1. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:ColumbiaX+CSMM.101x+3T2020+type@vertical+block@85c7645ab7c041bfa1a7f853de690f17?show_title=0&show_bookmark_button=0#transcript-end-a3acc4d4c8fc43328d41e5b48357c888)
2. So clearly, brute forcing, which is
3. generating all possible configurations,
4. and checking those for the constraints,
5. is not a good approach to CSPs.
6. Another example is called cryptarithmetic puzzles,
7. in which we have-- the problem is to find a substitution
8. of letters with digits.
9. There's ten digits, such that we have the arithmetic
10. sum here correct.
11. So the puzzle is this one here and we
12. want to find substitutions for T, W, O, F, U, and R.
13. So typically we could present this problem as a CSP
14. with the following representation as a hypergraph.
15. So why a hypergraph?
16. Because we have two different kinds of nodes.
17. We have the variable F, T, U, W, R,
18. O. We also have C1, C2, C3 that represent
19. the carryover of each column.
20. So O plus O is R, carryover C1.
21. C1 plus W plus W is U, carryover C2.
22. C2 plus C plus T is O, carryover C3.
23. C3 plus nothing is F. We also have
24. the constraint that no leading zeros are
25. allowed in the puzzle.
26. OK, so we have the variables are the circles.
27. We also have the squares that represent the constraints.
28. For example, we want all the digits to be different.
29. So we want all this instantiation
30. of digits to the variables to be distinct.
31. So, we're going to use a constraint called alldiff.
32. To spell that actually, we want all the variables
33. to be different.
34. So how can we formalize this problem as a CSP?
35. So first of all, the variables are straightforward.
36. We have the variables F, T, U, W, R, O, and we add to that--
37. to formalize the problem properly, we need to add C1,
38. C2, and C3--
39. the carryover digits.
40. The domain would be all digits between 0 and 9,
41. but then we have to be careful with the constraints.
42. So we want all the digits to be different.
43. So we're going evoke the constraint alldiff.
44. That's based on 1, 2, 3, 4, 5, 6 variables.
45. We call this n-ary constraint.
46. As it evolves, actually, six variables,
47. it's to be 6-ary constraint.
48. So this constraint is actually number one.
49. It's here.
50. All right.
51. Then we're going to spell that.
52. We don't want any leading 0.
53. So you want the T to be different than 0,
54. and you want the F to be different than 0.
55. And finally we're going to put together the columns
56. based on the carryover.
57. So we have that O plus O is equal to R, plus 10 times C1.
58. So remember, we have C1, C2, C3.
59. So O plus O is equal to R, plus 10 times C1.
60. C1 plus W plus W is U plus 10 times C2, et cetera.
61. So we have this--
62. second constraint is for the leading zeros.
63. The third, the fourth, the fifth and sixth
64. we'll put in relationship the carryover
65. with the different variables and use powers of 10, of course,
66. to represent the power of each of the columns.
67. So we're going to use the constraint for those.
68. So for C1, that's putting into relationship O and R and 10.
69. We're going to put a constraint C1 that links actually
70. the O here--
71. that's the contract number 3, would be this one--
72. that puts into relationship O, R, and C1.
73. We're going to put constraint four to put in relationship
74. C1, W, U, and C2.
75. C1, W, U, and C2, that's constraint number 4.
76. Constraint number 5 would be this one,
77. and constraint number 6 would this one, OK?
78. So this hypergraph shows all possible constraints
79. that we want to have in the CSP to solve it.
80. So solution for the CSP is something like 734 plus 734.
81. Let's see if it works.
82. 4 plus 4 is 8, 3 plus 3 is 6, 7 plus 7 is 14.
83. Right.
84. Then carry over 1.
85. So it's going to be we have an O, O, and O here.
86. So it's going to be 4, 4, and 4 are equal.
87. All digits are different and this actually
88. fulfills all the constraints that we have in the CSP.
89. So we see here that framing the CSP and formalizing properly
90. is really important in the solution.
91. So you could plug in this kind of problem,
92. for example, in CSP solver and you
93. would get the solution of the instantiation of the variables
94. with digits.
