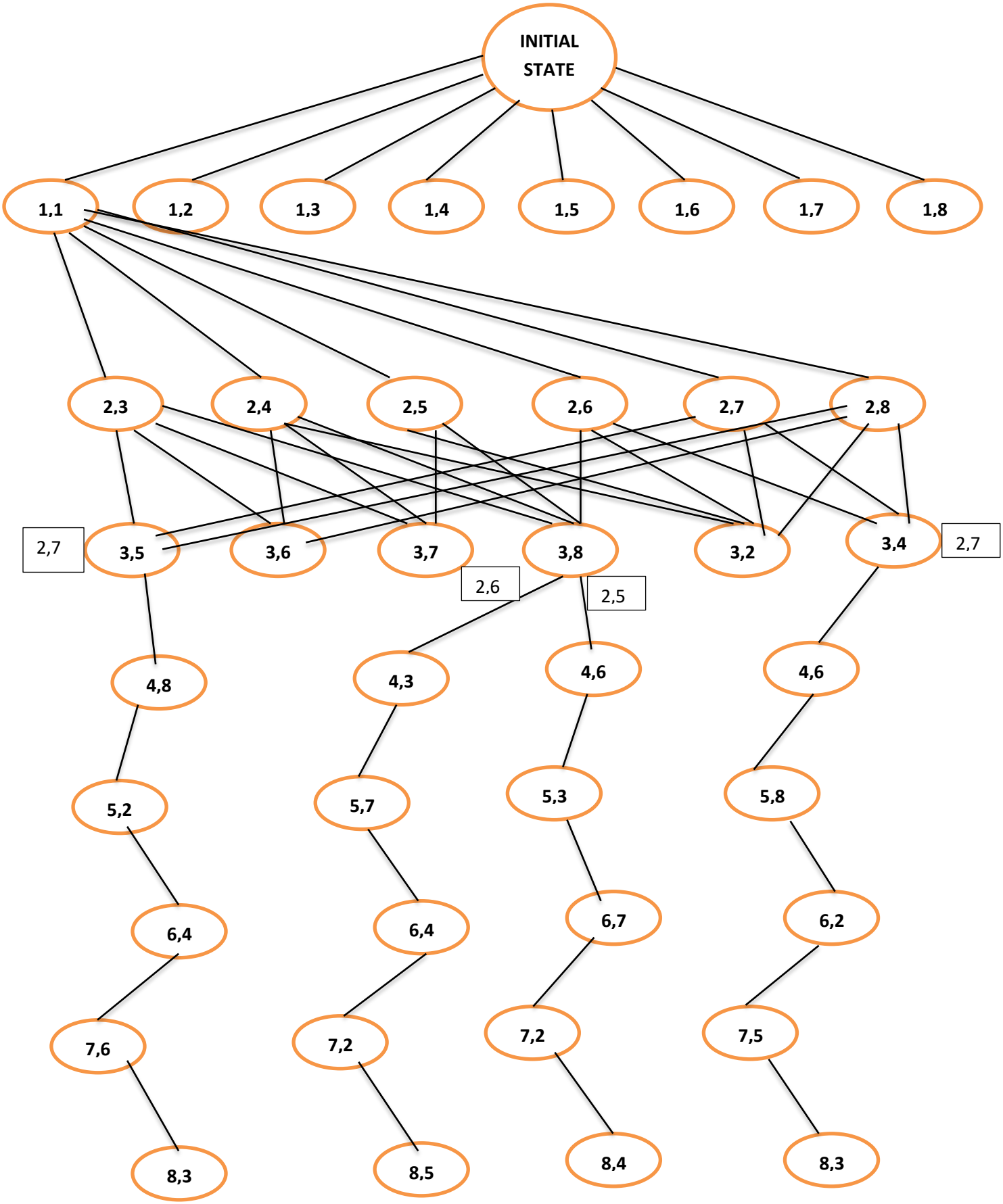
Create the Search tree for 8 queen's Problem.



Define a class cricket match. It is not one class that will be created but a whole hierarchy of classes. The classes should define an international cricket match. Pl. remember to use the concept of inheritance to define the types of international cricket matches. Your class should handle all the types of exceptions that may arise due to changes in the rules of the games. the class hierarchy thus created should be capable of instantiating any of the international cricket that has been played till date. Keep provision to handle any futuristic exception situation.

CODE:

```
class cricket:
   print("Cricket Match!")

   def match_info(self,city,stadium):
       print("Teams=2 and players in each team=11")
       self.city=city
       self.stadium=stadium
       print("The location of match:", self.stadium,",",self.city)

   def name_teams(self,x,y):
      self.x=x
      self.y=y
      print(self.x,"VS",self.y)

class international_cricket_match(cricket):
   def type_of_match(self):
      pass

class ODI(international_cricket_match):
   overs=50

class Test_match(international_cricket_match):
   overs_per_day=90
   days_played=5

class worldcup_match(international_cricket_match):
   conducted_span_years=4

class T20_Womens_worldcup(worldcup_match):
   overs=20

class T20_Mens_worldcup(worldcup_match):
   overs=20
```

```
class Under19_worldcup(worldcup_match):
   overs=20
   players_age_limit=19

obj=international_cricket_match()

obj.match_info("LONDON","LORD'S")

team1=input("Team 1st:")
team2=input("Team 2nd:")

obj.name_teams(team1,team2)
```

OUTPUT:

```
Cricket Match!
Teams=2 and players in each team=11
The location of match: LORD'S , LONDON

Team 1st:ENGLAND
Team 2nd:INDIA

ENGLAND VS INDIA
```

Create the problem formulation of Ludo game.

