Complexité avancée - Correction - Homework 1

Benjamin Bordais

September 23, 2020

Graph representation and why it does not matter. Let $\Sigma = \{\mathtt{0}, \mathtt{1}, /, \bullet, \#\}$ with $\#$ the end-of-word symbol. For a directed graph $G = (V, E)$ with $V = [0, n-1]$ for some $n \in \mathbb{N}$ and $E \subseteq V \times V$, we consider the following two representations of G by a word in Σ^*:

- By its adjacency matrix $m_G \in \Sigma^*$:

$$m_G \overset{\text{def}}{=} m_{0,0}m_{0,1} \ \ldots m_{0,n-1} \bullet \cdots \bullet m_{n-1,0} \ldots m_{n-1,n-1}\#$$

  where for all $0 \leq i, j < n$, $m_{i,j}$ is $\mathtt{1}$ if $(i,j) \in E$, $\mathtt{0}$ otherwise.

- By its adjacency list $l_G \in \Sigma^*$:

$$l_g \overset{\text{def}}{=} k_0^0/\ldots/k_{m_1}^0 \bullet \cdots \bullet k_0^{n-1}/\ldots/k_{m_{n-1}}^{n-1}\#$$

  where for all $0 \leq i < n$, $k_0^i, \ldots, k_{m_i}^i$ are binary words listing the (codes of) right neighbors of vertex i.

1. Show that it is possible to check in logarithmic space that a word $w \in \Sigma^*$ is a well-formed description of a graph (for any of the two representations).
2. Describe a logarithmic space bounded deterministic Turing machine taking as input a graph G, represented by its adjacency matrix, and computing the adjacency list representation of G.

Follow up: a new representation. With the same alphabet as in Exercise 1, we give a third representation of the graph $G = (V, E)$:

- By its lists of edges $e_G \in \Sigma^*$:

$$e_G \overset{\text{def}}{=} n \bullet v_{k_1}/v_{k_2} \bullet \cdots \bullet v_{k_{j-1}}/v_{k_j}\#$$

  where n is a unary word giving the number of vertices $|V|$, where for all $i = 1, \ldots, j$, v_{k_i} is a binary word denoting a vertex of V so that $E = \{(v_{k_1}, v_{k_2}), (v_{k_3}, v_{k_4}), \ldots, (v_{k_{j-1}}, v_{k_j})\}$.

1. Explain why n is part of e_G.
2. Show that it is possible to check in logarithmic space that a word $w \in \Sigma^*$ is a well-formed description of some G in list-of-edges representation.
3. - Describe a logarithmic space bounded deterministic Turing machine taking as input a graph G, represented by its adjacency list, and computing the list of edges representation of G.

OR

- Describe a logarithmic space bounded deterministic Turing machine taking as input a graph G, represented by its list of edges, and computing the adjacency matrix representation of G.

4. Explain why, with these two constructions, we can conclude that any of our three considered representations is logspace reducible to any other.

Solution. We make two preliminary observations.
– Firstly, it is easy to implement a binary counter on an empty worktape T_B. E.g., for little-endian representation we increment by reading T_B left to right, replacing `1`s by `0`s, stopping at the first `0` and replacing it with a `1`. If we reach the end of T_B without seeing a `0`, we add a `1` to the right. Then, we reset the TM head to the start of the tape, and the incrementing is over.
– Secondly, comparing two numbers written in binary (with no extra `0`s on the right) just needs logspace. If the counters are on different worktapes one needs no extra space and only linear time: one checks that the counters have different sizes and deduces which is larger, and if they have same size one checks if they coincide or reading from the right, finds the first bit where they differ and deduces which counter is the largest. If the counters are on the same worktape (usually the input tape) one just needs logarithmic space, e.g., to store pointers to the extremities of the counters before performing the comparison, or to copy them on different auxiliary worktapes and reuse the previous method.

In the following, N will denote the size of the input.

