Problem 1. (10 points)

In this problem we are given a text of size 2n and a pattern of size n and we want to compute for each position in the text the number of matched symbol with the pattern. A brute force approach would consist in aligning the pattern at each position in the text and computing the number of matches, for an overall $O(n^2)$ time. Here we show that one can solve this problem using the FFT.

Formally, let $t = a_0a_1a_2 \cdots a_{2n-1}$ and $p = b_0b_1b_2 \cdots b_{n-1}$ be two strings over an alphabet $\Sigma$ of size $\sigma$ (here we assume that $\sigma$ is not a constant). We define an array $C$ as follows $C(i) = \sum_{k=0}^{n-1} equal(a_{i+k}, b_k)$, for $i \in \{0, 1, \ldots, n\}$, where $equal(\alpha, \beta)$ is 1 if $\alpha = \beta$, zero otherwise. For example, if $t = \text{abracada}$ and $p = \text{abaa}$ then $C(0) = 3$, $C(1) = 1$, $C(2) = 1$, $C(3) = 2$, $C(4) = 1$. Observe that $C(i)$ gives the number of matched symbols when $p$ is aligned with the text at position $i$.

For each $\gamma \in \Sigma$ let us define $C_\gamma(i) = \sum_{k=0}^{n-1} equal_\gamma(a_{i+k}, b_k)$ for $i \in \{0, 1, \ldots, n\}$, where $equal_\gamma(\alpha, \beta)$ is 1 if $\alpha = \beta = \gamma$, zero otherwise. Give an $O(n \log n)$ algorithm to compute one of the arrays $C_\gamma$ based on convolution. This implies an efficient algorithm to compute $C$. What is the complexity as a function of $n$ and $\sigma$?

Answer: For a fixed $\gamma$ compute the binary vectors $A_\gamma$ and $B_\gamma$ as follows. We define $A_\gamma(i) = 1$ if $a_i = \gamma$ and 0 otherwise, for all $0 \leq i \leq 2n - 1$. Similarly, $B_\gamma(i) = 1$ if $b_i = \gamma$ and 0 otherwise, for all $0 \leq i \leq n - 1$. It is easy to see that $C_\gamma$ can be obtained from the circular convolution of binary vector $A_\gamma$ and $B_\gamma$. More specifically, $n + 1$ of the coefficients of $A_\gamma \otimes B_\gamma^R$ are exactly the values of $C_\gamma$, where $B_\gamma^R$ indicates the reverse of vector $B$.

In the example above, if $\gamma = a$ we get $A_a = 10010101$ and $B_a = 1011$. The vector $C_a$ corresponds to $n + 1$ coefficients for the circular convolution of $A_a \otimes B_a^R0000 = 10010101 \otimes 11010000$.

Therefore we can compute one of the $C_\gamma$ in $O(n \log n)$ using the FFT. Since $C(i) = \sum_{\gamma \in \Sigma} C_\gamma(i)$, this implies an $O(\sigma n \log n)$ algorithm to compute $C$.

Problem 2. (10 points)

You are given a list of $n$ jobs $j_1, j_2, \ldots, j_n$ each with a time $t(j)$ to perform a job and a weight $w(j)$. We want to order the jobs as $j_{\sigma(1)}, j_{\sigma(2)}, \ldots, j_{\sigma(n)}$ in such a way that the quantity

$$\sum_{i=1}^{n} w(j_{\sigma(i)}) \left( \sum_{k=1}^{i} t(j_{\sigma(k)}) \right)$$

is minimized. In other words, we want to minimize the weighted sum of the time each job has to wait before being executed. Give an efficient greedy algorithm for this problem, analyze the complexity, and prove its optimality.

