P1) We say a directed graph $G$ is strongly connected if for every pair of vertices $u, v$ there is a directed path from $u$ to $v$ and a directed path from $v$ to $u$. For example, in the following picture the left graph is strongly connected but the right one is not. Design an algorithm that runs in time $O(m + n)$ and outputs “yes” if $G$ is strongly connected and “no” otherwise.

P2) Given a sequence $d_1, \ldots, d_n$ of integers design a polynomial time algorithm that construct a outward-rooted tree such that the out-degree of vertex $i$ is $d_i$. An outward-rooted tree is a directed tree where the is a path from root to each vertex. If no such tree exists your algorithm must output “Impossible”, otherwise output the edges of the tree. For example, given $1, 2, 0, 0$, we can construct the following tree:

```
1 -> 2
   |
   3
```

**Hint:** Show that for every sequence $d_1, \ldots, d_n$ of integers there exists a outward-rooted tree where the out-degree of $i$ is $d_i$ if and only if $\sum_i d_i = n - 1$ and for all $i$, we have $d_i \geq 0$.

P3) You have $n$ jobs, labeled $1, \ldots, n$, which must be run one at a time, on a single processor. Job $j$ takes time $t_j$ to be processed. We will assume that no two jobs have the same processing time; i.e., $t_i \neq t_j$ for all $i \neq j$. You must decide on a schedule: the order in which to run the jobs. Having fixed an order, each job $j$ has a completion time $C_j$ under this order: this is the total amount of time that elapses (from the beginning of the schedule) before it is done being processed. For example, suppose you have a set of three jobs $\{1, 2, 3\}$ with $t_1 = 3, t_2 = 1, t_3 = 5$, and you run them in this order. Then the completion time of job 1 will be 3, the completion of job 2 will be $3 + 1 = 4$, and the completion time of job 3 will be $3 + 1 + 5 = 9$. Give a polynomial-time algorithm that takes the $n$ processing times $t_1, \ldots, t_n$, and orders the jobs so that the sum of the completion times of all jobs is as small as possible.

P4) You are given a graph $G$ with $n$ vertices and $m$ edges, and a minimum spanning tree $T$ of the graph. Suppose we add a new edge $e$ with weight $w(e)$ to $G$; call this new graph $G'$. Give an algorithm that runs in time $O(n)$ to test if $T$ is still the MST and “no” otherwise. You may assume that all edge weights are distinct.
P5) **Extra Credit:** Suppose \( G \) is a 3-colorable graph with \( n \) vertices, i.e., it is possible to color the vertices of \( G \) with 3 colors such that the endpoints of every edge have distinct colors. Design a polynomial time algorithm that colors vertices of \( G \) with \( O(\sqrt{n}) \) many colors.