COMP 6721 Applied Artificial Intelligence (Summer 2023)
Assignment #1
SOLUTION

Question 1) Time Machine

a) We saw that the main issues that resulted in the first AI winter were

- Relying too much on introspection, i.e., how human perform a task, instead of studying the structure of the task itself
- Overconfidence
- Overestimate and unrealistic expectations
- Lack of appreciation of intractability of larger problems (scaling and combinatorial explosion)
- Limitations of basic structures and algorithms

Therefore, we may discuss with the AI experts that **(any two for 2 points)**

- It is more rewarding and practical to work on the engineering and rational approach of AI, not to focus on necessarily how human act or think.
- We should not be over-confident considering the preliminary outcomes and look at them as the stepping stones for future
- We should not expect AI to become superior to humans in a general way (strong AI). It will take a long time (more than 50-60 years) to have an agent that may pass the Turing Test.
- The exponential complexity of some problems requires novel specific algorithms and data structures in addition to more powerful hardware.
- Multilayer perceptron networks can be very powerful and will become a breakthrough in AI but the requirements are to have a lot of data (Big Data) and very powerful machines with vector processing capabilities.

b) We may discuss any deep learning method or architecture. **[1p]**

Question 2) Prince looking for Cinderella

Assume that House W is empty, and the Witch is not home. Discuss and reason if the following statements are Ture.

(i) BFS Search expands (sending to closed list) more nodes than DFS Search

TRUE
With simple BFS
Open list = {Start}, Closed = {}
Open list = {A,B,C}, Closed = {Start}
Open list = {B, C, D, F}, Closed = {A, Start}
Open list = {C, D, F}, Closed = {B, A, Start}
Open list = {D, F, E}, Closed = {C, B, A, Start}
Open list = {F, E}, Closed = {D, C, B, A, Start}
Open list = {E, L}, Closed = {F, D, C, B, A, Start}
Open list = {L, W}, Closed = {E, F, D, C, B, A, Start}
Open list = {W}, Closed = {L, E, F, D, C, B, A, Start}
So, in addition to Start and L, we expanded (visited) 6 nodes (A-F)

With simple DFS
Open list = {Start}, Closed = {}
Open list = {A,B,C}, Closed = {Start}
Open list = {D, F, B, C}, Closed = {A, Start} [Since B is already in the list, we do not rearrange the list. Otherwise, B will become the first node in the Open List]
Open list = {E, F, B, C}, Closed = {D,A, Start}
Open list = {L, W, F, B, C}, Closed = {E, D,A, Start}
Open list = {W, F, B, C}, Closed = {L, E, D,A, Start}
So, in addition to Start and L, we expanded (visited) 3 nodes (A, D, E)

(ii) DFS Search can find a path to Cinderella's house

TRUE, we just saw that DFS found a path.

(iii) DFS Search <u>finds the shortest path</u> to Cinderella's house

FALSE. The shortest path is Start-A-F-L or Start-C-E-L at depth 3, but DFS found a path Start-A-D-E-L at depth 4.

(iv) Depth-limited search with depth=3 can find a path to Cinderella's house

Open list = {Start (0)}, Closed = {}
Open list = {A (1), B(1), C(1)}, Closed = {Start}
Open list = {D(2), F(2), B(1), C(1)}, Closed = {A(1), Start}
Open list = {E(3), F(2), B(1), C(1)}, Closed = {D(2), A(1), Start}
We do not expand E due to the depth limitation (already 3)
Open list = {F(2), B(1), C(1)}, Closed = {E(3), D(2), A(1), Start}
Open list = {L(3), B(1), C(1)}, Closed = {F(2), E(3), D(2), A(1), Start}
Open list = {B(1), C(1)}, Closed = { L(3), F(2), E(3), D(2), A(1), Start}

However, it may be argued that we don't do even the goal check because the depth of L is already equal to 3. So, it depends on the implementation, but it is more practical to do the goal check test but not to expand the node if the depth is already at the limit.

(v) Uniform cost Search (UCS) finds the shortest path to Cinderella's house

TRUE since without the presence of border conditions (e.g., negative weights), UCS always finds the lowest cost path. Here, there is no explicit costs, so it will be the same as BFS.

Open list = {Start (0)}, Closed = {}
Open list = {A (1),B(1),C(1)}, Closed = {Start (0)}
Open list = {B(1), C(1), D(2), F(2)}, Closed = {A(1), Start(0)}
Open list = {C(1), D(2), F(2)}, Closed = {B(1), A(1), Start(0)}
Open list = {D(2), **E(2), F(2)},** Closed = {C(1), B(1), A(1), Start(0)} (same depth so alphabetical)
Open list = {E(2), F(2)}, Closed = {D(2), C(1), B(1), A(1), Start(0)}
Open list = {F(2), L(3), W(3)}, Closed = {E(2), D(2), C(1), B(1), A(1), Start(0)}
Open list = {L(3), W(3)}, Closed = {F(2), E(2), D(2), C(1), B(1), A(1), Start(0)}
Open list = {W(3)}, Closed = {**L(3),** F(2), E(2), D(2), C(1), B(1), A(1), Start(0)}

(vi) With DFS Search, Prince can reach Cinderella before the Witch

True. We saw that considering the alphabetical order assumption, L was expanded before W.

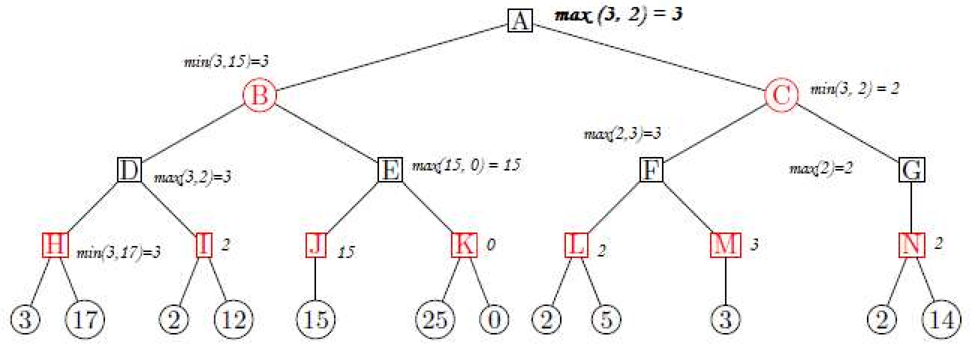
(vii) With BFS Search, Prince can reach Cinderella before the Witch

True. We saw that considering the alphabetical order assumption, L was expanded before W.

Question 3

Source: https://inst.eecs.berkeley.edu/~cs188/fa22/assets/exam/cs188-fa22-midterm-solutions.pdf

(i) [1 pt] Run minimax. What will be the value of the root (A)?
Equal to 3



(ii) [3 pts] After running minimax, indicate the branches that will be pruned with alpha-beta pruning? You can indicate an edge by the two sides (e.g., B-E or L-5). If there is a "deep cut", mention it. If pruning occurs at a lower depth, just indicate the main edge, and explain if all successor edges will also be pruned. **I -12, E-K, L-5, and C-G**

After running minimax on H, the minimax value of H becomes 3, so the alpha (lower bound) of D would become 3 (It is a max node). When I-2 is calculated equal to 2, the beta of I will become 2 (upper bound), but we see that the beta of a min node is now below the alpha of the parent (max node) (2 < 3). So I-12 will be pruned (regardless of the value of I-15, the value of D will not change).

After the step above and with the minimax value (D) =3, B will expect something <3 from the E side. As soon as we see J=15, the lower bound of E becomes 15 (it secured at least 15 from the J side). So, E will eventually send a value >=15 to B while we just discussed that B will consider only values of < 3. Therefore, E-K (and leaves 25 and 0) will be pruned. All successors of E-K are thus pruned.

