Midterm Exam

Name: _____

Perfect score: 100 - Available points: 115.

Uninformed Search: What are the computational properties of iterative-deepening depth-first (i.e., completeness, optimality, space and time complexity)? (5 points)

- Completeness: Under conditions, yes. The conditions are similar to those of BFS, i.e., finite brancing factor and a goal exists at a finite depth.
- Optimality: Under conditions, yes. The conditions are similar to those of BFS, i.e., if the cost is a non-decreasing function of the depth of the search tree, then the method is optimal (this includes the case that all edge weights have the same cost).
- Time Complexity: Exponential complexity, i.e., $O(b^d)$, where b is the branching factor of the search tree and d is the depth of the shallowest goal node.
- Space Complexity: Polynomial complexity, similar to DFS, i.e., $O(b \cdot d)$.

Prove that uniform-cost search is optimal when used in the context of GRAPH-SEARCH. (10 points)

In the context of GRAPH-SEARCH we have to show that uniform-cost search first selects for expansion the goal node that corrersponds to the optimal path before any suboptimal goal node.

We can prove that if G_2 is a node that corresponds to a suboptimal goal state, uniform-cost searh will select a reachable node n along the optimal path - and in this way eventually the optimal goal G^* - before selecting G_2 for expansion.

Note that a reachable node n along the optimal path always exists in the set of fringe nodes, since at least the root of the tree and one of its successors trivially belong to the optimal path.

Furthermore, $f(G_2) = g(G_2) + h(G_2) = g(G_2) + 0 > C^*$, where C^* is the optimum path to the goal. The heuristic value of G_2 is zero because G_2 is a goal node ($h(G_2) = 0$). Furthermore, the path cost from the root $g(G_2)$ has to be greater than the optimum cost to the goal C^* because G_2 is a goal node that is connected to the root with a suboptimal path ($g(G_2) > C^*$).

On the other hand: $f(n) = g(n) + h(n) \leq C^*$. This is due to the fact that $h(n)$ is optimistic and underestimates the distance of n from the goal node.

Thus, we can conclude that: $f(n) \leq C^* < f(G_2)$, which proves that n - and in this way eventually the optimal goal G^* - will be selected before the suboptimal goal node G_2. So, uniform-cost search is optimal.

Why do we use a variation of depth-first search to solve constraint-satisfaction problems? How is this variation called? (5 points)

The variant of depth-first search used for CSPs is called “backtracking search”.

The benefit of DFS in this context stems from its low space requirements and the fact that it is complete in this domain. Completeness is due to the depth of the search tree in CSPs being finite. The maximum depth equals the number of variables in the CSP.

Informed Search: Specify sufficient conditions for A* to expand the same nodes as Breadth-First search? (5 points)

[Alternative answers are possible here] Two sufficient conditions for A* to expand the same set of nodes with BFS are the following:

- The heuristic value, at every level/depth of the search tree has to be less than or equal to the heuristic value at a lower depth. This includes the case of a constant or zero heuristic value for all nodes.
- Edge Cost: The edge cost value, at every level/depth of the search tree has to be greater than or equal to the edge costs at a lower depth. This includes the case of a constant edge cost.

What is the definition of a consistent heuristic? Prove that a consistent heuristic is also admissible. (10 points)

A heuristic function h is consistent if, for every node n and every successor n' of n generated through an action a, the heuristic value $h(n)$ is no greater than the step cost of getting to n' plus the heuristic value $h(n')$:

$$h(n) \leq c(n, a, n') + h(n') \tag{1}$$

Consistency implies admissibility. An admissible heuristic always underestimates the cost to the goal. If $C^*(n)$ is the optimal cost from node n to the goal n_g, then: $h(n) \leq C^*(n)$
Proof:
If a node n_{g-1} is one edge away from the goal,
$h(n_{g-1}) \leq c(n_{g-1}, n_g) + h(n_g) \Rightarrow h(n_{g-1}) \leq c(n_{g-1}, n_g)$
If we are two edges away:
$h(n_{g-2}) \leq c(n_{g-2}, n_{g-1}) + h(n_{g-1}) \Rightarrow h(n_{g-2}) \leq c(n_{g-2}, n_{g-1}) + c(n_{g-1}, n_g) \Rightarrow h(n_{g-2}) \leq C(n_{g-2}, n_g)$