Answer: The greedy algorithm computes $r(j_i) = w(j_i)/t(j_i)$ for all $i$. Sort the $r(j_i)$ largest to smallest, and schedule the job according to this order. Time complexity is $O(n \log n)$. 
Let’s start with the greedy choice. Let \( j_i \) the first choice, that is \( r(j_i) \geq r(j_i') \) for every \( i' \). We need to prove that \( j_i \) is the first choice in an optimal solution. Let \( j_{\sigma(1)}, j_{\sigma(2)}, \ldots, j_{\sigma(n)} \) be any schedule, and let \( k \) be such that \( \sigma(k) = i \); i.e., \( j_i \) is the \( k \)-th job performed in the schedule. We claim that \( j_{\sigma(1)}, \ldots, j_{\sigma(k-2)}, j_i, j_{\sigma(k-1)}, j_{\sigma(k+1)}, \ldots, j_{\sigma(n)} \) is a schedule whose cost is at most that of the original schedule. In other words, if we swap \( j_i = j_\sigma(k) \) and the job preceding it only makes the cost smaller. This is because the delays for all jobs except \( j_i \) and \( j_{\sigma(k-1)} \) stay the same, and the delay for \( j_i \) decreases by \( t(j_\sigma(k-1)) \) and that for \( j_{\sigma(k-1)} \) increases by \( t(j_i) \). Thus the change in cost is

\[
w(j_{\sigma(k-1)})t(j_i) - w(j_i)t(j_{\sigma(k-1)}) = t(j_i)t(j_{\sigma(k-1)})(r(j_{\sigma(k-1)}) - r(j_i)) \leq 0
\]
since \( r(j_i) \geq r(j_{\sigma(k-1)}) \). Repeating this process \( k \) times, we can move \( j_i \) to be the first job without increasing the cost. Thus, there is an optimal schedule whose first job is \( j_i \).

Regarding the optimal substructure, let us assume that \( O = j_{\sigma(1)}, j_{\sigma(2)}, \ldots, j_{\sigma(n)} \) is an optimal solution for the set \( P \) containing the original \( n \) jobs. Note that the overall cost for \( P \) is

\[
w(j_{\sigma(1)})t(j_{\sigma(1)}) + w(j_{\sigma(2)})\left(t(j_{\sigma(1)}) + t(j_{\sigma(2)})\right) + \ldots + w(j_{\sigma(n)})\left(t(j_{\sigma(1)}) + \ldots + t(j_{\sigma(n)})\right) = \left( w(j_{\sigma(1)}) + w(j_{\sigma(2)}) + \ldots + w(j_{\sigma(n)}) \right) t(j_{\sigma(1)}) + \sum_{i=2}^{n} \left( w(j_{\sigma(i)}) \sum_{k=1}^{i} t(j_{\sigma(k)}) \right)
\]

Hence the optimal cost is \( t(j_i) \sum_{i' \neq i} w(j_{i'}) + \text{the cost of the other jobs if } j_i \text{ weren’t run, and the first term does not depend on the order the others were run in. Clearly, the solution } j_{\sigma(2)}, \ldots, j_{\sigma(n)} \text{ must be optimal solution for the set of } P - \{j_i\} \text{ jobs, otherwise we could create a solution which is better than } O, \text{ which creates a contradiction.}

**Problem 3.** (15 points)

Consider the following REVERSE-DELETE algorithm for the Minimum Spanning Tree (MST) problem. Start with a connected edge-weighted graph \( G = (V, E) \). Consider edges in order of decreasing weight (breaking ties arbitrarily). When considering an edge \( e \), delete \( e \) from \( E \) unless doing so would disconnect the current graph. Analyze the complexity of REVERSE-DELETE and prove that it computes an (optimal) MST.

**Answer:** First, let us prove that REVERSE-DELETE builds a spanning tree of \( G \). This is true because each time we remove an edge, REVERSE-DELETE guarantees that removal does not disconnect the graph. Now, suppose by the way of contradiction the output of REVERSE-DELETE contains a cycle \( C \). Consider the most expensive edge on \( C \), which would be the first encountered by the algorithm. This edge should have been removed, since its removal would not have disconnected the graph, and this contradicts the behaviour of REVERSE-DELETE.

Second, we need to prove that the spanning tree is optimal. Consider any edge \( e \) removed by REVERSE-DELETE. At the time it is removed, it lies on a cycle \( C \), and since it is the first edge encountered by the algorithm in decreasing order, it must be the most expensive edge on \( C \). Thus, by the property below (easy to prove), \( e \) does not belong to any MST, therefore it is safe to remove.
Fact: Let $G = (V, E)$ be a weighted undirected graph and let $C$ be a simple cycle in $G$. Let $e$ be an edge of highest weight in $C$. Prove that there is a minimum spanning tree $T$ of $G$ that does not contain $e$. 