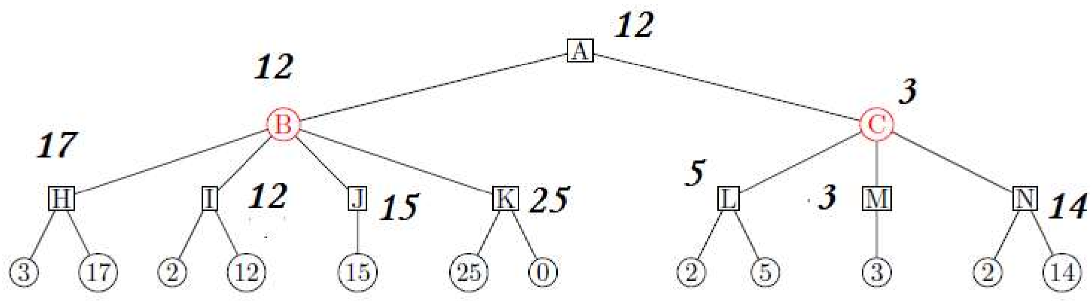
After the two steps above, the minimax value (B) is determined to be 3, so A (max node) will only consider > 3 values from the right (C) side.

After L-2, we have minimax value (L) <=2, so L will not send eventually any interesting value to A because A only considers values >3 from the C side. So, L-5 can also be pruned.

The minimax value of L is 2 and M's is 3, so the minimax value of F will be max(2,3)=3. Therefore, the upper bound of C (min node) will become 3. But we just said that A is interested only in >3 values from C. So regardless of the C right branch, the minimax value of A will be set to 3, and C-G is pruned. This is a deep cut. All successors of E-G are thus pruned.

(iii) Assume we are very generous, and we want to give two rounds to the Opponent (Min in red), so allow Min() to be run also at the third level and max() in the fourth level (D, E, F, G become Red running Min() instead of Max() and H-N will run Max()) (The layers will be thus Max(), Min(), Min() then Max()). We want to run Minimax with alpha-beta pruning. Which new edges or leaf nodes will be pruned?

If we have two min() levels back to back, we can combine them and have just one min() level because *min (min())* is applied. For example, instead of having B = min(D,E) = min (min(H, I), min (J,K)), we can just write B=min(H, I, J, K).



When the minimax value of B is determined to be 12, the lower bound of A will be 12, so A is interested only in > 12 values from the C side. After the determination of minimax(L) =5, we know that C is a Min node so what C will eventually send to A will be less than or equal to 5 (<=5). But it was discussed that A is interested only in>12 values, so C-M and C-N can be pruned.

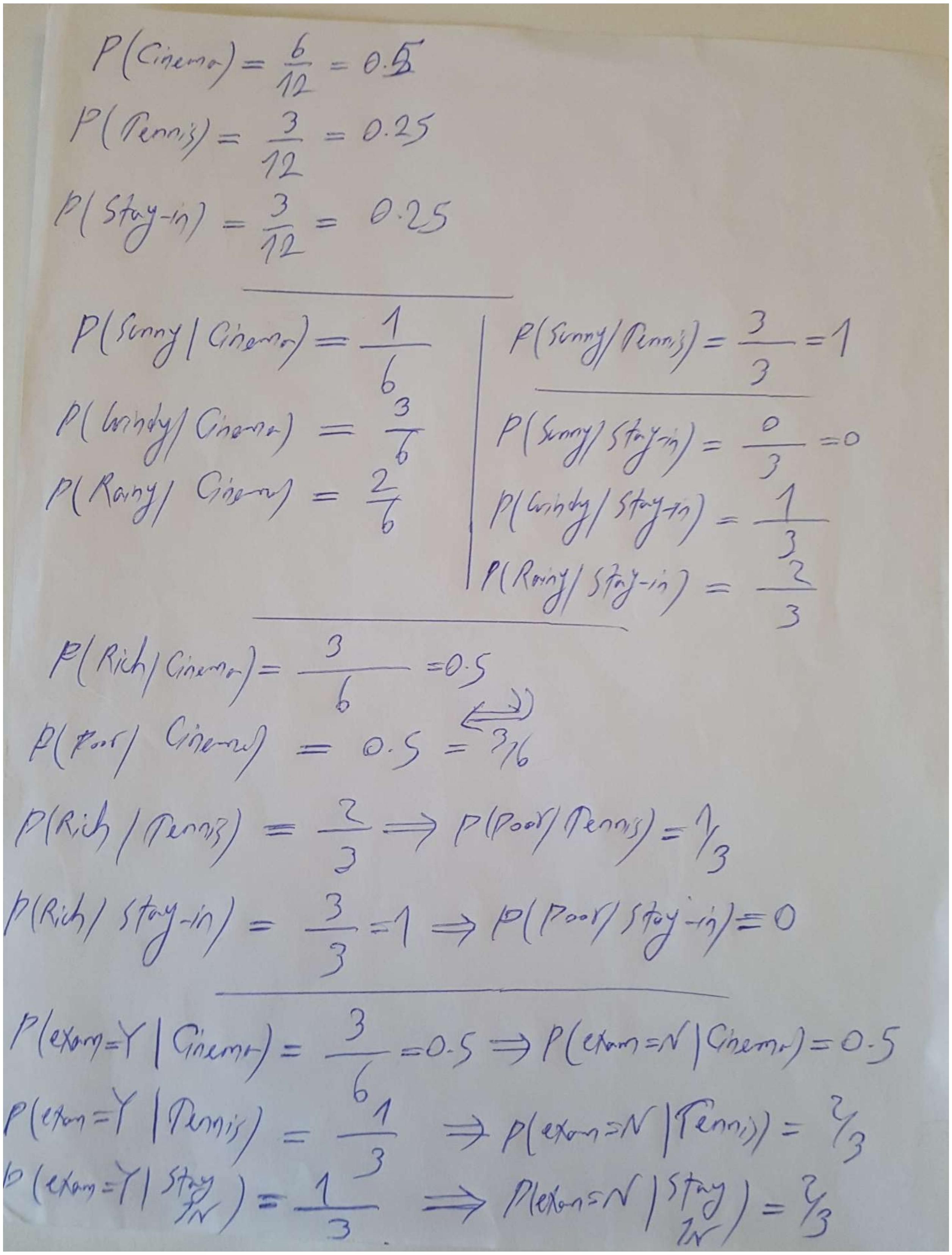
(iv) Now assume all the games with the same structure (back to the original tree) but any arbitrary values on the leaves. In the best and worst case, how many leaves will be pruned by alpha-beta pruning?

We know that the minimum is 0 nodes because in the worst case, we may not have any pruning.

The maximum number of leaves that may be pruned will be 12-6 = 6 because out of 12 leaves, 6 nodes with the current values of 3, 17, 2, 15, 2, 3 should always be checked to make any pruning decision on the others.

For the first 3 (nodes with current values of 3, 17, and 2) it seems evident. Why the node with the current value of 15 cannot be pruned? Because it is the most left leaf for J and E (E's lower and upper bounds are not initialized yet). However, depending on this node's value that is sent to E and B's current expectation, E-K may be pruned.

