

9/15/09 ① Constraint Satisfaction 1

①

Ques

First: Where are we?

- we've finished the "Search" section, i.e. we've learned many ways to define problem spaces, and different algorithms to find solns in those prob spaces

- Uninformed Search: BFS, DFS, UCS, DLS, ID DFS, BS

- Informed Search: Heuristics, Greedy BFS, A*, RBFS, IDA*, SMA*

- Local Search: HC, stochastic HC, SA, LBS, GA

- Did not cover local search in cont. spaces, but will later

- How would you use these methods in your own area?

Discussion

3/8

Main idea from before: problem solving is defined by searching in a space of states.

Standard search prob.

Constraint Satisfaction Problem (CSP)

CSP problems are defined by the constraints.

- set of variables X_i
- each variable can be assigned values V_i
- constraint

variables
domain of possible values
~~values~~
constraint

CSI ②

②

Example Map Coloring:

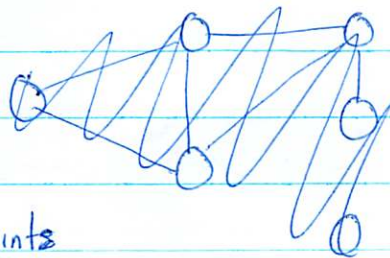
3/2



~~What~~ Problem: Color each region with R, G, or B such that no neighboring region has the same color.

How do we solve this?

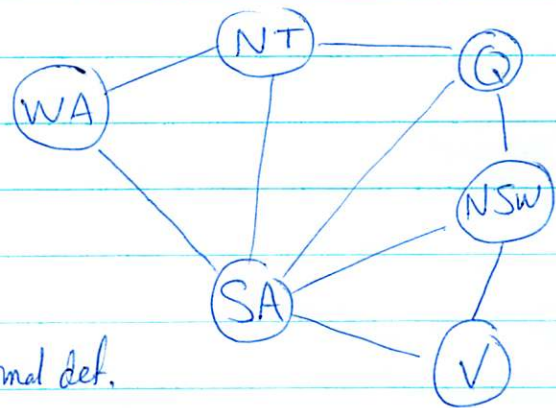
First how do we represent the problem?



Constraints

No two connected nodes can have same color.

→ (*) Formal def.



Postpone

Could solve by ~~to~~ local search methods

- initial state : no variable assignments
- successor fn : assign value to any unassigned variable provided it doesn't conflict
- Goal test : have all variables been assigned?
- ~~no path cost~~

Ⓣ

CS1 (3)

[?] What are the properties of the soln?

- all variables assigned
- satisfies all constraints \leftarrow given by successor func.

\Rightarrow Sdms have depth n

② does path matter? No only interested in soln.

