Introduction

Chapter 1

Quantum Computing

1.1 Learning Problems

1.1.1 A Motivating Example

Suppose you're involved in a simple card game: A dealer places two cards[1] face down on a table. You win the game and a substantial prize if you can guess whether the two face-down cards share the same color. You're allowed to ask the dealer to reveal cards to you, but for each card revealed your potential prize gets smaller.

How many of the cards do you need to see to determine with certainty whether the cards share the same color? Maybe you don't need to know for certain. How does the probability of you being able to guess the correct answer relate to the number of cards seen?

If you have no information at all, you can't do substantially better (or worse) than a fifty-percent chance. In fact, seeing only a single card, does not give you any more knowledge of what the answer to the question is. If you wanted to know with certainty what the answer was, you would need to see both cards.

This is of course a very simple game, but it is an example of a type of problem that we refer to as *learning problems* or often *oracle problems*. These problems consist of a learner who is trying to determine the answer to some question, generally to find the output of a certain function. The learner starts out with incomplete information, but is given access to an *oracle function* she can query to gain more information. An oracle function–also sometimes called a black-box function–is one that you can evaluate the oracle at any input of your choosing, but you have no information about the function other than its responses to your inputs. The learner's goal is to determine the answer to the question in as few questions as possible.

We rephrase the scenario given above in this language. Choose 0 to represent a black card and 1 to represent a red card. Suppose the two facedown cards are labeled a and b.

Example 1.1. Given oracle access to a function $f : \{a, b\} \to \{0, 1\}$. What is the minimum number of queries required to determine $f(a) \oplus f(b)$ where $\oplus$ denotes

[1] Assume that the cards are either red or black, with equal probability of each occurring

addition mod 2?

TODO: rewrite this example

Phrasing problems in terms of oracles is an extremely useful tool. It provides an approach for using information theoretic arguments to provide algorithmic lower bounds. We know of very few tools to show that there is no way to possible solve something quickly.

Example 1.2. **TODO**: Example 2?

1.2 Classical Computing

1.2.1 History, Bits, Information Theory

Before we delve into *quantum computation*, we first briefly give an overview of the relevant concepts from the study of *classical* computation. Although we will not need information theory for any of the arguments we will make, it will be used frequently as a source of intuition.

The fundamental idea of information theory is that any the amount of information that an object contains is related to the number of possible states it could have been in. In this way, the bit, which can be in exactly one of two possible states is the most basic unit of information.

In the 16th and 18th centuries mathematicians and logicians worked to convert formal systems of logic into algebra and arithmetic. One of the most prominent successes of these attempts was by George Boole who managed to faithfully encode propositional logic in the language of arithmetic and algebra. The idea was to let 1 and 0 represent true and false respectively. Then, by interpreting addition and multiplication as mod 2, one obtains exactly the logical operations `AND` and `XOR`. In modern terms, we recognize this as the structure of the field $\mathbb{F}_2$. **TODO**: remove this paragraph

TODO: *These next two paragraphs just repeat the previous two paragraphs, clean this up*

We define a bit as an element of $\mathbb{Z}/2\mathbb{Z} = \{0, 1\}$. It also can be useful to think of bits as representing a true or false value. This relationship gives a direct correspondence between classical computation and propositional logic. We write the state space of a bit as $\mathbb{Z}/2\mathbb{Z}$ to emphasize that it has a natural additive operation given by mod 2 addition. Alluding to propositional logic, this operation is called exclusive-or, usually written as `XOR` or $\oplus$, although we will oftentimes just write $+$.

Of course having just one bit, is not particularly interesting. As a reference, it is common today to measure memory in terms of *gigabytes*. One gigabyte is equivalent to eight billion bits. We most often work with *strings of bits*. A string of 2 bits has four possible states, each described by an element of $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$. Analogously, an n bit string has state space $(\mathbb{Z}/2\mathbb{Z})^n$. The group addition readily extends to n bit strings giving it the operation that is commonly known as bitwise XOR.

Thus any function $f : \{0, 1\}^m \to \{0, 1\}^n$ can be expressed by some sequence of logical operations and each sequence of logical operations corresponds to a function.

