CS 188
Summer 2022

Exam Prep 2B Solutions

Q1. CSPs

In this question, you are trying to find a four-digit number satisfying the following conditions:

1. the number is odd,
2. the number only contains the digits 1, 2, 3, 4, and 5,
3. each digit (except the leftmost) is strictly larger than the digit to its left.

(a) CSPs

We will model this as a CSP where the variables are the four digits of our number, and the domains are the five digits we can choose from. The last variable only has 1, 3, and 5 in its domain since the number must be odd. The constraints are defined to reflect the third condition above. Thus before we start executing any algorithms, the domains are

1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 3 5

(i) Before assigning anything, enforce arc consistency. Write the values remaining in the domain of each variable after arc consistency is enforced.

1 2	2 3	3 4	5

(ii) With the domains you wrote in the previous part, which variable will the MRV (Minimum Remaining Value) heuristic choose to assign a value to first? If there is a tie, choose the leftmost variable.

- ○ The first digit (leftmost)
- ○ The second digit
- ○ The third digit
- [✓ The fourth digit (rightmost)]

(iii) Now suppose we assign to the leftmost digit first. Assuming we will continue filtering by enforcing arc consistency, which value will LCV (Least Constraining Value) choose to assign to the leftmost digit? **Break ties from large (5) to small (1).**

- [✓ 1]
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5

(iv) Now suppose we are running min-conflicts to try to solve this CSP. If we start with the number 1332, what will our number be after one interation of min-conflicts? Break variable selection ties from left to right, and **break value selection ties from small (1) to large (5)**.

$\boxed{1232}$

(b) The following questions are completely unrelated to the above parts. Assume for these following questions, there are only binary constraints unless otherwise specified.

(i) [[✓ *true*] or *false*] When enforcing arc consistency in a CSP, the set of values which remain when the algorithm terminates does not depend on the order in which arcs are processed from the queue.

(ii) [*true* or [✓ *false*]] Once arc consistency is enforced as a pre-processing step, forward checking can be used during backtracking search to maintain arc consistency for all variables.

False. Forward checking makes the current variable arc-consistent, but doesn?t look ahead and make all the other variables arc-consistent.

(iii) In a general CSP with n variables, each taking d possible values, what is the worst case time complexity of enforcing arc consistency using the AC-3 method discussed in class?

○ 0 ○ $O(1)$ ○ $O(nd^2)$ [✓ $O(n^2d^3)$] ○ $O(d^n)$ ○ ∞

$O(n^2d^3)$. There are up to n^2 constraints. There are d^2 comparisons for enforcing arc consistency per each constraint, and each constraint can be inserted to the queue up to d times because each variable has at most d values to delete.

(iv) In a general CSP with n variables, each taking d possible values, what is the maximum number of times a backtracking search algorithm might have to backtrack (i.e. the number of the times it generates an assignment, partial or complete, that violates the constraints) before finding a solution or concluding that none exists?

○ 0 ○ $O(1)$ ○ $O(nd^2)$ ○ $O(n^2d^3)$ [✓ $O(d^n)$] ○ ∞

$O(d^n)$. In general, the search might have to examine all possible assignments.

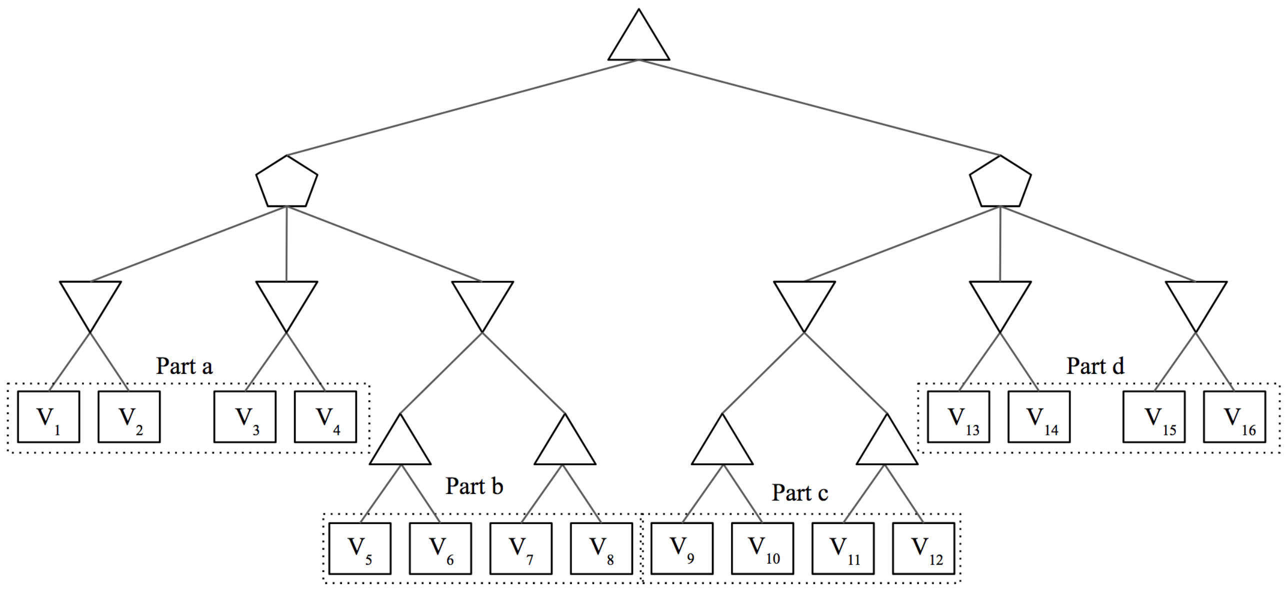
(v) What is the maximum number of times a backtracking search algorithm might have to backtrack in a general CSP, if it is running arc consistency and applying the MRV and LCV heuristics?

