CS3052 Practical 1 Report

140015533

28 February 2017

Contents

1 Overview

2 Turing Machine Simulation

3 Solutions to Problems

3.1 Palindromes

The Turing machine for recognising palindromes goes through the input from one side to the other comparing each half of the input with the other. It stops if there is a mismatch in one letter.

1. If there are no letters on the tape, accept.
2. Read a letter, remember it and mark it.
3. Go right until the end of the tape, or until a marked letter is found.
4. Go one cell left and check if the letter matches the remembered letter.
    1. If it does, mark it.
    2. If not, reject.
5. Go left until a marked letter is found.
6. Go one cell right and go back to 2.
7. Repeat while there are unmarked letters. If there are only marked letters, accept.

3.2 Binary Addition

3.3 Insertion Sort

3.4 Substring

3.5 Divisible by 4

4 Analysis

I have created a number of series of inputs for each solution. The inputs in a series grow linearly and all have a common characteristic (e.g. all are palindromes). These series are generated procedurally using Python generators.

Then I let the machine run on first 100 inputs in each series and counted the number of transitions the machine makes for each input. The output of the analysis program is a CSV file, with a column for each series and a row for each analysed input.

For example, for the palindromes solution the output file begins like this:

```
n,input 0,input 1,input 2
0,4,4,3
1,5,5,6
2,6,6,10
3,7,7,15
4,8,8,21
5,9,9,28
```

The first column, `n` is the position of the analysed input in the series. The other numerical values are numbers of transitions.

4.1 Palindromes

In the solution the machine needs to go back and forth through the tape for each letter in the first half of the input.

This means that if the input is a palindrome, the program has time complexity $O(n^2)$, as this is the worst case - the head needs to go through the whole tape twice ($2n$ transitions) for each letter in a half of the input ($\frac{n}{2}$ transitions). We can see this in Figure 2.

The best case is when the input is not a palindrome, and the first and last letters do not match. In this case the machine only goes through the whole tape once, resulting in $O(n)$ time complexity, as shown in Figure 1.

The series of inputs used for measurements are:

- Input 0: ab, aab, aabb, aaabb, ...
- Input 1: ab, aab, aaab, aaaab, ...
- Input 3: a, aa, aaa, aaaa, ...

Inputs 0 and 1 are both not palindromes, and the first and last letters do not match, resulting in the $O(n)$ time complexity in Figure 1.

Input 3 is a palindrome, which is the worst case for the algorithm.

4.2 Binary Addition

4.3 Insertion Sort

4.4 Substring

4.5 Divisible by 4

5 Conclusion

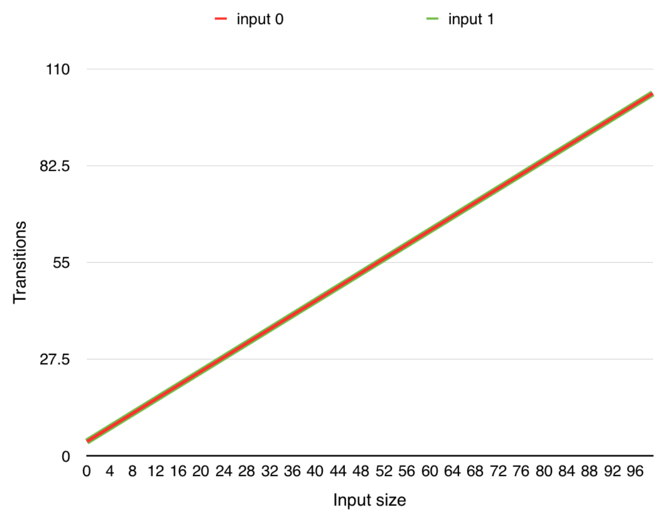

Figure 1: Best case for the palindrome TM

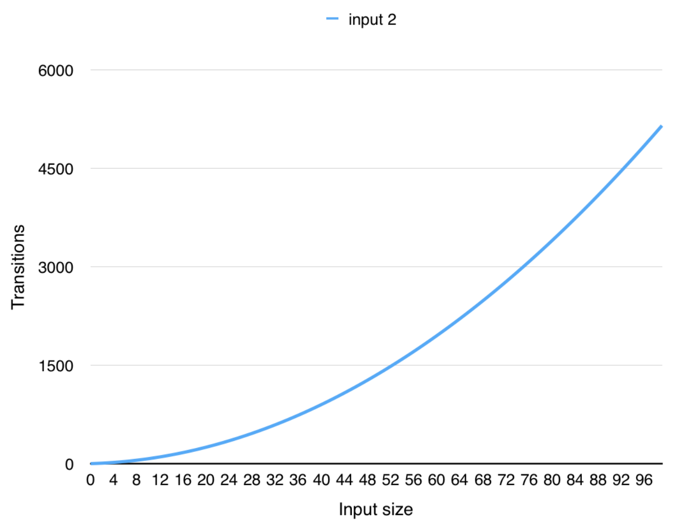

Figure 2: Worst case for the palindrome TM