INTRODUCTION TO QUANTUM INFORMATION SCIENCE

Problem Sheet 0
Hilary Term

ARTUR EKERT & ZHENYU CAI

Questions Label: A - Bookwork B - Standard C - Challenging/Optional

This is a set of preliminary exercises designed as a warm up for things to come. It will not be marked but you are strongly encouraged to attempt all the questions and figure out all the answers by yourself. Consult the lecture-notes (https://qubit.guide/), surf the internet or, if you are absolutely stuck and need further help, check the hints and solutions section or approach your tutors. We do hope you will learn something new.

0.1.A **Information is physical....** ...but the classical theory of computation usually does not refer to physics and as the result it is often falsely assumed that its foundations are self-evident and purely abstract. It is not! There is lots of power in mathematical abstraction but there are also limitations. Here is a puzzle that illustrates this fact. Physics plays a role here. Mathematicians find the puzzle difficult while most experimental physicists think it is relatively easy. Here it goes:

An old-fashioned incandescent lamp in the attic is controlled by one of three on–off switches downstairs labelled A, B and C. But which one? Your mission is to do something with the switches, then determine after *one* trip to the attic which switch is connected to the attic lamp.

Solution: At first this seems to be impossible. There are three possible answers as to which of the three switches controls the bulb, but when you get to the attic either the bulb is on or off. This gives you only one bit of information, which is not enough to choose one correct answer from the three possible answers. Let us bring in (classical) physics. First notice that it is an incandescent bulb which emits both light and heat, so you could feel the bulb and see if it's hot. Thus you can flip switches A and B on, wait 10 minutes or so, then flip switch B off. Go quickly up to the attic: if the bulb is on, it's switch A; if off but warm, switch B; off but cold, switch C. You could even do four switches: Flip A and B on, wait 10 minutes, then flip B off and C on and run up the stairs. The four possible states of the bulb (on/warm, off/warm, on/cool, or off/cool) will tell you what you need to know.

0.2.A **Binary strings.** Mathematicians view computation as an operation on abstract symbols. Any finite set of symbols is called an alphabet. A string over an alphabet is a finite sequence of symbols from that alphabet. Here, without any loss of generality, we will use the binary alphabet $\{0,1\}$. The set of all possible binary strings of length n is written as $\{0,1\}^n$. For example, $\{0,1\}^2$ contains the strings $00, 01, 10$ and 11. How many elements does the set $\{0,1\}^n$ contain?

Solution: There are 2^n binary strings of length n.

We say that a binary string is of even or odd parity if the number of 1's it contains is even or odd, respectively. The parity is often represented by the parity bit: 0 for even and 1 for odd. What is the number of n-bit strings with an even number of ones?

Solution: There are many ways to solve this problem. For example, the number of strings with an even number of 1s is given by the sum

$$\binom{n}{0} + \binom{n}{2} + \binom{n}{4} + \dots$$

which can be evaluated, of course, but with a bit of mathematical gymnastics. We prefer another approach. Consider all possible binary strings of length $n-1$. There are 2^{n-1} of them. Now construct all possible binary strings of length n with even number of ones by appending one more bit. If the number of ones in the $n-1$ bit string is even append 0, if not append 1. Thus the number of n-bit strings with an even number of ones is equal to the number of all possible $n-1$ bit strings, which is 2^{n-1}.

Binary strings can represent all kind of things. In particular, they can represent numbers using the base-2 numeral system. For example, the decimal numerals $0, 1, 2, 3, 4, 5 \dots$ can be represented in binary as $0, 1, 10, 11, 100, 101 \dots$. How many bits are needed to represent the decimal numeral 42?

Solution: $\log_2 42 \approx 5.4$ hence we need at least $\lceil \log_2 42 \rceil$, that is, 6 bits. Indeed 42 is represented by the string 101010.

We quantify how similar are any two binary strings by counting the number of binary places in which they differ. This distance is known as the Hamming distance. For example, $x = 0110$ and $y = 1100$ differ in the first and the third place (counting from the left to the right), which means that they are separated by the Hamming distance 2. Look up the defining properties of a metric and show that that the Hamming distance is a proper metric on $\{0,1\}^n$.

The Hamming weight of a binary string is the number of non-zero entries i.e. the number of entries that are 1. For example, 1000 has weight 1 and 1011 has weight 3. How many binary strings of length n have weight k ?

