of its tape, $T_1$ can decide what move $T$ makes next. $T_1$ makes these changes on its next sweep. Note that no two head markers can move apart by more than $\frac{1}{4}T(n)$ cells if $T$ is $\frac{1}{4}T(n)$ time bounded. Thus a sweep requires no more than $\frac{1}{4}T(n) + 2k$ moves. Hence, $T_1$ is of time complexity at most $T^0(n)$.

**Theorem 10.5.** If $L$ is accepted by a $k$-tape $T(n)$ time-bounded Turing machine $T$, then $L$ is accepted by a two storage tape $Tm$ $T_1$ in time $T(n) \log T(n)$.\[^{\dagger}\]

*Proof.* The proof is complicated, and only an indication of how $T_1$ could simulate $T$ in time proportional to $T(n) \log T(n)$ will be given. The $Tm$ $T_1$ that we shall describe has storage tapes infinite in both directions. The construction used in Theorem 6.1 shows that $T_1$ could be converted to a $Tm$ making the same number of moves as $T_1$, but with semi-infinite tapes. The first storage tape of $T_1$ will have two tracks for each storage tape of $T$. For convenience, we focus on two tracks corresponding to a particular tape of $T$. The other tapes of $T$ are handled in exactly the same way. The second tape of $T_1$ is used only for scratch, to transport blocks of data on tape 1.

One particular cell of tape 1, known as $B_0$, will hold the storage symbols scanned by each of the heads of $T$. That is, rather than moving head markers, $T_1$ will transport data across $B_0$ in the direction opposite that of the motion of the head of $T$ being simulated. To the right of cell $B_0$ will be blocks $B_1, B_2, \ldots$ of exponentially increasing length; that is, $B_i$ is of length $2^{i-1}$. Likewise, to the left of $B_0$ are blocks $B_{-1}, B_{-2}, \ldots$, with the length of $B_{-1}$ the same as the length of $B_1$. The markers between blocks are assumed to exist, although they will not actually appear until the block is scanned.

\[^{\dagger}\] By Theorems 10.1 and 10.3, constant factors are irrelevant, so we do not need to specify logarithmic bases.
Let us denote the contents of the cell initially scanned by this tape head of $T$ by $a_0$. The contents of the cells to the right of this cell are $a_1, a_2, \ldots$, and those to the left, $a_{-1}, a_{-2}, \ldots$. Initially these are all blank, however it is not their value, but their position on the tracks of tape 1 of $T_2$, that is important. Initially the upper track of $T_1$ for the tape of $T$ in question is assumed to be empty, while the lower track is assumed to hold $\ldots, a_{-2}, a_{-1}, a_0, a_1, a_2, \ldots$. These are placed in blocks $\ldots, B_{-2}, B_{-1}, B_0, B_1, B_2, \ldots$ as shown in Fig. 10.4.

Fig. 10.4. Blocks on Tape 1.

As we mentioned previously, data will be shifted across $B_0$ and perhaps changed as it passes through. The method of shifting data will obey the following rules.

1. For any $i > 0$, either $B_i$ is full (both tracks) and $B_{i-1}$ is empty, or $B_i$ is empty and $B_{i-1}$ is full, or the bottom tracks of both $B_i$ and $B_{i-1}$ are full, while the upper tracks are empty.

2. The contents of any $B_i$, $B_{-i}$ represents consecutive cells on the tape of $T$ represented. For $i > 0$, the upper track represents cells to the left of those of the lower track; and for $i < 0$, the upper track represents cells to the right of those of the lower track.

3. $B_i$ represents cells to the left of those of $B_j$, for $-\infty < i < j < \infty$.

4. $B_0$ always has only its lower track filled.

To see how data is transferred, let us imagine that on successive moves the tape head of $T$ in question moves to the left. Then $T_1$ must shift the corresponding data right. To do so, $T_1$ moves the head of tape 1 from $B_0$, where it rests, and goes to the right until it finds the first block, say $B_0$, which is not full. Then $T_1$ copies all the data of $B_0, B_1, \ldots, B_{i-1}$ onto tape 2 and stores it in the lower track of $B_1, B_2, \ldots, B_{i-1}$ and, if $B_i$ is completely empty, the lower track of $B_i$. If the lower track of $B_i$ is already filled, then the upper track of $B_i$, as well as the lower track of $B_1, B_2, \ldots, B_{i-1}$ receives all the data of $B_0, B_1, \ldots, B_{i-1}$.

