Comments on Polynomials and Feasibility

Two meaning of feasibility: reminder. There are two different notions of feasibility.

Practical feasibility. There is an intuitive notion meaning that an algorithm is feasible *in practice*. It means that *for all inputs of reasonable size, performing this algorithm requires reasonable time*.

Here, the word "reasonable" is understood in the meaning of common sense; e.g.:

- performing a billion computational steps is clearly reasonable, since this requires less than a second, while
- performing 10^{300} computational steps is clearly not reasonable, since it would take longer than the lifetime of the Universe.

Formal definition of feasibility. There is also a formal (precise) definition of feasibility: that an algorithm A is feasible if there exists a polynomial $P(n)$ such that for each input x of size $\text{len}(x) = n$, the computation time $t_A(x)$ is smaller than or equal to $P(n)$:

$$t_A(x) \le P(\text{len}(x)).$$

Comment. This is equivalent to saying that the worst-case complexity

$$t^w_A(n) = \max\{t_A(x) : \text{len}(x) = n\}$$

of the algorithm A is bounded by a polynomial:

$$t^w_A(n) \le P(n) \text{ for all } n.$$

What is the relation between the two notions of feasibility. In the ideal world, the formal definition of feasibility should reflect exactly our intuitive understanding, but, as the main lecture shows, these notions are different:

- there exist cases when the algorithm is formally feasible, but not practically feasible, e.g., when $t^w_A(n) = 10^{300}$, and

- there exists cases when the algorithm is practically feasible but not formally feasible, e.g., when $t_A^w(n) = \exp(10^{-100} \cdot n)$.

We will comment on these two examples later in this text, but meanwhile, two things need to be emphasized.

Comment 1. By definition, for an algorithm to be practically feasible, the computation time has to be reasonable for *all* inputs of reasonable size.

If even for *one* input of reasonable size, the computation time is *not* reasonable, this means that this algorithm is *not* practically feasible.

Comment 2. For an algorithm to be formally feasible, it is not necessary that the computation time $t_A^w(n)$ *is* a polynomial: it is sufficient to be *bounded* by a polynomial. For example:

- The time $t_A^w(n) = n \cdot \log(n)$ needed for mergesort is not a polynomial, but since it is bounded by n^2, it is formally feasible.
- Similarly, $\sqrt{n}$ is not a polynomial, but since $\sqrt{n} \le n$, an algorithm that requires time $\sqrt{n}$ is feasible.

What is a polynomial: reminder. Wikipedia defines a polynomial as follows: *a polynomial is an expression consisting of variables (also called indeterminates) and coefficients, that involves only the operations of addition, subtraction, multiplication, and non-negative integer exponentiation of variables.* In precise terms, this means the following:

- every variable is a polynomial;
- every number is a polynomial;
- for every two polynomials P and Q, their sum $P + Q$ is a polynomial;
- for every two polynomials P and Q, their difference $P - Q$ is a polynomial;
- for every two polynomials P and Q, their product $P \cdot Q$ is a polynomial;
- for every polynomial P and for every positive integer constant a, the power P^a is a polynomial;
- and only expressions obtained this way are polynomials.

What is a polynomial: example. For example, the expression $(1+n)\cdot(2+n)^3$ is a polynomial. Indeed:

- 1 and 2 are constant, thus polynomials;
- n is a variable thus a polynomial;
- since 1 and n are polynomials, their sum $1 + n$ is a polynomial;
- since 2 and n are polynomials, their sum $2 + n$ is a polynomial;

- since $2+n$ is a polynomial, and 3 is a positive integer, the power $(2+n)^3$ is a polynomial;
- since $1+n$ and $(2+n)^2$ are polynomials, their product $(1+n)\cdot(2+n)^3$ is a polynomial.

Comment. On the other hand, expressions $\sqrt{n}$, 2^n, and $n\cdot\log(n)$ are *not* polynomials.

What does $\exp(x)$ mean: reminder. It means e^x, where $e = 2.71828\ldots$

Why any function $\exp(a\cdot n)$, for $a>0$, is not bounded by a polynomial. This is something that you studied in CS2 and in Discrete Math: that an exponential function grows faster than any polynomial.

In particular, for $a = 10^{-100}$, this means that $\exp(10^{-100}\cdot n)$ is not bounded by any polynomial.

Comment. In this class, it is important to know this fact, but it is not required to understand how it was proven. For those who are interested, the proof follows from L'Hospital rule: that if $f(n)\to\infty$ and $g(n)\to\infty$, then the limit of the ratio of $f(n)$ and $g(n)$ is equal to the limit of the ratio of their derivatives $f'(n)$ and $g'(n)$:

$$\lim_{n\to\infty}\frac{f(n)}{g(n)} = \lim_{n\to\infty}\frac{f'(n)}{g'(n)}.$$

When $f(n)=\exp(a\cdot n)$, its derivative – computed by using the chain rule – is equal to $a\cdot\exp(a\cdot n)$. When $g(n)=n^k$ for some integer k, then $g'(n)=k\cdot n^{k-1}$. Thus,

$$\lim_{n\to\infty}\frac{\exp(a\cdot n)}{n^k} = \lim_{n\to\infty}\frac{a\cdot\exp(a\cdot n)}{k\cdot n^{k-1}}.$$

If $k=1$, then the numerator tends to infinity but the denominator is a constant, so the whole ratio tends to infinity. If $k>1$, then we can differentiate again and again until we get n^0 – i.e., constant – in the denominator and thus, the limit is equal to infinity.

If $\exp(a\cdot n)$ was limited by a polynomial, the limit of the ratio would be a finite constant, not infinity.

Why $t_A^w(n)=\exp(10^{-100}\cdot n)$ is practically feasible. Suppose that, as an input, we use all the knowledge that we have in the world, this is approximately $n=10^{20}$ bits. Even for this huge number, due to the formula $a^b\cdot a^c = a^{b+c}$, we have

$$10^{-100}\cdot 10^{20} = 10^{(-100)+20} = 10^{-80} = \frac{1}{10^{80}} \le 1.$$

The function $\exp(x)=e^x$ is increasing, so in this case,

$$\exp(10^{-100}\cdot n) \le \exp(1) = 2.71828\ldots$$

In other words, we need less than 3 computational steps to perform this algorithm. This is clearly reasonable.