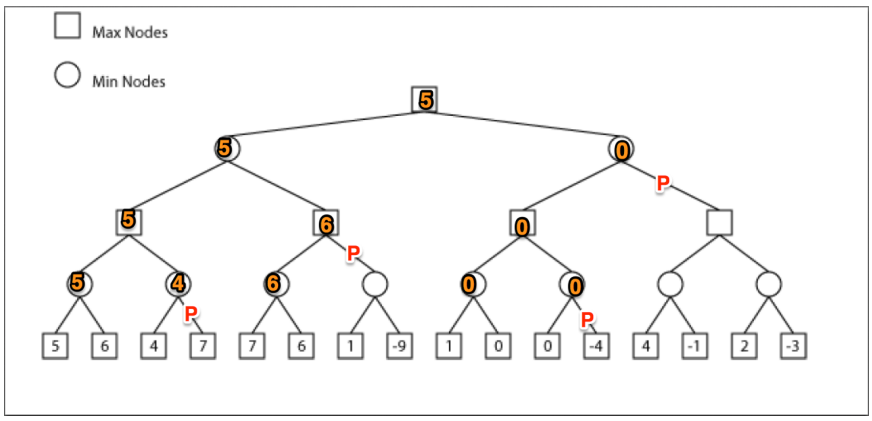
By induction, for the specific edge sequence $g, g-1, g-2 \ldots g-i$: $h(n_{g-i}) \leq C(n_{g-i}, n_g)$

One such sequence of edges correspond to the shortest path, where:
$h(n_{g-i}) \leq C^*(n_{g-i}, n_g) \Rightarrow h(n) \leq C^*(n)$

Which optimal informed search algorithm expands fewer nodes than A* for the same heuristic? Ignore the effects of ties. (5 points)

No such algorithm exists. A* expands the fewest nodes to reach the goal node with the optimal cost, given a specific admissible and consistent heuristic.

Adversarial Search: Which terminal nodes from the tree below will not be checked by the minimax algorithm if we apply alpha-beta pruning? Fill the nodes in the intermediate and top layers. (5 points)



Are the following statements true or false? Explain in a single sentence in each case. (5 points)

- A player using minimax is guaranteed to play optimally against any opponent.

  Answer : No, false, the minimax strategy assumes that the opponent plays optimally.

- A player using alpha-beta pruning in the context of Minimax is guaranteed to play optimally against an optimal opponent.

  Answer: Yes, true. Alpha-beta pruning is not removing any part of the state-space where an optimal solution may exist when playing against an optimal opponent.

- We can apply alpha-beta pruning in the context of the Expectiminimax algorithm and still get the optimal strategy against an optimal opponent.

  Answer: No, false. [There are multiple possible explanations here]

Local Heuristic Search: Define the "probabilistic completeness" property of algorithms. What is one way to achieve a hill-climbing approach that is probabilistically complete? (5 points)

An algorithm is probabilistically complete if the probability that it will produce a solution approaches 1, as more computation effort is invested.

Nevertheless, a probabilistically complete algorithm cannot determine if no solution exists.

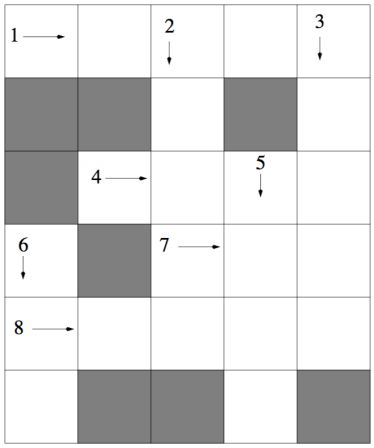
Random restarts is one way to achieve a probabilistically complete hill-climbing approach. [Multiple possible answers are valid here, e.g., random walk, simulated annealing etc.]

