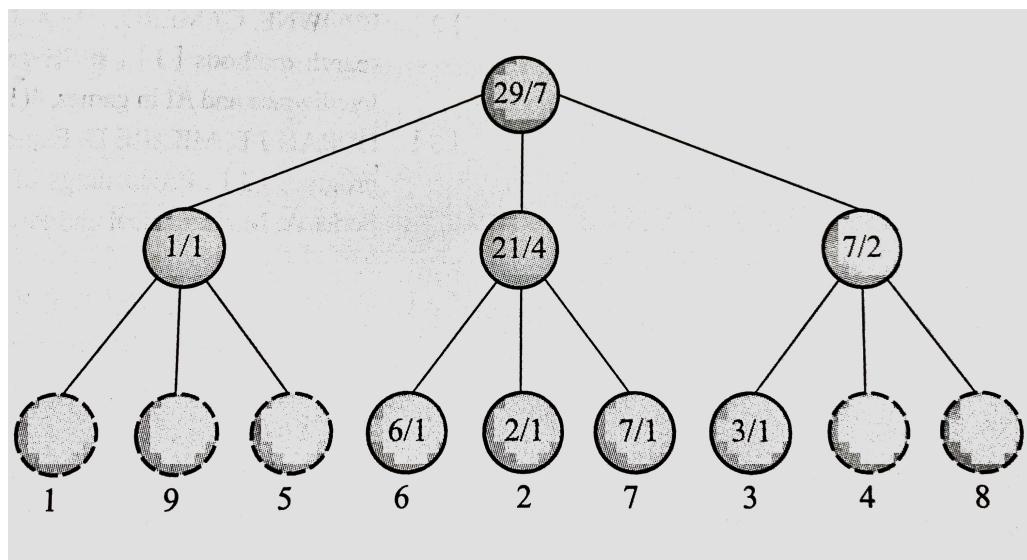


Question 1 (20 points)

The following figure shows an example of Monte Carlo Tree Search, where each leaf node is annotated with the utility value. To maximize the utility, here Monte Carlo Tree Search is applied to return a planning path. Suppose that after several steps, the current state of the MCTS search tree is also shown in the following figure, where the numbers within a node represent “estimated value / the number of visits” and those nodes encircled by dashed lines are not expanded yet.

- (5 points) Provided that the hyperparameter $C=1$ in the UCB bound, please show the path selected by MCTS.
- (5 points) Please continue the procedures of selection, expansion, roll out, and backpropagation, and visualize the state of the search tree afterwards.
- (10 points) Try more iterations of the four procedures of MCTS, and see if the leaf node with the largest utility (i.e., 9) can be returned. If not, please give your suggestions on how to modify the algorithm for higher efficiency.



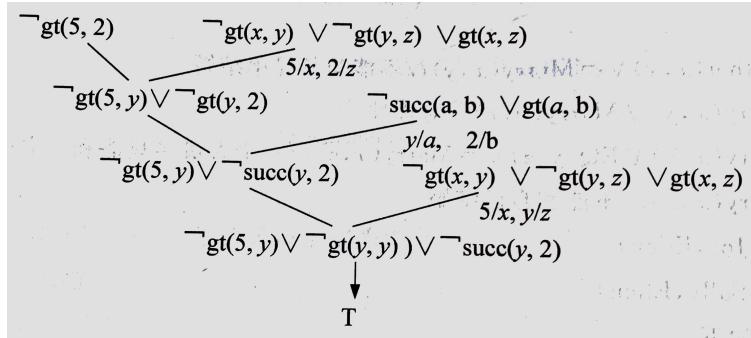
Question 2 (20 points)

Suppose that you are given the following facts.

- $\forall x \forall y \forall z \text{gt}(x, y) \wedge \text{gt}(y, z) \rightarrow \text{gt}(x, z)$
- $\forall a \forall b \text{succ}(a, b) \rightarrow \text{gt}(a, b)$
- $\forall x \text{gt}(x, x)$

To prove $\text{gt}(5, 2)$ with resolution, there is an attempt shown as below.

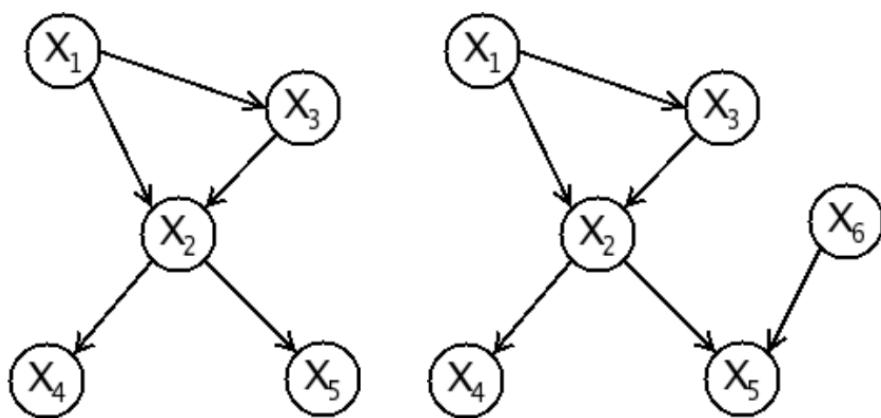
- $\forall x \text{gt}(x, x)$



- What kind of errors exist(s) in the above proof?
- To avoid such errors, do you need additional procedures? What kind of procedures? Can you rewrite the correct proof?

