Homework #5: Chapter 10,13

1. [20 points] Air cargo problem in Figure 10.1 on page 369 of the book
   a) [10 points] Using forward state-space search, draw a search tree to depth 1.

   ![Forward State-Space Search Tree]

   - Unload(C2,P1,LAX) \[At(P1,LAX) \land At(P2,JFK) \land At(C1,LAX) \land At(C2,LAX)\]
   - Load(C1,P1,LAX) \[At(P1,LAX) \land At(P2,JFK) \land In(C1,P1) \land In(C2,P1)\]
   - Fly(P1,LAX,LAX) \[At(P1,LAX) \land At(P2,JFK) \land At(C1,LAX) \land In(C2,P1)\]
   - Fly(P1,LAX,JFK) \[At(P1,JFK) \land At(P2,JFK) \land At(C1,LAX) \land In(C2,P1)\]
   - Fly(P1,LAX,ORD) \[At(P1,ORD) \land At(P2,JFK) \land At(C1,LAX) \land In(C2,P1)\]
   - Fly(P2,JFK,JFK) \[At(P1,LAX) \land At(P2,JFK) \land At(C1,LAX) \land In(C2,P1)\]
   - Fly(P2,JFK,LAX) \[At(P1,LAX) \land At(P2,LAX) \land At(C1,LAX) \land In(C2,P1)\]
   - Fly(P2,JFK,ORD) \[At(P1,LAX) \land At(P2,ORD) \land At(C1,LAX) \land In(C2,P1)\]

   Also acceptable if they create ground actions instead of using variables e.g.
   \[At(P1,LAX) \land At(P2,ORD) \land At(C1,JFK) \land In(C2,P2)\] and \[Fly(P1,LAX,JFK)\]. In that case they should have 6 predecessor nodes.

   Note the variables \(p\) and \(a\) in Unload and Load. The planning algorithm does not know
   that a plane or cargo cannot be in more than one place at once so these are technically
   needed.

   b) [10 points] Now draw the search tree to depth 1, but use backward state-space search.

   ![Backward State-Space Search Tree]

   - At(P1,x) \land At(P2,ORD) \land At(C1,JFK) \land In(C2,P2)
   - Fly(P1,x,JFK), x \neq JFK
   - At(P1,JFK) \land At(P2,x) \land At(C1,JFK) \land In(C2,P2)
   - Fly(P2,x,ORD), \ x \neq ORD
   - Unload(C1,p,JFK)
   - At(P1,JFK) \land At(P2,ORD) \land At(C1,JFK) \land In(C2,P2)
   - Load(C2,P2,a)
2. [30 points] Kitchen cleaning PDDL
   a) [5 points] We need the constants: Stove, Refrigerator, Floor, Oven, OvenCleaner, GarbageCan, Counters, Sink.
      We need the predicates: Clean(x), Swept(x), Applied(x, y)
   b) [15 points]
      - Action(CleanStove(),
        PRECOND: ⊥,
        EFFECT: Clean(Stove) ∧ ¬Clean(Floor))
      - Action(CleanRefrigerator(),
        PRECOND: ⊥,
        EFFECT: Clean(Refrigerator) ∧ ¬Clean(Floor) ∧ ¬Clean(GarbageCan) ∧ ¬Clean(Counters))
      - Action(ApplyCleaner(),
        PRECOND: ⊥,
        EFFECT: Applied(OvenCleaner, Oven) ∧ ¬Clean(Oven))
      - Action(RemoveCleaner(),
        PRECOND: Applied(OvenCleaner, Oven),
        EFFECT: Clean(Oven) ∧ ¬Applied(OvenCleaner, Oven))
      - Action(TakeOut(),
        PRECOND: ⊥,
        EFFECT: Clean(GarbageCan))
      - Action(SweepFloor(),
        PRECOND: Clean(GarbageCan),
        EFFECT: Swept(Floor))
      - Action(WashFloor(),
        PRECOND: Swept(Floor),
        EFFECT: Clean(Floor) ∧ ¬Clean(Sink))
      - Action(CleanCounter(),
        PRECOND: ⊥,
        EFFECT: Clean(Counter) ∧ ¬Clean(Sink))
      - Action(CleanSink(),
        PRECOND: ⊥,
        EFFECT: Clean(Sink))
   c) [5 points] This answer depends on what might be initially clean and what we consider clean in the end (e.g. maybe taking out the garbage is not considered “cleaning”).
      Init(⊥)
      Goal(Clean(Stove) ∧ Clean(Refrigerator) ∧ Clean(Floor) ∧ Clean(Oven) ∧ Clean(OvenCleaner) ∧ Clean(GarbageCan) ∧ Clean(Counters) ∧ Clean(Sink))
d) [5 points] The actions can be performed in any order that does not violate the following DAG (this graph does not need to be shown in the answer):
3. [30 points] Consider the problem of putting on one’s shoes and socks:
   a) [25 points] Planning graph and mutexes:

   \[ S_0 \quad A_0 \quad S_1 \quad A_1 \quad S_2 \]

   \[ \neg \text{LeftSockOn} \quad \neg \text{LeftSockOn} \quad \neg \text{RightSockOn} \quad \neg \text{RightSockOn} \quad \neg \text{RightSockOn} \]

   \[ \text{LeftSock} \quad \text{LeftSock} \quad \text{RightSock} \quad \text{RightSock} \quad \text{RightSock} \]

   \[ \text{LeftShoe} \quad \text{LeftShoe} \quad \text{RightShoe} \quad \text{RightShoe} \quad \text{LeftShoe} \]

   \[ \neg \text{LeftShoeOn} \quad \neg \text{LeftShoeOn} \quad \neg \text{RightShoeOn} \quad \neg \text{RightShoeOn} \quad \neg \text{LeftShoeOn} \]

   \[ S_0: \text{no mutex} \]

   \[ A_0 \]

   \[ \begin{array}{|l|l|l|} 
   \hline 
   \text{LeftSock} & \text{Persist } \neg \text{LeftSockOn} & \text{Inc. effects on LeftSockOn} \\
   \hline 
   \text{RightSock} & \text{Persist } \neg \text{RightSockOn} & \text{Inc. effects on RightSockOn} \\
   \hline 
   \end{array} \]

   \[ S_1 \]

   \[ \begin{array}{|l|l|l|} 
   \hline 
   \text{LeftSockOn} & \neg \text{LeftSockOn} & \text{negation} \\
   \hline 
   \text{RightSockOn} & \neg \text{RightSockOn} & \text{negation} \\
   \hline 
   \end{array} \]
\[ A_1 \]

<table>
<thead>
<tr>
<th>LeftSock</th>
<th>Persist (\neg)LeftSockOn</th>
<th>Inc. effects on LeftSockOn</th>
</tr>
</thead>
<tbody>
<tr>
<td>RightSock</td>
<td>Persist (\neg)RightSockOn</td>
<td>Inc. effects on RightSockOn</td>
</tr>
<tr>
<td>LeftShoe</td>
<td>Persist (\neg)LeftShoeOn</td>
<td>Inc. effects on (\neg)LeftShoeOn</td>
</tr>
<tr>
<td>RightShoe</td>
<td>Persist (\neg)RightShoeOn</td>
<td>Inc. effects on (\neg)RightShoeOn</td>
</tr>
<tr>
<td>LeftSock</td>
<td>LeftShoe</td>
<td>Competing need on LeftSockOn</td>
</tr>
<tr>
<td>RightSock</td>
<td>RightShoe</td>
<td>Competing need on RightSockOn</td>
</tr>
<tr>
<td>LeftSock</td>
<td>Persist LeftSockOn</td>
<td>Competing need on LeftSockOn</td>
</tr>
<tr>
<td>RightSock</td>
<td>Persist RightSockOn</td>
<td>Competing need on RightSockOn</td>
</tr>
<tr>
<td>Persist LeftSockOn</td>
<td>Persist (\neg)LeftSockOn</td>
<td>Competing need on LeftSockOn</td>
</tr>
<tr>
<td>Persist RightSockOn</td>
<td>Persist (\neg)RightSockOn</td>
<td>Competing need on RightSockOn</td>
</tr>
<tr>
<td>LeftShoe</td>
<td>Persist (\neg)LeftSockOn</td>
<td>Interference on LeftSockOn</td>
</tr>
<tr>
<td>RightShoe</td>
<td>Persist (\neg)RightSockOn</td>
<td>Interference on RightSockOn</td>
</tr>
</tbody>
</table>

