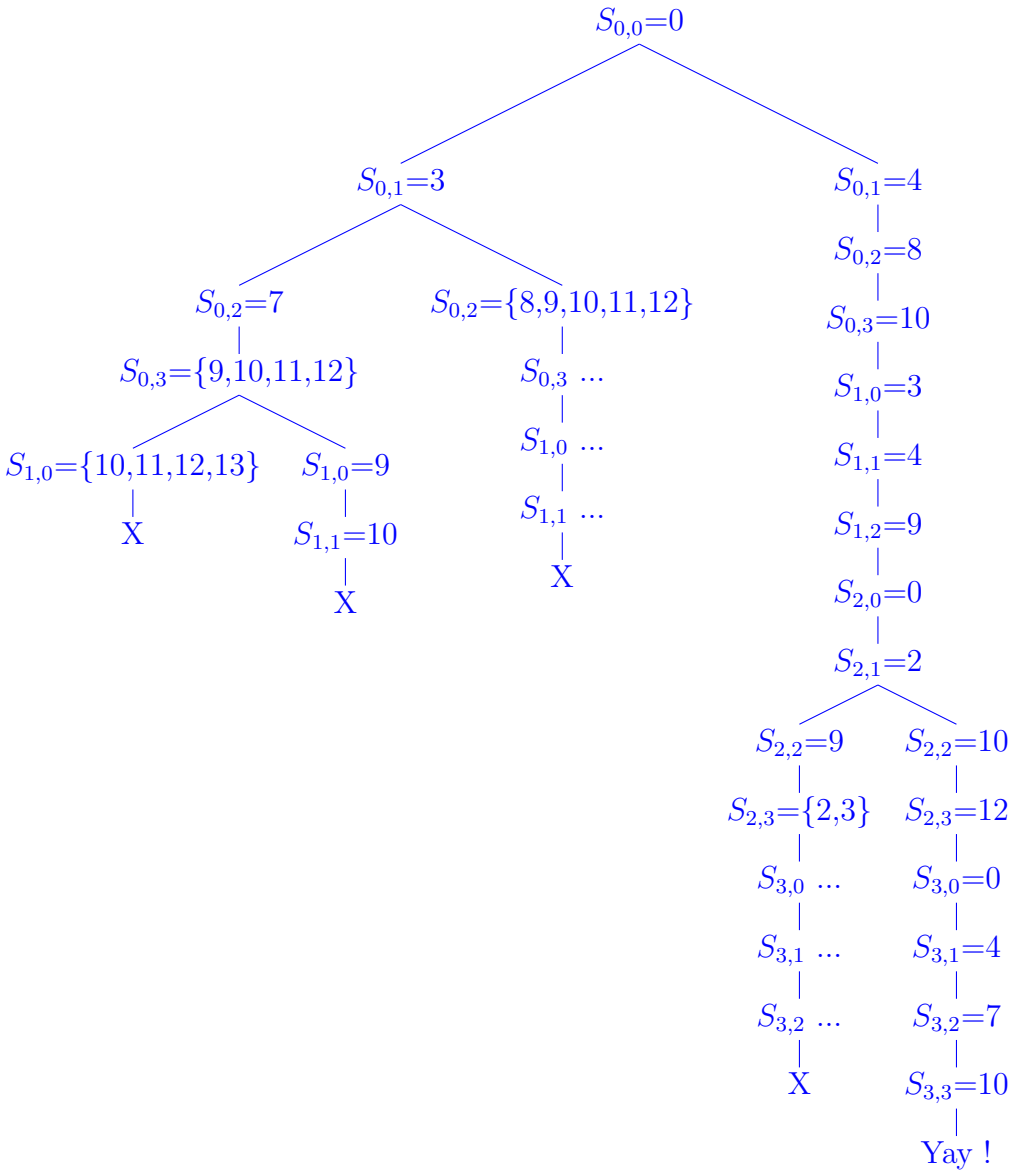
- Next, we go to assign values for course 3. $S_{3,0}$ is set to be 0. Observe that $S_{3,1}$ can only start at 4 (since $H_{2,1}$ uses the same resource). That, along with the fact that $H_{2,2}$ starts at 9 restricts us to set $S_{3,2} = 11$. However, that leaves out any possible values for $S_{3,3}$ and thus we have to backtrack.



- We keep on backtracking to $S_{3,2}, S_{3,1}, S_{3,0}, S_{2,3}$ but find that we still don't get solution. We now move on to $S_{2,2} = 10$.



