UP135 Notes

Gene Yang

2025 - 2026

Contents

1 Week 1 (Aug 24 - Aug 30)

1.1 Classical Computation Theory

- $f(x)$ takes input and gives output.
- From Physics: limited to 10^{43} operations/sec and 10^{122} total bits.
- From Math: more problems exist than programs to solve them.

1.2 Single-Tape Turing Machine (TM)

Software:

- A: Input/output alphabet (input string x, output string y)
- S: Internal alphabet (e.g. $+, -, \times, \otimes, ...$)
- $q_i \in Q$: Internal state of computer (start at q_0, stop at q_{halt})
- δ: Transition rule; what action to take given x_i, q_i

$$\delta : Q \times S \longrightarrow Q \times S \times \{-1, 0, 1\}$$

Hardware:

- Infinite tape
- Each cell contains 1 symbol $\in S$
- Tape head moves 1 cell left/right per step
- Read symbol s_i in cell, compute action based on q_i. Option is either erase/write to cell and move left/right/stay
- Label position by p_t at some time step t

Starting config:

- First n cells contain input $\alpha \in A$, rest blank
- Head is at left edge $p_0 = 1$ in state q_0
- General config $< s_0, s_1, s_2, ...; p_t, q_t > \quad \rightarrow \quad \delta(q_0, \alpha) \quad \rightarrow \; < \sigma_1; p_1, q_1 >$

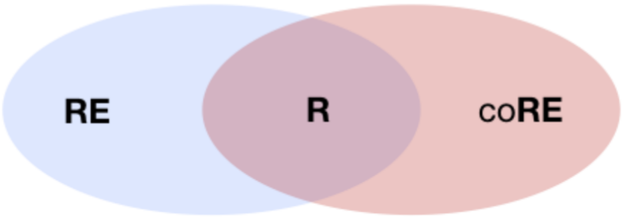
Possibilities:

- Halts: reach q_{halt}. Or q_{accept}/q_{reject}
- Never halts: runs forever

1.3 Church-Turing Thesis (1936)

The class of functions computable by a Turing Machine is equivalent to the class of functions computable by an algorithm.

- A definition of computation.
- A language is *Turing Recognizable* if some TM recongizes it. Also called *Recursively Enumerable (RE)*
- A language is *Turing Decidable* if some TM decides it. Also called *Recursive (R)*
- *RE*: class of decision problems for which 'yes' can be verified by TM in finite time
- *R*: class of decision problems for which 'yes' or 'no' can be verified by TM in finite time.



$R = RE \cap coRE$

1.4 The Halting Problem

Consider program $h([M])$, taking program $[M]$ as input.

- $h([M]) = 1$ if $M(x)$ halts
- $h([M]) = 0$ if $M(x)$ runs forever

Make small modification to create program P:

```
P(x){
    if(h(x) = 0): halt
    else: loop forever
}
```

Feed P into P:

- If P halts, P runs forever.
- If P runs forever, P halts.
- Thus P and subsequently h cannot exist.

1.5 Uncompubtability

Hyperoperations:

$$\begin{aligned}
\phi(a,b,0) &= a + b \\
\phi(a,b,1) &= a \cdot b \\
\phi(a,b,2) &= a \uparrow b = a^b \\
\phi(a,b,3) &= a \uparrow\uparrow b = a^{a^{a^{\cdot^{\cdot^{\cdot}}}}} && \text{(exponentiate } a\text{, } b-1 \text{ times)} \\
\phi(a,b,4) &= a \uparrow\uparrow (a \uparrow\uparrow b) && \text{(tetrating } a\text{, } b-1 \text{ times)} \\
&\mathit{etc.}
\end{aligned}$$

Example:

$$\begin{aligned} 3 \uparrow\uparrow 3 &= 3 \uparrow (3 \uparrow 3) \\ &= 3^{3^3} = 3^{27} \\ 3 \uparrow\uparrow\uparrow 3 &= 3 \uparrow\uparrow (3 \uparrow\uparrow 3) \\ &= 3 \uparrow\uparrow 3^{27} = \text{big} \end{aligned}$$

Graham's number: g_{64}

$$\begin{aligned} g_1 &= 3 \uparrow\uparrow\uparrow 3 \\ g_2 &= 3 \uparrow^{g_1} 3 \end{aligned}$$

Ackermann numbers: $A(n)$

$$A(n) = n \uparrow^{n-2} n \quad \text{OR} \quad A(n) = n \uparrow^{n} n$$

Disproved that only primitive recursive functions define computability — TM can list out $A(n)$.

2 Week 2 (Aug 31 - Sep 6)

2.1 Computational Complexity

Two areas of concern:

- TIME: amt of computational steps to solve problem
- SPACE: amt of memory storage needed to solve problem

Use Big-O notation to simplify to leading orders:

- Big-O: $O(f(n))$, upper bound for running time
- Big-Ω: $\Omega(f(n))$, lower bound for running time
- Big-Θ: $\Theta(f(n))$, upper bounded by O and lower bounded by Ω

Efficiency:

- Efficient algorithms: $O(n^k)$ for some k
- Inefficient algorithms: $\Omega(c^n)$ for some c

2.2 Church-Turing Revisited

- **Classic Church-Turing Thesis**: "The class of functions computable by a Turing Machine is equivalent to the class of functions computable by means of an algorithm."
- **Strong Church-Turing Thesis**: "Every *realistic* model of computation is polynomial time reducible to a Universal TM."
- **Extended Church-Turing Thesis**: "Any algorithmic process can be efficiently simulated using a Probabilistic TM."
- **Physical Church-Turing Thesis**: "Any physical process can be realized by a TM."

Quantum computers don't challenge the CT-Thesis, but they do challenge the Extended CT-Thesis.

2.3 Characterizing Computation

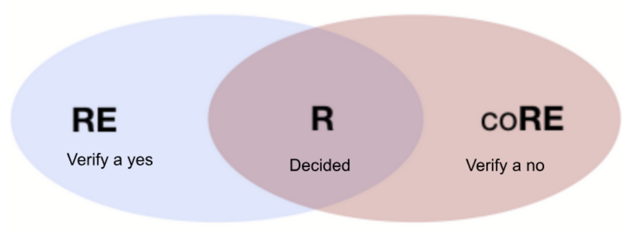
- TIME($f(n)$): Class of problems that grows in $O(f(n))$ in time for input length n
- SPACE($f(n)$): Class of problems that grows in $O(f(n))$ in space (memory)

TIME($f(n)$) $\subseteq$ SPACE($f(n)$), since a TM can only access 1 piece of memory in each time step.

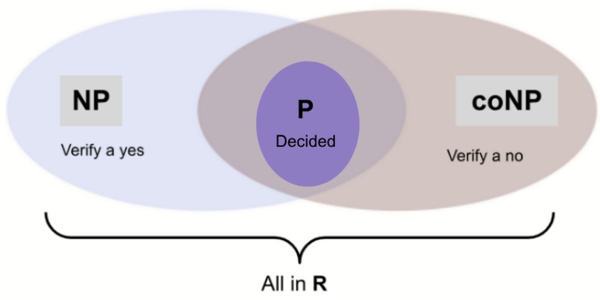
Classes:

- P: class of problems solvable by TM in TIME(n^k)
- $PSPACE$: class of problems solvable by TM in SPACE(n^k)
- EXP: class of problems solvable by TM in TIME(2^{n^k}) — infeasible, inefficient
- NP: class of problems for which proof of *yes* result is in TIME(n^k)
- $coNP$: class of problems for which proof of *no* result is in TIME(n^k)

Known: $P \subseteq NP \subseteq PSPACE$ Conjectured: $P \neq NP \neq PSPACE$



What can be computed?



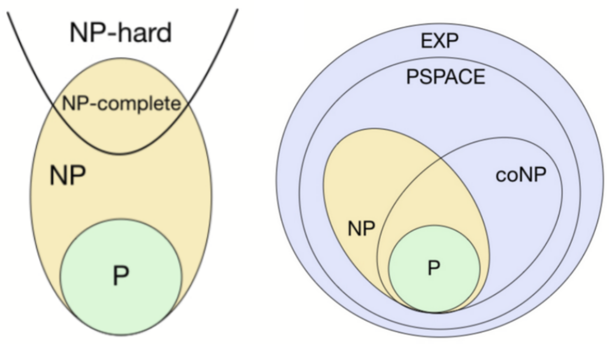
What can be computed *efficiently* (polynomial time)?

Note that $NP \cap coNP \neq P$.

2.4 More on the NP

NB: "language" = "decision problem"

- *NP-hard*: class of languages L for which any problem $L' \in NP$ is polynomial-time reducible to L
  - i.e. Any problem in *NP* can be reduced to an *NP-hard* problem in polynomial time.
- *NP-complete*: class of languages L for which $L \in$ *NP-hard* and $L \in NP$
  - "the hardest" problems in NP



Assuming $P \neq NP$

2.5 Boolean Satisfaction Problem (SAT)

- Given Boolean variables $x_1, x_2, ..., x_n$ and Boolean operators $\wedge, \vee, \neg$
- Conjunctive Normal Form (CNF): $\vee$ inside parentheses, $\wedge$ out. E.g. $(x_1 \vee x_3 \vee \neg x_4 \vee x_5) \wedge (x_2 \vee \neg x_4)...$

SAT-decision: Given a Boolean formula in CNF, does there exist $(x_1, x_2, ..., x_n)$ that satisfies the expression (eval to 1)?

Cook-Levin Theorem: SAT is NP-Complete. That is, *all NP problems can be reduced to SAT.*

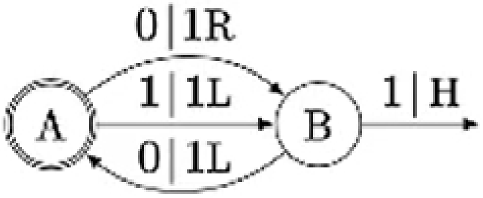
3 Week 3 (Sep 7 - Sep 13)

3.1 Non-Deterministic Turing Machines (NDTMs)

A deterministic TM (DTM) can be represented as

$$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$$

- δ: transition function
- Q: set of states
- Γ: alphabet

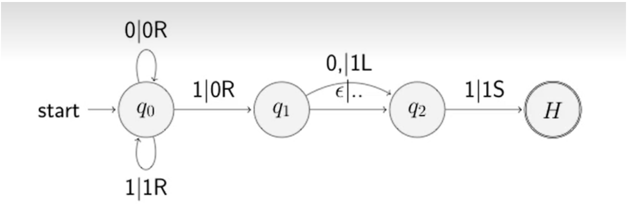


There's exactly one arrow out of each state for each transition (0 or 1).

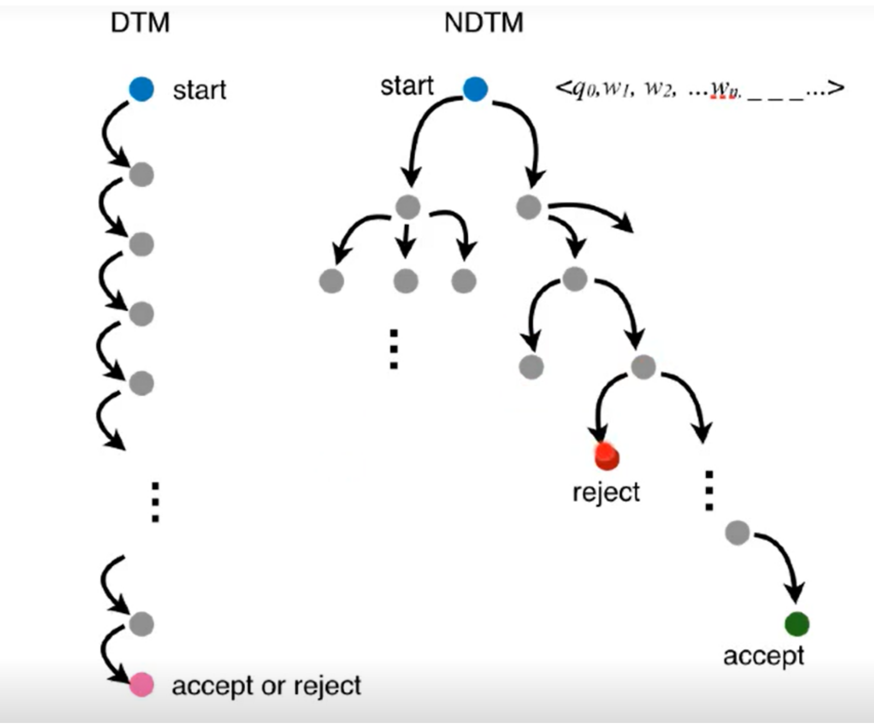
An NDTM can be represented as

$$\delta : Q \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$$

- The power set of all possible transitions.
- Imagine at every decision point, the Turing Machine clones itself and runs in parallel.



Can be more than one arrow out of each state for each transition (0 or 1).



Comparison of DTM and NDTM

Theorem: Any language accepted by an NDTM will be accepted by a DTM.

3.2 P and NP: Alternate Definitions

- **P**: class of languages decidable in polynomial time on single-tape DTM.

$$P = \bigcup_k TIME(n^k)$$

- **NP**:
  1) class of languages that have polynomial time verifiers.
     - Verifier for language A is algorithm V
     - $A = \{w|V$accepts <w, c> for some string c$\}$

    - Polynomial time verifier runs in $|w|^k$
2) class of languages that are decidable in polynomial time on a single-tape NDTM:

$$NP = \bigcup_k NTIME(n^k)$$

*Recall:

- **NP-hard** is the class of languages L for which any problem $L' \in NP$ is polynomial-time reducible to L.
- **NP-complete** is the class of languages that are
    1) $L \in$ *NP-hard*
    2) $L \in NP$

3.3 SAT → 3SAT

Recall the SAT problem: Given a Boolean formula in CNF, e.g.

$$(x_1 \vee x_3 \vee \neg x_4 \vee x_5) \wedge (x_2 \vee \neg x_4) \wedge (x_3 \vee x_{12} \vee \neg x_{12}) \wedge ...$$

Does there exist $(x_1, x_2, ..., x_n)$ that satisfies the expression?

We can reduce SAT to 3SAT (3 variables per AND clause).

- Use dummy variables, e.g:
- $(x_1 \vee x_2 \vee x_3 \vee x_4) \longrightarrow (x_1 \vee x_2 \vee y_1) \wedge (\neg y_1 \vee x_3 \vee x_4)$
- $(x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5) \longrightarrow (x_1 \vee x_2 \vee y_1) \wedge (\neg y_1 \vee x_3 \vee y_2) \wedge (\neg y_2 \vee x_4 \vee x_5)$

Obviously in *NP*, but is it *NP-complete*?

3.4 Cook-Levin Theorem Proof

Follows Sipser chp. 7

Cook-Levin Theorem: SAT is NP-complete.

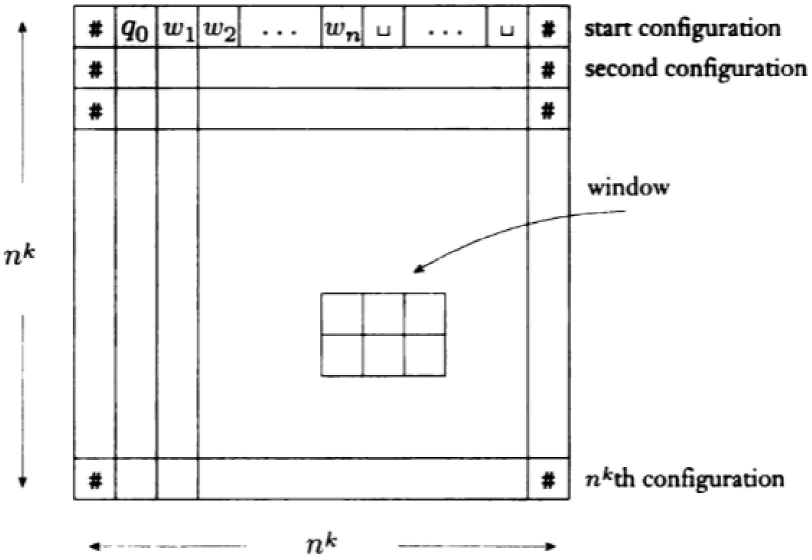
1) SAT is in *NP* (easy)
2) For all $A \in NP$, A is polynomial-time reducible to SAT (hard)

We need to show the second part.

3.4.1 The Table

Reduction for A: Boolean formula ϕ that, for the input w, simulates an NDTM for A on w.

Let N be a NDTM that solves A in $O(n^k)$. We need to show that all NDTMs can be reduced to SAT in polynomial time. Represent N by a $n^k \times n^k$ table:



First row is initial state. Each row is the result of applying a delta transition to the row above. Boundaries are marked with #.

