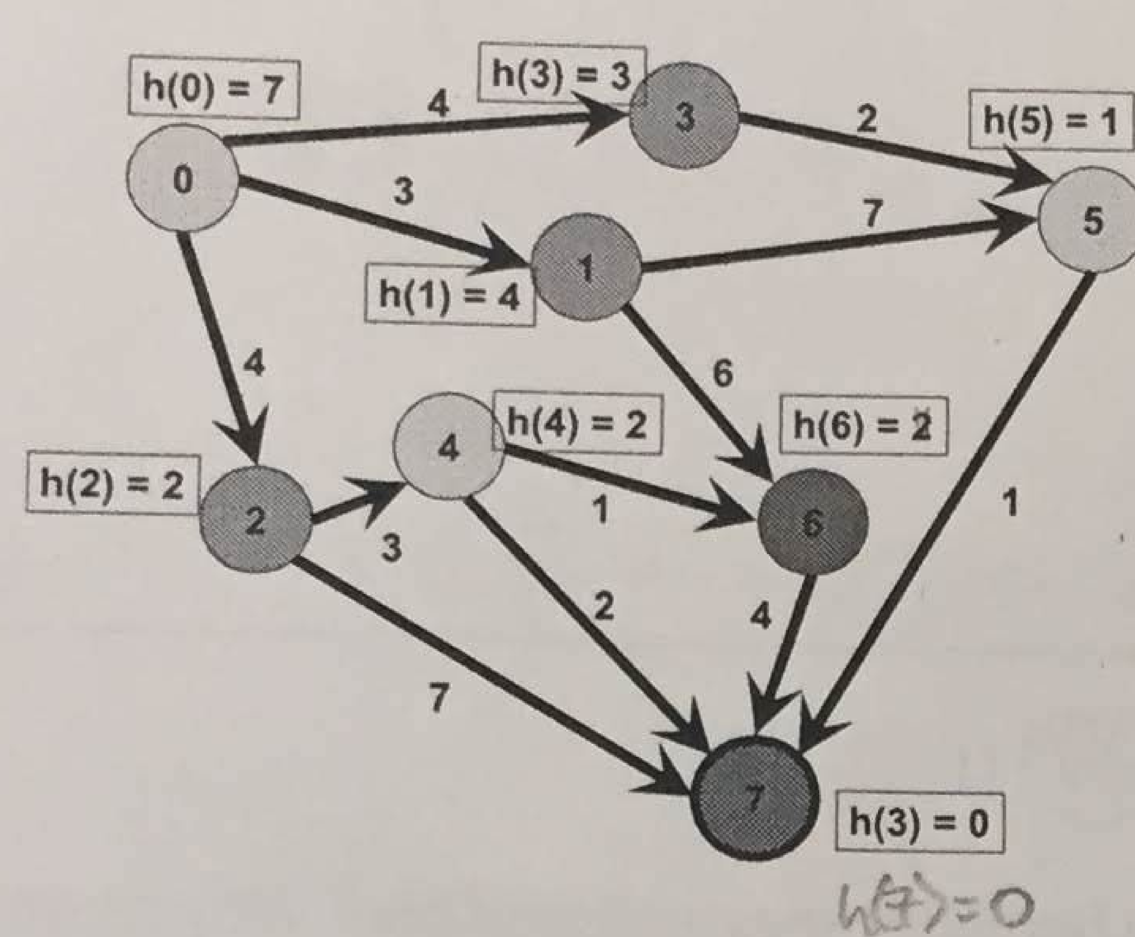
1. **Search (4-5 parts, 50 points total).**

a) **(10 points for 530, 5 points for 730) Data structures for graph search.** Write down the adjacency list for the following 8-node graph in the space to the right of the graph.



Adjacency list:

0 → 1(3) → 2(4) → 3(4) → NULL
1 → 5(7) → 6(6) → NULL
2 → 4(3) → 7(7) → NULL
3 → 5(2) → NULL
4 → 6(1) → 7(2) → NULL
5 → 7(1) → NULL
6 → 7(4) → NULL
7 → NULL

b) **(20 points) Uninformed and Heuristic Search.** Simulate the behavior of Breadth First Search and A/A* search for the above graph with start node **0** and goal node ~~3~~. 7 Show the evolution of the OPEN and CLOSED lists **and the path found**, with costs. For each search, *indicate whether the solution is optimal.* Break ties in ascending order of node number (lower-numbered nodes are expanded first in case of a tie).