Solution: In the set $\{0,1\}^n$ there are $\binom{n}{k}$ strings with weight k.

0.3.A **Binary addition and multiplication.** We define two elementary operations on binary digits: addition $\oplus$

$$0 \oplus 0 = 0, \quad 0 \oplus 1 = 1, \quad 0 \oplus 0 = 1, \quad 1 \oplus 1 = 0,$$

and multiplication $\times$,

$$0 \times 0 = 0, \quad 0 \times 1 = 0, \quad 1 \times 0 = 0, \quad 1 \times 1 = 1.$$

The addition is also known as the logical XOR (exclusive OR) and the multiplication as the logical AND ($\wedge$). We will often drop the multiplication symbol and write $x \times y$ as xy. Mathematicians will notice right away that the set $\{0,1\}$ together with the addition $\oplus$ forms a group Z_2 and the same set with both addition and multiplication is the simplest example of what is called a finite field. Various notation are used for this field, e.g. $\mathbb{F}_2$, $GF(2)$, $Z/2$ and Z_2. We can now do linear algebra over $\mathbb{F}_2$. Indeed, strings on length n can be also viewed as vectors in the n-dimensional vector space over the field $\mathbb{F}_2$, denoted as $\mathbb{F}_2^n$. Given two such vectors

$$x = (x_1 x_2 \dots x_n)$$
$$y = (y_1 y_2 \dots y_n)$$

we can define a bitwise inner product of x and y as

$$x \cdot y = (x_1 y_1) \oplus (x_2\ y_2) \oplus \dots (x_n y_n).$$

For example, if $x = 0110$ and $y = 1100$ then $x \cdot y = 1$. Two strings x and y are orthogonal when $x \cdot y = 0$.

Here you should be careful with your intuition - a string with an even number of ones can be orthogonal to itself!

Pick up an arbitrary non-zero $a \in \{0,1\}^n$. How may binary vectors in $\{0,1\}^n$ are orthogonal to a?

Solution: For any non-zero a there are 2^{n-1} binary vectors x such that $a \cdot x = 0$. If the Hamming weight of a is k then $a \cdot x$ is effectively the parity of k bits from x, positions of which are determined by the placing of 1's in a. The value of other bits in x is irrelevant. For example, if $a = 1011$ then $a \cdot x = x_1 \oplus x_3 \oplus x_4$, which is the parity of the first, the third and the fourth bit in x because a has 1 in the first, the third and the forth binary place. The value of the second bit in x is irrelevant. As we run through all possible binary strings x we will find half of them, that is 2^{n-1}, having an even number of 1's in these crucial k positions.

Multiply the following matrices, taking scalars in $\mathbb{F}_2$

$$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Compare your answer to what you get when doing the calculation with rational scalars.

Solution:

Linear algebra over $\mathbb{F}_2$

$$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Linear algebra over $\mathbb{R}$

$$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

0.4.A **One-time pad.**

The addition $\oplus$ of bits can easily be extended to the addition of binary strings; we simply add them bit by bit. For example, two binary strings $x = 0110$ and $y = 1100$ can be added bit by bit as $x \oplus y = 1010$. The set $\{0,1\}^n$ together with the addition $\oplus$ forms a group Z_2^n. This kind of addition is used for encrypting and decrypting messages in a scheme known as the one-time pad. For example, given a cryptographic key $K = 1100101$ the sender can encrypt his binary message, say $M = 1011100$, with key K as $C = M \oplus K$,

$$\begin{array}{rcccccccc} M & = & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ K & = & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ \hline C & = & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}$$

The resulting cryptogram $C = 0111001$, can be then publicly transmitted to the receiver, who can recover the message by adding the cryptogram and the key.

$$C \oplus K = (M \oplus K) \oplus K = M \oplus (K \oplus K) = M$$

This works because for any binary string K, $K \oplus K = 0$ and the binary addition is associative, that is, for any binary strings A, B, C of the same length we have $A \oplus (B \oplus C) = (A \oplus B) \oplus C$.

