COMS20010 — Problem sheet 2

This problem sheet covers week 2, focusing on interval scheduling and graphs.

1. [$\star\star\star$] In lectures we showed that substituting the greedy heuristic in our interval scheduling algorithm GreedySchedule with "add the compatible request with the earliest starting time" or "add the compatible request which takes least total time" breaks the algorithm. Show that the greedy heuristic "add the compatible request which renders fewest other requests incompatible" would also fail.

2. [$\star\star$] You are trying to check whether a log file contains a specific sequence of events, some of which may be duplicates. Since the log file contains records for the whole system, the events may not occur consecutively, but you know they will occur in order. Formally, you are given a *key sequence* of events $a_1, \ldots, a_m$ and a *log sequence* of events $b_1, \ldots, b_n$ with $n \geq m$, and you are able to check whether two events are equal in $O(1)$ time. Give a greedy algorithm to check whether $b_1, \ldots, b_n$ contains a *key subsequence* — indices $i_1 < i_2 < \cdots < i_m \in [n]$ such that $b_{i_j} = a_j$ for all $j \in [m]$ — and to output a key subsequence if one exists. Your algorithm should run in $O(n)$ time; prove it works.

3. [$\star\star\star\star$] A *closed* Euler walk is an Euler walk from a vertex to itself. Suppose G is a graph with exactly two connected components C_1 and C_2, each of which has more than one vertex. Suppose the graphs induced by C_1 and C_2 each have a closed Euler walk. What is the least number of edges we can add to G to give it a **closed** Euler walk?

4. [$\star\star\star\star$] You are consulting for a trucking company that does a large amount of business shipping packages from New York to Boston. The volume is high enough that they have to send a number of trucks each day between the two locations. Trucks have a fixed limit W on the maximum weight they are allowed to carry. Boxes arrive at the New York station one by one, and each package i has a weight $w_i \leq W$. The trucking station is quite small, so at most one truck can be at the station at any time. Company policy requires that boxes are shipped in the order they arrive; otherwise, a customer might get upset upon seeing a box that arrived after his make it to Boston faster. At the moment, the company is using a simple greedy algorithm for packing: they pack boxes in the order they arrive, and whenever the next box does not fit, they send the truck on its way.

   But they wonder if they might be using too many trucks, and they want your opinion on whether the situation can be improved. Perhaps one could decrease the number of trucks needed by sometimes sending off a truck early, allowing the next few trucks to be better packed?

   Prove that this is not the case — that for any given number k of packages, the greedy algorithm they are currently using minimises the number of trucks they need subject to their other constraints. (**Hint:** Use an argument like the one we used for GreedySchedule...)

5. The *complement* of a graph $G = (V, E)$ is the graph $G^c = (V, \overline{E})$, where $\overline{E} = \{\{u, v\} \colon u, v \in V, u \neq v\} \setminus E$. A graph is *self-complementary* if it is isomorphic to its complement. Show that:
   (a) [$\star$] a four-vertex path and a five-vertex cycle are both self-complementary;
   (b) [$\star\star\star$] every self-complementary graph is connected;
   (c) [$\star\star\star\star$] if G is self-complementary, then $|V(G)| \equiv 0$ or $1 \bmod 4$;
   (d) [$\star\star\star\star$] every self-complementary graph on $4k + 1$ vertices has a vertex of degree $2k$.

6. Consider a variant of the interval scheduling problem where we have multiple "satellites" available, and wish to satisfy **all** our requests while using as few of them as possible. Formally: writing our input as $\mathcal{R} = [(s_1, f_1), \ldots, (s_n, f_n)]$, instead of finding a maximum compatible set of requests, we must partition $\mathcal{R}$ into as few disjoint compatible sets as possible.

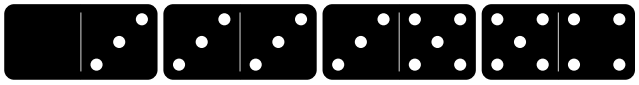
(a) [★★★★★] Prove that the following greedy algorithm returns the correct answer. (**Hint:** Rather than proving optimality directly, try to find a nice lower bound on the size of a minimum partition, and show the algorithm produces something which matches it.)

Algorithm: GreedyPartition

```
1 begin
2     Sort R according to start time, so that s_1 ≤ ··· ≤ s_n.
3     Initialise A_1, ..., A_n = [].
4     for i ∈ {1, ..., n} do
5         Find the least j such that (s_i, f_i) is compatible with A_j.
6         Append (s_i, f_i) to A_j.
7     Return the collection of non-empty lists A_j.
```

(b) Is the sorting step in line 2 necessary?

7. [★★★ and a half] A *numbered domino* is a rectangle divided into two halves, with a number on each half. A standard "double six" set of numbered dominoes contains one domino with each possible pair of numbers from zero to six, for a total of 28. Is it possible to lay them all out in a line so that each adjacent pair of dominoes agrees, as shown below for four dominoes? What about a "double k" set, which contains one domino with each possible pair of numbers from zero to $k \in \mathbb{N}$? (**Hint:** I will never ask a question like this unless there's a way to solve it quickly with pencil and paper.)



8. [★★★★★] Give an example of a self-complementary graph (see Question 3) with infinitely many vertices.