**RESEARCH NOTEBOOK:**

Q-Learning and Deep Q-Learning for Discrete Solution Space Task Optimization

**Dates conducted:** Nov 2023 - (ongoing)

**Name:** Joseph Parayil

**Email address:** josephaparayil@gmail.com

**EXPERIMENTAL PROPOSAL**

**Title:** Comparing the performance of Q-learning and Deep-Q learning on machine learning applications with limited solution spaces.

**Hypothesis:** If the Q-learning and Deep Q-learning models were implemented and compared in a Python environment under same hyperparameter constraints to solve the FrozenLake-v1 environment, the Q-learning model would prove to both have a faster rate of learning and a higher threshold of performance, because Q-learning models are more able to navigate discrete simple state environments, while Deep-q learning models would only be beneficial for non-discrete environments.

**New Procedure:**

1. Find the where the model converges (at what timestep and what win rate)
2. Perform binary search to find exactly at which timestep does the model train the agent to have a win rate closest to 10%, from range 0 to the timestep at which the model converges.
3. In doing binary search, instead of indexing list, run a test for the given timestep value. Each such test will be 100 trials. Each trial is training an agent for the given number of timesteps, and finding the performance afterward. After 100 trials, take the average of the performances and use that as the value.
4. Record the result of each test even if it is not the end result of binary search. That way, in binary searches for the other y-values, the program can refer to values already found from previous searches.
5. Repeat binary search for the other y-values (20%, 30%, etc.). Do not perform binary search for any values that are higherwhat the model converges at.
6. For any values that are higher than what the model converges at, the x-value is undefined, but can be treated as being infinite.
7. There should now be the set of y-values at fixed intervals of 10% and their corresponding number of timesteps.
8. For each pair, for the x-values, convert timestep value to the amount of computing time. Run tests of 100 trials each to find the average time it takes for the computer to train the model for that many timesteps. So now the x-value is computing time in seconds instead of iterational time
9. For each pair, obtain the slope by dividing the y-value over the x-values.
10. There are now 10 values denoting slopes. Find the average of all 10 values. This value is a metric representing the overal performance of the model on the environment
11. Do steps 1-10 process for both QL and DQN on the environment
12. Do steps 1-11 for all of Gymnasium’s toy-text environments: FrozenLake 4x4, Cliff Walking, FrozenLake 8x8, Blackjack, and Taxi.

**Old Procedure:**

1. Implement Q-learning (QL) and Deep Q-learning (DQN) machine learning agents and interface them with the FrozenLake-v1 training environment from the Gymnasium library, in Google Colab.
2. Set controls as such:
   * Timestep tests: **100,500,1000,2000,5000,10000,20000,50000,100000,200000,500000**
   * Amount of trials for each testValue: **25**
   * Amount of trials for performance check: **25**
3. Train the using the **QL algorithm** on the FrozenLake-v1 training environment forthe timestep test value number of iterations..
4. After training for the set amount of timesteps, record the performance of the agent on this iteration by running **25** test runs.
5. Repeat steps 3-4 for **25** trials
6. Repeat steps 3-5 **11** times total, for each of the timestep tests (at the end of this step, everything up to 25 trials of 500,000 iteration tests will have been completed)
7. Record all the data in json format, from all the trials of all the tests. The JSON file can be automatically stored in google drive. In addition, record the time it took to run the code
8. Repeat steps 2-7 but for the **DQN** algorithm
9. Perform the Data Analysis, using the two saved JSON files