1.2.2 Circuits and Logic Gates

Definition 1.1 (Nielsen and Chuang, pg 129)**.** A *circuit* is made up of wires and gates, which carry information around, and perform simple computational tasks, respectively.

Definition 1.2 (Nielsen and Chuang, pg 129)**.** A *logic gate* is a function $f : \{0,1\}^k \to \{0,1\}^\ell$ from some fixed number k of input bits to some fixed number ℓ of output bits.

Example 1.3. Here is an example of a logic gate

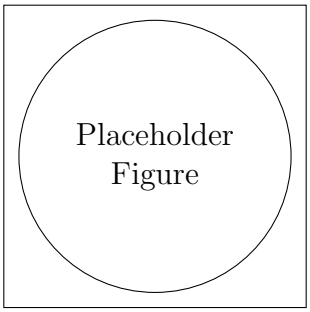


Figure 1.1: The logic gates `AND` and `XOR`

Example 1.4. Here is an example of a circuit

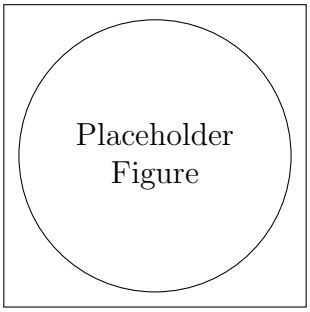


Figure 1.2: An example of a basic circuit, maybe make it match whatever example I use for a circuit family

We state the following important result without proof.

Theorem 1.1. *The set of gates* {`AND`, `NOT`, `OR`} *are* universal. *That is any function* $f : \{0,1\}^n \to \{0,1\}$ *can be represented by circuit a consisting of only* `AND`*,* `NOT`*, and* `OR` *gates.*

1.2.3 Circuits as Algorithms

Definition 1.3 (verbatim from Sipser – cleanup)**.** A *circuit family* is an infinite list of circuits $(C_0, C_1, C_2, ...)$, where C_n has n input bits.

Definition 1.4. The circuit complexity of a function is the size complexity of a minimal circuit family for that language.

Example 1.5. Example circuit family to compute something. Maybe how many input bits are odd.

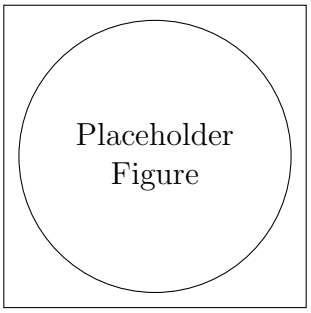


Figure 1.3:

1.2.4 Introduce controlled gate notation

Just example of Toffoli gate or whatever

1.2.5 Circuits with Oracles

We can consider circuits with oracle access. Give example from original motivating problem.

1.3 Quantum Computing

Now we introduce the concepts required for quantum computation. In the quantum world, the *qubit* is the fundamental unit of information. In the same way that we could express any classical algorithm as a circuit of logical operations on bits, we can express any quantum operation as a sequence of *unitary* operations on *qubits*.

1.3.1 Qubits

The following definitions will give a complete description of a model of quantum computing. These can all be derived from the postulates of quantum mechanics.

Whereas the state of a bit has *two* possible values—either 0 or 1—the state of a qubit is a unit vector in complex vector space spanned by *two* basis states. We give the following precise definition

Whereas a bit has two states—0 or 1—a qubit has *two basis states*. The following definition makes this precise

Definition 1.5. The state of a *qubit* is a unit vector in $\mathbb{C}^2$. That is, a pair of complex numbers

$$(\alpha, \beta) \in \mathbb{C}^2$$

satisfying the normalization condition

$$|\alpha|^2 + |\beta|^2 = 1$$

We refer to the space $\mathbb{C}^2$ as the *state space* of a qubit and consider it to be an inner product space equipped with the standard inner product on $\mathbb{C}^n$.[2] We choose the standard basis $\{(0,1),(1,0)\}$ to represent $\mathbb{C}$. In order to make the analogy with bits more clear, we denote

$$|0\rangle := \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle := \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

The symbol $|\ \rangle$ is called a *ket* and is used to denote a vector representing a quantum state. It comes from the *bra-ket* notation which is widely used in the greater theory of quantum mechanics.

