Formal Languages and Computability 10

Ragnar Björn Ingvarsson, rbi3

5. nóvember 2024

1

We see that the big O of $f(n)$ is $O(n^2)$.

a) This grows slower than n^2 so this is true.

b) 2^n grows faster than n^2 so this is false.

c) n^4 grows faster than n^2 so this is false.

d) We see that this depends on whether $log_2 n^n$ grows faster than n, and we can rewrite $log_2 n^n$ as $n + log_2 \frac{n}{2}^n$ which we can see grows faster than n so this is true.

e) Since 2^n grows faster than n^2 is this true.

So a), d) and e) are true.

2

a) We see that each revolution takes $n + n + 2 = 2n + 2$ steps, and the number of revolutions is $\frac{n}{4}$ since if the length of the string is n, then the number of dibits is $\frac{n}{2}$ and then the number of pairs of dibits is $\frac{n}{4}$. Then we just subtract the final rewind step since that isnt executed on the final step and we get $\frac{n}{4}(2n+2) - n = \frac{1}{2}(n^2 - n)$.

b) No, since we get

$$\lim_{n\to\infty} \frac{n^2}{1/2(n^2-n)} = \lim_{n\to\infty} \frac{n^2}{n^2(1/2 - 1/2n)}$$

$$= \lim_{n\to\infty} \frac{1}{1/2 - 1/2n} = \frac{1}{1/2} = 2 \neq 0$$

So n^2 grows at a faster pace than $\frac{1}{2}(n^2 - n)$.

c) A faster version of this algorithm would be to have it blank out the left side pairs as soon as we begin the search for a pair, so we travel back and forth the tape with an average of only n steps, making the exact running time $\frac{n}{4}(n+2) - \frac{n}{2} = {}^{n^2}/_4$

3

a) Tabulating the times for $n = 1, ..., 7$ we get

Length	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Steps	2	3	6	13	28	59	122

b) We simply experiment enough until we get the formula

$$r(n+1) = 2r(n) + n - 2 \tag{1}$$

c) To find the exact closed form expression we will use the formula for a linear first order difference equation of the form $x(n+1) = a(n)x(n) + c(n)$,

$$y(n) = \left(\prod_{k=1}^{n-1} a(k)\right) y_0 + \sum_{k=1}^{n-1} \left(\prod_{j=k+1}^{n-1} a(j)\right) c(k) \tag{2}$$

Where we get

$$r(n) = \left(\prod_{k=1}^{n-1} 2\right) \cdot 2 + \sum_{k=1}^{n-1} \left(\prod_{j=k+1}^{n-1} 2\right) \cdot (k-2)$$

$$= 2^n + \sum_{k=1}^{n-1} k2^{n-k-1} - 2^{n-k} = 2^n + 2^{n-1} \sum_{k=1}^{n-1} \frac{k}{2^k} - 2^n \sum_{k=1}^{n-1} \frac{1}{2^k}$$

$$= 2^n - n + 1$$

So the closed form is $r(n) = 2^n - n + 1$.

4

So the algorithm goes like this, we first mark one node, then, we iterate through each node in G to check if it is connected to any marked node and if so, we mark it. We then continue until no new nodes are marked. At last we iterate through each node and check if any remain unmarked, if so we reject otherwise we accept.

Looking at the worst (not even possible) case of this, we get that we have to iterate through every node (n times) and for each node we assume that they are connected to every other node, meaning we check $n - 1$ paths for each node. We then do all this a maximum of $n - 1$ times since at least one node must be marked every time, and then one more time to confirm everything which leaves it at n times.

We then get that the theoretical maximum amount of steps for this is $n * (n-1) * n = n^3 - n^2$, which is clearly in P.