Algorithms. Exercises from Week 2

1. Would it be correct to solve Interval Partitioning as follows: "Compute a largest set of pairwise disjoint intervals (using a correct algorithm for Interval Scheduling), remove these intervals, and iterate this procedure until no interval is left over." Give a correctness proof or a counterexample.

2. We are given n intervals $[l_i, r_i]$. We say that a point (a real number) x hits an interval $[l_i, r_i]$ if $l_i \leq x \leq r_i$. The problem is to find a smallest number of points that hit all n given intervals. Give a greedy algorithm, prove its correctness, and analyze its time.

(This problem appears in scheduling, but also in other contexts. For instance, the intervals describe suitable temperatures to store n goods, and we want to use a minimum number of rooms and cool them to selected temperatures, such that all goods can be stored.)

3. A group of n witches, indexed $1, \ldots, n$, is sitting in front of n watches indexed $1, \ldots, n$, too. Due to obstacles between witches and watches, each witch can see only a subset of the watches. (We know which watch is visible to which witch.) The witches cannot move. Now the n witches together wish to watch all n watches. But each witch can watch *only one* of her visible watches at a time. So the problem they have is: Which witch should watch which watch, such that each watch is watched by some witch?

The witches have a "greedy" mind-set and proceed as follows: Witch 1 selects the watch with smallest index which is visible to her. For $i = 2, \ldots, n$, witch i selects the watch with smallest index which is visible to her and has not already been selected by any of the previous witches $1, \ldots, i-1$.

Give a counterexample, as small as possible, witnessing that the witches' greedy algorithm does not work. That is, construct an instance such that a solution exists, but the witches fail to watch all watches.

(The problem has applications that are more serious. If you like, you may think further: What might be a better algorithm?)

Exercise 4 on the reverse page:

4. Binomial coefficients $C(n,k) := \binom{n}{k}$ can be computed by the recursion $C(n,k) = C(n-1,k) + C(n-1,k-1)$, with the base cases $C(n,0) = 1$ and $C(n,n) = 1$ for all n. Given n and k, we want to compute $C(n,k)$. How much time is needed if we really do the recursive calls? Is the time then polynomial or exponential in n and k? And, in comparison, how much time is needed if we memoize the already computed $C(i,j)$ values?