Homework set 1

Note: the homework sets are not for submission. They are designed to help you prepare for the quizzes. It is highly recommended that you solve all problems and write your solutions down formally. Whenever a question asks to design an algorithm for a problem, you should prove its correctness.

1. Consider the following variation of the Interval Scheduling Problem. You have a processor that can operate 24 hours a day, every day. People submit requests to run daily jobs on the processor. Each job j comes with a start time s_j and an end time t_j. If the job is accepted, then it must run during the time interval $[s_j, t_j)$ every day. Notice however that it is possible that s_j occurs before midnight, and t_j after midnight. Given the input set J of jobs, the goal is to accept as many jobs as possible, subject to the constraint that a processor can run at most one job at a time. Give an efficient algorithm to solve this problem and analyze its running time.

2. We are given a set J of n jobs. The execution of each job consists of two phases. First, it must be pre-processed on a supercomputer, and then it must be finished on a regular computer. Each job j is associated with parameters $p_1(j)$ - the processing time it requires on the supercomputer, and $p_2(j)$ - the processing time it requires on a regular computer. We only have one supercomputer, which can only execute one job at a time. However, we have an unbounded number of regular computers, that can execute any number of jobs simultaneously. Given a schedule S, the finish time $C(S)$ of the schedule is the earliest time by which all the jobs have been completed. Our goal is to find a schedule S of all jobs, with minimum finish time $C(S)$. Design an efficient algorithm, prove its correctness, and analyze the running time.

3. We are given a set of n points $X = \{x_1, \ldots, x_n\}$ on the real line. Design an algorithm that finds a minimum-cardinality set of unit-length closed intervals, that cover all points in X. Prove the algorithm's correctness and analyze its running time.

4. A string w of parentheses '(' and ')' is *balanced* if it satisfies one of the following conditions:

   - w is an empty string.
   - $w = (x)$ for some balanced string x
   - $w = xy$ for some balanced strings x and y.

   For example, the string $w = ((())()())(()())()$ is balanced, because $w = x, y$, where $x = ((())()())$ and $y = (()())()$.

   (a) Describe and analyze an algorithm to determine whether a given string of parentheses is balanced.

   (b) Describe and analyze a greedy algorithm to compute the length of a longest balanced subsequence of a given string of parentheses. Recall that a string s of length k is a subsequence of a string w, iff there are k indices $i_1 < i_2 < \ldots < i_k$, such that $s[i_j] = w[j]$ for all $1 \leq j \leq k$.

   For both problems, your input is an array $w[1..n]$, where for each i, either $w[i] =$'(', or $w[i] =$')'. Both of your algorithms should run in $O(n)$ time.

5. Suppose we have an alphabet Σ with 2^k characters, and a string in which all characters are almost equally common. That is, if $f(x)$ denotes the frequency of character x, then:

$$\min_{x\in\Sigma} f(x) > \max_{x\in\Sigma} f(x)/2.$$

How does the Huffman tree look like? What is its cost? Prove your answer.