1. First, consider the adjacency matrix case. Checking that $w \in \Sigma^*$ consists in sequences of `0`s and `1`s separated by the symbol $\bullet$ and ending with the symbol $\#$ is straightforward[1]. But we also have to check that between any two consecutive $\bullet$s (and also before the first $\bullet$) there are exactly $m+1$ symbols, where m is the number of $\bullet$s. To do so, we may use an additional tape T_B as a counter, incrementing it each time the symbol $\bullet$ or $\#$ is seen, so that in the end T_B stores $m+1$. Note that $m+1$ written in binary takes $O(\log N)$ bits. Then, we consider another working tape T_W where, for each sequence in $\{\texttt{0},\texttt{1}\}^*$ we increment a counter until we reach the next $\bullet$ symbol. Then, we can check that it is equal to the number $m+1$ on T_B.

   The case of adjacency list is analogous, but instead of checking the number of symbols between two $\bullet$s, we have to check that every number written (between /) is smaller than the value of the counter written on the tape T_B.

2. We first use the Turing Machine we described in the previous question to check that the word considered is well-formed. Then, we consider a Turing Machine M with a tape T_i where a counter will loop between 0 and $k-1$ (where k is the number of $\bullet$ symbol plus 1) assuming the word we consider is well formed. The counter i is incremented whenever a new symbol `0` or `1` is seen and reinitialized at each $\bullet$ symbol. Furthermore, whenever $\bullet$ is seen, $\bullet$ is written on the output tape and whenever a `1` is seen, the counter i is copied on the output tape, preceded by symbol / if it is not the first time the counter i is copied since the last time a $\bullet$ symbol was written. The space taken by this TM is in $O(\log N)$ since we only use a counter whose value, written in binary is at most k and the logarithmic space taken by the TM described in the first question. Then, we conclude with the speedup theorem.

[1]This is a "regular" constraint and any regular language is in $\mathsf{TIME}(n) \cap \mathsf{SPACE}(0)$.

Solution. In the following, N will denote the size of the input.

1. n indicates the number of vertices. It cannot be computed from the set of edges since some vertices may be isolated (have no edge). Note that we cannot identify graphs that have the same edges but "only differ by some extra isolated vertices" since these graphs are different, e.g., when one asks "is G connected ?", or "does G have an independent set of size 8 ?"

2. As previously, checking that $w \in \Sigma^*$ consists in sequences of `0`s and `1`s separated by the symbol $\bullet$ and then alternatively by the symbol / and $\bullet$ and ending with the symbol # is straightforward. In addition, here we have to test that every vertex we read is smaller than n (it can go from 0 up to $n-1$). This can be done by comparing successively every vertex appearing in the input tape with n by using two counters, as described in the solution of Exercise 1.

3.
   - We first use the Turing Machine we described in the first question of the exercise to check that the input word is well-formed. Then, we first have to write in the output tape the number n in unary. We go through the word given as input, and each time a symbol $\bullet$ is seen, a 1 is written on the output tape. Note that this can not be done in a working tape and then copied in the output tape, since it would use more than logarithmic space (recall that the space used is counted in the working tapes, not the input and output ones). Then, we use counter i that is incremented each time a $\bullet$ symbol is seen. Each time a new vertex v is seen, we copy the counter i (written in binary!) on the output tape followed by /, we copy v on the output tape and we add a $\bullet$ symbol. The space taken by this TM is in $O(\log N)$ since we only use a counter whose value, written in binary is at most k, the number of $\bullet$ symbol in the input tape, and the logarithmic space taken by the TM described in the first question. Then, we conclude with the speedup theorem.
   - We first use the Turing Machine we described in the previous question to check that the word considered is well-formed. Then, we run two nested loops, first with counter i that goes from 0 to $n-1$ (where n is the value of the unary number written at the beginning of the input tape) and then with another counter j that also goes from 0 to $n-1$. For given i, j, we go through the input word and check whether it contains i/j or not. We output 1 or 0 depending on whether we found i/j. We then proceed to the next iteration of the loops by incrementing j and moving the reading head at the start of the input tape. In case $j = n$, we reset j to 0, and increase i and output a $\bullet$ symbol. Since for each value of j, with the value of i fixed, exactly one number is written in the output tape, it ensures that two $\bullet$ symbols are separated by exactly n bits. The space taken by this TM is in $O(\log N)$ since we store two counters whose value, written in binary is at most n and the logarithmic space taken by the TM described in the first question. Then, we conclude with the speedup theorem.

4. Lemma 2.11 in the course gives that the relation of logspace-reducibility if transitive. Hence, all three representations are logspace inter-reducible.