1. Let p be a polynomial. Prove that there are infinitely many algebraic formulas representing p.

2. Define a *plus-minus formula* to be the same as an algebraic formula, except that we are also allowed to use the operation $-$. Let $\mathsf{L}^{\pm}(p)$ denote the size of the smallest plus-minus formula representing p. Prove that

$$\mathsf{L}^{\pm}(p) \leqslant \mathsf{L}(p) \leqslant 2\mathsf{L}^{\pm}(p)$$

for every polynomial p.

3. Let $\mathsf{L}^{\times}(p)$ denote the minimum number of $\times$ symbols in an algebraic formula computing p. Similarly, $\mathsf{S}^{\times}(p)$ denotes the minimum number of multiplication gates in an arithmetic circuit computing p.

   (a) Prove that if $\deg(p) = d$, then $\mathsf{L}^{\times}(p) \geqslant d - 1$.

   (b) Prove that if $\deg(p) = d$, then $\mathsf{S}^{\times}(p) \geqslant \log d$.

4. The *sparsity* of a polynomial $p(x)$, denoted $\mathrm{sp}(p)$, is the number of non-zero monomials in p. That is, if $p(x) = a_0 + a_1 x + \cdots + a_d x^d$, then $\mathrm{sp}(p)$ is the number of a_i which are non-zero.

   Prove that $\mathsf{S}(p) \leqslant O(\mathrm{sp}(p) \cdot \log \deg(p))$.

5. Let $\Psi_d(x) = 1 + x + x^2 + \cdots + x^d$.

   (a) Prove that $\mathsf{S}(\Psi_d) \leqslant O((\log d)^2)$.

   ⋆(b) Prove that $\mathsf{S}(\Psi_d) \leqslant O(\log d)$.

6. You may have noticed that I called $\mathsf{S}(p)$ the *circuit complexity* of p, but never told you what a circuit is. An algebraic circuit is an equivalent, but sometimes more convenient, way of thinking about straight-line programs. Read the definition in the lecture notes, and make sure you understand it!

⋆7. (a) Write down a formal definition of an algebraic circuit.

   (b) Prove that every algebraic circuit can be converted to a straight-line program of the same size, and vice versa.

   (c) Prove that an algebraic formula is the same as an algebraic circuit with no cycles (that is, the underlying undirected graph is a forest). Using this, conclude again that $\mathsf{S}(p) \leqslant \mathsf{L}(p)$ for every polynomial p.

✣⋆8. Read the proof of Theorem 3.1, which says that if $\mathsf{S}(f_d) \leqslant O(\log d)$, then integers can be efficiently factored. Ask me if you have questions about it!

⋆ means that a problem is hard.
? means that a problem is open.
✣ means that a problem is on a topic beyond the scope of the course.

1. Let $p(x_1, \ldots, x_n) = x_1 + \cdots + x_n$. Prove that $\mathsf{S}(p) \geqslant \Omega(n)$.

   [**Note:** This is a special case of Theorem 4.3, which we didn't prove. You should find a direct proof!]

2. Recall that $\mathsf{S}^\times(p)$ denotes the minimum number of multiplications used in any algebraic circuit computing p. Prove that $\mathsf{S}(x_1^d, \ldots, x_n^d) \geqslant \Omega(n \log d)$.

3. Let $p_1, \ldots, p_n$ be one-variable polynomials of degrees $d_1, \ldots, d_n$, respectively.

   (a) Prove that
   $$\mathsf{S}(p_1(x_1), \ldots, p_n(x_n)) \geqslant \sum_{i=1}^{n} \log d_i.$$
   Note that we are evaluating each p_i on a distinct variable x_i.

   (b) Prove a nearly matching upper bound, namely that
   $$\mathsf{S}(p_1(x_1), \ldots, p_n(x_n)) \leqslant 2 \sum_{i=1}^{n} \log d_i.$$

✣4. In this problem, we will see a sketch of Strassen's algorithm for matrix multiplication, which shows that $\mathsf{S}(\mathsf{MM}_k) \leqslant O(k^{\log 7}) \approx O(k^{2.807})$, beating the naive bound of k^3.

