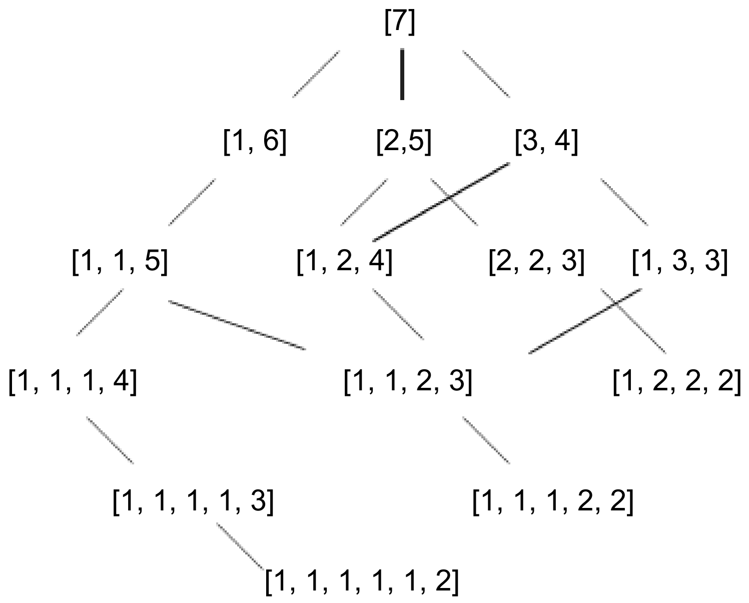
Name: Tim Flannagan
Date: 10/14/18

1.) **Adversarial Search:**

a.) **Draw a complete search space for Nim**

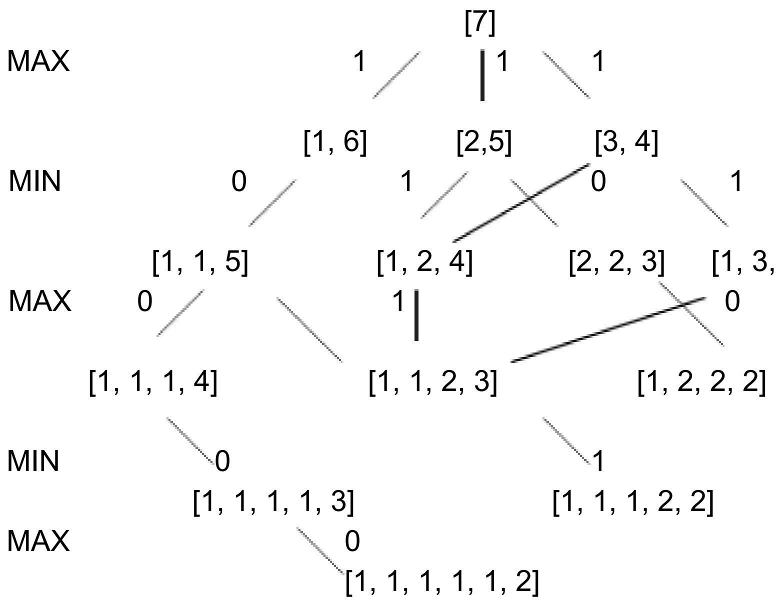
Assuming that (1, 6) is the same as (6, 1), and likewise for other instances

Invalid states = [(1, 3, 3), (2, 2, 3), (1, 1, 2, 2), (1, 1, 1, 4), (2, 5) ->(1, 1, 5)]



b.) **Apply the minimax algorithm to the search tree to assign utility functions to all states in the search tree.**

Note: MAX wants to choose a stack ordering such that there's no more legal division possible. Assign a utility to all the states in the search space.



c.) **If both min and max play a perfect game, who will win? Explain your answer.**
If both MIN and MAX play optimally then MAX will always win. The only case where max loses is if it goes down the path to [1, 1, 1, 1, 1, 2], but that's not playing optimally and MAX wouldn't choose that path.

d.) **Given the following search tree, apply the alpha-beta pruning algorithm to it and show the search tree that would be built by this algorithm. Make sure that you show where the alpha and beta cuts are applied and which parts of the search tree are pruned as a result**.