- Every accepting table for N on w corresponds to a computation branch of N on w.
- Whether N exists = Whether accepting table for N on w exists.

3.4.2 The Formula

$$\phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{move} \wedge \phi_{accept}$$

- ϕ_{cell}: each element in table has just 1 symbol.
- ϕ_{start}: start state is well-defined.
- ϕ_{move}: transitions between rows are legal.
- ϕ_{accept}: at least 1 row is the accept state (q_{accept}).

Let $x_{i,j,s} = 1$ if cell$[i, j]$ has symbol s, and $x_{i,j,s} = 0$ otherwise.

$$
\begin{aligned}
\phi_{cell} &= \bigwedge_{1 \le i,j \le n^k} \left[\underbrace{\left(\bigvee_{s \in C} x_{i,j,s} \right)}_{\text{at least 1 var in each cell}} \wedge \underbrace{\left(\bigwedge_{\substack{s,t \in C \\ s \neq t}} (\bar{x_{i,j,s}} \vee \bar{x_{i,j,t}}) \right)}_{\text{only 1 var in each cell}} \right] \\
\phi_{start} &= x_{1,1,\#} \wedge x_{1,2,q_0} \wedge \\
&= x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \ldots \wedge \\
&= x_{1,n+3,\sqcup} \wedge \ldots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#} \\
\phi_{move} &= \bigwedge_{1 \le i,j \le n^k} \text{(the 2 x 3 window at (i, j) is a legal transition)} \\
&= \bigvee_{a_1 \ldots a_6} x_{i,j,a_i} \qquad \text{(for each cell } x_{i,j} \text{ and its symbol } a_i\text{)} \\
\phi_{accept} &= \bigvee_{1 \le i,j \le n^k} x_{i,j,q_{accept}}
\end{aligned}
$$

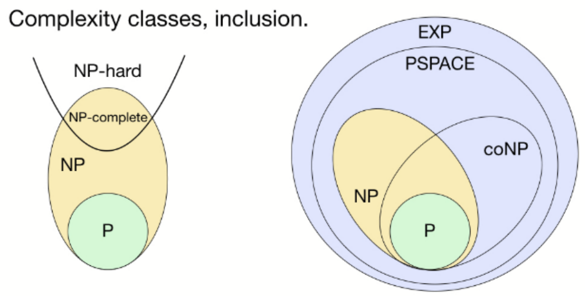
3.4.3 Analysis

- ϕ_{cell}: $O(n^{2k})$
- ϕ_{start}: $O(n^k)$
- ϕ_{move}, ϕ_{accept}: $O(n^{2k})$

Total: ϕ can be found in $O(n^{2k})$, a polynomial factor reduction.

Therefore, any NP problem can be reduced to SAT in polynomial time, and SAT is NP-complete.

3.5 Complexity Class Relations (so Far)



$$P \subseteq NP \subseteq PSPACE = NSPACE \subseteq EXP \text{ but } P \subset EXP \ldots$$

- *PSPACE*: class of languages solvable by DTM in polynomial space

- *NSPACE*: class of languages solvable by NDTM in polynomial space

(Click for Complexity Class Zoo Link)

4 Week 4 (Sep 14 - Sep 20)

4.1 Review: Complexity Classes

$$P \subseteq NP \subseteq \mathit{PSPACE} = \mathit{NSPACE} \subseteq \mathit{EXP}$$
$$P \subset \mathit{EXP}$$

4.2 PSPACE-hard & PSPACE-complete

- **PSPACE-hard**: All **PSPACE** problems can be polynomial-time reduced to **PSPACE-hard** problems.
- **PSPACE-complete**: Problems that are **PSPACE-hard** and in **PSPACE**.

Quantified Boolean Formulas (QBF): Boolean SAT problem with quantifiers ∀, ∃.

Truely Quantified Boolean Formula (TQBF): Determine if a full QBF is true or false.

$$\{\langle \phi \rangle \mid \phi \text{ is a true fully QBF}\}$$

The following limitations for a true, full QBF:

- Prenex normal form: all ∀, ∃ grouped at start of expression.
- Fully quantified: all variables bound by ∀, ∃.

4.3 Probabilistic Turing Machines

Deterministic TM (DTM):

$$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$$

Non-deterministic TM (NDTM):

$$\delta : Q \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$$

$$\mathcal{P} \text{ is power set}$$

Probabilistic TM (PTM):

$$\delta : Q \times \Gamma \longrightarrow \{Q \times \Gamma \times \{L, R\}, Q \times \Gamma \times \{L, R\}\}$$

2 choices

Bounded-error Probabilistic Polynomial time (BPP)

A string ω in the language L that the PTM accepts/rejects in polynomial-time probabilistically.

- $\omega \in L$: PTM accepts with prob. $\geq \frac{1}{2}$
- $\omega \notin L$: PTM rejects with prob. $\geq \frac{1}{2}$

Conjectured that **BPP** $=$ **P**, but no proof yet.

4.4 *NP* $\cap$ *co-NP*

PRIMES: Is a natural number prime or composite?

- In **NP** and **co-NP**
- Found to be in **P** in 2002 (AKS primality test)

FACTORING: is a given number a factor of another?

- In **NP** and **co-NP**
- Suspected not to be in **P** (no proof)

4.5 Reversible Computation

QM employs unitary evolution (reversible computation).

Classical computation can be made reversible by using *ancilla bits*.

5 Week 5 (Sep 21 - Sep 27)

5.1 Postulates of Quantum Mechanics

1. **Quantum State**
    - Reprsented by normalized ket $|\psi\rangle$
    - Contains all information that can be known about system
    - In Hilbert space (space of all possible quantum states)
2. **Observables**
    - Physical properties that can be measured in an experiment
    - Represented by Hermitian operators that act on Hilbert space

3. **Measurement**
   - Probability of obtaining eigenvalue a_n in measurement of observable A on the system in state $|\psi\rangle$ is $\mathcal{P}(a_n) = |\langle a_n|\psi\rangle|^2$
   - Note: $|a_n\rangle$ is eigenvector corresponding to eigenvalue a_n.
4. **Time Evolution**
   - For a closed quantum system, described by unitary transformation.
   - $|\psi'\rangle = U|\psi\rangle$

5.2 Quantum Computing Basics

5.2.1 Different Basis

- Computational basis (HV): $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$
- Hadamard basis (PM): $|+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $|-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix}$
- RL basis not used much.

5.2.2 Quantum Gates

Unitary operators, act on a qubit.

- $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$: HV observable; flip qubit
- $Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$: RL observable
- $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$: PM observable; phase shift by π: $Z = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi} \end{pmatrix}$
- $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$: turns state into superposition of computational basis
- $S = \sqrt{Z} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$, another phase shift
- $T = \sqrt{S} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$, phase shift by $\pi/4$ (but called $\pi/8$ for historical reasons)

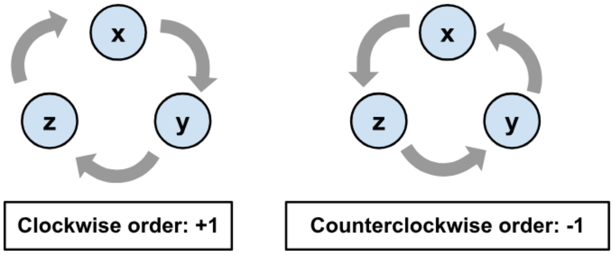
5.3 Relationship Between Gates

Pauli matrices anti-commute: $XY = iZ$, but $YX = -iZ$, for example.

$$\sigma_a\sigma_b = i\epsilon_{abc}\sigma_c$$

Also, $H = \frac{1}{\sqrt{2}}(X + Z)$.

The anti-commutation operator works a little weirdly:



E.g. $\epsilon_{yzx} = +1$ because clockwise $y \to z \to x$

When dealing with products, it's usually faster to use tricks instead of using matrices. E.g.

$$XHX = \frac{1}{\sqrt{2}}X(X + Z)X = \frac{1}{\sqrt{2}}(X^3 + XZX) = \frac{1}{\sqrt{2}}(X - iYX)\text{...etc}$$

5.4 Other Important Gates

CNOT gate: Flips second qubit if the first qubit is 1.

- $\text{CNOT} = \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & X \end{array}\right] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
- A white circle instead of a filled-in one means the reverse: flip if first qubit is 1.

Toffoli gate: Flips 3rd qubit if both the 1st and 2nd are 1.

5.5 The Hadamard Gate

In computational basis:

- $H\left|0\right\rangle = \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + \left|1\right\rangle\right) = \left|+\right\rangle$
- $H\left|1\right\rangle = \frac{1}{\sqrt{2}}\left(\left|0\right\rangle - \left|1\right\rangle\right) = \left|-\right\rangle$
- $H\left|+\right\rangle = \left|0\right\rangle$
- $H\left|-\right\rangle = \left|1\right\rangle$

More generally, it creates a superposition of all inputs, a normalized sum of all n-length bitstrings ("Walsh-Hadamard Transform"):

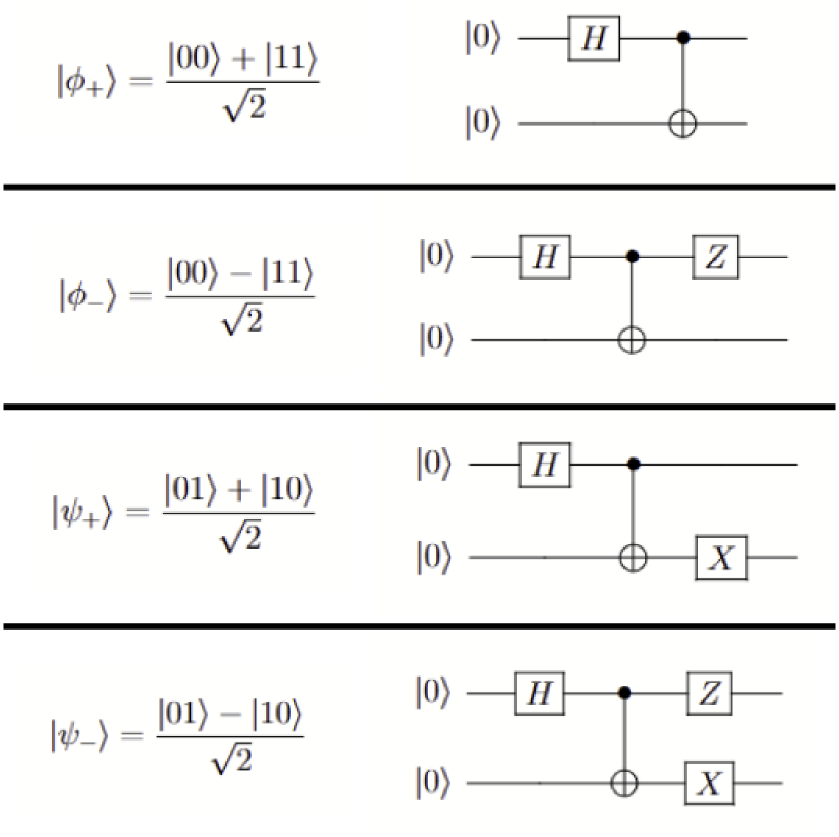
$$H \otimes H \otimes ... \otimes H \left|00...0\right\rangle = H^{\otimes n} \left|0\right\rangle_n = \frac{1}{2^{n/2}} \sum_{y_0=0}^{1} \left|y_{n-1}\right\rangle ... \left|y_0\right\rangle$$

6 Week 6 (Sep 28 - Oct 4)

6.1 Bell Basis

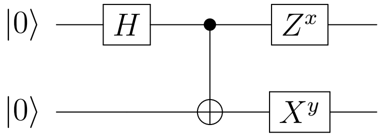
6.1.1 Constructing the Bell Basis

Four entangled states:


$|\phi_+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$
$|0\rangle$
$|0\rangle$
H
$|\phi_-\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}$
$|0\rangle$
$|0\rangle$
H
Z
$|\psi_+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}$
$|0\rangle$
$|0\rangle$
H
X
$|\psi_-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}$
$|0\rangle$
$|0\rangle$
H
Z
X

Bell States & Construction with Quantum Circuits

More simply,

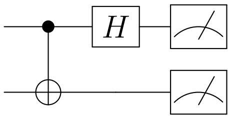


Here,

- $(x, y) = (0, 0) : |\phi_+\rangle$
- $(x, y) = (1, 0) : |\phi_-\rangle$
- $(x, y) = (0, 1) : |\psi_+\rangle$
- $(x, y) = (1, 1) : |\psi_-\rangle$

6.1.2 Measuring the Bell Basis

How to know what basis you have?



In this circuit:

- $|\phi_+\rangle$ gives $|00\rangle$
- $|\phi_-\rangle$ gives $|10\rangle$
- $|\psi_+\rangle$ gives $|01\rangle$
- $|\psi_-\rangle$ gives $|11\rangle$

6.2 Quantum Teleportation

Initially, Alice has an unknown qubit $a\,|0\rangle + b\,|1\rangle$. Alice and Bob share an entangled pair $|00\rangle + |11\rangle$. Ignore normalization factors.

Total state:

$$|\psi_1\rangle = a\,|000\rangle + b\,|100\rangle + a\,|011\rangle + b\,|111\rangle$$

Alice can "teleport" her qubit to Bob's using the entangled pair:

1. Alice applies CNOT to her entangled pair, using unknown qubit as control.

$$|\psi_2\rangle = a\,|000\rangle + b\,|110\rangle + a\,|011\rangle + b\,|101\rangle$$

2. Alice applies H to unknown qubit.

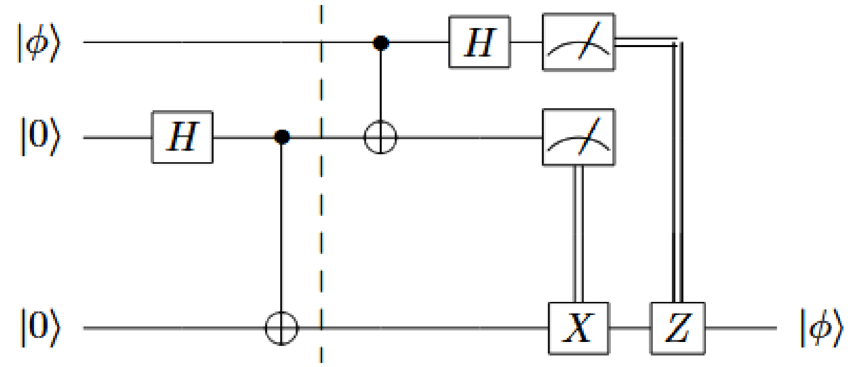
$$\begin{aligned}
\left|\psi_3\right\rangle &= a\left(|0\rangle+|1\rangle\right)|00\rangle + b\left(|0\rangle-|1\rangle\right)|10\rangle + a\left(|0\rangle+|1\rangle\right)|11\rangle + b\left(|0\rangle-|1\rangle\right)|01\rangle \\
&= |00\rangle\left(a|0\rangle+b|1\rangle\right) + |01\rangle\left(a|1\rangle+b|0\rangle\right) + |10\rangle\left(a|0\rangle-b|1\rangle\right) + |11\rangle\left(a|1\rangle-b|0\rangle\right)
\end{aligned}$$

3. Alice measures her qubits and sends to Bob classically (like email).
4. Depending on Alice's measurements, Bob applies some operators to his entangled qubit.
    - If Bob receives $|00\rangle$: his state is $a|0\rangle+b|1\rangle$, apply identity.
    - If Bob receives $|01\rangle$: his state is $a|1\rangle+b|0\rangle$, apply X.
    - If Bob receives $|10\rangle$: his state is $a|0\rangle-b|1\rangle$, apply Z.
    - If Bob receives $|11\rangle$: his state is $a|1\rangle-b|0\rangle$, apply X then Z.

In this way, Bob can recover Alice's qubit $a|0\rangle+b|1\rangle$.

NB: This isn't an instantaneous transmission of information, since Alice needs to send her results to Bob classically.

NB 2: This doesn't violate the no-cloning theorem, since the original qubit state is destroyed when Alice measures it.



Quantum Teleportation Rail Diagram

Double lines mean classical transmission of bits. In particular, here they mean we apply the gate on the bottom (X or Z) if the measurement is 1.

6.3 Superdense Coding

We can send **2 bits** of information by sending just 1 qubit, if an entangled resource exists.