(5 points) Breadth-First Search:

OPEN	closed
[0]	[ ]
[1, 2, 3]	[0]
[2, 3, 5, 6]	[0, 1]

when chosing 2 for expansion, the goal test on 7 will return it as a sol'n.

Path found: 0 → 2(4) → 7(7), Total Path Cost: 11

Optimal path cost in this case? Y / N n   f(n)

#(#)

f(n) = g(n) + h(n)

Before sorting

(10 points) A/A* search:

Before sorting	OPEN	CLOSED
	[0(7)]	[ ]
1(7), 2(6), 3(7)	[2(6), 1(7), 3(7)]	[0]
4(9), 7(11)	[1(7), 3(7), 4(9), 7(11)]	[0, 2]
5(11), 6(11)	[3(7), 4(9), 5(11), 6(11), 7(11)]	[0, 2, 1]
5(7)	[5(7), 4(9), 6(11), 7(11)]	[0, 2, 1, 3]
7(7)	[7(7), 4(9), 6(11)]	[0, 2, 1, 3, 5]

return

Path found: 0 → 3(4) → 5(2) → 7(1)   Total cost: 7

Optimal path cost in this case? Y / N

c) **(5 points) Admissibility.** Is the heuristic above admissible? Why or why not?

never overestimates

Yes, it never overestimates the actual cost to the goal,

d) **(15 points for 530, 10 for 730) Consistency.** Is the heuristic consistent? Why or why not? (Use an illustration if it helps.) Is it possible for a heuristic to be admissible but not consistent? Why or why not?

No, h(0) = 7, but c(0, 0→2, 2) + h(2) = 4 + 2 = 6 ≤ 7

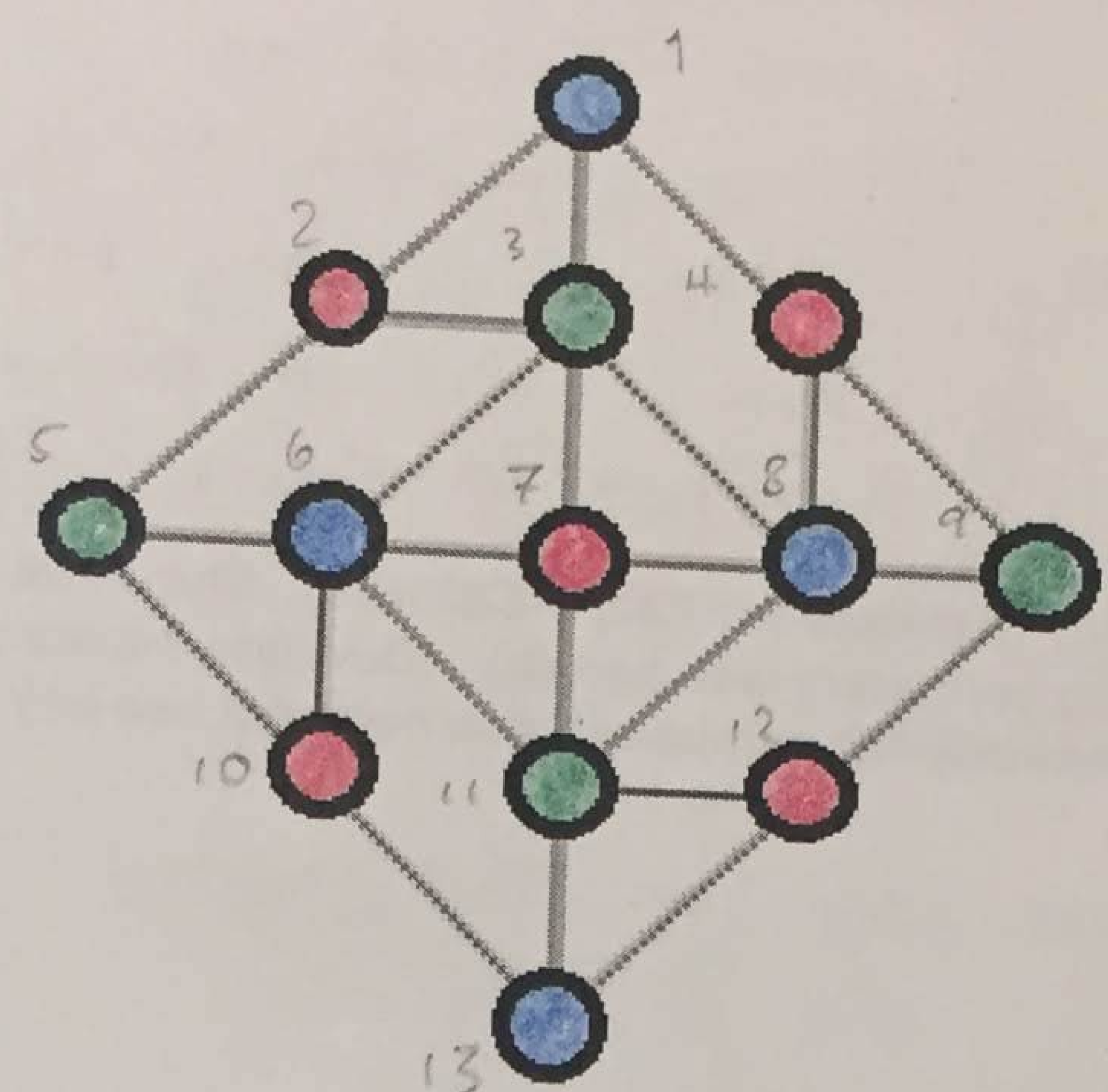
This breaks the triangle inequality required for consistency.

