**Massachvsetts Institvte of Technology**
**6.034 Artificial Intelligence, Fall 2011**
Recitation 4, Thursday, October 6

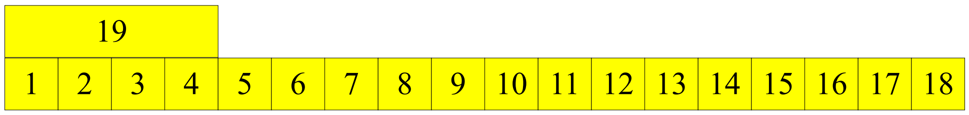
**Constraint Satisfaction** **Prof. Bob Berwick, 32D-728**

Constraint satisfaction algorithms that we have learned about:

1. **DFS with Backtracking** (**BT**). No constraint checks are propagated [“Assignments only”]. One variable is assigned a single value at a time, and then the next variable, and the next, etc. Check to see if the current partial solution violates any constraint. Whenever this check produces a zero-value domain, backup occurs.
2. **DFS with Backtracking + Forward-checking**. A variable is assigned a unique value, and then *only its neighbors* are checked to see if their values are consistent with this assignment. That is: assume the current partial assignment, apply constraints and reduce domain of neighboring variables. [“ Check neighbors of current variable only”].
3. **DFS with Backtracking + Forward-checking + propagation through singleton domains.** If during forward checking you reduce a domain to size 1 (singleton domain), then assume the assignment of the singleton domain and repeat forward checking from this variable. (Note that reduction to a singleton domain and repeated forward checking might lead to yet another singleton domain in some cases.)
4. **DFS with Backtracking + Forward checking + propagation through reduced domains (this is also known as: Arc-consistency, AC-2, since it ensures that all *pairs* of variables have consistent values)**. Constraints are propagated through all **reduced domains** of variable values, possibly beyond immediate neighbors of current variable, until closure. [“Propagate through reduced domains”; also called “Waltz labeling”]

**1.** Map coloring – comparing Backtracking (BT) with Forward-checking+propagation through singleton domains.

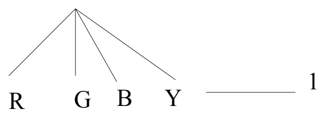
(a) How long will it take to color (two adjacent states cannot have the same color) the following map with 4 colors with just Backtracking (**BT**)?



Assume that:

- Color the states **in the order they are numbered**.
- Use R(ed), G(reen), B(lue) and Y(ellow) in **the given order & then in rotation**, starting with R; i.e., when you finish assigning Y, start again with R
- Assume that coloring a single location takes 1/100 second

Draw the shape of the search tree for **BT**: (we have started it for you):

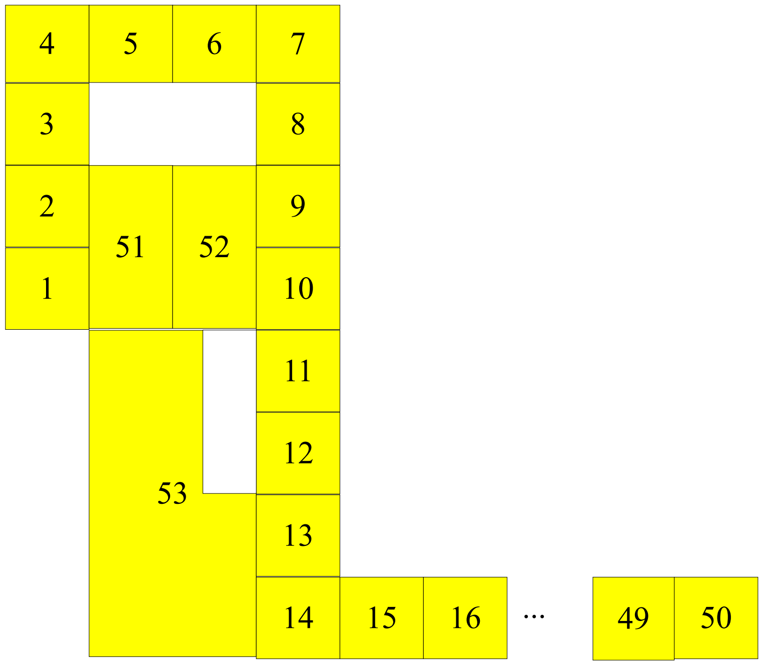


**BT** will therefore take _____ seconds

(b) How many seconds will **Forward checking+propagation through singleton domains** take on the *same* map?

