# 6.034 QUIZ 1

**Fall 2002**

| | |
|---|---|
| **Name** | |
| **E-mail** | |

| **Problem Number** | **Maximum** | **Score** |
|---|---|---|
| **Problem 1** | **30** | |
| **Problem 2** | **30** | |
| **Problem 3** | **10** | |
| **Problem 4** | **30** | |
| **Total** | **100** | |

## Problem 1: Rule-Based Book Recommendations (30 points)

### Part A: Forward chaining (15 points)

You want to recommend books to two of your friends, so you decide to use your forward-chaining book recommender.

Your rule based system is set up as follows (it is duplicated on a tear off sheet at the end of the exam).

Database of assertions:

(Max lives-in WashingtonDC)
(Jane lives-in SanFrancisco)
(Max likes science-fiction)
(Jane likes PhilipKDick)
(Pat likes TheThreeStigmataOfPalmerEldritch)
(PhilipKDick is-author-of Ubik)
(PhilipKDick is-author-of TheManInTheHighCastle)
(PhilipKDick is-author-of ThePenultimateTruth)

Rules:

| | | |
|---|---|---|
| R1 | If | (?x likes PhilipKDick) |
| | Then | (?x likes science-fiction) |
| R2 | If | (?x likes Ubik) |
| | Then | (?x likes alternate-realities) |
| R3 | If | (?x lives-in SanFrancisco) |
| | | (?x likes science-fiction) |
| | Then | (?x likes alternate-realities) |
| R4 | If | (?x lives-in WashingtonDC) |
| | Then | (?x likes politics) |
| R5 | If | (?x likes politics) |
| | | (?x likes science-fiction) |
| | Then | (ThePenultimateTruth is-recommended-for ?x) |
| R6 | If | (?x likes alternate-realities) |
| | Then | (TheManInTheHighCastle is-recommended-for ?x) |

Fill out the following table to show the details of running the forward chainer. Use rule ordering for the conflict resolution strategy. Assume new assertions are added to the end of the assertion database. Terminate when no further assertions can be made. You may abbreviate clauses as long as there is no ambiguity. (Note: there may be more steps in the table than you need.)

| Step | Triggered Rule(s) | Rule Instance Binding(s) | Rule Fired | Assertion(s) Added |
|---|---|---|---|---|
| 1 | | | | |
| | | | | |
| | | | | |
| | | | | |
| 2 | | | | |
| | | | | |
| | | | | |
| | | | | |
| 3 | | | | |
| | | | | |
| | | | | |
| | | | | |
| 4 | | | | |
| | | | | |
| | | | | |
| | | | | |
| 5 | | | | |
| | | | | |
| | | | | |
| | | | | |
| 6 | | | | |
| | | | | |
| | | | | |
| | | | | |
| 7 | | | | |
| | | | | |
| | | | | |
| | | | | |

## Part B: Backward chaining (15 points)

One of your friends suggests that Pat might like TheManInTheHighCastle, but you want your backward chainer to help you prove whether or not that statement is true. You use the same initial assertions as in your forward chaining system, plus a new assertion:

(Pat lives-in SanFrancisco)

You also use the same six rules as in your forward chaining system, plus a new rule:

|  |  |  |
|---|---|---|
| R7 | If | (?x likes TheThreeStigmataOfPalmerEldritch) |
|  | Then | (?x likes PhilipKDick) |

You then ask your backward chainer to prove the following assertion:

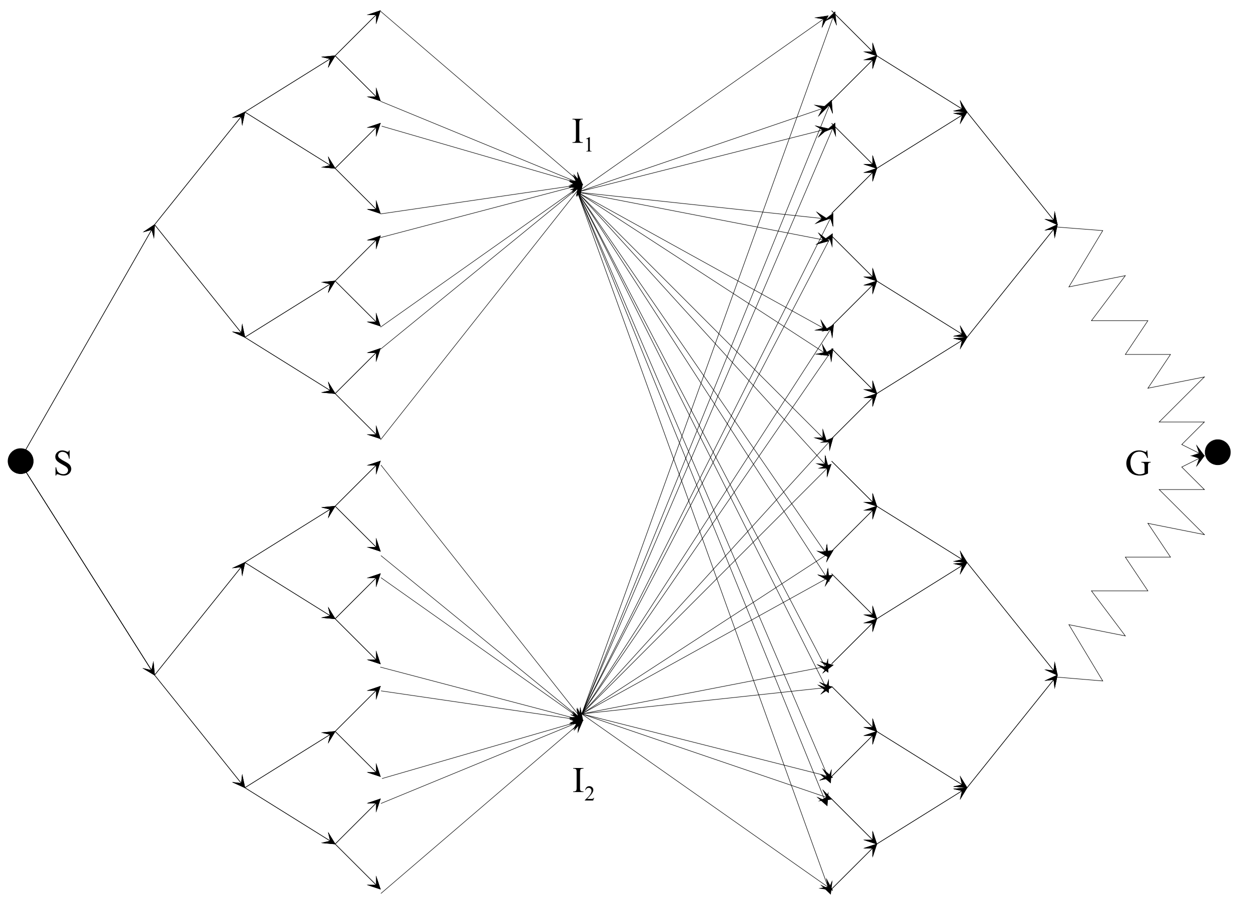
(TheManInTheHighCastle is-recommended-for Pat)

Using this assertion as the root node, **on the next page**, draw the goal (and/or) tree that your system uses to prove the assertion. (The root node is provided on the next page.) Assume that your system uses rule ordering as a conflict resolution strategy. Also assume that if an assertion cannot be proven via rules or existing assertions, then it fails. (Your system does not query you for an answer.) Label each branch of the tree with the name of the rule (e.g. R1) that it represents.

(TheManInTheHighCastle is-recommended-for Pat)

## Problem 2: Search in a Weird City (30 points)

You have decided to look for a job driving a cab. The recruiter, an MIT graduate, asks you to solve some search problems for streets laid out as shown by the lines in the figure. **The map is repeated on a tear-off sheet at the end of the exam!**



***The map is drawn to scale***, but **note that the final, jagged streets converging on G can be assumed to be 100 times longer than any other street**. All streets are one-way, left to right. In the absence of any other criterion, or a tie, you are to follow the option closer to the top of the page.

Your job is to find a path from the Start node, S, to the Goal node, G, according to various conditions.

You may express your answers either as numbers (e.g. 8) or variable-free expressions (e.g. $2^0 + 2^1 + 2^2 + 2^3$) or both. You get full credit if either is right. Also, you will increase your partial credit if you correctly note when two answers should be the same, even if both are wrong.

## Part A: Depth First Search, with backup (4 points)

You are ***not*** to use a enqueued list (also known as visited list) or an extended list (also known as an expanded list). Search is to terminate when a path that reaches the goal appears at the front of the search queue. Indicate the number of nodes extended.

## Part B: Breadth First Search (4 points)

You ***are*** to use an enqueued list. Search is to terminate when a path that reaches the goal appears at the front of the search queue. Indicate the number of nodes extended.