**JOURNAL:**

|  |  |
| --- | --- |
| **Date** | **Description** |
| **11/29 WED** | I just now created this research journal. There is a lot of history and things I have done for this science fair project so far but I have not recorded it in a journal. I hope to use today’s entry to recap everything that has happened so far.  Last year, my science fair project was “Proposing an ‘unlayered’ neural network model for machine learning”, where I formulated my own entirely new type of neural network model and wrote code for it in an engineering process to validate that a code implementation is possible.  This year, I was planning to make a continuation project of this, where I would take the code for this theoretical model and test it on an experimental machine learning application and compare it with existing conventional models to test for effectiveness.  However I ended up changing this project idea this year because this type of work takes multiple years of continuous work and many more years of prior experience as a machine learning researcher. Since I only had around 4 months and am only a high school junior, after discussion with several professors and graduate students from UCF, I decided that continuing along this path would not work and instead pursued a different type of project.  From August to October, my main objective was finding a mentor well-knowledgeable in machine learning to provide guidance in the field.. Dr. Urbina, my teacher who managed our school’s science fair program, told me that he knew a professor from UCF named Dr. Lobo, who worked in machine learning in the Computer Vision department. Since he was in machine learning and Dr. Urbina already knew him well, he seemed like a good person to try and contact. If he himself were not available for advice, he could help me to contact other professors within UCF that he knows of.  After some technical difficulties, we established contact through phone and email, and clarified more things about my project and its place in machine learning. (my planned project at this time was about the continuation of last year’s). And such progress was made in understanding more. At one point, in early October I physically went to UCF campus to meet Dr. Lobo and talk with him and two of his graduate students.  Eventually through this process of searching for mentors, I found one professor named Dr. Sean Mondesire who I got as a mentor. His scope in research was more aligned to my project idea and had time in his schedule for mentorship.  I discussed with him more about the project through weekly zoom meetings and I decided to pursue a project on Q-learning vs. Deep q-learning. He recommended videos to read up on machine learning topics and libraries and platforms to use. I clarified the project into a specific project plan, learned the necessary background knowledge, implemented the Q-learning algorithm, connected with the training environment, and more. Recently I have been running tests to determine certain hyperparameters to use for the experiment and have made the research plan.  From now on I will report on the science fair progress as it goes. |
| **12/4 MON** | It has been a hectic week so far so I have not been making that much progress since the last entry. I have been reading up on Deep q learning networks and learning how they are and how they fit into the broader context of reinforcement learning. |
| **12/6 WED** | Training the Q-learning model on Frozen lake 4x4 has been successful thus far, but it seems to be failing to learn the 8x8 environment, with up to even 100,000 time steps yielding the result of 0. Dr. Mondesire has advised me to go even further with the time steps, going up to *10 million*. This amount of computation will take a considerable amount of time to run. |
| **12/11**  **MON** | So far I have been working on implementing Deep q learning for the FrozenLake environment. Stable baselines library has an implementation of DQNs that are easy to code, but I am having difficulty implementing it with FrozenLake. I believe it’s something to do with that FrozenLake’s reset function returns a single number but the DQN expects a tuple, which is causing an issue in the function backend. I will have to ask Dr. Mondesire about this.  I am also working on a rudimentary poster plan which is due for my science fair class this Friday. |
| **1/1**  **MON** | Happy New Years.  As part of my new years resolution, this week itself I will start and finish the experiment itself.  Today I am settling all my forms. Then I will begin on experimentation.  I have already done a little bit of coding for the experiment. It seems straightforward to code but there will be plenty of issues, and I will have to thoroughly check and inspect for errors to verify scientific accuracy. |
| **1/2**  **TUE** | Finished up code for the QL learning experimentation, thoroughly checked it, and refactored it. Started running the code. It seems fit to actually run the experiment. |
| **1/5**  **FRI** | Had some doubts about the validity of the experiment. Namely, that, the epsilon value of the Q learning algorithm will decrease at a rate which depends on the total number of episodes. So what Dr. Mondesire recommended is what I can do instead is run multiple tests of different number of episodes and use the end result of each as result values. This kind of changes the procedure and changes the data analysis steps so I will have to change that. It also means, that instead of a line graph of data, I will have a bar graph.  I have made changes into the QL code and as I am writing this, the experiment is running. It has been 1 hour 50 minutes of run time, will take a bit more. I estimate it will get done by 2 hours 30 minutes at most.  In the meantime I will make changes to the research plan and make changes to the DQN experiment code.  I plan to run the DQN experiment today also. |
| **1/6**  **SAT**  **& 1/7 SUN** | Yesterday I ran the QL code and got a json file output of data. It took 2h 28m 17s to run. I ran the DQN experiment but my computer went to sleep due to inactivity.  Today I ran the DQN again, this time with settings preventing computer sleep, and it took over 6 hours 30 minutes to run the entire experiment. When I looked at the json file, I found that all the test results were all 0s. Something was wrong in the code, and I just wasted 6 hours 30 minutes of run time just for that.  I analyzed the code vigorously and eventually found that one line in the performance checking code was in the wrong indent space. I fixed that, and everything started to work.  At 7:00 PM I started the experiment up again so that it completes overnight. I estimated that it would be completed by 1 am.  The next day I found out, however, around 11:00 PM it mysteriously stopped running. I believe it has to due with the fact that I am using the free version of Google Colab which does not have priority to paid users of Google Colab.  That day I ran the experiment again, starting at noon, and it completed by 5:41 PM with 5 hours and 41 minutes of run time, and good JSON output.  Now I have to do the Data Analysis. I am planning to use a bar graph to represent the data. Dr. Mondesire recommended the Seaborn library so I will be learning how to use it and developing code for displaying my data into my desired graph.  There is a hard deadline on Wednesday 5pm. By then, my forms, including my abstract, has to be submitted for review. And implicitly the project has to be done by then. So Monday I am planning to get my data analysis all done. And on Tuesday all my forms including my Abstract I have to do/clean up. That Tuesday evening I will have a meeting with Dr. Mondesire to discuss everything about my project and if I should make any last-minute changes, and on Wednesday I will submit anything. |
| **1/8 MON** | Read up on the Seaborn library, and attempted implementing my data in the form of a bar graph. It is a messy process of experimenting and learning about usage of the Seaborn library and related libraries.  I was planning to do the rest of the Data analysis today, but I was feeling sick due to some Stomach flu towards the evening. After vomiting I recovered overnight though.  Here is the current state of the bar graph after today:    100 timesteps: Both at 0% win rate  500 timesteps: QL: 28% DQN: 0%  1000 timesteps: QL: 76%; DQN: 0%  2000 timesteps: QL: 96%; DQN: 0%  5000 timesteps: QL: 100%; DQN: 0.64%  10000 timesteps: QL: 100%; DQN: 0.96%  20000 timesteps: QL: 100%; DQN: 0%  50000 timesteps: QL: 100%; DQN: 0.32%  100000 timesteps: QL: 100%; DQN: 88.96%  200000 timesteps: QL: 100%; DQN: 93.6%  500000 timesteps: QL: 100%; DQN: 93.92%  It’s very clear that QL massively outperformed DQN, far exceeding my expectations actually. |
| **1/12 FRI** | On Wednesday was the deadline for the scps forms. I have submitted my forms including my abstract to that. I have made a last minute change from the category of IMEC to MACO, since the student Kevin Perez decided to change from MACO to ENEV. I have made the necessary changes to my Abstract and Research Plan. I am planning to do even more statistical tests and add a lot of information on the poster. Although one would naturally think of this project as an IMEC one, these QL and DQN models ultimately boil down to math, and navigating the mathematical relationships surrounding the science of decision making.  The first statistical test I have done are T-tests for certain categories.  I used the online statistical calculator [www.socscistatistics.com/tests/studentttest/default2.aspx](http://www.socscistatistics.com/tests/studentttest/default2.aspx)  Here are the T-tests so far:  alpha value = 0.05  QL vs DQN at 100:  t-value: 0  p-value: 1 (not sig)  QL vs DQN at 500:  t-value: 3.05505  p-value: 0.003667 (sig)  QL vs DQN at 1000:  t-value: 8.7178  p-value: < 0.00001 (sig)  QL vs DQN at 2000:  t-value: 24  p-value: < 0.00001 (sig)  QL vs DQN at 5000:  t-value: 155.25  p-value: < 0.00001 (sig)  QL vs DQN at 10000:  t-value: 103.16667  p-value: < 0.00001 (sig)  QL vs DQN at 20000: [here both standard deviations were 0]  t-value: N/A  p-value: N/A (sig)  QL vs DQN at 50000:  t-value: 311.5  p-value: < 0.00001 (sig)  QL vs DQN at 100000:  t-value: 2.41981  p-value: 0.019365 (sig)  QL vs DQN at 200000:  t-value: 8  p-value: < 0.00001 (sig)  QL vs DQN at 500000:  t-value: 6.55725  p-value:< 0.00001 (sig)  QL at 2,000 vs QL at 5,000: The t-value is -1. The p-value is .322325. The result is not significant at p < .05. |
| **2/16 FRI** | The regional science fair was on the 5th. I am accepted in to the State science fair.  The state science fair will be April 2nd-4th  Since there is a lot of time until then, I will be making various extensions to my project.  I have three “levels” of extensions I want to add.  **Level 1:** Run tests for 8x8 FrozenLake environment (will def do)  **Level 2:** Run tests for all environments of every state size in Toy Text (time consuming)  **Level 3:** Use regression on the data points to make performance graphs (difficult)  After this I will have to update my digital poster, then my physical poster.  Then I will have to re-practice my verbal presentations  I have around 5 weeks of work available on this. On Spring break I will not be available. |
| **2/24 SAT** | I have entered in the Orlando Science center Science challenge competition. I was hesitant to do so, because it would eat up my time and I am not sure if It was worth it, but there would be cash prize possibility. Dr. Urbina highly encouraged me to join it, so what I decided to do is flip a coin and if it lands on heads I would go, and if It landed on tails I wouldn’t. It landed on heads , so now I am entered. I am hoping to do the level 1 change by before then but I am not sure.  I am currently running performance tests on google Colab. Google Colab has some limitations, for example there is not guaranteed run time, especially for the free version. I have also been seeing if using a GPU or TPU to enhance processing would help with making training processes faster. I ran some rudimentary tests and concluded that GPU or TPU doesn’t help. In fact it seems to take slightly more time than CPU. I think it has to do with GPU only being able to optimize tasks that can be done parallelly, and maybe it can’t do that for what I am working on. I’m not sure, but I will stick with the CPU setting.  I am actually kind of radically reworking my project. Due to the gradual decay of epsilon, I can’t just create one learning graph and see how progress goes over time, because what I set the total timesteps to changes how fast the epsilon changes and that changes the learning rate. Additionally, with my kind of learning bar graph, it is hard to numerically compare the performance with one single metric, because the graphs take an irregular shape.  