In one-time-pads the key is a random sequence of 0's and 1's, and therefore the cryptogram—the plaintext plus the key—is also random and completely scrambled unless one knows the key. Both the sender and the receiver must have exact copies of the key beforehand; the sender needs the key to encrypt the plaintext, the receiver needs the key to recover the plaintext from the cryptogram. An eavesdropper, who has intercepted the cryptogram and knows the general method of encryption but not the key, will not be able to infer anything useful about the original message. The secrecy of communication depends entirely on the secrecy of the key. If the key is secret, the same length as the message, truly random, and never reused, then one-time-pad is unbreakable. However, failure to comply with any of these requirements may result is serious security breaches. One-time pads are not the most practical ciphers on the planet but, if used properly, they are unbreakable! Does it solve the problem of perfect security? Well, not quite. There is a snag. All one-time pads suffer from a serious practical drawback. Can you guess what it is?

The US National Security Agency, in a project VENONA, managed to decrypt some of KGB communications because of Soviet's reuse of keys in one-time pad encryptions.

Solution: **THE KEY DISTRIBUTION PROBLEM**

0.5.A **Boolean functions.** From a mathematical perspective a computer is an abstract machine that evaluates a function

$$f : \{0,1\}^n \mapsto \{0,1\}^m, \tag{1}$$

i.e. given n bits of input it produces m bits of output. Such a function is equivalent to m functions, each with one-bit, known as Boolean functions,

$$f : \{0,1\}^n \mapsto \{0,1\}. \tag{2}$$

Thus, one might as well say that computers evaluate Boolean functions. How many Boolean functions $\{0,1\}^n \to \{0,1\}$ can we construct?

Solution: There are 2^{2^n} Boolean functions acting on n bits.

0.6.A **Complex numbers.** Complex numbers have many applications in physics, however, not until the advent of quantum theory was their ubiquitous and fundamental role in the description of the actual physical world so evident. Even today, their profound link with probabilities appears to be a rather mysterious connection. Mathematically speaking, the set of complex numbers is a field. This is an important algebraic structure used in almost all branches of mathematics. You do not have to know much about algebraic fields to follow these lectures, but still, you should know the basics. Look them up.

(1) The set of rational numbers and the set of real numbers are both fields, but the set of integers is not. Why?

(2) What does it mean to say that the field of complex numbers is *algebraically closed*?

(3) Evaluate each of the following quantities:

$$1 + e^{-i\pi},\ |1+i|,\ (1+i)^4 2,\ \sqrt{i},\ 2^i,\ i^i$$

(4) Here is a simple proof that $+1 = -1$:

$$1 = \sqrt{1} = \sqrt{(-1)(-1)} = \sqrt{-1}\sqrt{-1} = i^2 = -1$$

What is wrong with it?

Solution:

(1) A field $\mathbb{F}$ is a set equipped with two types of operations: addition and multiplication. It is closed under these two operation. And for any element of the field, its inverse under addition and inverse under multiplication must also be elements of the field: $f \in \mathbb{F} \Rightarrow -f, f^{-1} \in \mathbb{F}$.

The set of integer $\mathbb{Z}$ is not a field because it lacks multiplicative inverses, that is, $n \in \mathbb{Z} \not\Rightarrow n^{-1} \in \mathbb{Z}$.

(2) Simply see "Algebraically closed field" wikipedia.

(3) 0, $\sqrt{2}$, -8, $e^{i\pi/4}$, $e^{i\ln 2}$, $e^{-2n\pi-\pi/2} \quad \forall n \in \mathbb{Z}$.

(4) The imaginary number i is defined by the property $i^2 = -1$, which means that i and $-i$ are both square roots of -1. Thus we cannot write that $\sqrt{-1}\sqrt{-1} = i^2$.

0.7.A **Quantum computation is a huge quantum interference experiment.** After we run some calculations on a quantum computer, it can generate a range of measurement outputs, with M being one of them. To obtain M as the output, the quantum computer starts calculations in some initial state, then follows n different computational paths to reach the output M. These n computational paths are followed with probability amplitudes $\frac{1}{\sqrt{n}}e^{ik\varphi}$, where φ is a fixed angle $0 < \varphi < 2\pi$ and $k = 0, 1, \cdots n-1$. Show that the probability of generating the output is

$$\frac{1}{n}\left|\frac{1-e^{in\varphi}}{1-e^{i\varphi}}\right|^2 = \frac{1}{n}\frac{\sin^2\left(n\frac{\varphi}{2}\right)}{\sin^2\left(\frac{\varphi}{2}\right)}$$

for $0 < \varphi < 2\pi$ and 1 for $\varphi = 0$. Plot the probability as a function of φ.