Constraint Satisfaction Problems: a) Name a heuristic for ordering variables in backtracking search. b) What heuristic would you use to order value assignments in backtracking search? (5 points)

a) MRV(Minimum Remaining Values), degree heuristic

b) Least-constraining value heuristic

The numbers $1, 2, \ldots, 8$ in the crossword puzzle below correspond to the words that will start at those locations. The arrows correspond to the direction of the alignment of the letters in the words. Each word from the following list appears only once in the puzzle: aft, ale, eel, heel, hike, hoses, keel, knot, laser, lee, line, sails, sheet, steer, tie.

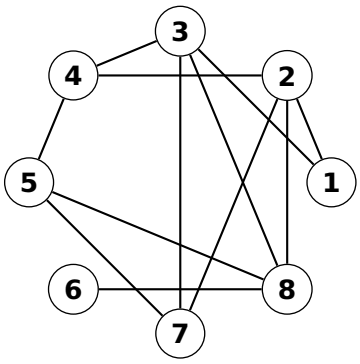


Model this puzzle as a binary CSP. State what are the variables, their respective domains, and the binary constraints between variables. Draw a graph representing the relationship between the variables and the corresponding constraints. [Hint: There are multiple possible answers here. One way is to model every variable as a vector (i.e., one dimension array) and express the constraints as an equality between the respective positions of two arrays.](15 points)

variables: $V = \{w_1, w_2, w_3, \ldots, w_8\}$
domains: $D = \{aft, ale, eel, heel, \ldots., tie\}$
binary constrains:

$$
\begin{aligned}
C = \{ & w_1[2] = w_2[0], w_1[4] = w_3[0], \\
& w_4[1] = w_2[2], w_4[3] = w_3[2], w_4[2] = w_5[0], \\
& w_7[0] = w_2[3], w_7[2] = w_3[3], w_7[1] = w_5[1], \\
& w_8[0] = w_6[1], w_8[2] = w_2[4], w_8[3] = w_5[2], w_8[4] = w_3[4], \\
& |w_1| = |w_2| = |w_3| = |w_8| = 5, \\
& |w_4| = |w_5| = 4, \\
& |w_6| = |w_7| = 3, \\
& w_1 \neq w_2 \neq w_3 \neq \ldots \neq w_8\}
\end{aligned}
$$

graph:



Logic-based Inference, Satisfiability and Classical Planning: Assume that the following proposition represents your knowledge base:

$$KB: \quad (Rain \Rightarrow WetGrass) \wedge (WetGrass \Rightarrow \neg Sprinkler) \wedge Rain$$

Use a proof by resolution to show that this knowledge base entails "$\neg Sprinkler$". (5 points)

We use resolution over the expression (KB $\vee Sprinkler$) :
$(\neg Rain \vee WetGrass) \wedge (\neg WetGrass \vee \neg Sprinkler) \wedge Rain \wedge Sprinkler$
$(\neg Rain \vee WetGrass) \wedge (\neg WetGrass) \wedge Rain$
$\neg Rain \wedge Rain$: Contradiction!

Thus, our KB entails Sprinkler.

Probability theory: In the domain of random variables (α, β), apply the marginalization rule to the following probability:

$$P(\alpha) = \sum_{\beta} P(\alpha, \beta)$$

In the domain of random variables (α, β), apply the conditioning rule to the following probability:

$$P(\alpha) = \sum_{\beta} P(\alpha|\beta)P(\beta)$$

(5 points)

Assume two random *binary* variables α, β. What is the minimum number of different probabilities that you need to know to compare $P(\alpha|\beta)$ and $P(\neg\alpha|\beta)$ using the Bayes rule? Which are these probabilities? (5 points)

We apply the Bayes rule and we get:

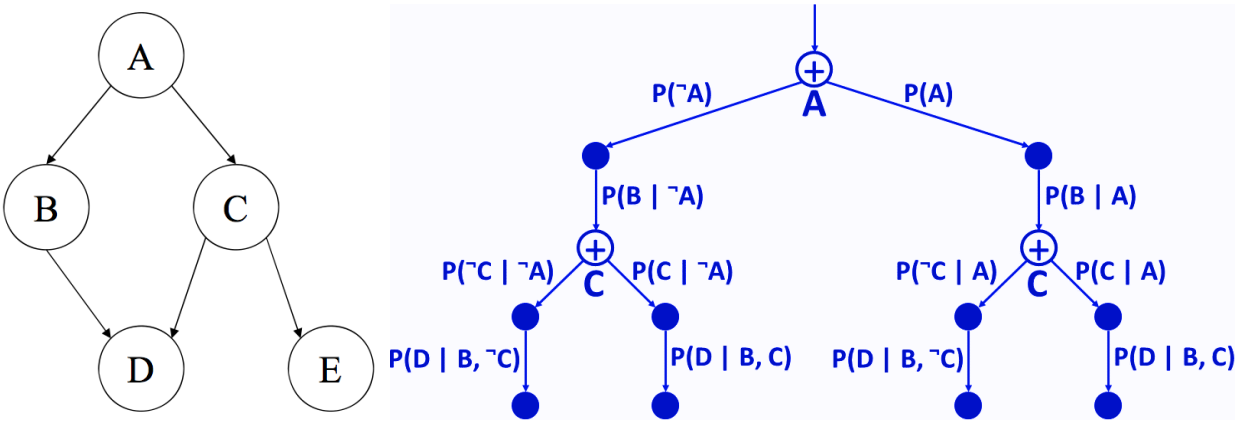
- $P(\alpha|\beta) = \gamma P(\beta|\alpha)P(\alpha)$
- $P(\neg\alpha|\beta) = \gamma P(\beta|\neg\alpha)P(\neg\alpha)$

We can calculate the probability of $P(\neg\alpha) = 1 - P(\alpha)$
Since we want to compare the 2 probabilities we will calculate the ratio:

- $\frac{P(\alpha|\beta)}{P(\neg\alpha|\beta)} = \frac{P(\beta|\alpha)P(\alpha)}{P(\beta|\neg\alpha)(1-P(\alpha))}$

The minimum number of probabilities we need to know is 3: $P(\beta|\alpha), P(\beta|\neg\alpha)$ and $P(\alpha)$.

Bayesian Networks: Consider the Bayesian network shown in the following figure. Show the steps of evaluating the conditional probability $P(B = true|D = true)$ using probabilities you can obtain from the Bayesian network. You will have to leave it in symbolic form because the Conditional Probability Tables are not provided. Then show the enumeration tree that arises from the symbolic representation in the context of inference by enumeration. (10 points)



In terms of the full, joint probability distribution function, the query is of the form

$$P(B = true|D = true) = \sum_A \sum_C \sum_E P(A, B = true, C, D = true, E)$$

Then, using the structure of the Bayesian Network to determine dependence relations,

$$P(B = true|D = true) = \alpha \sum_A \sum_C \sum_E P(A) \cdot P(B = true|A) \cdot P(C|A) \cdot P(D = true|B = true, C) \cdot P(E|C)$$

Then, rearrange the terms so as to consolidate terms under the sums, yielding

$$P(B = true|D = true) = \alpha \sum_A P(A) \cdot P(B = true|A) \cdot \sum_C P(C|A) \cdot P(D = true|B = true, C) \cdot \sum_E P(E|C)$$

Since $\sum_E P(E|C) = 1$, this expression can be further simplified to

$$P(B = true|D = true) = \alpha \sum_A P(A) \cdot P(B = true|A) \cdot \sum_C P(C|A) \cdot P(D = true|B = true, C)$$

Given the answer to the previous question, describe all the factors that would arise (their dimensionality and on which variables they would depend upon) when computing the probability $P(B = true|D = true)$ using Variable Elimination (5 points).

Factor	Dimension	Depends on	Comments
$f_D(c)$	2x1	C	From $P(D = true\|B = true, C)$.
$f_C(a, c)$	2x2	A,C	From $P(C\|A)$.
$f_{\bar{C}D}(a)$	2x1	A	Factor for summing out $\sum_C$.
$f_B(a)$	2x1	A	From $P(B = true\|A)$.
$f_A(a)$	2x1	A	From $P(A)$.
$f_{\bar{A}B\bar{C}D}$	1x1	-	Solution: summing out $\sum_A$.

