6.034 Practice with Constraint Satisfaction Problems

The two problems that follow have been taken from previous 6.034 quizzes. They have been chosen because they are most effective at demonstrating the various methods of solving constraint satisfaction problems.

These problems have been largely unedited, except some sections have been cut, and we have introduced **domain worksheets** as a way of recording your progress during constraint propagation. Domain worksheets should be useful to you, especially in a quiz setting, as they help you to demonstrate what you know.

The quiz problems appear first, followed by their solutions.

Part I: Problems

The Time Traveler's Convention (2009 Q2)

The MIT Time Travel Society (MITTTS) has invited seven famous historical figures to each give a lecture at the annual MITTTS convention, and you've been asked to create a schedule for them. Unfortunately, there are only four time slots available (1pm - 4pm), and you discover that there are some restrictions on how you can schedule the lectures and keep all the convention attendees happy. For instance, physics students will be disappointed if you schedule Niels Bohr and Isaac Newton to speak during the same time slot, because those students were hoping to attend both of those lectures.

After talking to some students who are planning to attend this year's convention, you determine that they fall into certain groups, each of which wants to be able to see some subset of the time-traveling speakers. (Fortunately, each student identifies with at most one of the groups.) You write down everything you know:

The list of guest lecturers consists of Alan **T**uring, Ada **L**ovelace, Niels **B**ohr, Marie **C**urie, **S**ocrates, **P**ythagoas, and Isaac **N**ewton.

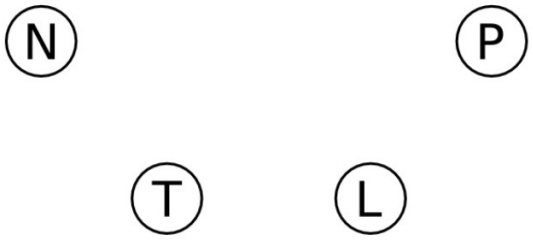
1. **T**uring has to get home early to help win World War II, so he can only be assigned to the 1pm slot.
2. The Course VIII students want to see the physicists: **B**ohr, **C**urie, and **N**ewton.
3. The Course XVIII students want to see the mathematicians: **L**ovelace, **P**ythagoras, and **N**ewton.
4. The members of the Ancient Greece Club wants to see the ancient Greeks: **S**ocrates and **P**ythagoras.
5. The visiting Wellesley students want to see the female speakers: **L**ovelace and **C**urie.
6. The CME students want to see the British speakers: **T**uring, **L**ovelace, and **N**ewton.
7. Finally, you decide that you will be happy if and only if you get to see both **C**urie and **P**ythagoras. (Yes, even if you belong to one or more of the groups above.)

Part A:

Diagram these constraints by drawing a line between the initials of each pair of guests who cannot share a time slot.

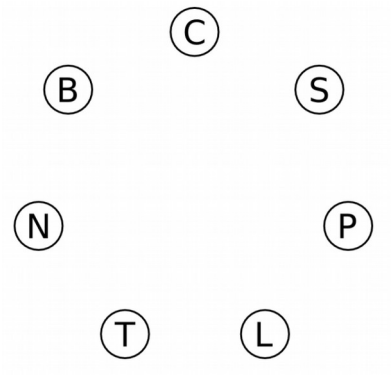
Part B

Search for a solution using **depth-first search only**—without any forward checking or propagation. The only check is to make sure that each new assignment violates no constraint with any previous assignment. As a tiebreaker, assign a lecturer to the earliest available timeslot. **Continue up to the first time you try and fail to assign any time to Newton and must backtrack, at which point you give up and move on to Part C to try a more sophisticated approach.**
Show your answers on the next two pages.



Show your work by (1) filling out the domain worksheet on this page and (2) drawing the search tree on the next page.

Constraint graph for this problem



Domains for this problem

T	1
L	1 2 3 4
B	1 2 3 4
C	1 2 3 4
S	1 2 3 4
P	1 2 3 4
N	1 2 3 4

Fill out this worksheet as you draw your search tree. There may be more rows than you need.