INITIAL STATE:-

1. Number_of_Players=2 to 4
2. Tokens_position=ZERO(for all players)
3. Starting_player with red color tokens
4. Dice=NOT ROLLED ONCE YET

ACTIONS:-

1. If game=INITIAL STATE, roll_dice()
   [players rolls dice by turn in a clockwise direction]
2. Move_dice() –(up,left,right)
   [Place token in box by counting as per the face value in dice]
3. Attack_opponent() – attack opponent tokens counting dice value (if in your leading way)

TRANSITION MODEL:-

1. If value in dice=6, either move new token from start point to star position or move old token already out from box.
2. New Position of token=Old Position of token + Moves as per dice face value
3. If attacked by opponent move your token to start point.

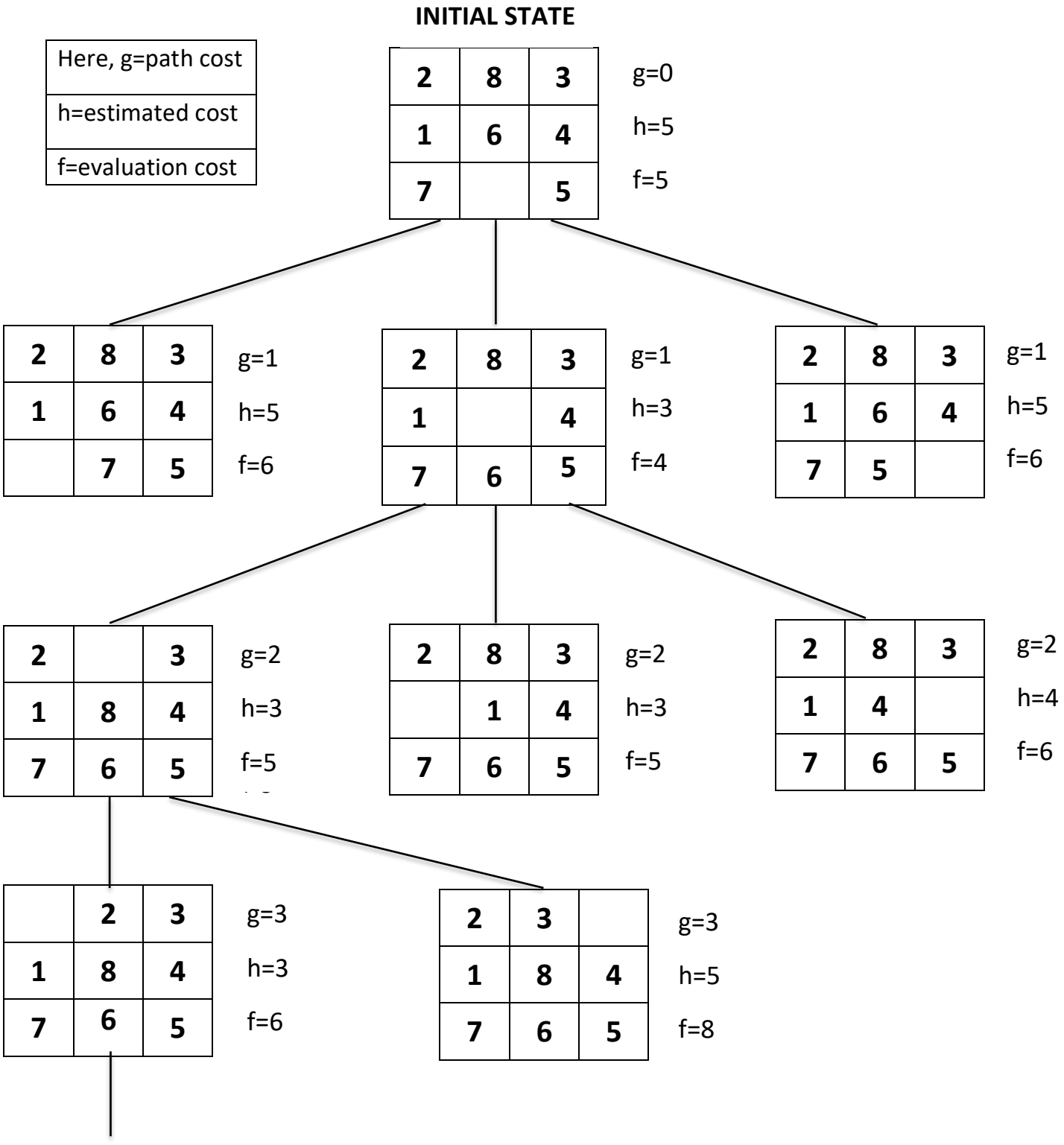
GOAL STATE:-

1. LUDO WIN- If all tokens reach home ahead of all other opponents
2. LUDO LOSE- If all of the tokens of other opponents reached home

PATH COST:-

1. Max_roll_dice()=(6,6,5)
2. Ignore_roll_dice()=(6,6,6)[no movement for token]
3. Path cost=count steps moved from current box to transition box

Complete the 8 puzzle problem given in the slides using A* algorithm.



CONTINUED FROM ABOVE--