$1 + z + z^2 + \cdots + z^n = \frac{1-z^{n+1}}{1-z}$

Solution: The total probability amplitude is just the sum of the probability amplitude of all possible paths. The probability is simply the mod square of this total probability amplitude:

$$\left|\sum_{k=0}^{n-1}\frac{1}{\sqrt{n}}e^{ik\varphi}\right|^2 = \frac{1}{n}\left|\frac{1-e^{in\varphi}}{1-e^{i\varphi}}\right|^2 = \frac{1}{n}\frac{\sin^2\left(n\frac{\varphi}{2}\right)}{\sin^2\left(\frac{\varphi}{2}\right)}$$

0.8.A **Square root of NOT.** Now that we have poked our heads into the quantum world, let us see how quantum interference challenges conventional logic and leads to qualitatively different computations. Consider the following task (which we will return to a few more times in later chapters): design a logic gate that operates on a single bit such that, when it is followed by another, identical, logic gate, the output is *always* the negation of the input. Let us call this logic gate the square root of NOT, or $\sqrt{\texttt{NOT}}$. A simple check, such as an attempt to construct a truth table, should persuade you that there is no such operation in logic. It may seem reasonable to argue that since there is no such operation in logic, $\sqrt{\texttt{NOT}}$ is impossible. Think again!

Fig. 1 shows a simple computation, two identical computational steps performed on two states labelled as 0 and 1, i.e. on one bit. An interplay of constructive and destructive interference makes some transitions impossible and the result is the logical NOT. Thus, quantum theory declares, the square root of NOT is possible. And it does exist! Experimental physicists routinely construct this and many other "impossible" gates in their laboratories. They are the building blocks of a quantum computer. Quantum theory explains the behaviour of $\sqrt{\texttt{NOT}}$, hence, reassured by the physical experiments that corroborate this theory, logicians are now entitled to propose a new logical operation $\sqrt{\texttt{NOT}}$. Why? Because a faithful physical model for

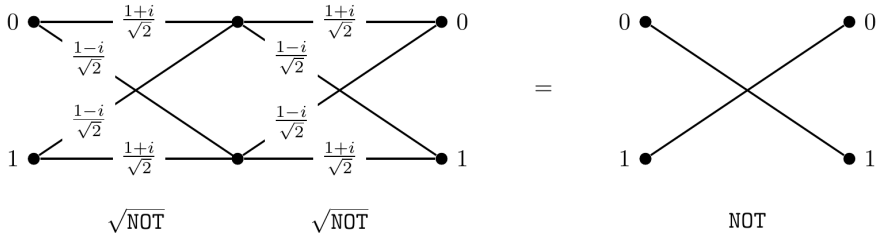


FIGURE 1. A computation that, when repeated, gives exactly NOT. An unlabelled line means that it has probability 1, and the lack of a line corresponds to having probability 0.

it exists in nature. Write a 2×2 matrix which describes the $\sqrt{\texttt{NOT}}$ operation. Is there just one such a matrix? Suppose you are given a supply of Hadamard and phase gates with tuneable phase settings. How would you construct the $\sqrt{\texttt{NOT}}$ gate?

Solution: See Sec. 2.5 of the online book.

0.9.A **Dirac notation.** In this course we will use vectors called kets $|v\rangle$, linear functionals called bras $\langle u|$, inner products $\langle u|v\rangle$ which are complex numbers, outer products $|u\rangle\langle v|$ which are operators, and tensor products $|a\rangle \otimes |b\rangle$ which describe composite systems. Kets can be identified with column vectors, e.g.

$$|a\rangle = \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix}, \; |b\rangle = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}, \; |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \; |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

The adjoint $|v\rangle^\dagger$, denoted by a dagger, is called a bra vector, or just a bra, and is written using a mirror image bracket $\langle v|$. Recall that for any complex matrix A (row and column vectors can be viewed as matrices) the adjoint $A^\dagger$ (also known as conjugate transpose or Hermitian conjugate) is formed by interchanging rows and columns and taking complex conjugate of each entry. Thus, in components bras are always written as row vectors.

(1) Express $\langle a|$, $\langle b|$, $\langle 0|$ and $\langle 1|$ in components as row vectors.

(2) Express the inner product $\langle a|b\rangle$ in terms of components of $|a\rangle$ and $|b\rangle$.