Give the names and briefly describe at least 2 techniques for approximate inference in Bayesian networks that are used to answer queries involving conditional probabilities? (5 points)

[Two out of the following three were sufficient.]
Rejection Sampling: Samples may be generated from the Bayesian network according to the underlying joint distribution. Discarding the samples that do not agree with the query, the query probability may be approximated as the frequency of the joint event in the remaining samples.

Likelihood Weighting: Samples that agree with the query conditions may be generated directly (with no rejection) taking the given assignments as fixed. However, the contribution of the remaining samples must be weighted according to the likelihood of the evidence/conditions for that query. No samples are generated that fail to match the conditions.

Markov Chain Monte Carlo Techniques: Treating variable assignments as a state space, a Markov chain may be constructed taking the transition probabilities to be the conditional probabilities associated with individual variable re-assignments, conditioned on their Markov blanket. Taking the evidence variables as fixed, this results in a stationary distribution equal to the desired conditional distribution. Taking multiple samples from the state of this Markov chain, a given query probability may be estimated by the frequency of occurrence.

Describe a possible run of one of these two approximate inference processes for sampling an atomic event of the probability $P(B = true, D = true)$ and its likelihood given the Bayesian network of the previous page. (5 points)

Note that the problem asks you to consider P(B = true, D = true), i.e., a joint probability rather than a conditional probability. [Only one possible way needed to be provided - below you can find three different alternatives.]

For Direct Sampling, consider generating a sequence of samples directly from the underlying base rates and (unspecified) conditional probabilities.

$$< A = True, B = False, C = True, D = True, E = False >$$
$$< A = True, B = True, C = False, D = True, E = True >$$
$$< A = False, B = True, C = False, D = True, E = False >$$
$$< A = True, B = True, C = False, D = False, E = True >$$

Note, E is irrelevant and need not be sampled. From the above, the estimated probability is P(B and D) $\sim 1/2$

For Likelihood Weighting, suppose the samples generated were

$$s_1 =< A = True, B = True, C = False, D = True, E = True >$$
$$s_2 =< A = False, B = True, C = False, D = True, E = False >$$

In this case, the likelihood weight associated with the samples would be

$$w(s_1) = P(B = True|A = True) * P(D = True|B = True, C = False)$$
$$w(s_2) = P(B = True|A = False) * P(D = True|B = True, C = False)$$

The estimate would then be P(B and D) $\sim (w(s_1) + s(s_2))/2$

For Markov Chain Monte Carlo, consider some initial state/assignment, for instance $< A = False, B = False, C = False, D = False, E = False >$. Picking a variable uniformly at random (C), we condition on the Markov blanket and re-assign C according to the distribution $P(c|A = False, D =$

$False, E = False, B = False) = P(C|A) * P(D|B, C) * P(E|C)$. This is iterated to generate a number of samples, for instance

$$< A = False, B = False, C = True, D = False, E = False >$$

We then uniformely select another variable (A) and repeat the above procedure. Notice that the value of 1 variable changes at every iteration.

CS440: Midterm Solutions

CS440 TAs

March 26, 2018

1

1.1 Each method carries 5 points.

BFS: A, EDCB, FEDC, GFED, HGFE, ZHGF
Path: A-E-Z

For DFS there are two possibilities:

