Quantum computation: lecture 6

Groups:

- finite groups, subgroups, equivalence classes
- Lagrange's theorem

Factoring:

- connection with the hidden subgroup problem and classical algorithm

Finite group

= finite set $G = \{g_1, g_2, \dots, g_n\}$ equipped with internal operation $g_1, g_2 \longmapsto g_1 \cdot g_2$ s.t.

1) $(g \cdot g') \cdot g'' = g \cdot (g' \cdot g'')$ $\forall g, g', g'' \in G$ associativity
2) $\exists e \in G$ s.t. $g \cdot e = e \cdot g = g$ $\forall g \in G$ neutral el.
3) $\forall g \in G, \exists g^{-1} \in G$ s.t. $g \cdot g^{-1} = g^{-1} \cdot g = e$ inverse

On top of that, we say that G is <u>abelian</u> if $g \cdot g' = g' \cdot g$ $\forall g, g' \in G$.

Subgroup

= set $\emptyset \neq H \subset G$ s.t. if $h, h' \in H$, then $h \cdot h' \in H$

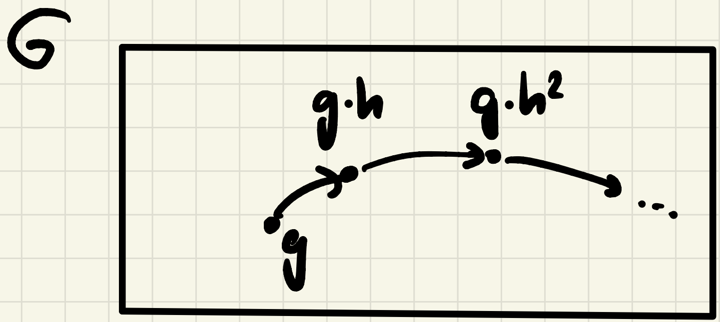
and if $h \in H$, then $h^{-1} \in H$

From this definition, it follows that H is a group, contains the neutral element e, and the associativity law holds inside H.

NB: $H = \{e\}$ & $H = G$ are always subgroups of G

Equivalence classes of a subgroup $H \subset G$

$E_g = \{ g \cdot h \; : h \in H \}$ = set reachable from an element g acting by all possible elements of H:



If G is abelian, then $E_g = \{ h \cdot g \; : h \in H \}$

Fundamental property

If $g \neq g'$ in G, then either $E_g = E_{g'}$ or $E_g \cap E_{g'} = \phi$

This is a direct cansequence of Lagrange's thm:

(i) Let $g, g' \in G$; then either $E_g = E_{g'}$ or $E_g \cap E_{g'} = \phi$

(ii) The number of equivalence classes of H is equal to $\frac{|G|}{|H|}$, i.e. $|H|$ divides $|G|$.

<u>Notation</u>: The set of equivalence classes of H is also denoted as G/H (the <u>quotient</u> group) (so observe that $|G/H| = |G|/|H|$)

<u>Proof of Lagrange's thm</u>:

(i) Let $g, g' \in G$. If $E_g \cap E_{g'} = \phi$, there is nothing to prove; assume therefore $\bar{g} \in E_g \cap E_{g'}$. By def., $\exists\, h, h' \in H$ s.t. $\bar{g} = g \cdot h = g' \cdot h'$

So $g' = g \cdot \underbrace{h \cdot (h')^{-1}}_{\in H} \in E_g$; i.e. $E_{g'} \subset E_g$

Likewise, $g = g' \cdot \underbrace{h' \cdot h^{-1}}_{\in H} \in E_{g'}$; i.e. $E_g \subset E_{g'}$ # (i)

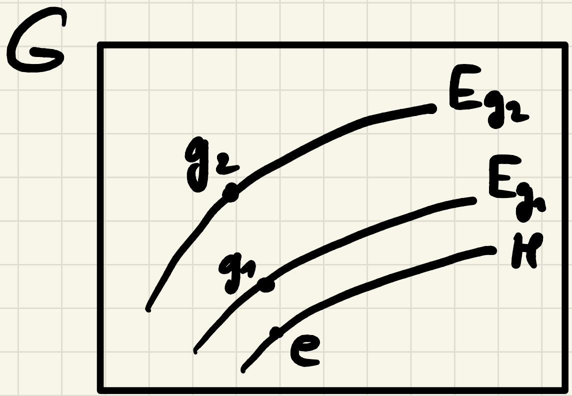
(ii) $|E_g| = |H| \ \forall g$ because the mapping

$$\begin{cases} H \longrightarrow E_g \\ h \longmapsto g \cdot h \end{cases}$$ is bijective

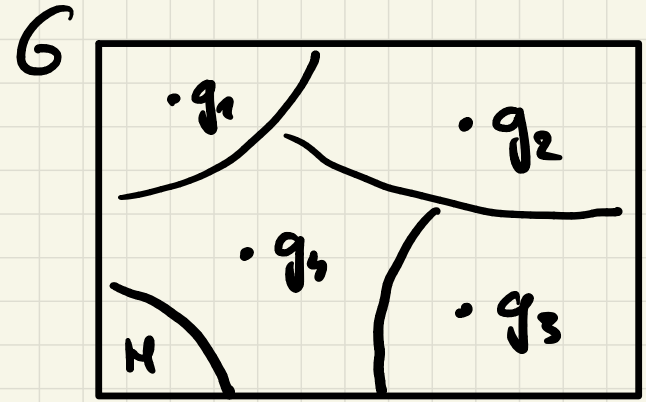
So $|G/H| \cdot |H| = |G|$ # (ii)

(Thanks to part i)

Here are some "pictures":



or



<u>NB</u>: • The equivalence classes of H form a <u>partition</u> of G

• H = subgroup? check first that $|H|$ divides $|G|$!