1. Every time you **assign a variable** or **remove a variable from the propagation queue**, fill out a new row in the table. (The same variable might appear in more than one row, especially if you have to backtrack.)
2. In that row, indicate **which variable you assigned or de-queued**; write its **assigned** value if it has one (e.g. X=x), otherwise just write its name (X). In the second column, list the **values that were just eliminated from neighboring variables** as a result. If no values were just eliminated, write **NONE** instead.
3. If your search has to backtrack after assigning or de-queuing a variable: first, **finish listing** all values eliminated from neighboring variables in the current row. Next, check the backtrack box in that row. Then, continue with the next assignment in the following row as usual.
4. At some point, you might add several variables to your propagation queue at once. Break ties by adding variables to your propagation queue **in alphabetical order**.

	Var assigned or de-queued	List all values eliminated from neighboring variables	Back track ?		Var assigned or de-queued	List all values eliminated from neighboring variables	Back track ?
ex	X	Y ≠ 3, 4 Z ≠ 3 (example)	☑	10			☐
1			☐	11			☐
2			☐	12			☐
3			☐	13			☐
4			☐	14			☐
5			☐	15			☐
6			☐	16			☐
7			☐	17			☐
8			☐	18			☐
9			☐	19			☐

Draw your search tree for part B below.

T

L

B

C

S

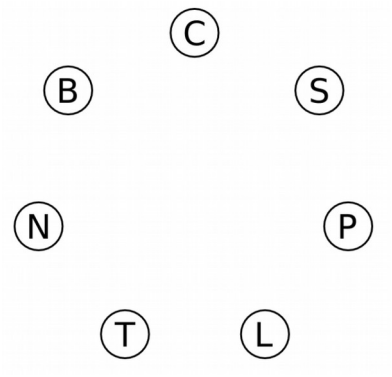
P

N

Part C

You're not fond of backtracking, so rather than wait and see how much backtracking you'll have to do, you decide to use **depth first search with forward checking and propagation through singletons (propagation through domains reduced to size 1)** to solve the problem. As before, show your work by filling out the domain worksheet below and drawing the search tree on the following page.

Constraint graph for this problem



Domains for this problem

T	1
L	1 2 3 4
B	1 2 3 4
C	1 2 3 4
S	1 2 3 4
P	1 2 3 4
N	1 2 3 4

Fill out this worksheet as you draw your search tree. There may be more rows than you need.

5. Every time you **assign a variable** or **remove a variable from the propagation queue**, fill out a new row in the table. (The same variable might appear in more than one row, especially if you have to backtrack.)
6. In that row, indicate **which variable you assigned or de-queued**; write its **assigned** value if it has one (e.g. X=x), otherwise just write its name (X). In the second column, list the **values that were just eliminated from neighboring variables** as a result. If no values were just eliminated, write **NONE** instead.
7. If your search has to backtrack after assigning or de-queuing a variable: first, **finish listing** all values eliminated from neighboring variables in the current row. Next, check the backtrack box in that row. Then, continue with the next assignment in the following row as usual.
8. At some point, you might add several variables to your propagation queue at once. Break ties by adding variables to your propagation queue **in alphabetical order**.

	Var assigned or de-queued	List all values eliminated from neighboring variables	Back track ?		Var assigned or de-queued	List all values eliminated from neighboring variables	Back track ?
ex	X=3	Y ≠ 3, 4 Z ≠ 3 (example)	☑	7			☐
1			☐	8			☐
2			☐	9			☐
3			☐	10			☐
4			☐	11			☐
5			☐	12			☐
6			☐	13			☐

Draw your search tree for Part C below.

