CS440 MIDTERM Solutions

Q1: (a)

BFS:

open list (Node)

Step 1: (A)

Step 2: (B, C, D, E)

Step 3: (C, D, E)

Step 4: (D, E, G)

Step 5: (E, G, F)

Step 6: (F, G)

Step 7: (G)

Step 8: ∅

(G) Goal Found!

path returned: A - C - G

DFS: (considering left most child to be the last pushed into the stack)

open list (Node)

Step 1: (A)

Step 2: (B, C, D, E)

Step 3: (C, D, E)

Step 4: (G, D, E)

Step 5: (D, E)

(G) Goal Found! path returned: A - B - C - G

Greedy Best First:

open list (Node, h(n))

Step 1: ((A, 2))

Step 2: ((C, 0), (D, 1), (E, 1), (B, 3))

Step 3: ((G, 0), (D, 1), (E, 1), (B, 3))

Step 4: ((D, 1), (E, 1), (B, 3))

(G) Goal Found! path returned: A - C - G

A* search

open list (Node, g(n), h(n), f(n))

Step 1: ((A, 0, 2, 2))

Step 2: ((C, 5, 0, 5), (B, 3, 3, 6), (D, 10, 1, 11), (E, 10, 1, 11))

Step 3: ((B, 3, 3, 6), (G, 7, 0, 7), (D, 10, 1, 11), (E, 10, 1, 11))

Step 4: ((G, 7, 0, 7), (D, 10, 1, 11), (E, 10, 1, 11)) (G) Goal Found!

Step 5: ((D, 10, 1, 11), (E, 10, 1, 11))

path returned: A - C - G

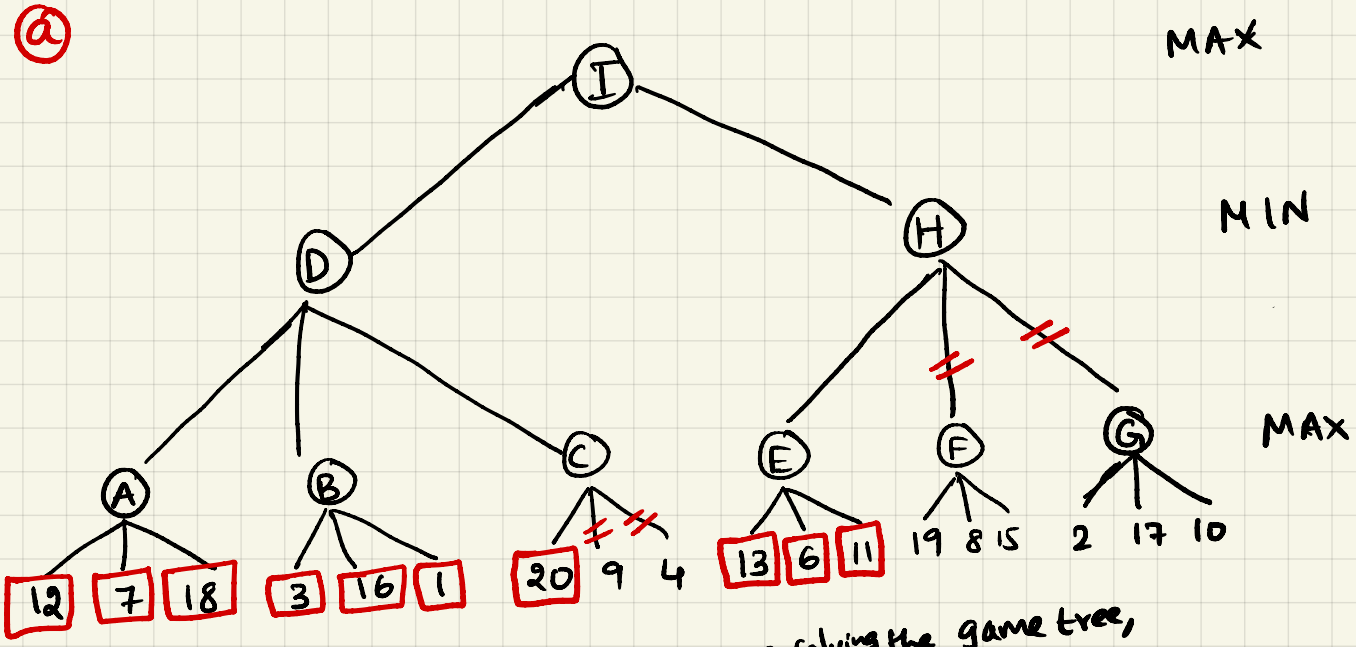
(b) Given graph has admissible h-values, but A* did not find the optimal path! for A* graph-search, heuristic needs to be consistent. Here $h(B) > h(C) + cost(B,C)$, where $h(B) = 3$, $h(C) = 0$, $cost(B,C) = 1$ i.e, $3 > 0+1$, where h is not consistent.

BFS, Greedy Best search did not find while DFS found the optimal path!

(c) Yes! as long as the graph is finite!

Problem 2:

(a)



(b) MAX(I) moves to node (D). Utility of (I) will be 16 upon solving the game tree, upon moving to D where as 13 upon moving to H. So, 'I' chooses 'D' to maximize its utility.

