6.034 Quiz 2
20 October 2010

Name	Uldis Sturms
email	uldis.sturms@gmail.com

Circle your TA and recitation time **(for 1 point)**, so that we can more easily enter your score in our records and return your quiz to you promptly.

TAs
Martin Couturier
Kenny Donahue
Charles Watts
Gleb Kuznetsov
Kendra Pugh
Mark Seifter
Yuan Shen

Thu	
Time	Instructor
1-2	Bob Berwick
2-3	Bob Berwick
3-4	Bob Berwick

Fri	
Time	Instructor
1-2	Randall Davis
2-3	Randall Davis
3-4	Randall Davis

Problem number	Maximum	Score	Grader
1	50		
2	50		
Total	100		

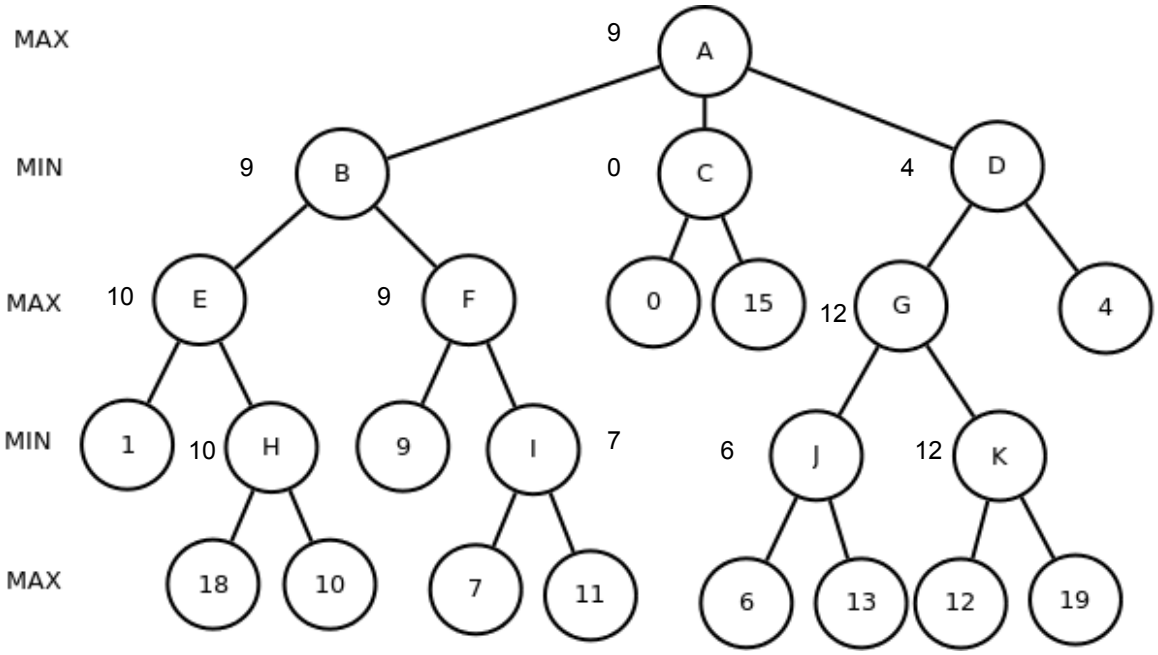
There are 15 pages in this quiz, including this one, but not including blank pages and tear-off sheets. Tear-off sheets are provided at the end with duplicate drawings and data. As always, open book, open notes, open just about everything, including a calculator, but no computers.

Problem 1: Games (50 points)

Part A: Minimax (10 points)

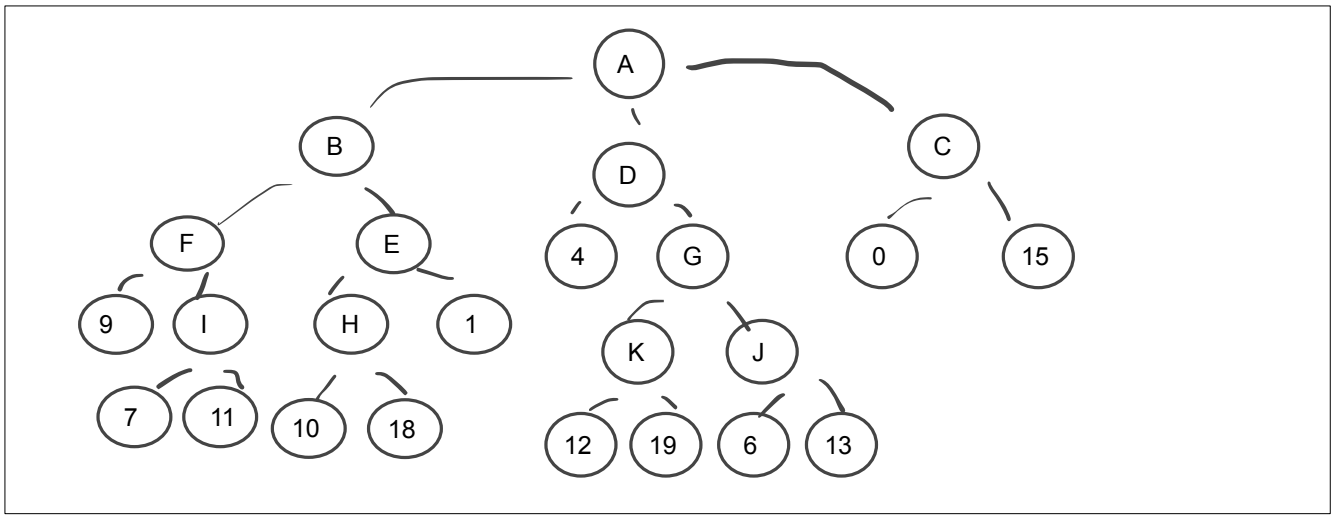
A1: Perform Minimax (5 points)

Perform the minimax algorithm, without alpha-beta, on this tree. **Write the minimax value at each node**.