T

L

B

C

S

P

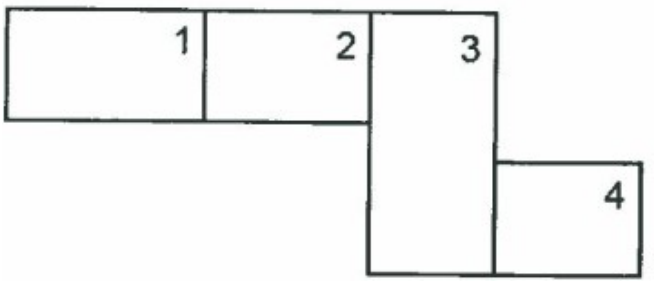
N

The Zoo in Killian Court (2011 Q2)

In honor of MIT 150, MIT has decided to open a new zoo in Killian Court. They have obtained seven animals and built four enclosures. Because there are more animals than enclosures, some animals have to be in the same enclosures as others. However, the animals are very picky about who they live with. The MIT administration is having trouble assigning animals to enclosures, just as they often have trouble assigning students to residences. As you have taken 6.034, they have asked you to plan where each animal goes.

The animals chosen are a **LION, ANTELOPE, HYENA, EVIL LION, HORNBILL, MEERKAT**, and **BOAR.**

They have given you the plans of the zoo layout.



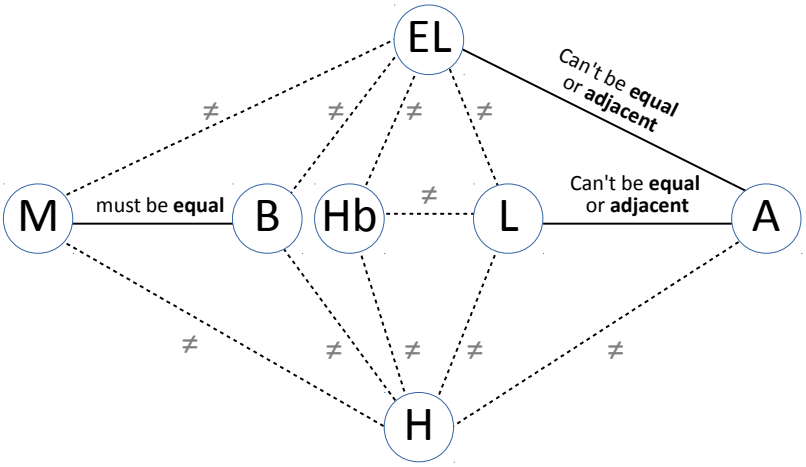
Each numbered area is a zoo enclosure. Multiple animals can go into the same enclosure, and not all enclosures have to be filled.

Each animal has restrictions about where it can be placed.

1. The **LION** and the **EVIL LION** hate each other, and do not want to be in the same enclosure.
2. The **MEERKAT** and **BOAR** are best friends, and have to be in the same enclosure.
3. The **HYENA** smells bad. Only the **EVIL LION** will share his enclosure.
4. The **EVIL LION** wants to eat the **MEERKAT**, **BOAR**, and **HORNBILL**.
5. The **LION** and the **EVIL LION** want to eat the **ANTELOPE** so badly that the **ANTELOPE** cannot be in either the same enclosure **or in an enclosure adjacent to** the **LION** or **EVIL LION**.
6. The **LION** annoys the **HORNBILL**, so the **HORNBILL** doesn't want to be in the **LION**'s enclosure.
7. The **LION** is king, so he wants to be in enclosure 1.

Using the reduced domains provided below, find one solution using **depth first search with forward checking and propagation through <u>domains reduced by any number of values</u> (propagation through reduced domains.)** Show your work by filling out the domain worksheet on this page and **drawing the search tree** on the next page. Break ties in numerical order (1,2,3,4).

Constraint graph for this problem



Domains for this problem

L	1
Hb	2 3 4
A	3 4
EL	2 3 4
H	2 3 4
M	1 2 3 4
B	1 2 3 4

<u>Reminder</u>: At some point, you might add several variables to your propagation queue at once. Break ties by adding variables to your propagation queue **<u>in alphabetical order</u>.**

	Var assigned or de-queued	List all values eliminated from neighboring variables	Back track ?
ex	X	Y ≠ 3, 4 Z ≠ 3 (example)	☑
1			☐
2			☐
3			☐
4			☐
5			☐
6			☐
7			☐
8			☐
9			☐
10			☐

	Var assigned or de-queued	List all values eliminated from neighboring variables	Back track ?
11			☐
12			☐
13			☐
14			☐
15			☐
16			☐
17			☐
18			☐
19			☐
20			☐
21			☐