queue

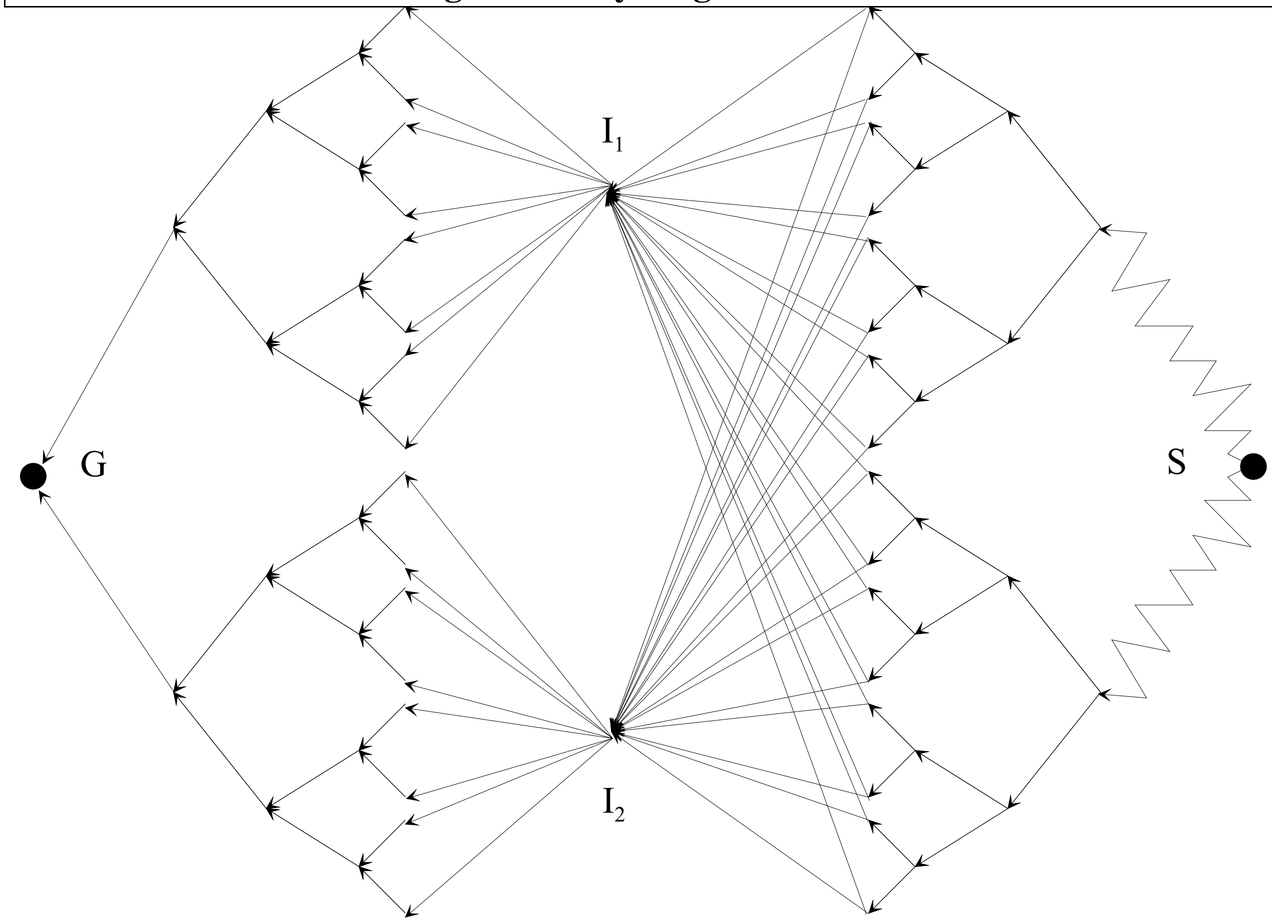
## Part C: Branch and Bound (4 points)

You are to find the optimal path, using ***neither*** an extended list, nor any heuristic. Indicate the number of partial paths that will be ***on the search queue*** when the goal appears at the front of the search queue.

## Part D: Branch and Bound (4 points)

You are to find the optimal path, ***using*** an extended list, and the distance to the goal as an admissible heuristic. Indicate the number of nodes extended.

**For the rest of this problem, consider the same graph, but with S and G reversed, along with the direction of the one-way streets. All streets are one-way, right to left. In the absence of any other criterion, or a tie, you are to follow the option closer to the top of the page. Remember that the map is drawn to scale, except that the connections leaving S are very long.**



## Part E: Depth First Search, with backup (4 points)

You are ***not*** to use an enqueued list or an extended list. Search is to terminate when a path that reaches the goal appears at the front of the search queue Indicate the number of nodes extended.

## Part F: Breadth First Search (4 points)

You ***are*** to use an enqueued list. Search is to terminate when a path that reaches the goal appears at the front of the search queue. Indicate the number of nodes extended.

queue

## Part G: Branch and Bound (3 points)

Suppose you were to look for the optimal path, using ***neither*** an extended list, nor any heuristic. Indicate (yes or no) whether all possible paths will be from S to G will be computed. ***Remember that the map is drawn to scale*** (except that the connections leaving S are very long).