Note: won't display the v comparisons to alpha/beta unless the comparison is true, and pruning was used.
1. (State A): v = -inf, a = -inf, b = inf
2. (State B): v = inf, a = -inf, b = inf
3. (State D): v = -inf, a = -inf, b = inf
4. (State H): v = 6
5. (State D): v = max(-inf, 6) = 6, a = max(-inf, 6) = 6, b = inf
6. (State I): v = 5
7. (State D): v = max(5, 6) = 6, a = max(5, 6) = 6, b = inf
8. (State B): v = min(inf, 6) = 6, a = -inf, b = min(inf, 6) = 6
9. (State E): v = -inf, a = -inf, b = 6
10. (State J): v = 8
11. (State E): v = max(-inf, 8) = 8, v' (8) >= 8: **prune** State K and return v
12. (State B): v = min(6, 8) = 6, a = -inf, b = 6
13. (State A): v = max(-inf, 6) = 6, a = max(-inf, 6) = 6, b = inf
14. (State C): v = inf, a = 6, b = inf
15. (State F): v = -inf, a = 6, b = inf
16. (State L): v = 2
17. (State F): v = max(-inf, 2) = 2, a = max(2, 6) = 6, b = inf
18. (State M): v = 1
19. (State F): v = max(1, 2) = 2, a = max(1, 6) = 6, b = inf
20. (State C): v = min(2, inf) = 2, v' (2) <= alpha (6): **prune** State G and return v
21. (State A): v = max(2, 6) = 6, a = max(2, 6) = 6, b = inf

Pruned states: State K, State G (which included states N and O, which were terminal nodes.)

2.) **Genetic Algorithms:**

a.) Use P1, P2, and T to answer questions:

- Uniform-crossover:
  C1 = [E, B, J, H, E, F, G, A, I, G]
  C2 = [A, F, C, D, B, C, I, H, D, J]
- Order-based crossover:
  P1 = [A, B, ~~C, D, E, F~~, G, H, I, J]

P2 = [~~E, F~~, J, H, B, ~~C,~~ I, A, ~~D~~, J]
C1 = [A, J, C, D, E, F, J, H, B, I]

b.) **f(x) = x^3 - 60 * x^2 + 900 * x + 100**

Chromosome:	Binary String:	X value:	f(x) value:
P_1	11100	28	212
P_2	01111	15	3,475
P_3	10111	23	1,227
P_4	00100	4	2,804

c.) **Apply one-point crossover and show the resulting children** (**use crossover point of 1)**:

Chromosome:	Parents:	Binary String	Crossover Point:	X value:	f(x) value:
C_1	~~1~~0111 ~~0~~1111	11111	1	31	131
C_2	~~1~~0111 ~~0~~1111	00111	1	7	3,803
C_3	~~00~~100 ~~01~~111	00111	2	7	3,803
C_4	~~00~~100 ~~01~~111	01100	2	14	3,684

e.) **Assume the initial population was x = { 17, 21, 4, 28 }. Using one-point crossover, what's the probability of finding the optimal solution. Explain your reasons.**
No idea how to approach this problem.

Chromosome:	Binary:	X value:	Probability
P_1	10001	17	
P_2	10101	21	
P_3	00100	4	
P_4	11100	28	

3. **Show that KB |= a2, and KB |= a3. Construct all 32 possible worlds from the initial position. Note: possible worlds is based on KB thus far. We encountered nothing in [1, 1], a breeze in [2, 1], and a stench in [1, 2].**
Note: KB is based on the current observations and wumpus rules. For every model, we care about there are 8 different combinations. In order to get 32 different possible worlds, we need to track the different binary strings for the initial percept, (1, 3), (2,2) and (3, 1). This possible worlds ordering is based on the implication that the Wumpus is located in (1, 3).

Possible Worlds (order is [3,1], [2, 2], [1, 3]):
1. [None, None, None]
2. [None, None, Pit]
3. [None, Pit, None]
4. [None, Pit, Pit]
5. [Pit, None, None]
6. [Pit, None, Pit]
7. [Pit, Pit, None]
8. [Pit, Pit, Pit]
9. [None, None, None]
10. [None, None, Pit]
11. [None, Pit, None]
12. [None, Pit, Pit]
13. [Pit, None, None]
14. [Pit, None, Pit]
15. [Pit, Pit, None]
16. **[Pit, Pit, Pit]**
17. **[None, None, None]**
18. **[None, None, Pit]**
19. **[None, Pit, None]**
20. **[None, Pit, Pit]**
21. **[Pit, None, None]**
22. **[Pit, None, Pit]**
23. **[Pit, Pit, None]**
24. **[Pit, Pit, Pit]**
25. **[None, None, None]**
26. **[None, None, Pit]**
27. **[None, Pit, None]**
28. **[None, Pit, Pit]**
29. **[Pit, None, None]**
30. **[Pit, None, Pit]**
31. **[Pit, Pit, None]**
32. **[Pit, Pit, Pit]**

In the above list, we can eliminate any possible worlds that don't follow the wumpus rules. Any order that has a Pit for (1, 3) can be eliminated as no breeze was perceived in (1, 2). In addition to this, any order that has doesn't have a pit in either (2, 2) or (3, 1) can be eliminated as the agent detected a breeze in (2, 1). The remaining 16 elements of the list of the possible world can be used as the KB would be true. In order for a proposition to follow a KB, the proposition needs to be true in all worlds where the KB is true. If we have a model of KB, denoted model(KB), then we need to prove that both the model(a2) and model(a3) are subsets of the model(KB), which are true in both cases.