1.3.2 Measurement

Although a qubit has a quantum state in $\mathbb{C}^2$, the we cannot directly observe this quantum state. Despite having what seems like a much richer state, when we try to measure qubits using any sort of detector, they snap into the states $|0\rangle$ or $|1\rangle$. This of course defies any reasonable expectations one would have about the world, and has raised much philosophical debate about how it should be interpreted. However, regardless of what interpretation we take, it remains an experimental fact. The following postulate formalizes this concept as a process called *measurement in the computational basis* .

Postulate 1. *Given a qubit ψ, we may perform* measurement *of ψ in the* computational basis. *If ψ was in state $\alpha|0\rangle + \beta|1\rangle$ prior to the measurement, after the measurement it will be in state $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$.*

This is only a specific case of a much more general postulate of measurement in quantum mechanics. However, this conception of measurement will be sufficient enough for many of the results in quantum computing and all of the results that we will need to consider. A more in depth discussion of the postulates of quantum mechanics and how they relate to quantum computing can be found in [2].

If we naively think of the state of a qubit as being in $\mathbb{R}^2$ rather than $\mathbb{C}^2$ by just ignoring the imaginary part, we can generate some limited geometric intuition about

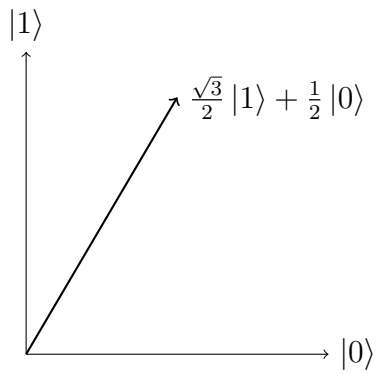


Figure 1.4: A qubit with state $\frac{\sqrt{3}}{2}|1\rangle + \frac{1}{2}|0\rangle$ pictured in the plane by ignoring the imaginary part.

what is going on.

A measurement on the pictured qubit will return 0 with probability 3/4 and 1 with probability 1/4. This perspective hopefully makes it clear why the state of a qubit had to be a unit vector. When written in the standard basis, the coefficients of $|0\rangle$ and $|1\rangle$ correspond to the probability of a measurement producing that outcome. Thus requiring that this state be a unit vector is equivalent to requiring that the total probabilities sum to one.

This also motivates the other postulate of quantum mechanics that we will need to formalize this model of quantum computation.

Postulate 2. *The valid operations on quantum states are those given by* unitary transformations.

The unitary transformations on $\mathbb{C}^n$ are exactly the *isometries* of $\mathbb{C}^n$, that is, the ones that preserve the inner-product. This is, perhaps not too surprising, given that preserving the metric is necessary condition to ensure that the probabilities sum to one.

Definition 1.6. The set of unitary transformations on $\mathbb{C}^n$ can be represented as the set of $n \times n$ matrices U satisfying the identity $U^{-1} = U^\dagger$ where $U^\dagger$ denotes the conjugate-transpose of U.

A consequence of this postulate of quantum mechanics is that quantum information cannot be destroyed without a making a measurement.

[2]In quantum mechanics literature, state spaces of quantum systems are always referred to as *Hilbert spaces* regardless of their dimension. This can be confusing as *Hilbert spaces* in mathematics are generally associated with infinite dimensional spaces. Since all of the spaces in this thesis will be finite dimensional (and isomorphic to $\mathbb{C}^{2^n}$), it is easier to avoid the confusion and use the term state space instead.

1.3.3 Gates

By postulate 2, the valid operations on a qubit are the 2×2 unitary matrices.

This part of quantum computing can be confusing at first, since the notation constantly switches between kets and matrices.

Example 1.6. The quantum `NOT` gate is the single qubit gate defined by

$$|0\rangle \mapsto |1\rangle$$
$$|1\rangle \mapsto |0\rangle$$

It should be noted that it is common to refer to the quantum `NOT` gate by the letter X.

