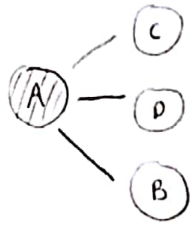
CS188 Exam Prep 2

Q1. a) i.

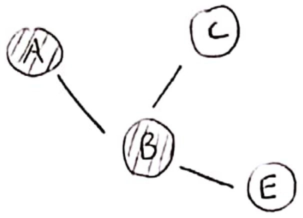


C → A

D → A

B → A

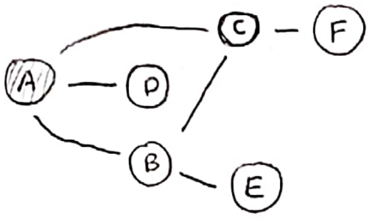
ii.



C → B

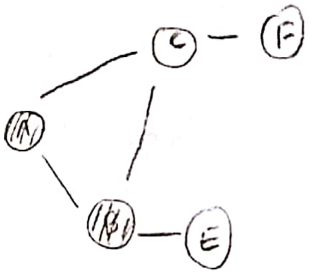
E → B

iii.



B → A
D → A
C → A
↓
⋮
↓
F → C
E → B

iv.



C → B
E → B
↓
⋮
↓
F → C

D → A doesn't appear
bc A → C, A → B never appear
(A is assigned so can't be tail)

- AC-3: enforce binary constraints
  - 3rd algo in a series of optimizations
  - either as preprocessing step or propagation during search
  - assigned vars can only be the head of an arc
  - runtime = $O(ed^3)$
    - enforcing arc = $O(d^2)$
    - enforcing arc is called at most $O(ed)$
      - bc modifying domain of size d removes at least 1 val and adds at most e arcs to the queue

start with arcs that have assigned var as head

Tree CSP:
if no empty domains,
there is a set of valid assignments

b) i. A - B - C - D - E : 0

☆ Same as Tree 2-Pass Algo.
→ No backtracking

- enforce AC
- assign A
- assign B (in same valid assignment as A)
- enforce AC ([illegible])
- assign C (in same valid assignment as A,B)
- assign D (in same valid assignment) as A,B,C)

...

- Backtracking:
  - iterate through vars
  - only consider vals consistent with previous assignments

- Tree CSP:
  - backward pass
    - enforce Pa(X) → X
      [illegible]
  - forward assignment

Tree 2-Pass Algo

(backward pass) - enforce D→E, C→D, B→C, A→B (subset of AC algo arcs) $O(ed^2)$

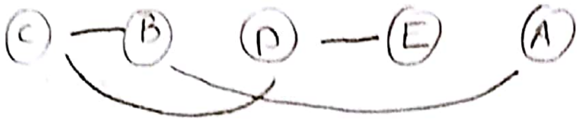
(forward assignment) - assign A, B, C, D, E in that order $O(n)$

- general: pick root, perform Topo Sort, enforce $Par(X_i) \to X_i$ for $i = n, \dots, 2$,

  assign X_i for $i = 1, \dots, n$

C - B - D - E - A : 0 · Same as Tree 2-Pass

Topo sort: each node doesn't have ≥ 2 edges from left



A - E - B - D - C : 2(d - 1)

- enforce AC (if no empty domain, there is a set of valid assignments)
- assign A
- assign E (might not be on same valid assignment as A)
  - worst case: backtrack through all values in domain of E except for correct assignment (d-1 back-tracks

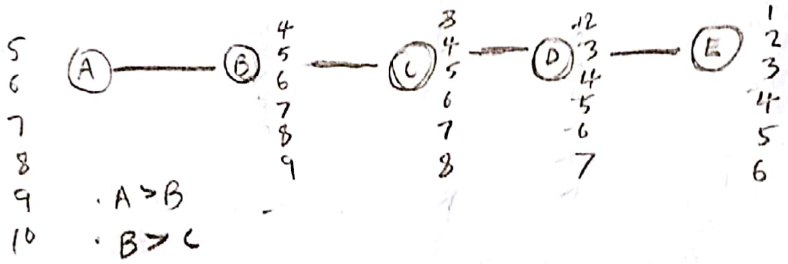
...

- assign D (same as E, d-1 backtracks)
- observe: E is not connected to previous var A (may lead to conflict),

  D is not connected to previous var B,

  B/C are assigned immediately after AC

Example



- A > B
- B > C
- C > D
- D > E

- Assign A = 5
- Assign E = 6
- Backtrack after AC
- Assign E = 5
- Backtrack after AC

...

d-1 backtracks

ii. A-B-C-D-E-F-G: d^2-1

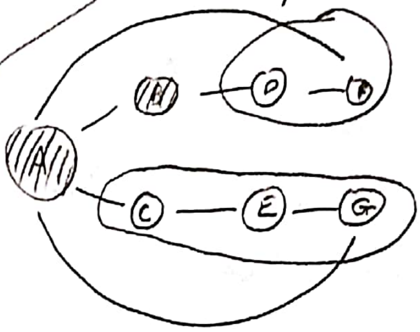
- enforce AC3 (doesn't ensure 0 backtracks)
- assign A (might not be valid)
  - CSP becomes 2 trees
  - next AC3 ensures 0 backtracks
- assign B (might not be valid)
- enforce AC (empty domain)
- backtrack A, B
- enforce AC (empty domain)
- backtrack A, B
- ...

d^2-1 backtracks
- d^2 total combos
- at least 1 valid

- enforce AC (non-empty domain, Tree CSP → 0 backtracks) & topo ordering
- ...

X A → ? B → ?

A, B will never be tails

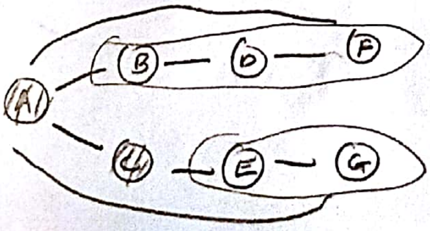


FD-B-A-C-G-E: $d^4-1+d-1$

- F-D-B-A → AC3 on Tree CSP = d^4-1 backtracks
- C-G → G might not be in same valid assignment as C = $d-1$

C-A-F-E-B-G-D: d^2-1 + d

- enforce AC
- assign C
- assign A
- backtrack d^2-1 times on C, A while enforcing AC
- finally perform AC with non-empty domains in Tree CSP



Assignment Order: F, B, D
↳ not topological!
↳ but AC3 occurs between F and B!
↳ no backtracks

Q2.
- Search ALL solutions
  - LCV doesn't matter
    - only changes order of exploring domain of an unassighed var
  - MRV affects order of node expliration
    - matters if there are edges

Final Notes

- CSP problems tend to be mechanical
- Understand runtime, Tree 2-Pass, AC, Forward-checking