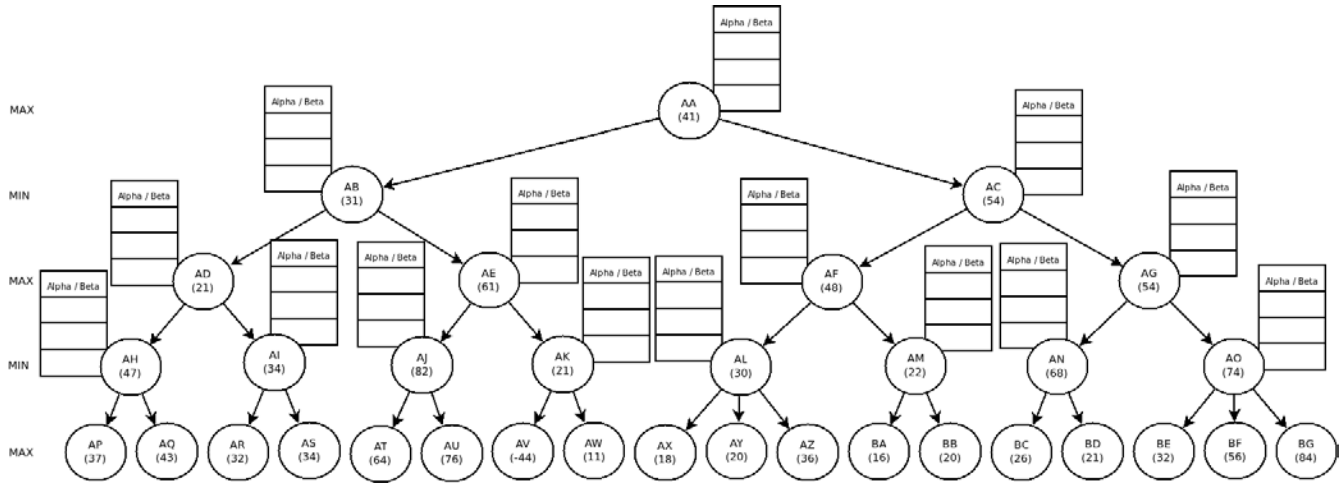
A2: Rotate the Tree (5 points)

Using the minimax calcluations from part A1, without performing any alpha-beta calculation, rotate the children of each node in the above tree **at every level** to ensure **maximum alpha-beta pruning**.



Part B: Alpha Beta (30 points)

While playing a game, you find yourself in the situation indicated by the following tree.



Because this tree is difficult to read, the the tree has been broken into two subtrees, shown on the next two pages. Do all your work on those subtrees, not the tree shown above. The tree above is merely meant to show how the two subtrees connect. Note that the root node AA is shown in both halves.

B1 The Game Tree (7 points)

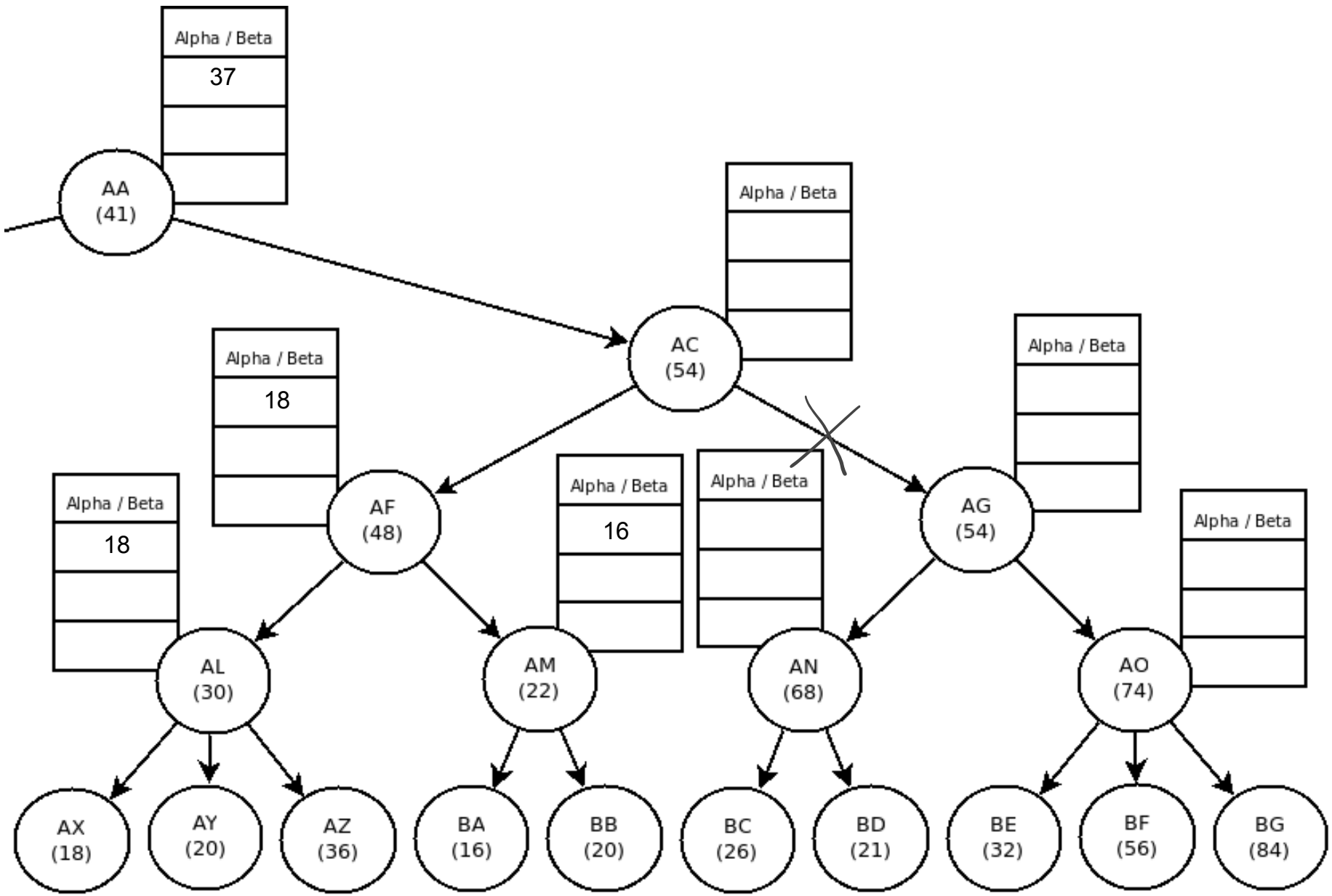
You are provided with static values at all the nodes. **In this part of the problem, you are to ignore the static values at intermediate notes.** You are to use only the values on the leaf nodes.