Question 4



$P(\text{Cinema}) = \frac{6}{12} = 0.5$

$P(\text{Tennis}) = \frac{3}{12} = 0.25$

$P(\text{Stay-in}) = \frac{3}{12} = 0.25$

$P(\text{Sunny} \mid \text{Cinema}) = \frac{1}{6}$

$P(\text{Windy} / \text{Cinema}) = \frac{3}{6}$

$P(\text{Rainy} / \text{Cinema}) = \frac{2}{6}$

$P(\text{Sunny} / \text{Tennis}) = \frac{3}{3} = 1$

$P(\text{Sunny} / \text{Stay-in}) = \frac{0}{3} = 0$

$P(\text{Windy} / \text{Stay-in}) = \frac{1}{3}$

$P(\text{Rainy} / \text{Stay-in}) = \frac{2}{3}$

$P(\text{Rich} / \text{Cinema}) = \frac{3}{6} = 0.5$

$P(\text{Poor} / \text{Cinema}) = 0.5 \Leftarrow = 3/6$

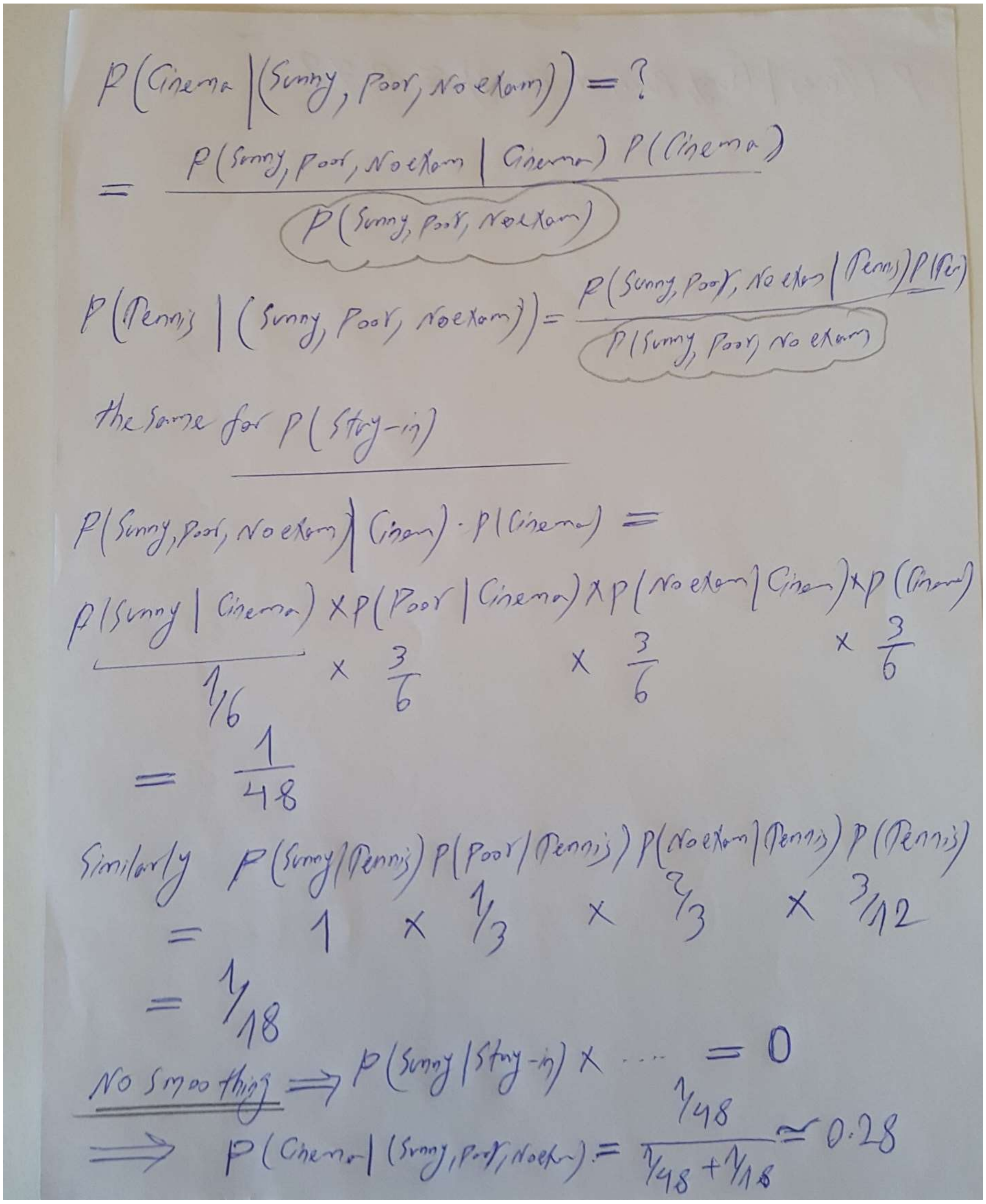
$P(\text{Rich} / \text{Tennis}) = \frac{2}{3} \Rightarrow P(\text{Poor} / \text{Tennis}) = 1/3$

$P(\text{Rich} / \text{Stay-in}) = \frac{3}{3} = 1 \Rightarrow P(\text{Poor} / \text{Stay-in}) = 0$

$P(\text{exam}=Y \mid \text{Cinema}) = \frac{3}{6} = 0.5 \Rightarrow P(\text{exam}=N \mid \text{Cinema}) = 0.5$

$P(\text{exam}=Y \mid \text{Tennis}) = \frac{1}{3} \Rightarrow P(\text{exam}=N \mid \text{Tennis}) = 2/3$

$P(\text{exam}=Y \mid \text{Stay IN}) = \frac{1}{3} \Rightarrow P(\text{exam}=N \mid \text{Stay IN}) = 2/3$



$$P(\text{Cinema} \mid (\text{Sunny}, \text{Poor}, \text{No exam})) = ?$$

$$= \frac{P(\text{Sunny}, \text{Poor}, \text{No exam} \mid \text{Cinema})\, P(\text{Cinema})}{\boxed{P(\text{Sunny}, \text{Poor}, \text{No exam})}}$$

$$P(\text{Tennis} \mid (\text{Sunny}, \text{Poor}, \text{No exam})) = \frac{P(\text{Sunny}, \text{Poor}, \text{No exam} \mid \text{Tennis})\, P(\text{Ten})}{\boxed{P(\text{Sunny}, \text{Poor}, \text{No exam})}}$$

the same for $P(\text{Stay-in})$

$$P(\text{Sunny}, \text{Poor}, \text{No exam} \mid \text{Cinema}) \cdot P(\text{Cinema}) =$$

$$\underline{P(\text{Sunny} \mid \text{Cinema})} \times P(\text{Poor} \mid \text{Cinema}) \times P(\text{No exam} \mid \text{Cinema}) \times P(\text{Cinema})$$

$$1/6 \times \frac{3}{6} \times \frac{3}{6} \times \frac{3}{6}$$

$$= \frac{1}{48}$$

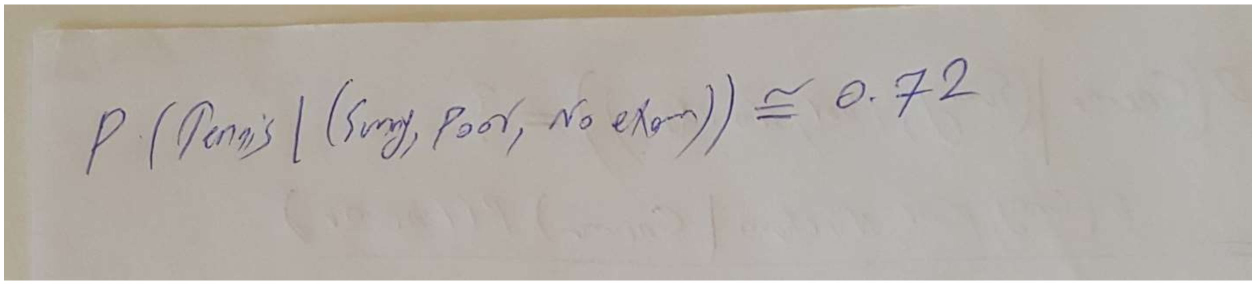
Similarly $P(\text{Sunny} \mid \text{Tennis})\, P(\text{Poor} \mid \text{Tennis})\, P(\text{No exam} \mid \text{Tennis})\, P(\text{Tennis})$