☐ Yes ☐ No

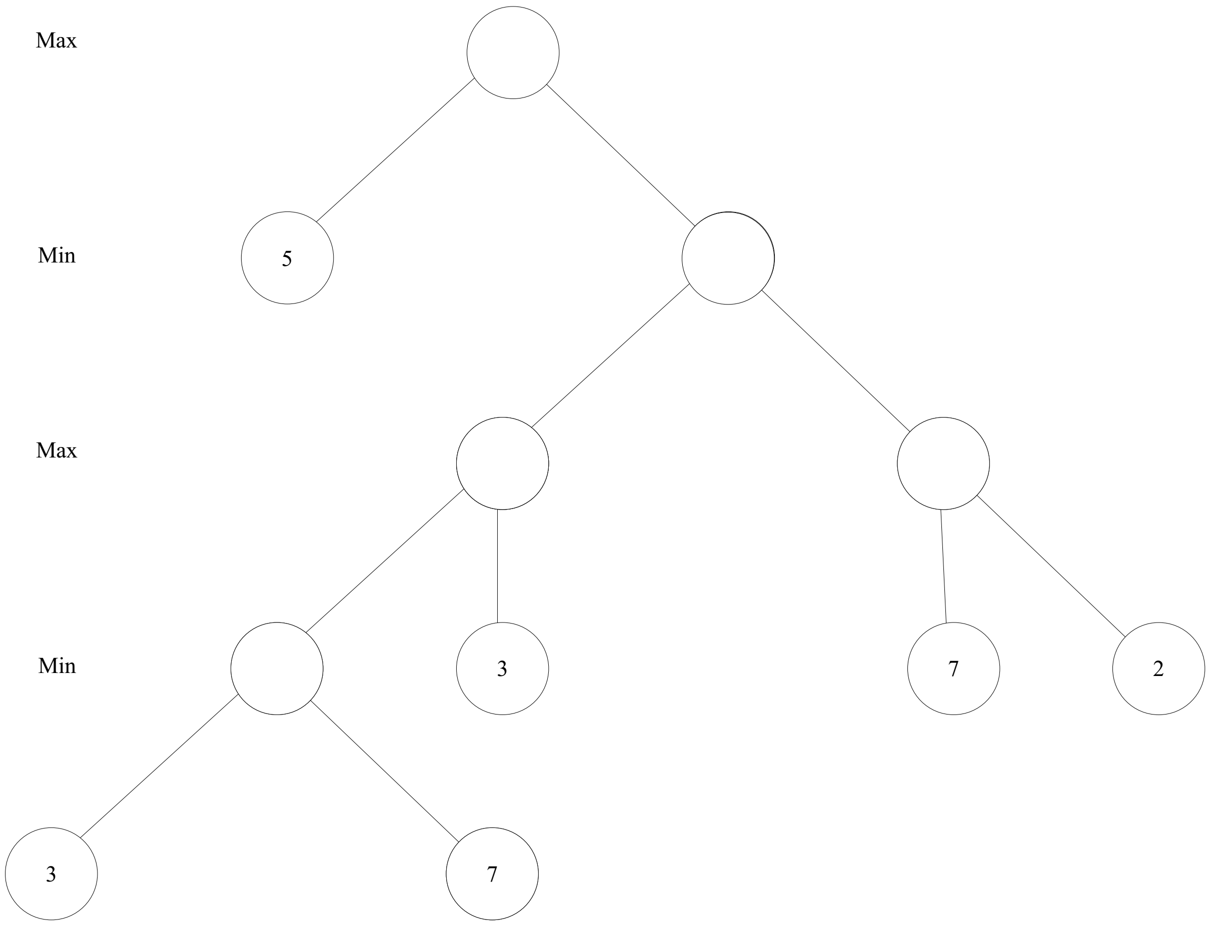
## Part H: A* (3 points)

Suppose you were to look for the optimal path, ***using*** an extended list (with no heuristic measure of distance remaining). Indicate the number of paths from S to $I_1$ that are extended beyond $I_1$.

## Problem 3: Games (10 points)

### Part A Mini-Max (5 points)

Show the path the maximizer expects to be followed in the following game tree. An oracle has provided static values. **Do not use alpha-beta.**



### Part B Alpha-Beta (5 points)

Now return to the diagram and cross out the static evaluations you would not perform if you combined minimax with alpha-beta.

## Problem 4: CSP/Resource Allocation (30 points)

### Part A (15 points)

You have a nightmare. You have become the MIT classroom scheduler, and you must not only assign TAs to classrooms, but also, you must do it using 6.034 ideas.