It is therefore possible for a heuristic to be admissible, but not consistent. This problem is an example.

e) **(10 points, 730 only) Heuristic Quality and Efficiency.** Is the heuristic worse or better than h = 0 (Branch and Bound)? Why? (What does this have to do with the number of nodes expanded?)

Slightly Better. Due to the tie breaking rule, node 4 is expanded before node 7. Since h=0 causes more nodes to be expanded than h(n) from the problem, it will take longer to find a sol'n and is worse.

2. **Constraint Satisfaction Problems (3-4 parts, 50 points total).** Consider the problem of coloring each node of this 13-node graph with one of three colors (**R, G, B**):



a) **(5 points) CSP Graph Representation.** Specify what the edges (links or neighborhood function) and vertices (nodes) in the above CSP mean.

The nodes are variables, and the links imply that a constraint exists between two nodes.

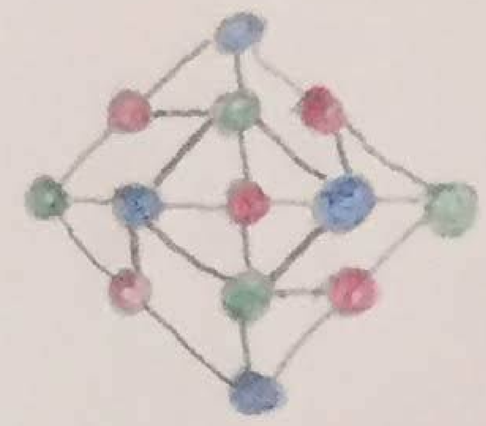
b) **(25 points for 530, 20 points for 730) CSP Methods.** Suppose Node 1 (top) is colored **Blue**. Define **all** of the following and **choose one** to illustrate with an example using the above graph.

i) *Most constrained variable / Minimum remaining values (MRV) heuristic* for variable selection
ii) *Least constraining value* for value ordering
iii) *Forward checking* for speeding up constraint checking

i) When selecting an unassigned variable, choose one with the fewest "legal" values.

ii) When selecting from the domain of the chosen value, select the value that imposes the least ammount of constraint on the neighboring values.

iii) When a variable is selected for $node_i$, check the domains of the downstream nodes, removing from their domains values that are outside constraints given the $node_i$ assignment.



