CS525 Winter 2012 \ Chapter #7 Preparation

Ariel Stolerman

1)

a. $\underline{t^+(n)\sim n}$:

The main loop of the algorithm will run at most n steps – in the case the carry (c) is 1 in all iterations, and eventually creates a new digit $a_n = 1$. In this case, the initialization takes some constant time c_1, the main loop takes at most $c_2 \cdot n$ time for some constant c_2, and the last step (at line 10) takes some constant time c_3. Therefore the total maximum running time for an input a in SLM-representation with n digits is $c_1 + c_2 \cdot n + c_3 \leq (c_1 + c_2 + c_3)n \underset{c \coloneqq c_1+c_2+c_3}{=} c \cdot n$ where c is a constant. Therefore $\forall n \in \mathbb{N}: t^+(n) \leq c \cdot n$ for the c above, therefore $t^+(n) \preccurlyeq n$.

For the other direction, in worst case the loop runs n iterations. Using the same notation as above:

$$t^+(n) = c_1 + c_2 n + c_3 \geq 1 + n + 1 \geq n$$

Therefore $\forall n \in \mathbb{N}: n \leq c \cdot n \leq c \cdot t^+(n)$ for $c = 1$, and so $n \preccurlyeq t^+(n)$.

In conclusion, $t^+(n)\sim n$, as required.

b. $\underline{t^-(n)\sim 1}$:

In the best case scenario (the minimum running time), the addition does not cause a carry that needs to be transferred forward. In this case the initialization will take some constant time c_1, the loop will run 1 iteration only, that will take in total some constant time c_2 (lines 3 through 9 – including the condition check), and the last line will take some constant time c_3. Let $c \coloneqq c_1 + c_2 + c_3$, then $\forall n \in \mathbb{N}: t^-(n) \leq c \cdot 1 \Rightarrow t^-(n) \preccurlyeq 1$. The opposite direction works by choosing $\frac{1}{c}$ as the constant. Therefore $t^-(n)\sim 1$, as required.

c. $\underline{t^*(n)\sim 1}$:

To calculate the number of steps it would take in average, we partition the space of n-length inputs of β-base represented numbers (approx. between 0 and $\beta^n - 1$).

For each $1 \leq m \leq n$, the chance the carry will yield m iterations of the main loop is:

The chance the first $m-1$ digits are $\beta - 1$ and the mth digit is $< \beta - 1 = \left(\frac{1}{\beta}\right)^m \cdot \frac{\beta-1}{\beta}$

Using the notations of the previous sections (c_1, c_2 and c_3), the averaged running time for inputs of size n is:

$$c_1 + \sum_{m=1}^{n} \underbrace{c_2 m}_{cost} \cdot \underbrace{\left(\frac{1}{\beta}\right)^m \cdot \frac{\beta-1}{\beta}}_{probability} + c_3 = c_1 + c_2 \frac{(\beta-1)}{\beta} \sum_{m=1}^{n} \frac{m}{\beta^m} + c_3 \leq (*)$$

Apply convergence test on $\sum_{m=1}^{\infty} \frac{m}{\beta^m}$: $\frac{a_{m+1}}{a_m} = \frac{m+1}{\beta^{m+1}} \cdot \frac{\beta^m}{m} = \frac{1}{\beta}\frac{m+1}{m} \Rightarrow \lim_{m\to\infty} \frac{a_{m+1}}{a_m} = \frac{1}{\beta} < 1, \forall \beta \geq 2 \Rightarrow \sum_{m=1}^{\infty} \frac{m}{\beta^m}$ converges, and therefore the sum $\sum_{m=1}^{n} \frac{m}{\beta^m} \leq c_4$ for all $n \in \mathbb{N}$ for some constant c_4. Therefore:

$(*) \leq c_1 + c_2 \frac{(\beta-1)}{\beta} \cdot c_4 + c_3$, which is a constant (for any $\beta \geq 2$). Denote that constant c, then $\forall n \in \mathbb{N}: t^*(n) \leq c \cdot 1$, therefore $t^*(n) \preccurlyeq 1$. The opposite is trivial: $\forall n \in \mathbb{N}: 1 \leq d \cdot t^*(n)$, where $d = 1$ suffices.

In conclusion, $t^*(n) \sim 1$, as required.

7.6)

Following are proofs of closure properties of P:

Let $L_1, L_2 \in P$ be two poly-time decidable languages, then there exist two polynomials p_1, p_2 such that L_1 is decidable by an $O(p_1(n))$ time Turing machine denoted M_1, and L_2 is decidable by an $O(p_2(n))$ Turing machine denoted M_2, for any input of size n.