How many variable assignments? ~~13~~ ~~14~~ ~~15~~ ~~16~~ ~~17~~ ~~18~~ ~~19~~ ~~20~~ ~~21~~ ~~22~~ ~~23~~ ~~24~~ ~~25~~ ~~26~~ ~~27~~ ~~28~~ ~~29~~ ~~30~~ ~~31~~ ~~32~~ ~~33~~ ~~34~~ ~~35~~ ~~36~~ ~~37~~ ~~38~~ ~~39~~ ~~40~~ ~~41~~ ~~42~~ ~~43~~ ~~44~~ ~~45~~ ~~46~~ ~~47~~ ~~48~~ ~~49~~ ~~50~~ ~~51~~ ~~52~~ ~~53~~ ~~54~~ ~~55~~ ~~56~~ ~~57~~ ~~58~~ ~~59~~ ~~60~~ ~~61~~ ~~62~~ ~~63~~ ~~64~~ ~~65~~ ~~66~~ ~~67~~ ~~68~~ ~~69~~ ~~70~~ ~~71~~ ~~72~~ ~~73~~ ~~74~~ ~~75~~ ~~76~~ ~~77~~ ~~78~~ ~~79~~ ~~80~~ ~~81~~ ~~82~~ ~~83~~ ~~84~~ ~~85~~ ~~86~~ ~~87~~ ~~88~~ ~~89~~ ~~90~~ ~~91~~ ~~92~~ ~~93~~ ~~94~~ ~~95~~ ~~96~~ ~~97~~ ~~98~~ ~~99~~ ~~100~~ ~~101~~ ~~102~~ ~~103~~ ~~104~~ ~~105~~ ~~106~~ ~~107~~ ~~108~~ ~~109~~ ~~110~~ ~~111~~ ~~112~~ ~~113~~ ~~114~~ ~~115~~ ~~116~~ ~~117~~ ~~118~~ ~~119~~ ~~120~~ ~~121~~ ~~122~~ ~~123~~ ~~124~~ ~~125~~ ~~126~~ ~~127~~ ~~128~~ ~~129~~ ~~130~~ ~~131~~ ~~132~~ ~~133~~ ~~134~~ ~~135~~ ~~136~~ ~~137~~ ~~138~~ ~~139~~ ~~140~~ ~~141~~ ~~142~~ ~~143~~ ~~144~~ ~~145~~ ~~146~~ ~~147~~ ~~148~~ ~~149~~ ~~150~~ ~~151~~ ~~152~~ ~~153~~ ~~154~~ ~~155~~ ~~156~~ ~~157~~ ~~158~~ ~~159~~ ~~160~~ ~~161~~ ~~162~~ ~~163~~ ~~164~~ ~~165~~ ~~166~~ ~~167~~ ~~168~~ ~~169~~ ~~170~~ ~~171~~ ~~172~~ ~~173~~ ~~174~~ ~~175~~ ~~176~~ ~~177~~ ~~178~~ ~~179~~ ~~180~~ ~~181~~ ~~182~~ ~~183~~ ~~184~~ ~~185~~ ~~186~~ ~~187~~ ~~188~~ ~~189~~ ~~190~~ ~~191~~ ~~192~~ ~~193~~ ~~194~~ ~~195~~ ~~196~~ ~~197~~ ~~198~~ ~~199~~ ~~200~~ ~~201~~ ~~202~~ ~~203~~ ~~204~~ ~~205~~ ~~206~~ ~~207~~ ~~208~~ ~~209~~ ~~210~~ ~~211~~ ~~212~~ ~~213~~ ~~214~~ ~~215~~ ~~216~~ ~~217~~ ~~218~~ ~~219~~ ~~220~~ ~~221~~ ~~222~~ ~~223~~ ~~224~~ ~~225~~ ~~226~~ ~~227~~ ~~228~~ ~~229~~ ~~230~~ ~~231~~ ~~232~~ ~~233~~ ~~234~~ ~~235~~ ~~236~~ ~~237~~ ~~238~~ ~~239~~ ~~240~~ ~~241~~ ~~242~~ ~~243~~ ~~244~~ ~~245~~ ~~246~~ ~~247~~ ~~248~~ ~~249~~ ~~250~~ ~~251~~ ~~252~~ ~~253~~ ~~254~~ ~~255~~ ~~256~~ ~~257~~ ~~258~~ ~~259~~ ~~260~~ ~~261~~ ~~262~~ ~~263~~ ~~264~~ ~~265~~ ~~266~~ ~~267~~ ~~268~~ ~~269~~ ~~270~~ ~~271~~ ~~272~~ ~~273~~ ~~274~~ ~~275~~ ~~276~~ ~~277~~ ~~278~~ ~~279~~ ~~280~~ ~~281~~ ~~282~~ ~~283~~ ~~284~~ ~~285~~ ~~286~~ ~~287~~ ~~288~~ ~~289~~ ~~290~~ ~~291~~ ~~292~~ ~~293~~ ~~294~~ ~~295~~ ~~296~~ ~~297~~ ~~298~~ ~~299~~ ~~300~~ ~~301~~ ~~302~~ ~~303~~ ~~304~~ ~~305~~ ~~306~~ ~~307~~ ~~308~~ ~~309~~ ~~310~~ ~~311~~ ~~312~~ ~~313~~ ~~314~~ ~~315~~ ~~316~~ ~~317~~ ~~318~~ ~~319~~ ~~320~~ ~~321~~ ~~322~~ ~~323~~ ~~324~~ ~~325~~ ~~326~~ ~~327~~ ~~328~~ ~~329~~ ~~330~~ ~~331~~ ~~332~~ ~~333~~ ~~334~~ ~~335~~ ~~336~~ ~~337~~ ~~338~~ ~~339~~ ~~340~~ ~~341~~ ~~342~~ ~~343~~ ~~344~~ ~~345~~ ~~346~~ ~~347~~ ~~348~~ ~~349~~ ~~350~~ ~~351~~ ~~352~~ ~~353~~ ~~354~~ ~~355~~ ~~356~~ ~~357~~ ~~358~~ ~~359~~ ~~360~~ ~~361~~ ~~362~~ ~~363~~ ~~364~~ ~~365~~ ~~366~~ ~~367~~ ~~368~~ ~~369~~ ~~370~~ ~~371~~ ~~372~~ ~~373~~ ~~374~~ ~~375~~ ~~376~~ ~~377~~ ~~378~~ ~~379~~ ~~380~~ ~~381~~ ~~382~~ ~~383~~ ~~384~~ ~~385~~ ~~386~~ ~~387~~ ~~388~~ ~~389~~ ~~390~~ ~~391~~ ~~392~~ ~~393~~ ~~394~~ ~~395~~ ~~396~~ ~~397~~ ~~398~~ ~~399~~ ~~400~~ ~~401~~ ~~402~~ ~~403~~ ~~404~~ ~~405~~ ~~406~~ ~~407~~ ~~408~~ ~~409~~ ~~410~~ ~~411~~ ~~412~~ ~~413~~ ~~414~~ ~~415~~ ~~416~~ ~~417~~ ~~418~~ ~~419~~ ~~420~~ ~~421~~ ~~422~~ ~~423~~ ~~424~~ ~~425~~ ~~426~~ ~~427~~ ~~428~~ ~~429~~ ~~430~~ ~~431~~ ~~432~~ ~~433~~ ~~434~~ ~~435~~ ~~436~~ ~~437~~ ~~438~~ ~~439~~ ~~440~~ ~~441~~ ~~442~~ ~~443~~ ~~444~~ ~~445~~ ~~446~~ ~~447~~ ~~448~~ ~~449~~ ~~450~~ ~~451~~ ~~452~~ ~~453~~ ~~454~~ ~~455~~ ~~456~~ ~~457~~ ~~458~~ ~~459~~ ~~460~~ ~~461~~ ~~462~~ ~~463~~ ~~464~~ ~~465~~ ~~466~~ ~~467~~ ~~468~~ ~~469~~ ~~470~~ ~~471~~ ~~472~~ ~~473~~ ~~474~~ ~~475~~ ~~476~~