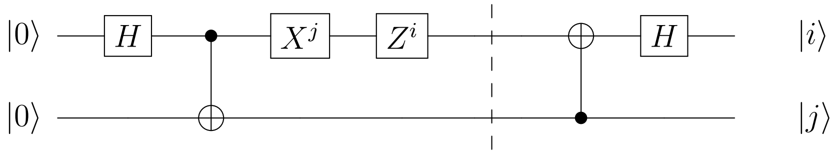
Say Alice and Bob share an entangled resource $|\phi\rangle = |00\rangle + |11\rangle$. Alice wants to send two bits (i, j) to Bob. She encodes them as:

$$Z^i X^j \otimes 1 \left|\phi\right\rangle$$

This produces the Bell basis:

- $Z^0 X^0 \otimes 1 \left|\phi\right\rangle = \left|00\right\rangle + \left|11\right\rangle$
- $Z^0 X^1 \otimes 1 \left|\phi\right\rangle = \left|10\right\rangle + \left|01\right\rangle$
- $Z^1 X^0 \otimes 1 \left|\phi\right\rangle = \left|00\right\rangle - \left|11\right\rangle$
- $Z^1 X^1 \otimes 1 \left|\phi\right\rangle = \left|01\right\rangle - \left|10\right\rangle$

Bob can then recover the two qubits by applying a CNOT and Hadamard:



Left of dotted line: entangle qubits, apply X and Z. Right of dotted line: Bob recovers Alice's bits.

6.4 Entanglement Swapping

We can "transfer" entanglement between two sets of qubits to unentangled qubits.

Say Alice and Chuck share $\left|\Phi_+\right\rangle$, and Bob and Chuck share a $\left|\Phi_+\right\rangle$ (entangled states).

If Chuck does a Bell State measurement on his state, he can make Alice and Bob's qubits become entangled:

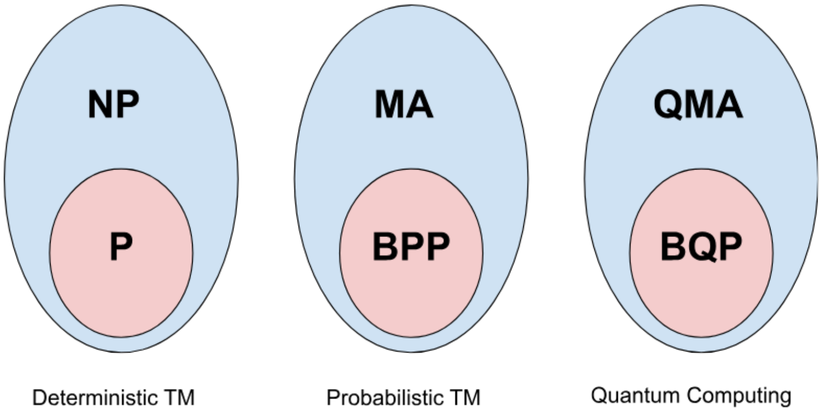
$$\begin{aligned} \left|\Phi_+\right\rangle_{AC} \left|\Phi_+\right\rangle_{CB} &= \frac{1}{2}\left(\left|00\right\rangle_{AC} + \left|11\right\rangle_{AC}\right)\left(\left|00\right\rangle_{CB} + \left|11\right\rangle_{CB}\right) \\ &= \frac{1}{2}\left(\left|0000\right\rangle + \left|0011\right\rangle + \left|1100\right\rangle + \left|1111\right\rangle\right)_{ACCB} \end{aligned}$$

After grouping terms:

$$\frac{1}{4}\left[\left|0\right\rangle\left(\left|\Phi_+\right\rangle + \left|\Phi_-\right\rangle\right)\left|0\right\rangle + \left|0\right\rangle\left(\left|\Psi_+\right\rangle + \left|\Psi_-\right\rangle\right)\left|1\right\rangle + \left|1\right\rangle\left(\left|\Psi_+\right\rangle - \left|\Psi_-\right\rangle\right)\left|0\right\rangle + \left|1\right\rangle\left(\left|\Phi_+\right\rangle - \left|\Phi_-\right\rangle\right)\left|1\right\rangle\right]_{ACCB}$$
$$\frac{1}{4}\left[\left|\Phi_+\right\rangle_{CC}\left(\left|00\right\rangle + \left|11\right\rangle\right)_{AB} + \left|\Phi_-\right\rangle\left(\left|00\right\rangle - \left|11\right\rangle\right) + \left|\Psi_+\right\rangle\left(\left|01\right\rangle + \left|10\right\rangle\right) + \left|\Psi_-\right\rangle\left(\left|01\right\rangle - \left|10\right\rangle\right)\right]$$

Note that Alice and Bob's qubits end up in an entangled Bell state.

6.5 Probabilistic & Quantum Complexity Classes



Equivalent of P and NP complexity classes for Probabilistic TMs & Quantum computers, where P is what can be proved and NP is what can be verified (as usual). This is assuming that $P \neq NP$.

Nitty gritty details are below if you're a masochist (or a mathematician). Not too important as long as you get the rough idea.

6.5.1 Probabilistic TMs: *BPP* and *MA*

BPP: Bounded-error Probabilistic Polynomial-Time, analogue of P for a probabilistic TM.

- A language $L \in BPP$ if there exists a probabilistic TM "M" such that:
  1. M runs for polynomial-time on all inputs.
  2. (Completeness) For all $x \in L$, M outputs 1 with probability $\geq 2/3$.
  3. (Soundness) For all $x \not\in L$, M outputs 1 with probability $\leq 1/3$.
- Alternative definition with deterministic TMs: A language $L \in BPP$ if there exists a polynomial p and deterministic TM "M" such that:
  1. M runs in polynomial-time for all inputs
  2. (Completeness) For all $x \in L$, the fraction of strings y with length $p(|x|)$ which satisfy $M(x, y) = 1$ is $\geq 2/3$.
  3. (Soundness) For all $x \not\in L$, the fraction of strings y with length $p(|x|)$ which satisfy $M(x, y) = 1$ is $\geq 1/3$.
- The 2/3 doesn't really matter, as long as it's higher than 1/2. We can get arbitrarily close to 1 by doing multiple runs.

MA: Merlin-Arthur, analogue of NP for a probabilistic TM.

- A language $L \in MA$ if there exists a deterministic TM "M" and polynomials p, q, such that for for every input string x,
  1. If $x \in L$, then there exists a certificate $z \in \{0,1\}^{q(n)}$ such that $\text{Prob}_{y\in\{0,1\}^{p(n)}}(M(x,y,z) = 1) \geq 2/3$
  2. If $x \not\in L$, then for every certificate $z \in \{0,1\}^{q(n)}$, $\text{Prob}_{y\in\{0,1\}^{p(n)}}(M(x,y,z) = 0) \geq 2/3$
- Definitions were pulled from Wikipedia. In a nutshell: certificate exists to verify result with probability $\geq 2/3$.
- The naming: Merlin tries to convince Arthur of the correct answer, but Arthur knows Merlin isn't trustworthy. Merlin supplies a certificate such that the answer can be verified.

6.5.2 Quantum Computing: *BQP* and *QMA*

BQP: Bounded-error Quantum Polynomial-time, equivalent of P for quantum computing.

- A language $L \in BQP$ if there exists a polynomial-time family of circuits $\{Q_n : n \in \mathbb{N}\}$ such that:
  1. For all $n \in \mathbb{N}$, Q_n takes n qubits as input and outputs 1 bit.
  2. (Completeness) For all $x \in L$, $\text{Prob}(Q_{|x|}(x) = 1) \geq 2/3$.
  3. (Soundness) For all $x \not\in L$, $\text{Prob}(Q_{|x|}(x) = 0) \geq 2/3$.
- Definitions from Wikipedia. Similar to Probabilistic TMs, but with a quantum circuit.

QMA: Quantum Merlin-Arthur, equivalent of NP for quantum computing.

- A language $L \in QMA$ if there exists a polynomial-time quantum verifier V and a polynomial p such that:
  1. For all $x \in L$, there exists a quantum state $|\psi\rangle$ such that the probability of V accepting the input $(|x\rangle, |\psi\rangle)$ is $\geq 2/3$.
  2. For all $x \not\in L$, and for all quantum states $|\psi\rangle$ with at most $p(|x|)$ qubits, the probability of V accepting the input $(|x\rangle, |\psi\rangle)$ is $\leq 1/3$.
- Definitions from Wikipedia. The key part is the quantum verifier.

7 Week 7 (Oct 5 - Oct 11)

7.1 Resource Theory

- Classical communication: sending 1 bit $[c \to c]$, sending 2 bits $2[c \to c]$
- Quantum communication: sending 1 qubit $[q \to q]$
- Entangled qubits: $[qq]$
- GHZ state: $[qqq]$

Some results:

- $[q \to q] \geq [c \to c]$: Sending qubits can simulate classical communication.
- $[q \to q] \not\geq 2[c \to c]$: Holveo bound, can only communicate 1 bit of information.
- $[q \to q] \geq [qq]$: Sending qubits can simulate entanglement.
- $[qq] \not\geq [c \to c]$: No-signaling theorem
- $[qq] \not\geq [q \to q]$: No-signaling theorem

Quantum teleportation: $[qq] + 2[c \to c] \geq [q \to q]$

Superdense coding: $[qq] + [q \to q] \geq 2[c \to c]$

7.2 Quantum Computation

7.2.1 Boolean Logic

Given a boolean function $0 \leq f(a) \leq 2^n - 1$, where a is a n-bit string, the goal is to compute $|a\rangle \to U_f |a\rangle = |f(a)\rangle$, where U_f is a unitary gate.

The problem is that most functions aren't injective (one-to-one). E.g. for $f(a) = a \mod 2$,

$$\begin{aligned} 2 = |010\rangle &\longrightarrow U_f |010\rangle = |000\rangle \\ 4 = |100\rangle &\longrightarrow U_f |100\rangle = |000\rangle \end{aligned}$$

So $\langle f(2)|f(4)\rangle = 1$, but also $\langle f(2)|f(4)\rangle = \langle 010| U_f^\dagger U_f |100\rangle = \langle 010|100\rangle = 0$.

Thus $U_f |a\rangle \neq |f(a)\rangle$.

To resolve this (by making U_f reversible), use an ancilla qubit to set

$$|a\rangle \otimes |b\rangle \longrightarrow U_f |a\rangle \otimes |b\rangle = |a\rangle \otimes |b \oplus f(a)\rangle$$

This ensures that outputs are reversible.

7.2.2 Oracles

An oracle is a unitary gate like above:

- Unknown function f, but we can "query" it.
- Efficiency of oracle isn't considered
- For QC: $O_f |a\rangle = |a \oplus f(a)\rangle$

1. Any quantum algorithm for a problem gives a lower bound on complexity.
    - The query to oracle is minimum 1 step in the algorithm.

- If the quantum algorithm can't be done efficiently with an oracle, it can't be done efficiently in general.

2. Relativized speedups: We can prove fast quantum algorithms relative to an oracle.
3. If we can instantiate an oracle in polynomial space w.r.t input n, then that proves there's a quantum speedup.

In a nutshell, if a quantum algorithm seems faster than its classical counterpart with an oracle, *maybe* there's a quantum speedup. But it depends on the oracle.

7.3 Deutsch's Algorithm

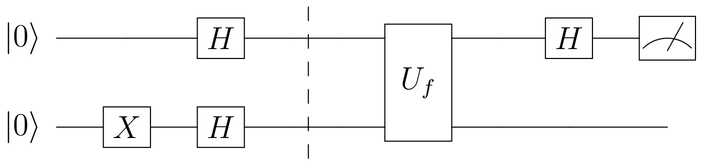
Kind of a toy algorithm, but shows a quantum algorithm faster than its classical counterpart.

Given: 1 input bit, boolean function f (our oracle)

Goal: determine if f is constant or balanced in the least number of queries.

- Constant: always give same output, e.g. $f(0) = f(1) = 0$ or $f(0) = f(1) = 1$.
- Balanced: give different outputs, e.g. $f(0) = 0$ and $f(1) = 1$, or $f(0) = 1$ and $f(1) = 0$

Classically, we need 2 queries. A quantum algorithm needs 1.



Before the dotted line, we're generating the state $|\psi\rangle = |x\rangle |y\rangle = (|0\rangle + |1\rangle)(|0\rangle - |1\rangle)/2$. Our oracle is $U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$.

Calculating the effect of U_f and ignoring normalization,

$$\begin{aligned}
U_f |\psi\rangle &= U_f |0\rangle (|0\rangle - |1\rangle) + U_f |1\rangle (|0\rangle - |1\rangle) \\
&= |0\rangle \Big(|0 \oplus f(0)\rangle - |1 \oplus f(0)\rangle \Big) + |1\rangle \Big(|0 \oplus f(1)\rangle - |1 \oplus f(1)\rangle \Big) \\
&= (-1)^{f(0)} |0\rangle (|0\rangle - |1\rangle) + (-1)^{f(1)} |1\rangle (|0\rangle - |1\rangle) \\
&= \Big[(-1)^{f(0)} |0\rangle + (-1)^{f(1)} |1\rangle \Big] (|0\rangle - |1\rangle)
\end{aligned}$$

If f is constant: $U_f |\psi\rangle = \pm(|0\rangle + |1\rangle)(|0\rangle - |1\rangle)$.

If f is balanced: $U_f |\psi\rangle = \pm(|0\rangle - |1\rangle)(|0\rangle - |1\rangle)$.

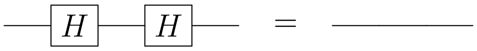
If we apply the Hadamard on the first qubit,

$$(H \otimes 1)U_f \left|\psi\right\rangle = \pm \left|z\right\rangle (\left|0\right\rangle - \left|1\right\rangle)$$

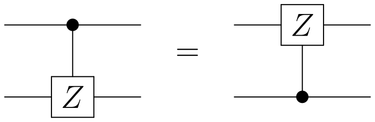
$z = 0$ if constant, $z = 1$ if balanced. So measuring the first qubit determines whether it's constant or balanced, in just 1 oracle query.

7.4 More Gate Identities

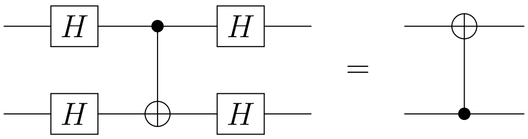
Application of two Hadamards is the identity ($H^2 = I$)



Controlled Z gate can be flipped. Doing the casework, the only change occurs when $\left|11\right\rangle \longrightarrow -\left|11\right\rangle$, and symmetry.



Surrounding a CNOT with Hadamards flips the CNOT. This can be proved with the previous identity & the fact that $HXH = Z$, $HZH = X$.



8 Week 8 (Oct 12 - Oct 18)

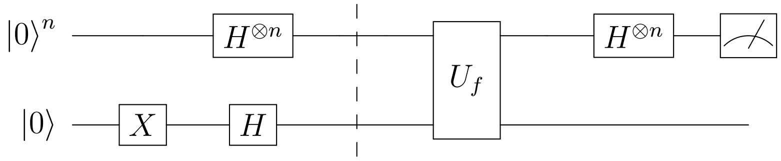
8.1 Deutsch-Jozsa Algorithm

An extension of Deutsch's algorithm with n input bits and appropriate ancillas.

Same problem as before: given a function f that's either constant (all same output) or balanced (half 0, half 1 output), in how many queries can we tell? But now with n bit inputs.

- Classically: best 2 queries, worst $2^{n-1} + 1$ queries.
- Quantum: 1 query

Circuit is just Deutsch's algorithm except with n-bit Hadamards:



By the dashed line, our state is

$$|\psi_{in}\rangle = \frac{1}{2^{(n+1)/2}} \sum_{x \in \{0,1\}^n} |x\rangle \Big(|0\rangle - |1\rangle \Big)$$

Applying the oracle,

$$U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$$
$$U_f |\psi_{in}\rangle = \frac{1}{2^{(n+1)/2}} \sum_x (-1)^{f(x)} |x\rangle \Big(|0\rangle - |1\rangle \Big)$$

The -1 factor is a phase kickback, see Deutsch's algorithm section for more.

Now recall the action of the Hadamard:

$$\text{1 qubit: } H|x\rangle = \frac{1}{\sqrt{2}} \sum_z (-1)^{xz} |z\rangle$$
$$\text{multiple qubits: } H^{\otimes n} |x\rangle = \frac{1}{2^{n/2}} \sum_z (-1)^{x \cdot z} |z\rangle, \; x \cdot z = x_1 z_1 + x_2 z_2 + ... + x_n z_n$$
$$\text{on our state: } \Big(H^{\otimes n} \otimes 1 \Big) \Big[U_f |\psi_{in}\rangle \Big] = \frac{1}{2^n} \sum_x \sum_z (-1)^{x \cdot z + f(x)} |z\rangle \Big(|0\rangle - |1\rangle \Big)$$

Measure the first term of the sum, $|0\rangle^{\otimes n}$, where $z = 0$. It has amplitude $\sum_x \frac{(-1)^{f(x)}}{2^n}$.

- If f is constant: $\sum_x \frac{(-1)^{f(x)}}{2^n} = (-1)^f \sum_x \frac{1}{2^n} = (-1)^f$. State is normalized, therefore $|0\rangle^{\otimes n}$ is the only term.
- If f is balanced: $\sum_x \frac{(-1)^{f(x)}}{2^n} = 0$, equal number of +1s and -1s. So $|0\rangle^{\otimes n}$ doesn't appear.