   (a) Given two 2×2 matrices
   $$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \qquad \text{and} \qquad \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix},$$
   define
   $$\begin{aligned} c_1 &= (a_{11} + a_{22}) \times (b_{11} + b_{22}) \\ c_2 &= (a_{21} + a_{22}) \times b_{11} \\ c_3 &= a_{11} \times (b_{12} - b_{22}) \\ c_4 &= a_{22} \times (b_{21} - b_{11}) \\ c_5 &= (a_{11} + a_{12}) \times b_{22} \\ c_6 &= (a_{21} - a_{11}) \times (b_{11} + b_{12}) \\ c_7 &= (a_{12} - a_{22}) \times (b_{21} + b_{22}). \end{aligned}$$
   Prove that
   $$AB = \begin{pmatrix} c_1 + c_4 - c_5 + c_7 & c_3 + c_5 \\ c_2 + c_4 & c_1 - c_2 + c_3 + c_6 \end{pmatrix}.$$

   (b) This is not a particularly efficient way of computing MM_2. However, note that we have computed MM_2 using only *seven* multiplications, rather than the *eight* that would be used in the usual way of computing MM_2.

(c) Let A, B be $k \times k$ matrices, where k is even. Write

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \qquad \text{and} \qquad \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix},$$

where A_{ij}, B_{ij} are $(k/2) \times (k/2)$ matrices. Define $C_1, \dots, C_7$ as above, except that now every $\times$ denotes multiplication of $(k/2) \times (k/2)$ matrices. Prove that

$$AB = \begin{pmatrix} C_1 + C_4 - C_5 + C_7 & C_3 + C_5 \\ C_2 + C_4 & C_1 - C_2 + C_3 + C_6 \end{pmatrix}.$$

(d) By recursing, come up with a way of computing MM_k whenever k is a power of two. Using this, prove that $\mathsf{S}(\mathsf{MM}_k) \leqslant O(k^{\log 7})$.

(e) Figure out how to extend this to work for all k.

✣5. **Note:** This problem requires some knowledge of multivariable calculus.

In this problem, you will explore the following theorem.

Theorem (Baur–Strassen)**.** *Let $p(x_1, \dots, x_n)$ be a polynomial. Then*

$$\mathsf{S}\left(\frac{\partial p}{\partial x_1}, \dots, \frac{\partial p}{\partial x_n}\right) \leqslant O(\mathsf{S}(p)).$$

This theorem was actually rediscovered in several different contexts by several different people, and is extremely useful in practice. For example, all of the modern work on AI requires rapid computation of gradients for multivariate optimization, and this theorem basically says that this can be done efficiently.

(a) Assuming the Baur–Strassen theorem is true, prove that $\mathsf{S}(t_{d,n}) \geqslant \Omega(n \log d)$ (Theorem 4.3 in the lecture notes).

⋆⋆(b) Prove the Baur–Strassen theorem.

Hint: Given a straight-line program computing p, add in its "mirror image" to compute partial derivatives of p.

If this hint is not enough, you can read a proof starting on page 78 here:
`https://www.math.ias.edu/~avi/PUBLICATIONS/ChenKaWi2011.pdf`

1. Construct a symbolic matrix whose determinant is $xy + xz + yz - 4$. (Or pick your favorite polynomial and construct a symbolic matrix with that determinant.)

2. If you know what eigenvalues are, do the rest of this problem. If not, read the rest of this problem and accept it as true.

   (a) Prove that if M is an $n \times n$ matrix with eigenvalues $\lambda_1, \dots, \lambda_n$, then M^d has eigenvalues $\lambda_1^d, \dots, \lambda_n^d$.

   (b) Recall that the *trace* of a square matrix is the sum of its diagonal entries. Prove that $\operatorname{tr}(M) = \lambda_1 + \cdots + \lambda_n$.

