Time bounds imply space bounds

Given a language $L \in \text{DTIME}(t(n))$, what is the "smallest" space complexity class including L?

Of course $L \in \text{DSPACE}(t(n))$ (in $t(n)$ steps a TM T accepting L can only write $O(t(n))$ different tape cells).

But one can do much better ... a clever algorithm can simulate T using just $O(\sqrt{t(n)})$ tape cells (on any input of length n) even if some computation of the original T may need much more than $O(\sqrt{t(n)})$ space.

Assumptions.

- $t(n) \geq n^2$, and $t(n)$ must be *time constructible*.
- All TMs referred to in this lecture are "traditional" one-tape one-head TMs.

Examples?

Suppose L is formed by strings of the form $X\#Y\#Z$, where Z is a longest common subsequence of X and Y.

Let T be a deterministic TM that accepts L by running a deterministic algorithm based on the dynamic programming algorithm we saw few weeks ago.

We argued that the time complexity of that algorithm was quadratic in the size of its input (so $t(n) \geq n^2$) just because it had to fill a matrix with $|X| \times |Y|$ entries.

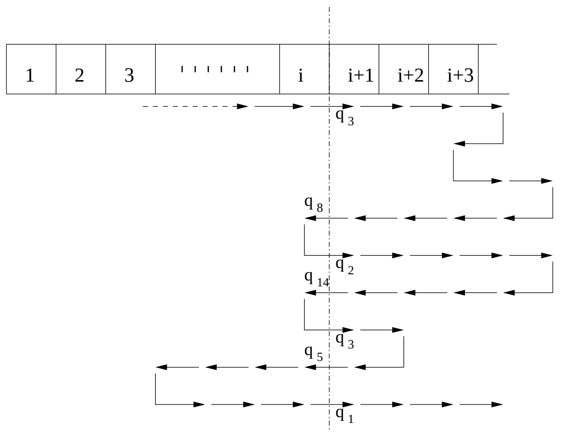
We are claiming now that a clever TM T' exists that can simulate T using $O(\sqrt{|X| \times |Y|})$ space. In some way T' will not need to store the matrix $lcs[i, j]$!!

Crossing sequences

The computation of a one-tape TM can be roughly described as a sequence of movements of the tape head occasionally paired with the updating of some of the symbols written on the tape.

Let T be a one-tape TM and x be an input. Let us focus our attention on the boundary between the ith and the $(i+1)$th tape cell for some $i \geq 0$ (cell "0" does not actually exist, so "the border between cell 0 and 1" will have a special meaning).

The sequence of states in which the machine is as its tape head crosses the boundary between cell i and $i+1$ is called the *crossing sequence at boundary i with input x*. In symbols $S_i(x)$.



Example. The crossing sequence at boundary i is $q_3, q_8, q_2, q_{14}, q_3, q_5, q_1$. Note that q_3 is the state the machine is in before analysing the content of the cell $i+1$. Similarly q_8 is the state the machine is in before analysing the content of cell i, and so on.

Even more practical example. The computation of the TM we saw last time:

$$
\begin{array}{|ccc|cccc}
0 & 0 & 0 & 1 & 1 & 1 & B \\
q_0 & & & & & & \\
X & 0 & 0 & 1 & 1 & 1 & B \\
 & q_1 & & & & & \\
X & 0 & 0 & 1 & 1 & 1 & B \\
 & & q_1 & & & & \\
X & 0 & 0 & 1 & 1 & 1 & B \\
 & & & q_1 & & & \\
X & 0 & 0 & Y & 1 & 1 & B \\
 & & q_2 & & & & \\
X & 0 & 0 & Y & 1 & 1 & B \\
 & q_2 & & & & & \\
X & 0 & 0 & Y & 1 & 1 & B \\
q_2 & & & & & &
\end{array}
$$

$$
\begin{array}{|ccc|cccc}
X & 0 & 0 & Y & 1 & 1 & B \\
 & q_0 & & & & & \\
X & X & 0 & Y & 1 & 1 & B \\
 & & q_1 & & & 1 & \\
X & X & 0 & Y & 1 & 1 & B \\
 & & & q_1 & & & \\
X & X & 0 & Y & 1 & 1 & B \\
 & & & & q_1 & & \\
X & X & 0 & Y & Y & 1 & B \\
 & & & q_2 & & & \\
X & X & 0 & Y & Y & 1 & B \\
 & & q_2 & & & & \\
X & X & 0 & Y & Y & 1 & B \\
 & q_2 & & & & &
\end{array}
$$

$$
\begin{array}{|ccc|cccc}
X & X & 0 & Y & Y & 1 & B \\
 & & q_0 & & & & \\
X & X & X & Y & Y & 1 & B \\
 & & & q_1 & & & \\
X & X & X & Y & Y & 1 & B \\
 & & & & q_1 & & \\
X & X & X & Y & Y & 1 & B \\
 & & & & & q_1 & \\
X & X & X & Y & Y & Y & B \\
 & & & & q_2 & & \\
X & X & X & Y & Y & Y & B \\
 & & & q_2 & & & \\
X & X & X & Y & Y & Y & B \\
 & & q_2 & & & &
\end{array}
$$

$$
\begin{array}{|ccc|ccccc}
X & X & X & Y & Y & Y & B & \\
 & & & q_0 & & & & \\
X & X & X & Y & Y & Y & B & \\
 & & & & q_3 & & & \\
X & X & X & Y & Y & Y & B & \\
 & & & & & q_3 & & \\
X & X & X & Y & Y & Y & B & \\
 & & & & & & q_3 & \\
X & X & X & Y & Y & Y & B & B \\
 & & & & & & & q_4
\end{array}
$$