8.2 3SAT with a Quantum Circuit

The 3SAT problem: Given a CNF boolean function with clauses containing 3 literals, does there exist a satisfying assignment?

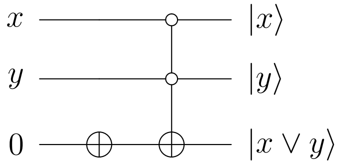
CNF = conjunctive normal form, OR inside parentheses and AND outside

e.g. $\Phi = \Big(x_1 \vee x_2 \vee \neg x_3 \Big) \wedge \Big(\neg x_1 \vee \neg x_2 \vee x_4 \Big) \wedge \Big(x_2 \vee \neg x_4 \vee x_3 \Big)$

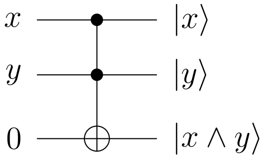
8.2.1 Implementation of AND and OR

Easy enough, based on the Toffoli gate.

The OR circuit:



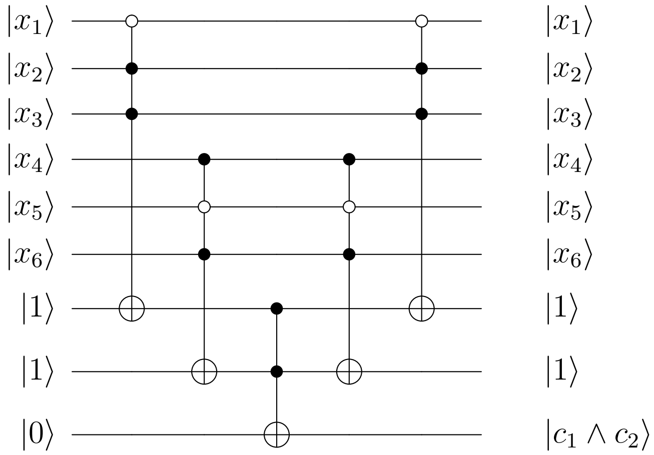
The AND circuit:



8.2.2 Combined Circuit

To make a circuit for the 3SAT, we "stack" the appropriate circuits, get each parentheses' result in an ancilla, and peform a CNOT on all those ancillas.

Here's an example for $\Big(x_1 \vee \neg x_2 \vee \neg x_3\Big) \wedge \Big(\neg x_4 \vee x_5 \vee \neg x_6\Big)$:



These control gates mix both open circles and closed ones. But they're easy to implement by adding a X gate appropriately. Note that we "retrace" our steps and uncompute the work register, because we don't want to mess with that.

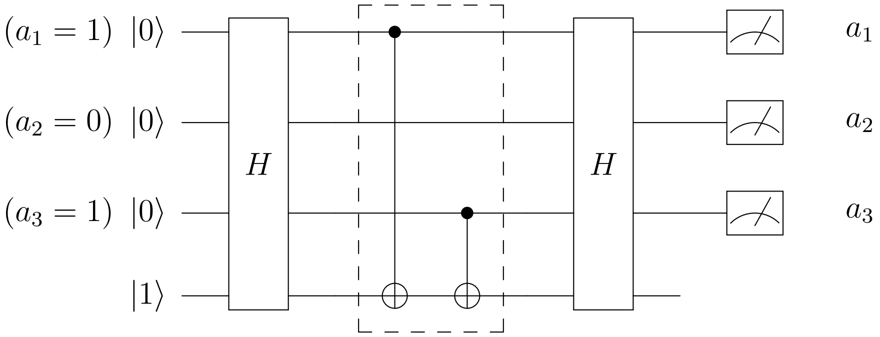
8.3 Bernstein-Vazirani Algorithm

Given a function $f(x)$ that evaluates the mod-2 inner product, $f(x) = a \cdot x = a_0x_0 \oplus a_1x_1 \oplus ... \oplus a_nx_n$

What's the least amount of calls to the oracle that we need to determine a?

- Classically: n queries. Quantum: 1 query.
- The oracle, as always, is $U_f \left|x\right\rangle \left|y\right\rangle = \left|x\right\rangle \left|y \oplus f(x)\right\rangle$

Circuit diagram:



This is an example with $a = 5 = 101_2$. The dashed section is the oracle. A CNOT is applied to the ancilla if the corresponding bit is 1.

We Hadamard all initial $|0\rangle$ qubits, query the oracle, un-Hadamard, and measure. Why does this work?

8.3.1 Mathematics

After Hadamarding, our input to the oracle is

$$\left|\psi_{in}\right\rangle = H^{\otimes n} \left|0\right\rangle^n H \left|1\right\rangle = \frac{1}{2^{(n+1)/2}} \sum_x \left|x\right\rangle^n \Big(\left|0\right\rangle - \left|1\right\rangle \Big)$$

We apply the phase kickback from the oracle. The $\Big(\left|0\right\rangle - \left|1\right\rangle \Big)$ is flipped if the control qubit is 1:

$$U_f \left|\psi_{in}\right\rangle = \frac{1}{2^{(n+1)/2}} \sum_x (-1)^{f(x)} \left|x\right\rangle^n \Big(\left|0\right\rangle - \left|1\right\rangle \Big)$$

Now we apply the Hadamard again on all qubits:

$$\begin{aligned} H^{\otimes n+1}U_f \left|\psi_{in}\right\rangle &= \frac{1}{2^n}\sum_{x=0}^{2^n-1}(-1)^{f(x)}\Big(\sum_{y=0}^{2^n-1}(-1)^{x\cdot y}\left|y\right\rangle\Big)\Big(\left|1\right\rangle\Big) \\ &= \frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{y=0}^{2^n-1}(-1)^{f(x)+x\cdot y}\left|y\right\rangle\left|1\right\rangle \\ &= \frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{y=0}^{2^n-1}(-1)^{(a+y)\cdot x}\left|y\right\rangle\left|1\right\rangle \end{aligned}$$

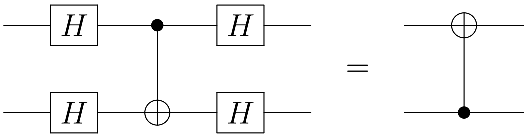
Focus on the sum over x:

$$\sum_{x=0}^{2^n-1}(-1)^{(a+y)\cdot x} = \prod_{j=0}^{n-1}\sum_{x_j=0}^{1}(-1)^{(a_j+y_j)x_j} = \prod_{j=0}^{n-1}\Big(1+(-1)^{a_j+y_j}\Big)$$

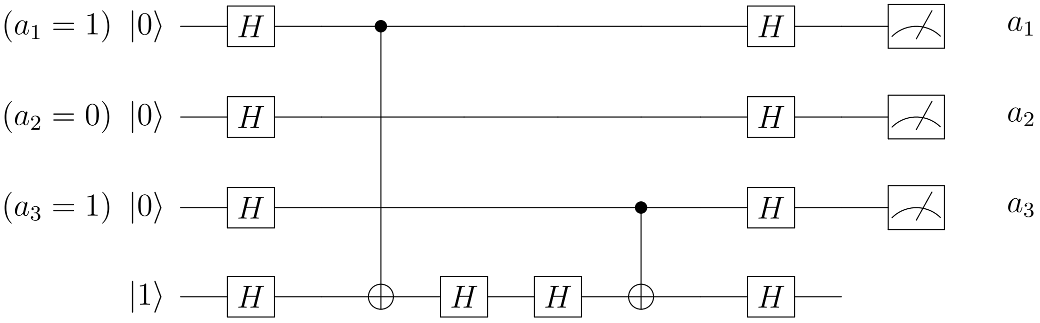
If $a_j \neq y_j$ at any point, the product vanishes. The only surviving output is $\left|a\right\rangle^n\left|1\right\rangle$. So we've identified a in 1 query to the oracle.

8.3.2 Gate Identities

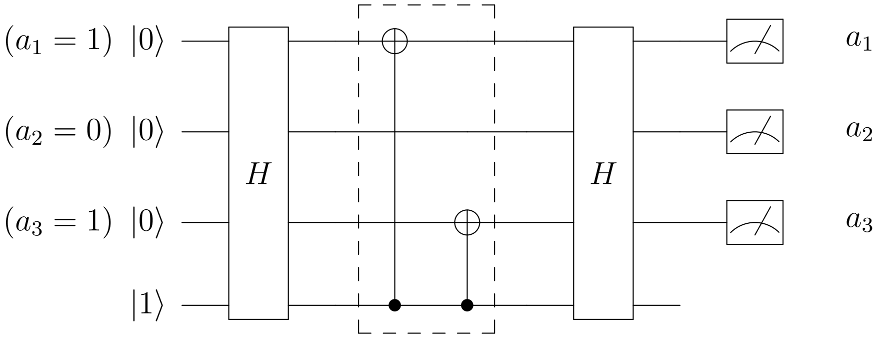
There's an easier way to see why this works. Recall the circuit identity:



Now imagine adding $H^2 = I$ in some parts of our original circuit. All Hadamards written out for clarity:



After grouping appropriately, we can see that the circuit is equivalent to



Since the bottom qubit is $|1\rangle$, the CNOT is applied on all qubits where the bit in a is 1. So obviously, a is reconstructed on the first 3 qubits.

8.4 Pure vs. Mixed States

Pure state: If the state of the system can be represented as a ket vector

- E.g. A beam of photons, all in $|+\rangle$, measured in HV basis

Mixed state: If the state of the quantum system is not precisely known, but has probabilities in a number of quantum states

- E.g. A machine emitting photons in either $|H\rangle$ or $|V\rangle$, measured in HV basis.
- The measurement is the same as with a pure state, but different fundamentally.

8.5 Density Matrices

A density matrix (represented with ρ) is useful to tell pure and mixed states apart.

For pure states, $\rho = |\psi\rangle\langle\psi|$

- E.g. $\rho_H = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$, $\rho_+ = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$

For mixed states, add density matrices.

- E.g. With the machine example, $\rho = \rho_H + \rho_V = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \neq \rho_+$

8.5.1 Properties

- Hermitian: $\rho = \rho^\dagger$
- Unit trace: $\mathrm{Tr}(\rho) = 1$

- If **pure state**:
  - Idempotent, $\rho^2 = \rho$
  - $\text{Tr}(\rho^2) = 1$
- If **mixed state**:
  - Not idempotent, $\rho^2 \neq \rho$
  - $\text{Tr}(\rho^2) \neq 1$

*Mixed states have density matrices that are diagonal with more than 1 entry, summing to 1.

*A "maximally mixed" state has no off-diagonal elements: $\rho_{ij} = 0, \forall i \neq j$

8.5.2 QM with Density Matrices

Some familiar QM concepts can be expressed in terms of density matrices:

- Born probability rule: $P(\text{outcome } \omega_i \text{ in state } \psi) = \text{Tr}(\Pi_i \rho)$, where $\Pi_i = |\omega_i\rangle \langle\omega_i|$ is the projector.
- Expectation value of an observable: $E(\Omega) = \langle\Omega\rangle_\psi = \text{Tr}(\Omega\rho)$
- Time evolution: $\rho' = U\rho U^\dagger$, where $|\psi'\rangle = U|\psi\rangle$.
  - Since $\rho' = |\psi'\rangle \langle\psi'| = U|\psi\rangle \langle\psi'| U^\dagger$

9 Week 9 (Oct 19 - Oct 25)

9.1 Partial Trace

Remember all the tracing we did with density matrices? Tracing over a system is practically impossible, but we can trace over a portion of it.

Consider a Φ_+ state shared by Alice and Bob. To examine Alice's marginal probabilities, we "sum over" Bob's system, since we're not concerned with it.

$$p(x_i y_j \text{ in } \Phi_+) = \Big| \langle x_i y_j | \Phi_+ \rangle \Big|^2 = \langle x_i| \langle y_j| \rho_{AB} |y_j\rangle |x_i\rangle$$

ρ_{AB} is the density matrix of the system. To "sum over" Bob's system,

$$p(x_i \text{ in } \rho_A) = \sum_{y=0}^{1} \langle x_i| \langle y| \rho_{AB} |y\rangle |x_i\rangle = \langle x_i| \rho_A |x_i\rangle$$

ρ_A is the "reduced density matrix" for A, and y is Bob's qubit.

$$\rho_A = \sum_{y=0}^{1} \langle y| \, \rho_{AB} \, |y\rangle$$

9.1.1 Example

With explicit numbers, the density matrix for Φ_+ is:

$$\Phi_+ = \rho_{AB} = \frac{1}{2} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} = \frac{1}{2} \Big[|00\rangle \langle 00| + |00\rangle \langle 11| + |11\rangle \langle 00| + |11\rangle \langle 11| \Big]$$

Now we find the reduced density matrix ρ_A:

$$\begin{aligned} \rho_A &= \sum_y \langle y| \, \rho_{AB} \, |y\rangle \\ &= \frac{1}{2} \langle 0| \Big[|00\rangle \langle 00| + |00\rangle \langle 11| + |11\rangle \langle 00| + |11\rangle \langle 11| \Big] |0\rangle + \\ &\quad \frac{1}{2} \langle 1| \Big[|00\rangle \langle 00| + |00\rangle \langle 11| + |11\rangle \langle 00| + |11\rangle \langle 11| \Big] |1\rangle \\ &= \frac{1}{2} \Big[|0\rangle \langle 0| + |1\rangle \langle 1| \Big] \end{aligned}$$

Note that this is a maximally mixed state — proportional to the identity matrix, with no cross terms. No quantum information (phases) remain. It's classically correlated now.

9.1.2 Implications & Uses

1. Tracing over half an entangled state yields a maximally mixed state. In a classically correlated system, if you know the whole system, you know its subparts. Here, you have knowledge of the quantum state (Φ_+) of the whole system, but when you consider a subsystem, you no longer have complete knowledge.
   - "The sum is not equal to the sum of the parts"
   - Another way quantum mechanics works differently than classical.
2. (Can skip) What if you throw one half of entangled pair into a black hole? How to represent state of qubit not thrown in — do a partial trace over black hole system. If what goes into a black hole is irretrievable, the initial quantum information is lost forever (per Hawking)
   - But that's a physical process where a pure state transitions into a mixed state, breaking quantum mechanics (this is only allowed for measurements, not unitary

processes).
- Perhaps somehow, quantum information is preserved on horizon of black hole…

9.2 Decoherence

The biggest enemy of quantum computation: Random interactions with the environment can change our quantum state.

We can't possibly measure our environment completely, so our only option is to partially trace over it.

But this eliminates a lot of coherence within the system, e.g. entanglement. We lose quantum information: the state "decoheres," acting like a mixed state, not a pure one.

We want a long decoherence time in order to squeeze out as many operations as possible before the system decoheres. Let τ_Q be the decoherence time and τ_{op} be the time for 1 operation. Then the max number of operations we can do is $n_{op} = \tau_Q/\tau_{op}$.

9.3 Simon's Algorithm

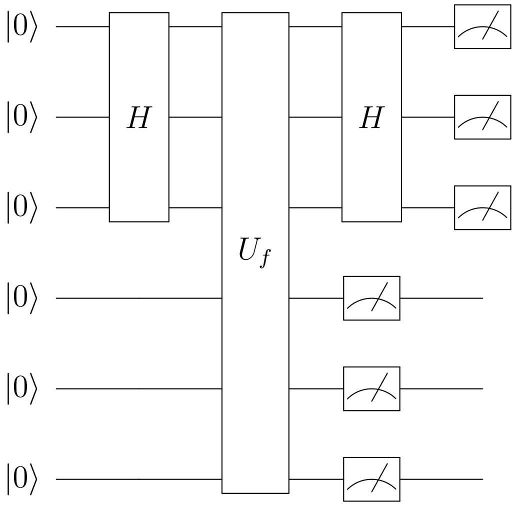
Consider $f(x)$ such that for 2 input bitstrings x and y, $f(x) = f(y)$ if $x \oplus y = a$, where a is some positive integer less than 2^n.

We see that $f(x) = f(x \oplus a)$, so a is the period of f.

Our goal is to determine a in a small number of oracle calls.

- Classically: at least $2^{n/2}$ calls to get a *favorable* probability of finding a.
- Quantum: n calls to oracle to determine a, plus n^2 classical steps.

Circuit diagram (for $n = 3$):



The top 3 qubits are the input, and the bottom 3 are ancillas. We Hadamard the inputs, send them through the oracle, and now the key step is the *measurement process*.