In standard vector notation this is

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

Which gives us that the quantum `NOT` gate is represented by the matrix

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

Example 1.7. A very important single qubit gate is the *Hadamard* gate. The Hadamard gate, denoted H, can be defined in terms of its action on the standard basis as

$$|0\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$|1\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

These states appear frequently and so it is common to give them their own notation. We define

$$|+\rangle := \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

and

$$|-\rangle := \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

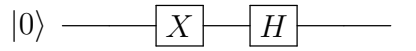
The Hadamard gate has matrix representation

$$H := \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

1.3.4 Quantum Circuits

Much like in classical computation, it is convenient to represent manipulations of qubits as circuits. A *quantum circuit* is made up of qubits, wires, and gates. It is perhaps easiest to define by giving examples.

Consider the following very simple circuit

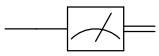


This circuit depicts a single qubit starting in state $|0\rangle$. As it travels through the wire left to right, X takes it to $|1\rangle$ and then H takes $|1\rangle$ to $|-\rangle$. This circuit represents the equation

$$|-\rangle = HX\,|0\rangle$$

Of course by associativity, we could have used a single gate HX.

We allow for one special gate that is not unitary to denote measurement. The measurement gate is depicted



The two parallel lines indicate that the wire is carrying classical information.

1.3.5 Multiple qubits

Up until this point we have only considered single-qubit systems. However, computation is not very interesting if we're only restricted to a single qubit.

Consider a string of two bits. This string has four possible values: 00, 01, 10, 11. Much like in the single bit case, while a string of two bits has four possible states, a string of two qubits has four basis states. Keeping with the analogy from before, we number these basis vectors in binary and call them $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ which represent $(1,0,0,0), (0,1,0,0), (0,0,1,0)$, and $(0,0,0,1)$ respectively.

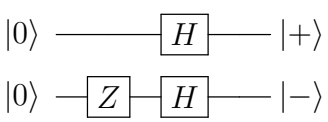
Formally, we think of the state space of a pair of qubits as being $\mathbb{C}^2 \otimes \mathbb{C}^2$, where $\otimes$ denotes the tensor product of two vector spaces. The tensor product is a complicated construction and defining it in full generality here would take us beyond the scope of this thesis. Luckily, all of the spaces we are working with are finite and we have already chosen bases, so we don't need to worry about this. For the purposes of quantum computing the following definition is sufficient.

Definition 1.7. Let V and W be vector spaces with bases $\{v_1, ...v_n\}$ and $\{w_1, ...w_m\}$ respectively. Define B to be the set of all formal pairs $v_i \otimes w_j$.

The tensor product $V \otimes W$ is given by span$B$.

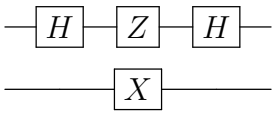
In general the state space of an n-qubit system is $\mathbb{C}^{\otimes 2n}$.

We can have circuits with multiple qubits



Let

Example 1.8. Deutsch-Josza



The Deutsch-Josza Algorithm

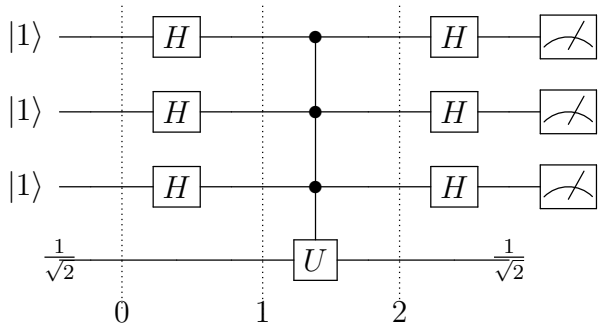
1.3.6 The Bernstein-Vazirani Algorithm

Problem 1. Let $s \in \{0,1\}^m$ be a hidden m-bit binary string and let $b \in \{0,1\}$ be a hidden bit. Given oracle access to a function $f : \{0,1\}^n \to \{0,1\}$ defined by $f(\vec{x}) = \vec{x} \cdot \vec{s} + b \pmod 2$, how many queries are required to determine the values of $\vec{s}$ and b?

The Bernstein Vazirani Algorithm

Define a circuit with state space $\mathbb{C}^{\otimes 2n} \otimes \mathbb{C}^2$. We initialize the system by setting the first n qubits to $|1\rangle$ and the response qubit to $|0\rangle$.

The key to this problem will be to utilize a technique sometimes known as the *phase-kickback trick*.



