TUTORIAL 1

1 Remainder of a sparse polynomial

In this exercise we are interested in computing a remainder of a sparse polynomial S after dividing by a polynomial D, where $S, D \in K[X]$. (Assume that operations in K have unit cost.)

1. Give an example showing that assuming that S is sparse does not lead to better bounds for the classical division algorithm.
2. What is the cost of an operation in $K[X]/(D(X))$?
3. Show that one can compute $X^N \mod D(X)$ in time $O((\deg D)^2 \log N)$. (Hint: use fast exponentiation.)
4. Assume that S has ω nonzero terms. Show that you get an algorithm of complexity $O(\omega(\deg D)^2 \log \deg S)$ which beats the classical division for ω at most $\frac{\deg S - \deg D}{\deg D \log \deg S}$.

2 Exact division

The goal of this exercise is to show that we can have a constant gain on (naive) polynomial division in the case where we know beforehand that the division is exact. As is always the case with constant gain, we'll need to argue in the end that our complexity model is sound. Let $A(X) = \sum_{k=0}^{2n-1} a_k X^k$ and $B(X) = \sum_{k=0}^{n-1} b_k X^k$.

1. Prove that one can compute $Q_\ell = \sum_{k=n-\ell}^{n} q_k X^k$ such that $\deg(A - BQ_\ell) < 2n - 1 - \ell$ using $\ell + 1$ divisions and $\ell(\ell+1)/2$ multiplications in K. *Note that we are not interested in the remainder $A - BQ_\ell$, only in Q_ℓ.*
2. Prove that the algorithm of **1**. can be used to compute $S_\ell = \sum_{k=0}^{\ell} s_k X^k$ such that $\mathrm{val}(A - S_\ell B) > \ell$ using $\ell + 1$ divisions and $\ell(\ell+1)/2$ multiplications in K. (Where $\mathrm{val}(P)$ is the smallest degree of monomial of P with nonzero coefficient.)
3. Assume that we know, for some reason, that $B|A$ and want to compute A/B. Use **1** & **2** to give an algorithm for this task, and compare this with the "schoolbook" division.
4. We only counted divisions and multiplications, but in a standard algebraic model, addition and subtraction also have a comparable cost. Does our result really make sense?

3 Multiplication of bivariate polynomials

Fact: Let $c_0, \dots c_d$ be $d + 1$ distinct elements of K and $Q_0, \dots Q_d \in K[X]$. There is a unique polynomial $P \in K[X, Y]$ of Y-degree at most d satisfying $P(X, c_i) = Q_i$ for every $i = 0, \dots d$.

Let us assume that we can efficiently find such P. Again, assume that operations in K have unit cost.

1. What is the cost of a naive multiplication of two bivariate polynomials A and B of X-degree at most D_1 and Y-degree at most D_2?
2. Give an algorithm that computes $A(X, c)$ for a given $c \in K$, with A of X-degree at most D_1 and Y-degree at most D_2. What is its cost?
3. Using the fact above, describe an algorithm for multiplying bivariate polynomials. (Which would, assuming that we have a fast algorithm for multiplication of polynomials of one variable, beat the naive multiplication.)

4 Integer division

Let $\beta = 2^w$ be the machine word; in the sequel, we shall assume that the processor, given as input $u \in [0, \beta^2 - 1]$ and $v \in [1, \beta - 1]$, can compute $\lfloor u/v \rfloor$ (the integer part of u/v).

Let $A = \sum_{i=0}^{n} a_i \beta^i$ and $B = \sum_{i=0}^{m} b_i \beta^i$ be the expansion of two positive integers in base β. We will prove that the following algorithm returns A/B and $A \mod B$.

Integer division

```
procedure GUESS(A, B)
    g ← ⌊(a_n β + a_{n−1})/b_m⌋
    return min(g, β − 1)
end procedure

procedure DIVIDE(A, B)
    while A ≥ B do
        q_{n−m−1} ← GUESS(A, B)
        A ← A − q_{n−m−1} β^{n−m−1} B
        while A < 0 do                                  ▷ Correction loop
            q_{n−m−1} ← q_{n−m−1} − 1
            A ← A + B β^{n−m−1}
        end while
    end while
    return (Σ q_k β^k, A)
end procedure
```

1. Prove that, up to multiplying A and B by suitable powers of 2, we can assume that $b_m \geq \beta/2$

2. Prove that, up to replacing A by $A - \beta^{n-m}B$, we can assume that $A < \beta^{n-m}B$.

   Remark: in the previous two questions, also explain how one should correct quotient and remainder in the end to get the result of the original division...

   Define the following loop invariant (for the outer while loop) ; we denote by A_k, Q_k the value of A, Q after the k-th iteration of the loop (where Q is the integer $\sum_i q_i \beta^i$ for all q_i's that have been defined so far).

   - $0 \leq A_k < \beta^{n-m-k}B$
   - $A_k + Q_k B = A$.

   where n and m are the maximal powers of β that appear in the original A and B (i.e. before the first loop).

3. Deduce that, if the loop invariant is correct, when we exit the loop (Q, A) are the quotient and the remainder of the Euclidean division of A by B.

4. Prove the second statement of the loop invariant, and the non-negativity part of the first one.

   We now turn to proving the loop invariant by induction – to simplify the notations we only deal with the first iteration, but the proof carries over to any iteration. We shall split the proof into two cases: either $g \leq \beta - 1$ in GUESS, or GUESS returns $\beta - 1$.

5. First subcase: $g \leq \beta - 1$. Prove that $q_{n-m-1} b_m \geq a_n \beta + a_{n-1} - b_m + 1$. Deduce that

$$A - q_{n-m-1} B \beta^{n-m-1} \leq \sum_{k=0}^{n-2} a_k \beta^k + (b_m - 1)\beta^{n-1},$$

and the last part of the loop invariant.

6. Second subcase: $g \geq \beta$; prove the last part of the loop invariant *(hint: use question 2)*.

We now estimate the number of iterations of the Correction loop.

7. Prove that we always have $\textsc{Guess}(A, B) b_m \leq a_n \beta + a_{n-1}$, and deduce that

$$A - \textsc{Guess}(A, B)\beta^{n-m-1} B > -\textsc{Guess}(A, B)\beta^{n-1}.$$

Deduce that the number of iterations of the Correction loop is at most 2 *(hint: use question 1)*.

8. Conclude on the correction and complexity of the algorithm.