9.3.1 Example

After Hadamarding, our state is

$$|\psi\rangle_1 = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_n |0\rangle_n$$

Then we send through the oracle, which as always is $U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$.

$$|\psi\rangle_2 = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_n |f(x)\rangle_n$$

Now we measure the bottom register, which collapses the state. There are two inputs (x and $x \oplus a$) for every output $f(x)$, so the result of the measurement is a superposition of those states.

$$|\psi\rangle_3 = \frac{1}{\sqrt{2}}\Big(|x\rangle_n + |x \oplus a\rangle_n \Big)$$

Now we do the Hadamard on the top register ($|\psi_5\rangle$). Phase kickback is used for the Hadamard, as usual.

$$\begin{aligned} |\psi\rangle_4 &= \frac{1}{\sqrt{2}} \cdot \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} \Big[(-1)^{x\cdot y} + (-1)^{(x \,\oplus\, a)\cdot y}\Big] |y\rangle_n \\ &= \frac{1}{2^{\frac{n+1}{2}}} \sum_{y=0}^{2^n-1} (-1)^{x\cdot y}\Big[1 + (-1)^{a\cdot y}\Big] |y\rangle_n \end{aligned}$$

Now if we measure, only the terms such that $a \cdot y = 0$ survive: $\sum_{i=0}^{n-1} a_i y_i = 0 \mod 2$. So by measuring the input, we learn some random value y such that $a \cdot y = 0$.

By repeating this process n times, discarding repeat inputs and the $|0\rangle_n$ input case, we can learn enough about a to deduce it with Gaussian elimination. That can be done classically in n^2 steps.

10 Week 10 (Oct 26 - Nov 1)

10.1 Fourier Analysis

10.1.1 Vector Spaces

Previously, we looked at orthonormal vector spaces:

$$\langle i|j\rangle = 0 \quad (\text{if } i \neq j)$$
$$\langle i|i\rangle = 1$$
$$\langle i|j\rangle = \delta_{ij} \quad (\text{Kronecker delta})$$

Any vector in the space can be expressed as a linear combination of those basis vectors:

$$|V\rangle = a_1 |1\rangle + a_2 |2\rangle + ... + a_n |n\rangle$$

To find a specific component, we perform the inner product (all irrelevant terms become 0):

$$\langle 27|V\rangle = \langle 27| a_{27} |27\rangle = a_{27}$$

10.1.2 Function Spaces

Now we consider an orthonormal basis of functions, not vectors.

We define the basis functions for the circle S^1 as:

$$f_n(\theta) = \frac{1}{\sqrt{2\pi}} e^{in\theta} \quad n = 0, \pm 1, \pm 2, ...$$

The inner product is defined as:

$$\langle f|g\rangle = \int_0^{2\pi} f^* g \; d\theta$$

It can be seen that this basis is orthonormal.

- The basis is normalized: $\langle f_n|f_n\rangle = \frac{1}{2\pi}\int_0^{2\pi} e^{-in\theta} e^{in\theta} = 1$
- The basis is orthogonal: $\langle f_n|f_m\rangle = \frac{1}{2\pi}\int_0^{2\pi} e^{i(m-n)\theta} = 0$
- The basis is complete. We can use an infinite sum to represent all members in the function space: $|f\rangle = \frac{1}{\sqrt{2\pi}}\sum_{j=-\infty}^{+\infty} a_j e^{ij\theta}$

To find the components, we do the same inner product business:

$$a_{27} = \langle 27|f\rangle = \frac{1}{\sqrt{2\pi}} \int_0^{2\pi} e^{-27i\theta} f(\theta)\ d\theta$$

Note: Be careful about sums vs. integrals.

- The inner product is defined as an **integral**.
- The element of the function space is a **sum** of the basis functions, but when performing the inner product, most of the irrelevant terms become 0.

10.1.3 Fourier Analysis

Now consider a space of functions over the periodic line $-\pi \le x \le +\pi$. Disregard the imaginary numbers, and let

$$|n\rangle = \cos(nx) + \sin(nx)$$
$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \Big[a_n \cos(nx) + b_n \sin(nx)\Big]$$

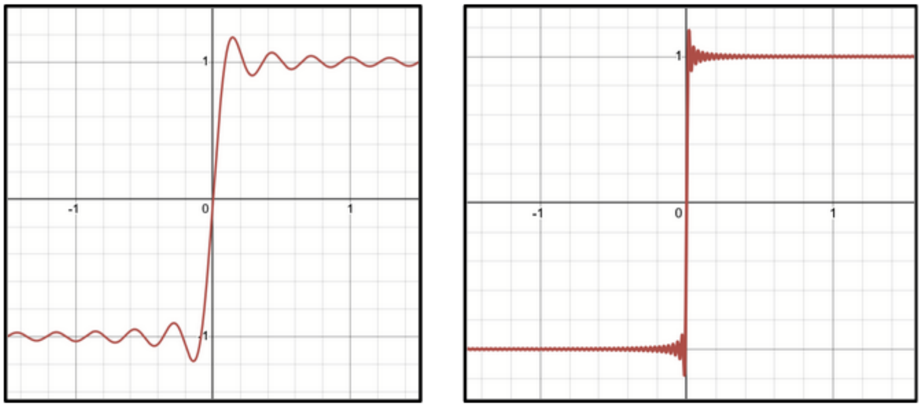
Expansion coefficients are found with the inner product, as before:

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(nx)\ dx$$
$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(nx)\ dx$$

As a solid example, consider the piecewise function $f(x) = -1$ for $-\pi < x < 0$, and $f(x) = +1$ for $0 < x < \pi$. By finding the expansion coefficients, we get

$$a_n = \frac{1}{\pi}\int_{-\pi}^{0} (-1)\cos(nx)\ dx + \frac{1}{\pi}\int_0^{\pi} \cos(nx)\ dx = 0$$
$$b_n = \frac{1}{\pi}\int_{-\pi}^{0} (-1)\sin(nx)\ dx + \frac{1}{\pi}\int_0^{\pi} \sin(nx)\ dx = \frac{2}{n\pi}\left(1 - \cos(n\pi)\right)$$

We get that $f(x) = \frac{\pi}{4}\left(\sin(x) + \frac{\sin(3x)}{3} + \frac{\sin(5x)}{5} + ...\right)$, which models the piecewise function quite well.



Left: 10 terms. Right: 100 terms.

10.2 Fourier Transform

10.2.1 Introduction

Great, so we have our function to model the general element in the function space:

$$f_n(\theta) = \frac{1}{\sqrt{2\pi}} \sum_{n=-\infty}^{+\infty} a_n e^{in\theta}$$

Now map the function onto a line of length L. We transform $x = \frac{\theta L}{2\pi}$. The coefficients are found with the inner product, as usual.

$$f(x) = \frac{1}{\sqrt{L}} \sum_{n=0}^{\infty} a_n e^{2\pi i \frac{nx}{L}}$$
$$a_n = \langle n|f\rangle = \frac{1}{\sqrt{L}} \int_0^L f(x) e^{-2\pi i \frac{nx}{L}}$$

10.2.2 Discrete Fourier Transform

Okay, now we discretize $x \to k$ and label $L \to N$. All the better to move to Quantum Fourier Transform (QFT), since things are discrete there.

Remember, F_k is the function, f_n are the coefficients:

$$F_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f_n e^{2\pi i \frac{kn}{N}}$$
$$f_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} F_k e^{-2\pi i \frac{kn}{N}}$$

Define $\omega_N = e^{i\frac{2\pi}{N}}$ as the N-th root of unity. Then we have

$$F_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f_n \omega_N^{n\cdot k}, \quad n\cdot k \text{ is mod } N$$

We can express this as a matrix operating on a vector:

$$|F\rangle = \frac{1}{\sqrt{N}} \begin{pmatrix} \omega^{0\cdot 0} & \omega^{0\cdot 1} & \omega^{0\cdot 2} & \dots & \omega^{0\cdot(N-1)} \\ \omega^{1\cdot 0} & \omega^{1\cdot 1} & \omega^{1\cdot 2} & \dots & \omega^{1\cdot(N-1)} \\ \omega^{2\cdot 0} & \omega^{2\cdot 1} & \omega^{2\cdot 2} & \dots & \omega^{2\cdot(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ \omega^{(N-1)\cdot 0} & \omega^{(N-1)\cdot 1} & \omega^{(N-1)\cdot 2} & \dots & \omega^{(N-1)\cdot(N-1)} \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{n-1} \end{pmatrix}$$

Since the dot product is taken mod N, it can be simplified to

$$\begin{aligned} |F\rangle &= \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{N-2} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{N-1} & \omega^{N-2} & \dots & \omega \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{n-1} \end{pmatrix} \\ &\equiv U_{FT} |f\rangle \end{aligned}$$

We define the previous matrix U_{FT} as the Quantum Fourier Transform (QFT). It'll be useful later on. The gate can be written compactly as

$$U_{FT} = \frac{1}{2^{n/2}} \sum_{j=0}^{2^n-1} \sum_{k=0}^{2^n-1} e^{2\pi i \frac{kj}{2^n}} |k\rangle \langle j|$$

Note that for the $N = 2$ case, $\omega_2 = e^{i\frac{2\pi}{2}} = -1$. So

$$U_{FT} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & \omega\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$$

Which is the Hadamard.

11 Week 11 (Nov 2 - Nov 8)

11.1 Fourier Transform Basis

Recall that the Hadamard register creates all bitstrings of length n:

$$\frac{1}{2^{n/2}}\Big[\sum_{i_0=0}^{1}|i_0\rangle \otimes \sum_{i_1=0}^{1}|i_1\rangle \otimes ... \otimes \sum_{i_{n-1}=0}^{1}|i_{n-1}\rangle\Big]$$

This is in terms of bits. In the **computational number basis**, we express it in terms of natural numbers:

$$\frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}|x\rangle = \frac{1}{2^{n/2}}\Big(|0\rangle + |1\rangle + |2\rangle + ... + |2^n - 1\rangle\Big)$$

The Fourier Transform basis is similar, using natural numbers inside kets. It's marked with a ~ and defined as

$$\begin{aligned}|\tilde{j}\rangle &= \frac{1}{2^{n/2}}\sum_{k=0}^{2^n-1}e^{2\pi i\frac{j\cdot k}{2^n}}|k\rangle\\ &= \frac{1}{\sqrt{N}}\sum_{k=0}^{2^n-1}\omega^{j\cdot k}|k\rangle\end{aligned}$$

where $N = 2^n$ is the number of terms, and $\omega = e^{i\frac{2\pi}{N}}$.

As a concrete example, $|\tilde{4}\rangle$ is defined as

$$|\tilde{4}\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{2^n-1}\omega^{4k}|k\rangle$$

Note that this is the 4th column of the QFT gate U_{FT} (assuming we count from 0). In fact, U_{FT} can be defined as

$$U_{FT} = \Big(|\tilde{0}\rangle \quad |\tilde{1}\rangle \quad |\tilde{2}\rangle \quad ... \quad |N \tilde{-} 1\rangle\Big)$$

11.2 More with the QFT Gate

From the previous representation of U_{FT}, now imagine that operating on a computational basis vector $|x\rangle$. For simplicity, let there be $n = 3$ bits, and $N = 2^3 = 8$ terms. The quantum fourier transform of $|x\rangle$ becomes

$$\begin{aligned} U_{FT}|x\rangle &= \frac{1}{\sqrt{8}} \sum_{k=0}^{7} e^{i\frac{2\pi}{8}x\cdot k} |k\rangle \\ &= \frac{1}{\sqrt{8}} \Big[|0\rangle + e^{i\frac{\pi}{4}x} |1\rangle + e^{i\frac{2\pi}{4}x} |2\rangle + e^{i\frac{3\pi}{4}x} |3\rangle + ... + e^{i\frac{7\pi}{4}x} |7\rangle \Big] \end{aligned}$$

Now express this in the computational basis (binary):

$$\frac{1}{\sqrt{8}} \Big[|000\rangle + e^{i\frac{\pi}{4}x} |001\rangle + e^{i\frac{2\pi}{4}x} |010\rangle + e^{i\frac{3\pi}{4}x} |011\rangle + e^{i\frac{4\pi}{4}x} |100\rangle + e^{i\frac{5\pi}{4}x} |101\rangle + e^{i\frac{6\pi}{4}x} |110\rangle + e^{i\frac{7\pi}{4}x} |111\rangle \Big]$$

By grouping terms and focusing on those with one $|1\rangle$ in it, we can factor this as

$$\frac{1}{\sqrt{8}} \Big(|0\rangle + e^{i\pi x} |1\rangle \Big) \otimes \Big(|0\rangle + e^{i\frac{\pi}{2}x} |1\rangle \Big) \otimes \Big(|0\rangle + e^{i\frac{\pi}{4}x} |1\rangle \Big)$$

In fact, the general QFT for n qubits can be factored like this, with $|x\rangle$ in the computational number basis:

$$\begin{aligned} U_{FT}|x\rangle &= \frac{1}{\sqrt{N}} \sum_{k=0}^{2^n-1} e^{2\pi i\frac{x\cdot k}{N}} |k\rangle \\ &= \frac{1}{\sqrt{N}} \Big(|0\rangle + e^{2\pi i\frac{x}{2}} |1\rangle \Big) \otimes \Big(|0\rangle + e^{2\pi i\frac{x}{4}} |1\rangle \Big) \otimes ... \otimes \Big(|0\rangle + e^{2\pi i\frac{x}{2^n}} |1\rangle \Big) \end{aligned}$$

In the computational basis (with $x = 2^{n-1}x_{n-1} + 2^{n-2}x_{n-2} + ... + 2x_1 + x_0$), we just plug that in and simplify with $e^{2\pi i} = 1$.

$$\begin{aligned} U_{FT}|x\rangle &= \frac{1}{\sqrt{N}} \sum_{k=0}^{2^n-1} e^{2\pi i\frac{x\cdot k}{N}} |k\rangle \\ &= \frac{1}{\sqrt{N}} \Big(|0\rangle + e^{i\pi x_0} |1\rangle \Big) \otimes \Big(|0\rangle + e^{i\pi x_1} e^{i\pi\frac{x_0}{2}} |1\rangle \Big) \otimes \Big(|0\rangle + e^{i\pi x_2} e^{i\pi\frac{x_1}{2}} e^{i\pi\frac{x_0}{4}} |1\rangle \Big) \otimes ... \end{aligned}$$

12 Week 12 (Nov 9 - Nov 15)

12.1 QFT Circuit

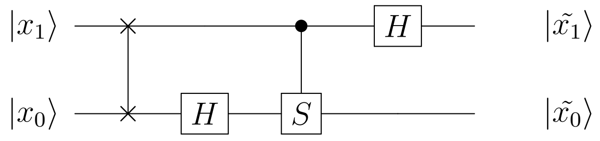
Recall the gates:

- $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$, phase shift by π
- $S = \sqrt{Z} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{pmatrix}$: phase shift by $\frac{\pi}{2}$
- $T = \sqrt{S} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$: phase shift by $\frac{\pi}{4}$

Now back to the U_{FT} gate. For $n = 2$ bits, we have

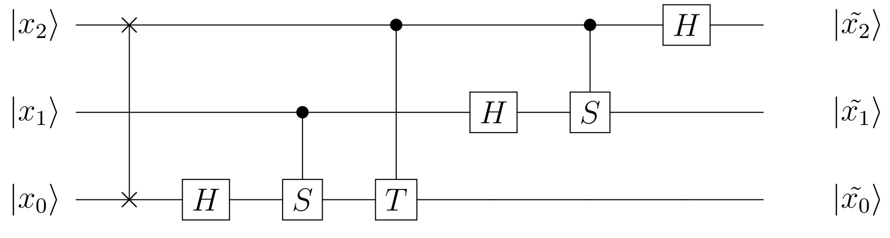
$$U_{FT}\left|x\right\rangle = \frac{1}{2}\underbrace{\Big(\left|0\right\rangle + e^{i\pi x_0}\left|1\right\rangle\Big)}_{\left|\tilde{x_1}\right\rangle} \otimes \underbrace{\Big(\left|0\right\rangle + e^{i\pi x_1}e^{i\pi\frac{x_0}{2}}\left|1\right\rangle\Big)}_{\left|\tilde{x_0}\right\rangle}$$

Note that the first parentheses is $\left|\tilde{x_1}\right\rangle$, but it depends on x_0. The second parentheses is $\left|\tilde{x_0}\right\rangle$, but also depends on x_1. To make the circuit easier, we can reverse the order of the input bits.