Union:

Let $L \coloneqq L_1 \cup L_2$ be the union of both languages above, then we can define a Turing machine M as follows:

"For input w:

- Simulate M_1 on w. If it accepts, *accept*.
- Otherwise, simulate M_2 on w. If it accepts, *accept*.
- Otherwise, *reject*."

Clearly M is a decider for L, and the time it takes is at most the time it would take M_1, followed by the time it would take M_2, plus some polynomial-time operations (e.g. for resetting the input tape). Therefore the total running time for any input of size n is $O(p_1(n) + p_2(n) + p_3(n))$, where p_3 is the polynomial for the additional operations. Since addition of polynomials is a polynomial itself, then L is decidable by a polynomial-time Turing machine, an so $L \in P$. Therefore P is closed under union.

Concatenation:

Let $L \coloneqq L_1 \circ L_2 = \{uv \mid u \in L_1, v \in L_2\}$ be the concatenation of both languages above, then we can define a Turing machine M as follows:

"For input w:

- For any partition of w into two consecutive parts w_1, w_2 (where $|w_1|, |w_2| = 0, \ldots, n$):
  - Simulate M_1 on w_1 and M_2 on w_2.
  - If both accept, *accept*. Otherwise, continue.
- If did not accept for any of the partitions, *reject*."

Clearly M is a decider for L, as it checks all possible partitions of w into two concatenated strings such that the first is in L_1 and the second in L_2. Each iteration takes $O(p_1(n_1) + p_2(n_2))$ where $n_1 + n_2 = n$, therefore each step takes $O(p_1(n) + p_2(n))$; Also, there are $O(n)$ partitions to check, therefore the total is $O\left(n \cdot (p_1(n) + p_2(n))\right)$, which is a polynomial itself. Therefore L is decidable by a polynomial-time Turing machine and so $L \in P$. Therefore P is closed under concatenation.

Complement:

Let $L = \overline{L_1}$ the complement of L_1, then we can define a Turing machine M as follows:

"For input w:

- Simulate M_1 on w.
- If it accepts, *reject*. Otherwise, *accept*."

Clearly M decides L, and it takes $O(p_1(n))$ time (we can use M_1 itself, only substituting transitions to the accepting state by transitions to the rejecting state and vice-versa), therefore $L \in P$ as well. Therefore P is closed under complement.

7.8)

Let $CONNECTED = \{\langle G \rangle \mid G \text{ is a connected undirected graph}\}$. Following is an analysis of the algorithm in page 157 showing that $CONNECTED \in P$.

1. Finding the start node of G and marking it can take $O(n)$ time.
2. The loop in the second stage can have $O(n)$ repetitions, because as long as there are unmarked nodes, each iteration marks at least one unmarked node.
3. For each iteration of the loop above, all $O(n)$ nodes need to be checked for adjacency with the checked node, and each check takes $O(n^2)$ – going over all edges. The total running time of each iteration is therefore $O(n^3)$.
4. The last step that scans all nodes and checks if they are all marked takes $O(n)$.

The total running time is therefore $O(n + n \cdot n^3 + n) = O(n^4)$, therefore $CONNECTED \in P$.

7.9)

Let $TRIANGLE = \{\langle G \rangle \mid G \text{ is an undirected graph that contains a triangle}\}$. Following is a proof that $TRIANGLE \in P$:

Let M be a TM defined as follows:

"For input G:

- Make sure G is a proper encoding of an undirected graph, otherwise *reject*.
- Enumerate over all possible triplets of nodes in G, and check if all are connected to each other.
- If found such triplet, *accept*. Otherwise, *reject*."

Note: we can assume a multi-tape TM for simplicity of running-time analysis (i.e. not take into consideration going back and forth over a single tape), as it maintains "polynomiality".

Correctness:

Clearly if G contains a triangle, it will be found and the machine will accept, and if not – the machine will reject.

Running time:

- The first step can take $O(n^2)$ time, including copying the nodes an edges to different tapes, and scanning for correctness of both lists (no duplicate nodes, edges encoded over nodes correctly etc.).
- There are $\binom{n}{3} = O(n^3)$ possible triplets. Each triplets needs to be checked for 3 adjacencies (of each possible pair), each takes $O(n^2)$ time for traversing over all edges. The total running time of this step is therefore $O(n^5)$.

The total running time is $O(n^2 + n^5) = O(n^5)$, therefore $TRIANGLE \in P$.

7.10)

Following is a proof that $ALL_{DFA} = \{\langle D \rangle \mid D \text{ is a DFA and } L(D) = \Sigma^*\}$ is in P:

Let M_E be the Turing machine that decides E_{DFA} as described in the proof of theorem 4.4 (page 168). Let M be the following TM:

"For input D:

- Check that D is a proper encoding of a DFA, otherwise *reject*.
- Let D' be D with the accepting states marked as not-accepting and vice-versa.
- Simulate M_E on D'. If it accepts, *accept*. Otherwise, *reject*."