	1	2	3	4	5	6	7	8	9	10	11	12	13
1	B	RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB
	B	RG	RG	RG									
3	B	RG	G	RG									
	B	R	G	R		RB	RB	RB					
2	B	R	G	R		RB	RB	RB					
	B	R	G	R	GB	RB	RB	RB					
4	B	R	G	R	GB	RB	RB	RB					
	B	R	G	R	GB	RB	RB	B	GB				
8	B	R	G	R	GB	RB	RB	B	GB				
							R		G		RG		
7							R						
						B					G		
6			G			B	R				G		
					G		R			RG	G		
5					G								
	B	R	G	R	G	B	R	B	G	R	G		
9									G				
				R				B				RB	
10										R			
					G	B							GB
11											G		
						B	R	B				RB	B
13													B
										R	G	R	
12	B	R	G	R	G	B	R	B	G	R	G	R	B

c) **(10 points, 730 only) AC-3.** Explain in your own words how to use a table to store coloring constraints and show how it is updated using the AC-3 algorithm for the above graph. Give enough details to distinguish AC-3 from forward checking.

The table includes the domain of each variable, then using a list of constrained variable pairs, AC-3 attempts to reduce their domains. Any reduction leads to the appending of further constraint pairs to this list. This preprocesses the entire CSP, as opposed to forward checking which happens after variable selection

d) **(20 points for 530, 15 points for 730) 3-Coloring.** Show that the graph **is** 3-colorable by finding a 3-coloring consistent with part (b). (Deciding 3-colorability, i.e., whether a graph is 3-colorable, is actually NP-complete.) You need not use any of the above heuristics, but they should help.

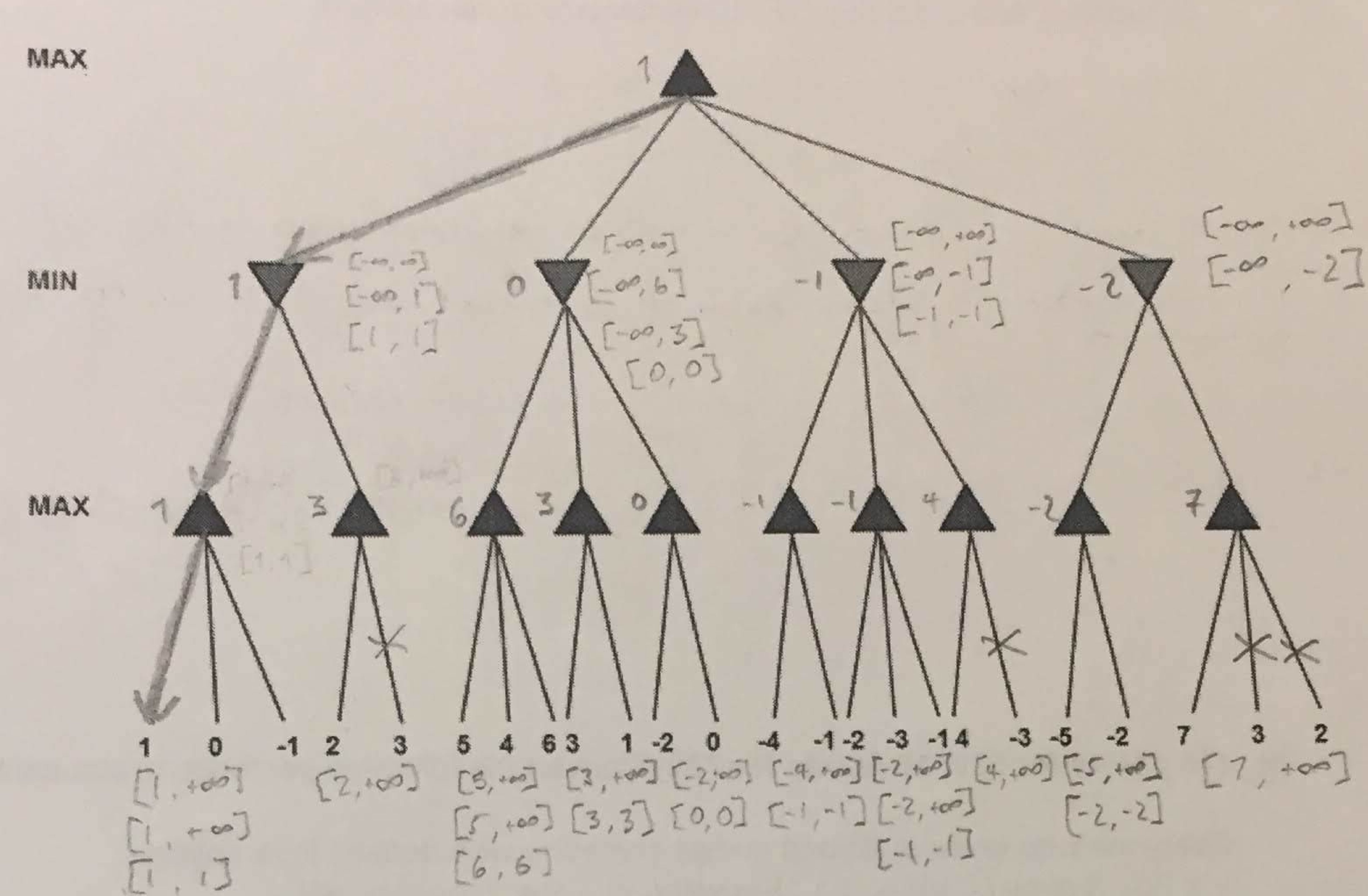
This can be seen on the graph.