Perform the alpha-beta algorithm on the graph tree on the next two pages.

1) Use the boxes to help you perform your alpha-beta calculations as needed.
2) **Draw an 'X' through any branch** that is blocked from further consideration by an alpha-beta calculation.
3) Assume no progressive deepening

4

MAX
MIN
MAX
MIN
MAX
Alpha / Beta
37
AA
(41)
AB
(31)
AD
(21)
AE
(61)
AH
(47)
AI
(34)
AJ
(82)
AK
(21)
32
64
AP
(37)
AQ
(43)
AR
(32)
AS
(34)
AT
(64)
AU
(76)
AV
(-44)
AW
(11)


Alpha / Beta
37
AA
(41)
Alpha / Beta
AC
(54)
Alpha / Beta
18
Alpha / Beta
AF
(48)
Alpha / Beta
16
Alpha / Beta
AG
(54)
Alpha / Beta
18
Alpha / Beta
AL
(30)
AM
(22)
AN
(68)
AO
(74)
AX
(18)
AY
(20)
AZ
(36)
BA
(16)
BB
(20)
BC
(26)
BD
(21)
BE
(32)
BF
(56)
BG
(84)

B2 Evaluations (20 points)

How many nodes were statically evaluated and which were they?

 # Evals 11 **Nodes:** AP, AQ, AR, AS, AT, AU, AX, AY, AZ, BA, BB

B3 Move (3 points)

What move is chosen as the best? What node in the deepest level is the maximizing player trying to move towards?

AA--> AB **Moving Toward:** AP

Part C: Progressive Deepening (10 points)

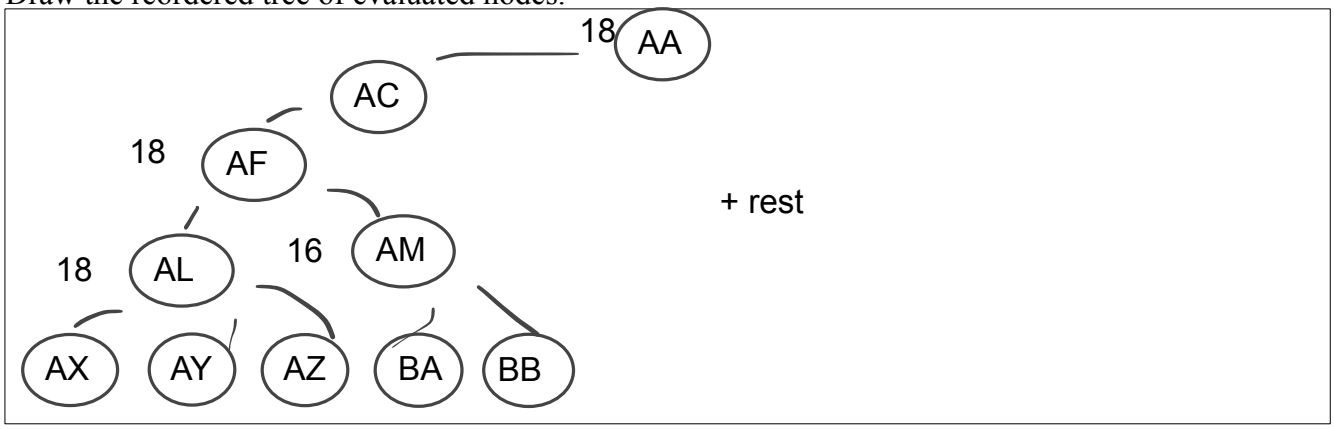
Assume you have the same setup as in **Part B**, except this time, you have an **5-<u>second time-limit for each move.</u>** Each static evaluation takes **<u>1 second</u>**. Assume there is no time associated with building the alpha-beta tree and no time associated with performing the alpha beta search. You are to use progressive deepening and alpha-beta together.

Before performing alpha-beta at any level, use all you know from previous calculations to reorder the nodes at ALL higher levels as much as possible. Naturally, for any reordering, a value produced by search from static values lower down is better than a static value or a value produced by a less-deep search. Assume reordering takes zero time.

Assume that the static values calculated are those shown in the diagram. You are to show us what work would be done by the maximizer at AA during the 5 seconds available to move.

C1 The tree (4 points)

Draw the reordered tree of evaluated nodes.



C2 Evaluations (4 points)

How many nodes were statically evaluated during the 5 seconds available to the maximizer at AA. In which order were they evaluated?

 # Evals 5 **Nodes:** AX, AY, AZ, BA, BB

C3 Move (2 points)

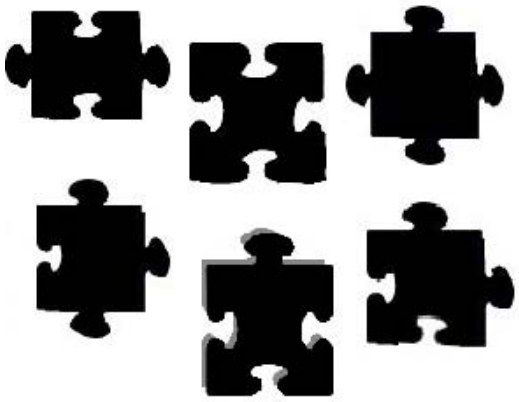
What move is chosen as the best during the available 5 seconds? What is the deepest node that we are trying to move toward? Use all the static evaluation values that the maximizer is able to calculate to make your decision.