○ 0 ○ $O(1)$ ○ $O(nd^2)$ ○ $O(n^2d^3)$ [✓ $O(d^n)$] ○ ∞

$O(d^n)$. The MRV and LCV heuristics are often helpful to guide the search, but are not guaranteed to reduce backtracking in the worst case.

Q2. MedianMiniMax

You're living in utopia! Despite living in utopia, you still believe that you need to maximize your utility in life, other people want to minimize your utility, and the world is a 0 sum game. But because you live in utopia, a benevolent social planner occasionally steps in and chooses an option that is a compromise. Essentially, the social planner (represented as the pentagon) is a median node that chooses the successor with median utility. Your struggle with your fellow citizens can be modelled as follows:

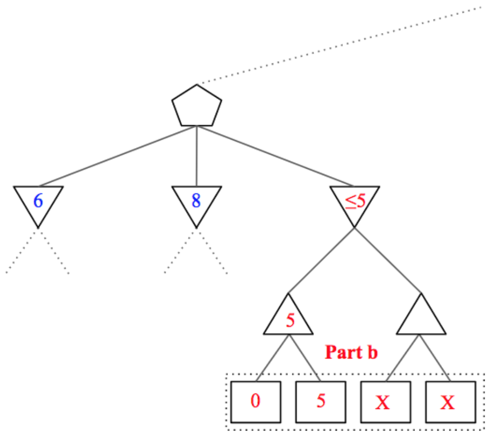


There are some nodes that we are sometimes able to prune. In each part, mark all of the terminal nodes such that **there exists a possible situation** for which the node **can be pruned**. In other words, you must consider **all** possible pruning situations. Assume that evaluation order is **left to right** and all V_i's are **distinct.**

Note that as long as there exists ANY pruning situation (does not have to be the same situation for every node), you should mark the node as prunable. Also, alpha-beta pruning does not apply here, simply prune a sub-tree when you can reason that its value will not affect your final utility.

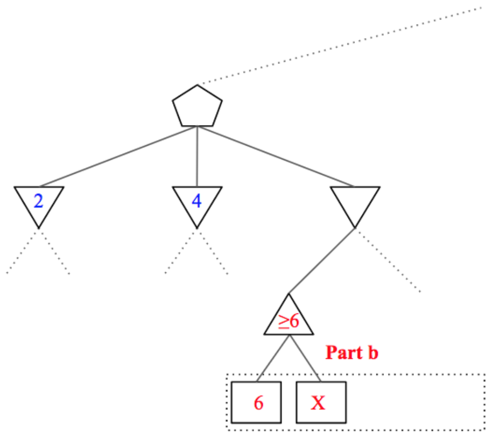
(a)		**(b)**		**(c)**		**(d)**	
	☐ V_1		☐ V_5		☐ V_9		☐ V_{13}
	☐ V_2		[✓ V_6]		☐ V_{10}		[✓ V_{14}]
	☐ V_3		[✓ V_7]		[✓ V_{11}]		[✓ V_{15}]
	☐ V_4		[✓ V_8]		[✓ V_{12}]		[✓ V_{16}]
	[✓ None]		☐ None		☐ None		☐ None

Part a:
For the left median node with three children, at least two of the childrens' values must be known since one of them will be guaranteed to be the value of the median node passed up to the final maximizer. For this reason, none of the nodes in part a can be pruned.



The value of this subtree will only get smaller.

The median node will **NOT** choose the value of this subtree. 6 is the median.