$$= 1 \times 1/3 \times 2/3 \times 3/12$$

$$= 1/18$$

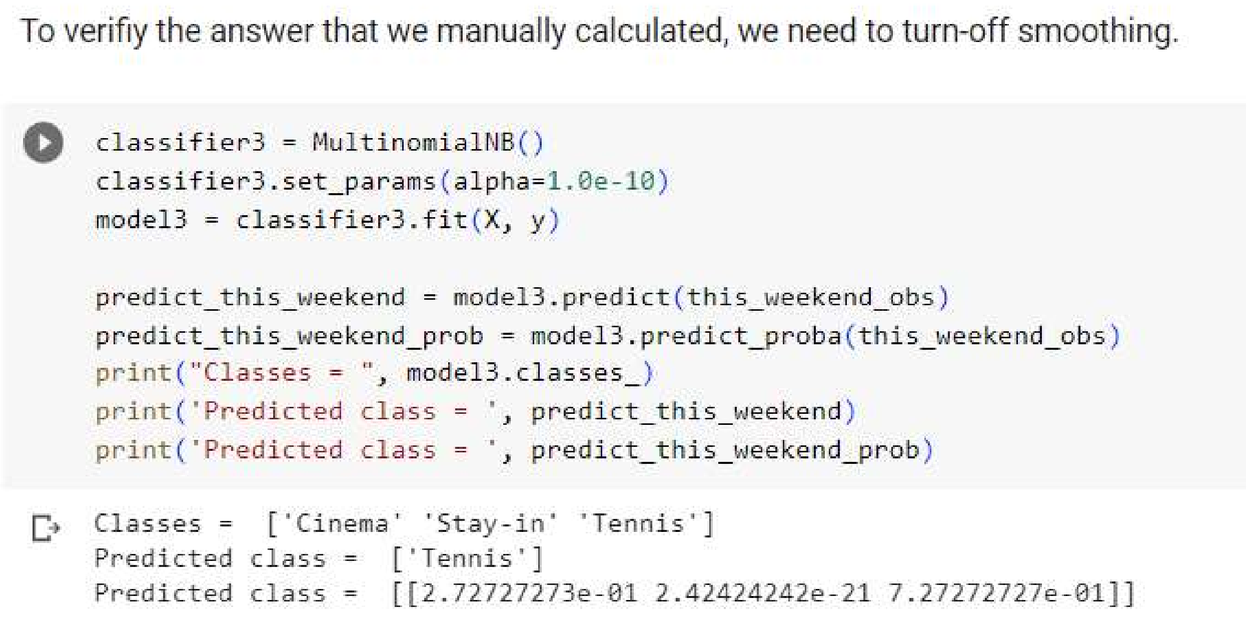
$\underline{\underline{\text{No Smoothing}}} \Longrightarrow P(\text{Sunny} \mid \text{Stay-in}) \times \dots = 0$

$$\Longrightarrow P(\text{Cinema} \mid (\text{Sunny}, \text{Poor}, \text{No exam})) = \frac{1/48}{1/48 + 1/18} \simeq 0.28$$



$P(\text{Tennis} \mid (\text{Sunny}, \text{Poor}, \text{No exam})) \cong 0.72$

b)



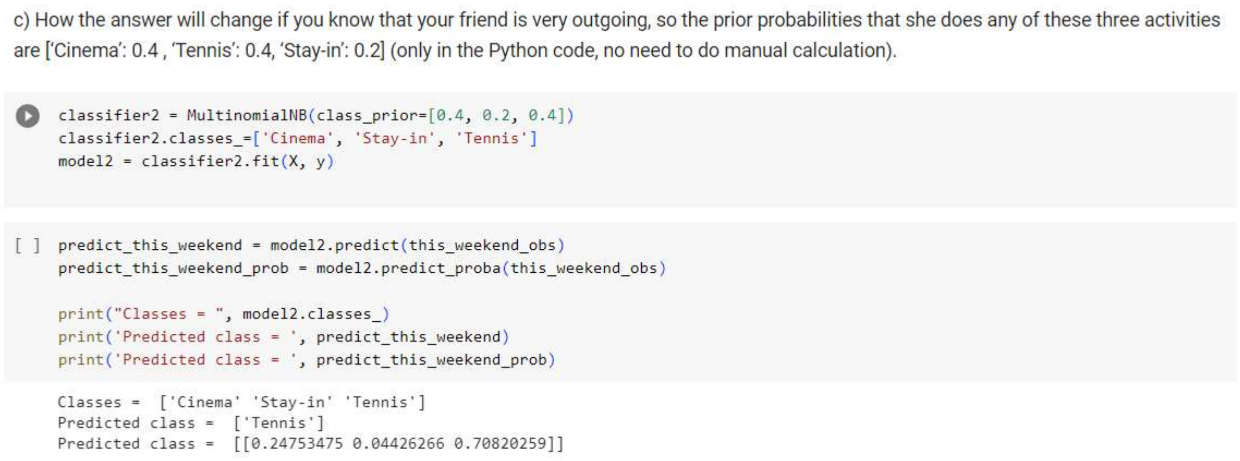
To verifiy the answer that we manually calculated, we need to turn-off smoothing.

```
classifier3 = MultinomialNB()
classifier3.set_params(alpha=1.0e-10)
model3 = classifier3.fit(X, y)

predict_this_weekend = model3.predict(this_weekend_obs)
predict_this_weekend_prob = model3.predict_proba(this_weekend_obs)
print("Classes = ", model3.classes_)
print('Predicted class = ', predict_this_weekend)
print('Predicted class = ', predict_this_weekend_prob)
```

```
Classes =  ['Cinema' 'Stay-in' 'Tennis']
Predicted class =  ['Tennis']
Predicted class =  [[2.72727273e-01 2.42424242e-21 7.27272727e-01]]
```

c)



c) How the answer will change if you know that your friend is very outgoing, so the prior probabilities that she does any of these three activities are ['Cinema': 0.4 , 'Tennis': 0.4, 'Stay-in': 0.2] (only in the Python code, no need to do manual calculation).