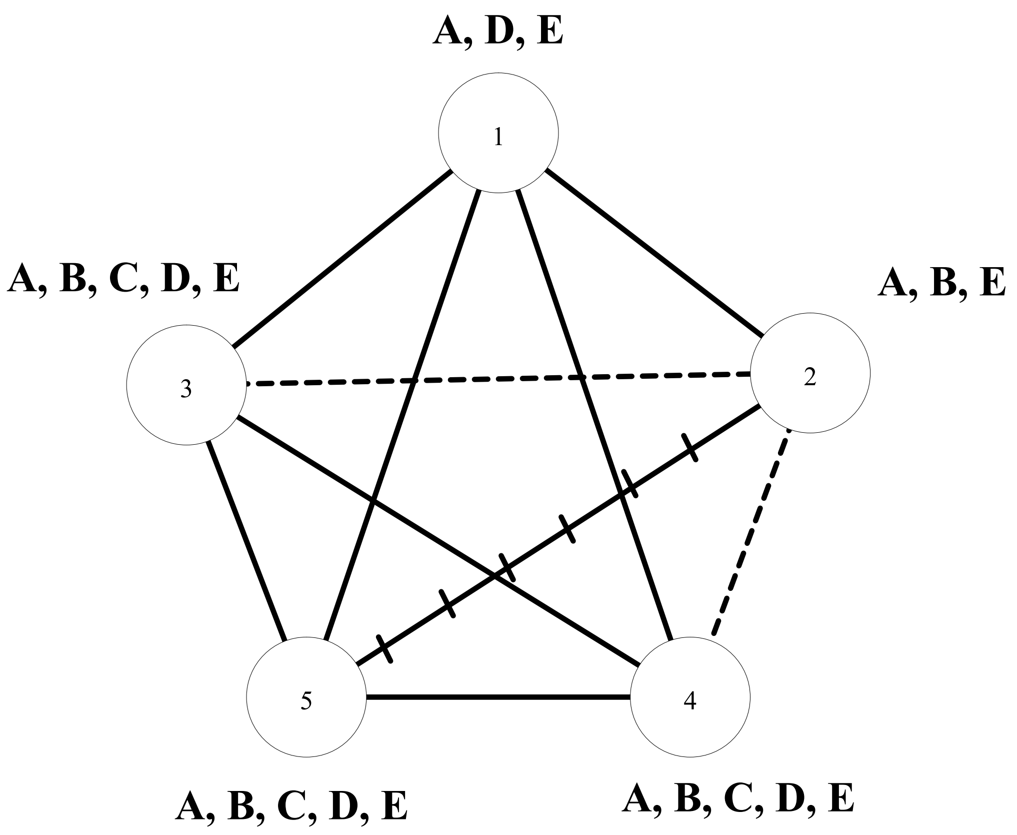
In particular, there are five TAs: A, B, C, D, and E, all to be scheduled at noon.

- Some will not accept assignment to particular rooms.
- Some are allergic to each other and cannot be assigned to classrooms that are close together.
- Others like each other and cannot be assigned to classrooms that are far apart.

And there are five rooms: 1, 2, 3, 4 and 5.

- 2-3 and 2-4 are close together
- 2-5 are far apart
- Others pairs are in between, neither close together nor far apart.

When all the information is assembled, you have the following diagram:



- Each TA's name is listed next to the rooms to which s/he is willing to be assigned.

- The solid lines, —, identify other room pairs that are neither close together nor far apart. Such a pair of rooms cannot have the same person in both rooms. Thus, no solid line can connect the following pairs:

  A-A, B-B, C-C, D-D, E-E

- The dashed lines, - - -, identify room pairs that are close together. Such a pair of rooms cannot have the same person in both rooms, nor can such a pair of rooms have two people who are allergic to each other. That is, no dashed line can connect the following pairs:

  A-A, B-B, C-C, D-D, E-E,                A-B, B-D, B-E, C-D

- The hatched line, -|-|-|-, identifies a room pair that is far apart. This pair of rooms cannot have the same person in both rooms, nor can this pair of rooms have two people who like each other. That is, no hatched line can connect the following pairs:

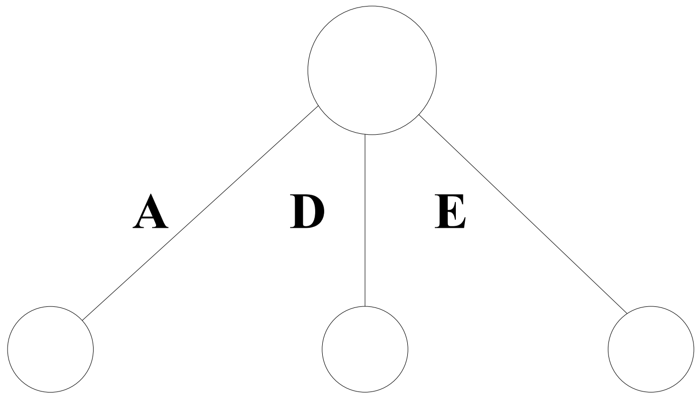
  A-A, B-B, C-C, D-D, E-E,                A-E, A-D, B-D

Your job is to find a consistent set of assignments using **the version of constraint checking** outlined below:

- You are to use depth-first search with backup.
- You are to use constraint propagation (also known as arc-consistency) to eliminate values inconsistent with values assigned by the depth-first search at neighboring nodes.
- Your constraint propagation is to ***continue through domains reduced to unique assignments.***
- You are to terminate search and backup whenever a domain becomes empty.
- Assignments are made to rooms in numerical order; assignments are to be tried in alphabetical order.