95. So how to solve CSPs, per se.
96. So first of all, unlike state-space search
97. that knows only one thing to do, which is search,
98. CSP algorithms can be based on two different techniques.
99. One of them is search, which means
100. that we need to choose a new variable assignment based
101. on the possibilities.
102. But the second one, we call it inference,
103. in which we are going to use a concept
104. called constraint propagation.
105. Constant propagation acts like spreading
106. the word of the constraints.
107. So we're going to use the constraint
108. to spread the word in this term that we are going
109. to choose the number of possibilities
110. of possible values for a variable.
111. By doing so, we're going to affect all the variables and so
112. on and so forth.
113. So we're going to spread the constraint
114. as much as we can among the variables
115. to reduce their domains.
116. So inference can be used as a preprocessing step.
117. So sometimes constraint propagation
118. can sometimes solve the full problem
119. without even doing a search.
120. But it can also be intertwined with search
121. in more difficult problems.
122. So we called for the search technique.
123. BFS develops the complete tree.
124. DFS is fine but is time consuming.
125. BTS is what we call Backtracking Search--
126. it's the basic uninformed search used in CSPs.
127. So it is essentially a DFS in which
128. we are going to assign one variable at a time
129. when we do assignments on the variables
130. So here the assignments are commutative,
131. which means that if we do WA equals red, NT equals green,
132. it's the same as if we do NT equals green and WA is red.
133. We also check the constraints on the go,
134. which means that we're going to consider the values that do not
135. conflict with previous assignments.
136. So for CSP, typically it's a DFS.
137. And it's a DFS with some backtracking search, which
138. means that we're going to step back
139. whenever we make an assignment that violates
140. the previous assignments.
141. More formally, we are going to start with an initial state
142. and it's the empty assignment, which means that we did not
143. assign any value to variables.
144. We use states that are partial assignments.
145. So it's like, for example, providing some variables
146. with digits in the cryptarithmetic puzzle.
147. We use a successor function that assigns a value
148. to an unassigned variable.
149. And, finally, we use a goal test to test
150. whether the current assignment is complete
151. and satisfies all constraints-- complete and consistent.
152. So going back to the map of Australia,
153. so this is how we will do a BTS or backtracking search.
154. I'm going to start with an empty map.
155. Remember we have three colors to color the map regions based
156. on the constraint that two adjacent regions don't
157. have the same color.
158. So initially we are going to have the possibility
159. to affect any color to the region on the left.
160. So I'm going to pick a variable--
161. this is this region here.
162. We're going to choose a color, which
163. are the possible values for that variable.
164. So it could be either red, green, or blue.
165. So I'm going to do a DFS.
166. So the next thing to explore would be the left hand side.
167. So I'm going to have two possibilities to color
168. this area here.
169. It's going to be either green or blue.
170. We can't do red, right?
171. So it's going to be the two only possibilities.
172. We do a DFS.
173. We keep going.
174. So we're going to next explore this area here.
175. We're going to color it either red or blue.
176. Green is not available for that area.
177. So whenever there is no possibility
178. or no possible values left to give to the variables,
179. we are going to backtrack and try something else.
180. This is exactly the essence of backtracking search for CSPs.
181. So just the first question, which variables
182. should be assigned next?
183. We are going to use the heuristic of the Minimum
184. Remaining Value, or MRV.
185. In this heuristic we use the concept
186. of choosing the variable with the fewest legal values
187. in its domain.
188. So we are going to start, in other words,
189. with the hardest variables to assign.
190. So if we consider the map of Australia again,
191. initially all the regions can take any color.
192. So it doesn't matter.
193. There is no hardest to pick at this point.
194. So I'm going to start, for example, by coloring
195. Western Australia with red.
196. Once I do that, I'm going to remove
197. this possibility of coloring the adjacent regions red.
198. So it's going to be not possible to color
199. this region or this region red.
200. So instead of doing this--
201. coloring and this one that has three possible values,
202. I'm going to start with this region
203. here because it has only two possible values to color.
204. So I'm going to do the northern territory in green.
205. Once I do that, I'm going to be left
206. with only one possible color to color southern Australia.
207. So it's going to be only blue left, which
208. means that I would prefer to start coloring South Australia
209. rather than doing Queensland.