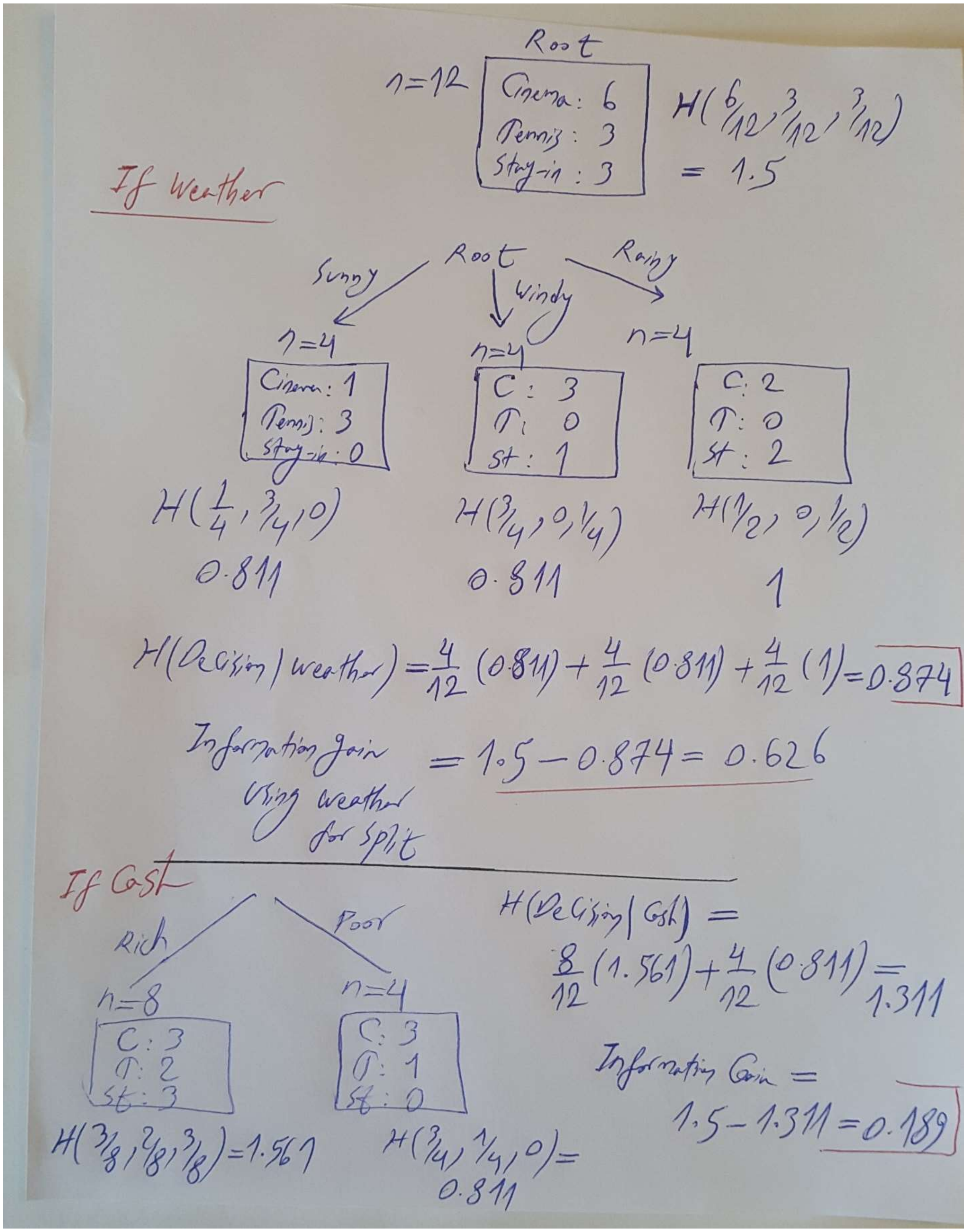
```
classifier2 = MultinomialNB(class_prior=[0.4, 0.2, 0.4])
classifier2.classes_=['Cinema', 'Stay-in', 'Tennis']
model2 = classifier2.fit(X, y)
```

```
predict_this_weekend = model2.predict(this_weekend_obs)
predict_this_weekend_prob = model2.predict_proba(this_weekend_obs)

print("Classes = ", model2.classes_)
print('Predicted class = ', predict_this_weekend)
print('Predicted class = ', predict_this_weekend_prob)
```

```
Classes =  ['Cinema' 'Stay-in' 'Tennis']
Predicted class =  ['Tennis']
Predicted class =  [[0.24753475 0.04426266 0.70820259]]
```

Question 5



Root

n=12

Cinema : 6
Tennis : 3
Stay-in : 3

$H(6/12, 3/12, 3/12) = 1.5$

If Weather

$$\begin{tikzcd}
 & \text{Root} \arrow[dl, "\text{Sunny}"'] \arrow[d, "\text{Windy}"] \arrow[dr, "\text{Rainy}"] & \\
n=4 & n=4 & n=4
\end{tikzcd}$$

Sunny, n=4	Windy, n=4	Rainy, n=4
Cinema : 1	C : 3	C : 2
Tennis : 3	T : 0	T : 0
Stay-in : 0	St : 1	St : 2
$H(\frac{1}{4}, 3/4, 0)$	$H(3/4, 0, 1/4)$	$H(1/2, 0, 1/2)$
0.811	0.811	1

$H(\text{Decision} \mid \text{weather}) = \frac{4}{12}(0.811) + \frac{4}{12}(0.811) + \frac{4}{12}(1) = \boxed{0.874}$

Information gain using weather for split $= \underline{1.5 - 0.874 = 0.626}$

If Cost

$$\begin{tikzcd}
 & \cdot \arrow[dl, "\text{Rich}"'] \arrow[dr, "\text{Poor}"] & \\
n=8 & & n=4
\end{tikzcd}$$

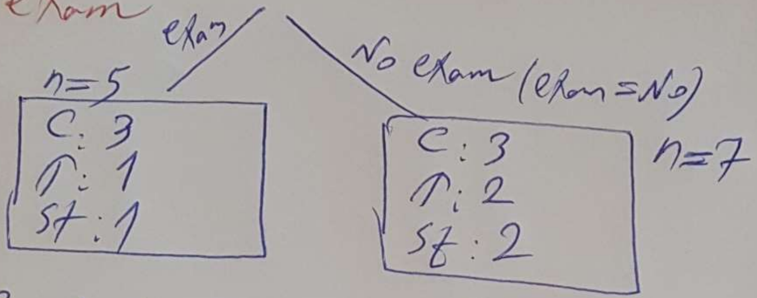
Rich, n=8	Poor, n=4
C : 3	C : 3
T : 2	T : 1
St : 3	St : 0
$H(3/8, 2/8, 3/8) = 1.561$	$H(3/4, 1/4, 0) =$ 0.811

$H(\text{Decision} \mid \text{Cost}) = \frac{8}{12}(1.561) + \frac{4}{12}(0.811) = 1.311$

Information Gain $=$

$1.5 - 1.311 = \boxed{0.189}$

If eXam

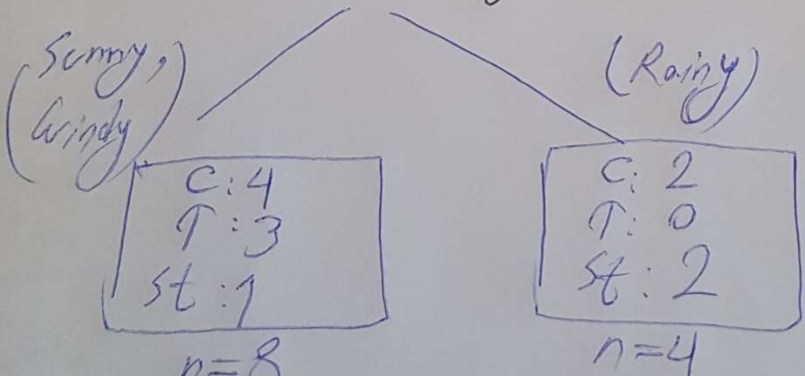


$H(3/5, 1/5, 1/5) = 1.371$ $H(3/7, 2/7, 2/7) = 1.557$

$H(\text{Decision} \mid \text{Exam}) = \frac{5}{12}(1.371) + \frac{7}{12}(1.557) =$
1.4795

$\text{Gain} = 1.5 - 1.4795 = \boxed{0.0205}$

Q2 If we can have only 2 children (one cut)