**FC+prop through singletons** will take _______ seconds.

(c) Can **FC+propagation through singletons** finish the map below quickly? Why or why not? What about AC-2? (Propagation through reduced domains.)



Suppose we proceeded not from square #1 to #53 but from #53 to #1. Would our result now change? What principle of efficient constraint propagation does this illustrate? Are there any other principles of efficient constraint propagation that you can think of?

**2.** Converting problems into constraint propagation form

"Paul, John, and Ringo are musicians. One of them plays bass, another plays guitar, and the third plays drums. As it happens, one of them is afraid of things associated with the number 13, another is afraid of Apple Computers, and the third is afraid of heights. Paul and the guitar player skydive; John and the bass player enjoy Apple Computers; and the drummer lives in an open penthouse apartment 13 on the thirteenth floor. What instrument does Paul play?"

How can we solve this problem? Try it yourself first, by any means you care, then we'll see how to do it by – ta-da! – the magic of constraint propagation! (You might want to think about constraints when you solve it, and what the constraints are.)

What are the constraints? How might they be represented? We want to use the facts in the story to determine whether certain identity relations hold ore are eXcluded. Here is our notation: assume X(Peter, Guitar Player) means "the person who is John is not the person who plays the guitar." Further, this relation is symmetrical, so that if we have X(Peter, Guitar Player) then we also have, X(Guitar Player, Peter). In this notation, the facts become the following (of course all the symmetrical facts hold as well):

1. X(Paul, Guitar Player)
2. X(Paul, Fears Heights)
3. X(Guitar Player, Fears Heights)
4. X(John, Fears Apple Computers)
5. X(John, Bass Player)
6. X(Bass Player, Fears Apple Computers)
7. X(Drummer, Fears 13)
8. X(Drummer, Fears Heights)

Now we can represent the possible relations implicitly by means of entries in tables, and use constraint propagation. An *X* entry in a table denotes that the identity relation is excluded, and an *I* denotes that the identity relation actually holds. We can then use three tables, one to represent the possible identities between people and instrument players; one to represent the possible identities between people and fears; and a third to represent the possible relationships between instrument players and fears.

1. X(Paul, Guitar Player)
2. X(Paul, Fears Heights)
3. X(Guitar Player, Fears Heights)
4. X(John, Fears Apple Computers)
5. X(John, Bass Player)
6. X(Bass Player, Fears Apple Computers)
7. X(Drummer, Fears 13)
8. X(Drummer, Fears Heights)

| People | Instrument Player | | |
|---|---|---|---|
| | Bass Player | Guitar Player | Drum Player |
| Paul | | | |
| John | | | |
| Ringo | | | |

| People | Fears | | |
|---|---|---|---|
| | 13 | Apple Computers | Heights |
| Paul | | | |
| John | | | |
| Ringo | | | |

| Instrument Player | Fears | | |
|---|---|---|---|
| | 13 | Apple Computers | Heights |
| Bass player | | | |
| Guitar player | | | |
| Drum Player | | | |