$$(\# \text{ values})^{\# \text{ variables}} = d^n$$

9/5 (*) Formal definition of CSPs

$$V = \{V_1, V_2, \dots, V_n\}$$

Values

$$D = \{D_1, D_2, \dots, D_d\}$$

Domain

$$C = \{ \{ \text{pair} \}, \{ \text{legal values} \} \}$$

$\{ (WA, NT), \{ (R, G), (R, B), (G, R), (G, B), (B, R), (B, G) \} \}$
 represent by function, not explicitly

A CSP is a triplet $\{V, D, C\}$

→ General problem statement for large class of CSPs

CS1 ④ ³ How do we solve this?

How about search:

- initial state:

no vars assigned

- successor fn

What should this do?

→ Assign value to any unassigned var provided it satisfies constraints

- goal test

All vars assigned. Why?

Only interested in the soln, not the path.

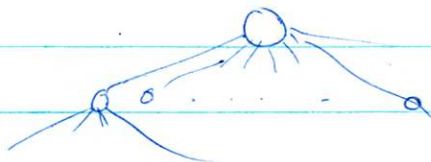
[?] what is the depth of the search tree?

depth n because there are n variables, when they are all assigned, we're done.

[?] How many possible assignments? Or, how big is the tree?

$$(\#vals)^{\#vars} = O(d^n) \Rightarrow \text{still exponential in general}$$

Which search method? ~~BFS~~ BFS? what's the branching factor



each of n vars can be assigned d vals
 $\Rightarrow n \times d$ not good.

Therefore:

generate successors
by considering only
a single variable
at each node.

Why does this happen?
Commutativity
order of "actions"
doesn't matter.
In tree we reach many
of the same partial
assignments

next level $(n-1) \times d$
 $\Rightarrow n! d^n$ for whole tree
Obviously bad. Only d^n
possible assignments

SA = R, G, or B not SA = R & WA = blue

CS1 ⑤ 12 What about DFS?

Called Backtracking Search:

- choose vals for one var at a time
- back track when var has no legal vals left to assign

func RBSearch(csp) returns soln or failure
return RB($\{\}$, csp)

α = assignment

func RB(~~assign~~ α , csp) returns soln or failure

if α complete return α

var \leftarrow selectUnassignedVar(Vars(csp), α , csp)

for each val in OrderDomainVals(var, α , csp) do | picks legal values

add (var = val) to α

result \leftarrow RB(α , csp)

if result \neq failure then return result

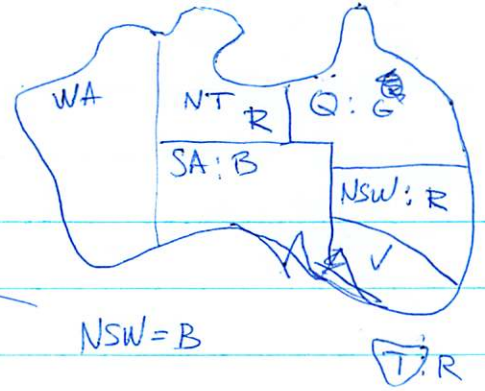
remove (var = val) from α

return failure

CSI (6)

18 Backtracking Success

{ }



foreach

$\alpha = \text{NSW} = \text{R}$

$\text{NSW} = \text{G}$

$\text{NSW} = \text{B}$

$\text{T} = \text{R}$

foreach

$\text{NT} = \text{R}$ $\text{NT} = \text{G}$ $\text{NT} = \text{B}$

$\alpha = \text{NSW} = \text{R}, \text{NT} = \text{R}$

foreach

$\text{Q} = \text{G}$ $\text{Q} = \text{B}$

$\alpha = \alpha + \text{Q} = \text{G}$

$\text{SA} = \text{B}$

$\begin{matrix} \circ \\ \circ \\ \circ \end{matrix}$

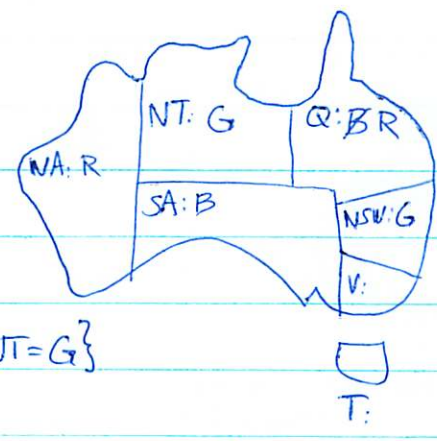
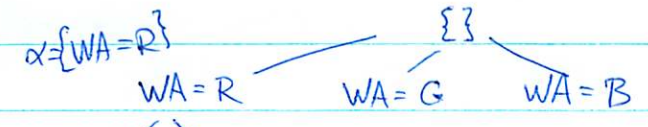
$\text{T} = \text{R}$ $\text{T} = \text{G}$ $\text{T} = \text{B}$

