(27.1-3) Suppose the algorithm runs with $x$ incomplete steps. Then there are at most $T_1 - x$ strands available to run in complete steps, and the number of complete steps is less than or equal to $\frac{T_1 - x}{P}$. Therefore,

$$T_P \leq \frac{T_1 - x}{P} + x.$$ 

Now find the maximum of this function for $0 \leq x \leq T_\infty$.

(27.1-6) Refer to the algorithm Mat-Vec-Wrong which, as described in the textbook, has a race condition. The problem is that there are multiple threads attempting to update the variable $y_i$. One approach to fixing the problem is to break down the matrix-vector multiply into two separate steps that eliminates the race:

1. For each $i$, compute the products $a_{ij} \cdot x_j$ and save them in a new matrix $b$ rather than summing to $y_i$.

2. Efficiently sum the rows of $b$ to compute $y_i$ (i.e. $y_i = \sum_{j=1}^n b_{ij}$).

Step (1) can be achieved by changing line 7 of Mat-Vec-Wrong to $b_{ij} = a_{ij} \cdot x_j$. The loop structure is unchanged, so the span remains $\Theta(\lg n)$, but we have removed the race condition. For step (2), we need some way to sum a row of $b$ in $\Theta(\lg n)$ time. This can be done simply with a multithreaded, divide-and-conquer summation:

P-Sum($L$)
1  $n = L.length$
2  if $n == 1$
3      return $L[1]$
4  $c = \lfloor n/2 \rfloor$
5  $x = \text{spawn P-Sum}(L[1..c])$
6  $y = \text{P-Sum}(L[c+1..n])$
7  sync
8  return $x + y$
Using the same analysis that we applied in class to P-Fib, we see that $T_\infty$ for P-Sum satisfies

\[ T_\infty(n) = T_\infty(n/2) + \Theta(1). \]

By MT(ii), $T_\infty(n) = \Theta(\lg n)$. Therefore, our combined algorithm (step 1 followed by step 2) is $\Theta(\lg n)$.

\textbf{(27.1-7)} We can assume the exchange on line 4 is $\Theta(1)$. The number of iterations of the inner loop (line 3) are $1, 2, \ldots, n - 1$ and the work $T_1$ is the sum, or $n(n - 1)/2$, which is $\Theta(n^2)$. Therefore, $T_1(n) = \Theta(n^2)$. To analyze the span, we use the following formula from the textbook:

\[ T_\infty(n) = \Theta(\lg n) + \max_{2 \leq j \leq n} \text{iter}_\infty(j) \]

where $\text{iter}_\infty(j)$ is the span of the $j^{th}$ iteration. The $\Theta(\lg n)$ term is the cost of recursive spawning introduced by the parallel for construct. In this case, $\text{iter}(j)$ is the code on lines 3 and 4; since this is another parallel loop, and line 4 is $\Theta(1)$, we have that

\[ \text{iter}_\infty(j) = \Theta(\lg n) + \Theta(1) = \Theta(\lg n). \]

and therefore,

\[ T_\infty(n) = \Theta(\lg n) + \Theta(\lg n) = \Theta(\lg n). \]

Finally, the parallelism is

\[ T_1/T_\infty = \Theta(n^2/\lg n). \]