Probabilistic Reasoning
CS:440 Assignment 2

Christopher Naporlee - cmn134
Michael Nelli - mrn73
Timothy Walker - tpw32

April 2022

Question 1

a) $P(A, B, C, D, E)$

$$\begin{aligned} &= P(D|A, B)P(E|B, C)P(A)P(B)P(C) \\ &= (0.1)(0.3)(0.2)(0.5)(0.8) \\ &= 0.0024 \end{aligned}$$

b) $P(\neg A, \neg B, \neg C, \neg D, \neg E)$

$$\begin{aligned} &= P(\neg D|\neg A, \neg B)P(\neg E|\neg B, \neg C)P(\neg A)P(\neg B)P(\neg C) \\ &= (1 - 0.9)(1 - 0.2)(1 - 0.2)(1 - 0.5)(1 - 0.8) \\ &= (0.1)(0.8)(0.8)(0.5)(0.2) \\ &= 0.0064 \end{aligned}$$

c) $P(\neg A|B, C, D, E)$

$$\begin{aligned} &= P(\neg A, B, C, D, E) \\ &= P(\neg A)P(B)P(C)\sum_D P(D|A, B)\sum_E P(E|B, C) \end{aligned}$$

At this point we know that B and C are set to be true so we can easily derive the probability values for those. Also we can easily find E to be 0.3, because B, C are true. Finally, A is false and B is true thus D can only be 0.6.

$$= (0.8)(0.5)(0.8)(0.3)(0.6) = 0.0576$$

Question 2

a) P(Burglary | JohnCalls = True, MaryCalls = True)

$$
\begin{aligned}
P(B|j,m) &= \alpha P(B,j,m) \\
&= \alpha \sum_e \sum_a P(B,j,m,e,a) \\
&= \alpha \sum_e \sum_a P(B)P(e)P(A|B,e)P(j|a)P(m|a) \\
&= \alpha P(B) \sum_e P(e) \sum_a P(a|B,e)P(j|a)P(m|a) \\
&= \alpha \underbrace{P(B)}_{f_1(B)} \sum_e \underbrace{P(e)}_{f_2(E)} \sum_a \underbrace{P(a|B,e)}_{f_3(A,B,E)} \underbrace{P(j|a)}_{f_4(A)} \underbrace{P(m|a)}_{f_5(A)}
\end{aligned}
$$

$$f_6(B,E) = \sum_a f_3(A,B,E) \times f_4(A) \times f_5(A)$$

$$= (f_3(a,B,E) \times f_4(a) \times f_5(a)) + (f_3(\neg a,B,E) \times f_4(\neg a) \times f_5(\neg a))$$

$$= (\begin{bmatrix} 0.95 & 0.29 \\ 0.94 & 0.001 \end{bmatrix} \times (0.9) \times (0.7)) + (\begin{bmatrix} 0.05 & 0.71 \\ 0.06 & 0.999 \end{bmatrix} + (0.05) \times (0.01))$$

$$= \begin{bmatrix} 0.598525 & 0.183055 \\ 0.592235 & 0.0011295 \end{bmatrix}$$

Simplifying:

$$P(B|j,m) = \alpha f_1(B) \times \sum_e f_2(E) \times f_6(B,E)$$

$$f_7(B) = \sum_e f_2(E) \times f_6(B,E)$$

$$f_7(B) = f_2(e) \times f_6(B,e) + f_2(\neg e) \times f_6(B,\neg e)$$

$$= ((.002) \times \begin{bmatrix} 0.598582 \\ 0.183055 \end{bmatrix}) + ((.998) \times \begin{bmatrix} 0.592235 \\ 0.0011295 \end{bmatrix}) = \begin{bmatrix} 0.59224 \\ 0.00149 \end{bmatrix}$$

$$
\begin{aligned}
P(B|j,m) &= \alpha f_1(B) \times f_7(B) \\
&= \alpha \begin{bmatrix} 0.001 \\ 0.999 \end{bmatrix} \times \begin{bmatrix} 0.59224 \\ 0.00149 \end{bmatrix} \\
&\approx \begin{bmatrix} 0.284 \\ 0.716 \end{bmatrix}
\end{aligned}
$$

b) In our final equation we had $P(B|j,m) = \alpha f_1(B) \times f_7(B)$.

$$f_7(B) = f_2(e) \times f_6(B,e) + f_2(\neg e) \times f_6(B,\neg e)$$
$$f_6(B,E) = (f_3(a,B,E) \times f_4(a) \times f_5(a)) + (f_3(\neg a,B,E) \times f_4(\neg a) \times f_5(\neg a))$$

From $f_6(B,E)$ we got:

$$f_7(B,E) = (\begin{bmatrix} 0.95 & 0.29 \\ 0.94 & 0.001 \end{bmatrix} \times (0.9) \times (0.7)) + (\begin{bmatrix} 0.05 & 0.71 \\ 0.06 & 0.999 \end{bmatrix} + (0.05) \times (0.01))$$

Which ends up resulting in 10 multiplications and 4 additions. This is because these operations are **pointwise**.
Next, from $f_7(B)$ we got:

$$= ((.002) \times \begin{bmatrix} 0.598582 \\ 0.183055 \end{bmatrix}) + ((.998) \times \begin{bmatrix} 0.592235 \\ 0.0011295 \end{bmatrix})$$

which is another 4 multiplications and 2 additions. Finally our last step being:

$$= \alpha \begin{bmatrix} 0.001 \\ 0.999 \end{bmatrix} \times \begin{bmatrix} 0.59224 \\ 0.00149 \end{bmatrix}$$

To solve this part: 2 more pointwise multiplications. Followed by 1 addition and 1 division to solve α. Finally doing 1 division to normalize. Bringing us to a grand total of 16 multiplications, 7 additions, and 2 divisions or 25 total operations.

As for enumeration we must:

$$P(b|j,m) = \frac{P(b,j,m)}{P(b,j,m) + P(\neg b,j,m)}$$

Where

$$P(b|j,m) = \alpha P(b) \sum_e \sum_a P(e)P(a|b,e)P(j|a)P(m|a)$$

In enumeration, we must do 3 multiplications over 2 values for each variable we are iterating over (e,a). Thus, 3 * 4 = 12 multiplications with 3 additions as part of the summations. Finally, one final multiplication using P(b) to get the result. Leaving us with 13 multiplications and 3 additions.
However, we must calculate $P(\neg b,j,m)$ in order to get the alpha. Thus: we get 16 operations + another 16 operations for 32 operations. Then another addition and division to apply our normalization.
Leaving us with a grand total of 34 operations to solve enumeration.

