2.23. An array $A[1 \ldots n]$ is said to have a *majority element* if more than half of its entries are the same. Given an array, the task is to design an efficient algorithm to tell whether the array has a majority element, and, if so, to find that element. The elements of the array are not necessarily from some ordered domain like the integers, and so there can be no comparisons of the form "is $A[i] > A[j]$?". (Think of the array elements as GIF files, say.) However you *can* answer questions of the form: "is $A[i] = A[j]$?" in constant time.

(a) Show how to solve this problem in $O(n \log n)$ time. (*Hint:* Split the array A into two arrays A_1 and A_2 of half the size. Does knowing the majority elements of A_1 and A_2 help you figure out the majority element of A? If so, you can use a divide-and-conquer approach.)

[Hint: Suppose A has a majority element x. How many copies of x need to be in A1 or A2? What does this say about the majority of A1 and A2?]

6.22. Give an $O(nt)$ algorithm for the following task.

Input: A list of n positive integers $a_1, a_2, \ldots, a_n$; a positive integer t.
Question: Does some subset of the a_i's add up to t? (You can use each a_i at most once.)

If you select a low-stress job for your team in week i, then you get a revenue of $\ell_i > 0$ dollars; if you select a high-stress job, you get a revenue of $h_i > 0$ dollars. The catch, however, is that in order for the team to take on a high-stress job in week i, it's required that they do no job (of either type) in week $i-1$; they need a full week of prep time to get ready for the crushing stress level. On the other hand, it's okay for them to take a low-stress job in week i even if they have done a job (of either type) in week $i-1$.

The problem. Given sets of values $\ell_1, \ell_2, \ldots, \ell_n$ and $h_1, h_2, \ldots, h_n$, find a plan of maximum value. (Such a plan will be called *optimal*.)

Example. Suppose $n = 4$, and the values of ℓ_i and h_i are given by the following table. Then the plan of maximum value would be to choose "none" in week 1, a high-stress job in week 2, and low-stress jobs in weeks 3 and 4. The value of this plan would be $0 + 50 + 10 + 10 = 70$.

	Week 1	Week 2	Week 3	Week 4
ℓ	10	1	10	10
h	5	50	5	1

5) Given a directed graph G=(V,E), a subset E' of edges is called a *feedback arc set* if the removal of edges E' makes G acyclic.

The Feedback Arc Set Problem is the following: Given a directed graph G and a number b, decide if there is feedback arc set of size at most b.

Show that this problem is in NP.

6) (p 497 Kleinberg-Tardos book) Consider the decision version of the **Traveling Salesman Problem (TSP)**: You are given the **complete** undirected graph where each edge *e* has length ℓ_e, and a distance target D, and you need to decide whether you can visit all nodes in the graph and come back to the starting point traversing length at most length D.

a) Show that **TSP** is in NP

b) Show a polynomial-time reduction of the **Hamiltonian Cycle Problem (HAM)** (is there a cycle visiting all nodes without traversing a node twice, except the first node) to the **Traveling Salesman Problem** (that is, if have polynomial-time algorithm for solving **TPS** can solve **HAM** in polynomial-time).

c) It is known that **HAM** is NP-complete. Does part (b) imply that **TSP** is NP-complete?

Livro Kleinberg-Tardos, Capitulo 8: Exercicios 3 [dica: problema Set Cover é NP-completo]

[O Capitulo 8 do livro tem dezenas de exemplos provando problemas em NP, NP-completo, vale a pena dar uma olhada]