Examples

a) $G = (\{0,1\}^n, \oplus)$ set of length n binary vectors equipped with addition mod 2

- $H = \{0, a\}$, $0 \neq a \in G$: $|H| = 2$, $|G/H| = 2^{n-1}$

  $E_x = \{ y \in G : y = x \oplus a \}$

  $E_x \cap E_z = \emptyset$ iff $z \ominus x \neq a$

- H = k-dim subspace of G: $|H| = 2^k$, $|G/H| = 2^{n-k}$

b) $G = (\mathbb{Z}, +)$ the set of integer numbers equipped with the usual addition

$H = r \cdot \mathbb{Z}$ with r some positive integer

eq. classes: $E_0 = H$, $E_q = \{ q + n \cdot r : n \in \mathbb{Z} \}$

$0 \leq q \leq r-1$

$G/H = \mathbb{Z}/r\mathbb{Z} = \{0, 1, \dots, r-1\}$, $|G/H| = r$

integers modulo r

c) $G = \mathbb{Z}/M\mathbb{Z} = \{0, 1, \dots, M-1\}$

$H = \{$ multiples of r between 0 & $M-1$ $\}$ (r fixed)

= subgroup of G if and only if r divides M

Note that is this case, G/H is isomorphic to $\mathbb{Z}/r\mathbb{Z}$

Simon's algorithm seen last time can be easily generalized to solve efficiently the <u>hidden subgroup problem</u>:

Let G be a finite group, H be a subgroup of G and $f : G \to G/H$ be s.t. $f(g_1) = f(g_2)$ if and only if $g_1 \cdot g_2^{-1} \in H$. The aim is then to recover H with as few calls as possible to the oracle f.

A new problem

Let $f : \mathbb{Z} \to \mathbb{Z}$ be a function such that $\exists r \in \mathbb{Z}^*$ with $f(x) = f(x+r)$ $\forall x \in \mathbb{Z}$ (and assume that r is the smallest value such that the above relation holds : r = period of f)

This is again a hidden subgroup problem, with the slight difference that $G = \mathbb{Z}$ is infinite

For the above problem to be interesting, we assume furthermore that:

i) r is very large ($n \sim 400$ digits, e.g.)

ii) the equation $f(x) = f(x+r)$ is "impossible" to solve in polynomial time in n

Example

$f(x) = a^x \pmod N$, where both a & N have order n digits.

We show below how the resolution of the above problem relates to the factoring pb.

As a reminder, the <u>factoring problem</u> is to find, given a (large) integer N, a number $2 \leq a \leq N-1$ such that $a | N$ (read this as "a divides N"). By repeatedly solving this pb, one finds the prime factor decomposition of N.

Here is the algorithm for factoring:

1. Choose $2 \leq a \leq N-1$ unif. at random and compute $d = \gcd(a, N)$ (this requires $O((\log N)^3)$ runtime with Euclid's algo)
2. If $d > 1$ (which happens with low prob.), then $a = d$ solves the factoring pb. Assume therefore $d = 1$ in the following.

3. Compute the smallest value of $r \in \mathbb{Z}^*$ such that $a^r \pmod{N} = 1$.

[NB: This is the part where Shor's algorithm is going to help us.]

4. If r is odd, declare failure and restart the algorithm in 1.

5. If r is even, then observe that

$$a^r - 1 = \underbrace{\left(a^{r/2} - 1\right)}_{:= d_-} \cdot \underbrace{\left(a^{r/2} + 1\right)}_{:= d_+}$$

Also by part 3, $a^r - 1 = kN$ for some $k \in \mathbb{Z}$

So

$$N \mid a^r - 1 = d_- \cdot d_+$$

Then three different things can happen:

a) Either $N \mid d_- = a^{r/2} - 1$, but this is actually impossible, as r is by assumption the smallest value s.t. $N \mid a^r - 1$.

b) or $N \mid d_+ = a^{r/2} + 1$; in this case, declare failure and restart the algorithm in 1.

c) or N shares non-trivial prime factors with both d_- and d_+ $\Rightarrow$ success!

Rabin & Miller showed in 1974 that the success probability of this algorithm is greater than or equal to $3/4$.

(So by repeating the algo T times, one can obtain an arbitrarily small error prob.)

The only weakness of the algorithm is the resolution in part 3 ...

Classically, for a & N with order n digits, finding the smallest value of $r \geq 1$ s.t.

$$a^r \pmod{N} = 1$$

requires order $\exp\left(\left(\frac{64n}{17}\right)^{1/3} \cdot (\log n)^{2/3}\right)$ runtime with the best known algorithm

$\Longrightarrow$ runtime superpolynomial in n.

As we shall see in the next lectures, Shor's algorithm finds in $O(n^3)$ runtime the smallest value of $r \geq 1$ s.t.

$$a^x = a^{x+r} \pmod N \quad \forall x \in \mathbb{Z}$$

i.e. $1 = a^r \pmod N$.

This opens therefore the possibility of a polynomial time resolution of the factoring pb.