Comparing: variable elimination does (34 - 25) = 9 less operations.

c) First solving $P(X_1|X_n = true)$ using enumeration. We can represent the equation like:

$$P(X_1|X_n) = P(X_1)\sum_{X_2}\sum_{X_3}...\sum_{X_{n-1}} P(X_2|X_1)P(X_3|X_2)...P(X_n|X_{n-1})$$

In words, we must account for all the hidden variables in the bayesian network. In this problem, there are $n-1$ hidden variables $(X_2...X_{n-1})$. Therefore, we must take into account 2^{n-1} entries. Thus, our total work to be done using enumeration will be $O(2^{n-1})$ or more simply $O(2^n)$.

Next solving $P(X_1|X_n = true)$ using variable elimination. We have:

$$P(X_1|X_n) = P(X_1)\sum_{X_2} P(X_2|X_1)\sum_{X_3} P(X_3|X_2)...\sum_{X_{n-1}} P(X_n|X_{n-1})$$

Variable elimination works by summing out the hidden variables that do not depend on the query:

$$P(X_1|X_n) = P(X_1)\sum_{X_2} P(X_2|X_1)\sum_{X_3} P(X_3|X_2)...\sum_{X_{n-2}} P(X_{n-3}|X_{n-2})f_1(X_{n-1})$$

$$= P(X_1)\sum_{X_2} P(X_2|X_1)\sum_{X_3} P(X_3|X_2)...\sum_{X_{n-3}} P(X_{n-4}|X_{n-3})f_2(X_{n-2})$$

$$...$$

Therefore, using this method, we are able to not repeat work already done unlike variable enumeration. Further, since this network is a singly linked chain we can reference one of it's properties. "The time and space complexity of exact inference in poly-trees is linear in the size of the network" (Bayesian Networks and Exact Inference Notes, Pg. 5).
Therefore, the amount of work done using this method will be $O(n)$.

Question 3

a) • $P(d|c)$

$$\begin{aligned} P(d|c) &= \alpha P(d,c) \\ &= \alpha \sum_a \sum_b P(a)P(b)P(c|a,b)P(d|b,c) \\ &= \alpha \sum_a P(a) \sum_b P(b)P(c|a,b)P(d|b,c) \\ &= \alpha < 0.3375, 0.1125 > \\ &=< 0.75, 0.25 > \end{aligned}$$

From our coded sample tests of $P(d|c)$ we got:
Rejection Sampling: $< 0.74413, 0.25586 >$ with % Error $\approx 0.80\%$
Weighted Sampling: $< 0.74606, 0.25393 >$ with % Error $\approx 0.53\%$

• $P(b|c)$

$$\begin{aligned} P(b|c) &= \alpha P(b,c) \\ &= \alpha \sum_a \sum_d P(a)P(b)P(c|a,b)P(d|b,c) \\ &= \alpha P(b) \sum_a P(a)P(c|a,b) \sum_d P(d|b,c) \\ &= \alpha < 0.45, 0 > \\ &=< 1, 0 > \end{aligned}$$

From our coded sample tests of $P(b|c)$ we got:
Rejection Sampling: $< 1, 0 >$ with % Error $= 0\%$
Weighted Sampling: $< 1, 0 >$ with % Error $= 0\%$

• $P(d|\neg a, b)$

$$\begin{aligned} P(d|\ a,b) &= \alpha P(d,\ a,b) \\ &= \alpha \sum_c P(a)P(b)P(c|a,b)P(d|b,c) \\ &= \alpha P(a)P(b) \sum_c P(c|a,b)P(d|b,c) \\ &= \alpha < 0.3825, 0.5175 > \\ &=< 0.425, 0.575 > \end{aligned}$$

From our coded sample tests of $P(d|\neg a, b)$ we got:
Rejection Sampling: $< 0.42200, 0.57799 >$ with % Error $\approx 0.71\%$
Weighted Sampling: $< 0.43604, 0.55295 >$ with % Error $\approx 2.50\%$

From our results we can see that our tests are very much similar to our expected probability.