AA--> AC **Moving Toward:** AX

Problem 2: Constraint Propagation(50 points)

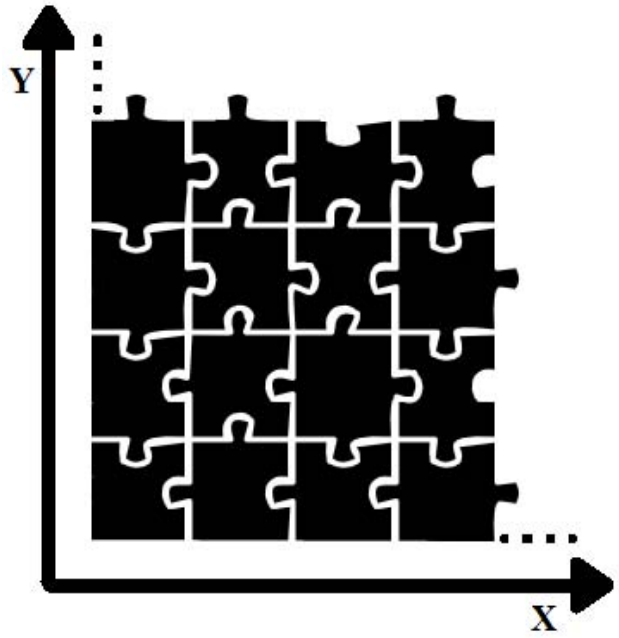
Two of your TA's, Martin and Kenny were given the task of putting together two 20x10, 200 piece jigsaw puzzles. Unfortunately, a child has painted all the pieces black.

The following shapes are **representative** of the shapes of interior pieces (thus, the shapes shown are not all of the shapes involved). Note that each side of the interior pieces has a concavity or a protrusion. In spite of our limited drawing skills, there is just one protrusion shape and one concavity shape, so any protrusion will fit into any concavity.



Edge pieces are like interior pieces except that they have one flat side. Corner pieces are like interior pieces except that they have two adjacent flat sides.

Because all pieces are the same size, the final puzzle can be represented as a grid of (X,Y) coordinates, where each set of coordinates is a location of a puzzle piece. The puzzle is 20 pieces wide by ten pieces tall.



Part A: Setting Constraints (15 points)

While Kenny sets out to solve the puzzle in the typical fashion, Martin sees that this is just another constraint problem and sets out to solve it using what he learned while taking 6.034:

1. He decides that the puzzle locations are the problem variables.

2. He decides to treat all 200 pieces as values (thus, **the pieces, not their shapes, are values**).

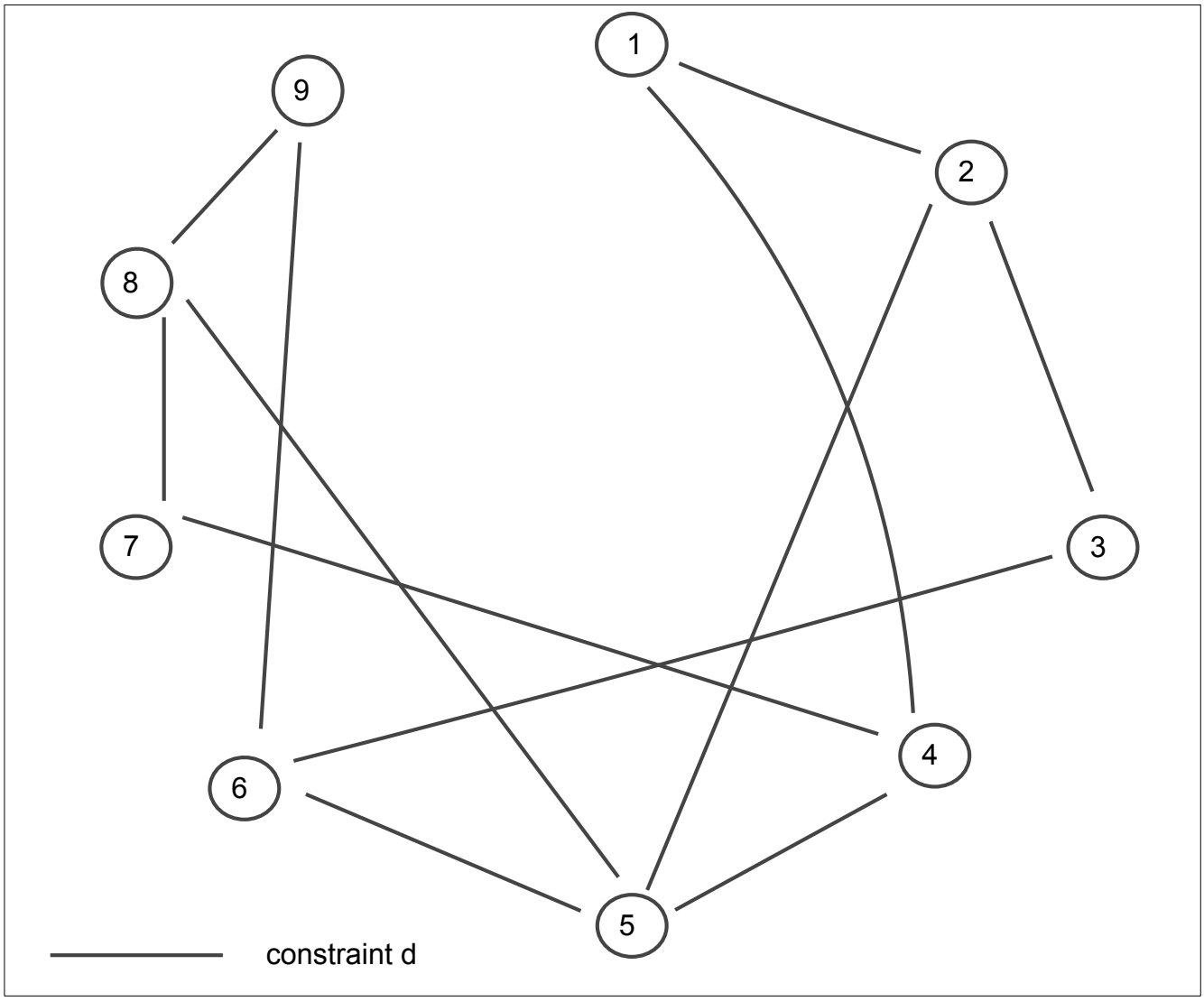
3. **He notes that an oracle has put all 200 pieces in their correct orientation** (thus, each physical piece is one value, not four. Rotation is **not** allowed to simplify your work).