DFS(1): A, EDCB, EDCF, EDCLI, EDCLG, EDCLJ, EDCLH, EDCLK, EDCLZ
Path: A-B-F-I-G-J-H-K-Z

DFS(2): A, EDCB, DCBZ
Path: A-E-Z

Greedy BFS: A, BCDE, CDEF, DEFG, HEFG, HFGZ
Path: A-E-Z

A*: A, BCDE, CDEF, DEFG, EFGH, ZFGH
Path: A-E-Z

1.2 7.5 points

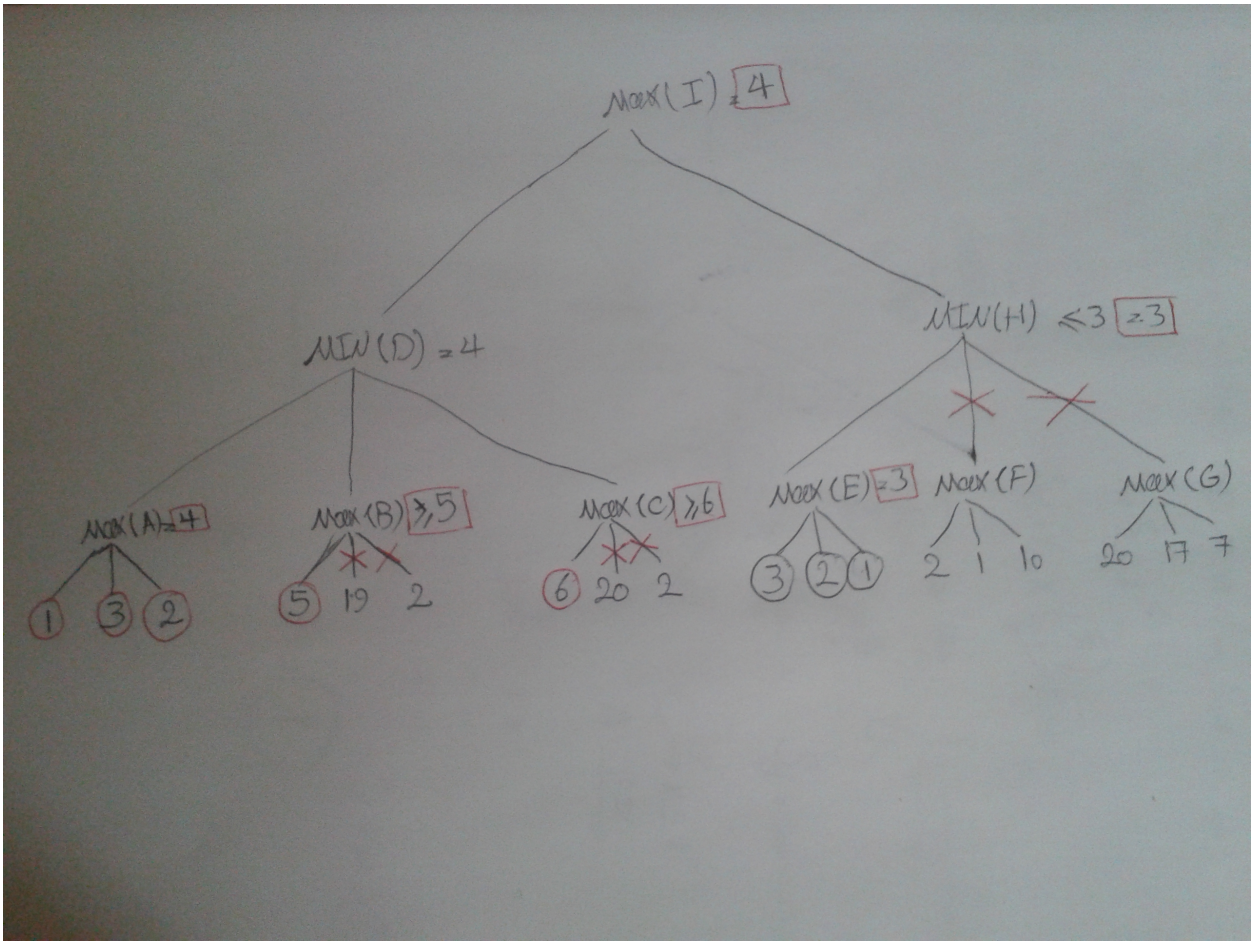
No, A*didn't find the optimal path because the heuristic function is not admissible and consistent. Other methods(BFS, DFS, GBFS) also didn't find optimal path.

1.3 2.5 points

Yes, All of them will find a solution. Only in the case if we have an infinity graph there is no answer for DFS.

