1. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:ColumbiaX+CSMM.101x+3T2020+type@vertical+block@3bc3351f710648b7a0710a04a5c8d72d?show_title=0&show_bookmark_button=0#transcript-end-36d7b7ca4af14a6f80f1070febbb55eb)
2. So here is the backtracking search algorithm
3. to solve a CSP.
4. So this algorithm will receive as input a CSP formalization,
5. which means the domain, the variables, and the constraints,
6. and will return either solution or failure.
7. This algorithm would call a function-- a recursive function
8. called backtrack.
9. Initially with the empty assignment
10. because we don't have yet assigned
11. any value to variables, and the CSP formalization.
12. All right, so the backtrack function
13. is a recursive, simple recursive function, which it calls itself
14. at this point here.
15. The function actually will choose one variable
16. at a time in this line here.
17. So it's going to be-- pick a variable.
18. And we'll try all possible values for that,
19. and assign variables.
20. So for each value in the domain of this value,
21. we are going to try them, and see if we can
22. find consistent assignments.
23. If an inconsistent assignment is found then
24. we're going to try all the values and the algorithm
25. backtracks after removing the value from the variable value
26. assignments, from the assignment.
27. So in other words, if the assignment
28. is complete and is consistent we are
29. going to return the assignment.
30. Otherwise, we're going to pick a variable, pick a value,
31. try it as a possible assignment.
32. If we need to fit failure, we're going
33. to backtrack, and try other values.
34. And so on and so forth.
35. So this is a simple, recursive function.
36. Note here that we are using a general statement for picking
37. the variables, such as select unassigned variables,
38. and order the domain value.
39. These are kept general on purpose
40. to implement the heuristics we have
41. seen in this lecture or other heuristics.
42. For example, we could use the minimum remaining
43. values to pick a variable, or the LCV,
44. the least constraining values to pick
45. the values of the domain.
46. That's pretty much it.
47. This is the function.
48. This is doing the search with BTS,
49. and it implements the heuristics we have seen in class.
50. A very popular CSP problem or CSP puzzle is called Sudoku.
51. So many of you probably have played Sudoku before,
52. or have seen people playing Sudoku in the train.
53. This originally puzzles are printed
54. on newspapers and journals, and the goal of Sudoku
55. is as follows.
56. So you have grid of 9 by 9 elements.
57. And you want to fill this screen with numbers, or with digits
58. between one and nine
59. so you don't have repetitions in this element
60. and this are actually three by three boxes or rows, columns,
61. or rows.
62. So you don't want the digits one to nine to be repeated.
63. And you want all of them to appear in each of these--
64. let's call these units.
65. OK, so that's the idea of the puzzle.
66. So I need to solve it by filling the grid with digits.
67. Let's formulize this problem as a CSP.
68. So we need to first to define what are the variables.
69. To make it easy for us, we're going
70. to give a name to the rows and the columns.
71. For example, this would be row A, row B,
72. until row I. We are going to give numbers to the columns.
73. column one, two, three, up to nine.
74. So we are defining variables for each of the cells,
75. or each of the components of the grid of nine by nine.
76. Which means that we're going to have
77. nine times nine equals 81 possible variables,
78. defining the Sudoku as CSP.
79. OK so the variables would be the set V, A1, A9, B1, B9, I1, I9.
80. How about the domain?
81. The domain has to be all that is between one and nine.
82. And so there is a typo here.
83. Zero is not part of the domain.
84. So the domain is one through nine
85. for all the cells that are empty.
86. For the cells that are fixed-- just 3, 7, and 6--
87. these have a specific value.
88. So the domain is simply one single viable.
89. So the domain for example for this variable here,
90. which is A8, the domain of A8, has a domain of what?
91. Of one single variable, which is the value 3.
92. So how about the constraints? We are going to express the constraint
93. that says that all three by three boxes, rows, and columns
94. have different values between one and nine.
95. So for example, first we're going
96. to focus on the first row.
97. The first row is the row A1, A2, A3, through A9.
98. And you want all these values to be different.
99. To express that we are going to use the constraint,
100. which is Alldiff,
101. that tells that A1 through A9 are different.
102. OK, so this constraint would be for this row here.
103. We're going to do that for the nine rows, right?
104. So it's going to be nine possible Alldiff
105. constraints for the rows.
106. We do the same for the column.
107. So for example, for this column here
108. it has to be that A1, B1, through I1 are different.