LION	
HORNBILL	
ANTELOPE	
EVIL LION	
HYENA	
MEERKAT	
BOAR	

Part II: Solutions

SOLUTION: The Time Traveler's Convention (2009 Q2)

The MIT Time Travel Society (MITTTS) has invited seven famous historical figures to each give a lecture at the annual MITTTS convention, and you've been asked to create a schedule for them. Unfortunately, there are only four time slots available (1pm - 4pm), and you discover that there are some restrictions on how you can schedule the lectures and keep all the convention attendees happy. For instance, physics students will be disappointed if you schedule Niels Bohr and Isaac Newton to speak during the same time slot, because those students were hoping to attend both of those lectures.

After talking to some students who are planning to attend this year's convention, you determine that they fall into certain groups, each of which wants to be able to see some subset of the time-traveling speakers. (Fortunately, each student identifies with at most one of the groups.) You write down everything you know:

> The list of guest lecturers consists of Alan **T**uring, Ada **L**ovelace, Niels **B**ohr, Marie **C**urie, **S**ocrates, **P**ythagoas, and Isaac **N**ewton.

1. **T**uring has to get home early to help win World War II, so he can only be assigned to the 1pm slot.
2. The Course VIII students want to see the physicists: **B**ohr, **C**urie, and **N**ewton.
3. The Course XVIII students want to see the mathematicians: **L**ovelace, **P**ythagoras, and **N**ewton.
4. The members of the Ancient Greece Club wants to see the ancient Greeks: **S**ocrates and **P**ythagoras.
5. The visiting Wellesley students want to see the female speakers: **L**ovelace and **C**urie.
6. The CME students want to see the British speakers: **T**uring, **L**ovelace, and **N**ewton.
7. Finally, you decide that you will be happy if and only if you get to see both **C**urie and **P**ythagoras. (Yes, even if you belong to one or more of the groups above.)

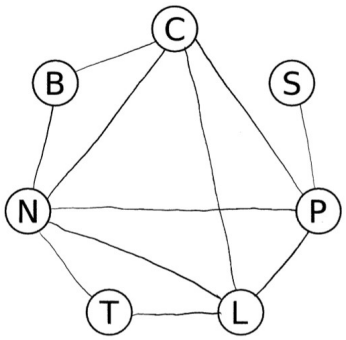
Part A:

Diagram these constraints by drawing a line between the initials of each pair of guests who cannot share a time slot.

Search for a solution using **depth-first search only**—without any forward checking or propagation. The only check is to make sure that each new assignment violates no constraint with any previous assignment. As a tiebreaker, assign a lecturer to the earliest available timeslot. **Continue up to the first time you try and fail to assign any time to Newton and must backtrack, at which point you give up and move on to Part C to try a more sophisticated approach.**
Show your answers on the next two pages.

Show your work by (1) filling out the domain worksheet on this page and (2) drawing the search tree on the next page.