Chapter 2

Representation Theory of Finite Groups

2.1 Introduction

The purpose of this chapter will be to introduce the terminology and results from representation theory that we will need to investigate symmetric oracle problems in chapters 3 and 4. A reader familiar with the basic results of the representation theory of finite groups can skip ahead to chapter 3.

This treatment of representation theory largely follows [1]. I have omitted many proofs here and instead leave the reader to refer to [1]

Although there are many analogues of the results given here for fields other than $\mathbb{C}$ and for infinite groups, we will not express them in their full generality. The representations that will arise in our study of quantum oracle problems will be of finite groups and over $\mathbb{C}$. Moreover, many proofs in representation theory are substantially simpler and more elegant when the underlying field is restricted to $\mathbb{C}$.

2.2 Representations

A representation of a group G is an encoding of G as a set of linear maps on some vector space V. When a basis of V is specified, this yields an explicit representation of each $g \in G$ as a matrix. Looking at the possible ways of representing G as a matrix group uncovers much information about the structure of G that may have been hidden otherwise. In particular, any encoding of G into a set of matrices will be unique only up to a change of basis. This exposes a deep connection between the symmetry arising in vector spaces by change of basis and symmetry inherent in the structure of groups, arising from conjugation.

Definition 2.1. Let G be a group and V be a vector space over a field F. A *representation* of G is a homomorphism $\rho : G \to \mathrm{Aut}(V)$, where $\mathrm{Aut}(V)$ denotes the group of linear maps from V to itself. Although, the representation is defined by the homomorphism ρ, it is common instead to refer to it by the vector space it is acting on.

Equivalently, a representation can be characterized as an action of a group G on a vector space V. In this case, the action is required to be compatible with the vector space structure on V. That is, in addition to the requirements for an action of a group on a set, we must have

$$g \cdot (\alpha v_1 + v_2) = g \cdot \alpha(v_1) + g \cdot (v_2)$$

for all $g \in G,\ v_1, v_2 \in V$, and $\alpha \in F$. We denote the action of $g \in G$ on a vector v as $g \cdot v$ or gv.

There is yet a third way to express the information of a representation. Although this is the most abstract, it provides the most concrete language for working with representations.

First we must define an object called the *group algebra* that is naturally associated to every group. [1] One can view the group algebra as an encoding of a group into the structure of polynomial arithmetic.

Definition 2.2. Let G be a finite group of order n. The *group algebra* of G, denoted $\mathbb{C}[G]$ or $\mathbb{C}G$, is defined as a polynomial ring over $\mathbb{C}$ with one indeterminant for every $g \in G$. Thus every element of $\mathbb{C}[G]$ is of the form

$$\alpha_1 g_1 + \alpha_2 g_2 + ... + \alpha_n g_n$$

where $\alpha_1, ..., \alpha_n \in \mathbb{C}$ and $\{g_1, ..., g_n\} = G$.

The operations of addition, multiplication, and scalar-multiplication are inherited from the properties of polynomial rings, except we define the multiplication of monomials to follow the group structure. For any $\alpha_1, \alpha_2 \in \mathbb{C}$ and $g_1, g_2 \in G$ we define

$$(\alpha_1 g_1)(\alpha_2 g_2) = (\alpha_1 \alpha_2)(g_1 g_2)$$

where $\alpha_1\alpha_2$ is the product of α_1, α_2 in $\mathbb{C}$ and $g_1 g_2$ is the product of g_1, g_2 in G. In particular, this means $\mathbb{C}[G]$ is a commutative ring if and only if G is abelian.

A representation $\rho : G \to \mathrm{Aut}(V)$ defines a $\mathbb{C}G$-module structure on V by the action

$$(\alpha_1 g_1 + ...\alpha_n g_n)\dot{v} := \alpha_1 \rho(g_1)v + ... + \alpha_n \rho(g_n)v$$

Conversely, a $\mathbb{C}G$-module with action $\varphi : \mathbb{C}G \times V \to V$ determines a representation ρ defined by $\rho(g) = \varphi(1g)$. This establishes a correspondence between representations of G and $\mathbb{C}G$-modules.

2.3 The structure of representations

Definition 2.3. Let ρ be a representation of a group G on a vector space V. A *subrepresentation* is a vector subspace $W \subset V$ that is invariant under the action of G. That is, for all $g \cdot w \in W$ for all $g \in G$, $v \in W$.

