COMPUTATIONAL COMPLEXITY

GRADO EN INGENIERÍA INFORMÁTICA

PRÁCTICA 1

AUTOR: TEODORO MARTÍNEZ MÁRQUEZ

Cádiz, 24 de Marzo de 2019

1. INTRODUCTION

In this practice I'm going to create a Turing Machine and a program following the RAM model for two decision problems, and then study their time complexities.

The first problem is the palindrome problem: given a number in binary format, check if said number is a palindrome. A palindrome number is a number which is read the same from left to right and from right to left. This algorithm only requires n/2 comparisons in the worst case, n being the length of the input. However, the real time complexity accounts all operations, which will be higher than this in both models, especially the single-tape TM.

The second problem is the equality problem: given two numbers in binary format of the same length, determine if they're equal. This algorithm requires n comparisons in the worst case, n being the length of either number. But again, the number of actual operations will surpass this.

2. METHODOLOGY

Let us start with the turing machines:

- **Turing Machine for the palindrome problem**

The concept of the algorithm is simple: read the first digit and remove it, then go to the last digit and check if it matches the first one. If it matches, remove the last digit and read the second-last digit, go back to the new first, former second digit of the number, check if it matches, and repeat the whole process. If at any point a pair of digit doesn't match, the algorithm ends in a "N" state, representing a negative or '0' output. If the whole number is deleted, then the algorithm will end in a "Y" state representing a positive or '1' output.

The purpose of each state is as follows:

- 0: Read digit from left-end, remove it and move to the right.
- 1: Reach the right-end of the number, a 1 has been read.
- 2: Reach the right-end of the number, a 0 has been read.
- 3: Read digit at the right-end of the number. If it matches 1, remove it and continue. Otherwise, output is negative.
- 4: Read digit at the right-end of the number. If it matches 0, remove it and continue. Otherwise, output is negative.
- 5: Read digit from right-end, remove it and move to the left.
- 6: Reach the left-end of the number, a 0 has been read.
- 7: Reach the left-end of the number, a 1 has been read.
- 8: Read digit at the left-end of the number. If it matches 0, remove it and continue. Otherwise, output is negative.
- 9: Read digit at the left-end of the number. If it matches 1, remove it and continue. Otherwise, output is negative.
- 98: Final state, positive output.
- 99: Final state, negative output.

Notice that states 0-4 have near identical functions to 5-9. Both sets do the same sequence, but one does it left to right, and the other right to left. States 0,3,4,5,8 and 9 will return a positive output if they find a blank or read cell, as it means the number has been processed completely.

- **Turing Machine for the equality problem**

The concept for this TM follows a similar scheme to the palindrome problem: Read and remove first digit of first number, read, remove and compare the first digit of the second number. If they match, read and remove the next digit of the second number, go back to the next digit of the first number to read, remove and compare it, rinse and repeat.

This time, the act of 'removing' doesn't actually leave the cell blank, instead it rewrites it with a symbol 'X' to mark it as read.

- 0: Read and remove first digit of first number.
- 1: Reach the second number, a one has been read.
- 2: Reach the second number, a zero has been read.
- 3: Read the first digit of the second number. If it matches one, remove it and continue. Otherwise output is negative.
- 4: Read first digit of the second number. If it matches zero, remove it and continue. Otherwise output is negative.
- 5: Read and remove next digit from second number and move to the left.
- 6: Reach the first number, a zero has been read.
- 7: Reach the first number, a one has been read.
- 8: Reach the next digit of first number, a zero has been read.
- 9: Reach the next digit of first number, a one has been read.
- 10: Read the next digit of first number. If it matches zero, remove it and continue. Otherwise, output is negative.
- 11: Read the next digit of first number. If it matches one, remove it and continue. Otherwise, output is negative.
- 98: Final state, positive output.
- 99: Final state, negative output.

Similarly to the previous TM, this one will return a positive output when both numbers have been fully removed. That is, when states 0 and 5 don't find a digit. Since it is assumed that both numbers are the same length, only these two states which start reading a new pair are needed to fulfill the requirements. The algorithm will return a negative output if the numbers aren't the same length, in any case.