$\left|x_1\right\rangle$
$\left|x_0\right\rangle$
H
H
S
$\left|\tilde{x_1}\right\rangle$
$\left|\tilde{x_0}\right\rangle$

For $n = 3$, the circuit is very similar.

$$U_{FT}\left|x\right\rangle = \frac{1}{2}\underbrace{\Big(\left|0\right\rangle + e^{i\pi x_0}\left|1\right\rangle\Big)}_{\left|\tilde{x_2}\right\rangle} \otimes \underbrace{\Big(\left|0\right\rangle + e^{i\pi x_1}e^{i\pi\frac{x_0}{2}}\left|1\right\rangle\Big)}_{\left|\tilde{x_1}\right\rangle} \otimes \underbrace{\Big(\left|0\right\rangle + e^{i\pi x_2}e^{i\pi\frac{x_1}{2}}e^{i\pi\frac{x_2}{4}}\left|1\right\rangle\Big)}_{\left|\tilde{x_0}\right\rangle}$$


$\left|x_2\right\rangle$
$\left|x_1\right\rangle$
$\left|x_0\right\rangle$
H
S
T
H
S
H
$\left|\tilde{x_2}\right\rangle$
$\left|\tilde{x_1}\right\rangle$
$\left|\tilde{x_0}\right\rangle$

The pattern continues for higher n: swap the order, apply finer and finer "rotations" as necessary with the appropriate control bits, etc. However, we'll need smaller rotation gates, such as $\sqrt{T}$ and more.

12.2 QFT with Decimals

Just like numbers can be represented in binary, decimals can be represented in binary. E.g. for $j = 0.j_1 j_2 j_3...$ and $j \in [0, 1]$,

$$j = \frac{j_1}{2^1} + \frac{j_2}{2^2} + ... + \frac{j_n}{2_n}$$

Each j_i is 0 or 1.

12.3 Quantum Phase Estimation

12.3.1 Analysis

The problem: given a unitary matrix, we want to estimate its eigenvalue.

$$U \left| u \right\rangle = \lambda \left| u \right\rangle = e^{2\pi i \phi} \left| u \right\rangle$$

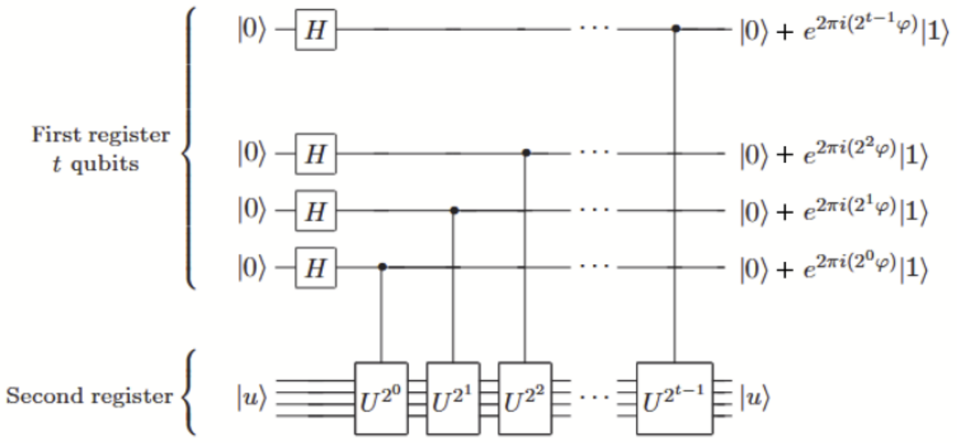
U is a unitary matrix, so its eigenvalue is a phase angle, represented by $\lambda = e^{2\pi i \phi}$, where $\phi \in [0, 1]$. By finding ϕ, we can find λ.

What if we apply U 2^j times? Then

$$U^{2^j} \left| u \right\rangle = e^{2\pi i (2^j \phi)} \left| u \right\rangle$$

Here it's starting to look a little like the QFT, especially the $e^{2\pi i}$ factor. Maybe if we can find a circuit to build this state, using some ancillas, then apply the inverse QFT, we should be able to get ϕ.

12.3.2 Step 1


First register
t qubits
Second register
$|0\rangle$
H
$|u\rangle$
U^{2^0}
U^{2^1}
U^{2^2}
$U^{2^{t-1}}$
$|u\rangle$
$|0\rangle + e^{2\pi i(2^{t-1}\varphi)}|1\rangle$
$|0\rangle + e^{2\pi i(2^{2}\varphi)}|1\rangle$
$|0\rangle + e^{2\pi i(2^{1}\varphi)}|1\rangle$
$|0\rangle + e^{2\pi i(2^{0}\varphi)}|1\rangle$

Quantum Phase Estimation circuit, from Nielsen & Chuang p. 222. Normalization factors disregarded.

This is the circuit that will get us there. Using the same gate identity trick as in Bernstein-Vazirani (seek Week 8), we insert H^2 as necessary to flip the control qubits. Then the final state of the first register is

$$|\psi_{top}\rangle = \frac{1}{2^{t/2}}\Big(\,|0\rangle + e^{2\pi i(2^{t-1}\phi)}\,|1\rangle\,\Big)\Big(\,|0\rangle + e^{2\pi i(2^{t-2}\phi)}\,|1\rangle\,\Big)...\Big(\,|0\rangle + e^{2\pi i(2^{0}\phi)}\,|1\rangle\,\Big)$$

Since ϕ is a decimal, we can write $\phi = 0.\phi_1\phi_2\phi_3...\phi_t$. Note that $2^k\phi = \phi_1\phi_2...\phi_k.\phi_{k+1}...\phi_t$. In an exponent,

$$e^{2\pi i(2^k\phi)} = e^{2\pi i(\phi_1\phi_2...\phi_k + 0.\phi_{k+1}...\phi_t)} = e^{2\pi i(0.\phi_{k+1}...\phi_t)}$$

Going back to the state of the first register, we can rewrite as

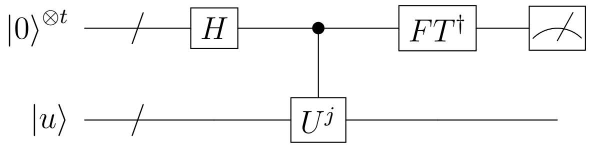
$$\begin{aligned}
|\psi_{top}\rangle &= \frac{1}{2^{t/2}}\Big(\,|0\rangle + e^{2\pi i(0.\phi_t)}\,|1\rangle\,\Big)\Big(\,|0\rangle + e^{2\pi i(0.\phi_{t-1}\phi_t)}\,|1\rangle\,\Big)...\Big(\,|0\rangle + e^{2\pi i(0.\phi_1\phi_2...\phi_t)}\,|1\rangle\,\Big)\\
&= \frac{1}{2^{t/2}}\Big(\,|0\rangle + e^{2\pi i(\frac{\phi_t}{2})}\,|1\rangle\,\Big)\Big(\,|0\rangle + e^{2\pi i(\frac{\phi_{t-1}}{2}+\frac{\phi_t}{4})}\,|1\rangle\,\Big)...\Big(\,|0\rangle + e^{2\pi i(\frac{\phi_1}{2}+\frac{\phi_2}{4}+...+\frac{\phi_t}{2^t})}\,|1\rangle\,\Big)\\
&= \frac{1}{2^{t/2}}\sum_{k=0}^{2^n-1} e^{2\pi i\overset{\leftrightarrow}{\phi}k}\,|k\rangle = FT\,|\overset{\leftrightarrow}{\phi}\rangle
\end{aligned}$$

Where $\overset{\leftrightarrow}{\phi} = \phi_t\phi_{t-1}...\phi_2\phi_1$, an integer. Note that this is “reversed” from the usual order of ϕ. For more details on the QFT product-to-sum, see Week 11.

12.3.3 Step 2

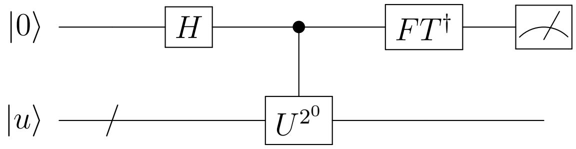
We have our QFT-esque result on the first register. Now we apply the inverse QFT to it in order to get $\overset{\leftrightarrow}{\phi}$. By measuring the wires in reverse order, we can get ϕ, then finally $\lambda = e^{2\pi i\phi}$, the eigenvalue of U.

Our complete circuit looks like



12.3.4 Examples

For $t = 1$ bit,

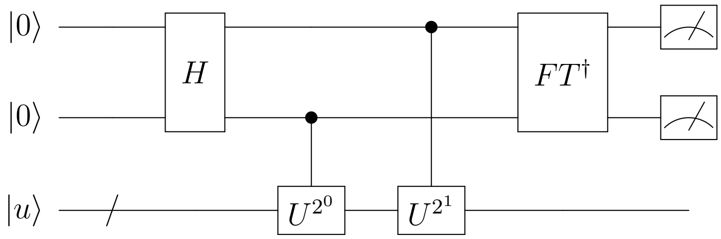


We have $\phi = 0.\phi_1$, and ϕ_1 is 0 or 1. The top register becomes

$$\begin{aligned}
|\psi_{top}\rangle &= \frac{1}{\sqrt{2}}\sum_{k=0}^{1} e^{2\pi i\phi k}|k\rangle = \frac{1}{\sqrt{2}}\sum_{k=0}^{1} e^{2\pi i(\frac{\phi_1}{2})k}|k\rangle \\
&= \frac{1}{\sqrt{2}}\sum_{k=0}^{1} e^{\pi i\phi_1 k}|k\rangle = \frac{1}{\sqrt{2}}(-1)^{\phi_1 k}|k\rangle \\
&= \frac{1}{\sqrt{2}}\Big(|0\rangle + (-1)^{\phi_1}|1\rangle\Big)
\end{aligned}$$

For $t = 1$, the inverse QFT is just the Hadamard. So by applying $FT^\dagger$, we get $|\phi_1\rangle$. This can be either $|0\rangle$ or $|1\rangle$. Since $\phi = 0.\phi_1$, only $\phi = 0$ or $\phi = \frac{1}{2}$ will work precisely. Others will be a bad estimate.

For $t = 2$ bits, we go through the same thing again.



Here $\phi = 0.\phi_1\phi_2$. Right before the inverse QFT, the top register is

$$\begin{aligned}|\psi_{top}\rangle &= \frac{1}{2}\sum_{k=0}^{3} e^{2\pi i\phi k}\,|k\rangle \\ &= \frac{1}{2}\Big(\,|00\rangle + e^{2\pi i\phi}\,|01\rangle + e^{4\pi i\phi}\,|10\rangle + e^{6\pi i\phi}\,|11\rangle\,\Big)\end{aligned}$$

Applying the inverse QFT gives

$$FT^{\dagger}\Big(\frac{1}{2}\sum_{k=0}^{3} e^{2\pi i\phi k}\,|k\rangle\Big) = |\overset{\leftrightarrow}{\phi}\rangle$$

Only $\phi = 0.00_2, 0.01_2, 0.10_2, 0.11_2 = 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}$ will give precise results.

The more qubits we have, the more accurately we can estimate ϕ.

13 Week 13 (Nov 16 - Nov 22)

13.1 Grover's Algorithm

The problem: Given an unstructured list (length n), find a specific entry "a" with the fewest calls to the oracle.

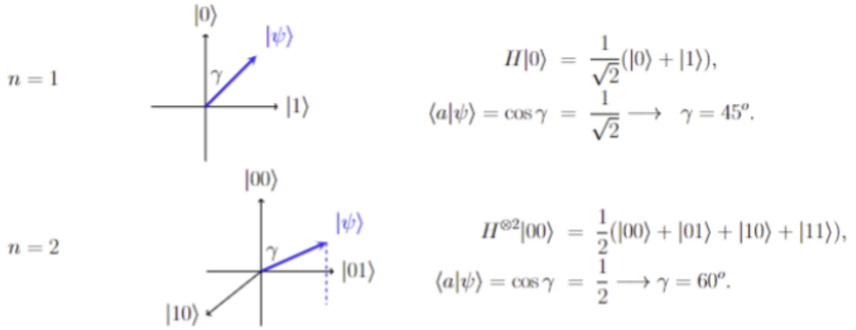
a is an n-bit string in the 2^n-dimensional Hilbert space of n qubits (i.e. the space of all n-bit strings). We define the oracle as usual:

$$U_f\,|x\rangle\,|y\rangle = |x\rangle\,|y \oplus f(x)\rangle\,, \quad f(x) = 1 \text{ if } x = a\text{, else } 0$$

13.2 Procedure

1. We Hadamard all n bits.
2. In the Hilbert space, our goal is to rotate $|\psi\rangle = H^{\otimes n}\,|0\rangle_n$ to $|a\rangle$.
   - The "Grover iteration" $G = WV$
   - V: First, the oracle call flips the current state about $|a_{\perp}\rangle$
   - W: Next, we flip the current state about $|\psi\rangle$
3. After repeating G the optimal amount of times, run the state through the oracle U_f. If the result is 0, start over. If the result is 1, we found a.

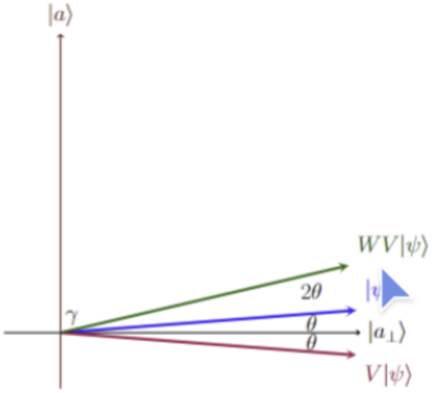
13.2.1 Geometry


$n = 1$
$|0\rangle$
$|\psi\rangle$
γ
$|1\rangle$
$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle),$
$\langle a|\psi\rangle = \cos\gamma = \frac{1}{\sqrt{2}} \longrightarrow \gamma = 45^o.$
$n = 2$
$|00\rangle$
$|\psi\rangle$
γ
$|01\rangle$
$|10\rangle$
$H^{\otimes 2}|00\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle),$
$\langle a|\psi\rangle = \cos\gamma = \frac{1}{2} \longrightarrow \gamma = 60^o.$

For n qubits, $|\psi\rangle = H^{\otimes n}|0\rangle_n$

In the n-th case, the angle γ between $|a\rangle$ and $|\psi\rangle$ is

$$\langle a|\psi\rangle = \cos\gamma = \frac{1}{2^{n/2}}$$
$$\gamma \approx \frac{\pi}{2}$$



Each iteration of Grover's algorithm

In each iteration, we rotate about $|a_\perp\rangle$, and then about $|\psi\rangle$. If the original angle between $|\psi\rangle$ and $|a_\perp\rangle$ is θ, each iteration rotates $|\psi\rangle$ closer to $|a\rangle$ by 2θ.

13.2.2 Time Complexity

Each iteration, we rotate towards $|a\rangle$ by 2θ. The angle θ between $|\psi\rangle$ and $|a_\perp\rangle$ is given by

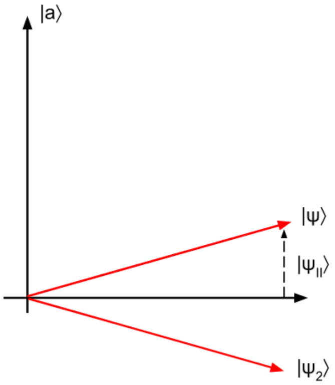
$$\sin\theta = \cos\gamma = \frac{1}{2^{n/2}}$$
$$\theta \approx \frac{1}{2^{n/2}} = \sqrt{\frac{1}{N}}$$

We need at least $\frac{\gamma}{2\theta}$ iterations to rotate $|\psi\rangle$ to $|a\rangle$. Note that $\gamma = \frac{\pi}{2} - \theta$.

$$\frac{\gamma}{2\theta} = \frac{\frac{\pi}{2} - \theta}{2\theta} = \frac{\pi}{4\theta} - 1$$
$$\frac{\gamma}{2\theta} \approx \frac{\pi}{4\theta} = \frac{\pi}{4}\sqrt{N}$$

Grover's algorithm runs in $O(\sqrt{N})$ time, where $N = 2^n$ and n is the number of bits. It offers a quadratic speedup over the classical brute force, and can be applied in NP problems.

13.2.3 Constructing V



V iteration of Grover's algorithm

In the diagram, $|\psi_\parallel\rangle = |a\rangle\langle a|\psi\rangle$. We want to flip this component, so $|\psi_2\rangle = |\psi\rangle - 2|\psi_\parallel\rangle$. Thus the gate V is

$$V = I - 2|a\rangle\langle a|$$

If we apply this to $|\psi\rangle$, we get $|\psi_2\rangle = V|\psi\rangle = |\psi\rangle - 2|a\rangle\langle a|\psi\rangle$ as desired. Note that since $|\psi\rangle$ is the result of Hadamarding, $V|\psi\rangle$ is just the same as reversing the amplitude of the $|a\rangle$ term. Expressed this way,

$$V|\psi\rangle = \frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle$$

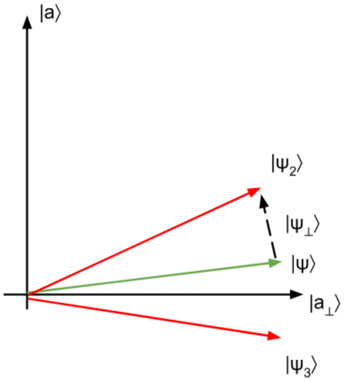
Where it's the superposition of all n-bit strings (result of Hadamard), but with the sign of $|a\rangle$ reversed. Remember, $f(x) = 1$ only when $x = a$.