A representation is said to be irreducible if it has no nontrivial subrepresentations (the only G-invariant subspaces of V are $\{0\}$ and V itself).

[1] The terms *group algebra* and *group ring* are often used synonymously.

The next two results are fundamental to the study of representations.

Theorem 2.1 (Maschke's Theorem)**.** *Let G be a finite group and V a representation of G over $\mathbb{C}$. If W is a subrepresentation of V, then there is a subrepresentation W' such that $V = W \oplus W'$.*

Proof. We are given that W is a subrepresentation of V. Let $\pi : V \to W$ be a projection of V onto W. Although $\ker \pi \oplus W = V$, we do not know that $\ker \pi$ must be a subrepresentation. In fact, it will not be a subrepresentation in general. We will need to modify it slightly.

We construct a new function $\pi_G : V \to W$ by averaging over G. Define it by

$$\pi_G(v) = \frac{1}{|G|} \sum_{g \in G} g \cdot \pi(v) \cdot g^{-1}$$

□

Theorem 2.2 (Schur's Lemma)**.** *If M and N are two finite-dimensional irreducible representations of a group G, and $\varphi : M \to N$ is a linear transformation that commutes with the action of the group, then φ is either an isomorphism or the 0 map.*

2.4 Character Theory

Definition 2.4. Let $\rho : G \to V$ be a representation. The map $\chi : G \to \mathbb{C}$ given by $\chi(g) = \operatorname{Tr} \rho(g)$ is called the *character* of V.

Definition 2.5. Let G be a group. We say that χ is a character of G if it is the character of some representation of G.

Moreover, if χ is the character of an irreducible representation of G, we say that χ is an *irreducible character of G*.

We give a few basic properties of characters.

Theorem 2.3. *Let χ be a character of a group G. Suppose $g \in G$ and g has order m. Then*

1. $\chi(1) = \dim V$.
2. *$\chi(g)$ is a sum of mth roots of unity.*
3. $\chi(g^{-1}) = \overline{\chi(g)}$.
4. *$\chi(g)$ is a real number if g is conjugate to g^{-1}*

2.4.1 Character Tables

Theorem 2.4. *The number of irreducible characters of G is equal to the number of conjugacy classes of G.*

Theorem 2.5. *The characters of a group G are constant on the conjugacy classes of G. That is, for all $g, h \in G$, $\varphi(g) = \varphi(h)$ if and only if $h = kgk^{-1}$ for some $k \in G$.*

Writing the information of the characters of a group into a table, gives us an extremely compact way write down the important information associated to a group.

Definition 2.6. A character table of a group G is of the form

2.4.2 Properties of Irreducible Characters

Definition 2.7. A *class function* of a group G is a function $\varphi : G \to \mathbb{C}$ that is constant on the conjugacy classes of G. As we have already seen, the characters of a group are always class functions.

The set of class functions on a group G form a vector space under pointwise addition. This space has an inner product given by

$$\langle \varphi, \psi \rangle := \frac{1}{|G|} \sum_{g \in G} \varphi(g)\overline{\psi(g)}$$

Theorem 2.6. *The set of irreducible characters of a group G form an orthonormal basis for the space of class functions on G.*

Theorem 2.7. *Let V be a representation of a group G and let ψ be its associated character. V is irreducible if and only if and only if $\langle \psi, \psi \rangle = 1$.*

Proof. For now, just refer to James and Liebeck. □

2.4.3 More Results on Characters

Theorem 2.8. *Let $\chi_1, ..., \chi_n$ be the irreducible characters of a group G. If ψ is any character of G, then*

$$\psi = d_1\chi_1 + ... + d_k\chi_k.$$

for some non-negative integers $d_1, ..., d_k$. Moreover, these coefficients count the number of times each irreducible character appears:

$$d_i = \langle \psi, \chi_i \rangle$$

and

$$\langle \psi, \psi \rangle = \sum_{i=1}^{k} d_i^2$$

This is redundant with orthonormal basis theorem, but I should introduce the combinatorial interpretation.