However, I am thinking of one such metric that I am inventing myself. Essentially how it would work is that I would split the y-axis (learning performance in terms of %) into fixed intervals, say 10%, 20%, 30%, 40%, (...), 90%, 100%. For each item here, I would obtain the corresponding x-value (iterational time). So for example I would take 60%, and find most closely as possible what is the number of timesteps that a QL model or DQN model would have to take to have roughly an average closest to 60%. I do that for all of the intervals. If the model can never ever reach that percent (for example, if DQN cannot ever reach 100%), then the x-value would be infinite or undefined. Then I would have a list of each percent win rate and the corresponding number of timesteps. And for each of these pairs, I could find the slope by dividing the y value over the x value. I would then have 10 slope values. If one were to plot 10 lines with these slopes (with y-intercept 0) on top of the graph of performance, each one would intersect the graph exactly where it reaches a percent interval. For any x-value that is undefined, I will treat it as an infinite number and the y-value divided by it would yield a slope of 0. So essentially I have 10 values denoting slopes. My final metric, what I dub the Parayil metric, would be the average of all these 10 values. Theoretically (I’m not sure), it would take into account learning rate, irregular/unique shapes of a line graph, and y values that one model never reaches ever. I don’t know if something like this already exists (it probably does), but I am calling this for now, the Parayil metric. And then I would have the Parayil metrics of QL and DQN on the environment.  So then… If I am doing it like this, I would not necessarily have to kind of do a whole scan of the entire x-axis space. Instead, I could selectively find the x values for each of the 10 y-values.  My new planned methodology for a given environment would be:   1. Find the where the model converges (at what timestep and what win rate) 2. Perform binary search to find exactly at which timestep does the model train the agent to have a win rate closest to 10%, from range 0 to the timestep at which the model converges. 3. In doing binary search, instead of indexing list, run a test for the given timestep value. Each such test will be 100 trials. Each trial is training an agent for the given number of timesteps, and finding the performance afterward. After 100 trials, take the average of the performances and use that as the value. 4. Record the result of each test even if it is not the end result of binary search. That way, in binary searches for the other y-values, the program can refer to values already found from previous searches. 5. Repeat binary search for the other y-values (20%, 30%, etc.). Do not perform binary search for any values that are higher than what the model converges at. 6. For any values that are higher than what the model converges at, the x-value is undefined, but can be treated as being infinite. 7. There should now be the set of y-values at fixed intervals of 10% and their corresponding number of timesteps. 8. For each pair, for the x-values, convert timestep value to the amount of computing time. Run tests of 100 trials each to find the average time it takes for the computer to train the model for that many timesteps. So now the x-value is computing time in seconds instead of iterational time 9. For each pair, obtain the slope by dividing the y-value over the x-values. 10. There are now 10 values denoting slopes. Find the average of all 10 values. This value is a metric representing the overal performance of the model on the environment 11. Do steps 1-10 process for both QL and DQN on the environment 12. Do steps 1-11 for all of Gymnasium’s toy-text environments: FrozenLake 4x4, Cliff Walking, FrozenLake 8x8, Blackjack, and Taxi.   Here is a list of the complexities of the environment based on state\*action size  FrozenLake 4x4: 64  -Observation space: 16  -Action space: 4  Cliff Walking: 192  -Observation space: 48  -Action space: 4  FrozenLake 8x8: 256  -Observation space: 64  -Action space: 4  Blackjack: 1408  -Observation space: 704 - Tuple(Discrete(32), Discrete(11), Discrete(2))  -Action space: 2  Taxi: 3000  -Observation space: 500  -Action space: 6 |
| **3/6 WED** | I participated in the OSC science fair last Saturday. I happened to be the only person in the MACO category, and automatically won first place for that category, meaning I won a cash prize of 400 dollars, which is not bad at all. Even though it was not much of a competition, it was nevertheless a very good experience to articulate my project again to various judges of high background, especially regarding my uncertain continuation research. It was also nice and fun to meet people from other counties. After the competition it was mostly just messing around and stuff.  I must continue my extension research. I kind of abandoned the idea of 3 levels of adjustments. I am anyway going to do level 3. I have a good framework for doing that in my new procedure that I did. I believe that itself is very powerful and unique mathematical analysis.  I have a week before spring break.  My entire spring break I will be unavailable due to vacation in Grand Canyon.  I will have one week after that before the SSEF.  I need a week to prepare the poster presentation and verbal presentation to the best thoroughness.  So that means I need to finish the experimentation by before my spring break, which is a pretty tight deadline. |
| **3/7**  **Thur** | I have been continuing the coding of the binary search algorithm. Once I code this, I will have to evaluate it, and then I can essentially scale this and apply this for all the tests, completing “level 1” and “level 2”.  I wanted some sample data for this, and I got kind of sidetracked (but anyways still productive) into plotting the data into desmos as a piecewise function  <https://www.desmos.com/calculator/hhwjj0nw6a>  The result is an actual line graph of the data, with much more accurate proportions compared to the bar graph. It helps much more to visualize the difference between QL and DQN. Also, the lines enable prediction of data between data points.  In that way, I was able to actually graphically evaluate where does each model reach 50% win rate. For QL, it is at 729.167 timesteps; for DQN, it is at 78023.466.  78023.466/729.167 = 107.003561598.  This is much more accurate than using this same metric on the bar graph. |
| **3/24**  **Sun** | Progress has been slower than expected. I am having dry eye problems with my eye due to excessive screen time, greatly limiting my work capacity.  This Spring break we took vacation to the Grand Canyon. I think that the break from screen time and time spent outdoors helped my eye health.  I have to be very careful especially considering that the human eye was not meant to be exposed to extended periods of screen time.  I am back now and will do what I can with the limitations of my eye condition.  Here’s a memo to myself:  Essentially, what I am doing is comparing the performance of two reinforcement learning models, Q-Learning and Deep Q-Learning, on the FrozenLake environment from the Gym library.  How I am doing this is that I am obtaining essentially a learning graph of performance for both QL and DQN showing how their performance changes over timesteps.  To compare these two learning graphs is then another challenge. In summary, I am essentially obtaining a singular metric that I have invented myself, which I have called the Parayil Metric. It involves splitting the Y-axis (performance) into fixed intervals and finding the x-value(timesteps) at that y-value. So I would be finding the number of timesteps that it would take to yield a certain amount of performance on average, and do that for fixed intervals on the y-axis. Then it involves converting the x-value of the learning graph, the timesteps, from the value of the number of iterations to the actual computational time it takes to run a given number of iterations. To do this is that for each timestep I run an entire test where I train the model for that amount of time then record the time it took, and do this for ten trials each. Then, to obtain the Parayil metric, I would obtain the slopes for all these computed “points” and take the average of all these slope values.  Due to the variable nature of converting from timesteps to actual computing time, results can vary significantly. To understand how much it varies due to randomness, I have run a total of 66 entire Parayil Metric tests. I am wondering how significant the variability is.  [423.1658383414716, 423.6902683155414, 397.6240730958863, 395.9443302126059, 386.1437695432387, 411.79979782101935, 426.19564790423476, 453.38875415780274, 373.5257010290828, 383.71357543984533, 403.6293330295842, 410.20171796830937, 381.520639221764, 380.44431691444476, 390.9335615353823, 429.7634508146719, 398.8344621645339, 424.00565500126976, 385.1227025600632, 403.8128295310963, 371.28393719647573, 424.1710515771615, 429.53978937276753, 393.2154904163104, 403.08699910754507, 389.77214140191325, 392.16064066837765, 373.3307631815686, 387.57975746143336, 381.28075608893823, 391.82124137186145, 398.56158643248426, 393.7795677063533, 381.59147185581617, 392.09577664894397, 375.50201335242826, 406.1375705478684, 422.4612644457692, 406.65298774099546, 393.1485532106023, 381.7634961334246, 391.65359458231984, 433.68588159278727, 439.8362995175177, 398.4845858132661, 421.96016168430043, 425.68275371373966, 391.56599269879695, 411.19925901669075, 430.9727856558504, 378.86189749942116, 412.6041604299511, 417.44590345572334, 411.0477013510346, 390.89385900826136, 451.8373303371389, 376.7845857791006, 374.82563149791895, 376.1180782013297, 396.775010129379, 422.5136323258645, 418.8506759627855, 388.2091155946193, 423.4764276729628, 388.13808443899677, 372.35757392255505]  Length: 66  Average: 401.7905797333816  Standard Deviation: 20.32435937394631  Maximum value: 453.38875415780274  Minimum value: 371.28393719647573  Range: 82.104816961327  I think I need to increase the number of time test trials.  Here is a temporary plan for this  1. Gather all constants in the code into one recognizable place  2. Change the timestep to time trials from 10 to 100  3. Add data saving feature to save progress on timestep trials as well.  -but have a way to delete the progress to run it again  4. Prepare for running a new batch of parayil metric evaluations  5. Run 50 new parayil Metric evaluations  Here is a plan for my science fair until the State Science fair comes around  **Plan**  -Develop an organized sense of how I’m going to get this done  -Email an update to Dr. Mondesire  **Finish experimentation**  -Finish code finally for QL  -Run it for QL 4x4,  -Apply changes to DQN 4x4  -Run QL 8x8 and DQN 8x8  -Organize data as needed  -Look into doing other environments, otherwise stop here  -Review code; clean it up, and possibly publish it  **Plan verbal presentation**  -Emphasize how unique my project is  -Lot of think work here  -Can technically do before I finish the entire experimentation  **Verbal presentation**  -Scheduling and doing mock presentations to science fair staff  -Will have to talk with Dr Urbina about this  -Will have to plan verbal presentation before  **Plan changes to the poster**   * Make changes to digital poster to reflect changes * While considering physical limitations of Benton’s printer   **Update the poster**   * Remove old or decaying things * Print out the new things * Stick them on * Re-learn how to assemble board   **Prepare for the trip**   * Research the when and how * What should I pack? * Thinking some hypothetical situations in mind * What school work will I miss |
| **3/29**  **Fri** | Some things have happened since the last entry.  I was successful in applying analysis code to DQN 4x4, and also for both algorithms in the 8x8 environment.  Things were looking up, except for one concerning development:  Runtimes starting becoming excessively long.  The runtime for QL 4x4 was okay. In fact, I was able to run it multiple times.  However, the runtime for DQN 4x4 was another story altogether… it took much much longer to run, and I had to run it day and night continuously as much as possible for a few days and still it is not complete yet.  What’s even more is that the 8x8 training hit even harder. As I started to recognize gradually the seriousness of this extended run time, I took measures to speed it up. I borrowed my dad’s laptop with his google account to begin running QL 8x8 concurrently. I switched the DQN 4x4 to a secondary account to run on my home laptop. I used my primary account on my school-provided laptop to start running DQN 8x8.  Regarding DQN 8x8… it’s runtime is another story altogether. To perform only a singular training run, it took 5 minutes, meaning a single timestep test of 100 trials takes, well, 500 minutes. And there would be around 200 timetstep tests total needed to obtain the needed data for a Parayil Metric. So, 500 \* 200 = 100,000 minutes = 1666 hours = 69 days. 69 days of continuous run time. I would have to have the data and conclusions on the poster ready to go in 4 days. It is simply not possible.  I have overcame many challenges, but I could not overcome the challenge of time.  If I had more time I could look into getting a paid access to multiple GPU units, but I simply cannot. |
| **5/3 Fri** | The college term has ended, so all my dual-enrollment classes are ended.  I have just now completed the first draft for my github repos for the Parayil Metric. It includes a descriptive readme and the code for the analysis. I am planning to use this to present my findings regarding my Parayil Metric and obtaining the PC graphs of machine learning models to various professors and graduate students in the field.  The github repos will be heavily edited and added on to. I will eventually add my data, refine the code, add credits, make a better presentation, make a technical explanation of the methodology, etc. |