Show your results by completing the tree **on the next page**. Be sure to include in your tree any assignment that you make, including those that lead to one or more empty domains.

Possible room 1 Assignments



Possible room 2 Assignments

Possible room 3 Assignments

Possible room 4 Assignments

Possible room 5 Assignments

## Part B (15 points)

It occurs to you that instead of assigning TAs to rooms, you could assign rooms to TAs (the so-called dual problem).

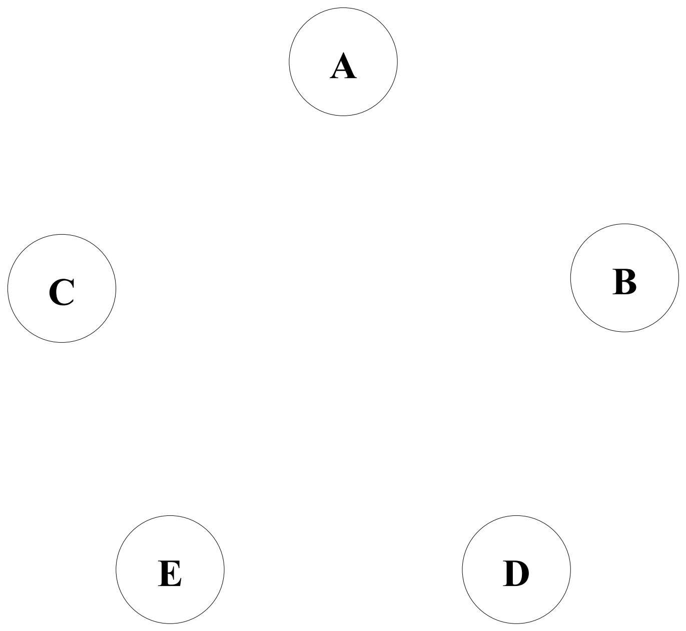
- Indicate which rooms are allowed for each TA.

| | |
|---|---|
| A | |
| B | |
| C | |
| D | |
| E | |

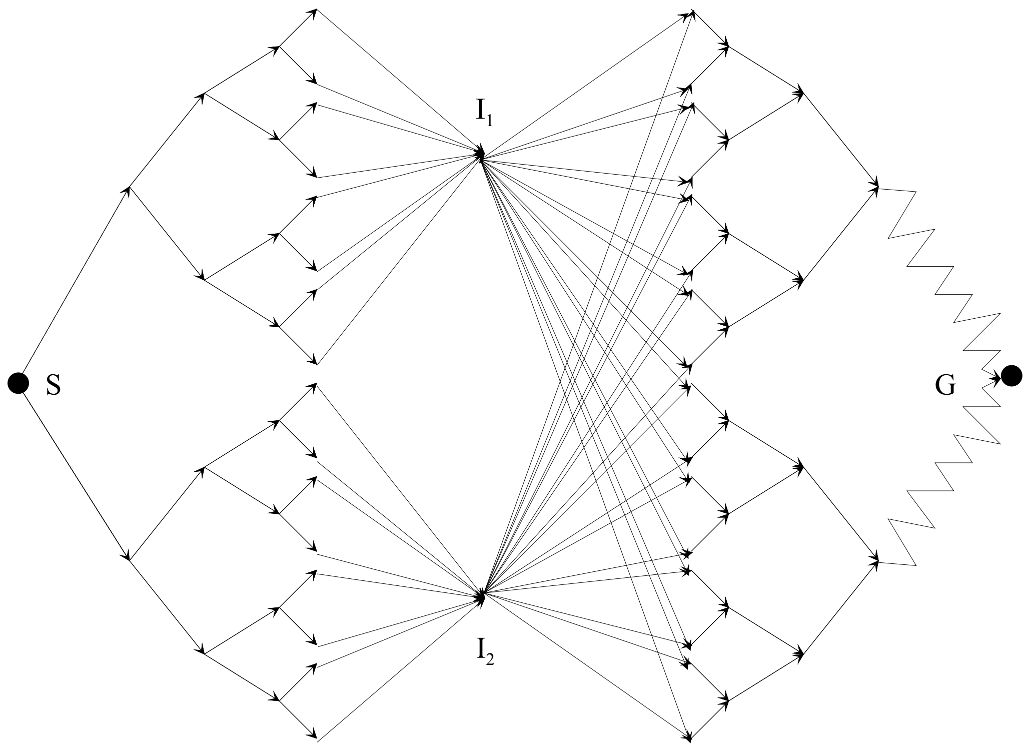
- Now, decorate the diagram below using dashed lines, - - -, to indicate no one classroom can be assigned to two different TAs and that classrooms that are close together cannot be assigned to people who are allergic to each other.

