Problem Set 6

This problem set is due **Thursday, December 6** at **11:59PM**.

Code for problem 4, as well as a LaTeXtemplate for the written problems, have been posted to the course website. Solutions should be turned in at `https://alg.csail.mit.edu` (our submission site).

Programming questions will be graded on a collection of test cases (including example test cases to help you debug). Unless you see an error message, *you will be able to see your grade immediately*. Your grade will be based on the number of test cases for which your algorithm outputs a correct answer within certain time and space bounds. You may submit as many times as you'd like, and only the final submission counts! **Therefore, make sure your final submission is what you want it to be.**

For written questions, full credit will be given only to correct solutions that are described clearly *and concisely*.

Problem 6-1. [20 points] **Houston, are you there?**
The space probe "Tranquility" (a cousin of "Curiosity") is recording signals from the outer space. It accumulates a sequence of n signal samples with amplitudes $x_1, x_2, \ldots, x_n$, where n is very large and all x_i's are non-negative. Once it gets all n samples, it sends the data back to Earth. However, most of the signal is just background noise, with only occasional bursts of activity. To save the energy, the probe identifies at most t disjoint time intervals $I_1, I_2, \ldots, I_t$ of length k, each of the form $[a_i, a_i + k - 1]$ where $a_i \in \{1, \ldots, n\}$ (both t and k are input parameters). The intervals maximize

$$\sum_{i=1}^{t} \sum_{j \in I_i} x_j$$

The ship sends only the $k \cdot t$ samples from the identified intervals. Design and argue the correctness of $O(n \cdot k \cdot t)$ algorithm that identifies the intervals for which the probe will send data back to Earth. Note that the intervals are **not** part of the input, i.e. it is your job to determine them. You may assume that $n > k > t$.

Problem 6-2. [20 points] **Going nuts**

A squirrel sits at the bottom of a tree. There are plenty of nuts on the tree, and the squirrel wants to eat as many nuts as it can. However, the squirrel only has time to climb k feet total, in order to be able to make it on time to the 6.006 office hours (it is an MIT squirrel).

Help the squirrel design the optimal route that is k feet long (including all movements going up and down the tree) and maximizes the number of nuts found. The tree is modeled (gasp!) as a tree, with nuts in some of its vertices (there is a total of n vertices). The squirrel starts and ends at the root of the tree and it knows the number of nuts in each vertex. For simplicity, you can assume that the tree edges are unweighted (each edge is 1 foot long), and that the tree is binary (the tree grows next to the EECS department). In order to get full credit, your algorithm should run in $O(n \cdot k^2)$ time and you should argue its correctness.

Problem 6-3. [20 points] **Florence + the Machine**
Florence is given a machine and a list of n tasks to perform on it. Each of the tasks is associated with the deadline $d_i \in \mathbb{N}$ by which it needs to get done, profit p_i and time it takes to perform it, t_i ($t_i \in 1, \ldots, k$, $\forall i \in \{1, n\}$ where k is a given parameter). Florence only gets paid p_i dollars if she completes the task before the deadline, otherwise she gets 0.

(a) [5 points] Prove that if we have a sequence of tasks that can all be accomplished by their respective deadlines, that without the loss of generality they can be executed in the increasing order of their deadlines.

(b) [15 points] Design and argue the correctness of $O(n^2 \cdot k)$ algorithm that will return the maximal profit Florence can achieve.

Problem 6-4. [40 points] **Optimal parenthesization**
Given an array of n numbers, your goal is to determine the largest value that can be obtained by interspersing parentheses, multiplication signs, addition signs and subtraction signs between them. Fill in the code for a function `find_largest_value(numbers)` for this problem. Your code should be able to handle a 100-element list in about a second and pass the following test cases:

```
# An optimal parenthesization: (1 + 2) * (3 * (4 * 5)) = 180
assert 179.99 < find_largest_value([1, 2, 3, 4, 5]) < 180.01

# An optimal parenthesization: (-1 - (-2 * -3)) * -4 = 28
assert 27.99 < find_largest_value([-1, -2, -3, -4]) < 28.01

# An optimal parenthesization: (-100 + 20) * (0.5 - 0.4) = -8
assert -8.01 < find_largest_value([-100, 20, 0.5, 0.4]) < -7.99
```