The value of this subtree will only get bigger.

*If the value of this subtree is chosen by the minimizer**, it will **NOT** be chosen by the median node.

**It is possible that the median is the value of the subtree to the right that we haven't looked at yet*

Part b (pruning V_7, V_8):
Let min_1, min_2, min_3 be the values of the three minimizer nodes in this subtree.

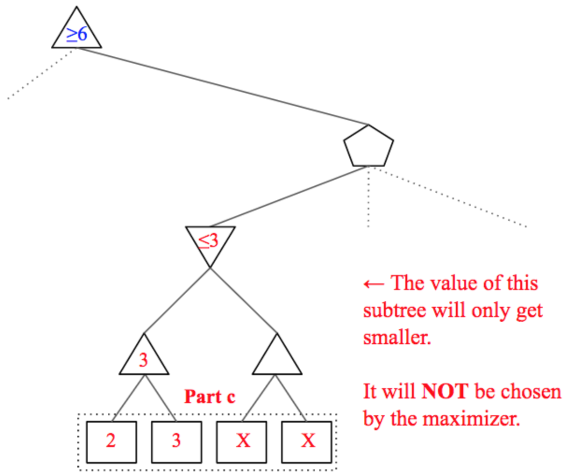
In this case, we may not need to know the final value min_3. The reason for this is that we may be able to put a bound on its value after exploring only partially, and determine the value of the median node as either min_1 or min_2 if $min_3 \leq \min(min_1, min_2)$ or $min_3 \geq \max(min_1, min_2)$.

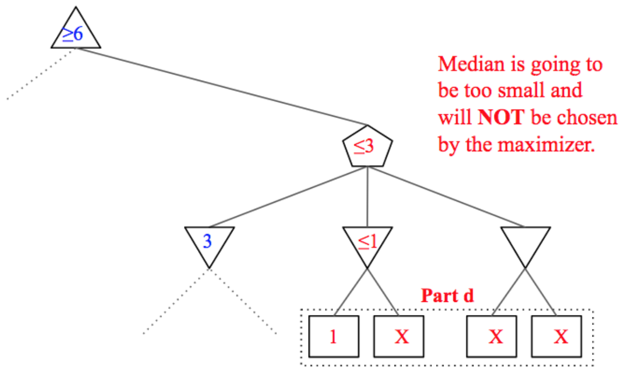
We can put an upper bound on min_3 by exploring the left subtree V_5, V_6 and if $\max(V_5, V_6)$ is lower than both min_1 and min_2, the median node's value is set as the smaller of min_1, min_2 and we don't have to explore V_7, V_8 in Figure 1.

Part b (pruning V_6):
It's possible for us to put a lower bound on min_3. If V_5 is larger than both min_1 and min_2, we do not need to explore V_6.

The reason for this is subtle, but if the minimizer chooses the left subtree, we know that $min_3 \geq V_5 \geq \max(min_1, min_2)$ and we don't need V_6 to get the correct value for the median node which will be the larger of min_1, min_2.

If the minimizer chooses the value of the right subtree, the value at V_6 is unnecessary again since the minimizer never chose its subtree.


≥6
≤3
3
2
3
X
X
Part c
← The value of this subtree will only get smaller.
It will **NOT** be chosen by the maximizer.


≥6
≤3
Median is going to be too small and will **NOT** be chosen by the maximizer.
3
≤1
Part d
1
X
X
X

Part c (pruning V_{11}, V_{12}):
Assume the highest maximizer node has a current value $max_1 \geq Z$ set by the left subtree and the three minimizers on this right subtree have value min_1, min_2, min_3.

In this part, if $min_1 \leq \max(V_9, V_{10}) \leq Z$, we do not have to explore V_{11}, V_{12}. Once again, the reasoning is subtle, but we can now realize if either $min_2 \leq Z$ or $min_3 \leq Z$ then the value of the right median node is for sure $\leq Z$ and is useless.

Only if both $min_2, min_3 \geq Z$ will the whole right subtree have an effect on the highest maximizer, but in this case the exact value of min_1 is not needed, just the information that it is $\leq Z$. Clearly in both cases, V_{11}, V_{12} are not needed since an exact value of min_1 is not needed.

We will also take the time to note that if $V_9 \geq Z$ we do have to continue the exploring as V_{10} could be even greater and the final value of the top maximizer, so V_{10} can't really be pruned.

Part d (pruning V_{14}, V_{15}, V_{16}):
Continuing from part c, if we find that $min_1 \leq Z$ and $min_2 \leq Z$ we can stop.

We can realize this as soon we explore V_{13}. Once we figure this out, we know that our median node's value must be one of these two values, and neither will replace Z so we can stop.