4. He devises this list of constraints:
   a) Piece locations around the edge of the puzzle must contain a piece with one flat edge that is facing outward from the center of the puzzle.
   b) Piece locations at the corners must contain a piece with two flat edges that are facing outward from the center of the puzzle.
   c) Piece locations not around the edge of the puzzle must not contain a puzzle piece with any flat edges
   d) All four sides of a puzzle piece must be compatible with all of the pieces around it.
   e) No two locations can contain the same piece.

A1 (6 points)

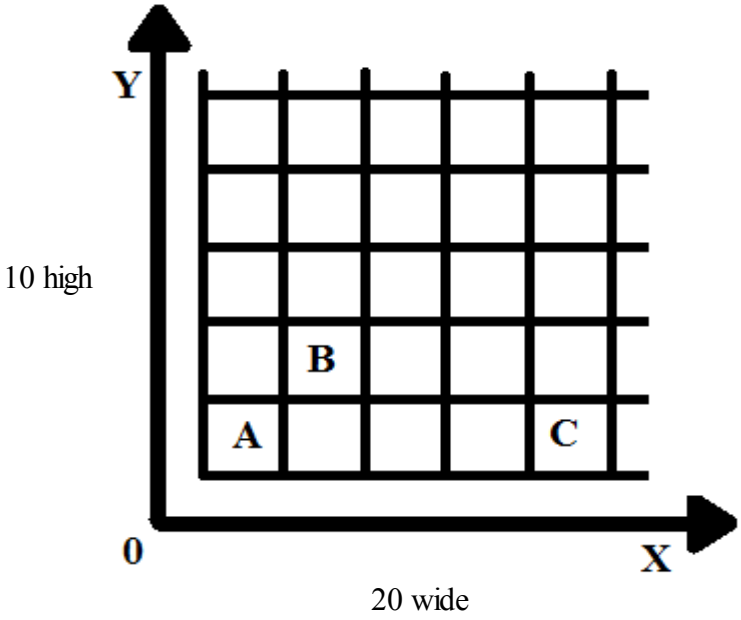
Draw a graph of the variable nodes for the subset of the puzzle containing the 9 piece locations shown below. Indicate constraints between pairs of variable nodes by drawing **one line for each** such pairwise constraint. Suggestion: arrange all your nodes in a circle.

1	**2**	**3**
4	**5**	**6**
7	**8**	**9**



A2 (9 points)

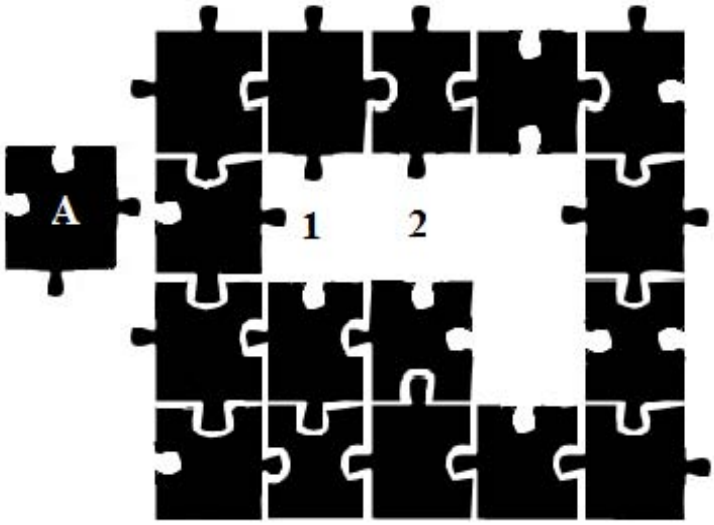
Thinking that constraints 1-4 can serve as a good starting point to solve any puzzle, and remembering that oracle has spoken, so **all pieces are provided in the correct orientation**, Martin decides to use the constraints to simplify the starting domains for each of the piece locations. Describe the domains for A, B, C after the initial constraints have been applied.



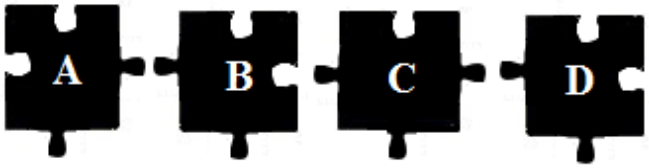
Piece Location	Description of pieces in the domain	Size of domain
A	a piece with 2 flat edges facing outward of the centre	1
B	a piece with no flat edges	200 - (28*2) = 144
C	a piece with1 flat edge facing outward of the centre	18

Part B: Checking Neighbors (8 points)

While solving the puzzle, Martin places piece **A** in location **1**. He is running the Domain Reduction Algorithm using **forward checking**. The domain of location **2** is shown as it was before **A** was placed.



