1. [Start of transcript. Skip to the end.](https://courses.edx.org/xblock/block-v1:ColumbiaX+CSMM.101x+3T2020+type@vertical+block@182e0d2027fa49f9bef9779cfe40de19?show_title=0&show_bookmark_button=0#transcript-end-c4cf43c42a4f44c8b20fa65729dfea00)
2. So solving a Sudoku leverages a concept
3. called constraint propagation.
4. But this seems something like we have already seen before.
5. Remember, forward checking is also something-- or technique--
6. to propagate the information from assigned
7. to unassigned variables.
8. So how is it different?
9. Observe, for example, in the map of Australia.
10. Forward checking helps us improve the backtracking
11. search.
12. But it checks, actually, the interaction
13. between assigned and unassigned variables, but does not
14. check the interaction between unassigned variables
15. themselves.
16. So for example, here, SA and NT have only one possible value
17. in their domain, which is blue--
18. for SA and blue for NT.
19. And both of them cannot be blue.
20. Because if they are blue, they are adjustment regions.
21. They must not be blue, because of the constraints.
22. So forward checking, while it allows
23. us to improve backtracking search,
24. it does not really look very far in the future.
25. So it doesn't detect, in other words, inevitable failures
26. early enough.
27. So, in this case, we use a more general concept
28. called constraint propagation, which is reasoning
29. from constraint to constraint.
30. And there are different ways of doing this kind of constraint
31. propagation.
32. And one of them is called arc consistency test.
33. Constraint propagation aims to make the graph, or the CSP
34. problem, consistent.
35. But there are different kinds of consistency.
36. It could be a different level.
37. So, for example, if we consider unary constraints,
38. we talk about what we call node-consistency.
39. In which case, we want each variable in the CSP
40. to be node-consistent.
41. In other words, a variable Xi is called node-consistent if all
42. the values of its domain-- domain of Xi--
43. satisfy all unary constraints.
44. For example, if the Tasmanian people
45. don't like the color green, and we
46. want Tasmania to be absolutely not green,
47. we're going to add the unary constraint
48. T different than green, and use that constraint
49. to make the node Tasmania consistent.
50. So we talk about node-consistency in CSP if each
51. single node in the CSP graph is node-consistent,
52. which means that it cannot take any, for example,
53. forbidden value.
54. Another kind of consistency is called arc-consistency.
55. This is the consistency between edges.
56. When we talk specifically about what
57. we call binary constraints--
58. so we have a constraint between X and Y
59. and we have the edge on the graph--
60. on the constraint graph--
61. X implies Y. So X implies Y is arc-consistent if and only
62. if every value x of X here is consistent with some value
63. of Y. Means that--
64. means that any value we take from this X,
65. we would find a consistent value for Y on the other side.
66. So we call these binary constraints arc-consistency.
67. Which means that to make a graph arc-consistent,
68. we need to check all possible edges
69. for their arc-consistency status.
70. And if they are all arc-consistent,
71. we talk-- we call the problem arc-consistent.
72. Finally, path-consistency.
73. That acts on n-ary constraints--
74. actually, is a generalization of arc-consistency from binary
75. to multiple constraints.
76. And, in general, it's possible to transform
77. any n-ary constraints problem into a binary constraint
78. problem.
79. So just duplicate, you know-- just
80. flatten this n-ary constraint into as many binary constraints
81. as needed.
82. And, often, CSP solvers are designed
83. to only work with binary constraints,
84. assuming that all path-consistency constraints
85. are actually transferable to, again, binary constructs.
86. So let's dive into arc consistency.
87. Arc consistency, or AC to make it simple,
88. is the simplest form of propagation
89. that makes an arc consistent.
90. Again in arc-consistent, X implies Y--
91. is consistent if and only if for every value of X--
92. every value in the domain of Xi--
93. has some allowed value in Y. Right?