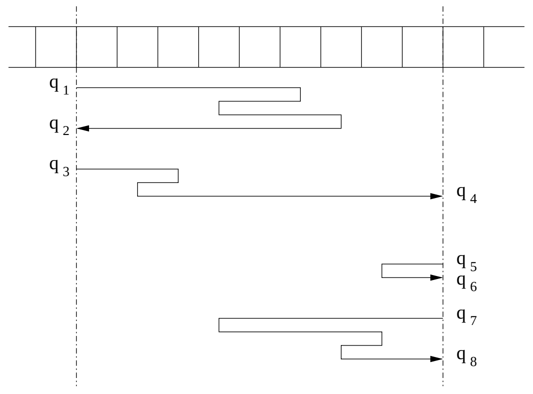
$S_0(000111) = q_0$ always; $S_3(000111) = q_1 q_2 q_1 q_2 q_1 q_2 q_0$.

Compatibility

Let $S = q_1, \ldots, q_r$ and $S' = q'_1, \ldots, q'_s$ be two lists of states. We say that S and S' are *compatible with respect to a portion of tape* y, if, starting T at the leftmost tape cell of y in state q_1, S and S' are generated at the left and right boundaries of y whenever the following two rules are applied:

1. if T crosses the left boundary of y moving left in state q_i with i even, then its tape head is placed on the leftmost tape cell of y and its state is set to q_{i+1};
2. if T crosses the right boundary of y moving right in state q'_i with i odd, then its tape head is placed on the rightmost tape cell of y and its state is set to q'_{i+1}.

Compatibility



Example

The sequences q_0 and $q_1q_2q_1q_2q_1q_2q_0$ are compatible w.r.t. the three leftmost cells of the tape of the Turing machine T accepting strings 0^n1^n.

Claim. Given $S_0(000111)$, $S_3(000111)$ and the initial content of the first three cells in the tape we can simulate T on those cells without any need of running the TM on any other cell.

Remarks

1. In words, the odd subsequence of S, $q_1, q_3, \ldots$, denotes the sequence of states of T when entering y from the left, while the even subsequence denotes the sequence of states of T when leaving y to the left. States in S' have a similar interpretation.
2. If T runs in time $O(t(n))$ then, for any input x, at most $c \times t(|x|)$ cells will ever be used in the computation of T on x, where c is a constant. We can improve on this by splitting these $c \times t(|x|)$ cells into blocks of (almost) equal size and check compatibility among crossing sequences at the boundaries of such blocks.

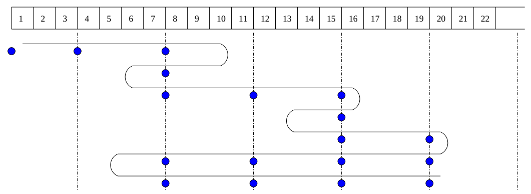
Preliminaries

For any positive integers i and d let $l = \lceil (c \cdot t(|x|) - i)/d \rceil$. The first $c \cdot t(|x|)$ cells are partitioned into $l + 1$ blocks. The first block y_0 contains the i left-most cells; block y_j for $1 \leq j \leq l$ contains cells from $i + d(j-1)$ to $i + dj - 1$.

The *i-th crossing sequence list of distance d and length l with input x* is the list of crossing sequences

$$S_0(x), S_i(x), S_{i+d}(x), S_{i+2d}(x), \ldots, S_{i+ld}(x).$$

A *state sequence sample of length l* is a list of $l + 2$ sequences of states.



Example. Assume $c \cdot t(|x|) = 20$, hence the TM uses cells 1, up to 20. Assume further that $i = 3$ and $d = 4$. Hence all blocks, except the first one, will have length four. The total number of blocks is one plus $l = \lceil (20-3)/4 \rceil = 5$. y_0 contains cells 1 to 3, y_1 cells 4 to 7, y_2 cells 8 to 11, y_3 contains cells 12 to 15, y_4 cells 16, to 19 and y_5 contains only cell 20.

Technical result

Let i, d be given and l defined as above. Then a state sequence sample $S_0, \ldots, S_{l+1}$ of length l is the ith crossing sequence list of distance d and length l with input x if and only if S_j, and S_{j+1} are compatible with respect to y_j, for any j with $0 \leq j \leq l$.

