For definition of complexity notions, we assume the model of Turing Machine with multiple, but fixed, number of tapes.

**Time Complexity**

$Time_M(x)$: Time used by a machine $M$ on input $x$ before halting (if $M$ does not halt on input $x$, then $Time_M(x) = \infty$).

$M$ is $T(n)$ time bounded, if for any input $x$ of length $n$, $Time_M(x) \leq T(n)$.

We usually assume $T(n) \geq n$. 
Space Complexity

Read only input tape.

$Space_M(x)$: maximum number of cells touched by the $M$, on input $x$, on any of its worktapes (input tape is not counted; in many cases output tape is also not counted, which is then one-way write only tape).

End markers for input: $x$

If the machine does not halt on an input, then $Space_M(x)$ is taken to be infinite.

$M$ is $S(n)$ space bounded, if for any input $x$ of length $n$, $Space_M(x) \leq S(n)$. 
Tape Compression

**Theorem:** Fix $c > 0$. If a language $L$ is accepted by a machine $M$, with $k$ tapes, that is $S(n)$ space bounded, then $L$ is accepted by a machine $M'$, with $k$ tapes, that is $\lceil cS(n) \rceil$ space bounded.

**Proof:**
Suppose $M$ is $S(n)$ space bounded and accepts $L$.
Construct $M'$, which simulates $M$ but uses less space.

Each cell of a worktape of $M'$ codes $m$ cells of the corresponding tape of $M$. (This increases the alphabet size used by $M'$, but that is ok.)

Simulation: finite control of $M'$ keeps track of which of the $m$ cells represented by the presently scanned cell of the tape(s) of $M'$ is actually being scanned by $M$. $M'$ accepts an input iff $M$ does.
Space used by $M'$ on input $x$ is:

$$\left\lceil \frac{Space_M(x)}{m} \right\rceil$$

Take $m > \frac{1}{c}$.
Thus, space used is at most

$$\left\lceil c \times Space_M(x) \right\rceil$$
Linear Speedup

**Theorem:** Fix $c > 0$. Suppose $L$ is accepted by a machine $M$, with $k \geq 2$ tapes, that is $T(n)$ time bounded, where $\lim_{n \to \infty} T(n)/n = \infty$. Then $L$ is also accepted by a machine $M'$ that is $\lceil cT(n) \rceil$ time bounded.

**Proof:**
We use a similar coding as in the tape compression theorem except that we code the input tape also.
Initialization:

First copy the input tape into one of the working tapes, coding it along the way ($m$ cells to one).

Reset the head of this working tape to the beginning.

From now on use the above working tape as input tape, and the input tape as a work tape in the simulation below. (Do not need to reset the head of input tape! — just mark a special symbol on the tape denoting the new beginning of the tape).
In one “basic step” $M'$ will simulate several steps of $M$. One basic step of $M'$ consists of
1. reading the cells scanned by the heads of $M'$ (let us call them home cells);
2. reading the cells to the left and right of the home cells of each tape;
3. determine the contents of the home cells and the cells to the left and right (for each tape) when a head of $M$ first leaves the cells represented by the corresponding region
4. Updating the home cells and the cells to the left and right of home cells;
5. Repositioning the heads of $M'$ to the new home cells.

If during the process of a basic step, $M$ accepts, then $M'$ also accepts.
In one basic step $M'$ has simulated at least $m$ steps of $M$ since it takes at least that much time for any head of $M$ to leave the region represented by the home cells and the cells to their left and right.

Step 3 can be done in the logic of $M'$ and thus can be done instantly.

Thus only need to count the steps needed to visit the respective cells to read/write and repositioning the home cells. This is $\leq 8$.

Thus in 8 time steps of $M'$ we can simulate $m$ time steps of $M$.