**Extension Experimentation:**

**Description**

**Performance-Complexity Graph**

The plan for the extension research was to compose overlapping performance complexity graphs of QL and DQN for varying environment complexities. The Y-axis would be performance; the X-axis would be complexity. However, I was not able to complete this graph. I was only able to get one data point for QL and DQN that could be charted on the graph.

**Parayil Metric**

Parayil Metric would be obtained for a given environment to have a singular number reprresenting algorithm performance.

The Parayil Metric recognizes the mathematical relationship between timesteps (x value) and performance (y value) that exists for any machine learning algorithm, called a learning graph. The metric aims to obtain a single numeric value to describe the nature of a given learning graph.

How the Parayil Metric works is that it splits the y-values (performance, in terms of percentage win rate) into fixed intervals (ex: : 5%, 10% [...] 90% 95% 100%). For each of these y values, I find the corresponding x value (timestep). In other words, I find what is that timestep that yields on average a win rate closest to the given y value. If the algorithm is not able to achieve the given win rate ever, an x value of infinity is assigned. I then transform the x-values from timesteps to actual physical time. So I would run tests determining on average how much time does it take to run the given amount of timesteps of training, running plenty of trials to account for randomness and differences in computational resources. For each of these transformed x-y pairs, I find the slopes by dividing the y over the x. Finally, the Parayil Metric would then be the average of all these slope values. Such a metric takes into account learning over time and is appropriately sensitive to performance limits that the algorithm may encounter in the environment. The Parayil Metric of one algorithm can then be compared with that of another algorithm, to find the comparative performance on a given environment. It is also applied in constructing the performance-complexity graph.

The benefit of using the Parayil Metric is that I only need to run timestep tests specifically fo the y-value intervals, instead of scanning the entire timestep space (which would take astronomically long). To find the corresponding x values, I use the binary search algorithm, which is normally used for searching for items in a list, but can be applied in this case quite well.

**Environment Complexity**

To evaluate environment complexity, there are endless factors that could be considered:

* State Space Size
* Action Space Size
* Dynamics and Stochasticity
* Exploration-Exploitation Trade-off
* Rewards and Goals
* Partial Observability
* Temporal Dependencies
* Multi-Agent Interactions
* Continuous State or Action Spaces
* Hierarchical Structure
* Environment Non-Stationarity
* Transferring Learning and Generalization
* Function Approximation Complexity
* Memory and History
* Resource Constraints
* Latency or Communication Constraints
* Uncertainty or Risk

For the purposes of this experiment however, the environment complexity metric would be simply the state size times the action space size.

The Gym Toy Text environments could then each be assigned this complexity metric:

|  |  |  |  |
| --- | --- | --- | --- |
| **Toy Text Environment** | **Observation Space** | **Action Space** | **Complexity Metric** |
| FrozenLake (4x4) | 16 | 4 | 64 |
| Cliff Walking | 48 | 4 | 192 |
| FrozenLake (8x8) | 64 | 4 | 256 |
| Blackjack | 704 | 2 | 1408 |
| Taxi | 500 | 6 | 3000 |

**Challenge of Time**

Despite the efficient nature of the Parayil Metric, running the full experiment obtaining Parayil Metrics for both QL and DQN for all the environments would take far, far too long given my current computing restrictions with Google Colab and the tight deadline for State Science Fair.

I was only able to obtain Parayil Metrics for QL FrozenLake 4x4 and DQN FrozenLake 4x4.

The runtime environment that I was using is known as Google Colab. which is a popular, established cloud-based platform that provides a convenient and collaborative environment for running and developing machine learning models. Google Colab provides limited computational resources to ensure fair usage and prevent abuse of the platform. This means that I don’t have access to unlimited CPUs or GPUs. The amount of runtimes that I can have at one time is also limited to around 1 or 2. What’s more is that a Google Colab runtime can get automatically disconnected for using certain resources for certain amounts of time. Google Colab is intentionally secretive of it’s specific conditions that would cause your code to get disconnected to prevent users from intentionally circumventing these conditions.

This makes running code for long periods of time difficult, as you don’t truly know when it can get disconnected due to approaching a runtime limit due to certain rules that you are not aware of.

To circumvent these obstacles (in a non-abusive way) I had to rely on anecdotal practices on the web. But more importantly, incorporating periodic and intelligent saving of data into the code was needed to ensure all progress would not be lost in the common case of disconnect. I also used multiple accounts on different computers in our household to run different tests at the same time.

But despite these measures, I could not overcome the fundamental challenge of time.

The QL 4x4 experimentation took a manageable amount of time. It took around a school day of running to complete, and I was able to run multiple instances of the experiment instead of just one. I failed to record the elapsed time; partly due to the periodic saving and intermittent nature of running the experiment. In retrospect, I wish that I did implement a way to record elapsed time, but at the moment I can only give a qualitative description of how long experimentation would take.

The DQN 4x4 experimentation, on the other hand, took significantly longer. This was because DQN naturally demands more computation for one iteration of training. It required around 3 days of running, day and night. Although I was not able to utilize all 24 hours of each day due to the runtime disconnections, I did run it for most of the day at least. Here I realized that time is a real concern. I began using multiple accounts and laptops as mentioned before.

When I began to run QL 8x8 concurrently, I realized that it takes EVEN longer than DQN 4x4. I estimate it was taking at least 3 or 4 times slower than DQN 4x4. This is due to the fact that each iteration for the 8x8 environment would take longer than the 4x4 environment because there is a larger max amount of timesteps the agent can take in one simulated run, as well as the fact that the amount of iterations required to achieve the desired win rates were naturally larger. I began realizing that runtime will be a huge obstacle that I may not be able to overcome

However, the final nail in the coffin came when I opened up a third laptop and Google Account and started running DQN 8x8. I realized now that completing this is impossible. It takes 5 minutes to perform only a singular training run, meaning a single timestep test of 100 trials takes 500 minutes. And there would be a number of timestep tests roughly around the magnitude of 200 total needed to obtain the needed data for a Parayil Metric. So, 500 \* 200 = 100,000 minutes = 1666 hours = 69 days. 69 days of continuous run time to obtain the Parayil Metric for DQN 8x8.

This was simply not possible, and I was forced to accept that. In the future, what I can do is get access to a paid subscription for more computational resources and split the processing time into different compute units to achieve this.

But what I was able to achieve is pioneer a unique and powerful method for evaluating the otherwise unevaluateable complexity-performacne graph.