- **RAM model program for the palindrome problem**

The procedure of this program is as follows: First run through the whole number to get a pointer, 'i', to its end and the value of half of its length rounded down, 'm'. Then get another pointer, 'j', to its beginning. Finally, start a loop: compare the values of both pointers. If they match, decrease i and increase j by one, otherwise finish with a negative output. Repeat the process m times. If no iteration has broken the loop, return a positive output.

The program uses six registers. The register 0 is left untouched to do unconditional jumps. The first loop uses registers 1 to 4. Register 1 represents the 'i' pointer, reaching the end of the input. Register 2 represents the 'm' value, as 'i' is running through the number, m will increase its value every two iterations. Register 3 represents the 'a' variable, which contains the value pointed by 'i', and register 4 is an auxiliar variable, in each iteration it goes from 0 to -1 or viceversa, it is used to correctly calculate m.

Before we transition to the second loop, i and m are decreased by 1, so that i points exactly to the last digit, and m doesn't count an extra iteration (since our only condition possible is >=0, not >0). For the second loop, register 3 will be repurposed to be the second pointer, 'j', starting at 1. Register 4 will also be repurposed to store the values pointed by 'i', variable 'a', and register 5 will store the values pointed by 'j', variable 'b'.

In each iteration, the operations are the following: load values a,b and compare them. If they're unequal, end with a negative output. If they're equal, move to next comparison. If there is no next comparison (m < 0), end with positive output.

- **RAM model program for the equality problem**

This time, the procedure is quite simpler than the palindrome problem. First, get a pointer, 'i', to point to the first digit of the second number (running through the entire first number). Then, a second pointer, 'j', to point to the first digit of the first number. Finally, advance both of them at the same time, comparing the values they point to every step.

The first loop only uses two registers. Register 1 represents the pointer 'i', and register 2 the value pointed by it, 'a'.

For the second loop, register 2 is reset to value 1, representing the second pointer 'j'. Register 3 will be the new 'a', and register 4 will be its counterpart, 'b'. It follows the same structure as the palindrome problem: Compare 'a' and 'b', if they're equal increase pointers, if not end the loop with negative output. There is only a little difference: the way to get out of the loop with a positive output is that 'a' has a value less than zero, in which case the end of the number has been reached.

3. RESULTS

Every time complexity discussion will always assume the worst case scenario.

- **Time Complexity of comparison problem: Turing Machine**

The Turing Machine has the following time complexity:

$$O = (n + 2) * (n) + n - (n \; mod \; 2)$$

Where 'n' is the length of the input. The formula can be simplified like this:

$$O = (n + 3) * (n) - (n \; mod \; 2)$$

However, I'll use the first formula to describe how it was calculated.

First of all, I calculated the time that it takes the program to compare one pair of digits. If the comparison is from left to right, it takes n+2 steps, if it's right to left it takes n+4 steps. In total, the program will always do n comparisons, so assuming that every comparison is at least n+2 steps, the TM would take (n+2)*n steps. Now, we have to

account for the extra two steps of every right to left comparison. Since the TM always starts with a left to right comparison, the number of right to left comparisons is n/2 rounded down. 2*(n/2) = n extra steps to add. However, since that n/2 is originall rounded down, the formula would count one extra step for odd n values, so I include – (n mod 2) to compensate.

This time complexity is aproximately half of the complexity that it would take if every comparison was left to right, since the TM would need to go back to the beggining of the tape after every comparison. Regardless, it belongs to the order of n quadratic, due to the limitations of a single tape TM.

- **Time Complexity of comparison problem: RAM model**

The RAM model program has the following time complexity:

$$O = 3 * (\mathrm{n} + 1) + 3 + 3 * (n + 1) + n * 6 + 5$$

Simplifying the formula:

$$O = 12 * \mathrm{n} + 14$$

Where n is the length of one of the numbers. In reality, the actual length of the input would be 2*n+1, but I will use just n for simplicity.

