TUTORIAL 6

1 Faster characteristic polynomial

Let A be an $n \times n$ matrix. In this exercise, we will denote by n^ω the number of operations in K needed to multiply two n by n matrices with coefficients in K. You will see in class that given a n by n matrix $M \in \mathcal{M}_n(K)$, we can compute M^{-1} using $O(n^\omega)$ operations in K (computing the inverse is asymptotically the same as multiplying).

1. Assume that v is a vector such that $v, Av, A^2v, \ldots, A^{n-1}v$ is a basis of K^n; then if B is the matrix with columns $v, Av, A^2v, \ldots, A^{n-1}v$, prove that $B^{-1}AB$ is a *companion matrix*, that is, a matrix of the following form.
$$C = \begin{bmatrix} 0 & 0 & \cdots & 0 & c_0 \\ 1 & 0 & \cdots & 0 & c_1 \\ 0 & 1 & \cdots & 0 & c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & c_{n-1} \end{bmatrix}.$$
2. If B is given, what is the cost of computing the characteristic polynomial of A using the previous question.
3. Explain why from an $n \times n$ matrix multiplication in time $O(n^\omega)$ we can deduce a $n \times m$ by $m \times k$ matrix multiplication algorithm in time $O(\max(n, m, k)^\omega)$
4. Define $w_0 = v, w_1 = (v, Av), w_2 = (v, Av, A^2v, A^3v), \ldots, w_k = (v, Av, A^2v, \ldots, A^{2^k-1}v)$
Prove that w_k can be computed in time $O(kn^\omega)$ for $k < \log n$.
5. Under the assumption that v exists and that you know it, give a $O(n^\omega \log n)$ algorithm for computing the characteristic polynomial of a square matrix.
6. Does there always exist a v as in Question 1 ?

2 Sylvester matrices

Let K be a field, and $P = \sum_{i=0}^{d_P} p_i X^i$, $Q = \sum_{i=0}^{d_Q} q_i X^i$ be two polynomials in $K[X]$ of respective degree d_P and d_Q. Put $D = d_P + d_Q$, define $v_P = (p_0, p_1, \ldots, p_{d_P}, 0, \ldots, 0) \in K^D$ and $v_Q = (q_0, q_1, \ldots, q_{d_Q}, 0, \ldots, 0) \in K^D$.

For $x = (x_0, \ldots, x_{D-1})$ a vector in K^D, define $C(x) = (0, x_0, \ldots, x_{D-2})$. The *Sylvester matrix* of P and Q is the matrix of size D whose colums are
$$(v_P, C(v_P), \ldots, C^{d_Q-1}(v_P), v_Q, C(v_Q), \ldots, C^{d_P-1}(v_Q)).$$

It is probably better illustrated on an example: if P has degree 2 and Q degree 3, then we have

$$S(P,Q) := \begin{pmatrix} p_0 & 0 & 0 & q_0 & 0 \\ p_1 & p_0 & 0 & q_1 & q_0 \\ p_2 & p_1 & p_0 & q_2 & q_1 \\ 0 & p_2 & p_1 & q_3 & q_2 \\ 0 & 0 & p_2 & 0 & q_3 \end{pmatrix}.$$

2.1 Solving linear systems

1. Let $v = (v_0, \ldots, v_{d_Q-1}, w_0, \ldots, w_{d_P-1}) \in K^D$. Compute $S(P,Q) \cdot v$ and express it in terms of the polynomials $V = \sum v_i X^i$ and $W = \sum w_i X^i$.
2. What is the best complexity you can achieve for computing a product $S(P,Q) \cdot v$ using fast arithmetic?
3. If P, Q are coprime, what is the best complexity you can achieve for solving the equation $S(P,Q) \cdot v = w$? Or computing the inverse of $S(P,Q)$?

2.2 Computing $\det(S(F,G))$

Recall simple facts about the resultant $\mathsf{Res}(F,G)$ for $F = \mathsf{LC}(F)\prod_t(x-u_i), G = \mathsf{LC}(G)\prod_i(x-v_i)$ for $u_i, v_i \in \bar{K}$, where $\mathsf{LC}()$ is the leading coefficient:

1. $\mathsf{Res}(F,G) = \mathsf{LC}(F)^{\deg G}\mathsf{LC}(G)^{\deg F}\prod_{i,j}(u_i - v_j)$
2. $\mathsf{Res}(F,G) = \mathsf{LC}(F)^{\deg Q}\prod_i G(u_i)$

1. Prove that for $F = GQ + R$:

$$\mathsf{Res}(F,G) = (-1)^{\deg F \deg G}\mathsf{LC}(G)^{\deg F - \deg R} \cdot \mathsf{Res}(G,R).$$

2. Using the above equality deduce an algorithm to compute $\det(S(F,G))$ and analyse its complexity.

3 Cauchy matrices

Let $\mathbf{a} = (a_i)_{0\leqslant i\leqslant n-1} \in K^n, \mathbf{b} = (b_i)_{0\leqslant i\leqslant n-1} \in K^n$. We assume that $a_i \neq b_j$ for all i,j and that $a_i \neq a_j$ and $b_i \neq b_j$ for $i \neq j$. The *Cauchy matrix* associated to these n-tuples is the matrix $C(\mathbf{a},\mathbf{b}) = (1/(a_i - b_j))_{0\leqslant i,j\leqslant n-1}$. The goal of this exercise is to find H, the inverse of C, and compute $C \cdot y$

1. Let $A_i(x) = \frac{A(x)}{A'(a_i)(x-a_i)}, B_i(x) = \frac{B(x)}{B'(b_i)(x-b_i)}$ be the fundamental polynomials of the Lagrangian interpolation with $A(x) = \prod_i(x-a_i), B(x) = \prod_i(x-b_i)$. Prove that

$$h_{i,j} = (a_j - b_i) \cdot A_j(b_i) \cdot B_i(a_j).$$

   In case C is symmetric, prove that

$$h_{i,j} = (a_j - b_i) \cdot A_j(b_i) \cdot A_i(b_j).$$

2. Conclude on the complexity of computing H.