(3) Show that $\langle a|b\rangle^\star = \langle b|a\rangle$. What is the interpretation of $\sqrt{\langle a|a\rangle}$?

(4) Explain why the outer product $|a\rangle\langle b|$ is an operator and write a matrix associated with this operator.

(5) We can think about $\langle a|$ as the linear functional whose value on $|b\rangle$ is the inner product $\langle a|b\rangle$. Explain why $|a\rangle\langle b|$ acting on some ket $|c\rangle$ gives a vector that is proportional to $|a\rangle$. What is the proportionality constant?

(6) Explain why the composition of two operators, $|a\rangle\langle b|$ followed by $|c\rangle\langle d|$ can be written as $\langle b|c\rangle\, |a\rangle\langle d|$

(7) Express $|0\rangle\langle 0|$, $|1\rangle\langle 1|$ and $|0\rangle\langle 0| + |1\rangle\langle 1|$ in the matrix form.

(8) When $|a\rangle\langle a|$ is a projection operator?

We call $|a\rangle$ and $|b\rangle$ orthogonal if $\langle a|b\rangle = 0$. Any maximal set of pairwise orthogonal vectors of unit length forms an orthonormal basis and any vector can be expressed as a linear combination of the basis vectors. For example, vectors $|0\rangle$ and $|1\rangle$ form an orthonormal basis, $\langle i|j\rangle = \delta_{ij}$ (the Kronecker delta), and in this basis

$$|a\rangle = \alpha_0\,|0\rangle + \alpha_1\,|1\rangle\,, \quad |b\rangle = \beta_0\,|0\rangle + \beta_1\,|1\rangle\,.$$

In more abstract terms, bras form the dual vector space. Bras $\langle 0|$ and $\langle 1|$ form the dual basis,

$$\langle a| = \alpha_0^\star \langle 0| + \alpha_1^\star \langle 1|\,, \quad \langle b| = \beta_0^\star \langle 0| + \beta_1^\star \langle 1|\,.$$

We call $\{|0\rangle\,, |1\rangle\}$ the standard or the computational basis. The two kets represent two logical values, 0 and 1, of a quantum bit, called a qubit. One of the most useful formula in quantum mechanics – no, this is not an exaggeration – is the decomposition of the identity. For any orthonormal basis $|e_k\rangle$ the sum of projectors $\sum_k |e_k\rangle\langle e_k|$ is the identity operator

$$\sum_k |e_k\rangle\langle e_k| = \mathbb{1}.$$

This is because you can insert the identity operator just about anywhere in a formula and the view it as a sum of projectors on any basis of your choice. For example,

$$\langle a|b\rangle = \langle a|\,\mathbb{1}\,|b\rangle = \langle a| \sum_k |e_k\rangle\langle e_k|b\rangle = \sum_k \langle a|e_k\rangle\,\langle e_k|b\rangle\,,$$

which gives us the inner product in terms of components.

Given an operator A and an orthonormal basis $|e_k\rangle$ what is the interpretation of $\langle e_i|\,A\,|e_j\rangle$? The trace of a square matrix A, denoted $\operatorname{tr} A$ is defined to be the sum of elements on the main diagonal of A. Show that

$$\operatorname{tr} |a\rangle\langle b| = \langle b|a\rangle\,,$$

0.10.A **Big-O notation.** In order to make qualitative distinctions between how different functions grow we will often use the asymptotic big-O notation. For example, suppose an algorithm running on input of size n takes $an^2 + bn + c$ elementary steps, for some positive constants a, b and c. These constants depend mainly on the details of the implementation and the choice of elementary steps. What we really care about is that, for large n, the whole expression is dominated by its quadratic term. We then say that the running time of this algorithm grows as n^2, and we write it as $O(n^2)$, ignoring the less significant terms and the constant coefficients. More precisely, let $f(n)$ and $g(n)$ be functions from positive integers to positive reals. You may think of $f(n)$ and $g(n)$ as the running times of two algorithms on inputs of size n. We say $f = O(g)$, which means that f grows no faster than g, if there is a constant $c > 0$ such that $f(n) \le cg(n)$ for all sufficiently large values of n. Essentially, $f = O(g)$ is a very loose analogue of $f \le g$. In addition to the big-O notation, computer scientists often use Ω for lower bounds: $f = \Omega(g)$ means $g = O(f)$. Again, this is a very loose analogue of $f \ge g$.