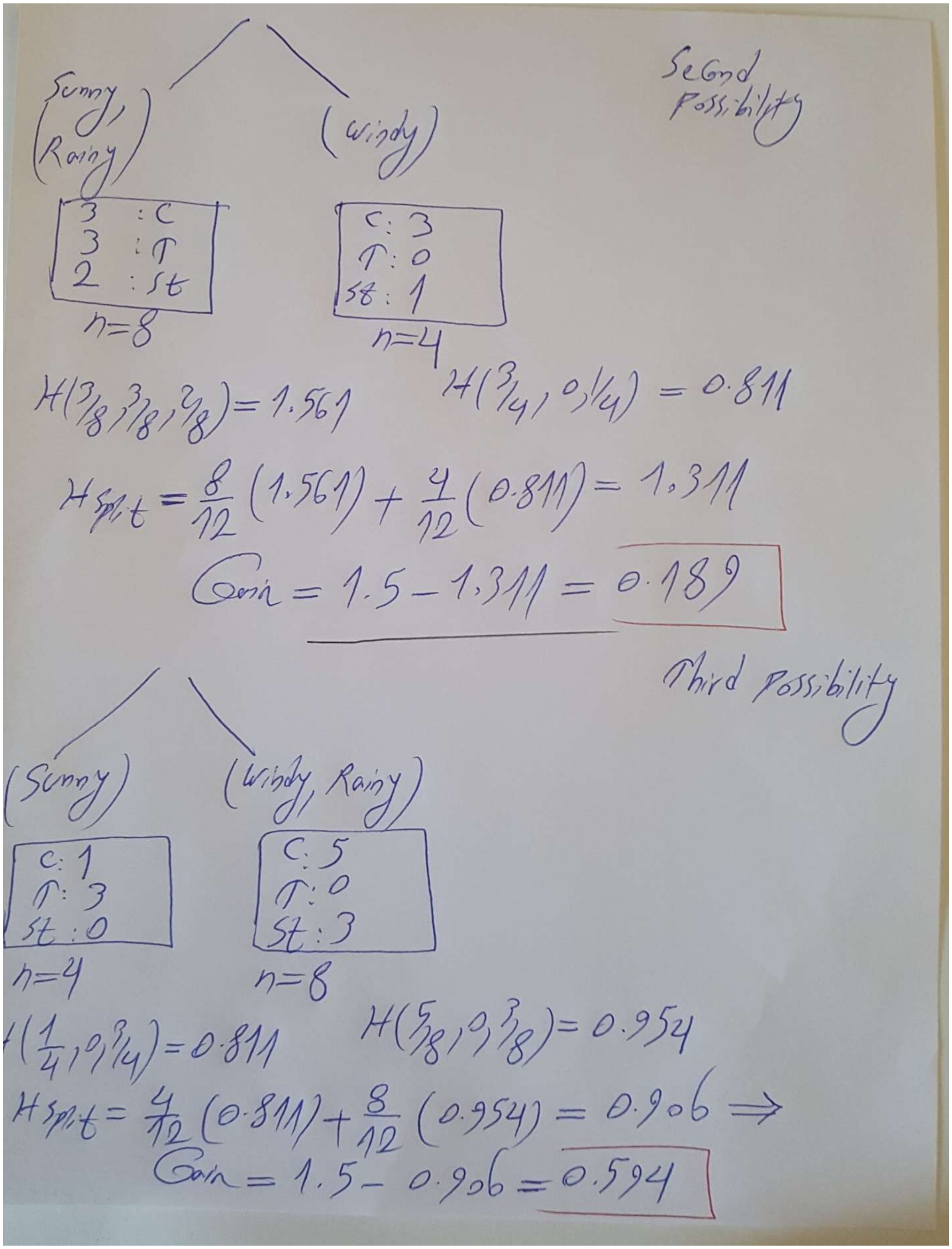
First possibility:

$H = H(4/8, 3/8, 1/8)$
$= 1.406$

$H = H(\frac{2}{4}, 0, \frac{2}{4}) = 1$

$H(\quad) = \frac{8}{12}(1.406) + \frac{4}{12}(1) = 1.2707$

$\text{Gain} = 1.5 - 1.2707 = \boxed{0.229}$



Second Possibility

(Sunny, Rainy)

3	: C
3	: T
2	: St

$n=8$

(Windy)

C	: 3
T	: 0
St	: 1

$n=4$

$H(3/8, 3/8, 2/8) = 1.561$ $\quad H(3/4, 0, 1/4) = 0.811$

$$H_{split} = \frac{8}{12}(1.561) + \frac{4}{12}(0.811) = 1.311$$

$$\text{Gain} = 1.5 - 1.311 = \boxed{0.189}$$

Third Possibility

(Sunny)

C	: 1
T	: 3
St	: 0

$n=4$

(Windy, Rainy)

C	: 5
T	: 0
St	: 3

$n=8$

$H(\frac{1}{4}, 0, 3/4) = 0.811$ $\quad H(5/8, 0, 3/8) = 0.954$

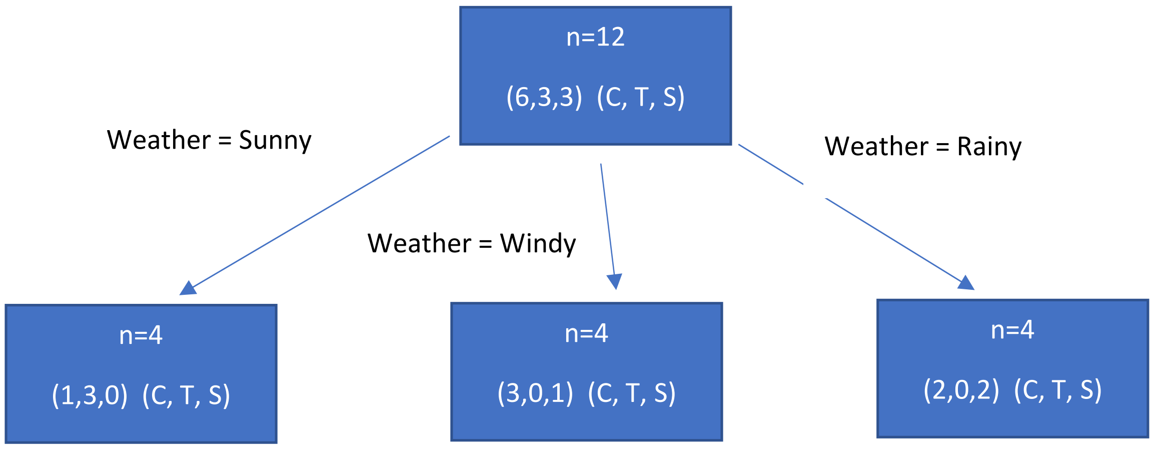
$$H_{split} = \frac{4}{12}(0.811) + \frac{8}{12}(0.954) = 0.906 \Rightarrow$$

$$\text{Gain} = 1.5 - 0.906 = \boxed{0.594}$$

a) The information gain obtained from using any of the three variables to do the split is as follow. Therefore, split based on weather will be the optimal split in the first round (starting from root).

Split using	**Information gain**
Weather (3 children)	0.626
Cash (2 children)	0.189
Exam (2 children)	0.0205

So, we select the variable Weather, and the resulted tree will be



Prediction = Tennis

Classification error = 1/4

Prediction = Cinema

Classification error = 1/4

Prediction = Cinema*[1]

Classification error = 2/4

Observation	**Weather**	**Cash**	**Exam**	**Decision**	**Prediction**	**Correct Prediction?**
1	Sunny	Rich	Yes	Cinema	Tennis	No
2	Sunny	Rich	No	Tennis	Tennis	Yes
3	Windy	Rich	No	Cinema	Cinema	Yes
4	Rainy	Poor	Yes	Cinema	Cinema	Yes
5	Rainy	Rich	No	Stay-in	Cinema	No
6	Rainy	Poor	No	Cinema	Cinema	Yes
7	Windy	Poor	Yes	Cinema	Cinema	Yes
8	Windy	Rich	Yes	Stay-in	Cinema	No
9	Windy	Rich	No	Cinema	Cinema	Yes
10	Sunny	Rich	No	Tennis	Tennis	Yes
11	Sunny	Poor	Yes	Tennis	Tennis	Yes
12	Rainy	Rich	No	Stay-in	Cinema	No

Accuracy: Is the number of Correct Prediction = "Yes" out of 12 observations, equal to 8/12 ≈ 0.67.