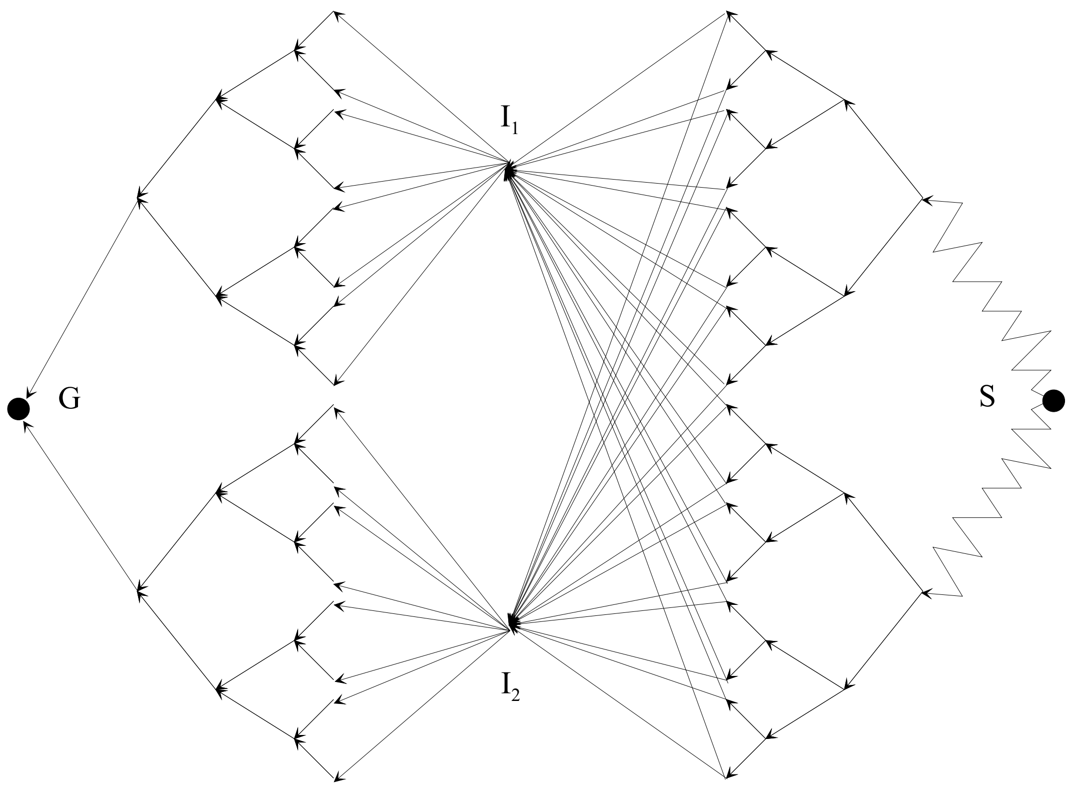
- Next use hatched lines, -|-|-|-, to indicate no one classroom can be assigned to two different TAs and that classrooms that are far apart cannot be assigned to people who like each other.

- Finally, use solid lines, —, to indicate only that no one classroom can be assigned to two different TAs.