Constraint graph for this problem



Domains for this problem

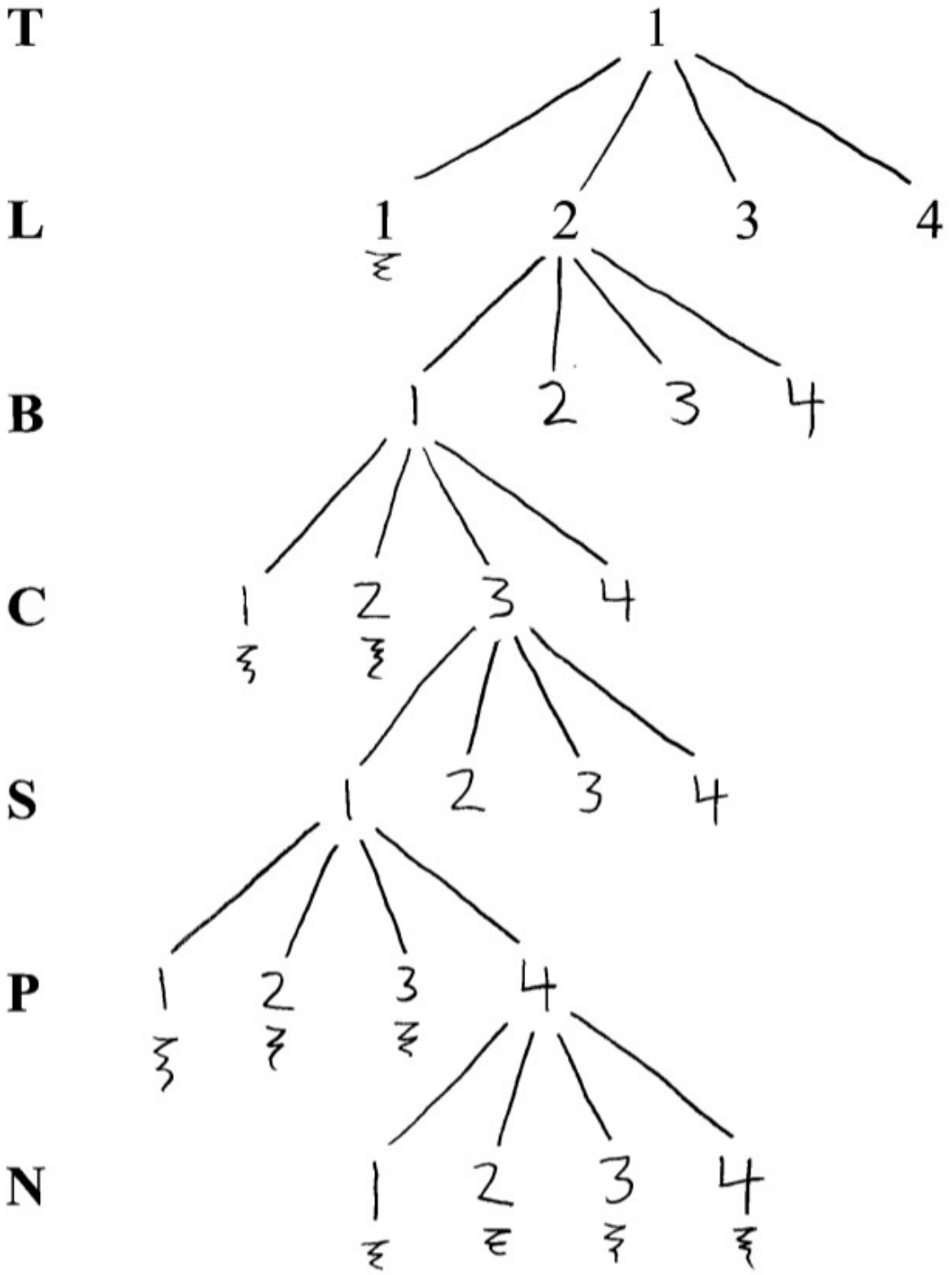
T	1
L	1 2 3 4
B	1 2 3 4
C	1 2 3 4
S	1 2 3 4
P	1 2 3 4
N	1 2 3 4

Fill out this worksheet as you draw your search tree. There may be more rows than you need.

1. Every time you **assign a variable** or **remove a variable from the propagation queue**, fill out a new row in the table. (The same variable might appear in more than one row, especially if you have to backtrack.)
2. In that row, indicate **which variable you assigned or de-queued**; write its **assigned** value if it has one (e.g. X=x), otherwise just write its name (X). In the second column, list the **values that were just eliminated from neighboring variables** as a result. If no values were just eliminated, write **NONE** instead.
3. If your search has to backtrack after assigning or de-queuing a variable: first, **finish listing** all values eliminated from neighboring variables in the current row. Next, check the backtrack box in that row. Then, continue with the next assignment in the following row as usual.
4. At some point, you might add several variables to your propagation queue at once. Break ties by adding variables to your propagation queue **in alphabetical order**.

