Assignment #4:
Due: June 28
Proofs or counter-examples absolutely necessary for this assignment!

1. Let $G = [V, E]$ be an undirected graph. We want to check if it is connected. The only questions that we are allowed to ask are of the form: "Is there an edge between vertices $i$ and $j$?". Using an adversary argument show that any correct deterministic algorithm to decide if $G$ is connected must ask $\Omega(n^2)$ questions.

2. Consider the activity selection problem discussed in 16-1 of your book. For each of the following greedy algorithms, either prove that the algorithm solves the problem correctly or give a counter-example:

   (a) Select the activity with the least duration among those that are compatible with previously selected activities at each step of the algorithm.

   (b) Select the activity with the maximum starting time among those that are compatible with previously selected activities at each step of the algorithm.

3. The following problem is known in the literature as the knapsack problem: We are given $n$ objects each of which has a weight and a value. Suppose that the weight of object $i$ is $w_i$ and its value is $v_i$. We have a knapsack that can accommodate a total weight of $W$. We want to select a subset of the items that yields the maximum total value without exceeding the total weight limit.

   (i) If all $v_i$ are equal, what would the greedy algorithm yield? Is this optimal?

   (ii) If all $w_i$ are equal, what would the greedy algorithm yield? Is this optimal?

   (iii) How should the greedy algorithm be designed in the general case? Is this optimal? [Be careful to distinguish between two versions of the problem: in one we are allowed to select fractional items and in the other we are not allowed to do this.]

4. Consider the following generalization of a scheduling example done in class: We have $n$ customers to serve and $m$ identical machines that can be used for this (such as tellers in a bank). The service time required by each customer is known in advance: customer $i$ will require $t_i$ time units ($1 \leq i \leq n$). We want to minimize $\sum_{i=1}^{n} C_i(S)$, where $C_i(S)$ represents the time at which customer $i$ completes service in schedule $S$. How should the greedy algorithm work in this case? Is it guaranteed to produce optimal solutions?
5. Consider the following problem: we have \( n \) boxes one of which contains the object we are looking for. The probability that the object is in the \( i^{th} \) box is \( p_i \) and this value is known at the outset. It costs \( c_i \) to look in the box \( i \) and these values are also known. The process is to look in some box; if the object is found, the process stops; if not, we look in some other box and so on till the object is found.

Describe the greedy algorithm for this problem that works. What is the time complexity of the algorithm?

To show that the algorithm works, you will need to answer the following:

(a) If we look into box \( i \) first and don’t find the object, what are the new probabilities for finding the object in box \( j \neq i \)? (These are called conditional probabilities).

(b) Consider two scenarios: (I) First look in box \( i \) and if the object is not found, look in box \( j \). (II) First look in box \( j \) and if the object is not found, look in box \( i \). In each case, what is the probability of finding the object in box \( k \neq i \) or \( j \), if we don’t find the object in either box \( i \) or box \( j \)? Are they the same or are they different?

(c) Show that the algorithm works.

6. Challenge Problem I [not to be turned in]: Show that for any binary tree with \( k \) leaf nodes, the sum of the depth of the leaf nodes is at least \( k \lg k \). Using this show that the average time complexity of any sorting algorithm based on comparisons is \( \Omega(n \lg n) \).