ChenruiXu 515Midterm

Chenrui Xu

October 2021

1 Question1

So for this question, we will use automata theroy.

So at first, we need to define what is the metric of M1. It should be the amount of path possibly exist by given a walk α. As non-deterministic is the amount of choices we have. The higher ND we have, the more choices we have.

Here is the idea. So here, the graph we have can be seen as a non-deterministic finite automaton (NFA). And we can turn the non-deterministic finite automaton into a deterministic finite automaton (DFA). After that, it is easy to figure out, if we are given some walk in the set of P, the amount of paths that possible is the multiplications of the number of elements in each node (if the walk α will go that way) in the deterministic finite automaton. We can know that the essential of the problem is path counting. However, if we directly use the original graph to do the path counting, we can use tree method. We can also figure out some numbers. However, there are many shortcomings if we use the NFA directly. For example, the amount of paths will increase in the speed of exponential, which will take us or machine a branch of time on counting or running code (time complexity). On the other hand, some times the walk will go into the some path circle and will not get out forever. In this case, we cannot do the path counting as there are infinity possibly methods to go.

Here is the method: When we get the NFA graph G, we will know the description of automata: N = (S,I,f,A,S0). Where S is the finite set of states and I is a finite set of inputs. A can be seen as the result after walking, in another word, acceptable state. S0 means the initial state. Function f is the mapping if given S and a element in I (state transaction). So here, I is color set and S is the node sets. Let u=S0.

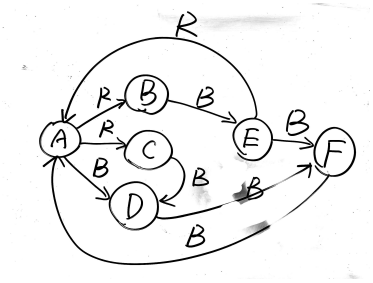

Figure 1: Figure 1

Here is an example of the NDA graph. And we can let $u = A$ to simplify the description, the graph is not complicated, but since we know how it works, we can turn very huge NFA graph easily.

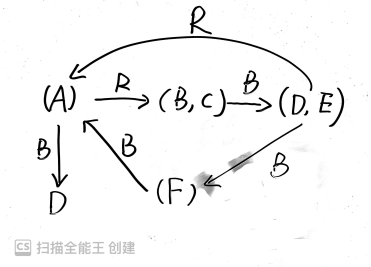

Figure 2: Figure 2

And here is the corresponding DFA graph. Start from A, if by going through the same color, we get different nodes, we will put that nodes into a set as a new node. In the graph, by going through Red edge, we can get to B and c, so the corresponding node is (B,C). Any element in the node set can get access to other nodes. Note that there are two Ds here. That means two different result as from A, it went through different path. At the end, we can get a DFA graph N' = (S',I',f',A',S0).

As given information, we can walk in DFA graph based on given information. For example, we take a walk, there are 1*2*2*1=4 choices to go for the same color RBR (each number here is the number of elements in the passing set).That is nodes in the graph are seen as all possible states from the previous state by going through same color. So here one walk of M1 is 4.

By using this method, it is easy for us to figure out how the M1 is.

2 Question2

This question is different from the first one. For this one, we need to figure out how many choice we have in terms of colors. We can say that after passing one node, if there are two different kinds of color to choose, it has the non-deterministic. And if only one kind of color to choose, even there are two edges, we can say there is no non-deterministic here.

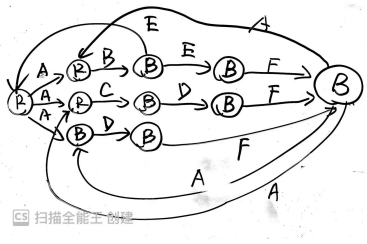

Figure 3: Figure 3

Here, above graph shows the core idea of the question. It is actually the same NFA graph we are using in the question one. Why we say they are kind of same? Because here I turn all the nodes into edges and all edges into nodes. That means I turned all colors into nodes and all original nodes to edges, so that we can analysis original nodes with same color more convenient.

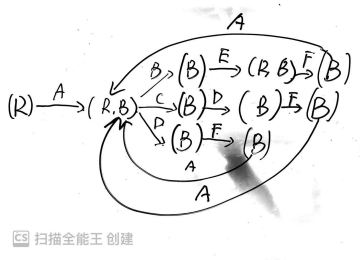

Figure 4: Figure 4

And the following things are quite similar with question 1. We turn NFA into DFA, that is what is showing in figure 4.

