On this problem set you must work on your own

Problem 1. CLRS Problem 30-5. Polynomial evaluation at multiple points.

Problem 2. MAX-2-SAT. A 2-CNF formula consists of a conjunction of clauses, where each clause contains at most 2 literals. A literal is either a Boolean variable $x_i$ or its negation $\neg x_i$. For example $(x_1) \land (\neg x_1 \lor x_2)$ is a 2-CNF formula.

The MAX-2-SAT problem is the problem of finding, for a given 2-CNF formula, an assignment that satisfies as many clauses as possible.

1. Prove that MAX-2-SAT can be efficiently written as a strict quadratic program. Hint: For a MAX-2-SAT instance in variables $x_1, \ldots, x_n \in \{0, 1\}$, let $y_0, y_1, \ldots, y_n$ be variables $\in \{-1, 1\}$. Think of a variable $x_i, i \in \{1, \ldots, n\}$, in the MAX-2-SAT instance as being true if $y_i = y_0$ and false otherwise. Prove that the value of MAX-2-SAT can be written as

$$\max \sum_{0 \leq i < j \leq n} a_{i,j}(1 + y_i y_j) + b_{i,j}(1 - y_i y_j),$$

where the $a_{i,j}$ and $b_{i,j}$ are constants computable in polynomial time given the MAX-2-SAT instance.

2. Consider the vector relaxation of the above. Recall this is the program where each variable $y_i$ is replaced with a vector $v_i$ (in $n+1$ dimensions) of length 1, and $y_i \cdot y_j$ is replaced with the inner product of the two vectors. You can assume that such programs can be solved exactly.

Give a randomized algorithm that produces a better approximation than 2 for MAX-2-SAT. Use (a) the above relaxation to a vector program, (b) the fact that the inner product between two vectors of length 1 equals the cosine of the angle between them, and (c) the following two trigonometric inequalities that hold for an absolute constant $\alpha > 0.87$ and any angle $\theta \in [0, \pi]$:

$$\frac{\theta}{\pi} \geq \frac{\alpha}{2} (1 - \cos \theta), \quad 1 - \frac{\theta}{\pi} \geq \frac{\alpha}{2} (1 + \cos \theta).$$

Problem 3. Merging suffix arrays. In class we have seen a recursive procedure to compute a suffix array. The analysis had one missing detail. This problem asks you to complete that.

You are given as input an array $T[0..n - 1]$ of characters, a suffix array $A[0..(2n/3) - 1]$ for the suffixes of $T$ at positions $i \equiv 1, 2 \mod 3$, and a suffix array $B[0..n/3 - 1]$ for the suffixes of $T$ at positions $i \equiv 0 \mod 3$.

Show how to construct a suffix array $S[0..n - 1]$ for all the suffixes of $T$ in time $O(n)$. 