Explaining each part in detail: 3*(n+1) comes from the first loop to obtain the pointer for the second number, as it's three instructions done n+1 times (it has to go through the full first number and the separation symbol). Then, the preparation for the second loop takes three constant instructions (+3), to put both pointers at their correct spots.

The second loop is a bit more complex, I'll divide it into two parts. The first part involves just loading the values of both pointers and verifying that there are still digits to be read. This takes three instructions performed a total of n+1 times (one for every pair for digits and a last one to end the loop), so 3*(n+1). The second part involves comparing both digits and responding accordingly. For the worst case, the digits will always be equal. And for this program specifically, this part takes one extra step if both digits are zero, with a total of six steps every iteration. The second part is only done n times, so n*6.

Lastly, the steps taken after getting out of the loop: that's one to skip the loop instructions and four more to store a 1 as the output, so a total of 5.

- **Time complexity of palindrome problem: Turing Machine**

The Turing Machine has some similarities with the comparison one, since it also has to go back and forth through the input, and it also does left to right and right to left comparisons. However, this time the input is completely symetrical, which means that all pair comparisons take the same time regardless of direction.

The time that one comparison takes is also n+2, but this time each comparison is actually shrinking the length of the input, meaning that each consecutive comparison will take two less steps from n+2, n, n-2, until there is no input left. For even lengths, the last comparison has only two digits left and takes four steps plus one final step to return positive output. For odd lengths, the last comparison has three digits left, taking five steps and three extra steps to read the remaining digit and return a positive output.

So the formula follows an arithmetic proggresion which starts at 4 for even lengths and 5 for odd lengths, with a difference of 2, and ends at n+2. Such progression is then equivalent to:

$$\left(\lceil\frac{n}{2}\rceil + 3\right) * \lfloor\frac{n}{2}\rfloor$$

Lastly, when I add the extra last steps to return the positive output, the complexity would be as follows:

$$O = \left(\lceil\frac{n}{2}\rceil + 3\right) * \lfloor\frac{n}{2}\rfloor + 1 + 2 * (n \bmod 2)$$

As you can see, this order also belongs to n quadratic. However, the acts of shrinking the input length and doing comparisons in both directions have managed to reduce the complexity significantly, which otherwise would be around n^2 as opposed to the average n^2/4 of this TM.

- **Time complexity of palindrome problem: RAM model**

For the RAM model program, the complexity will be analyzed in parts. First, the first loop to obtain the length and end pointer for the input:

$$O1 = (n + 1) * 6 - ((n + 1) \bmod 2)$$

The first loop does a total of n+1 iterations, where the odd iterations have 5 instructions, and the even iterations have 7 instructions. In average, all the iterations would have six instrucctions, but since the loop starts with an odd iteration, we need to discount one instrucction that would come from the loop also ending with an odd iteration, so discount one step for even lengths n.

Then, we have four constant instructions before we start the second loop. The complexity of the second loop plus all the constant instructions is:

$$O2 = 4 + 11 * \lfloor\frac{n}{2}\rfloor + 4$$

The loop has 11 instructions in worst case (both digits are zero), and it has the minimum amount of comparisons for a palindrome problem, being n/2 rounded down. Lastly, it has four final instructions to return a positive output.

So, the final time complexity of the problem will be:

$$O = 11 * \lfloor\frac{n}{2}\rfloor + 6 * n + 14 - ((n + 1) \bmod 2)$$

This complexity belongs to the linear order, which is less than its Turing Machine counterpart.

4. CONCLUSIONS

We observe that the theoretical time that both problems present is linear. The RAM models are able to get a linear time as well, but the Turing Machines are both quadratic. This is because both problems require digits from different places of the input to be read at the same time to be compared. The RAM model is capable of managing multiple pointers at once, so it can achieve a linear complexity. However, the Turing Machines are single-tapes, which means they only have one pointer and can only read one cell at a time. Since they can't store memory in a FIFO stack or similar either, they have no option but to move back and forth through the input to compare each pair of digits, resulting in the quadratic time that we see in the third point.