Thus the total time used by $M'$ for the simulation of $M$ on input $x$ of length $n$ is

$\leq n + \left\lceil \frac{n}{m} \right\rceil + 8\left\lceil \frac{T(n)}{m} \right\rceil \leq n + \frac{n}{m} + \frac{8T(n)}{m} + 9$. 
We need to pick $m$ such that

$$n + \frac{n}{m} + \frac{8T(n)}{m} + 9 \leq cT(n)$$

Need to worry only about large enough $n$ (smaller values of $n$ can be easily taken care of).
Without loss of generality assume $0 < c < 1$.
Pick $m > 40/c$. Then,

$$\frac{8T(n)}{m} \leq cT(n)/4$$

Since $\lim_{n \to \infty} T(n)/n = \infty$, for large enough $n$,

$$9 \leq n/m \leq n \leq \frac{cT(n)}{4},$$

Thus, for large enough $n$, time complexity of $M'$ is bounded by $\lceil cT(n) \rceil$. 
We really do not need $\lim_{n \to \infty} \frac{T(n)}{n} = \infty$ to get the linear speed up. We can get the speed up as long as we can find $m$ such that

$$n + \left\lceil \frac{n}{m} \right\rceil + 8\left\lceil \frac{T(n)}{m} \right\rceil \leq cT(n)$$

**Corollary**: Fix $c > 0$. Suppose $L$ is accepted by a machine $M$, with $k \geq 2$ tapes, that is $d \times n$ time bounded, for some constant $d$. Then $L$ is also accepted by a machine $M'$ that is $(1 + c)n$ time bounded.

Proof: In the simulation, choose $m > \max(24d/c, 3/c)$. 
Arbitrarily difficult problems

Suppose we are given a total recursive function $f$. We want to construct a recursive function $g$ such that no $f(n)$ time bounded machine can compute $g$. Define $g$ as follows:

$$g(x)$$

1. Simulate $M_x$, on input $x$.
2. If $M_x$ does not halt within $f(|x|)$ steps, then output 0.
3. Otherwise output something different from the output of $M_x(x)$. (say $M_x(x) + 1$).

End
Claim: \( g \) cannot be computed correctly by any \( f(n) \) time bounded machine.
Proof: Suppose by way of contradiction machine \( M_y \) does so.
Consider \( M_y(y) \).
If \( M_y(y) \) halts within \( f(|y|) \) steps, then by construction of \( g \), \( g(y) \neq M_y(y) \).
If \( M_y(y) \) does not halt within \( f(|y|) \) steps, then \( M_y \) is not \( f(n) \) time bounded.
Blum Complexity Measure

A complexity measure $\Phi$ is called a Blum Complexity measure iff $\Phi(x, y)$ is a partial recursive function in $x$ and $y$ and

(A1) $\varphi_x(y) \downarrow \iff \Phi(x, y) \downarrow$.

(A2) The predicate ‘$\Phi(x, y) \leq z$?’ is recursive in $x, y, z$.

We usually write $\Phi_x(y)$ for $\Phi(x, y)$.

Note that most complexity measures such as time and (modified) space complexity measures are Blum complexity measures.
DSPACES, DTIME, NSPACE, NTIME

\[ DSPACE(S(n)) = \{ L : \text{some } S(n) \text{ space bounded deterministic machine accepts } L \} \].

\[ DTIME(T(n)) = \{ L : \text{some } T(n) \text{ time bounded deterministic machine accepts } L \} \].

\[ NSPACE(S(n)) = \{ L : \text{some } S(n) \text{ space bounded nondeterministic machine accepts } L \} \].

\[ NTIME(T(n)) = \{ L : \text{some } T(n) \text{ time bounded nondeterministic machine accepts } L \} \].

We can similarly define the classes for function computation.
Space/Time constructible functions

A function $S(n)$ is said to be space constructible if there exists a $S(n)$ space bounded Turing machine $M$ such that, for every $n$, $M$ uses space exactly $S(n)$ for some input of length $n$.

A function $T(n)$ is said to be time constructible if there exists a $T(n)$ time bounded Turing machine $M$ such that, for every $n$, $M$ uses time exactly $T(n)$ for some input of length $n$.

A function $S(n)$ is said to be fully space constructible if there exists a $S(n)$ space bounded Turing machine $M$ such that, on all inputs of length $n$, it uses space exactly $S(n)$.

A function $T(n)$ is said to be fully time constructible if there exists a $T(n)$ time bounded Turing machine $M$ such that, on every input of length $n$, it halts and uses time exactly $T(n)$.