1.4.10 (a), (f) and (j)

Let f and g be functions from $\mathbb{N}$ to $\mathbb{N}$, where $\mathbb{N} = \{0, 1, 2, 3, \dots\}$. We write $f(n) = \mathcal{O}(g(n))$ if there are positive integers c and n_0 such that, for all $n \geq n_0$, $f(n) \leq c \cdot g(n)$. Similarly, $f(n) = \Omega(g(n))$ if $g(n) = \mathcal{O}(f(n))$, and $f(n) = \Theta(g(n))$ if both $f(n) = \mathcal{O}(g(n))$ and $f(n) = \Omega(g(n))$.

In the following we compare the pairwise rate of growth of the following functions: (a) n^2, (f) $n^{\log n}$ and (j) $h(n) = \begin{cases} n^2 & \text{if } n \text{ is odd} \\ 2^n & \text{otherwise} \end{cases}$. All logarithms are base 2 if not otherwise stated.

First of all, note that $f(n) \leq c \cdot g(n)$ iff $\frac{f(n)}{g(n)} \leq c$ whenever c, $f(n)$ and $g(n)$ are all positive. Therefore, in order to $f(n) = \mathcal{O}(g(n))$ to hold, the ratio $\frac{f(n)}{g(n)}$ should be limited by a constant for all $n \geq n_0$, n_0 sufficiently large.

Case 1: n^2 versus $n^{\log n}$

We consider a generalized case: n^k vs. $n^{\log n}$, where k is a *fixed* positive constant.

$n^k = \mathcal{O}(n^{\log n})$ for any fixed $k > 0$ because the ratio $\frac{n^k}{n^{\log n}} = n^{k-\log n}$ is limited by a constant, for instance by 1, since $k - \log n \leq -1$ for all $n \geq 2^{k+1}$ and therefore $n^{k-\log n} \leq 1$ for all $n \geq 2^{k+1}$. Another way to show $n^k = \mathcal{O}(n^{\log n})$ is to note that by choosing $c = 1$ and $n_0 = 2^k$, $n^k = n^{\log n_0} \leq 1 \cdot n^{\log n}$ for all $n \geq n_0$.

On the other hand, $n^{\log n} \neq \mathcal{O}(n^k)$ as the ratio $\frac{n^{\log n}}{n^k} = n^{\log(n)-k}$ is not limited by any constant since $\log(n) - k > 1$ for all $n > 2^{k+2}$. This implies that $n^k \neq \Omega(n^{\log n})$ and $n^k \neq \Theta(n^{\log n})$.

Therefore, *all polynomials grow strictly slower than* $n^{\log n}$.

Case 2: n^2 versus $h(n)$

$n^2 = \mathcal{O}(h(n))$ because $n^2 = \mathcal{O}(n^2)$ and $n^2 = \mathcal{O}(2^n)$ (left as home exercise). However, because $n^2 \neq \Omega(2^n)$ (left as a home exercise), $n^2 \neq \Omega(h(n))$ and $n^2 \neq \Theta(h(n))$.

Case 3: $n^{\log n}$ versus $h(n)$

To compare $n^{\log n}$ and $h(n)$, we have to find out the relationship between $n^{\log n}$ and 2^n — we already know that between $n^{\log n}$ and n^2 from the first part of the exercise. Since the 2-base logarithm of the ratio $\frac{n^{\log n}}{2^n}$, namely $\log(n) \cdot \log(n) - n \cdot \log(2) = \log^2(n) - n$, is negative for large values of n, the ratio $\frac{n^{\log n}}{2^n}$ itself must be in range $]0, 2[$ and is therefore limited by a constant for sufficiently large n. Thus $n^{\log n} = \mathcal{O}(2^n)$. On the other hand, the 2-base logarithm of $\frac{2^n}{n^{\log n}}$, $n \cdot \log(2) - \log(n) \cdot \log(n) = n - \log^2(n)$, is positive and grows without bounds when n grows and therefore the ratio $\frac{2^n}{n^{\log n}}$ itself must grow without an upper limit. Thus $2^n \neq \mathcal{O}(n^{\log n})$, i.e., $n^{\log n} \neq \Omega(2^n)$. Therefore 2^n *grows strictly faster than* $n^{\log n}$. We can now conclude that $n^{\log n} \neq \mathcal{O}(h(n))$ since $n^{\log n} \neq \mathcal{O}(n^2)$ and $h(n) \neq \mathcal{O}(n^{\log n})$ since $2^n \neq \mathcal{O}(n^{\log n})$. Therefore, $n^{\log n}$ *and* $h(n)$ *cannot be ordered according to their rate of growth by using the big-oh notation!*