[1] For (2,0,2), we may have other criteria to decide, for example which class has majority in the whole dataset. Here, we assumed the decision will be Cinema (Stay-in will also be discussed).

Confusion matrix

True class is	Model predicts	Cinema	Tennis	Stay-In
Cinema		5	1	0
Tennis		0	3	0
Stay-In		3	0	0

Recall and precision per class

Class (0,1)	**Recall**	**Precision**	**F1-score**	
Cinema (6,6)	5/6 (Out of 6 real Cinema, 5 are correctly predicted)	5/8 (8 predicted cinema, 3 are wrong)	2pr/(p+r) = 0.714	
Tennis (9,3)	1 (3/3)	0.75 (3/4)	6/7 = 0.857	
Stay-In (9,3)	0	0	Undef	

IF FOR THE NODE (2,0,2), we assume that the prediction is stay-in:

Observation	**Weather**	**Cash**	**Exam**	**Decision**	**Prediction**	**Correct Prediction?**
1	Sunny	Rich	Yes	Cinema	Tennis	No
2	Sunny	Rich	No	Tennis	Tennis	Yes
3	Windy	Rich	No	Cinema	Cinema	Yes
4	Rainy	Poor	Yes	Cinema	**Stay-in**	**No**
5	Rainy	Rich	No	Stay-in	**Stay-in**	**Yes**
6	Rainy	Poor	No	Cinema	**Stay-in**	**No**
7	Windy	Poor	Yes	Cinema	Cinema	Yes
8	Windy	Rich	Yes	Stay-in	Cinema	No
9	Windy	Rich	No	Cinema	Cinema	Yes
10	Sunny	Rich	No	Tennis	Tennis	Yes
11	Sunny	Poor	Yes	Tennis	Tennis	Yes
12	Rainy	Rich	No	Stay-in	**Stay-in**	**Yes**

Accuracy: Does not change. 8/12 ≈ 0.67.

Recall and precision per class

Class (0,1)	**Recall**	**Precision**	**F1-score**	
Cinema (6,6)	3/6 (Out of 6 real Cinema 3 are correctly predicted)	3/4 (4 predicted cinema, 1 is wrong)	2pr/(p+r) = 18/30 = 0.6	
Tennis (9,3)	1 (3/3)	0.75 (3/4)	6/7 = 0.857	

Stay-In (9,3)	2/3	2/4	8/14 = 0.571	

2) [5ps]

We want to compare

- Information gain using weather with 3 children (2 cuts)
- Information gain using weather with 2 children (1 cut)

Structure	**Information gain**
using weather with 3 children (2 cuts)	0.626
using weather with 2 children (1 cut)	0.594

The first one was already calculated, and the gain was equal to 0.626. We now find the information gain of the split with only one cut. We will see that the gain is equal to 0.594**. So, if we have the option to decide one versus two cuts, two cuts resulting in three children will be the better decision.**

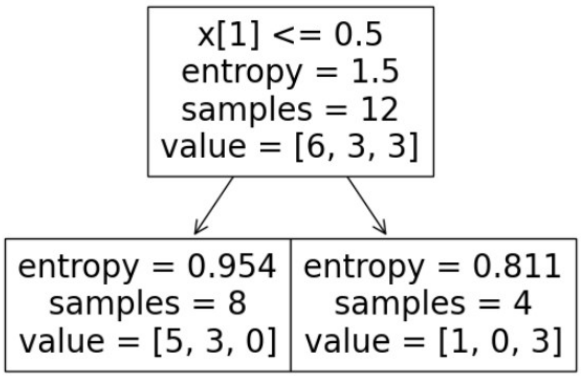
Calculation (please see the handwritten images attached).

If we cannot have a split with 2 cuts (3 children), and only 1 cut is allowed, we should consider the following possibilities and compare them:

- (Sunny, Windy) together as one child and (Rainy) as the other child
- (Sunny, Rainy) together as one child and (Windy) alone as the other child
- (Rainy, Windy) together as one child and (Sunny) alone as the other child

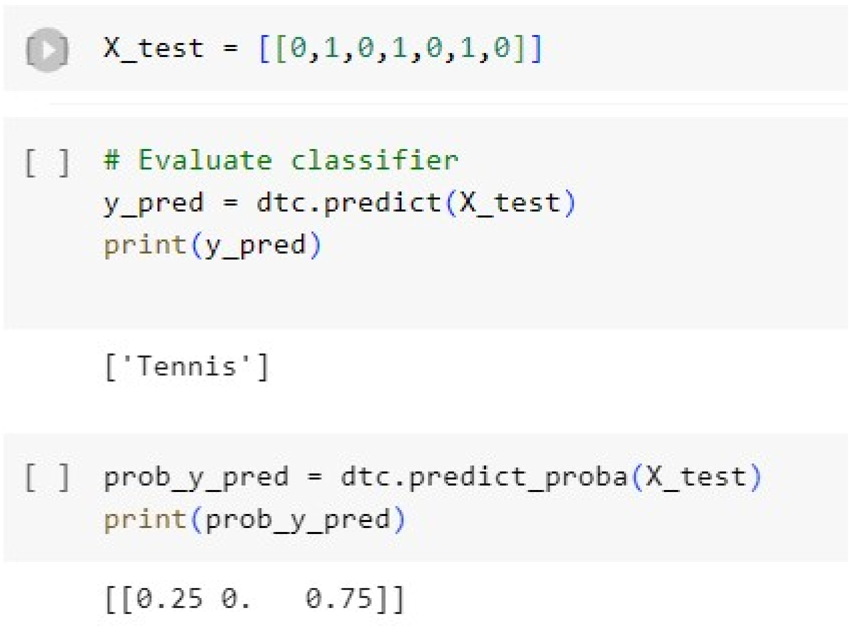
So, we should now check among these three which one is the best split for the case of only two children being allowed. That best split's gain will be the gain of a 2-children split.

Among the three possible splits with only 2 children, the best with the highest information gain is the one with [(Sunny), (Windy, Rainy)]. The information gain is 0.594. We can validate our result in the code, and we can see that Scikit-Learn also decided to use [(Sunny), (Windy, Rainy)].



3) [2pts]

Still Tennis is the most probable activity with 0.75 confidence.



```
X_test = [[0,1,0,1,0,1,0]]
```

```
# Evaluate classifier
y_pred = dtc.predict(X_test)
print(y_pred)
```

```
['Tennis']
```

```
prob_y_pred = dtc.predict_proba(X_test)
print(prob_y_pred)
```

```
[[0.25 0.   0.75]]
```

Question 6 [6 pts BONUS] Informed Search

a) [2pts] Is h1(n) consistent?