Notice that in some nodes the degree of the set is two, that means when passing, we will have two choices then. And the last thing we need to do is time possible situation together. Like AB there are two possible sequences here (R,R) and (R,B) or BEF possible choices like (B,B) and (B,A).

3 Question3

3.1 Approach

So in this question, what we only know is that the input and output. We do not know how the black box run inside. Also, we need to know that given several series of test results, if for index i, the pair in test1 is different from the pair in test2 (given the same input), we can say that these two pairs are non-deterministic. However, regardless how index from i+1 to the end look like, we cannot say those pairs have non-deterministic property as the previous pairs have already been different. For example, in test 1 $(b1, r), (b2, g)$ and test 2 $(b1, g), (b2, r)$. It is not hard for us to say that $(b1, r) and (b1, g)$ are non-deterministic and then, we do not have to make judgment on $(b2, g) and (b2, r)$.

At first, we need to define the metric of these black box problem. Here we use information entropy as the idea. Entropy was first used by physicist to describe how mess the item is. So the basic idea of the information entropy is that, after something happened, we can always use the information entropy to quantify how much information we get from that thing. It is one method to measure non-deterministic. It also has the property that if something always happen without any hesitation, the corresponding information entropy can be very small as we all know that something as usual will not take us much useful information. However, if something is not deterministic, and it happens for some situations, the small chance things happened, which will give us a lot of information to deal with. In this question, given a black box, if every time we push button one, the color is always red. We will have a very low information entropy and that means the level of non-deterministic is very low then. On the opposite, if we push same button several times and each time we can barely get the same result, we will say that, the system has a high level of non-deterministic. On the other hand, the corresponding information entropy can be very high. And it is known that when the results are not balanced, the information entropy can be smaller than the case that every result has the same chance to appear (which is known as the upper bound of information entropy). However, information entropy will not be the last formula of the quantify. I will explain that in the nest two paragraph and give an improved formula.

So, given that situation, we need to find one or several baseline, so that we can have something to compare with. I will use leave one out cross validation idea here (not exactly same).

Every time, we should take one test out of all tests as the baseline and compare it with other tests. How do we compare them? So, if we have one baseline result (the input should be all same, so we don't need to write that down): A-B-C-D-E-F-G, and we have another test: A-B-C-Z-E-S-O. It is obvious that the first three results are same and at position four, it has non-deterministic property. We will record the position 4 into the record list[]. So in a 100 times test set,

there will be 100 positions to be recorded (The times of test should be much much larger than the length of the test sequence). We make statistics and will get the frequency of positions showed. If we plug in the corresponding frequency into information entropy formula, we can get many values to represent the level of non-deterministic. However, how can let them represent the corresponding positions? We cannot just add those values together as that might cause mistakes. For example, in test one, the record is [6,6,6,6,6,6.....1], that contains 99 six and 1 one. The information entropy is same with [1,1,1,1.....6] (99 one and 1 six). However, the first test's level of non-deterministic, of sure, is lower than the second one as in the most of situations in the second test, non-deterministic happen in the first position. So, give the corresponding entropy value the position information is of great significant. And we hope the larger the position is, the smaller the information entropy value will be.

So, let us set the logic here. If ND happened at front position, we want ND to be large. If ND happened at the end, it means all the previous nodes are same. We want the ND here to be small. As information entropy is related to ND, it will increase/decrease with ND. So we have:

$$Position(front) == parameterinentropy(big) == entropy(small) == ND(small)$$

$$Position(closetoend) == parameterinentropy(small) == entropy(big) == ND(big)$$

In this case, by following the logic, the key things we need to break out is the ND happened position as this is the only difference between test. Let us think in this way, if ND happened at third position in the Test sequence, that means first two are same and things after the third we do not need to consider. If there is a probability p that in the corresponding position compared with baseline, it is different, we say the chance not changing is 1-p. If only first two are same, so here we set all things behind are all not same as we all know third one is ND.

$$parameter = (\frac{1-p}{p})^2 * p^{n-2}$$

Here, if position is i, we use other positions to replace 2. And p can also be replaced by other numbers. At first, my thought was, since we will have change or not change, p can be 0.5 to represent two situations. But the idea will cause unbalanced as the result is not binary, instead, it should be multiple result. So here we can use all data we test to see the result frequency f_{i1}, f_{i2}, f_{i3}, etc (exclude baseline corresponding result), and use that to replace term p. So we will have:

$$parameter = (\prod_{y=1}^{i-1} \frac{1-f_{y_{base}}}{f_{y_{base}}}) * f_i * \prod_{x=i+1}^{n} f_x$$