So we could think of V as just the oracle $U_f = V \otimes I$:

$$U_f |x\rangle |y\rangle = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle |y\rangle$$

We don't have to construct any new gates. Just use the given oracle U_f.

13.2.4 Constructing W



W iteration of Grover's algorithm

Now we want to rotate $|\psi_2\rangle$ about $|\psi\rangle$ (our original state) to get $|\psi_3\rangle$. To do that, we flip the $|\psi_\perp\rangle$ component of $|\psi_2\rangle$.

$$\begin{aligned} |\psi_3\rangle = W|\psi_2\rangle &= |\psi\rangle \langle\psi|\psi_2\rangle - |\psi_\perp\rangle \langle\psi_\perp|\psi_2\rangle \\ &= \Big(2|\psi\rangle\langle\psi| - I\Big) |\psi_2\rangle \end{aligned}$$

In the second step, we used the resolution of identity: $I = |\psi\rangle\langle\psi| + |\psi_\perp\rangle\langle\psi_\perp|$. So our W gate is found to be

$$W = 2|\psi\rangle\langle\psi| - I$$

13.2.5 Example: 2 Bits

So now we have our Grover's iteration: $G = WV$. Consider searching for a string $|a\rangle = |10\rangle$, in the space of all 2-bit strings.

- First, we Hadamard the zero bits: $|\psi\rangle = \frac{1}{2}\Big(|00\rangle + |01\rangle + |10\rangle + |11\rangle\Big)$
- Next, we apply $G = WV$, i.e. $G|\psi\rangle = WV|\psi\rangle$

- We apply V first, which flips the amplitude of $|a\rangle$: $|\psi_2\rangle = V|\psi\rangle = \frac{1}{2}\Big(|00\rangle + |01\rangle - |10\rangle + |11\rangle\Big)$
- Then we apply W, which gives: $W|\psi_2\rangle = 2|\psi\rangle\langle\psi|\psi_2\rangle - |\psi_2\rangle = |a\rangle$

Thus we've found the desired string in 1 call to the oracle ($U_f = V \otimes I$). Classically, we'd take on average 2 calls.

13.3 Example: 3SAT

13.3.1 V and W in Circuits

V is easy enough, we just use the standard oracle for 3SAT (see Week 8 for more info).

W is more tricky. If we start with $W = 2|\psi\rangle\langle\psi| - 1$, and $|\psi\rangle = H^{\otimes n}|0\rangle_n$,

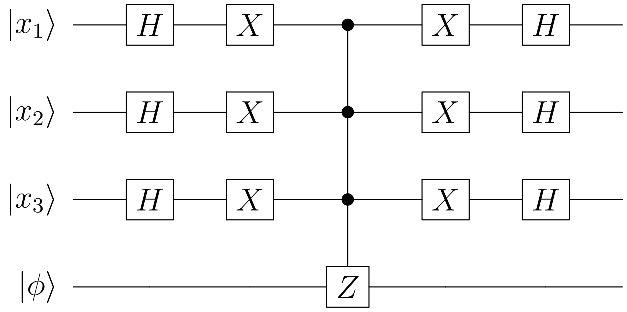
$$\begin{aligned} W &= 2\Big(H^{\otimes n}|0\rangle_n\langle 0|_n H^{\otimes n}\Big) - 1 \\ &= H^{\otimes n}\Big(2|0\rangle_n\langle 0|_n H^{\otimes n}\Big) - 1 \\ &= H^{\otimes n}\Big(2|0\rangle_n\langle 0|_n - I\Big)H^{\otimes n} \end{aligned}$$

In the last step we used $H^2 = I$ to bring out the Hadamard inside the parentheses. Anyways, the middle gate $\Big(2|0\rangle_n\langle 0|_n - I\Big)$ is going to flip the phase of everything except $|0\rangle_n$.

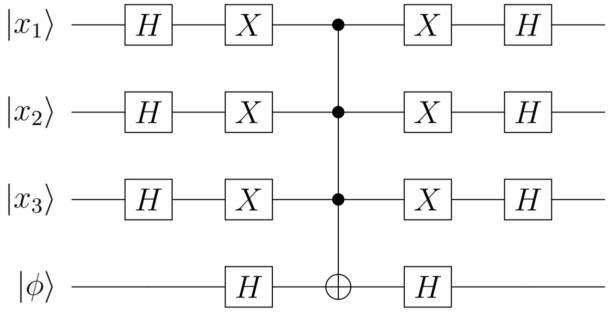
We can factor out a global factor of (-1) to use $\Big(I - 2|0\rangle_n\langle 0|_n\Big)$ instead, which will only flip the phase of $|0_n\rangle$. It's like the controlled-Z, exept on $|0\rangle_n$ instead of $|1\rangle_n$. So we first flip all bits, apply the controlled-Z, and then flip back:

$$\begin{gathered} 2|0\rangle_n\langle 0|_n - I = XC_ZX \\ W = H^{\otimes n}\Big(2|0\rangle_n\langle 0|_n - I\Big)H^{\otimes n} = HXC_ZXH \end{gathered}$$

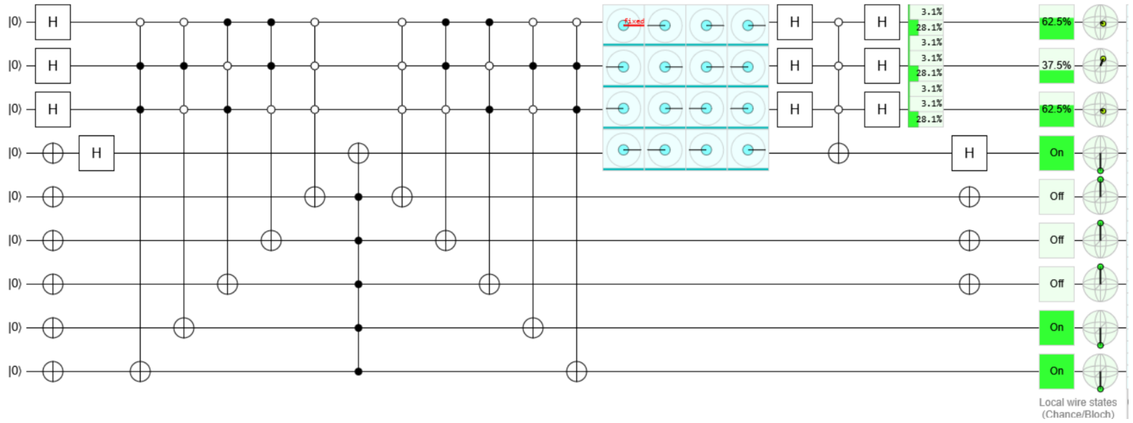
Here's a circuit example with $n = 3$:



Now we insert $H^2 = I$ surrounding the 3 controls, and use the identity $Z = HXH$. Then we get the followign circuit:



13.3.2 Screenshot



Example of Grover's algorithm with a certain 3SAT problem

Note the first half of the circuit, which does the necessary oracle for 3SAT (and uncomputes the registers). The second half of the circuit is our W gate, with the white circles meaning the control is applied when the bit is 0, not 1.

This is one iteration of $G = WV$. The probabilities have spiked to 28.1% in certain bits. If we run a few more times, those should spike more, and that will tell us the solution to this 3SAT.

Link: https://tinyurl.com/y5c3twlz

14 Week 14 (Nov 23 - Nov 29)

14.1 Basics of Cryptography

Setup: Alice and Bob want to send messages to each other. Eve is eavesdropping. We want to find a way for Alice and Bob to thwart Eve.

14.1.1 The One-Time Pad

A “gold standard” in cryptography. If done right, it’s theoretically uncrackable.

Take the example of Soviet Russia and the US. Anastasia (A) and Boris (B) want to send messages without Eisenhower (E) eavesdropping. This was achieved by distributing a “pad” of keys to agents beforehand. Every time Anastasia and Boris communicate, they would use the key at the top, then discard it.

At one point the US managed to successfully crack the encoded messages, not because the one-time pad is insecure, but because agents started re-using pads.

14.1.2 Key Distribution

The goal of cryptography is to distribute those keys to Alice and Bob (or Anastasia and Boris) securely. It’s cumbersome to do this through private channels, e.g. in physical meetings. We want to do it through public channels, e.g. the internet.

But anyone can theoretically eavesdrop over public channels. Can we securely distribute keys nevertheless?

14.2 Diffie-Hellman Key Exchange

Diffie-Hellman is a classical key distribution process. It leverages the asymmetrical difficulty in the discrete log problem:

$$\begin{gathered} g, p = \text{known} \\ A = g^a \mod p \\ \text{Easy to find } A \text{ given } a\text{, but hard to find } a \text{ given } A. \end{gathered}$$

Here’s how it works:

Stage 1: Setup

1. A large prime p and a primitive root g is chosen such that $2 \leq g \leq p - 2$. These are announced publicly.
    - “Primitive root” g: for every $A < p$, we can find some a such that $A = g^a \mod p$.
    - That is, $g^a \mod p$ can eventually generate every integer less than p.
    - We call g a generator of p.

Stage 2: Alice & Bob generate strings

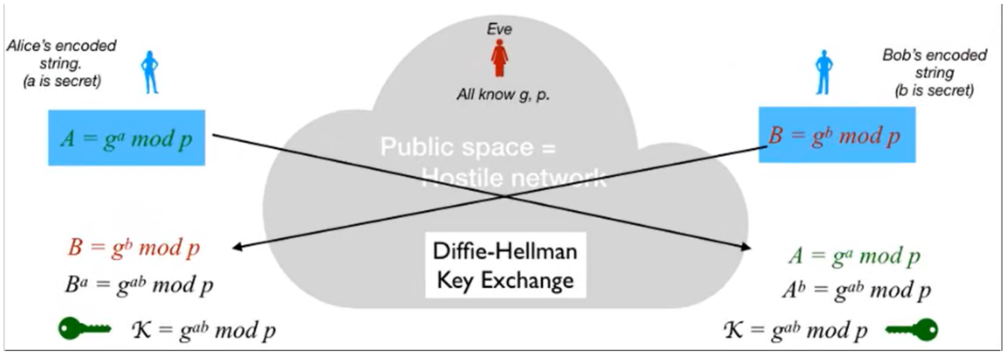
2. Alice chooses a random a and generates $A = g^a \mod p$.
3. Bob chooses a random b and generates $B = g^b \mod p$.

4. Alice sends A to Bob. Bob sends B to Alice.

Stage 3: Shared key generation

5. Alice computes $B^a \mod p = g^{ab} \mod p$.
6. Bob computes $A^b \mod p = g^{ab} \mod p$.
7. Alice and Bob now share the same key $K = g^{ab} \mod p$.

Note that Eve might know g, p, A, and B. However, by nature of the discrete log problem, it's difficult for her to figure out what a, b could be.



Diffie-Hellman Key Exchange

14.3 RSA

The RSA is another public key exchange system, except only one side needs to generate the key. This makes it easier in some cases than Diffie-Hellman, where both Alice and Bob need to generate and send their public keys.

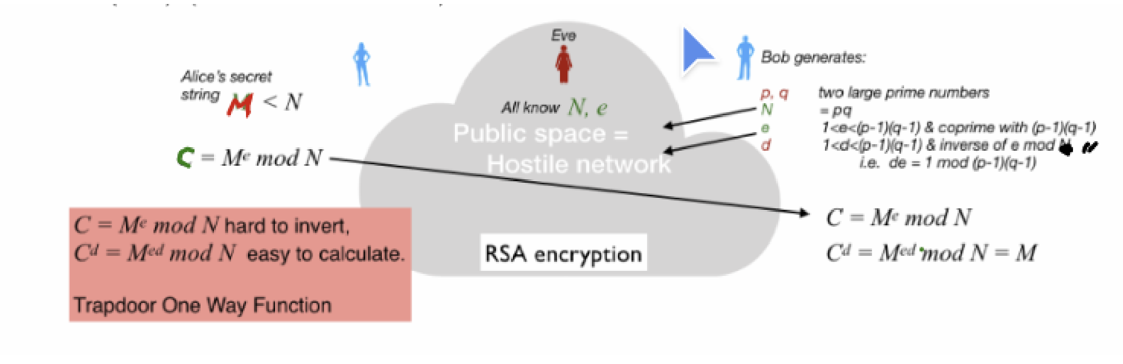
Step 1: Bob setup

1. Bob generates p, q: two large prime numbers.
2. Bob computes $N = pq$ and shares it publicly.
3. Bob finds some e such that $1 < e < (p-1)(q-1)$, and e is coprime with $(p-1)(q-1)$. He shares this publicly.
4. Bob also generates d, the modular inverse of e such that $1 < d < (p-1)(q-1)$ and $d = e^{-1} \mod (p-1)(q-1)$.
   - d solves the equation $de = 1 \mod (p-1)(q-1)$

Step 2: Alice sends her string

5. Alice generates a secret string M less than N.
6. Alice computes $C = M^e \mod N$. She sends this publicly to Bob.
   - Recall that N and e were announced publicly, so Eve could try to brute force what M is. But the discrete log problem takes a very long time classically.

7. Bob computes $C^d = M^{ed} \mod N = M$. Alice and Bob have now exchanged the secret message securely.


Eve
Bob generates:
Alice's secret string $M < N$
All know N, e
Public space = Hostile network
p, q two large prime numbers
N = pq
e 1<e<(p-1)(q-1) & coprime with (p-1)(q-1)
d 1<d<(p-1)(q-1) & inverse of e mod
i.e. de = 1 mod (p-1)(q-1)
$C = M^e \mod N$
$C = M^e \mod N$
$C^d = M^{ed} \mod N = M$
$C = M^e \mod N$ hard to invert,
$C^d = M^{ed} \mod N$ easy to calculate.
Trapdoor One Way Function
RSA encryption

RSA Key Exchange

14.4 Quantum Key Distribution (BB84)

In quantum key distribution, we leverage the no-cloning principle: if Alice sends some qubits to Bob, Eve can't make a copy of those qubits. She can only measure them, which risks changing their state. In BB84, this is used to test a public channel for potential eavesdroppers.

Stage 1: Alice sends key

1. Alice generates a long n-bit string A.
2. Alice also generates a long $2n$-bit string b (needs to be longer than n).
3. Alice takes n photons, corresponding to her n-bit string A. Depending on b_i, she encodes each photon in the computational or Hadamard basis.
   - If $b_i = 0$: $|A_i\rangle$
   - If $b_i = 1$: $(1 - A_i)\,|+\rangle + A_i\,|-\rangle$
4. Alice sends her photons to Bob.

Stage 2: Bob generates string

5. Bob generates his own $2n$-bit string b'.
6. Taking the photons from Alice, Bob measures each photon in the computational basis of $b'_i = 0$, and in the Hadamard basis if $b'_i = 1$.
7. Bob now has the result of his measurement: A'

Stage 3: Alice and Bob compare strings

7. Alice now publicly announces her bitstring b.
8. Bob compares Alice's b and his b'. Wherever they're different, discard that bit. Publicly announce to Alice which are to be discarded.

9. Alice selects a small subset of her original string A without the discarded bits, then sends to Bob. Bob compares it with his corresponding subset in A'.
10. If the bits match to a high degree, then no one has tampered with the channel. Otherwise, someone has altered the bits midway (i.e. made a measurement). Alice and Bob start over on a new channel.

15 Winter Break

16 Winter Break

17 Week 17 (Jan 12 - Jan 16)

17.1 Review of Number Theory

17.1.1 Generators (Primitive Roots mod N)

We call g a generator mod N (or "primitive root mod N") if for every integer a coprime to N, there exists k such that

$$g^k \equiv a \pmod{N}$$

It's a "generator mod N" because every integer from 1 to $N-1$ appears in a periodically. That is, from $k = \{1, 2, ...N-1\}$, $g^k \text{mod } N$ gives every integer $\{1, 2, ..., N-1\}$ without repeating.

As an example, consider $N = 13$, which is obviously coprime with integer a for $a < N$. Consider $g = 7$:

$$\begin{aligned}
7^1 &\mod 13 = 7 & 7^2 &\mod 13 &= 10 & 7^3 &\mod 13 = 5 \\
7^1 &\mod 13 = 9 & 7^2 &\mod 13 &= 11 & 7^3 &\mod 13 = 12 \\
7^1 &\mod 13 = 6 & 7^2 &\mod 13 &= 3 & 7^3 &\mod 13 = 8 \\
7^1 &\mod 13 = 4 & 7^2 &\mod 13 &= 2 & 7^3 &\mod 13 = 1
\end{aligned}$$

Every number from 1 to $N-1 = 12$ has been generated, so 7 is a generator mod N. This has some uses in cryptography, because the distribution of a appears to be random.