	Var assigned or de-queued	List all values eliminated from neighboring variables	Back track ?		Var assigned or de-queued	List all values eliminated from neighboring variables	Back track ?
ex	X	Y ≠ 3, 4 Z ≠ 3 (example)	☑	10	P = 2	NONE	☑
1	T = 1	NONE	☐	11	P = 3	NONE	☑
2	L = 1	NONE	☑	12	P = 4	NONE	☐
3	L = 2	NONE	☐	13	N = 1	NONE	☑
4	B = 1	NONE	☐	14	N = 2	NONE	☑
5	C = 1	NONE	☑	15	N = 3	NONE	☑
6	C = 2	NONE	☑	16	N = 4	NONE	☑
7	C = 3	NONE	☐	17			☐
8	S = 1	NONE	☐	18			☐
9	P = 1	NONE	☑	19			☐

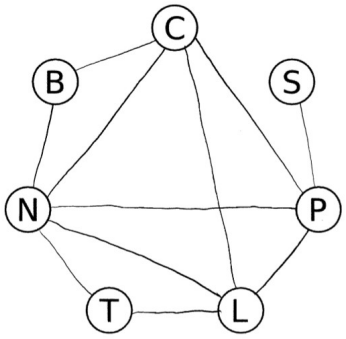
Draw your search tree for part B below.



Part C

You're not fond of backtracking, so rather than wait and see how much backtracking you'll have to do, you decide to use **depth first search with forward checking and propagation through singletons (propagation through domains reduced to size 1)** to solve the problem. As before, show your work by filling out the domain worksheet below and drawing the search tree on the following page.

Constraint graph for this problem



Domains for this problem

T	1
L	1 2 3 4
B	1 2 3 4
C	1 2 3 4
S	1 2 3 4
P	1 2 3 4
N	1 2 3 4

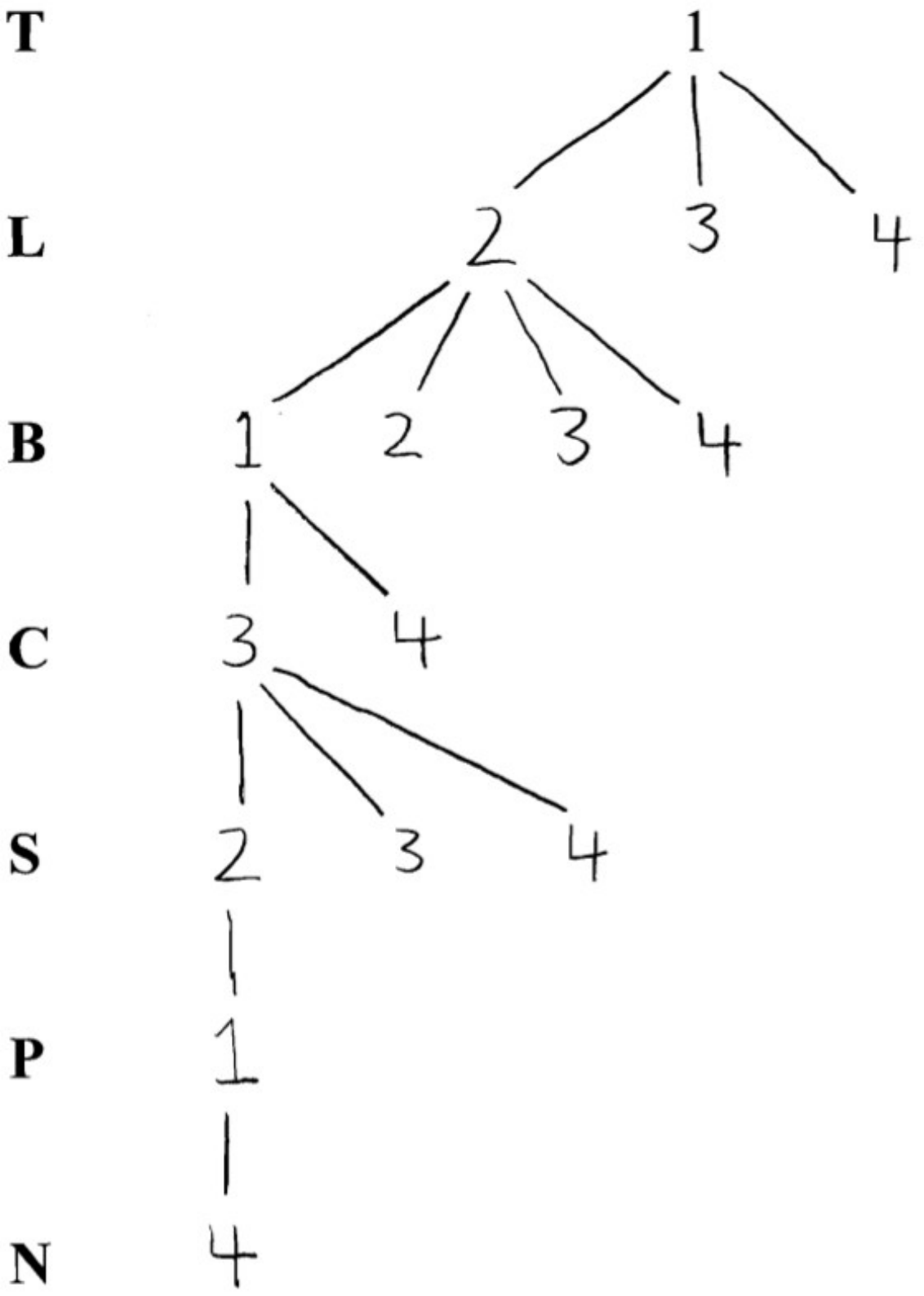
Fill out this worksheet as you draw your search tree. There may be more rows than you need.