b) Sampling Graphs on $P(d|c)$

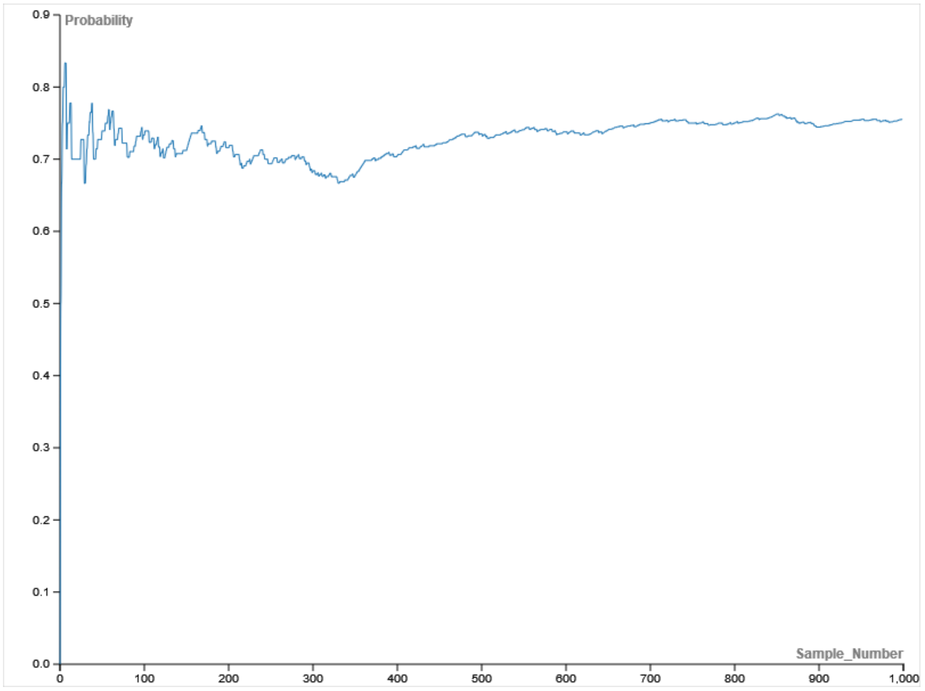

Figure 1: Rejection Sampling on $P(d|c)$

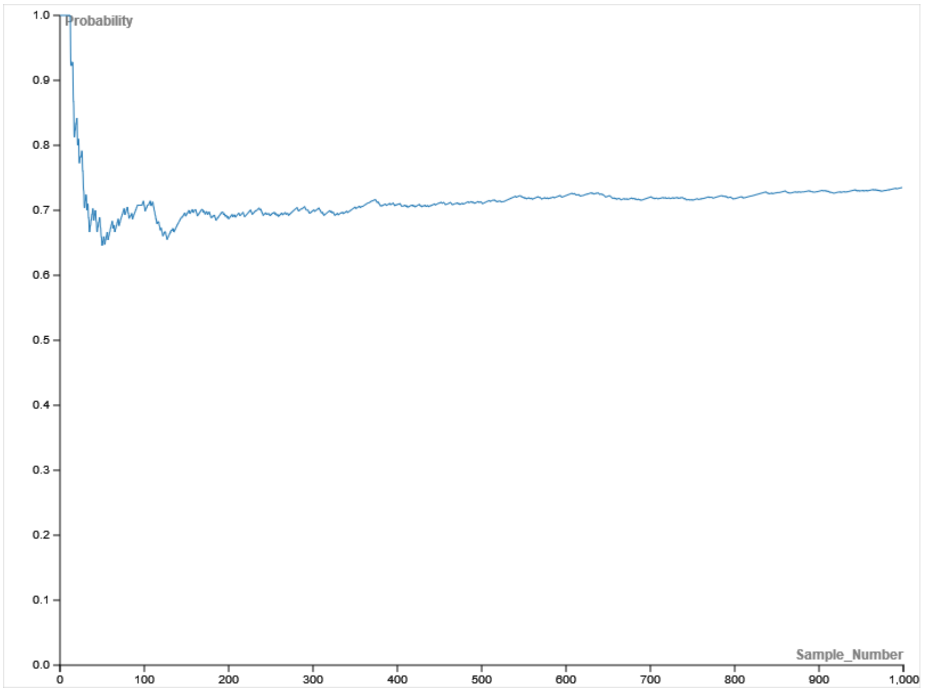

Figure 2: Weighted Sampling on $P(d|c)$

As you can see, the weighted sampling graph starts out not quite close to the expected probability, but levels out much quicker. As opposed to the rejection graph which starts closer to the expected probability, but fluctuates a lot more. This is because (if you look close enough) there are samples being rejected which cause it to take much longer to stabilize at our expected probability.

c) Our formulated query we used was: $P(d|\neg a, b, \neg c)$

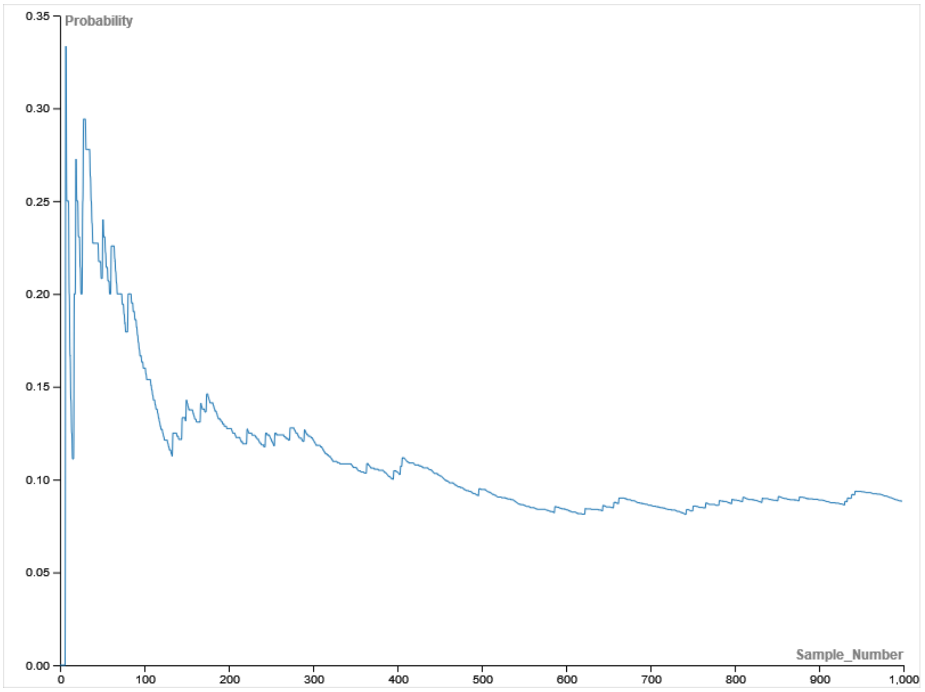

Figure 3: Rejection Sampling on $P(d|\neg a, b, \neg c)$

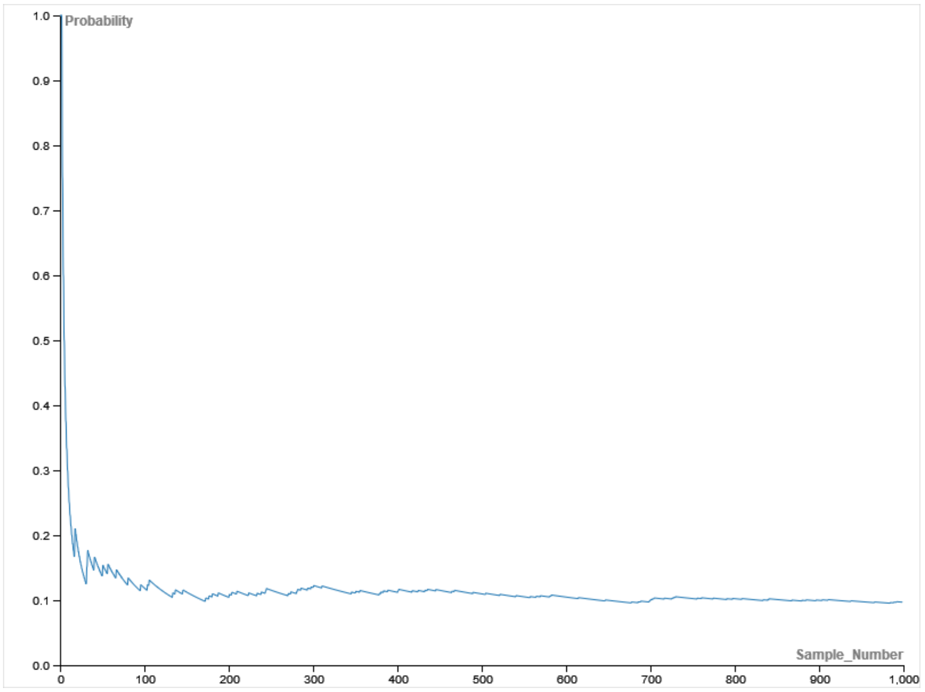

Figure 4: Weighted Sampling on $P(d|\neg a, b, \neg c)$

From these graphs, it is apparent that rejection sampling was notably worse. This is due to rejection samplings main pitfall - the fact that it will throw away too many samples. Given large amounts of evidence variables, as we have in our query of $\neg a, b, \neg c$, rejection sampling will throw out a large amount of samples as it struggles to find the correct atomic event to validate as a matching sample.