Question 3 (20 points):

Consider the Bayesian network given in the figure below. The prerequisite here is that all of the variables are Boolean. Show the variable elimination process for the following ones.



- $P(X_3|X_4)$ for the left figure, where the order of elimination follows X_5, X_1, X_2 ;
- $P(X_3|X_4)$ for the left figure, where the order of elimination follows X_2, X_1, X_5 ;
- $P(X_3|X_4)$ for the right figure, where the order of elimination follows X_5, X_1, X_2 ;
- $P(X_3|X_4)$ for the right figure, where the order of elimination follows X_2, X_1, X_5 .

Programming (40 points)

In this project, your drone agent will find paths through an urban environment, both to reach a particular location and to cover a set of specified locations along the path. You will build general search algorithms and apply them to this path planning problem.

Instructions before coding:

1. Clone the project repository from <https://github.com/udacity/FCND-Motion-Planning>
2. Download the simulator from <https://github.com/udacity/FCND-Simulator-Releases/releases> (selecting the simulator according to your operating system)
3. Set up your Python Environment using Anaconda. If you are not familiar with Anaconda, please kindly refer to this site <https://github.com/udacity/FCND-Term1-Starter-Kit>.
 - 3.1 If you install the package **udacidrone** by Anaconda, you can ignore the next step in 3.2.
 - 3.2 If you have problems in installing the package **udacidrone** by Anaconda, you can 1) download it from <https://github.com/udacity/udacidrone>, and 2) put it into the file **FCND-Motion-Planning3** making sure that this package can be called by **motion_planning.py**. 3) When you run the project **motion_planning**, you may be reminded some packages are missing. The solution is to install these required packages following this instruction: **pip --upgrade package's name** (you can refer to <https://linuxize.com/post/how-to-install-pip-on-ubuntu-18.04/> to learn how to use **pip**, if you do not know about this.)
4. After finishing the setup of your Python Environment, you can test if this project runs well in your computer. First, open the simulator and choose MOTION PLANNING shown in Fig. 1. Second, run the project **motion_planning**, and you will see this drone flies automatically as shown in Fig. 2.

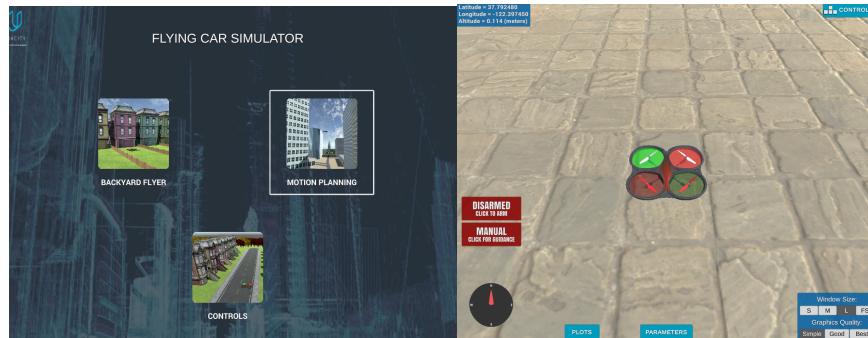


Figure 1 Click MOTION PLANNING and you will see a drone.

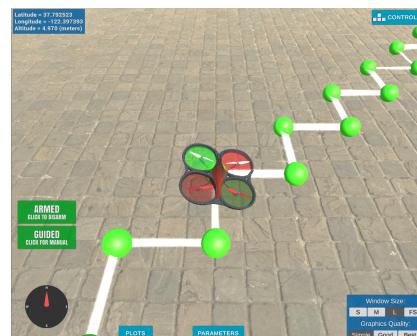


Figure 2 The drone flies automatically.

5. Write your planner with A* search algorithm based on **`motion_planning.py`**. If it is difficult for you, you can refer to the file **Untitled.ipynb** which you can open via github or via the link <https://github.com/udacity/FCND-Motion-Planning/blob/master/Untitled.ipynb>.
6. After you finish your planner, you can test it. Through adjusting the values of parameters `grid_start` and `grid_goal`, you can set up the start point and end point for your drone. **Reminder:** **if these points is in the obstacle map recorded by the file `colliders.csv`, the search algorithm may take a very long time and fail to output the path. In this situation, you should better reset these parameters and run the updated program again.**

Files you'll edit:

1. **`motion_planning.py`**: this function leverages some extra functions in `planning_utils.py` to plan the path for the drone.
2. **`planning_utils.py`**: where all of your search algorithms will reside.

Files to edit and submit:

You are expected to submit the two files above with your code and comments. Please do not change the other files in this distribution or submit any of our original files other than these files.

Question 4 (20 points): Implementing the iterative deepening A* search algorithm

You are expected to write an iterative deepening A* search algorithm in `planning_utils.py` named `iterative_astar(grid, h, start, goal)` to help the drone plan routes. Procedure for the iterative deepening A* search algorithm can be found in the lecture slides.

Question 5 (20 points): Implementing different heuristics for A* and the A* search for traversing 3 fixed points

In the current A* version in `planning_utils.py`, the Manhattan distance is used as the heuristic. You are encouraged to propose one valid heuristic, implement that, and see how the planned routes change.

You are also required to set three fixed points in `motion_planning.py` which the drone has to traverse before reaching to the destination. Building on this, you re-implement the A* search algorithm in order to go through the 3 points.