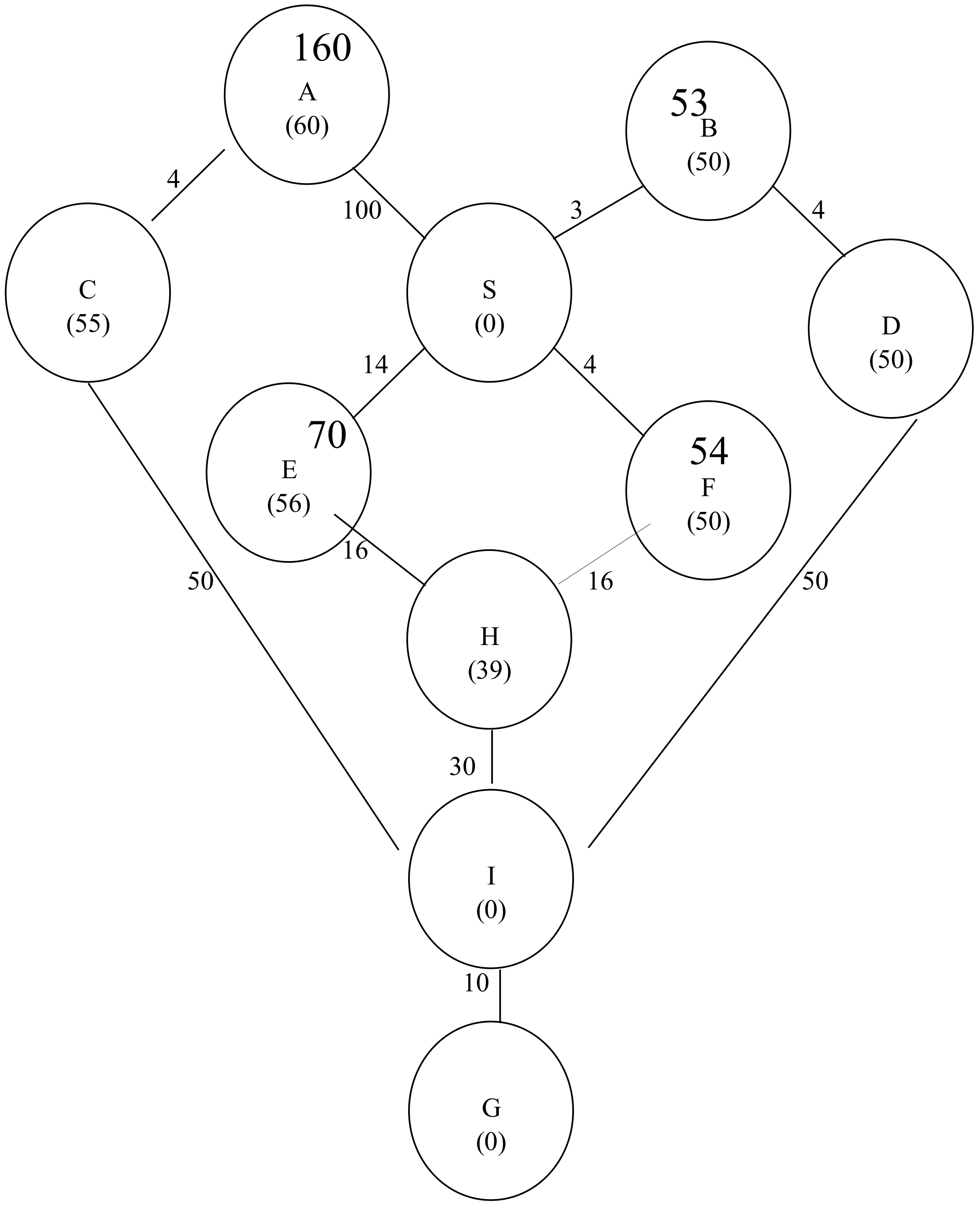
How do we do constraint propagation in this system? Note that we can deduce rules like the following to fill in the three tables:

1. If all but one entry in a row are *X,* then the remaining entry is *I.*
2. If you know *I*(*x,y*) and *X*(*y,z*) then you may conclude *X*(*x,z*).
Question: what property of the world does this last constraint rule capture?

What other rules do we need?

## Optimal Search

Now that Mark has his new stronghold, he wants to invade parallel universes. So Mark programs his evil supercomputer to find the shortest path of jumps from his starting universe S to his goal universe G.