(c) (I)'s best first move is 'D' → MAX(D:20, H:17) as the utility of that move turns out to be '20' where as choosing 'H' as the first move yields an utility of '17'

↓ MAX(20,9,4)

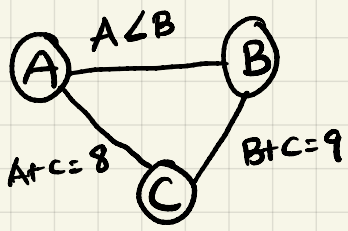
↳ MAX(2, 17, 10)

③ $D(A) = \{2,3,4\}$
$D(B) = \{3,4,5\}$
$D(C) = \{4,5,7\}$

$A < B$
$B + C = 9$
$A + C = 8$

$A - B < 0$
$B + C = 9$
$A + C = 8$

ⓐ



ⓑ NO! For example B is not arc-consistent with A for the binary constraint $B > A$

Arcs - should consider both directions

$A < B$
$B > A$
$B = 9 - C$
$9 - C = B$
$C = 9 - B$
$9 - B = C$
$A = 8 - C$
$8 - C = A$
$C = 8 - A$
$8 - A = C$

→

Initial Agenda(Queue)

$A < B$
$B > A$
$B = 9 - C$
$9 - C = B$
$C = 9 - B$
$9 - B = C$
$A = 8 - C$
$8 - C = A$
$C = 8 - A$
$8 - A = C$

AC-3 algo:

1) Apply $A < B$

$D(A) = \{2,3,4\}$ - No reduction

$A < B$ is now satisfied & removed from agenda

2) Apply $B > A$

$D(B) = \{3,4,5\}$ - No reduction

$B > A$ is now satisfied & removed from agenda

3) Apply $B = 9 - C \rightarrow D(9-C) = \{5,4,2\}$

$D(B) = \{4,5\}$ - Reduced

$B = 9 - C$ is now satisfied & removed from agenda

4) Load $A < B$, $9 - C = B$, $C = 9 - B$ into the agenda

Agenda Now:
$9 - C = B$ ← already exist
$C = 9 - B$ ←
$9 - B = C$
$A = 8 - C$
$8 - C = A$
$C = 8 - A$
$8 - A = C$
$A < B$ ← newly added

(All Arcs with Var B on the right side of the Arc constraint)

Apply $9 - C = B$

$D(9-C) = \{5,4,2\}$, $D(B) = \{4,5\}$

$D(C) = \{5,4\}$ - Reduced

$9 - C = B$ is Now satisfied & removed from agenda

5) Load $B = 9 - C$, $9 - B = C$, $A = 8 - C$, $8 - A = C$ into agenda. Ignore if already exists in the agenda

Agenda Now:
$C = 9 - B$
$9 - B = C$ ←
$A = 8 - C$ ← already exist
$8 - C = A$
$C = 8 - A$
$8 - A = C$ ←
$A < B$
$B = 9 - C$ ← newly added

Apply $C = 9 - B$

$D(C) = \{5,4\}$ $D(9-B) = \{5,4\}$

$D(C) = \{5,4\}$ - No reduction

$C = 9 - B$ is now satisfied & removed from agenda

6) Apply $9-B=C$

$D(9-B) = \{5,4\}$ $D(C) = \{5,4\}$

After Constraint Satisfaction

$D(B) = \{4,5\}$ - NO reduction

$9-B=C$ is now satisfied & removed from agenda

7) Apply $A = 8-C$

$D(A) = \{2,3,4\}$ $D(8-C) = \{3,4\}$

After Constraint Satisfaction, $D(A) = \{3,4\}$ - reduced

$A=8-C$ is now satisfied & removed from agenda

8) Load $B>A$, $8-C=A$, $C=8-A$ into agenda

agenda now:

$8-C=A$ ← Already exist

$C=8-A$ ← exist

$8-A=C$

$A<B$

$B=9-C$

$B>A$ ← Newly added

apply $8-C=A$

$D(8-C) = \{3,4\}$ $D(A) = \{3,4\}$

After Constraint Satisfaction, $D(C) = \{5,4\}$ - No reduction

$8-C=A$ is now satisfied & removed from agenda

9) Apply $C=8-A$

$D(C) = \{5,4\}$ $D(8-A) = \{5,4\}$

After Constraint Satisfaction,

$D(C) = \{5,4\}$ - No reduction

$C=8-A$ is now removed from the agenda

$C=8-A$

$8-A=C$

$A<B$

$B=9-C$

$B>A$

10) Apply $8-A=C$

$D(A) = \{3,4\}$ - No reduction

$8-A=C$ is now removed from the agenda

11) Apply $A<B$

$D(A) = \{3,4\}$ $D(B) = \{4,5\}$

After Constraint Satisfaction,

$D(A) = \{3,4\}$ - No reduction

$A<B$ is now removed from the agenda

12) Apply $B=9-C$

$D(B) = \{4,5\}$ $D(9-C) = \{4,5\}$

After Constraint Satisfaction, $D(B)$ has No reduction

$B=9-C$ is now removed from the agenda

13) Apply $B>A$

$D(B) = \{4,5\}$ $D(A) = \{3,4\}$

After Constraint Satisfaction, $D(B)$ has No reduction

$B>A$ is now removed from the agenda

Agenda (Queue) is now empty and AC-3 is Now complete.

Final Domains $D(A) = \{3,4\}$ $D(B) = \{4,5\}$ $D(C) = \{4,5\}$

4) Worst Case TC of AC-3 algo is $O(bd^3)$, where 'b' is Number of Arcs and 'd' is the size of the largest domain.

Justification:

1) There are at most $O(b)$ arcs in the CSP
2) Each arc is checked at most $O(d)$ times
   - due to domain reduction, we will need to re-check related arcs.
3) Each check takes $O(d^2)$ time
   - Consistency check for arc (X_i, X_j) involves checking all pairs of values in the worst case.

$\therefore$ Total WC TC would be $O(b \cdot d \cdot d^2) = O(bd^3)$