Phys 396 2018

Lecture 1

Course Intro:

* Syllabus – assignments – HW
– Term Paper / presentation
– Exams – two in class
– grades pg 3

Course coverage

Quantum information deals with the interplay of quantum physics and information processing / information theory. Roughly there are two paths

1) Quantum systems can be used so that fundamental features of quantum mechanics enable one to do ordinary information processing tasks in ways which conventional information processing devices cannot do. For example, it is possible to search an unsorted database with fewer steps than one can with a conventional computer. This requires

encoding the database information in terms of states of and operations on quantum systems.

Alternatively one can harness quantum systems to distribute cryptographic keys. This will require transmitting quantum systems from one party to another and performing quantum measurements on such systems.

2) Information theory, which describes how to measure information and what that information measure means in terms of transmitting + receiving information, can be used to quantify various aspects of quantum states /systems.

One example involves measurements and estimating parameters from such measurements. There are measures of information that bound the success with which such estimation can be done.

These are the theoretical aspects of quantum information. Experimental and practical aspects include:

1) developing small scale quantum information processing devices
2) developing intermediate scale devices that operate in noisy circumstances

3) developing quantum key distribution schemes.

Course outline

* This course will introduce you to some of the fundamental concepts and features of quantum physics needed to work with quantum information.

    - states + measurements
    - evolution of quantum systems
    - entangled states of multi-particle quantum systems.

* Basic ideas of and procedures in quantum information processing

    - quantum gates, circuits
    - quantum algorithms
    - quantum cryptography.

* Basic ideas from classical information

One eventual aim of the course is to be able to use IBM's cloud quantum device

Show IBM Q Experience website

Classical computing

Although most of classical computing involves coding and machinery, we want to focus on the very most basic building blocks. We also want to consider computer tasks at basic levels from which the specifics of the device are excluded. This entails:

1) ignoring details of device hardware – does your computer have the latest processor?
   – how much RAM?

2) ignoring details of coding ~ is Python better than C++?

Consider an example:

Example: Factorize the following

a) 21

b) 221

c) 1147

Answers: a) $21 = 3 \times 7$

b) divide by primes $2, 3, 5, 7, 11, 13, \ldots$

gives $221 = 13 \times 17$

c) divide by primes: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, ...

$1147 = 31 \times 37$

Note that

1) we can try a brute force method - divide by increasingly large numbers 2, 3, 4, ... - although we can see that it is only necessary to divide by prime numbers. If one can divide by any product of primes then one can also divide by a factor.
So we can make the algorithm/procedure more or less sophisticated

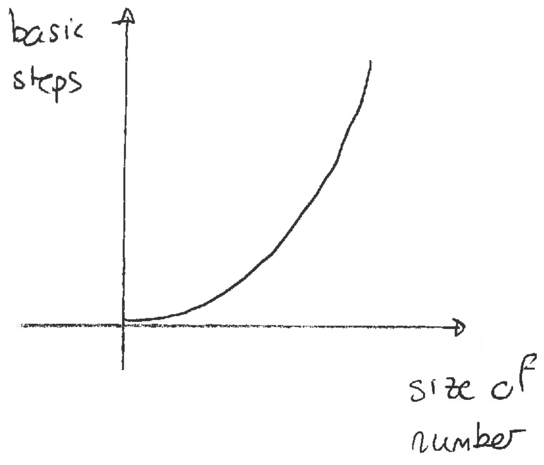
2) as the size of the number increases the amount of effort increases. We can quantify this by counting the number of basic steps needed to complete the task. If we agree that any single arithmetic operation (multiply, divide, add, subtract) takes the same effort then we can count these. So for factoring by dividing by successive primes:

$21 \rightsquigarrow$ 2 steps

$221 \rightsquigarrow$ 6 steps

$1147 \rightsquigarrow$ 11 steps.

We see that the number of steps grows as the size of the number to be factorized grows. In this case the grawth is exponential ~ increasing the number by one digit multiplies the total number of steps



Why does this matter? It matters because actual physical computing requires physical resources:

- hardware
- energy
- time

If increasing the input to the problem by one digit requires roughly doubling the number of resources then at some stage we will never attain sufficient resources to do the computation.

This type of thinking will be key in theoretical computing + quantum info:

- count basic steps
- results independent of generic device
- what kind of growth as input increases

Binary arithmetic

In order to make computing and the associated counting more concrete, we will need a system for representing data and specifically numbers. Consider any positive integer x. This can be represented as a sum of powers of 2 where the coefficients are either zero or one.

$$x = a_0 2^0 + a_1 2^1 + a_2 2^2 + \dots$$

$$\boxed{x = \sum_{k=0}^{n-1} a_k 2^k}$$

n represents the number of terms in the sum.

where $a_k = 0, 1$. In order to represent the number x we just need to list the binary coefficients $a_0, a_1, \dots$. We do this as follows:

$$\boxed{x = a_{n-1}\, a_{n-2} \dots a_2 a_1 a_0}$$

Note that this is just a list. It is not a product.

Example: a) $8 = 2^3 = 1\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 0\cdot 2^0$

so $8 = 1000$

b) $13 = 1\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^0$

so $13 = 1101$

Computers represent numbers in this form:

$$X = a_{n-1}\, a_{n-2} \dots a_0$$

where each <u>bit</u> a_i is either zero or one. The entire collection of bits is called a bit <u>string</u>.

We would like to do binary addition + multiplication with such binary numbers. We can do this by bitwise addition and carry operations.