**Part B1 Branch & Bound search**

First, Mark programs a simple **branch-and-bound search with an extended list**. As usual, he breaks ties of equal length in lexicographic order. List the nodes Mark's computer adds to the extended list, in order. Distances are shown next to edges. Ignore the numbers in parentheses for this part of the problem. Extra space is provided below in case you want to show your work.

Remember: for B&B, f(n)=g(n) (total path length so far), and we sort the agenda by total path length.

(3 S B)(4 S F)(14 S E)(100 S A)

(4 S F)(7 S B D)(14 S E)(100 S A)

etc.

The answer is:

S B F D E H I G

The path found is:

S F H I G  with cost 60

What path does Mark's computer find?

**Part B2**

Frustrated by branch-and-bound's speed, Mark reprograms his computer to use **A***. Mark counts the number of subspace anomalies between each universe and the goal and uses this count as the **heuristic** for A* (**these are the numbers in parentheses**). List the nodes Mark's computer adds to the extended list, in order. Extra space is provided below in case you want to show your work. REMEMBER: For A*, f(n)=g(n)+h(n)= total path length + heuristic value at that node. We have done the calculation of f for the first nodes away from S on the first page.

Remember: We sort the agenda by total path length.
(53 S B)(54 S F)(70 S E)(160 S A)
(54 S F)(57 S B D)(70 S E)(160 S A)
(57 S B D)(59 S F H)(70 S E)(160 S A)
(57 S B D I)(59 S F H)(70 S E)(160 S A)
(59 S F H)(67 S B D I G)(70 S E)(160 S A)
~~(50 S F H I)~~(67 S B D I G)(70 S E)(160 S A) [Why???]
(67 S B D I G)(70 S E)(160 S A)
Path: S B D I G, length= 67, which is non-optimal.
Compute the f values in F, H, I and let's see why this happens... remember, f values should be monotonically increasing on each path, e.g., S-F-H-I, are they?
What is the term for this?

What path does Mark's computer find now?