Knowing this, rejection sampling will need a much larger set of samples in order to reach the expected probability output. As opposed to weighting, that uses all the samples, it levels out much nicer and more quickly around 400 samples. For rejection sampling, it has still yet to level out even at 400 samples.

Question 4

a) Given: $X_1 = A$ Thus, $P(X_1|hot_1) = 1$

$$\begin{gathered} P(X_2|hot_1, cold_2) \\ = \alpha P(cold_2|X_2) \sum_{X_1} P(X_2|X_1)P(X_1|hot_1) \\ = \alpha < 0,1,1,0,1,1 > \times < 0.2, 0.8, 0, 0, 0, 0 > \\ = \alpha < 0, 0.8, 0, 0, 0, 0 > \\ =< 0, 1, 0, 0, 0, 0 > \end{gathered}$$

This answer makes sense, as if we were to start on A on day 1 and day 2 read cold we must have moved from A to B (a cold tile). Continuing...

$$\begin{gathered} P(X_3|hot_1, cold_2, cold_3) \\ = \alpha P(cold_3|X_3) \sum_{X_2} P(X_3|X_2)P(X_2|cold_2) \end{gathered}$$

First we will solve for the summation:

$$\begin{gathered} \sum_{X_2} P(X_3|X_2)P(X_2|cold_2) \\ = [(< 0.2, 0.8, 0, 0, 0, 0 > \times 0) + (< 0, 0.2, 0.8, 0, 0, 0 > \times 1) \\ +(< 0, 0, 0.2, 0.8, 0, 0 > \times 0) + ...] \\ =< 0, 0.2, 0.8, 0, 0, 0 > \end{gathered}$$

Bring our answer back into the original equation:

$$\begin{gathered} P(X_3|hot_1, cold_2, cold_3) \\ = \alpha P(cold_3|X_3) < 0, 0.2, 0.8, 0, 0, 0 > \\ = \alpha < 0,1,1,0,1,1 > \times < 0, 0.2, 0.8, 0, 0, 0 > \\ = \alpha < 0, 0.2, 0.8, 0, 0, 0 > \\ =< 0, 0.2, 0.8, 0, 0, 0 > \end{gathered}$$

Therefore, on the third day, we find that the probability of the rover being on B is 0.2, and the probability that it is on C is 0.8. Any other position has a probability of 0.

b) $P(X_2|hot_1, cold_2, cold_3)$

$$= \alpha P(X_2|hot_1, cold_2)P(cold_3|X_2)$$
$$= \alpha < 0, 1, 0, 0, 0, 0 > \times P(cold_3|X_2)$$

Solving for $P(cold_3|X_2)$:

$$= \sum_{X_3} P(cold_3|x_3)P(|x_3)P(x_3|X_2)$$
$$= (0 * 1 * < 0.2, 0, 0, 0, 0, 0 >) +$$
$$(1 * 1 * < 0.8, 0.2, 0, 0, 0, 0 >) +$$
$$(1 * 1 * < 0, 0.8, 0.2, 0, 0, 0, 0 >) +$$
$$(0 * 1 * < 0, 0, 0.8, 0.2, 0, 0, 0 >) +$$
$$(1 * 1 * < 0, 0, 0, 0.8, 0.2, 0 >) +$$
$$(1 * 1 * < 0, 0, 0, 0, 0.8, 0.2 >)$$
$$=< 0.8, 1, 0.2, 0.8, 1, 0.2 >$$

Back to the previous equation:

$$\alpha < 0, 1, 0, 0, 0, 0 > \times P(cold_3|X_2)$$
$$= \alpha < 0, 1, 0, 0, 0, 0 > \times < 0.8, 1, 0.2, 0.8, 1, 0.2 >$$
$$= \alpha < 0, 1, 0, 0, 0, 0 >$$
$$=< 0, 1, 0, 0, 0, 0 >$$

Thus,
$$P(X_2|hot_1, cold_2, cold_3) =< 0, 1, 0, 0, 0, 0 >$$

c) Finding MLE given $(E_1 = hot, E_2 = cold, E_3 = cold)$
Our first point of explanation is given because we know the rover fell into A on the first day:

$$V_1(X) = P(X_1) =< 1, 0, 0, 0, 0, 0 >$$

Next solving for day 2:

$$\begin{aligned} V_2(X) &= max_1[V_1(x_1)P(X_2|x_1)P(E_2 = cold|X_2)] \\ &= max[1 * 0.2 * 0, 1 * 0.8 * 1, 1 * 0 * 1, \ldots] \\ &= max[< 0, 0.8, 0, 0, 0, 0 >] \\ &= 0.8 \end{aligned}$$

That being that our most likely explanation for day 2 is that the rover went to B. Then, for day 3:

$$\begin{aligned} V_3(X) &= max_2[V_2(x_2)P(X_3|x_2)P(E_3 = cold|X_3)] \\ &= max[0, 0.8 * 0.2 * 1, 0.8 * 0.8 * 1, 0 \ldots] \\ &= max[< 0, 0.16, 0.64, 0, 0, 0 >] \\ &= 0.64 \end{aligned}$$

From this we can gather that the rover most likely went to C on day 3. Therefore, our most likely explanation is:
Day 1: A
Day 2: B
Day 3: C

d) $P(hot_4, hot_5, cold_6 | hot_1, cold_2, cold_3)$
Given from previous information on the distribution of the rover to day 3, we know the rover must be on either B or C. Knowing this we can create paths that the rover could have taken.

Starting Point Day 3	Path	Probability
B	BBB	$\alpha 0.008$
B	BBC	$\alpha 0.032$
B	BCC	$\alpha 0.032$
B	BCD	$\alpha 0.128$
B or C	CCC	$\alpha 0.008$
B or C	CCD	$\alpha 0.032$
B or C	CDD	$\alpha 0.032$
B or C	CDE	$\alpha 0.128$
C	DDD	$\alpha 0.008$
C	DDE	$\alpha 0.032$
C	DEE	$\alpha 0.032$
C	DEF	$\alpha 0.128$

Where $\alpha = 1.666$

From this table, we can see that the ideal path for $hot_4, hot_5, cold_6$ given we are cold on day 3 is DDE when starting on C.
Therefore,

$$P(hot_4, hot_5, cold_6 | hot_1, cold_2, cold_3) = \alpha 0.032 = .0533$$

e) Solve $P(X_4|hot_1, cold_2, cold_3)$ and $P(X_5|hot_1, cold_2, cold_3)$

$$\begin{gathered} P(X_4|hot_1, cold_2, cold_3) \\ = \sum_{X_3} P(X_3|hot_1, cold_2, cold_3)P(X_4|X_3) \end{gathered}$$