- Things now become smooth and we just assign values one by one until we run out of variables to assign to.



Thus, the solution obtained here is

- $S_{0,0} = 0, S_{0,1} = 4, S_{0,2} = 8, S_{0,3} = 10$
- $S_{1,0} = 3, S_{1,1} = 4, S_{1,2} = 9$
- $S_{2,0} = 0, S_{2,1} = 2, S_{2,2} = 10, S_{2,3} = 12$
- $S_{3,0} = 0, S_{3,1} = 4, S_{3,2} = 7, S_{3,3} = 10$

(ii) For this part, we will use the MRV(Minimum Remaining Variables) heuristic to choose a variable and the LCV(Least Constraining Value) heuristic to choose value for the selected variable. The search proceeds as follows :

- Initially, the only constraints that we can take a look at are the completion constraints. These constraints restrict the domains of each of the variables. We observe that the variables with the minimum remaining values are $S_{1,1}$ and $S_{1,2}$. We randomly break the tie and select $S_{1,1}$. Now, we have to apply the LCV heuristic while assigning a value. Setting $S_{1,1} = 0$ makes it impossible for $S_{1,0}$ to take any possible values (because the precedence constraint would be violated) and so we don't assign this value. For any of the remaining values, we observe that the number of values restricted for $S_{1,0}$ and $S_{1,2}$ combined are similar and so we set $S_{1,1} = 1$. The domains of $S_{1,0}, S_{1,2}, S_{2,0}, S_{2,2}$ are reduceed due to this assignment.

$$S_{1,1}{=}1$$

- At this stage, $S_{1,0}$ has only one possible value and so the MRV heuristic directs us to choose this variable. Since there's only one possible value, there's no confusion there.

$$\begin{array}{c} S_{1,1}{=}1 \\ | \\ S_{1,0}{=}0 \end{array}$$

- Now, the variable with the smallest domain is $S_{1,2}$. LCV doesn't help much here and so we use the first value in the domain and set $S_{1,2} = 6$.

$$\begin{array}{c} S_{1,1}{=}1 \\ | \\ S_{1,0}{=}0 \\ | \\ S_{1,2}{=}6 \end{array}$$

- There's now a tie between $S_{2,0}$ and $S_{2,2}$ when using the MRV heuristic. We just pick $S_{2,0}$ and set it to be 6 (using LCV). The domains of the remaining homeworks of course 2 are reduced.

$$\begin{array}{c} S_{1,1}{=}1 \\ | \\ S_{1,0}{=}0 \\ | \\ S_{1,2}{=}6 \\ | \\ S_{2,0}{=}6 \end{array}$$

- The domains of $S_{2,1}, S_{2,2}, S_{2,3}$ are the same (and smaller than the other variables) and so we use MRV and a tie-breaking to assign a value of 8 to $S_{2,1}$. The domains of $S_{2,2}, S_{2,3}, S_{3,1}, S_{3,2}$ are reduced.

$$S_{1,1}{=}1$$
$$|$$
$$S_{1,0}{=}0$$
$$|$$
$$S_{1,2}{=}6$$
$$|$$
$$S_{2,0}{=}6$$
$$|$$
$$S_{2,1} = 8$$

- MRV directs us to choose $S_{2,2}$ or $S_{2,3}$. Lets take up $S_{2,2}$ and set it to be 10. Why are we using 10 ? Because any other value would leave out 0 possible values for $S_{2,3}$. The domains of $S_{2,3}, S_{3,1}, S_{3,2}$ are now reduced.

$$S_{1,1}{=}1$$
$$|$$
$$S_{1,0}{=}0$$
$$|$$
$$S_{1,2}{=}6$$
$$|$$
$$S_{2,0}{=}6$$
$$|$$
$$S_{2,1} = 8$$
$$|$$
$$S_{2,2} = 10$$

- We have no choice here. $S_{2,3}$ has only one value we set that value. Thus, we now put $S_{2,3} = 12$.

$$S_{1,1}{=}1$$
$$|$$
$$S_{1,0}{=}0$$
$$|$$
$$S_{1,2}{=}6$$
$$|$$
$$S_{2,0}{=}6$$
$$|$$
$$S_{2,1} = 8$$
$$|$$
$$S_{2,2} = 10$$
$$|$$
$$S_{2,3} = 12$$