210. Queensland has two possible choices,
211. which would be blue and red.
212. But South Australia would only have a blue.
213. So it's a hard region to color.
214. So I'm going to start with it and get rid of it
215. to avoid the problems of not being able to assign
216. a color afterwards to that.
217. So with this principle we're going to--
218. with this heuristic we're going to be
219. able to start with the hardest to make sure
220. that these are served first before we go to the easiest
221. variable to color or to assign.
222. To answer the second question, in what order should
223. its value be tried?
224. We're going to use the heuristic, or the principle,
225. of the Least Constraining Values, or LCV.
226. Given a variable, we need to choose
227. the least constraining value--
228. the one that rules out the fewest values in the remaining
229. variables.
230. So in the example of Australia, we're
231. going to suppose we colored this first region red,
232. the second region green.
233. Then we need to color this region here
234. in either the colors red or blue.
235. Should we choose red or should we choose blue?
236. So we're going to check the neighbors, check the variables,
237. and see whether we are left with enough remaining variables
238. or values to color afterwards.
239. So for example, if we pick the value red for Queensland,
240. we're going to leave at least SA (Southern Australia) with one
241. possibility to be colored as blue.
242. However, if we pick blue, and knowing
243. that South Australia is connected to the three regions
244. here--
245. it's adjacent to the three regions--
246. then we are going to leave no chance for Southern Australia
247. to be colored.
248. We are going to have zero possible values
249. for Southern Australia.
250. So I'm going to avoid that and go
251. with the colors that actually allow to color the remaining
252. regions properly.
253. So the principle here is to pick the ones that
254. are likely to work, not only for the variable that we assigning,
255. but likely to work for the remaining variables.
256. To address the third and final question,
257. can we detect inevitable failure early?
258. We are going to use the heuristic of forward checking.
259. In forward checking, we are going
260. to keep track of the remaining legal values
261. of the unassigned variables.
262. This actually will terminate the process
263. when any variable has no legal value possible.
264. So in the Australia map, we're going
265. to start with three colors, which are the domain of all
266. the variables.
267. So we can color Western Australia
268. in red, green, and blue.
269. And same thing for NT, Q, et cetera.
270. So after we assign a color, suppose we assign
271. red to Western Australia.
272. So it's going to be--
273. we're going to color this red, which
274. means that we're going to have only assigned
275. the value to the variable.
276. It's red and that's it.
277. So if it's red then we are going to affect the domain
278. of the remaining variables.
279. So for example, Northern Territory
280. and Southern Australia can't take red anymore.
281. So I'm going to take away the red
282. from these two domains of SA.
283. So we're going to take away these two colors.
284. We're going to color next--
285. suppose we color Queensland in green.
286. So I'm going to put green in the domain of Queensland.
287. By doing so, we are going to remove
288. the possibility of coloring green all the adjacent regions,
289. which will be NT.
290. So we're going to remove green.
291. And SA-- we're going to remove green, and also in New South
292. Wales, removing green here.
293. OK, we keep going.
294. So suppose we are going to color Victoria blue.
295. If we color Victoria blue, we are actually
296. going to remove all hope to color Southern Australia blue
297. because this was the last possible color to do that.
298. So if we do that, we are going to remove
299. the possibility of coloring New South Wales blue,
300. the possibility of coloring South Australia blue.
301. And actually, there is no possible legal value
302. for southern Australia.
303. So are going to actually--
304. we are checking here, forward checking
305. that we are not going ahead with an inevitable failure.
306. If we do this process, then we are
307. going to fail because there was no possible color
308. to color SA anymore.
309. So we're going to not do that.
310. So we're going to do this forward checking
311. and, if we hit a wall, if we hit that we are not
312. given a chance to SA to be colored,
313. then we're going not to do that.
314. This is called forward checking.
315. [End of transcript. Skip to the start.](https://courses.edx.org/xblock/block-v1:ColumbiaX+CSMM.101x+3T2020+type@vertical+block@85c7645ab7c041bfa1a7f853de690f17?show_title=0&show_bookmark_button=0#transcript-start-a3acc4d4c8fc43328d41e5b48357c888)