No. We can find examples where the inequality of h(n) ≤ c(n,n') + h(n') is not satisfied.

h(n)	**h(n')**	**c(n,n')**	**Condition**
h(B) = 12	**h(C) = 10**	**c(B,C) = 1**	**12 <= 1 + 10 ? No!**
h(D) = 8	**h(F) = 4.5**	**c(D,F) = 3**	**8 <= 3 + 4.5 ? No!**

b) [2 pts] Run Algorithm A with this heuristic. Show the intermediate steps and the found path

		Closed_list	open_list	Event
1	A is expanded	[A]	[f(B): 1+12 = 13 (via A), f(C): 4+10 = 14 (via A)]	
2	B is expanded (lower f(.))	[A, B]	[f(C): 2+10 = 12 (via B), f(D): 6+8=14 (via B)]	f(C) is updated through the edge BC
3	C is expanded	[A, B, C]	[f(D): 5+8=13 (via C)]	F(D) is updated via CD
4	D is expanded	[A, B, C, D]	[f(F): 8+4.5=12.5 (via D), f(E): 13+1=14 (via D), f(G): 14+0=14 (via D)]	
5	F is expanded	[A, B, C, D, F]	[f(E): 13+1=14 (via D), f(G): 13+0=13 (via F)]	

The returned path from Algorithm A is A-B-C-D-F-G .

c) [1pts] Which range of values of h2(B) results in h2() being admissible?
To be admissible, we should not overestimate the cost, so the estimated heuristic cost should be below the real cost. At B, the best optimal cost to the goal is via B-C-D-F-G equal to 12. The estimated value can not be negative, so the answer is **0 ≤ h3(B) ≤ 12**

d) [1 pts] which range of values of h2(B) results in h2() being consistent?

For consistency, we should make sure that the condition h(n) ≤ c(n,n') + h(n') is satisfied. We should check this condition for all links that involve B, i.e.,

h(A) ≤ c(A, B) + h(B)	10 <= 1 + h(B)	**9 <= h(B)**
h(C) ≤ c(C, B) + h(B)	9 <= 1 + h(B)	8 <= h(B)
h(D) ≤ c(D, B) + h(B)	7 <= 5 + h(B)	2 <= h(B)
h(B) ≤ c(B, C) + h(C)	h(B) ≤ 1+ 9	**h(B) ≤ 10**
h(B) ≤ c(B, D) + h(D)	h(B) ≤ 5 + 7	h(B) ≤ 12
h(B) ≤ c(B, A) + h(A)	h(B) ≤ 1 + 10	h(B) ≤ 11

Therefore, 9 ≤ h3(B) ≤ 10.

Question 7) [6 pts BONUS] Our version of Tic-Tac-Tao

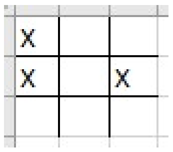
a) How do you represent the state?

State representation should embed all the information required to make a decision. Let’s have an example and consider Question 2’s graph (Cinderella). Right now, we have no limitation, so our state representation is just the letter indicating the node (A or B, …).

Now assume that the prince’s horse is too old so that it can visit only 3 houses (states). Will the state variable still be just the letter? No, because we now need to keep track of the number of nodes already visited. If Prince is in state F and he has already visited 3 houses, we cannot continue, and the search is over. But if it is in state F and he has only visited 2 houses so far, he can go to L. So, the state variable will be, for example, (n, S) where n is the number of already visited nodes and S is the letter indicating the house/node (Being in (2, F): love story, being in (3, F): sad story).

In this version of Tic-Tac-Tao, we need to keep track of the number of moves by the previous player. If the previous player was Max and now it is Min’s turn, she should know whether Max played 2 times, so she can also play 2 times or not.

Assume we have a state (node) with 3 X’s on the board. It is not enough as a state representation because



If Max played 1 round first, then Min, then Max played 2 rounds, now Min can play 2 rounds.

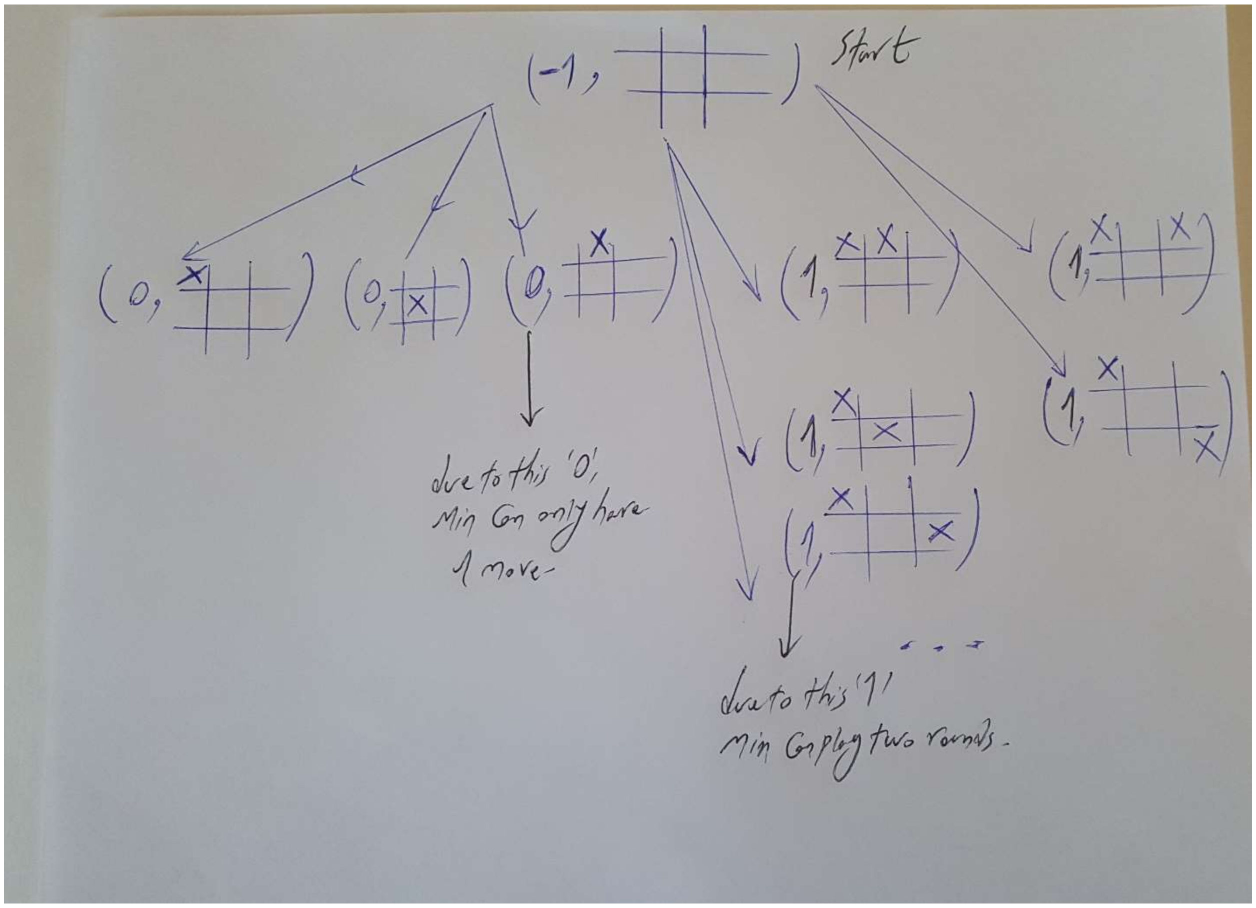
If Max played 2 rounds first, then Min, then Max played 1 round, now Min can play 1 round.

Therefore, the state representation should be a 2-tuple of (n, S):

n: [0,1] or [1,2] (and -1 for start), Whether we can play 1 round or 2 rounds (meaning whether previous player played 1 or 2 rounds)

S: The board with 9 cells and X’s and O’s as before.

b) Draw only the first round of the game (starting from an empty board, Max (X) moves.


Start
(-1,
(0,
(1,
due to this '0', Min Can only have 1 move.
due to this '1' Min Can play two rounds.

c) What is the branching factor?

Max can play 1 game or 2 games. If she plays 1 round, we have 9 possible moves. If she plays 2 rounds (2 X’s), we have $\binom{9}{2}$ = 36 possibilities. But there is no difference between putting a cross on cell 1and one on cell 2or one on cell 2 and one on cell 1. Therefore, we should divide it 36 by 2.

So, the branching factor for the first round is 18+9 = 27.