1. Every time you **assign a variable** or **remove a variable from the propagation queue**, fill out a new row in the table. (The same variable might appear in more than one row, especially if you have to backtrack.)
2. In that row, indicate **which variable you assigned or de-queued**; write its **assigned** value if it has one (e.g. X=x), otherwise just write its name (X). In the second column, list the **values that were just eliminated from neighboring variables** as a result. If no values were just eliminated, write **NONE** instead.
3. If your search has to backtrack after assigning or de-queuing a variable: first, **finish listing** all values eliminated from neighboring variables in the current row. Next, check the backtrack box in that row. Then, continue with the next assignment in the following row as usual.
4. At some point, you might add several variables to your propagation queue at once. Break ties by adding variables to your propagation queue **in alphabetical order**.

	Var assigned or de-queued	List all values eliminated from neighboring variables	Back track ?
ex	X	Y ≠ 3, 4 Z ≠ 3 (example)	☑
1	T = 1	L ≠ 1 N ≠ 1	☐
2	L = 2	P ≠ 2 N ≠ 2 C ≠ 2	☐
3	B = 1	C ≠ 1	☐
4	C = 3	N ≠ 3 P ≠ 3	☐
5	N	P ≠ 4	☐
6	P	S ≠ 1	☐

	Var assigned or de-queued	List all values eliminated from neighboring variables	Back track ?
7	S = 2	NONE	☐
8	P = 1	NONE	☐
9	N = 4	NONE	☐
10			☐
11			☐
12			☐
13			☐

Draw your search tree for Part C below.

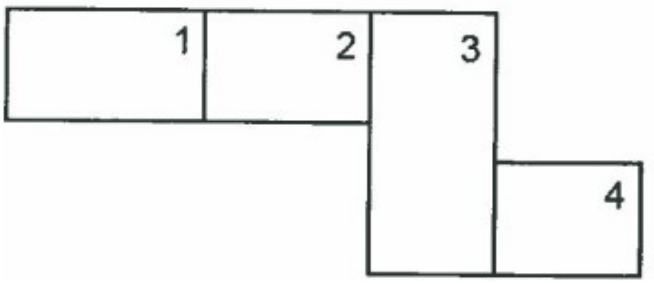


SOLUTION: The Zoo in Killian Court (2011 Q2)

In honor of MIT 150, MIT has decided to open a new zoo in Killian Court. They have obtained seven animals and built four enclosures. Because there are more animals than enclosures, some animals have to be in the same enclosures as others. However, the animals are very picky about who they live with. The MIT administration is having trouble assigning animals to enclosures, just as they often have trouble assigning students to residences. As you have taken 6.034, they have asked you to plan where each animal goes.