Definition 2.8. Suppose that ψ is a character of G, and that χ is an irreducible character of G. We say that χ is a *constituent* character of ψ if $\langle \psi, \chi \rangle \neq 0$. Thus, the constituents of ψ are the irreducible characters χ_i of G for which the integer d_i in the expression $\psi = d_1\chi_1 + ... + d_k\chi_k$ is nonzero.

2.4.4 Products of Characters

Definition 2.9. Tensor product of representations

Definition 2.10. Let G be a group. Define a product on the space of class functions of G by pointwise multiplication. i.e. for all class functions χ, ψ of G,

$$\chi\psi(g) = \chi(g)\psi(g) \text{ for all } g \in G$$

Proposition 2.1. *Let G be a group. Let V and W be representations of with characters χ and ψ respectively. Then the character of the representation $V \otimes W$ is the product character $\chi\psi$*

Theorem 2.9. *(Burnside-Brauer) Let χ be a faithful character of G, and suppose that $\chi(g)$ takes precisely r different values as g varies over all the elements of G. Then every irreducible character of G is a constituent of one of the powers $\chi^0, \chi^1, ..., \chi^{r-1}$*

Proof. TODO: Prove this one. The proof is not hard and it is very relevant. □

2.4.5 If there is time

Frobenius Reciprocity and Induced Subgroups

Chapter 3

Symmetric Oracle Problems

Chapter 4

Constructions of the Quantum Base Size

4.1 Character Table of the Dihedral Group

The *dihedral group* of order $2n$ is given by

$$D_{2n} := \langle a, b \mid a^n = b^2 = 1,\ bab = a^{-1} \rangle$$

It occurs naturally as the planar symmetries of the n-gon. For this reason, elements of the form a^i are often referred to as rotations and elements of the form $a^i b$ are referred to as reflections.

4.1.1 Conjugacy Classes of D_{2n}

Another writeup of this can be found in chapter 12 of [1] The conjugacy classes will be slightly different if n is even or odd. We first give the case for n even.

Suppose $n = 2k$ (we are considering the Dihedral group of order $4k$)

The center of D_8 is given by

$$\mathbf{Z}(D_8) = \{1, a^\}$$

Thus [1] and $[a^k]$ are both of size 1.

Let a^i be a rotation. Then $ba^ib^{-1} = a^{-1}$

4.1.2 Irreducible Characters of D_8

Again, this will be different depending on whether n is even or odd.

First we give the even case, for D_{4k}.

Linear Characters

The linear characters are the homomorphisms $\chi : D_8 \to \mathbb{C}^\times$. Since any group homomorphism is determined by its action on the generators, this is routine to check. Since $b^2 = 1$, we must have $\varphi(b)^2 = 1$. Hence $\varphi(b) = 1$ or $\varphi(b) = -1$

Similarly, $\varphi(a) = \pm 1$ **TODO**: Expand this

The linear characters are precisely the homomorphisms $\lambda : D_{2n} \to \mathbb{C}^\times$. We can compute all such homomorphisms by looking at how they act on the generators.

Let $\lambda : D_{2n} \to \mathbb{C}^\times$ be a homomorphism. Since $b^2 = 1$ we must have $\lambda(b)^2 = 1$, hence the only possible values for $\lambda(b)$ are 1 and -1. Since we have the relation $abab = 1$, then $1 = \lambda(abab) = \lambda(a)\lambda(b)\lambda(a)\lambda(b) = \lambda(a)^2$. Thus $\lambda(a)$ must also be either 1 or -1.

The degree 2 irreducible representations of D_{2n} are ρ_j given by

$$a \mapsto \begin{bmatrix} \zeta_n^j & 0 \\ 0 & \zeta_n^{-j} \end{bmatrix}, \qquad b \mapsto \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

and thus the irreducible characters of degree 2 are given by

$$\chi_j(a^i) = \zeta_n^j + \zeta_n^{-j}$$
$$\chi_j(a^i b) = 0$$

TODO: Finish this

4.2 Quantum Base Size of the Dihedral Group

Here we compute the quantum base size for the standard action of the Dihedral group of order $2n$ on the vertices of the n-gon.

Permutation Representation

Let $G = D_{4k}$

References

[1] G. James and M. Liebeck. *Representations and Characters of Groups*. Cambridge University Press, 2001.

[2] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 10th edition, 2011.