- The domains of $S_{3,1}$ and $S_{3,2}$ are now the smallest. So, we now choose $S_{3,1}$ as the next variable. We use LCV to set $S_{3,1} = 2$. Why ? Because setting a value less than 2, would rule out any possible value for $S_{3,0}$.

$$S_{1,1}{=}1$$
$$|$$
$$S_{1,0}{=}0$$
$$|$$
$$S_{1,2}{=}6$$
$$|$$
$$S_{2,0}{=}6$$
$$|$$
$$S_{2,1} = 8$$
$$|$$
$$S_{2,2} = 10$$
$$|$$
$$S_{2,3} = 12$$
$$|$$
$$S_{3,1} = 2$$

- Next, we see that $S_{3,0}, S_{3,2}$ have their domains restricted to a size of 1. So, we pick either and set it to be this remaining value. We set $S_{3,0} = 0$ and then set $S_{3,2} = 5$

$$S_{1,1}{=}1$$
$$|$$
$$S_{1,0}{=}0$$
$$|$$
$$S_{1,2}{=}6$$
$$|$$
$$S_{2,0}{=}6$$
$$|$$
$$S_{2,1} = 8$$
$$|$$
$$S_{2,2} = 10$$
$$|$$
$$S_{2,3} = 12$$
$$|$$
$$S_{3,1} = 2$$
$$|$$
$$S_{3,0} = 0$$
$$|$$
$$S_{3,2} = 5$$

- The next variable to be set now is $S_{3,3}$. We set it to be 8.

$$S_{1,1}{=}1$$
$$|$$
$$S_{1,0}{=}0$$
$$|$$
$$S_{1,2}{=}6$$
$$|$$
$$S_{2,0}{=}6$$
$$|$$
$$S_{2,1} = 8$$
$$|$$
$$S_{2,2} = 10$$
$$|$$
$$S_{2,3} = 12$$
$$|$$
$$S_{3,1} = 2$$
$$|$$
$$S_{3,0} = 0$$
$$|$$
$$S_{3,2} = 5$$
$$|$$
$$S_{3,3} = 8$$

- Of the remaining variables, $S_{0,1}$ has the minimum remaining values (because of the length of the homework and the resource constraint of $H_1, 0$). LCV tells us to set $S_{0,1} = 4$ because values less than 4 would leave no possible assignments for $S_{0,0}$.

$$S_{1,1}{=}1$$
$$|$$
$$S_{1,0}{=}0$$
$$|$$
$$S_{1,2}{=}6$$
$$|$$
$$S_{2,0}{=}6$$
$$|$$
$$S_{2,1} = 8$$
$$|$$
$$S_{2,2} = 10$$
$$|$$
$$S_{2,3} = 12$$
$$|$$
$$S_{3,1} = 2$$
$$|$$
$$S_{3,0} = 0$$
$$|$$
$$S_{3,2} = 5$$
$$|$$
$$S_{3,3} = 8$$
$$|$$
$$S_{0,1} = 4$$

- Now that we've given a value to $S_{0,1}$, the domain of $S_{0,0}$ gets restricted and so

MRV gets us to choose that variable for assignment. We set $S_{0,0} = 1$.

$$
\begin{array}{c}
S_{1,1}{=}1 \\
| \\
S_{1,0}{=}0 \\
| \\
S_{1,2}{=}6 \\
| \\
S_{2,0}{=}6 \\
| \\
S_{2,1} = 8 \\
| \\
S_{2,2} = 10 \\
| \\
S_{2,3} = 12 \\
| \\
S_{3,1} = 2 \\
| \\
S_{3,0} = 0 \\
| \\
S_{3,2} = 5 \\
| \\
S_{3,3} = 8 \\
| \\
S_{0,1} = 4 \\
| \\
S_{0,0} = 1
\end{array}
$$

- The last two variables left are $S_{0,2}$ and $S_{0,3}$. They have the same domain and so we set these variables in order. We set $S_{0,2} = 8$ and $S_{0,2} = 10$. This concludes the run of the search algorithm. No backtracking !

$S_{1,1}$=1
|
$S_{1,0}$=0
|
$S_{1,2}$=6
|
$S_{2,0}$=6
|
$S_{2,1} = 8$
|
$S_{2,2} = 10$
|
$S_{2,3} = 12$
|
$S_{3,1} = 2$
|
$S_{3,0} = 0$
|
$S_{3,2} = 5$
|
$S_{3,3} = 8$
|
$S_{0,1} = 4$
|
$S_{0,0} = 1$
|
$S_{0,2} = 8$
|
$S_{0,3} = 10$
|
Yay !