The animals chosen are a **LION, ANTELOPE, HYENA, EVIL LION, HORNBILL, MEERKAT**, and **BOAR.**

They have given you the plans of the zoo layout.



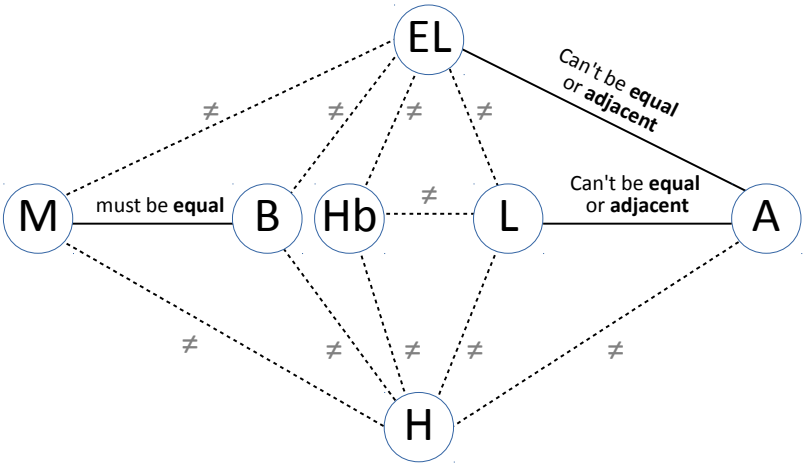
Each numbered area is a zoo enclosure. Multiple animals can go into the same enclosure, and not all enclosures have to be filled.

Each animal has restrictions about where it can be placed.

1. The **LION** and the **EVIL LION** hate each other, and do not want to be in the same enclosure.
2. The **MEERKAT** and **BOAR** are best friends, and have to be in the same enclosure.
3. The **HYENA** smells bad. Only the **EVIL LION** will share his enclosure.
4. The **EVIL LION** wants to eat the **MEERKAT**, **BOAR**, and **HORNBILL**.
5. The **LION** and the **EVIL LION** want to eat the **ANTELOPE** so badly that the **ANTELOPE** cannot be in either the same enclosure **or in an enclosure adjacent to** the **LION** or **EVIL LION**.
6. The **LION** annoys the **HORNBILL**, so the **HORNBILL** doesn't want to be in the **LION**'s enclosure.
7. The **LION** is king, so he wants to be in enclosure 1.

Using the reduced domains provided below, find one solution using **depth first search with forward checking and propagation through <u>domains reduced by any number of values</u> (propagation through reduced domains.)** Show your work by filling out the domain worksheet on this page and **drawing the search tree** on the next page. Break ties in numerical order (1,2,3,4).

Constraint graph for this problem



Domains for this problem

L	1
Hb	2 3 4
A	3 4
EL	2 3 4
H	2 3 4
M	1 2 3 4
B	1 2 3 4

<u>Reminder</u>: At some point, you might add several variables to your propagation queue at once. Break ties by adding variables to your propagation queue **<u>in alphabetical order</u>.**

	Var assigned or de-queued	List all values eliminated from neighboring variables	Back track ?
ex	X	Y ≠ 3, 4 Z ≠ 3 (example)	☑
1	L = 1	NONE	☐
2	Hb = 2	EL ≠ 2 H ≠ 2	☐
3	EL	A ≠ 3, 4	☑
4	Hb = 3	EL ≠ 3 H ≠ 3	☐
5	EL	A ≠ 3	☐
6	H	NONE	☐
7	A	EL ≠ 4 H ≠ 4	☐
8	EL	M ≠ 2 B ≠ 2	☐
9	H	NONE	☐
10	B	NONE	☐

	Var assigned or de-queued	List all values eliminated from neighboring variables	Back track ?
11	M	NONE	☐
12	A = 4	NONE	☐
13	EL = 2	NONE	☐
14	H = 2	NONE	☐
15	M = 1	B ≠ 3,4	☐
16	B	NONE	☐
17	B = 1	NONE	☐
18			☐
19			☐
20			☐
21			☐

LION	
HORNBILL	
ANTELOPE	
EVIL LION	
HYENA	
MEERKAT	
BOAR