<u>Example:</u> Consider $4+2=6$

Here $4 = 1\,0\,0$

$2 = 0\,1\,0$

So

$$\begin{array}{r} 4 \\ +\ 2 \\ \hline 6 \end{array} \equiv \begin{array}{r} 1\,0\,0 \\ 0\,1\,0 \\ \hline 1\,1\,0 \end{array} \begin{array}{l} = 1\cdot 2^2 + 0\cdot 2^1 + 0\cdot 2^0 \\ \underline{+\,0\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0} \\ = 1\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 \end{array}$$

<u>Example:</u> Consider $4+4=8$

$$\begin{array}{r} 4 \\ +\ 4 \\ \hline 8 \end{array} \equiv \begin{array}{r} 1\,0\,0 \\ 1\,0\,0 \\ \hline 1\,0\,0\,0 \end{array} \equiv \begin{array}{l} 1\cdot 2^2 + 0\cdot 2^1 + 0\cdot 2^0 \\ +\,1\cdot 2^2 + 0\cdot 2^1 + 0\cdot 2^0 \\ = 2\cdot 2^2 = 2^3 . \end{array}$$

two "1" give "0" carry of "1"

So if two ones occur in a column we add these to give zero and carry "1" to the next column.

<u>Exercise:</u> Represent 13 and 15 in terms of binary numbers. Use this representation to add them and convert the result to a decimal digit

<u>Answer:</u>

		2^4	2^3	2^2	2^1	2^0
13	=	0	1	1	0	1
15	=	0	1	1	1	1
			carry 1 ←		carry	carry 1
		carry 1 ←				
		1	1	1	0	0

This is $2^4 + 2^3 + 2^2 = 16 + 8 + 4 = 28$ ∎

Binary multiplication proceeds by shift and addition. Consider multiplying two three bit numbers

$$a = a_2 2^2 + a_1 2^1 + a_0 2^0$$

$$b = b_2 2^2 + b_1 2^1 + b_0 2^0$$

So

$$ab = (a_2 2^2 + a_1 2^1 + a_0 2^0)(b_2 2^2 + b_1 2^1 + b_0 2^0)$$

$$\begin{aligned} &= b_0 (a_2 2^2 + a_1 2^1 + a_0 2^0) \\ &\quad + b_1 (a_2 2^3 + a_1 2^2 + a_0 2^1) \\ &\quad + b_2 (a_2 2^4 + a_1 2^3 + a_0 2^2) \end{aligned}$$

This is the same as adding:

$$\begin{array}{rl} a_2 a_1 a_0 & \times b_0 \\ a_2 a_1 a_0 0 & \times b_1 \\ a_2 a_1 a_0 0\,0 & \times b_2 \end{array}$$

Example: Consider 5×3

$5 = 101$

$3 = 011$

$$\begin{array}{r} 5 \\ \times 3 \end{array} = \begin{array}{r} 101 \\ \times\ 011 \\ \hline 101 \\ 101 \\ 000 \\ \hline 01111 \end{array} = 15$$

2 Binary multiplication

Represent the two numbers 7 and 6 using binary digits. Use these binary representations to multiply the numbers and convert the result into a decimal number.

Answer:

$7 = 111$

$6 = 110$

So $\begin{array}{r} 7 \\ \times 6 \end{array} =$

$$\begin{array}{rrrrrr} &&&1&1&1\\ &&&1&1&0\\ \hline &&&0&0&0\\ &&1&1&1&\\ &1&1&1&&\\ \hline 1&0&1&0&1&0\end{array}$$

This is $2^5+2^3+2^1 = 32+8+2 = 42.$

Addition modulo two

Consider two binary digits, a_k and b_k. Suppose that we add these:

$$a_k 2^k + b_k 2^k = (a_k + b_k) 2^k$$

If $a_k = b_k = 1$ then this gives

$$1 \cdot 2^{k+1} + 0\, 2^k$$

So effectively these combine to give 0 as a coefficient of 2^k. If $a_k = b_k = 0$ then adding gives:

$$0 \cdot 2^{k+1} + 0 \cdot 2^k$$

again the coefficient of 2^k is zero. If only one of these is 1 then we get

$$0 \cdot 2^{k+1} + 1 \cdot 2^k$$

We want an addition that gives the coefficient of 2^k. Thus we define <u>addition of binary digits modulo two</u> or modular addition via:

a_k	b_k	$a_k \oplus b_k$
0	0	0
0	1	1
1	0	1
1	1	0

The symbol $a_k \oplus b_k$ means modular addition. An alternative notion is

- add the digits, divide by 2, keep remainder.

Note that various addition rules are valid:

a) $(a_k \oplus b_k) \oplus c_k = a_k \oplus (b_k \oplus c_k)$

b) $a_k \oplus b_k = b_k \oplus a_k$

c) $a_k \oplus 0 = a_k$

So we see that

$$\boxed{a_k 2^k + b_k 2^k = (a_k b_k) 2^{k+1} + (a_k \oplus b_k) 2^k}$$

<u>Negation</u>

We can define the negation of a single bit

$$a \to \bar{a}$$

via

a	$\bar{a}$
0	1
1	0

One can prove

$$\boxed{\bar{a} = 1 \oplus a}$$

Functions

We can construct functions of binary numbers. Consider a function that maps integers to integers. So for any integer a $f(a)$ is another integer. So using binary representation

$$f(a_{n-1}a_{n-2}\dots a_1a_0) \quad \text{is an integer}$$

As an example consider functions that map n bit numbers to a single bit number

$$f(a_{n-1}\dots a_0) = 0 \text{ or } 1$$

We often need to construct these.

Example: Let $f(a) = \begin{cases} 0 & a \text{ even} \\ 1 & a \text{ odd} \end{cases}$

Then we see that

$$f(a_{n-1}a_{n-2}\dots a_1a_0) = a_0$$

performs this task.