This result which can be proved by induction guarantees that the acceptance of an input can be checked by testing iteratively for the compatibility of state sequences with respect to portions of the tape. This will be exploited in the main algorithmic construction.

Main result

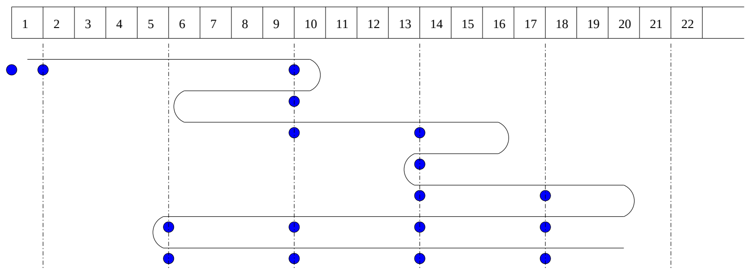
We will use $d = \sqrt{c \cdot t(n)}$ (hence $l = O(\sqrt{t(n)})$ too).

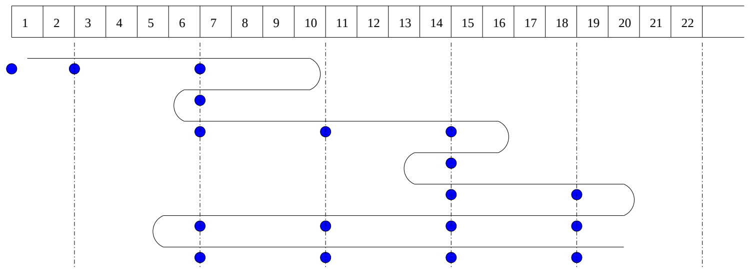
For any given machine T we define a new Turing machine T' which systematically generates state sequence samples of length l using $\sqrt{c \cdot t(n)}$ tape cells. For each of these, T' tests adjacent crossing sequences and the corresponding portion of the tape for compatibility (according to the definition of compatibility, this test can be performed using $O(\sqrt{t(n)})$ tape cells, and accept if all local tests have been successful.

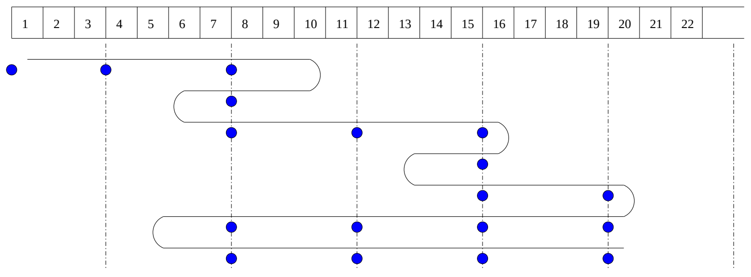
To complete the argument we need to convince ourselves that there exists an i such that the ith crossing sequence sample of distance d and length l "fits" in about $\sqrt{c \cdot t(n)}$ tape cells.

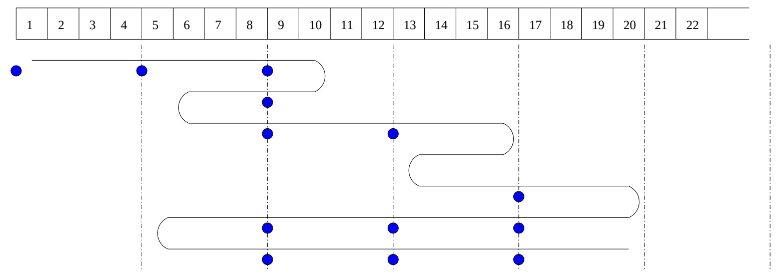
Claim. There is an $i \in \{1, \ldots, \sqrt{c \cdot t(n)}\}$ such that the sum of the lengths of the crossing sequences included in the ith crossing sequence sample of distance d and length l is, at most $\sqrt{c \cdot t(n)}$.

This claim must be true because the sum over all i of the lengths of all crossing sequence samples of distance d and length l is the total length of the computation of T. This is at most $c \cdot t(n)$. If all the samples had length larger than $\sqrt{c \cdot t(n)}$ their total length would exceed this bound.









Final remarks

1. The construction we have described is due to Hopcroft and Ullman (1968).
2. Similar result can be proved for non-deterministic or off-line TMs.
3. The time taken by T' is exponential in $t(n)$ (T' has to find the right sequence of states out of a large number of possibilities).
4. The assumption $t(n) \geq n^2$ can be dropped but a weaker result holds. The following relation was proved by Hopcroft, Paul and Valiant (1977).
$$\text{DTIME}(t(n)) \subseteq \text{DSPACE}(t(n)/\log t(n))$$
5. Ibarra and Moran (1983) presented a polynomial time simulation result:
$$\text{DTIME}(t(n)) \subseteq \text{DTIMESPACE}((t(n))^2, \sqrt{t(n)}) \text{ if } t(n) \geq n^2.$$