Domain of location **2** before placement of **A :**



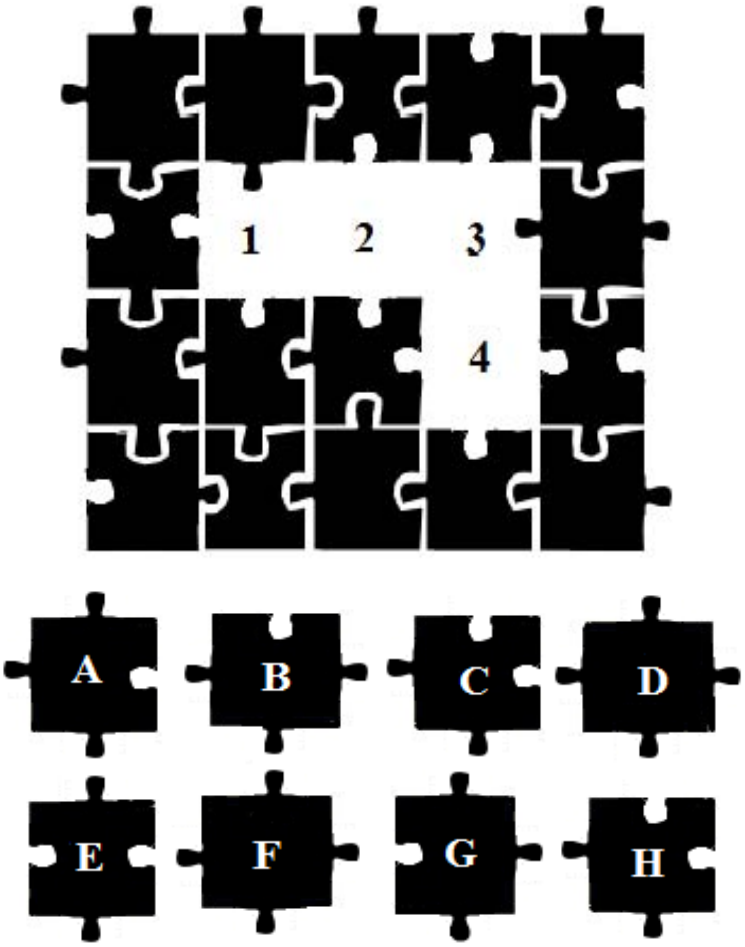
Again, noting that no rotation is allowed, and that, all protrusions fit successfully into all concavities in spite of our limited drawing skills, what happens after he places piece **A** and why?

Constraints for location 2 cannot be satisfied as the only piece A that would satisfy them is already used and any one piece can only be used for one location. Piece A needs to be removed and piece that precedes it or we need to extend our domain for location 2 somehow.

Part C: Neighbors of Neighbors (27 points)

C1 (5 points)

Nearing completion of the puzzle, Martin encounters a cluster of 4 empty piece locations. (there are 4 other empty spaces elsewhere). He still has a total of 8 pieces. Considering only constraints imposed by the neighbors of the 4 empty piece locations shown, fill in the domain table for each of the piece locations. Once again note that **all pieces are shown in the only orientation allowed**,



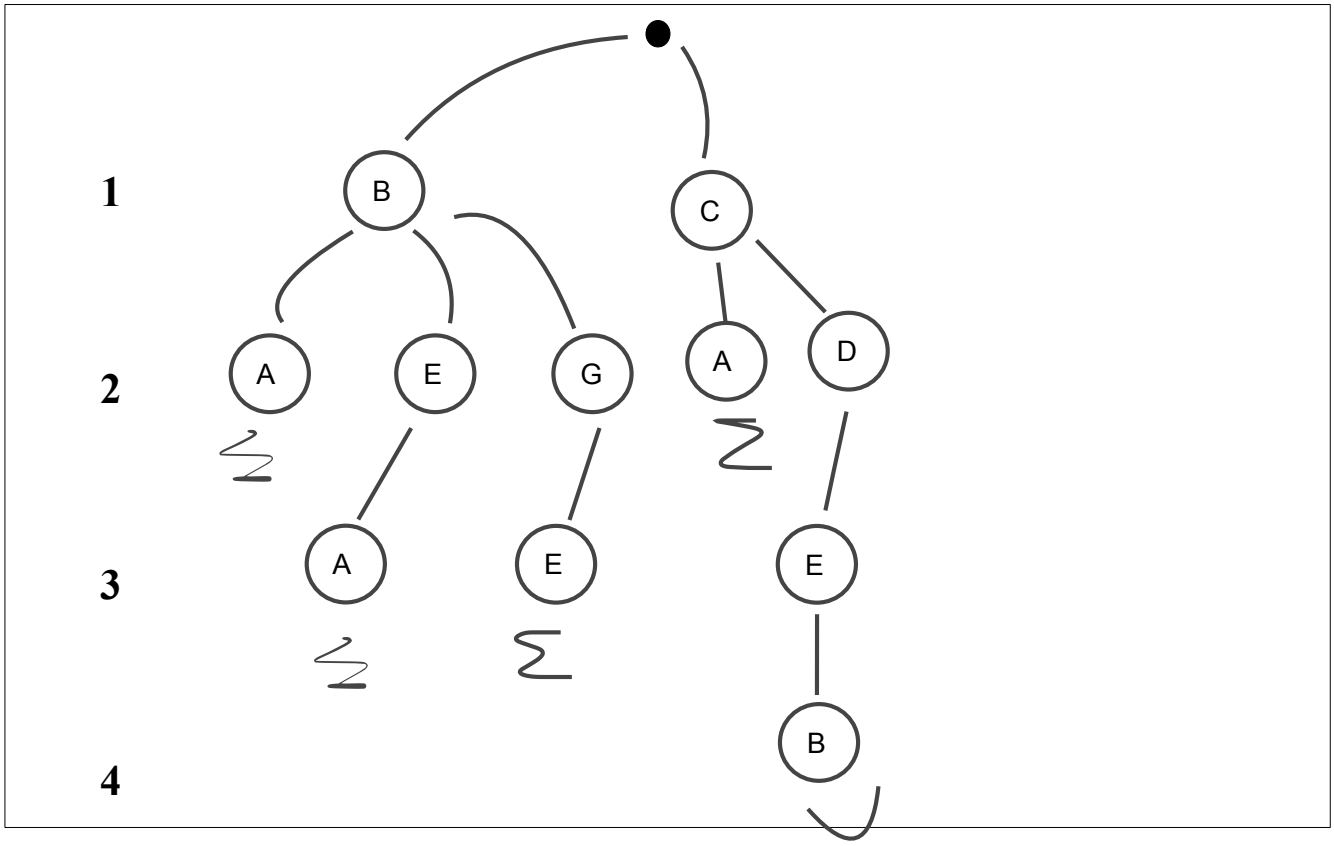
Variable	Domain
1	B, C, H
2	A, D, E, F, G
3	A, E
4	B, D, F

C2 (10 points)

Once again noting that all pieces are shown in the only orientation allowed, complete this section of the puzzle using **depth-first search with forward checking and propagation through singleton domains**. Again assume that, all protrusions fit successfully into all concavities.