The following formula might help to understand what does the formula shown above mean.

$$parameter = P(same-with-baseline) * P(ND-position) * P(items-dont-consider)$$

Note that f_{ybase} is the frequency of corresponding result same to the baseline and f_x is the randomly pick from frequency and f_i the corresponding frequency in $(f_{i1}, f_{i2}, f_{i3}), etc.$ And as in a sequence, previous results will affect other results. So we should consider bayes.

$$parameter = \frac{(\prod_{y=1}^{i-1} \frac{1-f_{y_{base}}}{f_{y_{base}}}) * f_i * \prod_{x=i+1}^{n} f_x}{(\prod_{y=1}^{i-1} \frac{1-f_{y_{base}}}{f_{y_{base}}})}$$

$$= f_i * \prod_{x=i+1}^{n} f_x$$

Then, we can know the corresponding ND (we can also say M3 or adapted entropy)

$$ND(baseline_j) = \sum_{i=1}^{lengthoftest} -parameter * (\log parameter)$$

where p is the frequency of appearance (like, 20 times in 100 sequence, that is 20/100). And since we have n baselines (based on the times of test), we will calculate the mean of non-deterministic.

$$ND(M3) = \frac{1}{times - 1} \sum_{j=1}^{times} ND(baseline_j)$$

3.2 Application

Based on the description in the midterm write up, what we know that for the application, there are something we need to know at first.

First, the application itself is a black box. The only thing we can know is that the input and output.

Second, both input and output should be in the same category set. In the example in write up, both input and output are in the color set. If input and output are not the same kind of things, we cannot use that as an example.

Third, input and output should be one on one mapping. That does not mean one input can only have one corresponding output. On the opposite, same input can have many possible outputs or same outputs can be from different inputs. But overall, every time we put in one input, we can only get one output.

After summarizing some rules above, some applications can be selected. I will take Gene mutation as an example (We can also use C program to do the simulation). In my first version, my application was about Enigma code machine, but after considering the value of ND, if can be very very huge (one single input

A can have 26 and one sequence input will have billions of possible results)

This is a very fun application. From a molecular level, gene mutation means that changes in the base pair composition or arrangement sequence of a gene in the gene structure. Although genes are very stable (this is actually most of time and situations, if it reproduces naturally) and can accurately replicate themselves during cell division, this is supposed to be relative stability. And Under certain conditions, a gene can also suddenly change from its original form of existence to another new form of existence, that is, at one site, a new gene suddenly appears to replace the original gene. This gene is called a mutant gene. Therefore, new traits never existed in the ancestors suddenly appeared in the performance of the offspring.

If you are very familiar with biology (or in some areas of genetics), then several forms of genetic mutations should be considered common sense. Of course, I am not an expert in this area, but through more than a week of reading materials and reading comprehension, I have a general understanding of several gene mutations. In general, gene mutations are divided into Base substitution mutation and frame shift mutation. The first kind of meaning refers to the mutation caused by the substitution of a base pair by a different base pair in a DNA molecule, also called a point mutation. The second meaning refers to a mutation that causes a series of coding sequences after the insertion or loss site to be misaligned when one or several base pairs are inserted or lost at a certain site in a DNA fragment. In our application, we mainly consider the first kind of mutation, that is, base substitution mutation. The reason is that, if we consider the second situation, there might have some loss, which cause the length will be different.

For this application, we want to use the algorithm to test how can one enzyme (some reaction reaction) affect some specific DNA snippet. The function of enzyme can be seen as a black box (combine with reacting environment like hp, temperature, etc). For convenience, we can replace some snippet terms into ABCD. If we do not use enzyme, the DNA will generate by itself and the corresponding postition will always be the same (if A, forever A). And using enzyme, the result might be different.

Here are some details and steps about the application. First we set the standard environment in the reacting machine and every time put one single simple into it to see the reaction result. For one single test, we will use, for example, 20 samples (given they are generated by same cell/virus, so RNA/DNA will be the same) Here is one DNA simple:

$$===== A ======$$

And after reaction, the DNA in this specific position turn to be:

$$===== B ======$$

We may say that the enzyme has non-directional effect on this DNA cut. And next time, we input the same specific position but in this simple, another DNA cut showed in the same position:

$$=====C======$$

And after reaction, no change.

$$=====C======$$

Overall, we have two pairs of data (A,B) and (C,C). And after the test, we can get many pairs:

$$T1 : (A, B)(C, C)(B, C)(A, C)(A, A)(D, D)(D, A)(C, B)...$$

Note that we can also put one same simple several times to see whether it will go on reacting (result will also count as one pair in this sequence test).

$$T2 : (A, B)(C, C)(B, A)(A, A)(A, A)(D, C)(D, A)(C, B)...$$

$$T3 : (A, B)(C, C)(B, C)(A, C)(A, A)(D, D)(D, D)(C, B)...$$

$$UntilT_n$$

Then, what we need to do is following the method described in Section 3.1 to get the M3.