Solving for the summation first:

$$\begin{gathered} \sum_{X_3} P(X_3|hot_1, cold_2, cold_3) \\ = (0* < 0.2, 0.8, 0, 0, 0, 0 >) \\ +(0.2* < 0, 0.2, 0.8, 0, 0, 0 >) \\ +(0.8* < 0, 0, 0.2, 0.8, 0, 0 >) \\ +(0* < 0, 0, 0, 0.2, 0.8, 0 >) \\ +(0* < 0, 0, 0, 0, 0.2, 0.8 >) \\ +(0* < 0, 0, 0, 0, 0, 0.2 >) \\ =< 0, 0.04, 0.32, 0.64, 0, 0 > \end{gathered}$$

Thus,

$$P(X_4|hot_1, cold_2, cold_3) =< 0, 0.04, 0.32, 0.64, 0, 0 >$$

Next, solving for $P(X_5|hot_1, cold_2, cold_3)$

$$\begin{gathered} = \sum_{X_4} P(X_4|hot_1, cold_2, cold_3)P(X_5|X_4) \\ = (0* < 0.2, 0.8, 0, 0, 0, 0 >) \\ +(0.04* < 0, 0.2, 0.8, 0, 0, 0 >) \\ +(0.32* < 0, 0, 0.2, 0.8, 0, 0 >) \\ +(0.64* < 0, 0, 0, 0.2, 0.8, 0 >) \\ +(0* < 0, 0, 0, 0, 0.2, 0.8 >) \\ +(0* < 0, 0, 0, 0, 0, 0.2 >) \\ =< 0, 0.008, 0.096, 0.384, 0.512, 0 > \end{gathered}$$

Finally coming to:

$$P(X_5|hot_1, cold_2, cold_3) =< 0, 0.008, 0.096, 0.384, 0.512, 0 >$$

Question 5

a) Finding $P(X_t|e,\alpha)$ where α denotes our action. Each matrix denotes the probability that we are at that position for each step. For full work, please check the included documents labeled "question5_partA_work" under the folder "q5".

- $P(X_1|N,R)$

$$= P(N|X_1)\sum_{X_0} P(X_1|X_0)P(X0|)$$
$$= \begin{bmatrix} 0.00131 & 0.01315 & 0.025 \\ 0.02368 & 0.2368 & 0.45 \\ 0.2368 & & 0.013 \end{bmatrix}$$

- $P(X_2|N,N,R)$

$$= P(N|X_2)\sum_{X_1} P(X_2|X_1)P(X_1|N)$$
$$= \begin{bmatrix} 7.66 * 10^{-6} & 1.46 * 10^{-4} & 0.0021 \\ 0.0025 & 0.0474 & 0.698 \\ 0.249 & & 7.6 * 10^{-4} \end{bmatrix}$$

- $P(X_3|N,N,H,D)$

$$= P(H|X_3)\sum_{X_2} P(X_3|X_2)P(X_2|N,N)$$
$$= \begin{bmatrix} 1.78 * 10^{-6} & 2.247 * 10^{-5} & 1.796 * 10^{-5} \\ 2.198 * 10^{-5} & 0.00406 & 0.00613 \\ 0.02148 & & 0.9682 \end{bmatrix}$$

- $P(X_4|N,N,H,H,D)$

$$= P(H|X_4)\sum X_3 P(X_4|X_3)P(X_3|N,N,H)$$
$$= \begin{bmatrix} 1.825 * 10^{-7} & 2.304 * 10^{-6} & 1.023 * 10^{-7} \\ 2.165 * 10^{-7} & 2.324 * 10^{-4} & 3.583 * 10^{-5} \\ 0.00122 & & 0.9985 \end{bmatrix}$$

Given this information, we can conclude that we will have ended up in the bottom right of the map (3,3) with a probability of 0.9985.

b) For example readings generated, please refer to the included python scripts and the files under the folder "q5" and the generated results under "results" within "q5".

c) **Filtering Visualizations**
For the visualizations of the heat map, we created a program for you to interactively use. For relevant documentation read the README under 'q5' folder. As for the pictures, we have attached heatmaps for iterations 10, 50, and 100 along with the perceived action/sensors and the true actions. As for the meaning of the heat map:

- Grey cells represent a blocked cell with 0 probability.
- White cells represent a very low probability.
- Blue cells represent low/medium probability.
- Light red cells represent medium/high probability.
- Dark red cells represent an extremely high probability.

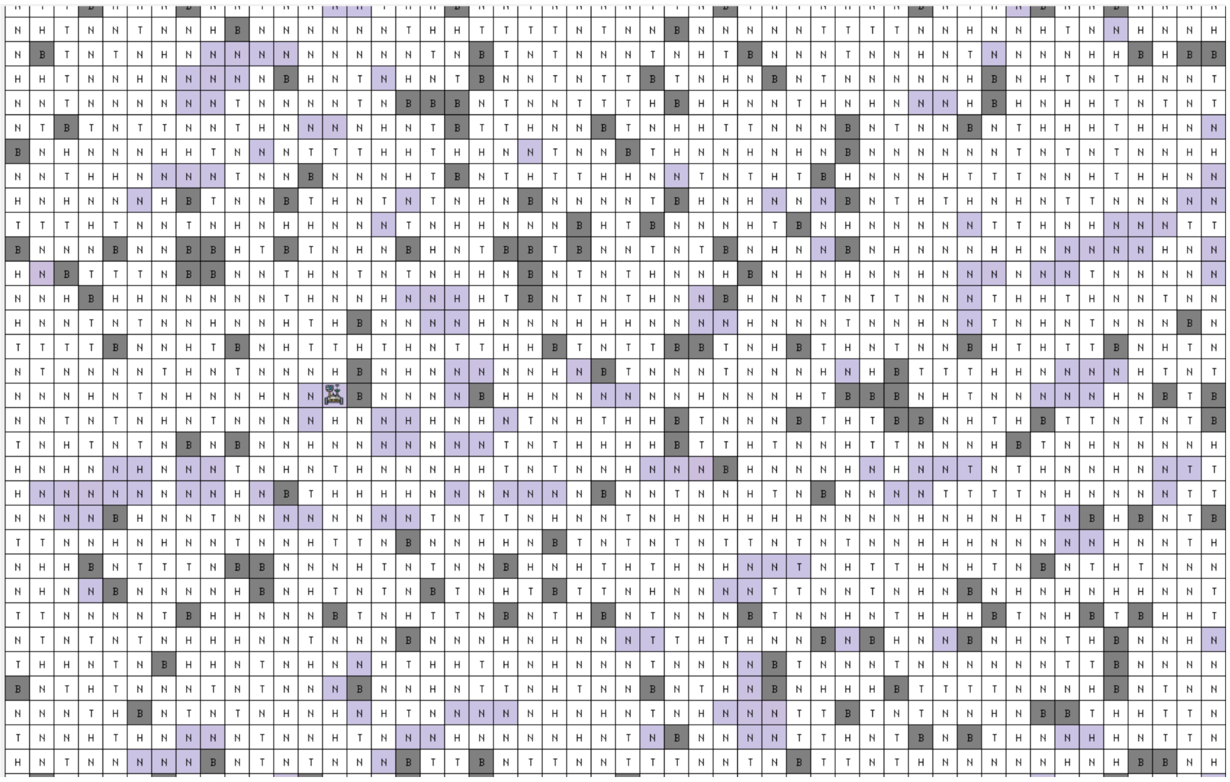

Figure 5: Heatmap after 10 iterations.
Actions: LLUDUDRURR
Sensors: NNHNHNNNNN
True Movement: LLUDUDRURR