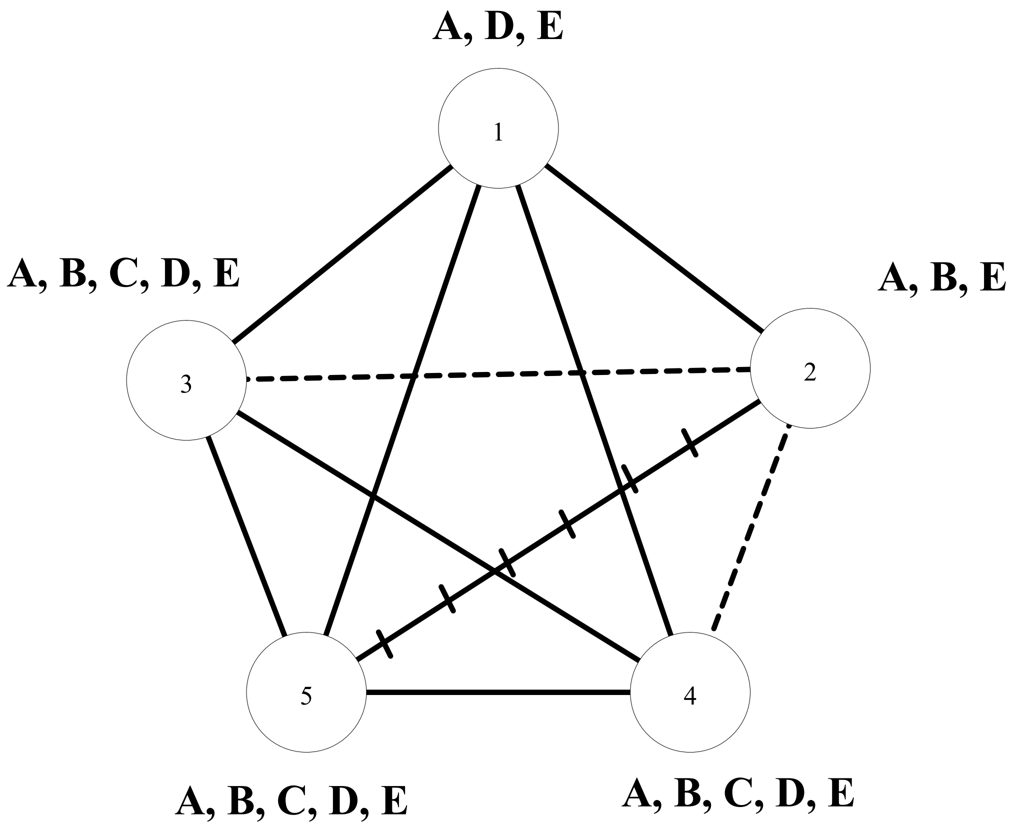


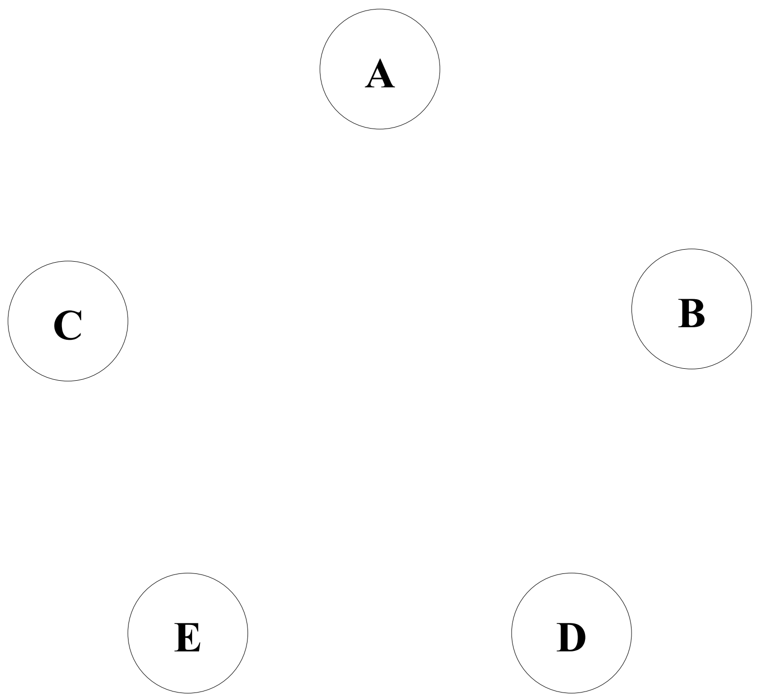
**This is a scratch work page.  Do not hand it in.**





**This is a scratch work page. Do not hand it in.**





**This is a scratch work page. Do not hand it in.**

**Database of assertions:**

(Max lives-in WashingtonDC)
(Jane lives-in SanFrancisco)
(Max likes science-fiction)
(Jane likes PhilipKDick)
(Pat likes TheThreeStigmataOfPalmerEldritch)
(PhilipKDick is-author-of Ubik)
(PhilipKDick is-author-of TheManInTheHighCastle)
(PhilipKDick is-author-of ThePenultimateTruth)

**Rules:**

| | | |
|---|---|---|
| R1 | If | (?x likes PhilipKDick) |
| | Then | (?x likes science-fiction) |
| R2 | If | (?x likes Ubik) |
| | Then | (?x likes alternate-realities) |
| R3 | If | (?x lives-in SanFrancisco) |
| | | (?x likes science-fiction) |
| | Then | (?x likes alternate-realities) |
| R4 | If | (?x lives-in WashingtonDC) |
| | Then | (?x likes politics) |
| R5 | If | (?x likes politics) |
| | | (?x likes science-fiction) |
| | Then | (ThePenultimateTruth is-recommended-for ?x) |
| R6 | If | (?x likes alternate-realities) |
| | Then | (TheManInTheHighCastle is-recommended-for ?x) |

**Added in Part B**

(Pat lives-in SanFrancisco)

| | | |
|---|---|---|
| R7 | If | (?x likes TheThreeStigmataOfPalmerEldritch) |
| | Then | (?x likes PhilipKDick) |