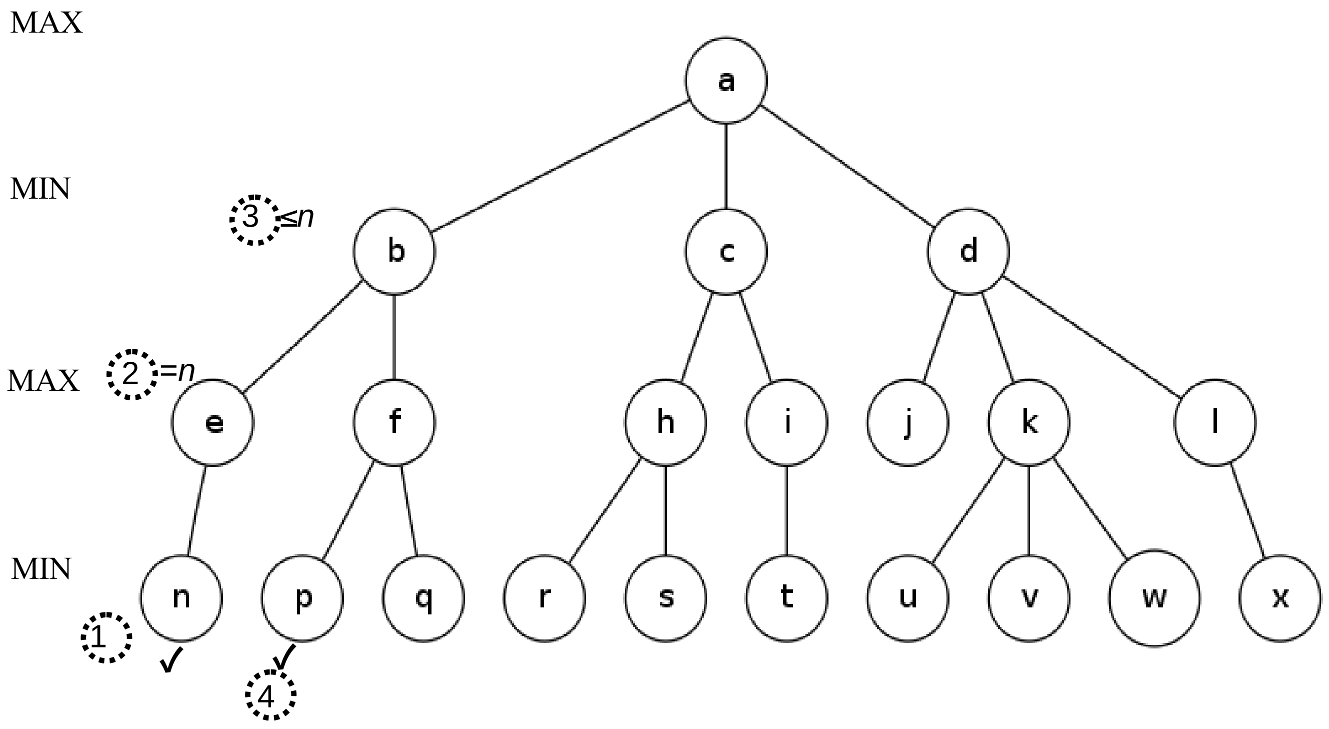
**Part B3**

Mark is confused. Give a **brief** but **specific** explanation of what happened and why.

# Problem 1:Games

## Part A:  Working with a maximally pruned tree (25 points)

For the following min-max tree, cross out those leaf nodes for which alpha-beta search would **not do static evaluations** in the best case possible (minimum number of static evaluations, maximum pruning of nodes to be statically evaluated).



### Part A1

Now, list the leaf nodes at which alpha-beta would do static evaluations in the best case possible.

## Part A2

What is the final value returned by the alpha beta search in the best case possible for the given tree? Express your answer as the simplest function of the static values of the leaf nodes (e.g. take **n** to be the static value at the leaf node labeled n). Your function may contain operations such as **max** and **min**.

## Part A3

What constraints ensure best case possible (minimum static evaluation) for the given tree? State your constraints as inequalities on the static values of the leaf nodes.

## Part A4

Suppose your static evaluation function, S(node), is modified as follows:

*S'(*node) = 42 x S*(*node) + 1000. (If S(node) = 1, S'(node) = 1042)

Would your answer for Part A1 be the same for all possible S(node) values? **Yes** **No**

Suppose your function were

*S'(*node) = - 42 x S*(*node) + 1000.

Would your answer for Part A1 be the same for all possible S(node) values? **Yes** **No**

# Problem 2: Time Travelers' Convention

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, 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.