Assinging the variables, 1, 2, 3, 4, in numerical order, draw the search tree that is evaluated using forward checking with **singleton propagation. Be sure to try assignments in lexicographic order.**

Work carefully, because we will award points only for the fraction of the search done right, so early mistakes will be worse that late mistakes or not completing the search.



C3 (4 points)

What is the final solution to this region of the puzzle?

Variable	Value
1	C
2	D
3	E
4	B

C4 (3 points)

List the partial paths in your tree where constraint checking fails.

B, A
B, E, A
B, G, E
C, A

C5 (5 points)

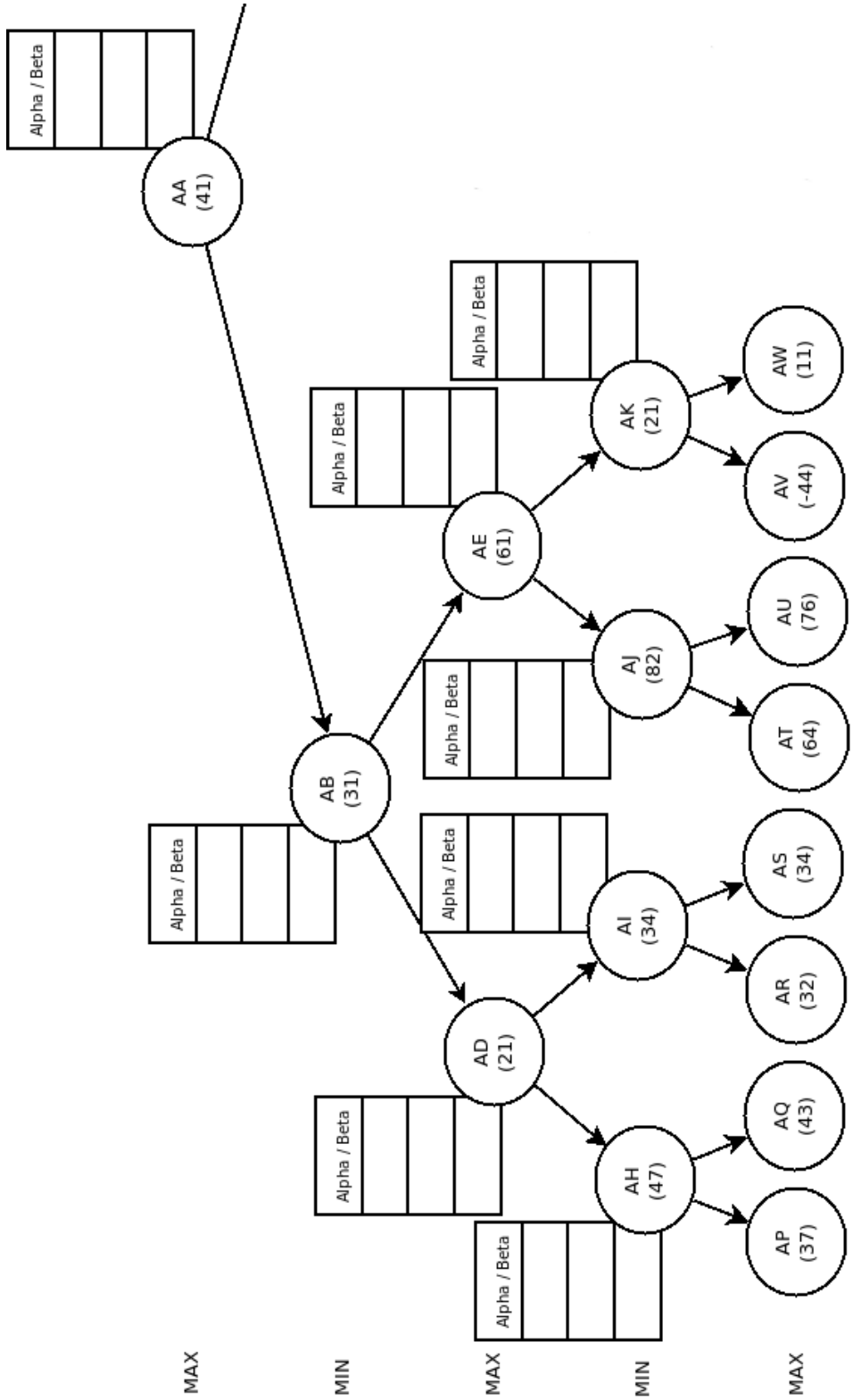
Looking at your answer to **C1,** how might we adjust the search to find solutions with less backtracking?