94. So this is the purpose of baye and arc-consistent.
95. For example, if we think of the map of Australia again,
96. and we have SA implies NSW or there
97. is an edge between S and NSW, because these two
98. areas are actually adjacent.
99. So that's-- SA is here.
100. That's Southern Australia.
101. And NSW is New South Wales.
102. Right?
103. So there is an edge because there
104. is a constraint of adjacency between these two regions.
105. So if you want to make this arc consistent,
106. we need to check all possible values for--
107. for every possible value for SA, which is just blue,
108. there must be an allowable value in NSW.
109. And here it's possible if we affect the blue to SA,
110. we-- it's possible to affect the red to NSW
111. and remain consistent-- and the CSP remaining consistent.
112. If you look at the orientation NSW implies SA--
113. so in this configuration, we'll have
114. a value for NSW, which is blue, for which there
115. is no allowable value in SA.
116. So which means that this is actually
117. not a possible assignment, because it
118. violates the arc-consistency.
119. We are going to have--
120. if we affect blue to NSW, it would not
121. be possible to have-- to affect any color to SA.
122. So, in this case, we're going to remove NSW--
123. the color-- from the NSW domain, the color blue
124. to make the arc arc-consistent.
125. So we could keep going.
126. For example, from V to NSW.
127. We have that for the red color, it's
128. not possible to affect any variable for NSW.
129. So I'm going to remove the value red from V.
130. And we could go on with, you know,
131. checking all the consistencies between the arcs.
132. For SA implies NT, for example, we have actually blue and blue,
133. so it's not possible to affect blue to both regions.
134. And we are going to move the value from SA--
135. from X. Because it doesn't have an allowable value
136. Y. So this is how we check all possible arrows, or edges,
137. between the different variables.
138. And we eliminate all the possible values that actually
139. violate the constraint.
140. This is how we propagate information
141. among the assigned and unassigned variables
142. altogether.
143. A well-known algorithm to make a CSV arc-consistent
144. is called AC3.
145. In AC3, we expect a CSP formalization
146. that's actually a binary one.
147. So we expect a binary CSP with components, variables, domain,
148. and constraints.
149. We're going to use in CSP a local variable called queue.
150. This function AC3 actually returns either true or false--
151. either failure or success.
152. It returns failure if there was no consistent assignment, which
153. means that some domains of variables are empty.
154. Or, otherwise, it returns true, which
155. means that all the arcs in the CSP were made consistent.
156. So let's simulate how it works.
157. So in this queue we are going to put initially all the arcs, right,
158. that are in the CSP.
159. They're going to pick the arcs one by one
160. and check for their consistency.
161. So while the queue is not empty, we are going to pick one--
162. remove the first element-- remove one element,
163. Xi implies Xj, so it could be--
164. so we have the arc Xi implies Xj.
165. Remember to check arc-consistency,
166. we need to check that for all values in Xi,
167. there is a permissible or allowed value in Xj.
168. So if there is none, we need to revise domain I, which
169. is a domain of Xi.
170. Right?
171. So if we revise the CSP based on the Xi implies Xj,
172. then remember-- so we are going to revise
173. by removing the values.
174. Remember the example of the Australia map.
175. So revising the CSP means that you are going
176. to make the domain smaller.
177. So if, for each value X in Di--
178. in the domain Xi-- we are going to check for each value.
179. If there is no value Y in Dj, that actually
180. satisfies the constraints,
181. then you are going to remove X--
182. the value X will be removed from the domain Di.
183. And if we revise, we are going to return
184. a value true for revised.
185. OK.
186. So if, after checking the consistency of Xi implies Xj,
187. we end up revising the domain of Xi.
188. Then we are going to check if the domain is empty,
189. which means that the size of the I is zero.
190. There is no possible remaining value
191. to affect-- to assign to Xi, then we can't solve the problem
192. and we're going to return false.
193. This function AC3 will return false,
194. which means that it's not possible to make this arc
195. arc-consistent, unless we remove all the values.
