Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
S. Daum, H. Ghodselahi, O. Saukh November 5, 2013

Algorithm Theory, Winter Term 2013/14
Problem Set 2

hand in by Thursday, November 14, 2013

Exercise 1: Polynomial Multiplication using FFT

Compute the coefficient representation of $(p(x))^2$ by using the FFT algorithm.

$$p(x) = x^3 - 3x^2 + 2x + 1$$

Remark: Instead of using exact numbers for the point-wise evaluations (which involves irrational numbers) you can also round numbers to, say, 3+ digits after the decimal point. This also reflects what happens in an implemented version of FFT, as algebraic representations do not lead to an $O(n \log n)$ running time. Those unfamiliar with complex numbers should ask fellow students for some help - calculating roots of unity and multiplying 2 complex numbers is all you need for this exercise.

Exercise 2: Distribution of Sum

Let A and B be two sets of integers between 0 and $n-1$, i.e., $A, B \subseteq \{0, \ldots, n-1\}$. We define two random variables X and Y, where X is obtained by choosing a number uniformly at random from A and Y is obtained by choosing a number uniformly at random from B. We further define the random variable $Z = X + Y$. Note that the random variable Z can take values from the range $\{0, \ldots, 2n-2\}$.

(a) Give a simple $O(n^2)$ algorithm to compute the distribution of Z. Hence, the algorithm should compute the probability $\Pr(Z = z)$ for all $z \in \{0, \ldots, 2n-2\}$. How fast can your algorithm be in case A and B have a low cardinality?

(b) Can you get a more efficient algorithm to compute the distribution of Z? You can use algorithms discussed in the lecture as a black box. What is the time complexity of your algorithm?

Hint: Try to represent A and B using polynomials.

Remark: Exercise 2 was an exam question in Fall 2012.

Exercise 3: Scheduling

The wildly popular Spanish search engine "El Goog" needs to do a serious amount of computation every time it recompiles its index. Fortunately, the company has a single supercomputer (SC) at its disposal, together with an essentially unlimited supply of high-end PCs.
The computation can be broken into n distinct jobs $J_1, J_2, ..., J_n$. Each job J_i needs to be *preprocessed* for p_i time units on the SC, before it can (and should be) *finished* within f_i time units on one of the PCs.
Due to the large amount of PCs all the finishing of the jobs can be done in parallel, however, the SC has to run the jobs sequentially. El Goog needs a scheduling of the jobs on the SC that minimizes the *completion time* of the computation, which is the earliest point in time for which all jobs have been finished processing on the PCs.
Find a fast algorithm that computes an optimal scheduling and prove its correctness.

Exercise 4: Matroids

We have defined matroids in the lecture. For a matroid (E, I), a maximal independent set $S \in I$ is an independent set that cannot be extended. Thus, for every element $e \in E \setminus S$, the set $S \cup \{e\} \notin I$.

a) Show that all maximal independent sets of a matroid (E, I) have the same size. (This size is called the rank of a matroid.)

b) Consider the following greedy algorithm: The algorithm starts with an empty independent set $S = \emptyset$. Then, in each step the algorithm extends S by the minimum weight element $e \in E \setminus S$ such that $S \cup \{e\} \in I$, until S is a maximal independent set. Show that the algorithm computes a maximal independent set of minimum weight.

c) For a graph $G = (V, E)$, a subset $F \subseteq E$ of the edges is called a forest iff (if and only if) it does not contain a cycle. Let $\mathcal{F}$ be the set of all forests of G. Show that $(E, \mathcal{F})$ is a matroid. What are the maximal independent sets of this matroid?