[β, α]

← format

3. **Game Tree Search (2-3 parts, 50 points total).**

Consider this game tree:

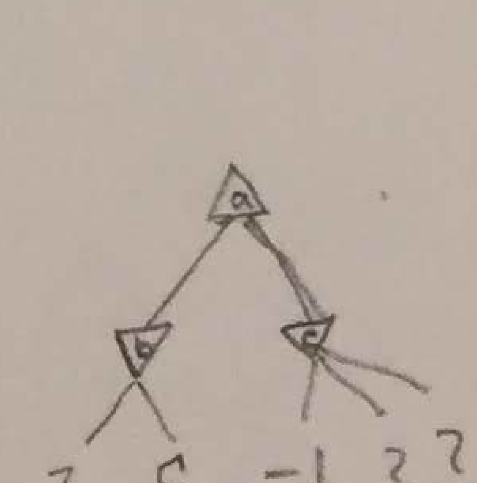


a) **(10 points) Minimax.** Simulate the behavior of the ***minimax algorithm*** on the above game tree.

b) **(30 points for 530, 20 points for 730) Alpha-Beta Pruning.** Simulate the behavior of minimax with alpha-beta (α-β) pruning on the above game tree. Show your work: mark the ~~pruned branches~~ by crossing them out, **indicate which values are** α **and which are** β, and number the static evaluations and all value updates and inequality tests versus α and β, in order.

c) **(10 points, 730 only).** Explain your own words what α and β stand for with respect to a value *v*. Use an illustration as needed.

a and β constrain min and max node selection based on the information known thus far. For example in the following tree, we can guarantee node 'b' will select 3, the min(3,5), Since 'c' sees the -1, the max player knows that if he choses to move from a → c, his score, at best, is worse than the score at move a → b, therefore, one can save time by pruning the other two branches attached to 'c'.



α and β are the best case result from selecting a particular node for max and min respectively.

4. **First-Order Logic (3-4 parts, 50 points total).**

a) **(10 points, 730 only) Clausal form.** Write down and define the steps for converting an arbitrary first-order logic (FOL) sentence into clausal form.

1. Implications Out - Replace $\Leftrightarrow$ + $\Rightarrow$ with $\wedge$, $\vee$, + $\neg$
2. Negations Out - Move $\neg$ inward
3. Standardize Variables Apart - Each quantifier should use a separate variable
4. Existentials Out (Skolemize) - Replace each existential variable with a skolem function
5. Universals Made Implicit - Simply remove universal quantifiers
6. Distribute And over Or - ex. $(A \wedge B) \vee C$ becomes $(A \vee C) \wedge (B \vee C)$

b) **(15 points for 530, 10 points for 730)** Convert the following sentence to clausal form.