2

2.1 15 points



2.2 5 points

It will select node D because its value is greater.

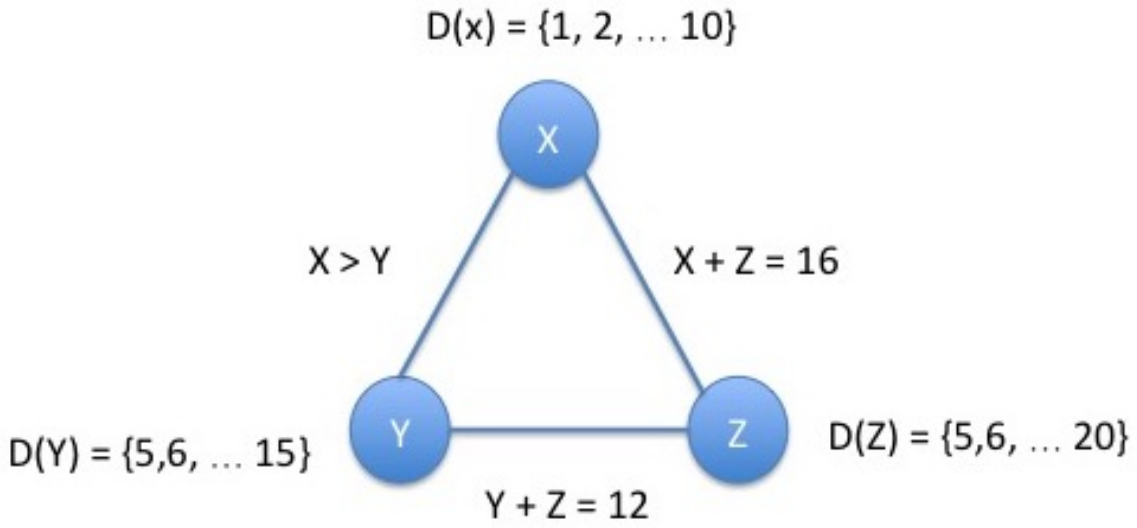
2.3 10 points

In this case the value for both D and H will be 20. So it doesn't matter to select either D or D.

3

3.1 10 points

the constraint graph can be found in figure below :



3.2 20 points

No, the constraints are not arc consistent. One example is sufficient to prove this claim. So, let us consider X=1, for which there exists no value in D(Y) such that C(X,Y) is satisfied. Further, applying arc consistency algorithm :

$$\begin{aligned} \textit{Applying}, \quad C(X,Y) : X > Y \\ D(X) : \{6,7,...10\}, \quad D(Y) = \{5,...9\} \end{aligned}$$

$$\begin{aligned} \textit{Applying}, \quad C(Y,Z) : Y + Z = 12 \\ D(Y) : \{5,...7\}, \quad D(Z) = \{5,...7\} \end{aligned}$$

$$\begin{aligned} \textit{Applying}, \quad C(X,Z) : X + Z = 16 \\ D(X) : \{9,10\}, \quad D(Z) = \{6,7\} \end{aligned}$$

$$\begin{aligned} \textit{Applying}, \quad C(Y,Z) : Y + Z = 12 \\ D(X) : \{5,6\}, \quad D(Y) = \{6,7\} \end{aligned}$$

So the final domains are,

$$D(X) = \{9, 10\}$$
$$D(Y) = \{5, 6\}$$
$$D(Z) = \{6, 7\}$$

4

4.1 5 + 5 Points

answer:
Most Constrained Variable: In the search tree, we alternate between choosing variables and choosing values for the variables. At the stage where we choose a variable, we break the search and backtrack if we find one variable that cannot be satisfied. Finding only one such variable is sufficient to say that something was wrong earlier, and we go up in the search to try other assignments. Therefore, we want to fail quickly, which will save us the trouble of trying many variables before finding the one that fails.

Least Constraining Value: Once we choose a variable, we have to try all the possible values before we can say that it failed. Therefore, we will be only losing time with values that fail (since we will still have to check the remaining values). But if we succeed, the search stops and we don't have the try the remaining values.