\[ S_2 \]

<table>
<thead>
<tr>
<th>LeftSockOn</th>
<th>(\neg)LeftSockOn</th>
<th>negation</th>
</tr>
</thead>
<tbody>
<tr>
<td>RightSockOn</td>
<td>(\neg)RightSockOn</td>
<td>negation</td>
</tr>
<tr>
<td>LeftShoeOn</td>
<td>(\neg)LeftShoeOn</td>
<td>negation</td>
</tr>
<tr>
<td>RightShoeOn</td>
<td>(\neg)RightShoeOn</td>
<td>negation</td>
</tr>
<tr>
<td>LeftShoeOn</td>
<td>(\neg)LeftShoeOn</td>
<td>inconsistent support</td>
</tr>
<tr>
<td>RightShoeOn</td>
<td>(\neg)LeftSockOn</td>
<td>inconsistent support</td>
</tr>
</tbody>
</table>

a) [5 points] LeftShoeOn and RightShoeOn first appear in \( S_2 \) and they are not mutex. So the estimated cost to the goal is 2 with max-level, 4 with level-sum, and 2 with set-level.
4. [10 points, 2 points each]

a) \[ P(\neg a) = \]
\[ P(\neg a \land b \land c) + P(\neg a \land b \land \neg c) + P(\neg a \land \neg b \land c) + P(\neg a \land \neg b \land \neg c) = \]
\[ 0.2 + 0.05 + 0.01 + 0.04 = 0.3 \]

b) \[ P(c) = P(a \land b \land c) + P(\neg a \land b \land c) + P(a \land \neg b \land c) + P(\neg a \land \neg b \land \neg c) = 0.1 + 0.2 + 0.2 + 0.01 = 0.5 \]

1
\[ P(C) = c, \neg c = \langle 0.51, 0.49 \rangle \]

c) \[ P(a \land \neg b) = P(a \land b \land c) + P(a \land \neg b \land \neg c) = 0.2 + 0.25 = 0.45 \]

d) \[ P(c \lor a) = 1 - P(\neg a \land c) = 1 - (P(\neg a \land b \land c) + P(\neg a \land \neg b \land c)) = 1 - (0.2 + 0.01) = 0.79 \]

e) \[ P(\neg a \mid b \land c) = P(\neg a \land b \land c) / P(b \land c) = 0.2 / (P(a \land b \land c) + P(\neg a \land b \land c)) = 0.2 / (0.1 + 0.2) = 2/3 \]

5. [10 points] Consider the set of all possible five-card poker hands dealt fairly from a standard deck of fifty-two cards. Such a deck has 13 cards from each suit (hearts, clubs, spades, and diamonds). These 13 cards have ranks of Ace, Jack, Queen, King, and the numbers 2 through 10, inclusive.

a) There are 52 choose 5 = 52!/(5!47!) = 2,598,960 events

b) Each hand is equally likely so each has a probability of 1/(52 choose 5)

c) A royal flush always has the same number values, so there are only four possible royal flushes: one for each suit. So the probability is 4/(52 choose 5).

d) There are 13 choices for the value of the four-of-a-kind and the fifth card in the hand can be any of the 52-4=48 remaining cards in the deck. So there are 13*48=624 different four-of-a-kind hands. So the probability of getting one is 624/(52 choose 5)