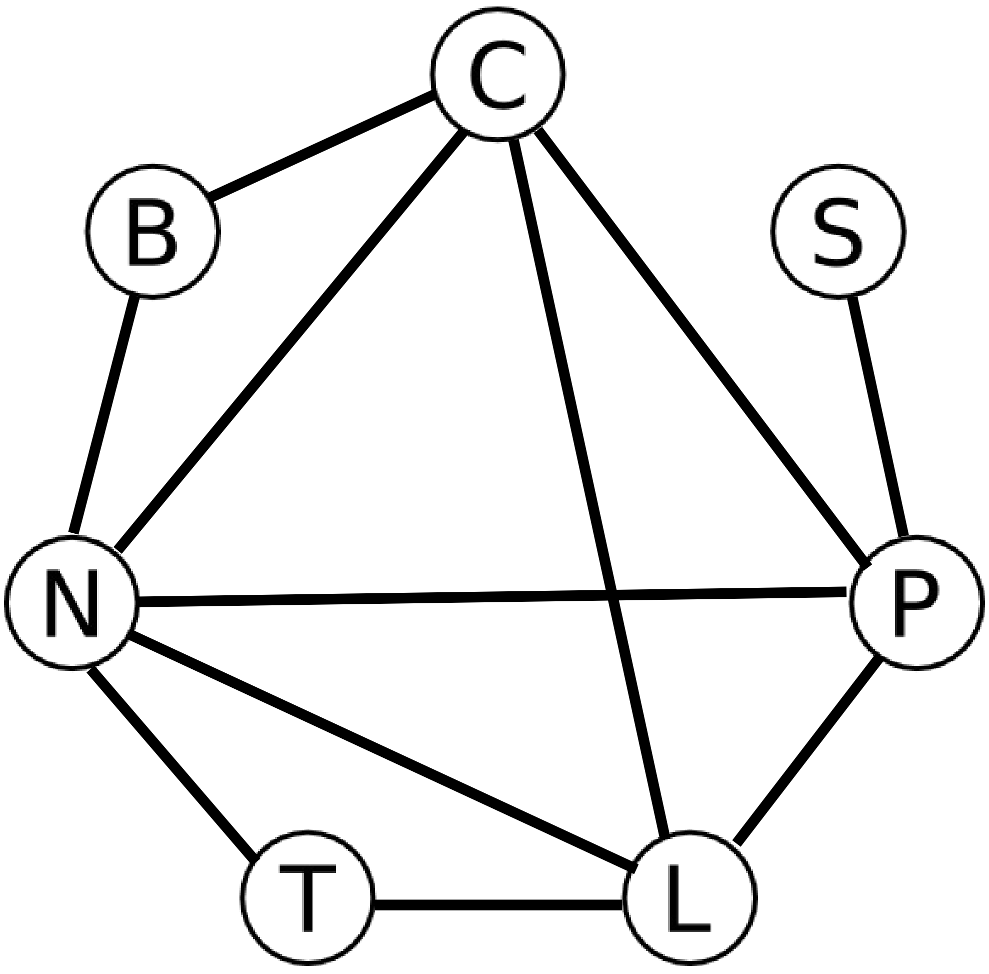
After talking to some students who are planning to attend this year's convention, you determine that they fall into certain groups, each of which wants to be able to see some subset of the time-traveling speakers. (Fortunately, each student identifies with at most one of the groups.) You write down everything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

The list of guest lecturers consists of Alan **T**uring, Ada **L**ovelace, Niels **B**ohr, Marie **C**urie, **S**ocrates, **P**ythagoras, 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 (5 points)

That's a lot of preferences to keep track of, so you decide to draw a diagram to help make sense of it all. Draw a line between the initials of each pair of guests who must not



## Part B (15 points)

You decide to first assign the time slots (which conveniently happen to be 1, 2, 3, and 4 pm) by using a **depth-first search with no constraint propagation**. The only check is to be sure each new assignment violates no constraint with any previous assignment. As a tiebreaker, assign a lecturer to the earliest available time slot (so as to get them back to their own historical eras as soon as possible).

In the tree below, Alan Turing has already been scheduled to speak at 1 pm, in accordance with constraint #1. Continue filling in the search tree up to the first time you try (and fail) to assign a time slot to Isaac Newton, at which point you give up in frustration and move on to Part C in search of a more sophisticated approach.

**T** 1

**L** 1 2 3 4

**B**

**C**

**S**

**P**

**N**

## Part C (20 points)

You're not fond of backtracking, so rather than wait and see just how much backtracking you'll have to do, you decide to start over and use **depth-first search with forward checking (constraint propagation through domains reduced to size 1)**. As before, your tiebreaker is to assign the earliest available time slot.

**T** 1

**L**

**B**

**C**

**S**

**P**

**N**

What is the final lecture schedule you hand in to MITTTS?

1 pm:

2 pm:

3 pm:

4 pm:

## Part D (10 points)

Now, rather than backtracking, you're concerned about the amount of time it takes to keep track of all those domains and propagate constraints through them. You decide that the problem lies in the ordering of the guest list. Just then, you get a call from the MITTTS president, who informs you that **Alan Turing's schedule has opened up and he is now free to speak during any one of the four time slots.**

Armed with this new information, you reorder the guest list to maximize your chances of quickly finding a solution. In particular, which lecturer do you now assign a time slot to first, and why?