196. And the CSP is actually not solvable.
197. If the domain is not empty, that means
198. that there is some remaining value in the domain Di,
199. but we have removed some.
200. And we're going to need to check all the other arcs that
201. are actually Xk implies Xi.
202. We're going to check all the neighbors of Xi,
203. because removing a value from the domain of Xi
204. might affect the domain of the other values
205. or the other variables that are connected to Xi.
206. So for each Xk (neighbor of Xi), we are
207. going to add the arc Xk, Xi to the queue
208. to be checked, actually, for arc-consistency as well.
209. So, again, the reason is that domain Xi has been revised.
210. We removed some values from it.
211. If this is the case, then we need to check all the neighbors
212. Xk for arc-consistency as well.
213. OK?
214. So I'm going to keep going doing that until the queue is
215. actually empty.
216. We have checked for arc-consistency
217. all of the edges in the CSP.
218. So if we end up being--
219. finding some domains that are 0, then we're
220. going to return false.
221. Again the CSP is not solvable.
222. If not, then we have made all the arcs
223. in the CSP arc-consistent.
224. So, now how about the complexity of the algorithm AC3?
225. So suppose we have n-variables.
226. And we have d, the domain size.
227. If every node or every variable is connected
228. to every other node in the CSP, then we
229. have n times n minus 1 possible arcs or constraints.
230. Remember that AC3 will check for all nodes--
231. for all arcs in the CSP.
232. So there's going to be an order of n-squared--
233. of checking of the arcs.
234. Each arc can be inserted in the queue d times.
235. So it was going to be because we have
236. d possible values for variable Xi.
237. So it's going to be in order of d.
238. And finally checking the consistency
239. of an arc, which costs, by itself, d-squared,
240. because we have two variables, Xi implies Xj.
241. And we need to check that all values on the Xi domain
242. have an allowable value in the Xj domain.
243. So we need to do that in O d-squared.
244. We put this together to give us the complexity--
245. an overall complexity of n-squared times d-cubed.
246. Now recall from the previous slides
247. that it's possible to solve a CSP to do either search
248. or inference to constraint propagation.
249. Or do both, which means that it can do search intertwined
250. with inference.
251. Constraint satisfaction is a kind of inference.
252. And search is a kind of assignment
253. of one variable at a time and search
254. through the possible assignments to find a consistent one.
255. OK?
256. So recall this is the algorithm-- backtracking
257. search algorithm we have seen.
258. But they're going to include the new piece, which
259. is the inference.
260. Intertwine the inference and the search
261. to make the search faster and more effective.
262. So we're going to actually include
263. only one piece, which is make the inference at this level
264. here, of the algorithm.
265. And then, also, handle the inferences
266. that are actually leading to failure at this level.
267. So this is exactly the backtracking search algorithm.
268. But after we pick a value for a variable,
269. we are going to check whether we can make any constraint
270. propagation through arc consistency or the techniques.
271. So if we do inferences, and we find
272. some inferences-- means that inferences does not--
273. do not lead to failure,
274. then we are going to add our inferences to the assignment.
275. And then we call again backtrack, just
276. like in regular BTS assignment.
277. If we find a failure, it means that there
278. is no consistent assignment.
279. We are going to remove both the variable value.
280. This is a part of the regular BTS algorithm.
281. But you're going, also, to remove all the inferences that
282. come from this section here.
283. OK?
284. So we are going to expand the BTS algorithm by intertwining
285. the inference that could be done in different ways,
286. including arc consistency.
287. And whenever we do some changes, we
288. are going to include that into the assignment of the problem
289. to be solved.
290. [End of transcrip](https://courses.edx.org/xblock/block-v1:ColumbiaX+CSMM.101x+3T2020+type@vertical+block@182e0d2027fa49f9bef9779cfe40de19?show_title=0&show_bookmark_button=0#transcript-start-c4cf43c42a4f44c8b20fa65729dfea00)