$f = O(g)$ is pronounced as "f is big-oh of g".

(1) When we say that $f(n) = O(\log n)$, why don't we have to specify the base of the logarithm?

(2) Let $f(n) = 5n^3 + 1000n + 50$. Is $f(n) = O(n^3)$, or $O(n^4)$, or both?

(3) Which of the following statements are true?

   (a) $n^k = O(2^n)$ for any constant k

   (b) $n! = O(n^n)$

   (c) if $f_1 = O(g)$ and $f_2 = O(g)$, then $f_1 + f_2 = O(g)$.

Solution:

(1) $O(\log_a n) = O(\frac{\log n}{\log a}) \equiv O(\log n)$.

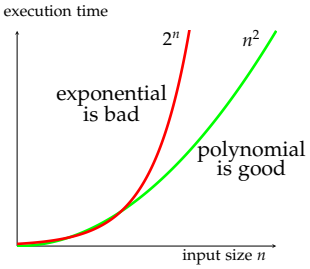
(2) Both.

(3) All true.

0.11.A **Number of paths.** A quantum machine has N perfectly distinguishable configurations. What is the maximum number of computational paths connecting a specific input with a specific output after k steps of the machine? Suppose you are using your laptop to add together amplitudes pertaining to each of the paths. As k and N increase you may need more time and more memory to complete the task. How does the execution time and the memory requirements grow with k and N? Will you need more time or more memory or both?

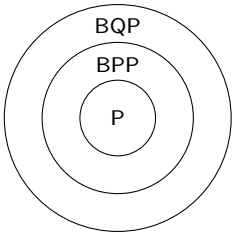
Solution: The are N^{k-1} paths connecting a specific input with a specific output. The probability amplitude to follow a given path is a product of k complex factors, one factor for each computational step. Each step is described by at most N^2 complex numbers, hence we have to store kN^2 complex numbers. Our classical simulator has to add up the N^{k-1} probability amplitudes, each term in the sum is a product of k complex factors. But we do not need to store N^{k-1} probability amplitudes in order to sum them all because only the accumulated subtotal need be stored for the next addition. The execution time, on the other hand, scales with the number of paths N^{k-1}. Thus we will need more time than memory. In fact, one can show that a classical computer with memory space scaling like nk can simulate a quantum circuit with k gates acting on n qubits. In the field of computational complexity this implies $BQP \subseteq PSPACE$. Look it up on the internet.

0.12.A **It is all about computational complexity.** In order to solve a particular problem, computers (classical or quantum) follow a precise set of instructions — an *algorithm*. Computer scientists quantify the efficiency of an algorithm according to how rapidly its running time, or the use of memory, increases when it is given ever larger inputs to work on. An algorithm is said to be *efficient* if the number of elementary operations taken to execute it increases no faster than a polynomial function of the size of the input. We take the input size to be the total number of binary digits (bits) needed to specify the input. For example, using the algorithm taught in elementary school, one can multiply two n digit numbers in a time that grows like the number of digits squared, n^2. In contrast, the fastest-known method for the reverse operation — factoring an n-digit integer into prime numbers — takes a time that grows exponentially, roughly as 2^n. That is considered inefficient.



Explain why the technological progress alone, such as increasing the speed of classical computers, will never turn an inefficient algorithm (exponential scaling) into an efficient one (polynomial scaling).

0.13.A **Complexity classes.** The class of problems that can be solved by a deterministic computer in polynomial time is represented by the capital letter P, for *polynomial* time. The class of problems that can be solved in polynomial time by a probabilistic computer is called BPP, for *bounded-error probabilistic polynomial* time. Finally, the complexity class BQP, for *bounded-error quantum polynomial*, is the class of problems that can be solved in polynomial time by a quantum computer. Provide few examples of problems in P. Any problems which are in BPP but not known to be in P? How about problems which are in BQP but not known to be in BPP?



Solution: See wiki pages of BPP and BQP.