But as of now, I only have Parayil Metrics for the FrozenLake 4x4 environment.

**Data and Results**

**FrozenLake 4x4**

**Parayil Metric Results:**

QL Parayil Metric Test 1:241.52472611055873

QL Parayil Metric Test 2: 252.48861428487203

QL Parayil Metric Test 3: 242.022027824301

**QL Parayil Metric AVG**: 245.34512274

DQN Parayil Metric Test 1: 4.474995695676713

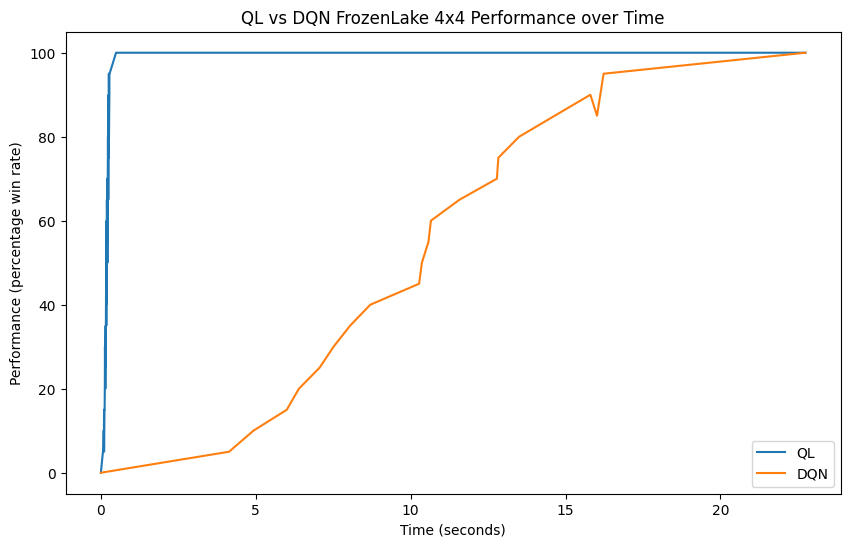
**DQN Parayil Metric AVG**: 4.474995695676713

245.34512274/4.474995695676713 = **54.8257784867**

**QL is approximately 55 times better than DQN at FrozenLake 4x4.**

**Graph**:

(Based on the points obtained from fixed y-intervals)