Everyone who loves everyone knows someone who doesn't love anyone.
$\forall x\, [\forall y\, .\, Loves(x, y) \rightarrow \exists z\, .\, Knows(x, z) \wedge \forall w\, \neg Loves(z, w)]$

① $\forall x [\forall y\ \neg Loves(x,y) \vee (\exists z\ Knows(x,z) \wedge \forall w\ \neg Loves(z,w))]$

② Negations are fine

③ Variables are already separate

④ $\forall x [\forall y\ \neg Loves(x,y) \vee (Knows(x, K(x)) \wedge \forall w\ \neg Loves(K(x), w))]$

⑤ $\neg Loves(x,y) \vee (Knows(x, K(x)) \wedge \neg Loves(K(x), w))$

⑥ $\boxed{[\neg Loves(x,y) \vee Knows(x, K(x))] \wedge [\neg Loves(x,y) \vee \neg Loves(K(x), w)]}$

c) (10 points) Sentences in FOL. Write the following English sentences in FOL. (Use the predicates indicated and give their meaning in your own words.)

i. (5 points) Everyone who likes everyone has someone they don't love. **Predicates:** Person (x), Likes (x, y), Loves (x, y).

$\forall x\, [\forall y\ \mathrm{Person}(x) \wedge \mathrm{Person}(y) \wedge \mathrm{Likes}(x, y) \rightarrow \exists z\ \mathrm{Person}(z) \wedge \neg \mathrm{Loves}(x, z)$

ii. (5 points) Every consistent heuristic is admissible. **Predicates:** $x \leq y \equiv$ Less-Than-Or-Equal (x, y). **Functions:** h(node), cost (parent-node, child-node), $c_1 - c_2 \equiv$ difference (c_1, c_2)

$\forall n_0, n_1, g\quad \mathrm{LTOE}(\mathrm{diff}(h(n_0), h(n_1)), \mathrm{cost}(n_0, n_1)) \rightarrow \mathrm{LTOE}(h(n_0), \mathrm{cost}(n_0, g))$

c) (15 points for 530, 10 points for 730) Decidability of First-Order Satisfiability and Unsatisfiability: In terms of a first order knowledge base KB and logical sentence α, define the formal languages $L_{FOL-SAT}$ and $L_{FOL-SAT}{}^{C}$ (the language of unsatisfiable sentences), and explain the *decision problem* for membership in each in terms of what entails or does not entail $\perp$ (and from what resolution does or does not derive $\perp$). Are these languages *duals* or *complements*? State the decidability properties for these two languages (decidable, *i.e.*, recursive; semi-decidable, *i.e.*, recursive enumerable but not recursive; or undecidable, *i.e.*, not recursive enumerable). Is the decidability class the same for $L_{FOL-VALID}$ and $L_{FOL-VALID}{}^{C}$? What is the difference between $L_{FOL-SAT}$ and $L_{FOL-VALID}$? Give an example of each.

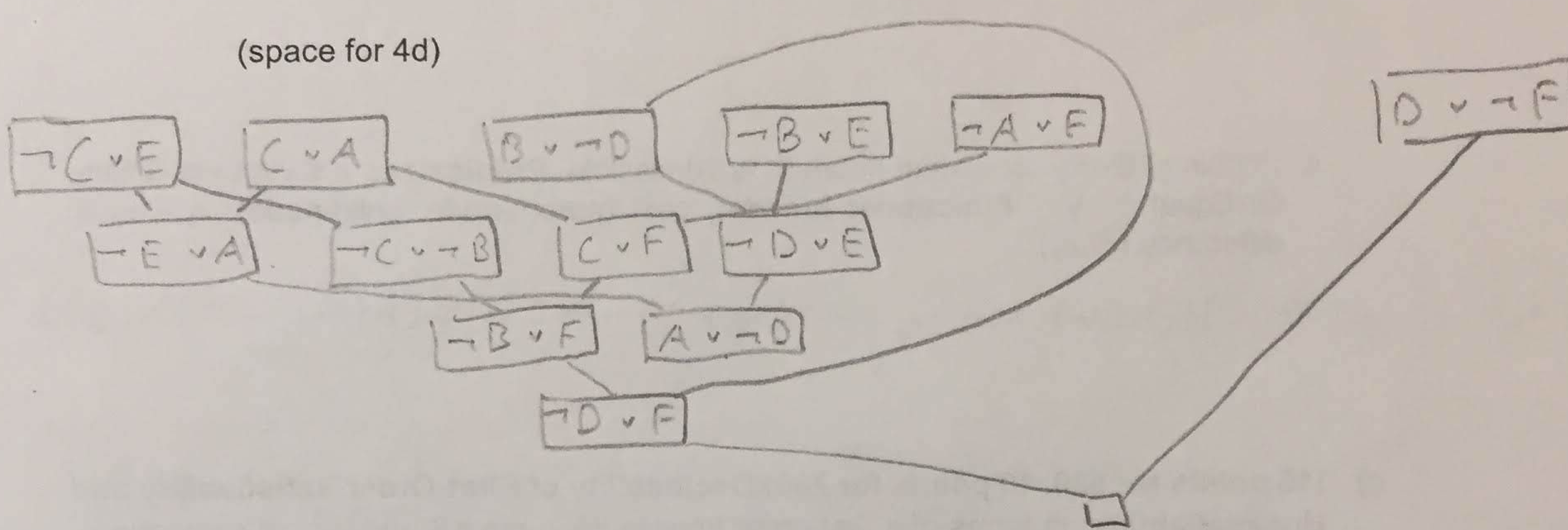
d) (10 points) Resolution theorem proving. (Based on a logic problem by Lewis Carroll.) Consider the following knowledge base:

i) $C \rightarrow \neg E$
ii) $\neg B \rightarrow \neg D$
iii) $\neg C \rightarrow A$
iv) $B \rightarrow E$
v) $A \rightarrow F$

Prove the theorem D → F by **refutation resolution**. Show your work.
dentify examples of *input resolution* and *linear resolution*. Is there a case of *unit resolution*? **Why or why not?**

Hint: Convert every sentence above into a CNF propositional logic statement, e.g., ¬A → ¬B ≡ A ∨ ¬B for part (i). Negate the query (theorem) and derive ⊥.

(space for 4d)



5. **Knowledge Representation and Reasoning (2-3 parts, 50 points total).**

a) **(20 points) Forward *vs.* Backward Chaining.** Consider again the problem of $D \rightarrow F$ as in Problem 4(d):

i) $C \rightarrow \neg E$
ii) $\neg B \rightarrow \neg D$
iii) $\neg C \rightarrow A$
iv) $B \rightarrow E$
v) $A \rightarrow F$

What is different about your resolution-based solution versus "regular" backward chaining? What about versus forward chaining? At what point can you stop? **Illustrate your solution for forward chaining** by numbering the clauses and the query (you may use any number of terms for the query, but indicate exactly what the goal is).

b) **(20 points for 530, 15 points for 730) Inheritance.** Draw an example of an inheritance hierarchy for university courses that includes the classes Course, Online-Course, and MOOC (Massively Open Online Course). Give examples of i) a non-transitive inheritance relationship and ii) a case where minimum path length yields an incorrect result. You may include additional classes to solve 5.b.ii.

c) **(10 points for 530, 5 points for 730) Inheritance.** Define what a *relation* is in an ontology and give an example of a slot/attribute representation of a relationship such as Publisher for a MOOC as in Problem 5(b).

d) **(10 points, 730 only) Successor-State Axioms, the Representational Frame Problem, and the Qualification Problem.** Show how successor state axioms solve the representational frame problem but not the qualification problem. What is the difference between the representational frame problem and qualification problem? **Give an example.**

Extra Credit (20 points).

a) **(10 points) Converting from an adjacency list to an adjacency matrix.** Write pesudocode (identify which programming language this is based on) to scan through an adjacency list (as an associative array or an array of head pointers) and convert it to an adjacency matrix (as a sparse 2-D array).

b) **(10 points) Pathmax.** Explain in your own words the purpose of the pathmax heuristic.

Class Participation (required)

Post your responses to the class message board "Discussion and Questions" under the thread for Homework 3:

a) Give an example of a heuristic that is admissible but not consistent. Create a graph of your own design that illustrates this and show how Algorithm A* with an inconsistent heuristic wastes steps, but still returns the optimal path.
b) Create a heuristic on the same graph that is inadmissible and show how algorithm A (not A*) may fail to find the optimal path.

Extra Credit (10%)

Watch the 28 Aug 2018 TED talk "How AI Can Save our Humanity" by Kai-Fu Lee of Sinovation Ventures at one of these sites:

- https://www.ted.com/talks/kai_fu_lee_how_ai_can_save_our_humanity
- https://youtu.be/ajGgd9Ld-Wc

Post a brief reaction paragraph to the PS3 discussion thread titled "PS3 Extra Credit: reaction commentary". What did you learn, what inferences did you draw, and what critique do you have on this talk in terms of the present and future impact of AI?