4. **Scheduling MWF computer science courses:**
a.**) Formulate this problem as a CSP problem. State the domains, and constraints**.

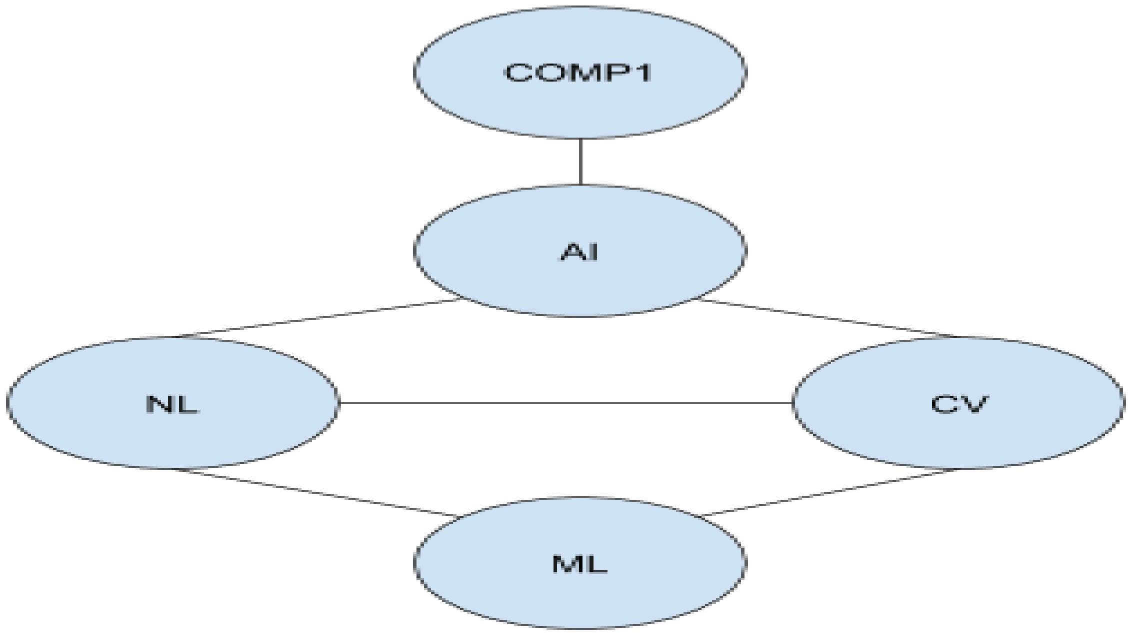
Variables: classes, denoted by X
Domain: possible values that each variable in X can take, denoted by D
Constraints: allowable combinations of values for subsets of variables, denoted by C

Professor A can teach Natural Langauge and ML
Professor B can teach AI, Natural Language, Computer Vision, and ML
Professor C can teach all of the classes

Note the values in C are classes that can't be scheduled at the same time. For example, Computing 1 ends at 9:00 am, but AI starts at 8:30 am.

X = { Computing 1, AI, NL, Computer Vision, ML }
D = { Computing 1: [C], AI: [B, C], NL: [A, B, C], Computer Vision: [A, B, C], ML: [B, C] }
C = { (COMP1, AI), (AI, NL), (AI, CV), (NL, CV), (NL, ML), (CV, ML) }

b.) **Draw the constraint graph associated with your CSP:**



c.) **Show the domains of the variables after running arc-consistency on this initial graph.**
For each edge in the CSP:

We can eliminate B from AI from Computing 1, AI edge
We can eliminate B from NL from AI, NL edge
We can eliminate B from CV from NL, CV
We can eliminate A from ML from CV, ML

Computing I: [C]
AI: [B]
NL: [A, C]
CL: [A, C]
ML: [B, C]

d.) **Give one solution to this CSP.**
Computing 1: Professor C
AI: Professor B
NL: Professor A
CV: Professor C
ML: Professor B

e.) **Your CSP should look nearly tree-structured. Briefly explain why we might prefer to solve tree-structures.**
A tree-structured CSP has no loops, therefore the time complexity would be $O(nd^2)$ greatly diminished from the worst-case general CSP time of $O(d^n)$. (Source: CSP/Chapter 6 Lecture, slide 36 states the theorem on this.)

f.) **Name a standard technique for turning these kinds of nearly tree-structured problems into tree-structured ones.**
Tree decomposition, or cutset conditioning.