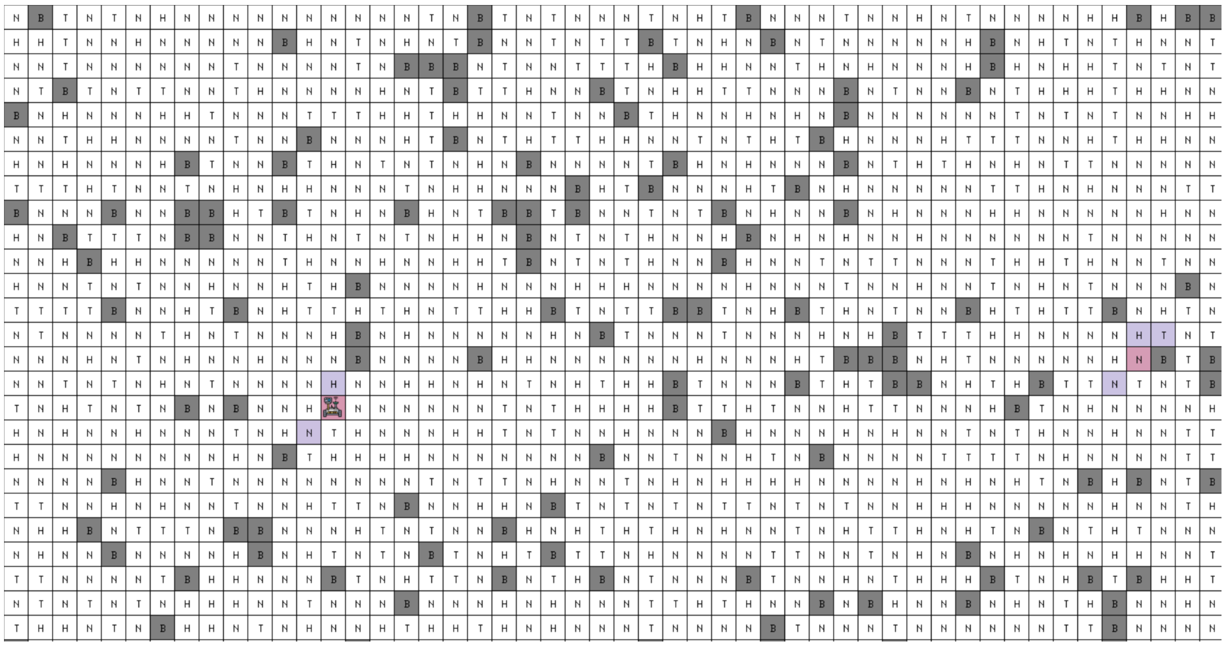

Figure 6: Heatmap after 50 iterations
Reported Actions:
LLUDUDRURRLDLRLLRDDLLLRLLRDDURDDDUUULRRUUDDRUUDRUD
Reported Sensors:
NNHNHNNNNNNNNNNNNNHNHNTNTNHTNNNHNNNHNTNHNNHTNNHHNHN
True Movement (Dashes denote moments the rover stays still):
LLUDUDRURRLDLRL-RDDLLLRLLRDD–DDDUUULRR-UDDR-U-RUD

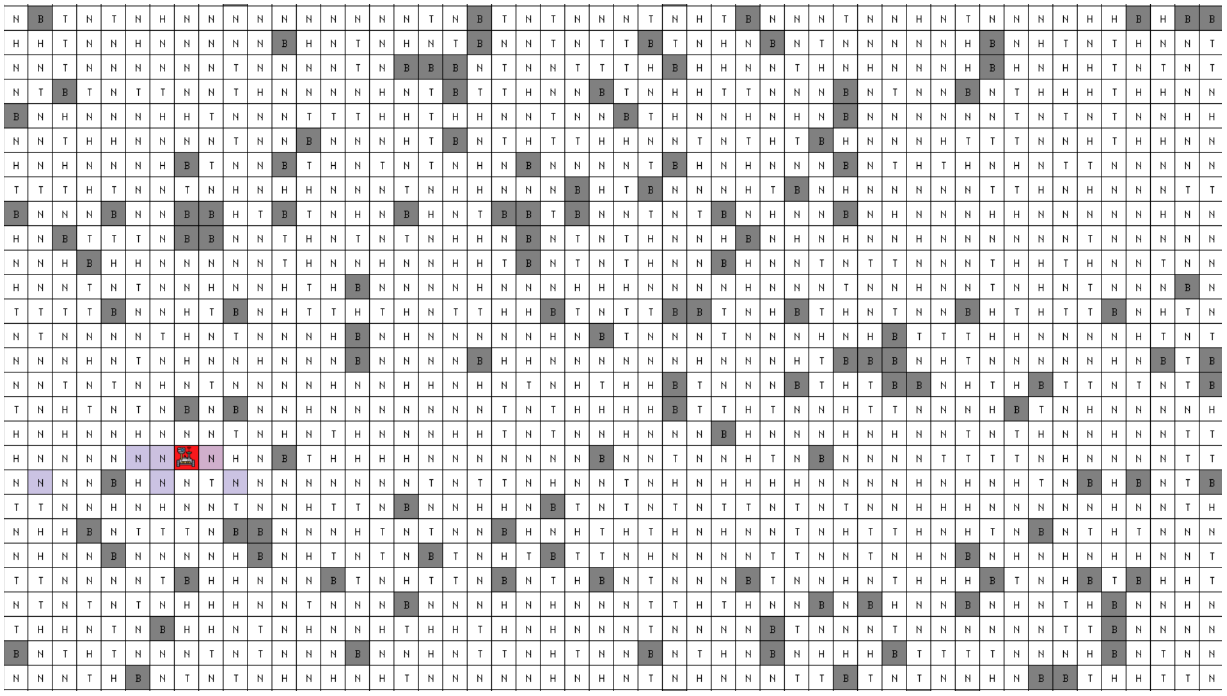

Figure 7: Heatmap after 100 iterations

Reported Actions:
LLUDUDRURRLDLRLLRDDLLLRLLRDDURDDDUUULRRUUDDRUUDRUD
DDDUDLDLLLURDDRLULDULLLDULUDRRDLLDLRLLLLRLRURRRLL

Reported Sensors:
NNHNHNNNNNNNNNNNNNHNHNTNTNHTNNNHNNNHNTNHNNHTNNHHNHN
THHTHTTNNNHNNNNNNHNHHNNHNNTNNNNNNNNHNNHHNNHNNTNHNN

True Movement (Dashes denote moments the rover stays still):
LLUDUDRURRLDLRL-RDDLLLRLLRDD–DDDUUULRR-UDDR-U-RUD
D-D-UDLDLLLURD-RLULDU-LLDULU-RRDLLDLR-LLLRLRURRRLL

As you can see, by the 100th iteration, we are dark red on the cell that the agent is on. This deep red color denotes a probability of almost 1 meaning that our filtering algorithm is just about certain that it is on that square.