Some argues: How the application satisfy the rules? First, since people is studying on this enzyme, we have no idea about the function of it (just know that it might have some effect on this DNA cut). Based on that, it is a black box. Second, input and output are in the same set ABCD (simplified DNA cut set). Third, it is one on one mapping. And also, every reaction in one test is dependent as after doing a reaction, the environment we built must have changed. Enzyme concentration (where the concentration of the enzyme is not necessarily the higher the better, there will be certain low fluctuations in the efficiency of the enzyme), the temperature caused by the rupture of the DNA fragment, and the pH value of the solution after the reaction will all undergo non-directional changes, some free ABCD (DNA cut) in solution. These changes may directly affect the results of subsequent experiments. Every time we start a new test, we will reset the environment to the initial state.

Next we will talk about the strength and weakness.

We all know that if we use enzyme to trigger gene mutation, we cannot make sure what will happen for sure. And in some cases, there is even no changes (It always happen). After some black box test, what we want to know is the

level of uncertainty. If for some cut, there is a stable transfer from A to B (or stable no transfer, no change), we say the enzyme has a stable and solid effect on DNA cut given A. Once we figure out the value of every enzyme, we can take a rank of them, so that we can figure out which one is more stable and which is more non-deterministic. We might not want to use an enzyme with much non-deterministic in terms of making some medical treatment or medicine. People will never take the treatment when the medicine has an uncertain affect. Maybe he will have a chance to die. It is hard to say. So by using the method I wrote in Section 3.1, we can easily figure out the quantity value.

Even is the same term non-deterministic, this can be different from Question one and Question two as this is a black box test. The only things we know are the input and output. Counting path may not be a good idea here as you have no path to count. By using the method, we can figure out a no path counting value, the method is not an integer, instead, it is a value that can represent how mess the results are. Compared with path counting, when data is not large, path counting can be a very good method. Since there are two dimensions here (length of test, number of test), once we add one, there might be an exponential growth of value, which means if we use that method, the running time will be very long.

The strength of information entropy I have mentioned in Section 3.1. Other strength of technical aspect is that, the method hires leave one out cross validation to set the baseline and make comparison. Why do not I use frequency to get baseline? Because, the pairs of results have some conditional aspect, if pairs at the end of sequence are always affected by the previous tests. An other reason, what if, in a very special situation, the frequency for some small probability event (pairs) can be large and then become one part of the baseline then. What is more, the binary case can be more likely to be the part of the baseline rather than multi category case as the they have more chance of appearance. Anyway, by using LOOCV, there is a balance to keep it from unbalanced baseline. Also, the method I used can reflect the position information given a banch of sequence. We only calculate the probability after ND. By doing so, we give information entropy the position information. It means the level of mess given the first several items are not mess.

There are also some weakness in terms of this application. For example, when we do the same test, we must make sure all the simple are generated by the same parent cell/virus. There might be a very very small chance for them to have gene mutation naturally. As we need to make sure each test should have the same samples, this might be one barrier. Another thing is that, as I mentioned in the beginning, we only consider no DNA cut will lose case. What if the enzyme lead to the lost of some DNA cut? That is hard to say for us as all things are not certain.

Technically, some details are ignored in Section 3.1. The dependency is one of

them. Recall how we get rid of bayes, we cancel all the items before f_i in that formula. Actually, there might be some intersection when I want to express the numerator but so far, as time is so compact and the problems are hard, I haven't figure out one way to get rid of that wisely. But, overall, the value is positive related as we can turn the unaviodable part as a constant α while α is a positive number.

Further more, here are some other ideas about the application. If time is enough, I will think about how to combine path counting and information entropy together. I am not sure using breadth-first search is good or not (Tree need to be pruned so that we do not consider the result after ND happened). If possible, I think I can give build a BFS tree at first and then give every branch one corresponding weight. By using that weight, it is easy to calculate the information entropy. And we can add them together to get the final M3. Also, information maybe can be replaced by variance of positions.