2.8.7

(a)

A *two-dimensional Turing machine* (2D-TM) M is a tuple

$$M = \langle K, \Sigma, \delta, s \rangle,$$

where

- K is a finite set of *states*,
- Σ is a finite set of *symbols* such that $K \cap \Sigma = \emptyset$ and $\Sigma \supset \{\sqcup, \triangleright, \triangle, \Diamond\}$
- $\delta : K \times \Sigma \to (K \cup \{h, \text{“yes”}, \text{“no”}\}) \times \Sigma \times \{\to, \leftarrow, \uparrow, \downarrow, -\}$ is the *transition function*, where $\{\to, \leftarrow, \uparrow, \downarrow, -, h, \text{“yes”}, \text{“no”}\} \cap (K \cup \Sigma) = \emptyset$ and
- $s \in K$ is the *initial state*.

The transition function δ is restricted in the following way:

1. $\delta(q, \triangleright) \in K \times \{\triangleright\} \times \{\to\}$,
2. $\delta(q, \Diamond) \in K \times \{\Diamond\} \times \{\to\}$, and
3. $\delta(q, \triangle) \in K \times \{\triangle\} \times \{\uparrow\}$.

A blackboard is a two-dimensional array unbounded in right and upwards directions only, i.e. it corresponds to the the upper right quadrant of the standard coordinate system. Thus, the contents of the board could be described by a function $\mathbb{N} \times \mathbb{N} \to \Sigma$. However, we define a configuration of a 2D-TM to be a triple $\langle q, w, u \rangle \in K \times (\Sigma^+)^+ \times (\Sigma^+)^+$, where q is the current state, w describes the rows from the bottom of the board up to the row of the cursor plus the row of the cursor right to and including the cursor, and u describes the right part of cursor row (including at least one symbol) plus the non-empty rows above the cursor row. The configuration is finite because with a finite initial configuration and a finite number of steps the number of the rows and symbols in the rows (i.e. the number of the coordinates visited) is finite. The initial configuration is $\langle s, \langle \Diamond, \triangleright \rangle, \langle x_1 x_2 \dots x_n \rangle \rangle$, where $x_1 x_2 \dots x_n$ is the input. The configuration is updated in the obvious way during the computation so that the origo always contains $\Diamond$, the other positions in the x-axis will always contain $\triangle$ and the other positions in the y-axis will contain $\triangleright$. The rows are always filled up to one position right to the cursor either with blanks or, in the case of the first row, with

$\triangle$s. For example, if the contents of the board are

$\triangleright$	$\underline{x_1}$	x_2	x_3
$\Diamond$			

, where the cursor position is denoted by underlined symbol, then if we move down we get the following board contents:

$\triangleright$	x_1	x_2	x_3
$\Diamond$	$\underline{\triangle}$	$\triangle$	

. The corresponding configurations are $\langle q, \langle \Diamond, \triangleright x_1\rangle, \langle x_2 x_3\rangle\rangle$ and $\langle q', \langle \Diamond\triangle\rangle, \langle \triangle, \triangleright x_1 x_2 x_3\rangle\rangle$.

(b)

We simulate a 2D-TM $M = \langle K, \Sigma, \delta, s\rangle$ with a 3-string TM M' as follows. The first tape contains the rows below the cursor and the symbols in the row of the cursor from left up to and including the symbol under cursor. The second tape contains the symbols to the right of the cursor in the row of the cursor and the (finite number of already accessed) rows above the cursor. The symbols in second tape are listed backwards, starting from the upper right corner of the board, the last symbol being the symbol right to the cursor. The ends of rows are denoted by a special symbol $\triangleleft$ not in Σ. The symbol $\triangleright$ is primed in order to distinguish it from the true special symbol of the simulating machine M'.

For example, the board

$\triangleright$	$\underline{x_1}$	x_2	x_3
$\Diamond$			

is coded in the first two tapes as $\langle \triangleright\Diamond \triangleleft \triangleright' x_1, \triangleright \triangleleft x_3 x_2\rangle$.