Correctness:

Clearly $L(D) = \Sigma^* \Leftrightarrow L(D') = \emptyset$, as switching accepting states with rejecting and vice-versa results with the complement language.

Running-time:

Note: we can assume a multi-tape TM for simplicity of running-time analysis (i.e. not take into consideration going back and forth over a single tape), as it maintains "polynomiality".

- Checking the input is a proper encoding of a DFA is $O(n)$ (or $O(n^2)$, depending on the way we check; either way – polynomial time).
- Reversing the type of state in D to generate D' can also be $O(n)$, if the states have a bit that describes whether they are accepting or rejecting (and then it can simply be reversed).
- Simulating M_E on D', using the TM described in theorem 4.4 is similar to exercise 7.8: marking the start state and then looping and marking all reachable states from marked states until no further changes are possible, followed by checking whether no accepting state is marked (reachable). Similarly to 7.8, this takes polynomial time in the length of the input (it can be shown to take $O(n^4)$).

The total running time is therefore $O(n^2 + n + n^4) = O(n^4)$, and so $ALL_{DFA} \in P$.

7.12

7.13)

Let $PP = \{\langle p, q, t \rangle \mid p = q^t \text{ where } p, q \text{ are permutations on } \{1, \dots, k\} \text{ and } t \text{ is a binary integer}\}$.

Following is a proof that $PP \in P$:

Let M be a TM defined as follows:

"On input $\langle p, q, t \rangle$:

- Let $q^1 \coloneqq q$. For $i = 2, \dots, \lceil \lg t \rceil$:
  - Calculate $q^i = q^{i-1} \circ q^{i-1}$, where $\circ$ is denoted composition
- Construct q' be the composition of all q^i for all i such that the ith bit of t is 1.
- Compare p and q'. If all transitions are equal, *accept*. Otherwise, *reject*."

Correctness:

We can calculate reach q^i from the previous q^{i-1} by composing the latter with itself, as it simulates doubling the number of steps. Eventually, since t is a binary number, we can simulate composing q for t times by composing all elements in the sum of powers of 2 that form t – namely all q^i where the ith bit of t is 1. Therefore the construction of $q' = q^t$ is correct. The rest is trivial.

Running time:

The loop that calculates all q^i runs $O(\lg t)$, and since t is a binary integer it is equal to $O(n)$. Each iteration can take $O(n^2)$, as for each element of the $O(n)$ elements in the permutation q^{i-1} we need to scan $O(n)$ transitions in itself in order to "double" the permutation. The total running time for constructing all q^i is therefore $O(n^3)$. Constructing q' takes similar time: calculating at most $O(n)$ compositions, each $O(n^2)$. The last step of comparing the permutations can be done in linear time (using multi-tape TM; $O(n^2)$ otherwise). The total running time is therefore $O(2n^3 + n^2) = O(n^3)$, an so $PP \in P$, as required.

7.14)

Following is a proof that P is closed under the star operation:

We describe a dynamic-programming algorithm. Let $A \in P$ be a polynomially-decidable language, and let M be a TM decider for A. Following is a description of a TM M^* which is a decider for A^*:

"On input w:

- Let n be the length of the input $w = w_1 w_2 \dots w_n$.
- Initialize a table of size $n \times n$, denoted S which will hold at each cell where $i \leq j$:

$$S_{ij} = \begin{cases} 1, & w_i w_{i+1} \dots w_j \in A^* \\ 0, & otherwise \end{cases}$$

- For $i = 1, \dots, n$:
  - Simulate M on w_{ii}. If it accepts, mark $S_{ij} = 1$.
  - Otherwise, mark $S_{ij} = 0$.
- For all the rest of the cells where $i \leq j$:
  - Simulate M on $w_i \dots w_j$. If it accepts, mark $S_{ij} = 1$.
  - Otherwise, for $k = i, i+1, \dots, j-1$:
    - If $S_{ik} = 1$ and $S_{(k+1)j} = 1$, mark $S_{ij} = 1$ and break the loop.
  - Otherwise, mark $S_{ij} = 0$
- If $S_{1n} = 1$, *accept*. Otherwise, *reject*."

Correctness:

The construction of the table is based on the fact that if $u, v \in A^*$ then $w = uv \in A^*$ as well. Each substring of w is checked for that or for membership in A (using the decider of A) – the only two possibilities in which $w \in A^*$. Therefore eventually S_{1n} will be 1 iff $w_1 \dots w_n = w \in A^*$.

Running time: The construction is clearly polynomial, therefore P is closed under the star operation.