109. So express that as this constraint here.
110. This set of constraints will be for the rows.
111. And finally for three-by-three boxes.
112. For example, this box here.
113. We need that A1, A2, A3, B1, B2, B3, C1, C2,
114. C3 all be different.
115. So this is going to be expressing that for this box
116. we want all these values different.
117. By this specification, we have defined the CSP
118. with variables, domains and constraints.
119. So how do Sudoku players solve the puzzle?
120. So the idea is to look at the cells,
121. and try to fill them little by little.
122. So for example, if you focus on this first draw here,
123. we have two missing values.
124. This cell and this cell.
125. What we do in general is to look at the possible values,
126. since we have all these digits, we are left with the digits
127. two and four to put, right?
128. So this can take either two or four,
129. and this will take either two or four.
130. We're going to use the constraint of all three-
131. by-three boxes: must contain one to nine.
132. And if we look at this box here we
133. see that actually number two is already taken
134. by this cell here, which is B1.
135. So this value, too, can be put in this cell here.
136. So this will actually help us remove the possible values
137. from the cell here, so this I can only take the value for.
138. Which remains with this cell here
139. with only the possible value of two,
140. so this cell would be actually two.
141. So then given these two values in our set now,
142. we're going to look into the values here.
143. We're going to try to guess what to put in here and in here.
144. So here we know that the remaining values to be put here
145. are actually-- we are missing three and we are missing five.
146. However, three is already in this row.
147. So this can't be a possibility for this cell here.
148. So the only value that's possible at this point is five.
149. And we're going to add three here,
150. because we need to have all the digits between one and nine
151. in this three by three box.
152. So this how Sudoku's are being solved.
153. They're going to add digits one at a time
154. by looking into the environment, whether it's
155. within the same roll, the same column, or the same block,
156. to see what are the permissible assignments
157. of digits to the variables.
158. So how do humans solve Sudoku in general?
159. So the idea is to try to fill the cells one by one
160. by using those constraints, propagating
161. the information among the cells so as
162. to fill only the permissible values for the variables.
163. So for example, if you focus on this first row
164. here we see that we're missing two values.
165. These two values here.
166. The two possible values can be either a two or a four.
167. Just because we have the values one, three, six, seven, five,
168. eight, nine.
169. And we only can fill the values, the missing values--
170. the missing variables with the missing two digits left.
171. OK, so we have either the possibility
172. to fill this value here with a two or a four.
173. And same thing here, we can either
174. fill with a two or a four.
175. But then we're going to use our constraint that
176. says that, basically a three-by-three box
177. can only have this between one and nine,
178. and there is no repetition.
179. So for example, if we look at this box here--
180. the values here are one, two, six, eight, nine, and seven.
181. Which means that we have already a two.
182. The two cannot be a solution for this cell here.
183. So the only remaining value possible for the cell A2
184. is the value four.
185. This actually would reduce the domain of the cell A3, four,
186. five, five--
187. A5 to only two.
188. Because actually we already have the four in this row here.
189. So we use the information that the values in the row
190. are different, but the values also in this three-by-three box
191. must be different.
192. So we can keep going.
193. So for example, if you want to fill this value and that value
194. here, we know that we are missing the value three,
195. and the value five.
196. And fortunately we have a three here.
197. This cell here cannot be three.
198. The only possible value left is five,
199. and we are left with the three in this box here.
200. So good, keep going with this reasoning
201. and refill most of the cells, if not all.
202. So it depends a lot on the complexity
203. or the difficulty level of the Sudoku.
204. And some simple ones can just use these kinds of techniques
205. to propagate information among the cells.
206. This is called, typically, constraint propagation.
207. Propagating the information among the cells--
208. so you're propagating the information among the cells
209. to fill them with the digits.
210. But in some fancy Sudokus, or difficult Sudokus, it's
211. not enough to use these kinds of constraints.
212. And Sudoku players and lovers like
213. to use advanced techniques, such that for example using
214. the naked double, or triple, the locked pair, the locked triple,
215. etc.
216. And the idea of this is simply to make inference
217. on the possible values of the cells.
218. For example, for naked double we need to find, in a three-
219. by-three grid, two cells that have two possible remaining
220. values.
221. This would actually help eliminate these two values
222. from all the possible assignments in that same box.
223. So by doing so I'm going to use an advanced thinking,
224. or an advanced inference of the possible values for the cells.
225. And these are typically some of the techniques used by Sudoku
226. to solve the puzzle.
227. So just for fun here's the solution of this Sudoku.
228. So something from this Sudoku configuration,
229. we need to fill it with these values,
230. and none of these assignments of values to variables
231. actually violates the constraints of the Sudoku.