**Raw Text Result Output:**

From result.txt file

|  |  |  |
| --- | --- | --- |
| **QL FrozenLake 4x4 Test 1:** | **QL FrozenLake 4x4 Test 2:** | **QL FrozenLake 4x4 Test 3:** |
| **-----------------------------------------------**  QL 4x4 TEST RESULTS  TRIALS PER TEST:1000  Y-INTERVALS: 5  TIME ELAPSED: 0.636402  CURRENT TIME: 2024-03-26 21:51:21  -----------------------  RESULTS WITH X-AXIS MODE: ITERATIONS  POINTS:  (74, 5)  (105, 10)  (126, 15)  (147, 20)  (172, 25)  (202, 30)  (217, 35)  (243, 40)  (269, 45)  (300, 50)  (312, 55)  (351, 60)  (403, 65)  (458, 70)  (495, 75)  (560, 80)  (687, 85)  (779, 90)  (968, 95)  (1929, 100)  SLOPES:  0.06756756756756757,  0.09523809523809523,  0.11904761904761904,  0.1360544217687075,  0.14534883720930233,  0.1485148514851485,  0.16129032258064516,  0.1646090534979424,  0.16728624535315986,  0.16666666666666666,  0.1762820512820513,  0.17094017094017094,  0.16129032258064516,  0.15283842794759825,  0.15151515151515152,  0.14285714285714285,  0.12372634643377002,  0.11553273427471117,  0.0981404958677686,  0.05184033177812338  (Incomplete) PARAYIL METRIC:  0.13582934279459938  -----------------------  RESULTS WITH X-AXIS MODE: TIME  Trials for each time test: 100  POINTS:  (0.07091304999999999, 5)  (0.09957491999999997, 10)  (0.11017667000000006, 15)  (0.12678253, 20)  (0.14064601, 25)  (0.15757235999999997, 30)  (0.16864259000000004, 35)  (0.17347864999999998, 40)  (0.17218035000000007, 45)  (0.18017673999999995, 50)  (0.20672877999999997, 55)  (0.20990858999999992, 60)  (0.21213413000000003, 65)  (0.2384446699999999, 70)  (0.25423864000000007, 75)  (0.2284403899999999, 80)  (0.24979698999999989, 85)  (0.2601355300000001, 90)  (0.2803084, 95)  (0.5042343, 100)  SLOPES:  70.50888376681021,  100.4268946437517,  136.1449751567187,  157.75044085332576,  177.75122095536165,  190.38872045833423,  207.53950707232374,  230.57592389611057,  261.35386529299063,  277.5052984086626,  266.04907163869495,  285.83870722012864,  306.40991150268934,  293.5691537999152,  294.9984313950074,  350.200767911489,  340.27631798125367,  345.97350081321053,  338.91242645600346,  198.3205029883925  PARAYIL METRIC:  241.52472611055873  -----------------------  RESULTS OF ALL TESTS PERFORMED (X-AXIS MODE: ITERATIONS):  (1000, 96.0)  (500, 75.5)  (250, 43.4)  (125, 13.2)  (62, 3.4)  (93, 8.9)  (77, 6.4)  (69, 3.0)  (73, 2.6)  (75, 5.6)  (74, 5.8)  (109, 10.1)  (101, 8.3)  (105, 10.2)  (103, 9.2)  (104, 8.7)  (187, 26.4)  (156, 21.1)  (140, 16.7)  (132, 15.2)  (128, 17.1)  (126, 15.9)  (148, 20.0)  (144, 18.5)  (146, 16.8)  (147, 20.3)  (171, 22.4)  (179, 25.9)  (175, 25.9)  (173, 25.8)  (172, 25.0)  (218, 35.7)  (202, 29.4)  (210, 30.9)  (206, 31.5)  (204, 31.0)  (203, 31.4)  (214, 33.4)  (216, 33.7)  (217, 34.6)  (234, 36.3)  (242, 39.9)  (246, 42.5)  (244, 40.6)  (243, 39.9)  (375, 60.2)  (312, 55.5)  (281, 46.5)  (265, 43.8)  (273, 45.6)  (269, 45.6)  (267, 42.4)  (268, 41.5)  (296, 48.4)  (304, 51.0)  (300, 50.7)  (298, 48.1)  (299, 48.7)  (308, 52.9)  (310, 52.9)  (311, 52.4)  (343, 57.8)  (359, 61.5)  (351, 61.3)  (347, 59.3)  (349, 58.1)  (350, 57.2)  (437, 67.4)  (406, 65.1)  (390, 64.2)  (398, 63.0)  (402, 62.5)  (404, 66.0)  (403, 66.0)  (468, 72.2)  (452, 68.1)  (460, 70.9)  (456, 68.8)  (458, 69.0)  (459, 71.5)  (484, 71.5)  (492, 73.4)  (496, 75.3)  (494, 72.2)  (495, 75.3)  (750, 87.7)  (625, 83.6)  (562, 80.2)  (531, 77.8)  (546, 77.3)  (554, 78.4)  (558, 79.9)  (560, 78.9)  (561, 81.9)  (687, 84.8)  (718, 89.7)  (702, 89.3)  (694, 85.2)  (690, 87.9)  (688, 85.8)  (875, 92.5)  (812, 91.6)  (781, 90.7)  (765, 89.7)  (773, 88.7)  (777, 89.2)  (779, 90.1)  (778, 88.9)  (937, 94.9)  (968, 95.3)  (952, 94.9)  (960, 93.3)  (964, 94.4)  (966, 94.8)  (967, 93.4)  (1500, 98.3)  (1750, 99.5)  (1875, 99.8)  (1937, 100.0)  (1906, 99.9)  (1921, 99.7)  (1929, 100.0)  (1925, 99.8)  (1927, 99.7)  (1928, 99.9)  -----------------------  END OF QL 4x4 RESULTS.  ----------------------------------------------- | **-----------------------------------------------**  **QL 4x4 TEST RESULTS**  **TRIALS PER TEST:1000**  **Y-INTERVALS: 5**  **TIME ELAPSED: 0.509643**  **CURRENT TIME: 2024-03-27 19:01:50**  **-----------------------**  **RESULTS WITH X-AXIS MODE: ITERATIONS**  **POINTS:**  **(73, 5)**  **(93, 10)**  **(128, 15)**  **(157, 20)**  **(169, 25)**  **(193, 30)**  **(222, 35)**  **(247, 40)**  **(282, 45)**  **(308, 50)**  **(336, 55)**  **(356, 60)**  **(407, 65)**  **(450, 70)**  **(503, 75)**  **(579, 80)**  **(640, 85)**  **(800, 90)**  **(987, 95)**  **(1963, 100)**  **SLOPES:**  **0.0684931506849315,**  **0.10752688172043011,**  **0.1171875,**  **0.12738853503184713,**  **0.14792899408284024,**  **0.15544041450777202,**  **0.15765765765765766,**  **0.16194331983805668,**  **0.1595744680851064,**  **0.16233766233766234,**  **0.1636904761904762,**  **0.16853932584269662,**  **0.1597051597051597,**  **0.15555555555555556,**  **0.14910536779324055,**  **0.1381692573402418,**  **0.1328125,**  **0.1125,**  **0.09625126646403243,**  **0.05094243504839532**  **PARAYIL METRIC:**  **0.13463749639430508**  **-----------------------**  **RESULTS WITH X-AXIS MODE: TIME**  **Trials for each time test: 100**  **POINTS:**  **(0.06663858000000002, 5)**  **(0.08023739, 10)**  **(0.10257449000000002, 15)**  **(0.14435933, 20)**  **(0.13658008000000005, 25)**  **(0.14216934000000003, 30)**  **(0.15545329000000002, 35)**  **(0.17350957000000006, 40)**  **(0.16819119999999999, 45)**  **(0.17048196000000002, 50)**  **(0.21556605, 55)**  **(0.18453430000000004, 60)**  **(0.19415799999999994, 65)**  **(0.20802389000000002, 70)**  **(0.23433044000000008, 75)**  **(0.2616203900000001, 80)**  **(0.24504070999999994, 85)**  **(0.25141562, 90)**  **(0.25786839000000006, 95)**  **(0.49000038999999973, 100)**  **SLOPES:**  **75.03161081763746,**  **124.63017553287811,**  **146.23518966557862,**  **138.5431755605959,**  **183.04279804199845,**  **211.01596166937256,**  **225.1480171310623,**  **230.53483447627693,**  **267.5526424688093,**  **293.28616353307996,**  **255.1422174317338,**  **325.14280542966804,**  **334.7788914183295,**  **336.49981259364006,**  **320.06085082245386,**  **305.78656350141506,**  **346.8811366078723,**  **357.97298513115453,**  **368.40498364301254,**  **204.08147022087076**  **PARAYIL METRIC:**  **252.48861428487203**  **-----------------------**  **RESULTS OF ALL TESTS PERFORMED (X-AXIS MODE: ITERATIONS):**  **(1000, 95.3)**  **(500, 73.3)**  **(250, 40.9)**  **(125, 13.6)**  **(62, 3.4)**  **(93, 10.0)**  **(77, 5.6)**  **(69, 3.8)**  **(73, 4.6)**  **(75, 5.9)**  **(74, 5.8)**  **(85, 6.3)**  **(89, 8.3)**  **(91, 8.2)**  **(92, 8.5)**  **(187, 26.4)**  **(156, 19.3)**  **(140, 17.7)**  **(132, 16.4)**  **(128, 14.6)**  **(130, 15.4)**  **(129, 15.8)**  **(171, 26.6)**  **(163, 24.1)**  **(159, 22.3)**  **(157, 20.3)**  **(167, 23.2)**  **(169, 26.0)**  **(168, 23.4)**  **(218, 33.6)**  **(202, 34.3)**  **(194, 30.3)**  **(190, 26.9)**  **(192, 27.3)**  **(193, 29.8)**  **(234, 39.2)**  **(226, 35.4)**  **(222, 36.3)**  **(220, 32.9)**  **(221, 33.3)**  **(242, 39.6)**  **(246, 36.9)**  **(248, 40.1)**  **(247, 41.7)**  **(375, 61.7)**  **(312, 51.1)**  **(281, 43.6)**  **(296, 49.8)**  **(288, 47.9)**  **(284, 47.7)**  **(282, 44.8)**  **(283, 49.1)**  **(304, 48.9)**  **(308, 48.3)**  **(310, 52.1)**  **(309, 52.0)**  **(343, 58.6)**  **(327, 54.1)**  **(335, 52.6)**  **(339, 56.1)**  **(337, 56.8)**  **(336, 54.2)**  **(359, 61.7)**  **(351, 57.5)**  **(355, 56.9)**  **(357, 61.2)**  **(356, 59.4)**  **(437, 69.9)**  **(406, 63.4)**  **(421, 66.3)**  **(413, 66.4)**  **(409, 65.9)**  **(407, 65.8)**  **(468, 71.4)**  **(452, 71.3)**  **(444, 67.7)**  **(448, 67.3)**  **(450, 70.9)**  **(449, 68.4)**  **(750, 88.8)**  **(625, 84.4)**  **(562, 79.5)**  **(531, 76.9)**  **(515, 77.6)**  **(507, 75.7)**  **(503, 75.8)**  **(501, 74.3)**  **(502, 74.2)**  **(593, 80.7)**  **(577, 79.3)**  **(585, 82.0)**  **(581, 80.0)**  **(579, 79.4)**  **(580, 82.7)**  **(687, 88.4)**  **(656, 85.3)**  **(640, 85.7)**  **(632, 83.2)**  **(636, 84.7)**  **(638, 82.8)**  **(639, 83.6)**  **(875, 92.1)**  **(812, 90.2)**  **(781, 89.7)**  **(796, 89.7)**  **(804, 90.4)**  **(800, 89.8)**  **(802, 93.0)**  **(801, 91.2)**  **(937, 94.8)**  **(968, 93.6)**  **(984, 94.4)**  **(992, 95.3)**  **(988, 95.9)**  **(986, 94.3)**  **(987, 94.9)**  **(1500, 99.2)**  **(1750, 99.6)**  **(1875, 99.5)**  **(1937, 99.8)**  **(1968, 100.0)**  **(1952, 99.7)**  **(1960, 99.8)**  **(1964, 100.0)**  **(1962, 99.8)**  **(1963, 100.0)**  **-----------------------**  **END OF QL 4x4 RESULTS.**  **-----------------------------------------------** | **-----------------------------------------------**  QL 4x4 TEST RESULTS  TRIALS PER TEST:100  Y-INTERVALS: 5  TIME ELAPSED: 0.439608  CURRENT TIME: 2024-03-27 19:58:12  -----------------------  RESULTS WITH X-AXIS MODE: ITERATIONS  POINTS:  (94, 5)  (112, 10)  (146, 15)  (148, 20)  (155, 25)  (155, 30)  (216, 35)  (273, 40)  (278, 45)  (324, 50)  (338, 55)  (381, 60)  (437, 65)  (442, 70)  (479, 75)  (500, 80)  (688, 85)  (744, 90)  (750, 95)  (1527, 100)  SLOPES:  0.05319148936170213,  0.08928571428571429,  0.10273972602739725,  0.13513513513513514,  0.16129032258064516,  0.1935483870967742,  0.16203703703703703,  0.14652014652014653,  0.1618705035971223,  0.15432098765432098,  0.16272189349112426,  0.15748031496062992,  0.14874141876430205,  0.1583710407239819,  0.15657620041753653,  0.16,  0.12354651162790697,  0.12096774193548387,  0.12666666666666668,  0.06548788474132286  PARAYIL METRIC:  0.13702495613124752  -----------------------  RESULTS WITH X-AXIS MODE: TIME  Trials for each time test: 100  POINTS:  (0.10299303999999998, 5)  (0.09491453, 10)  (0.12113869000000001, 15)  (0.12722308999999998, 20)  (0.14085497000000008, 25)  (0.13158173000000004, 30)  (0.17096640999999996, 35)  (0.17051967, 40)  (0.18493386999999994, 45)  (0.21118075999999994, 50)  (0.20345293999999994, 55)  (0.21584956000000005, 60)  (0.23939756999999984, 65)  (0.20613378000000013, 70)  (0.23130877999999996, 75)  (0.23685252999999992, 80)  (0.25081310000000007, 85)  (0.27420083000000006, 90)  (0.27502262999999993, 95)  (0.4053928499999999, 100)  SLOPES:  48.54696977582175,  105.3579467759046,  123.82501412224285,  157.20416789122166,  177.48752493433483,  227.99517835796803,  204.71857600566105,  234.57704322322462,  243.33022393356075,  236.7639930834609,  270.33278555719085,  277.97137969611794,  271.51486959537664,  339.5852926191911,  324.24190728946826,  337.7629109556061,  338.8977688964411,  328.22657757819326,  345.4261200251049,  246.6743061699288  PARAYIL METRIC:  242.022027824301  -----------------------  RESULTS OF ALL TESTS PERFORMED (X-AXIS MODE: ITERATIONS):  (1000, 99.0)  (500, 83.0)  (250, 38.0)  (125, 14.0)  (62, 3.0)  (93, 2.0)  (109, 9.0)  (101, 10.0)  (97, 8.0)  (95, 9.0)  (94, 7.0)  (117, 10.0)  (113, 13.0)  (111, 4.0)  (112, 16.0)  (187, 31.0)  (156, 34.0)  (140, 14.0)  (148, 19.0)  (144, 12.0)  (146, 10.0)  (147, 21.0)  (152, 21.0)  (150, 23.0)  (149, 25.0)  (154, 18.0)  (155, 28.0)  (218, 39.0)  (202, 24.0)  (210, 25.0)  (214, 31.0)  (216, 36.0)  (215, 27.0)  (375, 58.0)  (312, 48.0)  (281, 47.0)  (265, 38.0)  (273, 39.0)  (277, 41.0)  (275, 44.0)  (274, 46.0)  (279, 50.0)  (278, 42.0)  (343, 57.0)  (327, 53.0)  (319, 49.0)  (323, 43.0)  (325, 55.0)  (324, 54.0)  (335, 53.0)  (339, 58.0)  (337, 50.0)  (338, 59.0)  (437, 62.0)  (406, 63.0)  (390, 64.0)  (382, 61.0)  (378, 51.0)  (380, 58.0)  (381, 62.0)  (468, 73.0)  (452, 72.0)  (444, 70.0)  (440, 68.0)  (438, 73.0)  (442, 68.0)  (443, 73.0)  (484, 78.0)  (476, 71.0)  (480, 79.0)  (478, 65.0)  (479, 74.0)  (492, 62.0)  (496, 75.0)  (498, 73.0)  (499, 70.0)  (750, 96.0)  (625, 83.0)  (687, 77.0)  (718, 86.0)  (702, 85.0)  (694, 85.0)  (690, 88.0)  (688, 86.0)  (734, 88.0)  (742, 87.0)  (746, 92.0)  (744, 87.0)  (745, 94.0)  (748, 91.0)  (749, 90.0)  (1500, 98.0)  (1750, 100.0)  (1625, 100.0)  (1562, 100.0)  (1531, 100.0)  (1515, 98.0)  (1523, 98.0)  (1527, 100.0)  (1525, 98.0)  (1526, 98.0)  -----------------------  END OF QL 4x4 RESULTS.  ----------------------------------------------- |