Thus, the solution obtained here is

- $S_{0,0} = 1, S_{0,1} = 4, S_{0,2} = 8, S_{0,3} = 10$
- $S_{1,0} = 0, S_{1,1} = 1, S_{1,2} = 6$
- $S_{2,0} = 6, S_{2,1} = 8, S_{2,2} = 10, S_{2,3} = 12$
- $S_{3,0} = 0, S_{3,1} = 2, S_{3,2} = 5, S_{3,3} = 8$

Observe how we didn't have to backtrack a single time unlike before where we had to backtrack loads of times. The sizes of the search trees speak volumes. This show how heuristics help improve performance while searching.

(e) **Where MRV Fails :** Consider the following example :

$H_{0,0}(2) - - - - H_{0,1}(2) - - - - H_{0,2}(10)$

Here, we have one course with three homeworks. The semester ends at T=14.

Now, if we were to use MRV, we would first select an assignment for $S_{0,2}$. Now, if you were to set values in order and set $S_{0,2} = 0, 1, 2...$, then you would have to backtrack several times since setting $S_{0,2}$ to a value less than 4, will leave out no values for the other two variables.

However, if you were to go in a deterministic order, then the assignments would be $S_{0,0} = 0, S_{0,1} = 2, S_{0,2} = 4$. This would be done in just three steps with no backtracking whatsoever. Thus, in this case, using a heuristic actually has a negative effect.

In general, MRV could fail in cases where an ordering has to be followed. In such cases, MRV could select a variable without considering the ordering that has to be maintained and thus assign values that will eventually violate this constraint.

Where LCV Fails : Consider the following example :

$H_{0,0}(2) - - - - H_{0,1}(2) - - - - H_{0,2}(9, R)$
$H_{1,0}(13) - - - - H_{1,1}(1, R)$
$H_{2,0}(3, R)$

$H_{2,0}, H_{1,1}$ and $H_{0,2}$ share a resource R and $T = 14$.

Let's take a deterministic ordering for variable assignment. So, we'll be assigning variables in the following order $S_{0,0}, S_{0,1}, S_{0,2}, S_{1,0}, S_{1,1}, S_{2,0}$.

We first set the values $S_{0,0} = 0$ and $S_{0,1} = 2$ without having to think too much. Now, when assigning a value for $S_{0,2}$, we have two options - {4,5}.

(i) Assigning 4 would restrict the domain of $S_{1,1}$ to {0,1,2,3,13} and $S_{2,0}$ to {0,1}.

(ii) Assigning 5 would restrict the domain of $S_{1,1}$ to {0,1,2,3,5} and $S_{2,0}$ to {0,1,2}.

Thus, LCV directs us to set $S_{0,2} = 5$. However, once this has been set, we move on and set $S_{1,0} = 0$ thereby leaving no possible values for $S_{1,1}$. Thus, we have to backtrack and set $S_{0,2} = 4$.

Observe that if we followed a deterministic ordering of variables and values, we'd have the following assignments in order : $S_{0,0} = 0, S_{0,1} = 2, S_{0,2} = 4, S_{1,0} = 0, S_{1,1} = 13, S_{2,0} = 0$ and that's a valid assignment ! No backtracking at all !

In general, LCV aims to limit the number of values ruled out after each step and chooses a value that opens up several possibilities in the next level. This would not be that helpful when you had a fixed assignment/solution (which you would discard in favor of greater options using LCV).

3 Planning

1. GraphPlan can backtrack during its second phase while searching backwards from the goals through the planning graph. Explain, in words and with a simple example, why GraphPlan needs to backtrack in its backwards search.

2. If GraphPlan cannot find a solution during the backwards search, does this mean that the problem is not solvable? If so, explain why. If not, explain what GraphPlan can do to find a solution in such cases.

3. It is possible for GraphPlan to terminate after finding an n time step plan of k operators, while there actually exists an n time step plan with less than k operators in the planning graph. Show a concrete example of this and explain why, in general, this can occur.