Plotting Error

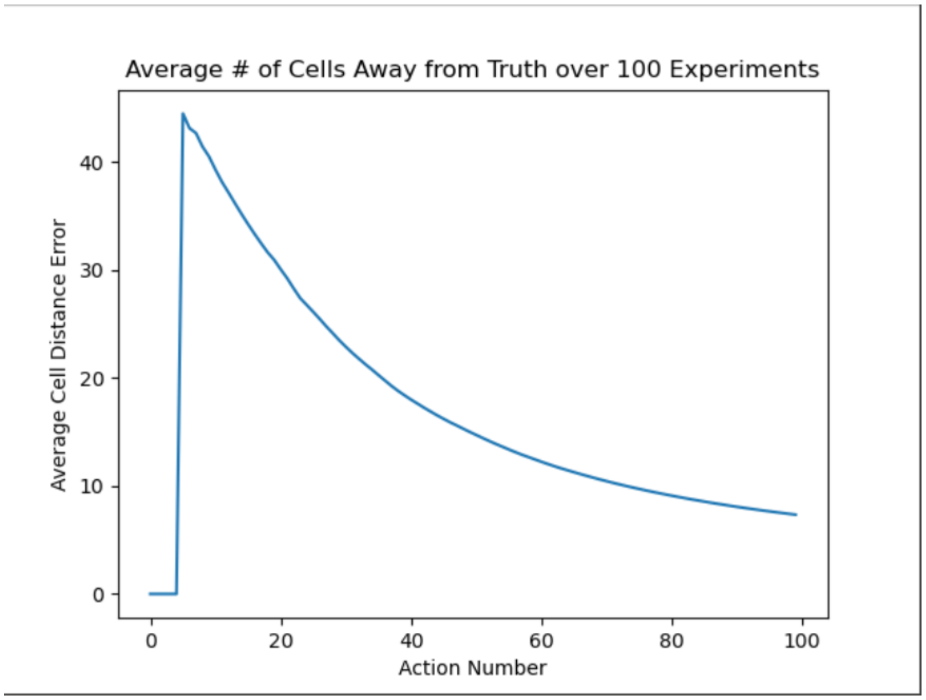

Figure 8: Average number of cells off from true agent location for 100 actions over 100 experiments with first 5 actions skipped.

For plotting the error we found the number of cells away the max probability was from the true location of the agent for each action across each experiment. We took the average at each iteration across 100 experiments and got this plot. As you can see, as the number of actions increases the filtering algorithm gets closer and closer, on average, to the true location of the agent.

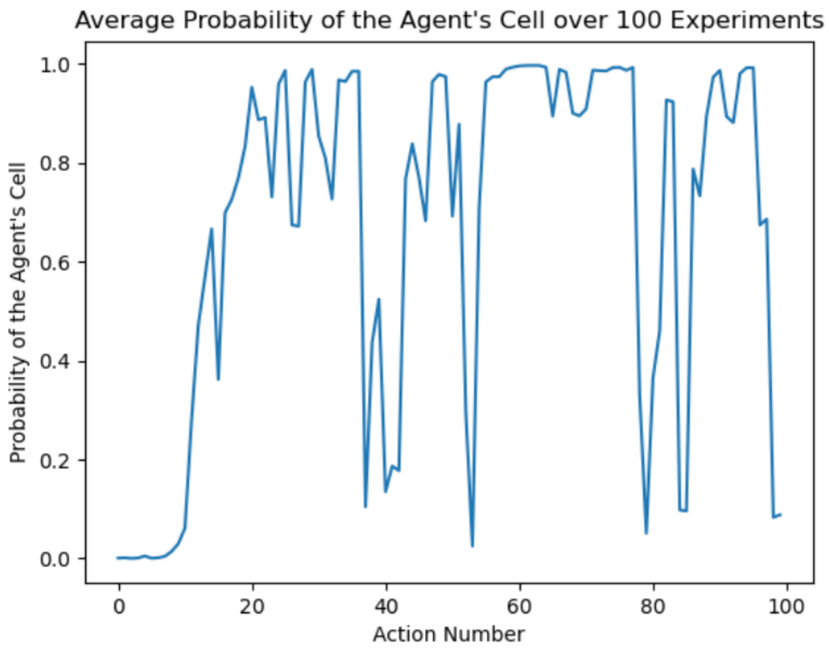

Figure 9: Average Probability of Agent's Cell over 100 experiments

For plotting the probability of the agent's cell, we simply kept track of where he was in each experiment. Then, for each action we average the probability over 100 experiments. From the resulting graph we can see that for most of the time the filtering algorithm has a pretty good guess of where the agent truly is. However, we can also notice a lot of dips that indicate that the experiments have overlapping areas in which the agent reports one thing, but does another. Thus tricking the filtering algorithm into thinking the agent is somewhere else.