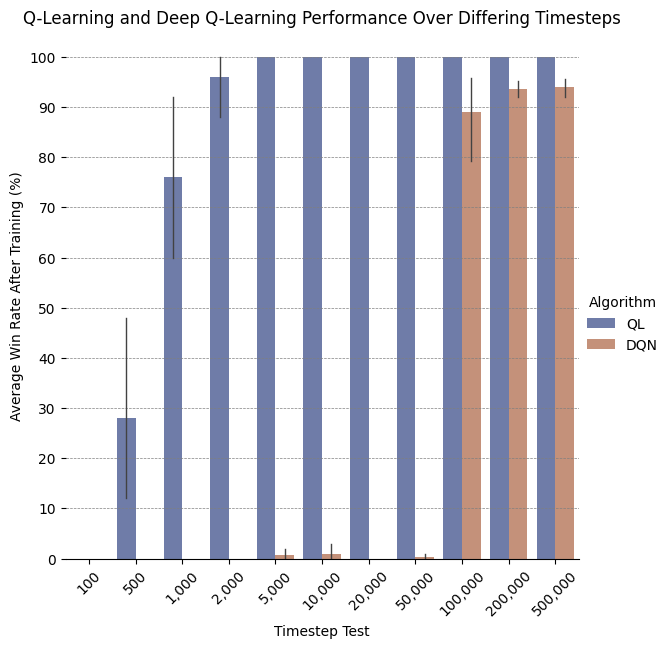
|  |
| --- |
| **DQN FrozenLake 4x4 Test 1:** |
| -----------------------------------------------  DQN 4x4 TEST RESULTS  TRIALS PER TEST:100  Y-INTERVALS: 5  TIME ELAPSED: 22.717244  CURRENT TIME: 2024-03-30 12:45:14  -----------------------  RESULTS WITH X-AXIS MODE: ITERATIONS  POINTS:  (237, 5)  (279, 10)  (322, 15)  (355, 20)  (374, 25)  (375, 30)  (414, 35)  (438, 40)  (515, 45)  (516, 50)  (516, 55)  (531, 60)  (556, 65)  (615, 70)  (615, 75)  (655, 80)  (762, 85)  (766, 90)  (895, 95)  (1220, 100)  SLOPES:  0.02109704641350211,  0.035842293906810034,  0.046583850931677016,  0.056338028169014086,  0.06684491978609626,  0.08, 0.08454106280193237,  0.091324200913242,  0.08737864077669903,  0.09689922480620156,  0.1065891472868217,  0.11299435028248588,  0.11690647482014388,  0.11382113821138211,  0.12195121951219512,  0.12213740458015267,  0.1115485564304462,  0.1174934725848564,  0.10614525139664804  , 0.08196721311475409  PARAYIL METRIC:  0.08892017483625304  -----------------------  RESULTS WITH X-AXIS MODE: TIME  Trials for each time test: 100  POINTS:  (4.13868912, 5)  (4.92321137, 10)  (6.00208012, 15)  (6.392633440000002, 20)  (7.060726490000003, 25)  (7.511734529999999, 30)  (8.043682250000003, 35)  (8.69832836, 40)  (10.273650100000003, 45)  (10.362943770000001, 50)  (10.576971559999999, 55)  (10.653614070000001, 60)  (11.576857619999993, 65)  (12.78363452, 70)  (12.831841620000002, 75)  (13.502980610000003, 80)  (16.01703726, 85)  (15.803400120000006, 90)  (16.229609699999997, 95)  (22.745284580000007, 100)  SLOPES:  1.2081120023820489,  2.0311945290295346,  2.4991335837083097,  3.1286010980789154,  3.5407121399486456,  3.993751360646128,  4.351241000351547,  4.5985847331245155,  4.3801374936839625,  4.824883846687126,  5.199976164065625,  5.631891638440024,  5.614649685913649,  5.475751038602127,  5.844835232621893,  5.924617853687341,  5.306849114490978,  5.694976987015625,  5.853498744335178,  4.396515666721105  PARAYIL METRIC:  4.474995695676713  -----------------------  RESULTS OF ALL TESTS PERFORMED (X-AXIS MODE: ITERATIONS):  (1500, [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (750, [100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0])  (375, [0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0])  (187, [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (281, [0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0])  (234, [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (257, [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (245, [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (239, [0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (236, [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0])  (237, [0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (269, [0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (275, [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (278, [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (279, [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (328, [0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0])  (304, [0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0])  (316, [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (322, [0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0])  (319, [0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (320, [100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (321, [100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0])  (351, [0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (363, [0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0])  (357, [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0])  (354, [0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0])  (355, [0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (356, [0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (369, [0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (372, [0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0])  (373, [100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (374, [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0])  (562, [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0])  (468, [100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0])  (421, [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0])  (398, [100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0])  (409, [0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0])  (415, [0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0])  (412, [0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0])  (413, [0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0])  (414, [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0])  (444, [100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0])  (432, [100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0])  (438, [100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0])  (435, [0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (436, [100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0])  (437, [100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0])  (515, [100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0])  (538, [100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0])  (526, [0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0])  (520, [100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0])  (517, [0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0])  (516, [0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (532, [100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0])  (529, [0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0])  (530, [100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0])  (531, [100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0])  (550, [100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0])  (556, [100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0])  (553, [100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0])  (554, [0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0])  (555, [100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0])  (656, [100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0])  (609, [0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0])  (632, [100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0])  (620, [100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0])  (614, [100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0])  (617, [100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0])  (615, [0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0])  (644, [0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0])  (650, [100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0])  (653, [100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0])  (654, [100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0])  (655, [100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0])  (1125, [100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0])  (937, [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (843, [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (796, [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (773, [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (761, [0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0])  (767, [100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (764, [100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (762, [100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 0.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0])  (765, [100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (766, [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0])  (890, [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (913, [0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (901, [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (895, [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (892, [0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (893, [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (894, [100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (1312, [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (1218, [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (1265, [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (1241, [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (1229, [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (1223, [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (1220, [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  (1219, [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 0.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0])  -----------------------  END OF DQN 4x4 RESULTS.  ----------------------------------------------- |