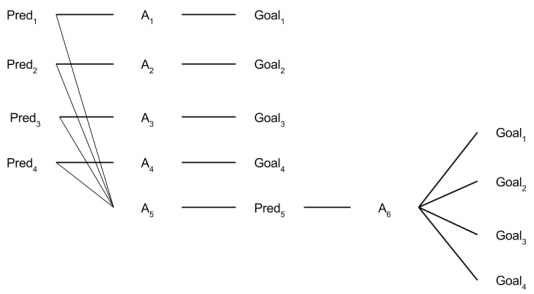
4. FF performs forward heuristic search using as an heuristic the number of levels to the goal of the relaxed GraphPlan graph (no deletes). Is this heuristic admissible? Explain why, and if not give a counterexample, i.e., show an example for which FF does not find the optimal solution.

5. **Extra credit:** We have seen in class that linear planning using a stack of goals and means-ends-analysis was incomplete (remember the one-way-rocket domain). Can you find an example for which nonlinear planning (i.e., using a set of goals, instead of a stack of goals) with means-ends-analysis is incomplete?

1. The reason why GraphPlan needs to backtrack during the backwards search phase is that, if one precondition (or goal) can be attained with multiple actions (or no-op), it might select a "dead end". We have seen sn example of this situation in the one-way rocket domain presented in class. In layer 5 we need to achieve *in*(o1, R), *in*(o2, R) and *at*(R, B). We have two ways to do so, the first and correct one is select as action *move*(A B) and two no ops. This correspond to first loading the two objects and then moving the rocket. A "dead end" would be selecting the two *load* actions and a no-op for *at*(R, B) as we would end up trying to load the two object but the rocket would no longer be there.

2. If GraphPlan cannot find a solution during the backwards search it does not necessarily mean that the problem is not solvable; it only means there is not a plan of length n, where n is the last layer reached in the extend step. In this situation GraphPlan can extend the planning graph with one more proposition level.If in the new proposition layer nothing has been added and we cannot find a plan when backtracking then we can conclude that there is no plan.

3. In the following example, we can get to the goal (specified by $Goal_1$, $Goal_2$, $Goal_3$, $Goal_4$) in two ways, either in a one step plan by applying actions 1 through 4 (that is 1 step 4 operators), or in 2 step by first applying action 5 and then action 6 (that is 2 step and 2 operators). In the image we omitted no-ops only to make it more readable (otherwise they should be there).



4. Yes, the heuristic is admissible. If during the forward step of GraphPlan we find the goal at depth n the plan contains at least n actions and therefore we are not overestimating the true cost. Notice how, in most of the cases, even if we find the goal at depth n the plan requires a larger number of actions. There are two reasons for this: (1) at each step we can take more then one action, (2) when we find the goal at step n we are not guaranteed to be able to backtrack and we might need to expand more.

5. **Extra Credit:** Consider the following operators and their effects:

	Precondition	Add	Delete
Op_1	D	B	
Op_2		C	A,B
Op_3	A	D	

Let's assume the initial state is {A,B} and the goal is {B,C}. If we use means-ends analysis we need to select a goal to achieve. Since B is already in the initial state, the only possible choice is C. To achieve C we select Op_3, we delete A,B and add D. The current state is therefore {C}. Since we are not in the goal state means-ends analysis select the next goal to achieve, B. To achieve B we can use Op_1 but we first need to achieve D. Therefore, means-ends analysis adds D as subgoal. In the current state there is no way to achieve D and means-ends analysis fail to find a solution.

4 Q-Learning

A robot moves deterministically in a world of 12 states laid out as shown in the Fig. 3. The robot can take four actions at each state, namely N, S, E and W. An action against a wall leaves the robot in the same state (note: not all walls are shown). Otherwise, the outcome of an action is deterministic (e.g. if a robot takes action N and it does not hit a wall, it will always end up in the state above it). The robot converged to the Q-table shown in Fig. 4, where we show only the maximum values for each state, as the other values (represented with a dash) do not matter in terms of the final policy.

1. The robot uses this learned Q-table to move in the world. Write the sequence of actions that the robot takes and states reached in three different episodes (of six actions each) starting in three different start states, namely S1, S6 and S7.

S_9	S_{10}	S_{11}	S_{12}
S_5	S_6	S_7	S_8
S_1	S_2	S_3	S_4

Figure 3: The robot world

	N	S	E	W
S1	60	-	60	-
S2	-	-	66	-
S3	-	-	73	-
S4	81	-	-	-
S5	66	-	-	-
S6	-	-	59	59
S7	-	66	-	-
S8	90	-	-	-
S9	-	-	73	-
S10	-	-	81	-
S11	-	-	90	-
S12	100	-	100	-

Figure 4: Q-table

Note: If there is more than one best action available, choose one randomly.