Question 6

a) Expected net gain from buying C_1:

- \$4000 market value if in good shape
- \$3000 to buy
- \$1400 in repairs to make it in good shape (if not already)
- \$100 for mechanic test
- If in (70% chance) good shape : (4000 - 3000) = \$1000 net profit
- If in (30% chance) bad shape: (4000 - (3000 + 1400)) = -\$400 net profit

Net gain:

$$(0.7 * 1000) + (0.3 * -400) = \$580$$

b)

$$\begin{aligned} P(pass(C_1)) &= \sum_q P(pass(C_1)|q) \\ &= P(pass(C_1)|q^+)P(q^+) + P(pass(C_1)|q^-)P(q^-) \\ &= (0.8 * 0.7) + (0.35 * 0.3) \\ &= 0.665 \end{aligned}$$

$$P(\neg Pass(C_1)) = 1 - P(pass(C_1)) = 1 - 0.665 = 0.335$$

Using this information we can find:

$$P(q^+|Pass) = \frac{P(Pass|q^+)P(q^+)}{P(Pass)} \approx 0.842$$

$$P(q^+|\neg Pass) = \frac{P(\neg Pass|q^+)P(q^+)}{P(\neg Pass)} \approx 0.418$$

$$P(q^-|Pass) = 1 - P(q^+|Pass) = 1 - 0.842 = 0.158$$

$$P(q^-|\neg Pass) = 1 - P(q^+|\neg Pass) = 1 - 0.418 = 0.582$$

c) Using:

$$EU(A|E) = \sum_i P(Result_i(A)|Do(A), E)U(Result_i(A))$$

We can find:

$$\begin{aligned} EU(buy|Pass) &= \sum_i P(Result_i(buy)|Pass)U(Result_i(buy)) \\ &= P(q^+|Pass)U(q^+|buy) + P(q^-, Pass)U(q^-|buy) \\ &= (0.842 * 1000) + (0.158 * -400) = 778.8 \end{aligned}$$

$$\begin{aligned} EU(buy|\neg Pass) &= \sum_i P(Result_i(buy)|\neg Pass)U(Result_i(buy)) \\ &= P(q^+|\neg Pass)U(q^+|buy) + P(q^-, \neg Pass)U(q^-|buy) \\ &= (0.418 * 1000) + (0.582 * -400) = 185.2 \end{aligned}$$

$$\begin{aligned} EU(\neg buy|Pass) &= \sum_i P(Result_i(\neg buy)|Pass)U(Result_i(\neg buy)) \\ &= P(q^+|Pass)U(q^+|\neg buy) + P(q^-, Pass)U(q^-|\neg buy) \\ &= (0.842 * 0) + (0.158 * 0) = 0 \end{aligned}$$

$$\begin{aligned} EU(\neg buy|\neg Pass) &= \sum_i P(Result_i(\neg buy)|\neg Pass)U(Result_i(\neg buy)) \\ &= P(q^+|\neg Pass)U(q^+|\neg buy) + P(q^-, \neg Pass)U(q^-|\neg buy) \\ &= (0.418 * 0) + (0.582 * 0) = 0 \end{aligned}$$

Therefore, the best decision given a pass is to buy, and the best decision given a fail is to still buy.

d) Let the mechanic test be denoted by T_j.
The value of perfect information is given by:

$$VPI(T_j) = (\sum_{t_j} P(T_j = t_j) EU(\alpha_{t_j} | T_j = t_j)) - EU(\alpha)$$
$$= [P(T_j = Pass) EU(\alpha_{Pass} | Pass) + P(T_j = \neg Pass) EU(\alpha_{\neg Pass} | \neg Pass)] - EU(\alpha)$$
$$= [P(T_j = Pass) EU(buy | Pass) + P(T_j = \neg Pass) EU(buy | \neg Pass)] - EU(\alpha)$$
$$= ((0.665 * 778.8) + (0.335 * 185.2)) - 580$$
$$\approx -0.06$$

Therefore, you should not take C_1 to the mechanic as you will end up losing more money from it costing \$100.

Question 7

For the following sections of information, it was all acquired from the code under the folder "q7".

- **Original Utilities**
  For our original utilities, we start with an array of 0's. As value iteration runs, it will update this array. To understand this decision further refer to the algorithms written under the **implementation** section.

- **Intermediate Results**

Iteration	Optimal Utilities U	Policy π
1	$[0, 0, 0, 0]$	$[1, 2, 3, 1]$
2	$[0, 0, 1.0, 0]$	$[1, 2, 3, 1]$
3	$[0, 0.72, 1.0, 0.81]$	$[2, 2, 3, 1]$
20	$[3.2929, 3.7708, 4.3930, 3.8304]$	$[2, 2, 3, 1]$
40	$[3.8516, 4.3366, 4.9499, 4.3972]$	$[2, 2, 3, 1]$
60	$[3.9199, 4.4050, 5.0181, 4.4656]$	$[2, 2, 3, 1]$
80	$[3.9282, 4.4133, 5.0264, 4.4739]$	$[2, 2, 3, 1]$
100	$[3.9292, 4.4143, 5.0274, 4.4749]$	$[2, 2, 3, 1]$
120	$[3.9293, 4.4144, 5.0276, 4.4751]$	$[2, 2, 3, 1]$

Note: For any s_i the utility is $U[i-1]$. Thus, the utility of s_1 is $U[1-1] = U[0]$. Similarly, for any s_i the policy is $\pi[i-1]$. Thus, for the policy of s_3 the action to take is $\pi[3-1] = \pi[2] = 3 = a_3$ (for the 120^{th} iteration).

- **Implementation**
  To calculate the optimal utilities and the optimal policy, we used the following algorithms and converted them to python code. We set our convergence criterion to be within 5 decimal places or 0.00001. We also set our discount factor, γ, to be 0.9.

Algorithm 1 Finding Optimal Utilities

```
**INPUTS:** Set of states $S$, transitional model $P(s'|s,a)$, discount factor $\gamma$, and max error allowed in the utility of any state $\epsilon$
**OUTPUT:** The optimal utilities $U$

Initialize $U'$ to be array of zeros with length = $|S|$
while True do
    $U \leftarrow U'$
    $\delta \leftarrow 0$
    for $s \in S$ do
        $U'[s] \leftarrow r(s) + \gamma max_a \sum_{s'} P(s'|s,a)U[s']$
        $\delta \leftarrow max(\delta, |U'[s] - U[s]|)$
    end for
    if $\delta \leq \frac{\epsilon(1-\gamma)}{\gamma}$ then
        break
    end if
end while
return $U$
```

Algorithm 2 Finding Optimal Policy

```
**INPUTS:** Set of states $S$, transitional model $P(s'|s,a)$, and the optimal utilities of each state $U$
**OUTPUT:** The optimal policy $\pi^*$

Initialize $\pi^*$ to be array of actions with length = $|S|$
for $s \in S$ do
    $\pi^*(s) \leftarrow argmax_a \sum_{s'} P(s'|s,a)U[s']$
end for
return $\pi^*$
```

These algorithms gave us a computation time: 0.113s over 124 iterations and our final result to be:
$V_{opt} \approx [3.92937, 4.41448, 5.02761, 4.47512]$ and $\pi^* = [2, 2, 3, 1]$
Thus, $\pi^*(s_1) = a_2, \pi^*(s_2) = a_2, \pi^*(s_3) = a_3$, and $\pi^*(s_4) = a_1$

Note: For any s_i the utility is $U[i-1]$. Thus, the utility of s_1 is $U[1-1] = U[0]$. Similarly, for any s_i the policy is $\pi[i-1]$. Thus, for the policy of s_4 the action to take is $\pi[4-1] = \pi[3] = 1 = a_1$.