   (c) Prove that $\det(M) = \lambda_1 \cdots \lambda_n$.

3. In this problem, you will see one way of efficiently computing the determinant. For this problem, you will need the results of problem 2 (but you can do this problem without knowing what eigenvalues are; all you need about them is the results of problem 2).

   (a) Let $t_d(x_1, \dots, x_n) = x_1^d + \cdots + x_n^d$. Prove that if an $n \times n$ matrix has eigenvalues $\lambda_1, \dots, \lambda_n$, then
   $$\operatorname{tr}(M^d) = t_d(\lambda_1, \dots, \lambda_n).$$

   ⋆(b) Let $e_d(x_1, \dots, x_n)$ be the dth *elementary symmetric polynomial*, defined by
   $$e_d(x_1, \dots, x_n) = \sum_{\substack{S \subseteq \{1,2,\dots,n\} \\ |S|=d}} \prod_{i \in S} x_i.$$

   For example,
   $$e_2(x_1, x_2, x_3, x_4) = x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4.$$

   Prove the *Newton identities*, which state that for all $n \geqslant d \geqslant 1$, we have
   $$d \cdot e_d(x_1, \dots, x_n) = \sum_{i=0}^{d} (-1)^{i-1} e_{d-i}(x_1, \dots, x_n) t_i(x_1, \dots, x_n).$$

   (c) Come up with a way of efficiently computing the determinant (i.e. show that $\mathsf{S}(\det_k)$ is at most some polynomial in k).

4. (a) Let $p(x)$ be a degree-n polynomial in one variable. Prove that p is uniquely determined by the $n + 1$ numbers $p(0), p(1), \dots, p(n)$.

   (b) Strengthen part (a) as follows. Prove that for every i, there exist real numbers $\alpha_0^{(i)}, \alpha_1^{(i)}, \dots, \alpha_n^{(i)}$ such that the ith coefficient of p equals $\alpha_0^{(i)} p(0) + \cdots + \alpha_n^{(i)} p(n)$.

⋆ means that a problem is hard.
? means that a problem is open.
✣ means that a problem is on a topic beyond the scope of the course.

(c) Recall that the elementary symmetric polynomials are defined by

$$e_d(x_1, \ldots, x_n) = \sum_{\substack{S \subseteq \{1,2,\ldots,n\} \\ |S|=d}} \prod_{i \in S} x_i.$$

Prove that $e_d(x_1, \ldots, x_n)$ is the coefficient of t^{n-d} in the polynomial

$$\prod_{i=1}^{n} (t + x_i).$$

(d) Using the previous parts, prove the following theorem of Ben-Or:

$$\mathsf{S}(e_d(x_1, \ldots, x_n)) \leqslant O(n^2).$$

This is surprising, since e_d has $\binom{n}{d}$ monomials, and cannot obviously be expressed in a simple manner!

(e) Using the same idea, we can see how Strassen's division elimination argument works. Let $p = f/g$, where p, f, g are polynomials. Prove that

$$p(x_1, \ldots, x_n) = \sum_{k=0}^{\infty} f(x_1, \ldots, x_n)(1 - g(x_1, \ldots, x_n))^k.$$

(f) Assume for simplicity that $g(0, \ldots, 0) = 1$. Let $H_i(p)$ denote the ith homogeneous part of p, that is, the sum of all the monomials in p of degree exactly i. Prove that

$$H_i(p) = \sum_{k=0}^{i} H_i(f(1 - g)^k).$$

(g) Using the same interpolation idea as above as in parts (a)–(d), show how to compute $H_i(q)$ given a way of computing q. Using this and the above, figure out how to compute p given ways of computing f and g.

✣ 5. In practice, the algorithm used for computing determinants is the Berkowitz–Samuelson algorithm.

(a) Read the German Wikipedia article for the Berkowitz–Samuelson algorithm: `https://de.wikipedia.org/wiki/Algorithmus_von_Samuelson-Berkowitz` (you can use Google translate if you don't speak German).

⋆ (b) Edit the English Wikipedia article to be better.