State	Action	State	Action	State	Action	State	Action	State	Action	State	Action	State
S1												
S6												
S7												

You are told that the discount factor for Q-learning is $\gamma = 0.9$. Based on this knowledge and the learned Q-table, answer the following questions:

2. A friend of yours, Chris, tells you that there is a single state s, such that $r(s, a) > 0$, for all actions a. What is that state and what is the reward?

3. Chris also tells you that the robot world has interior walls, besides the outside walls shown in the picture above. Where are those walls?

Note: As posted in Piazza, $Q(S1, N) = Q(S1, E) = 59$, not 60 as the Q-table says.

1.

State	Action	State	Action	State	Action	State	Action	State	Action	State	Action	State
S1	E	S2	E	S3	E	S4	N	S8	N	S12	N	S12
S6	W	S5	N	S9	E	S10	E	S11	E	S12	N	S12
S7	S	S3	E	S4	N	S8	N	S12	E	S12	N	S12

2. Notice that the Q-value of state S12 for the optimal actions does not depend on any other state, because the optimal actions are to go N or E both of which remain in state S12. Thus we will first find the optimal reward for S12:

$$Q(S12, N) = r(S12, N) + \gamma Q(S12, N) \implies r(S12, N) = 0.1Q(S12, N) = 10$$

Similarly

$$r(S12, E) = 10$$

Now we can compute the optimal reward for the states going into S12, such as S11:

$$Q(S11, E) = r(S11, E) + \gamma Q(S12, E) \implies r(S11, E) = 0.1Q(S12, E) - Q(S11, E) = 0$$

We can continue this procedure to find that the reward for the optimal action in every state except for S12 is 0. Thus S12 can be the only state with positive reward for every action, and the reward for the optimal actions for S12 is 10.

3. Based on the calculations above, we find that if there were no walls, and no state could give negative reward, then the optimal action(s) in each state would be to move closer to S12. Therefore, in any state where an action that should move us closer to S12 is not optimal, we know there must be a wall in the direction of that action. This means there is a wall between S2 and S6, between S6 and S10, between S7 and S11, and between S7 and S8.

5 8-Queens

1. For the chess board in Fig. 5, the red queens are attacking each other. Show two levels of the CSP local search using the min-conflict heuristic, justifying your choice of the queens to move.

2. Discuss the relationship between the MRV and min-conflict heuristics.

   (a) The min-conflicts heuristic involves two step: (1) randomly select a variable conflicting one or more constraints, (2) find a new assignment for the variable selected that minimize the number of conflicts.

   In this problem the variables are the 4 queens drawn in red and their value is the position on the board. First we randomly select the queen in H6. We now have to minimize its number of conflicts. It turns out that, for ever possible move, this queen has at least one conflict, we can therefore choose any cell with exactly one conflict. We decide to move this queen to A6.

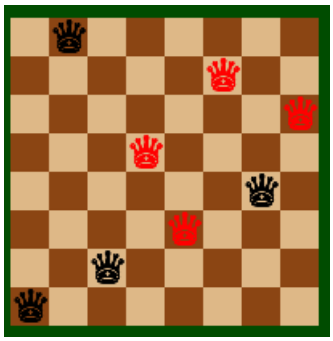


Figure 5: An 8-queen problem.

We can now move to the second level of the CSP but, before applying the min-conflict heuristic, we need to update the set of queens that violates the constraint (the one in E3 in now fine while the one in A1 has a conflict). As before we randomly select one of the five queens, the one in A1. We now look for the position that minimize its conflicts and we move it to H8 where it has only one conflict.

After two step of the CSP we have reduced the number of conflicts to 2 (the queen in D5 and the one in H8)

(b) There are several differences between MRV and min-conflicts but the two main points we are looking for are:

(i) MRV starts with an empty assignment and tries to build a consistent one. On the other hand min-conflicts starts with an inconsistent assignment and tries to fix problem as it goes.

(ii) When using MRV, if coupled with backtracking, we can always find a solution or conclude that the CSP has no solution. Min-conflict, due to it's random nature, cannot give us the same guarantees as MRV. On the other hand on specific class of problems (e.g. 8-queens) min-conflict has proven to be much faster then MRV.