$\text{V} = \text{G}$

$\text{WA} = \text{G}$

$\frac{1}{2}$

CSI ⑦
foreach

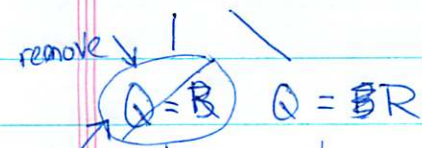


foreach



$\alpha = \{WA=R, NT=G\}$

foreach



$\alpha = \{WA=R, NT=G, Q=B\}$
 $Q=R$

back track

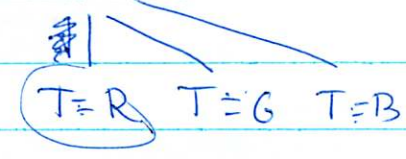
$SA: \{\}$ → Whoops!
 No legal choices

$SA=B$

$\alpha = \alpha + \{SA=B\}$

$NSW=G$

$V=R$



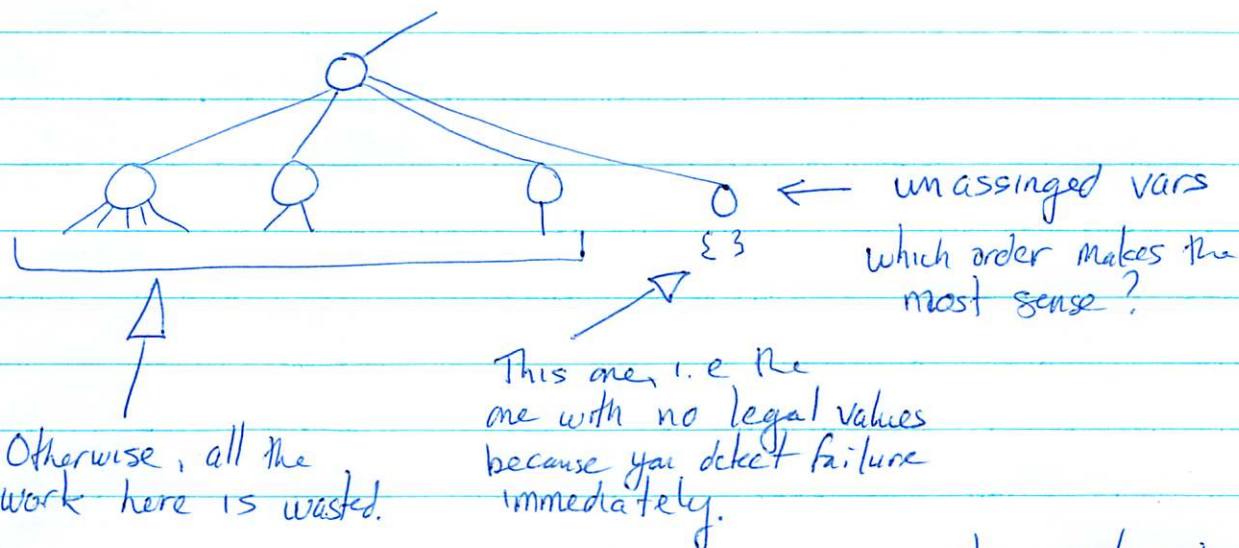
CSI ⑧

In Backtracking Search, we had choices of

- 1) what variable should be assigned next
- 2) what order to consider the values

Do these choices matter? Can they improve search?

Suppose you're deep in the tree:



Minimum remaining values (MRV) heuristic:
- choose var with fewest "legal" values

aka = most constrained var

• fail-first

R&N: 3-3000 times better than simple backtracking

[?] What about the first node? MRV doesn't help.

~~But what would~~ which would make the most sense?

SA because it constrains 5 other variables

This results in the greatest reduction of the search tree size

Called the degree heuristic.