Note that, in either case, there is just enough room to distribute the data. Also, the data can be picked up and stored in its new location in time proportional to the length of $B_i$. Finally, note that the data can be easily stored in a manner that satisfies Rules 1, 2, and 3, above.
Next, in time proportional to the length of $B_0$, $T_1$ can find $B_{-i}$ (using tape 2 to measure the distance from $B_i$ to $B_0$ makes this easy). If $B_{-1}$ is completely full, $T_1$ picks up the upper track of $B_{-1}$ and stores it on tape 2. If $B_{-i}$ is half full, the lower track is put on tape 2. In either case, what has been copied to tape 2 is next copied to the lower tracks of $B_{-(i-1)}$, $B_{-(i-2)}$, $\ldots$, $B_0$. (By Rule 1, these tracks have to be empty, since $B_1$, $B_2$, $\ldots$, $B_{i-1}$ were full.) Again, note that there is just enough room to store the data, and all the above operations can be carried out in time proportional to the length of $B_i$.

We call all that we have described above a $B_i$ operation. The case in which the head of $T$ moves to the right is analogous. The successive contents of the blocks as $T$ moves its tape head in question five cells to the right is shown in Fig. 10.5.
We note that on any pair of tracks $T_1$ can perform a $B_i$ operation at most once per $2^{i-1}$ moves of $T$, since it takes this long for $B_1, B_2, \ldots, B_{i-1}$, which are half empty after a $B_i$ operation, to fill. Also, a $B_i$ operation cannot be performed for the first time until the $2^{i-1}$th move of $T$. Hence, if $T$ operates in time $T(n)$, $T_1$ will perform only $B_i$ operations, for those $i$ such that $i \leq \log_2 T(n) + 1$.

We have seen that there is a constant $m$, such that $T_1$ uses at most $m2^i$ moves to perform a $B_i$ operation. If $T$ makes $T(n)$ moves, $T_1$ makes at most

$$T_1(n) = \sum_{i=1}^{\log_2 T(n) + 1} m2^i \frac{T(n)}{2^i}$$

(10.1)

moves.

From (10.1), we obtain

$$T_1(n) = 2mT(n)[\log_2 T(n) + 1]$$

(10.2)

and from (10.2),

$$T_1(n) \leq 4mT(n) \log_2 T(n).$$

The reader should be able to see that $T_1$ operates in time $T_1(n)$ even when $T$ makes moves using different storage tapes rather than only the one upon which we have concentrated.

10.4 SINGLE-TAPE TURING MACHINES AND CROSSING SEQUENCES

For single-tape Turing machines we can prove some results of the form that “such and such a language requires $T(n)$ time to be recognized by a single-tape Tm.” In such a case, it is possible that the language could be recognized in less than $T(n)$ steps by a Tm with more than one tape.

First, let us give a speed up theorem for single-tape Tm’s.

**Theorem 10.6.** If $L$ is accepted by a single-tape Tm $T$ of time complexity $T(n)$ and $\inf_{n \to \infty} T(n)/n^2 = \infty$, then, for any $c > 0$, $L$ is accepted by a single-tape Tm of time complexity $cT(n)$.†

**Proof.** In $n^2$ steps, a single-tape Tm $T_1$ can condense its input by encoding $m$ symbols into 1. The proof then proceeds as in Theorem 10.3.

For these simple machines, a useful tool has been developed known as the crossing sequence. We imagine that when the Tm makes its move it first overprints the symbol scanned and changes state, then moves its head. Thus, for any pair of adjacent cells on the input tape, we may list the sequence of states in which the Tm crosses from one to the other. Note that the first

† Again, we replace $cT(n)$ by $n + 1$ if $cT(n) < n + 1$. 