0.14.A **Primality testing.** Given a positive integer N, how can we efficiently determine whether N is prime or not? There exists a randomised algorithm – the Miller-Rabin test – which can determine whether a given number N is prime, but with 1-sided error. If N is prime the algorithm will always correctly output `yes`, and if N is composite the algorithm will correctly output `no` with probability at least $3/4$. Thus the probability the test returns `yes` when N is not prime is ϵ,

Primality used to be given as the classic example of a problem in BPP (in co-RP to be more precise) but not in P. However, in 2002 a deterministic polynomial time test for primality was proposed by Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Thus, since 2002, primality has been in P.

which is not greater than $1/4$. In practical implementations, one performs several iterations of the Miller-Rabin test, and if they all return `yes`, conclude that N is "probably prime". Suppose you run the algorithm (for the same N) r times and each time the algorithm returns `yes`. What is the probability that N is not prime? Solve this problem assuming that it is equally likely that the number N you were given is prime or composite. If you feel like going for a more realistic scenario solve the problem assuming that the number N was chosen randomly from the interval $[1, e^{1000}]$. In the latter case you will have to use the prime number theorem (look it up).

Solution: We know that $\mathbb{P}[\text{yes}|\text{not prime}] = \epsilon \leq 1/4$ and $\mathbb{P}[\text{yes}|\text{prime}] = 1$. In a single run, if N is equally likely to be prime or composite, the probability that the number is not prime given the output `yes` is

$$\mathbb{P}[\text{not prime}|\text{yes}] = \frac{\epsilon}{(1+\epsilon)},$$

and it is smaller than $1/5$. After r independent runs of the test, all of which return `yes`, the probability that N is composite will be less than $(1/5)^r$. If the number was chosen from the interval $[1, e^{1000}]$ the probability that it is prime, using the prime number theorem, is $1/1000$. In this case

$$\mathbb{P}[\text{not prime}|\text{yes}] = \frac{\epsilon(1-10^{-3})}{10^{-3} + (1-10^{-3})\epsilon}$$

which is approximately $1/(1+4\times 10^{-3})$. Running the test r times with all `yes` answers will reduce this probability to roughly $1/(1+4r\times 10^{-3})$.

0.15.A **Chernoff bound.** Suppose a randomised algorithm solves a decision problem, returning `yes` or `no` answers. It gets the answer wrong with a probability $\frac{1}{2} - \delta$, where $\delta > 0$ is a constant. If we are willing to accept a probability of error no larger than ϵ, then it suffices to run the computation r times, where $r = O(\log 1/\epsilon)$.

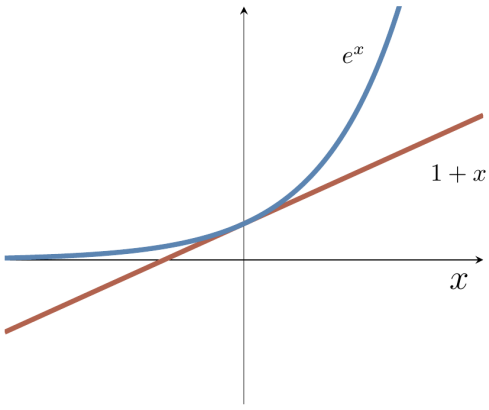


FIGURE 2. Comparison between a linear function and an exponential function.

(1) If we perform this computation r times, how many possible sequences of outcomes are there?

(2) What is the probability of any particular sequence with w wrong answers.

(3) If we look at the set of r outcomes, we will determine the final outcome by performing a majority vote. This can only go wrong if $w > r/2$. Give an

upper bound on the probability of any single sequence that would lead us to the wrong conclusion.

(4) Using the bound $1 - x \leq e^{-x}$, conclude that the probability of our coming to the wrong conclusion is upper bounded by $e^{-2r\delta^2}$.

Solution:

(1) 2^r

(2) The probability of any particular sequence with w wrong answers is

$$\left(\frac{1}{2} - \delta\right)^w \left(\frac{1}{2} + \delta\right)^{r-w}$$

(3) The majority is wrong only if $w \geq r/2$, so the probability of any sequence with an incorrect majority is bounded by

$$\left(\frac{1}{2} - \delta\right)^{\frac{r}{2}} \left(\frac{1}{2} + \delta\right)^{\frac{r}{2}} = \frac{1}{2^r}(1 - 4\delta^2)^{\frac{r}{2}}.$$

(4) There are not more than 2^{r-1} sequences with an incorrect majority thus the total probability of an incorrect majority is

$$\frac{1}{2}(1 - 4\delta^2)^{\frac{r}{2}} \leq e^{-2r\delta^2}$$