Eliminate items from domains that do not satisfy any previous domain constraints, e.g.,
make variable 4 = {B}, since nor D nor F will satisfy either A or E constraints

Blank page

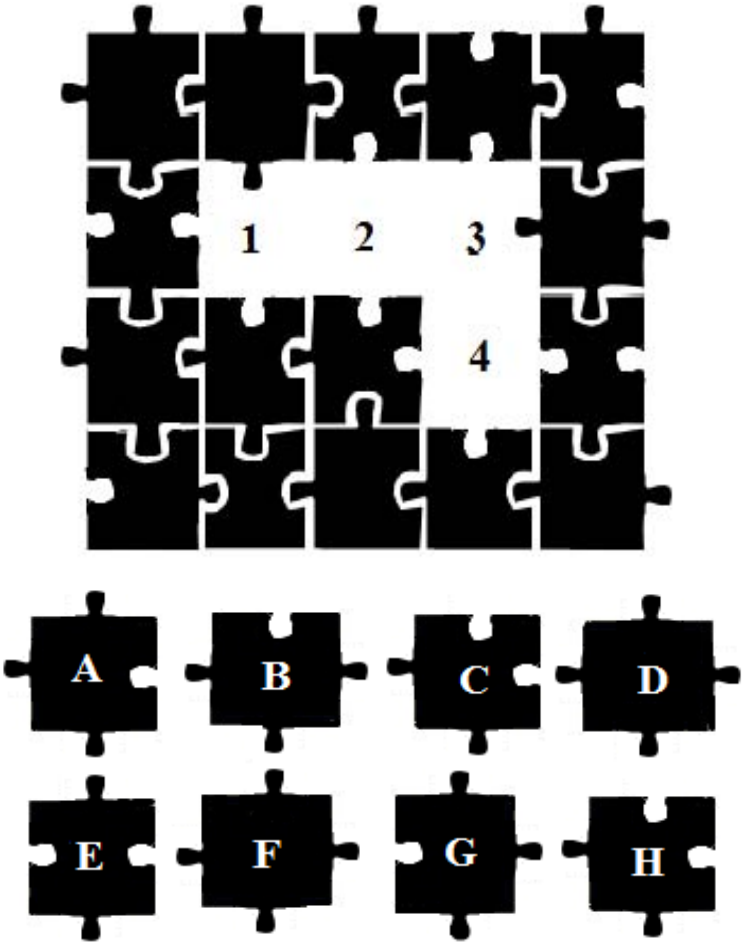
Tear off sheet, you need not hand this in

Problem 1, Part B, Left Side



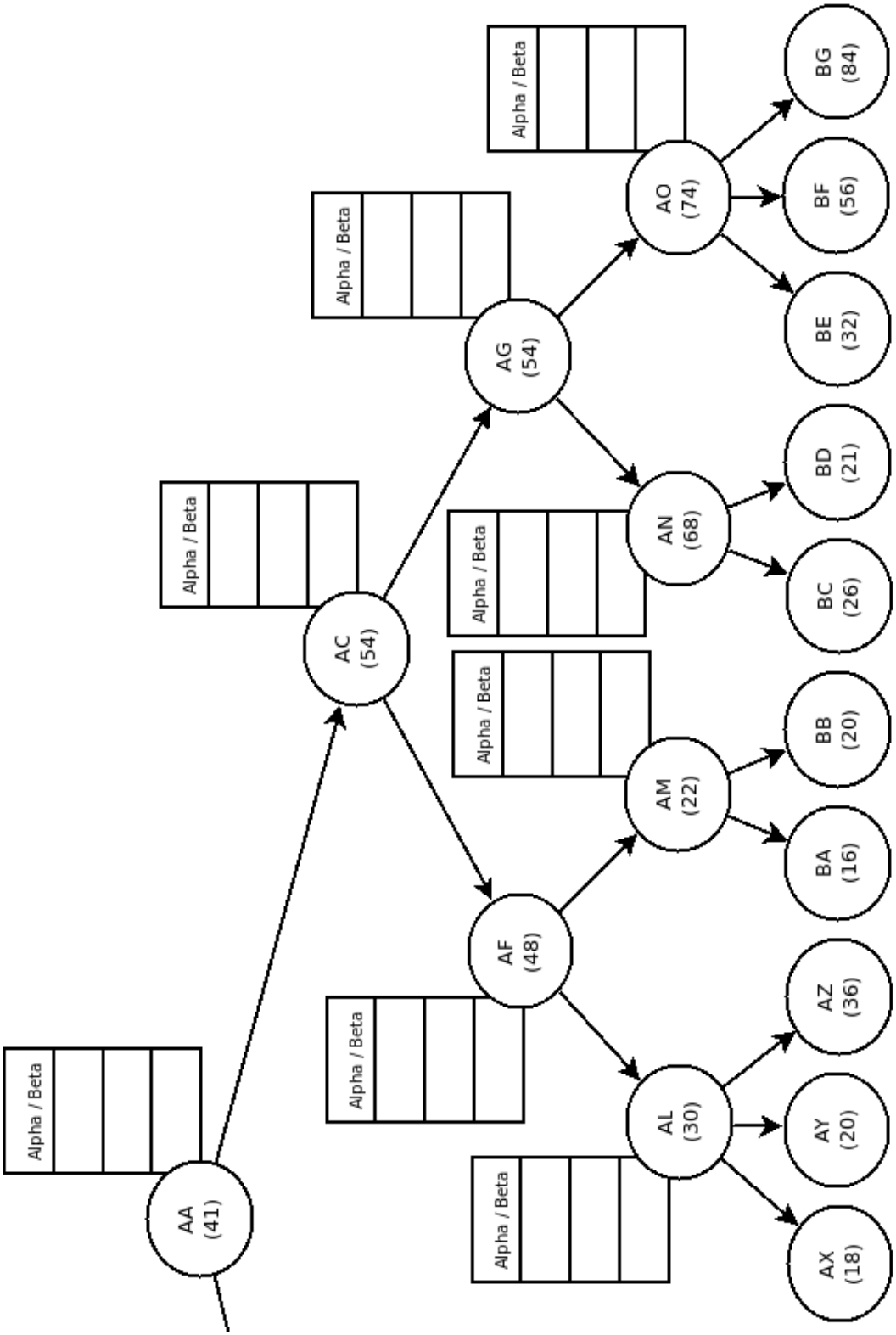
Tear off sheet, you need not hand this in

Problem 2, Part B



Tear off sheet, you need not hand this in

Problem 1, Part B, Right Side



Blank, final page

6.034 Artificial Intelligence

Fall 2010