**Original Experimentation:**

**Results Summary**

The data of this experiment came in the form of two JSON files. One JSON file represents the experimentation on the QL model, and the other represents the experimentation on the DQN model. In each file, for each of the 11 testValues, there are 25trials, each represented by a number value from 0-25 representing the amount of wins out of 25 test runs of the trained agent.

A dual-bar graph was made. Blue bars represent QL and red bars represent DQN. On the horizontal axis, each category will represent each of the 11 testValues. The vertical axis will represent the average performance (total wins out of 25) after training for that testValue, ranging from 0-25. For each testValue category, the blue and red bar will be shown

* + 
  + Graph was made in python environment using the Seaborn library
  + In addition to the graph, a table representing 11 t-tests for each category was made. The point of the T-tests is to provide statistical backing for the results shown in the graph.

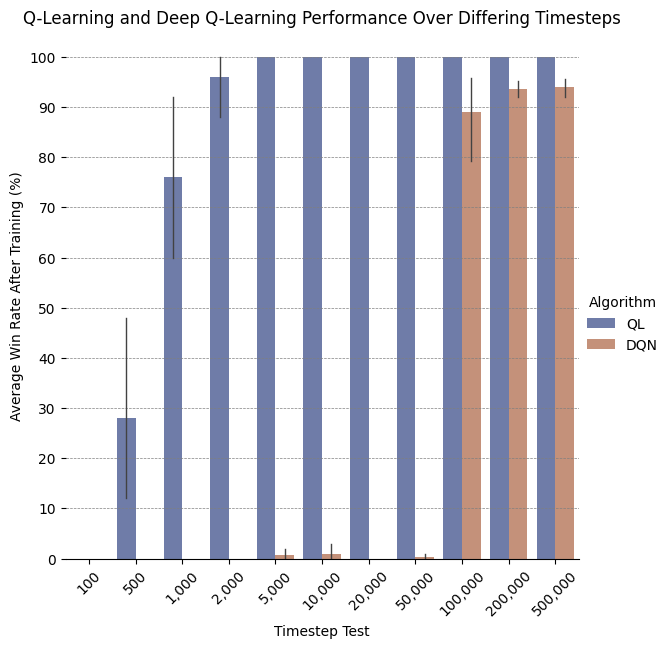
Q-learning starts getting results at 500 timesteps. By the 5,000 timestep test, it achieves 100% win rate.

Whereas, Deep Q-learning stays close to 0% until the 100,000 timestep test, where it reaches an important threshold and gets a result of about 89%. It levels at a win rate of around 94%, never reaching a 100% win rate like Q-learning.

The smallest time step test where Q-learning reaches above a 50% win rate is 1,000, whereas for DQN, it is 100,000. That means DQN took a 100 times longer than QL, regarding this metric.

QL reaches a 100% win rate by the 5,000 timestep test, whereas DQN never reaches a 100% win rate

It’s worth noting that although the graph suggests that a given Q-learning model’s learning progress was a gradual, continuous climb to 100%, a more accurate picture for the progress of Q-learning is staying at 0%, then at some threshold jump to 100%. The exact point of this threshold varies from trial to trial due to randomness, and thus an average of the different trials show a gradual climb.

**Bar Graph:**

**T-tests**

Yellow rows indicate T-tests with a P-value low enough to be considered significant.

|  |  |  |  |
| --- | --- | --- | --- |
| **Comparison** | **T-value** | **P-value** | **Significant?**  (Alpha value = 0.05) |
| QL vs DQN at 100 | 0 | 1 | **No** |
| QL vs DQN at 500 | 3.05505 | 0.003667 | **Yes** |
| QL vs DQN at 1000 | 8.7178 | Less than 0.00001 | **Yes** |
| QL vs DQN at 2000 | 24 | Less than 0.00001 | **Yes** |
| QL vs DQN at 5000 | 155.25 | Less than 0.00001 | **Yes** |
| QL vs DQN at 10000 | 103.16667 | Less than 0.00001 | **Yes** |
| QL vs DQN at 20000  (both standard deviations were 0, so T-test could not be performed) | N/A | N/A | **N/A** |
| QL vs DQN at 50000 | 311.5 | Less than 0.00001 | **Yes** |
| QL vs DQN at 100000 | 2.41981 | Less than 0.00001 | **Yes** |
| QL vs DQN at 200000 | 8 | Less than 0.00001 | **Yes** |
| QL vs DQN at 500000 | 6.55725 | Less than 0.00001 | **Yes** |

**RAW DATA:**

|  |  |
| --- | --- |
| **QL\_experiment\_data.json** | **DQN\_experiment\_data.json** |
| {  "testValues": [  100,  500,  1000,  2000,  5000,  10000,  20000,  50000,  100000,  200000,  500000  ],  "numTestValues": 11,  "amountOfTrials": 25,  "performanceCheckTrials": 25,  "tests": [  [  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0  ],  [  0,  0,  0,  0,  25,  0,  0,  25,  0,  25,  0,  0,  0,  25,  25,  25,  0,  0,  0,  25,  0,  0,  0,  0,  0  ],  [  25,  0,  25,  25,  25,  25,  25,  0,  25,  25,  25,  25,  0,  0,  0,  25,  25,  25,  25,  25,  25,  25,  25,  25,  0  ],  [  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  0,  25  ],  [  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25  ],  [  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25  ],  [  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25  ],  [  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25  ],  [  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25  ],  [  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25  ],  [  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25,  25  ]  ]  } | {  "testValues": [  100,  500,  1000,  2000,  5000,  10000,  20000,  50000,  100000,  200000,  500000  ],  "numTestValues": 11,  "amountOfTrials": 25,  "performanceCheckTrials": 25,  "tests": [  [  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0  ],  [  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0  ],  [  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0  ],  [  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0  ],  [  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  4.0,  0.0,  0.0,  0.0  ],  [  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  6.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0  ],  [  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0  ],  [  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  2.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0  ],  [  24.0,  25.0,  23.0,  25.0,  24.0,  23.0,  25.0,  24.0,  24.0,  23.0,  23.0,  24.0,  24.0,  5.0,  24.0,  24.0,  24.0,  23.0,  23.0,  24.0,  25.0,  23.0,  25.0,  23.0,  2.0  ],  [  22.0,  24.0,  23.0,  23.0,  25.0,  25.0,  22.0,  24.0,  24.0,  25.0,  23.0,  23.0,  25.0,  23.0,  24.0,  22.0,  23.0,  22.0,  24.0,  23.0,  23.0,  22.0,  24.0,  24.0,  23.0  ],  [  24.0,  23.0,  23.0,  25.0,  23.0,  23.0,  24.0,  24.0,  24.0,  24.0,  24.0,  25.0,  23.0,  25.0,  24.0,  24.0,  25.0,  21.0,  21.0,  23.0,  24.0,  23.0,  24.0,  23.0,  21.0  ]  ]  } |