We now define how the moves of M are simulated. Note that in any computation halting in time $f(n)$, any row is at most of length $f(n)$. Let $\delta(q,\sigma) = \langle p, \rho, D\rangle$ and w, u be the contents of the first and second tape. Assume that the cursors of the simulating machine are at the end of the strings.

- If $D = -$, then the last symbol of the first tape is replaced by ρ. This is clearly a constant time operation.
- If $D = \leftarrow$, then the last symbol of the first tape is replaced by ρ and is removed from the first tape and appended into the second tape. This is clearly a constant time operation.
- If $D = \rightarrow$, then the last symbol of the first tape is replaced by ρ and the last symbol of the second tape is appended into the first tape. However, if the last symbol of the second tape was $\triangleleft$, then the second tape is left intact but the first tape is appended with $\sqcup$. For example,

$\triangleright$	x_1	x_2	$\underline{x_3}$
$\Diamond$			

  with $\rightarrow$ changes to

$\triangleright$	x_1	x_2	x_3	$\underline{\sqcup}$
$\Diamond$				

  and the contents of the first two tapes are $\langle \triangleright\Diamond \triangleleft \triangleright' x_1 x_2 x_3, \triangleright\triangleleft\rangle$ and $\langle \triangleright\Diamond \triangleleft \triangleright' x_1 x_2 x_3 \sqcup, \triangleright\triangleleft\rangle$, respectively. Clearly this is a constant time operation.
- If $D = \downarrow$, then the last symbol of the first tape is replaced by ρ. M' then calculates the cursor position in the current row by moving in the first

tape to the nearest $\triangleright'$ and calculating this distance in unary in the third tape. The contents right to the cursor or the first tape are simultaneously copied to the end of the second tape. It then finds the beginning of the row below by moving left in the first tape until $\triangleright'$ or $\Diamond$ is encountered. The contents right to the cursor of the first tape are again simultaneously copied to the end of the second tape. It then moves right the amount of symbols in the third tape, copying symbols from the end of the second tape to the first tape, or if $\triangleleft$ is encountered, stops the copying and pads the first tape with the amount of $\triangle$s or $\sqcup$s depending on whether the row is the first one or not. For example, $\begin{array}{|c|c|c|c} & & & \\ \hline \triangleright & x_1 & \underline{x_2} & x_3 \\ \hline \Diamond & & & \\ \hline \end{array}$ with $\downarrow$ changes to $\begin{array}{|c|c|c|c} & & & \\ \hline \triangleright & x_1 & x_2 & x_3 \\ \hline \Diamond & \triangle & \underline{\triangle} & \\ \hline \end{array}$ and the contents of the first two tapes are $\langle \triangleright\Diamond \triangleleft \triangleright' x_1 x_2, \triangleright \triangleleft x_3 \rangle$ and $\langle \triangleright\Diamond\triangle\triangle, \triangleright \triangleleft x_3 x_2 x_1 \triangleright' \triangleleft \rangle$, respectively. Clearly this is a $\mathcal{O}(f(n))$ time operation because the maximum length of any row is $f(n)$. Note that in the simulation we only need to pad up to cursor, not up to the position after cursor because we have the special $\triangleleft$ symbol denoting the end of row.

- If $D = \uparrow$ is analogous to the previous case. For example, $\begin{array}{|c|c|c|c} & & & \\ \hline \triangleright & x_1 & \underline{x_2} & x_3 \\ \hline \Diamond & & & \\ \hline \end{array}$ with $\uparrow$ changes to $\begin{array}{|c|c|c|c} & & & \\ \hline \triangleright & \sqcup & \underline{\sqcup} & \\ \hline \triangleright & x_1 & x_2 & x_3 \\ \hline \Diamond & & & \\ \hline \end{array}$ and the contents of the first two tapes are $\langle \triangleright\Diamond \triangleleft \triangleright' x_1 x_2, \triangleright \triangleleft x_3 \rangle$ and $\langle \triangleright\Diamond \triangleleft \triangleright' x_1 x_2 x_3 \triangleleft \triangleright' \sqcup\sqcup, \triangleright\triangleleft \rangle$, respectively. Clearly this is a $\mathcal{O}(f(n))$ time operation because the maximum length of any row is $f(n)$.

Thus each move of M can be simulated in time $\mathcal{O}(f(n))$ and thus each halting computation of the 2D-TM M can be simulated by a 3-string TM in time $\mathcal{O}(f(n)^2)$.