17.1.2 Miscellaneous Topics

Chinese Remainder Theorem: For prime p, q and any integer x, y,

$$x \equiv y \mod pq \implies \begin{cases} x \equiv y \mod p \\ x \equiv y \mod q \end{cases}$$

Fermat's Little Theorem: For prime p and integer a,

$$a^{p-1} \equiv 1 \mod p$$

In equivalent forms:

$$\begin{aligned} a^p &\equiv a \mod p \\ a^p - a &\equiv 0 \mod p \\ p &\mid a^p - a \end{aligned}$$

I.e. "p divides $a^p - a$"

Order & Period-Finding: Given a random b and large N, if b and N are coprime, we define

$$f(x) = b^x \mod N$$

The *order* of f, denoted r, is the smallest nonzero integer such that

$$\begin{aligned} b^r &\equiv 1 \mod N \\ b^r - 1 &\equiv 0 \mod N \end{aligned}$$

The *period* of f, denoted r', is the smallest nonzero integer such that

$$f(x + r') = f(x)$$

Here, $r' = kr$, where r is the order and k is some integer. The function is known to be periodic, since

$$\begin{aligned} f(x + r') &= b^x b^{r'} \mod N \\ &= b^x b^{kr} \mod N \\ &= b^x (b^r)^k \mod N \\ &= b^x \mod N \\ &= f(x) \end{aligned}$$

We used the fact that $b^r \equiv 1 \mod N$, from the definition of the order.

In the future, period-finding will become useful for factoring $N = pq$. If r is even, for example, then

$$b^r - 1 = (b^{r/2} + 1)(b^{r/2} - 1) \equiv 0 \mod N$$

If either of the terms in the parentheses is a multiple of N, we learn nothing. Otherwise, we can gain information about p and q using $\text{gcb}(b^{r/2} - 1, N)$. More on that later.

Summary: finding the period of $f(x)$ allows us to determine the prime factors of N.

Note: this is similar to Simon's algorithm. But Simon's algorithm used mod-2 addition, while we're using natural addition. That makes things much harder.

17.2 Breaking RSA

Recall from Week 14 the details of RSA. Bob chooses prime p, q and generates $N = pq$. He also generates e, an integer that's coprime to $\varphi(N) = (p-1)(q-1)$. He shares N and e publicly. Privately, he finds the modular inverse of e mod N, denoted d. Here $ed = 1$ mod N.

Alice has a secret message M she wants to send to Bob. She calcualtes $C = M^e$ mod N and sends publicly. An eavesdropper could know this, but the discrete log problem is hard classically.

From Fermat's little theorem, Bob knows that

$$\begin{aligned} M^{p-1} &= 1 \mod p \\ M^{q-1} &= 1 \mod q \\ M^{(p-1)(q-1)} &= 1 \mod pq \end{aligned}$$

Thus he can find

$$\begin{gathered} C^d = M^{ed} \bmod N = M^{1+k(p-1)(q-1)} \bmod pq \\ C^d = M \mod N \end{gathered}$$

This relies on the fact that the discrete log problem is hard classically. But say Eve (the eavesdropper) knows the period of C, denoted r. That is,

$$C^r = 1 \mod N$$

If this happens, we can break RSA. Two options are given below.

17.2.1 1. Finding p and q

Since $C^r = 1 \bmod N$, if r is even, we have

$$(C^{r/2}+1)(C^{r/2}-1) = 1 \mod N$$

Thus we know that

$$\begin{aligned} C^{r/2}+1 &= p \mod N \\ C^{r/2}-1 &= p \mod N \end{aligned}$$

Or vice versa. We can find $\gcd(C^{r/2}+1, N)$ and $\gcd(C^{r/2}-1, N)$ to factor N as pq. Then we can calculate the totient function $\varphi(N) = (p-1)(q-1)$. From here, find the multiplicative inverse d where $ed = 1 \bmod \varphi(N)$, a classically easy problem.

Then we intercept Alice's encrypted message C and recover the original message M.

$$C^d = M^{ed} \bmod N = M$$

17.2.2 2. Without Finding p and q

There's also a way to break RSA without having to find p and q. We can immediately find a modular inverse d instead.

This involves some number theory. First, note that C is periodic in r: $C^r = 1 \bmod N$. Thus

$$\begin{gathered} C^d = M^{ed} \bmod N = M \\ C^d = C^{d \bmod r} = M \end{gathered}$$

Note that:

$$C^{ed \bmod r} = \left(C^{d \bmod r}\right)^e = M^e \bmod N = C$$

The last step was by definition. So we can say

$$ed = 1 \mod r$$

And finding the modular inverse is a classically easy problem. Note that the modulus here is

r, which we assume to know. We can therefore find d.

Next, intercept Alice's encrypted message C and recover the original message M.

$$C^d = M^{ed} \bmod N = M$$

Interestingly, the order of C is also an order of M. We had $C^r = 1 \bmod N$ and $M^{ed} = M \bmod N$, plus $ed = 1 \bmod r$.

17.3 Shor's Algorithm: Overview

Shor's algorithm can be used to find the period r of $C \bmod N$. Using this, we can break RSA and factorize $N = pq$, where p and q are large primes.

1. Pick a random a between 2 and N. Use Euclid's algorithm to compute $\gcd(a, N)$ classically.
   - If this isn't 1, we found a factor of N, so we can stop.
   - If this is 1, a and N are coprime. Proceed to step 2.
2. Since a and N are coprime, we can find the order of $a \bmod N$, i.e. the smallest integer r such that

   $$\begin{aligned} a^0 &\equiv a^r \mod N \\ a^r &\equiv 1 \mod N \end{aligned}$$

   In other words, $a^r - 1 \equiv 0 \bmod N$, so N divides $a^r - 1$. **This part uses a quantum algorithm.**
3. Now we know that N divides $a^r - 1$. If r is odd, redo with a different a until we get an even r. Then

   $$\begin{aligned} N &\mid a^r - 1 \\ N &\mid (a^{r/2} - 1)(a^{r/2} + 1) \end{aligned}$$

   It can't be the case that N divides $(a^{r/2} - 1)$, since that would imply $a^{r/2} \equiv 1 \bmod N$, where $r/2$ is the order of a. But we were assuming that r was the smallest order of a in step 2. Thus, perhaps N divides $(a^{r/2} + 1)$.
4. Compute

   $$d = \gcd(N, a^{r/2} + 1)$$

   If $d = 1$, then we failed to find a factor of N, so restart with a different a. Otherwise, we found d as one factor of N, and N/d as the other factor.

The only classically hard part of the algorithm is step 2, where we have to find the order of a mod N. This can be done efficiently with a Quantum Fourier Transform (QFT).

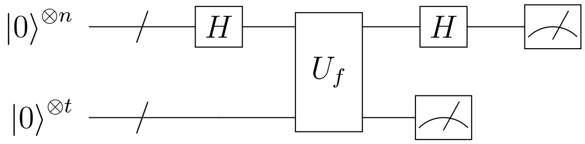
It might be that we have to redo the algorithm a few times, e.g. in step 1 (picking a), step 3 (finding even r), and step 4 (finding nontrivial divisor). Usually, a few runs are enough to factor N.

18 Week 18 (Jan 19 - Jan 23)

18.1 Quantum Subroutine of Shor's Algorithm

Our goal is to find the order of a mod N, given a and N are coprime. The order is denoted r.

Recall Simon's algorithm from Week 9:



Shor's algorithm is similar, except with normal addition instead of mod 2 addition for the order.

Algorithm step	Simon's: $f(x) = f(x \oplus a)$	Shor's: $f(x) = f(x + kr)$
Initialize qubits*	$\|0\rangle^{\otimes n} \|0\rangle^{\otimes t}$	$\|0\rangle^{\otimes n} \|0\rangle^{\otimes t}$
Hadamard input	$\frac{1}{\sqrt{N}} \sum_{x=0}^{2^n-1} \|x\rangle \|0\rangle$	$\frac{1}{\sqrt{N}} \sum_{x=0}^{2^n-1} \|x\rangle \|0\rangle$
Apply oracle	$\frac{1}{\sqrt{N}} \sum_{x=0}^{2^n-1} \|x\rangle \|f(x)\rangle$	$\frac{1}{\sqrt{N}} \sum_{x=0}^{2^n-1} \|x\rangle \|f(x)\rangle$
Measure output**	$\frac{1}{\sqrt{2}} \sum_{k=0}^{1} \|x_0 \oplus ka\rangle$	$\frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} \|x_0 + kr\rangle$
Hadamard/QFT input***	$\frac{1}{\sqrt{2N}} \sum_{y=0}^{2^n-1} \left[(-1)^{x_0 y} + (-1)^{(x_0 \oplus a)y}\right] \|y\rangle$	$\frac{1}{\sqrt{mN}} \sum_{y=0}^{2^n-1} \sum_{k=0}^{m-1} e^{i2\pi(x_0+kr)y/N} \|y\rangle$
Measure input	—	—
Post-processing	$a \cdot y = 0$	$y \longrightarrow y_0 \sim j\frac{2^n}{r}, \quad j \in \mathbb{Z}^+$

At the end of Shor's algorithm, we get a random value $y_0 = j\frac{2^n}{r}$, where j is an integer.

*The input register has n qubits, and the work register has t qubits.

**Here m is the largest integer less than $2^n/r$, since that's how many times we can "fit" r into 2^n.

***Instead of Hadamarding the second time for Shor's, we apply QFT on the input register.

18.2 Shor's Algorithm: More Steps

The basic procedure is the same as above.

First, we initialize the input register $|0\rangle^{\otimes n}$. We send the input register through the Hadamard:

$$|\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |0\rangle$$

Now we send through the oracle $U|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$. The function is $f(x) = a^x \bmod N$.

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |f(x)\rangle$$

Next, we measure the output register. Then the input register collapses to a superposition of all inputs that give that output. Here $m = \left\lfloor \frac{2^n}{r} \right\rfloor$, the largest integer less than $2^n/r$ (the order is r, so we can fit in m times that maximum).

$$|\psi_3\rangle = \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} |x_0 + kr\rangle$$

Now things get interesting. Instead of a Hadamard, we apply QFT to the input register:

$$\begin{aligned}|\psi_4\rangle &= \frac{1}{\sqrt{2^n m}} \sum_{k=0}^{m-1} \sum_{y=0}^{2^n-1} e^{i2\pi(x_0+kr)y/2^n} |y\rangle \\ &= \frac{1}{\sqrt{2^n m}} \sum_{y=0}^{2^n-1} e^{i2\pi x_0 y/2^n} \sum_{k=0}^{m-1} e^{i2\pi\frac{ykr}{2^n}} |y\rangle\end{aligned}$$

Notice that we've separated the y and k within the exponent. So we can rewrite our state as

$$|\psi_4\rangle = \sum_{y=0}^{2^n-1} a_y |y\rangle, \quad a_y = e^{i2\pi x_0 y/2^n} \frac{1}{\sqrt{2^n m}} \sum_{k=0}^{m-1} e^{i2\pi\frac{ykr}{2^n}}$$

Then our probability to measure outcome $y \longrightarrow b$ is

$$P(b) = |a_b|^2 = \frac{1}{2^n m} \left| \sum_{k=0}^{m-1} e^{i2\pi \frac{bkr}{2^n}} \right|^2$$

Note that we've managed to get rid of x_0. Now we can analyze the results of our measurement.

18.2.1 Measuring an integer

If $b = j\frac{2^n}{r}$ exactly, where j is a random integer, then the probability of measuring it is

$$P(b) = \frac{1}{2^n m} \left| \sum_{k=0}^{m-1} e^{i2\pi jk} \right|^2 = \frac{1}{2^n m} \cdot m^2$$
$$P(b) = \frac{m}{2^n}$$

Since $m = \left\lfloor \frac{2^n}{r} \right\rfloor$, we have

$$P(b) = \frac{1}{r}$$

The probability of measuring exactly b is $1/r$.

18.2.2 Measuring a non-integer

What if the outcome isn't exactly b? Consider being within $\frac{1}{2}$ of another integer $j\frac{2^n}{r}$. Say the outcome is $b = j\frac{2^n}{r} + \delta_j$. Using angle identities and simplifying, we have

$$P(b) = \frac{1}{2^n m} \left| \sum_{k=0}^{m-1} e^{i2\pi jk} e^{i2\pi \delta_j kr/2^n} \right|^2 = \frac{1}{2^n m} \left| \frac{1 - e^{i2\pi\delta_j mr/2^n}}{1 - e^{i2\pi\delta_j r/2^n}} \right|$$
$$P(b) = \frac{1}{2^n m} \frac{\sin^2(\pi\delta_j rm/2^n)}{\sin^2(\pi\delta_j r/2^n)}$$

Now since $m = \left\lfloor \frac{2^n}{r} \right\rfloor$, we have $mr \approx 2^n$, so

$$P(b) \approx \frac{1}{2^n m} \frac{\sin^2(\pi\delta_j)}{\sin^2(\pi\delta_j r/2^n)}$$

By choosing a large enough 2^n, we can use the small angle approximation:

$$P(b) \approx \frac{1}{2^n m} \left(\frac{\sin(\pi \delta_j)}{\pi \delta_j r / 2^n} \right)^2 = \frac{2^n}{mr} \left(\frac{\sin(\pi \delta_j)}{\pi \delta_j r / 2^n} \right)^2$$
$$P(b) \approx \frac{1}{r} \left(\frac{\sin(\pi \delta_j)}{\pi \delta_j r / 2^n} \right)^2$$

The sinc function is spiked around b, so we know that the probability of measuring b (with $\delta_j = 0$) is highest.

In particular, if we measure $y \to b\frac{2^n}{r}$, there's a high probability that b will be within $\frac{1}{2}$ of $j\frac{2^n}{r}$ for some integer j.

18.2.3 Interpreting results

After all that, we want to find the form of $b = j\frac{2^n}{r}$, where j is a random integer. Rearrange to get

$$\frac{b}{2^n} = \frac{j}{r}$$

We calculate the continued fraction $\frac{b}{2^n}$, cutting off when a sufficiently large value is encountered in the denominator (since anything else will be negligible). For example, if $b = 11490$ and $n = 14$,

$$\frac{11490}{2^{14}} = 0 + \frac{1}{1 + \frac{1}{2 + \frac{1}{2 + \frac{1}{1 + \frac{1}{7 + \frac{1}{35 + \cdots}}}}}} \approx \frac{54}{77}$$

Here we might reasonably guess that the order is $r = 77$.

Note that this isn't even, so in Shor's algorithm, we'd have to try again until we get an even r.

To summarize Shor's algorithm:

1. Basic steps are similar to Simon's, except with QFT.
2. Probability of measuring b is spiked near b.
3. Use continued fractions to approximate $b/2^n$ and get r from that.
4. Redo as many times as necessary.

19 Week 19 (Jan 26 - Jan 30)

19.1 Review: How to Crack RSA with Shor's

19.1.1 RSA Procedure

Bob generates $N = pq$, where $p and q are large primes. He also finds e, an integer that's coprime to $\varphi(N) = (p-1)(q-1)$. Bob announces N and e publicly. He also finds the modular inverse d of e mod $\varphi(N)$, which satisfies $ed = 1 \bmod \varphi(N)$.

Alice wants to send a secret message M to Bob. She encodes this as $C = M^e \bmod N$ and sends C publicly. From Fermat's little theorem, Bob knows

$$M^{p-1} = 1 \bmod p \quad , \quad M^{q-1} = 1 \bmod q$$
$$M^{(p-1)(q-1)} = 1 \mod pq$$

To recover M, Bob can find

$$C^d = M^{ed} \bmod N = M^{1+k(p-1)(q-1)} \bmod pq = M$$

19.1.2 Shor's Algorithm

Shor's algorithm can factor $N = pq$ and intercept the message. Let Eve be the eavesdropper. First, Eve intercepts C from Alice. Eve also knows N and e, which are public. Eve wants to determine the modular inverse d, which can be used to recover the original message M.

Construct $f(x) = y^x \bmod N$ with $y < N$ and $\gcd(y, N) = 1$. Eve uses a **quantum subroutine** to determine the period r of $f(x)$. That is, r is the smallest nonzero integer such that

$$f(r) = f(0)$$
$$y^r = 1 \mod N$$

If r is odd, start over. Otherwise, rewrite as

$$y^r - 1 = 0 \mod N$$
$$(y^{r/2} + 1)(y^{r/2} - 1) = 0 \mod N$$

If either $(y^{r/2} + 1)$ or $(y^{r/2} - 1)$ are multiples of N, start over. Otherwise, we know that $(y^{r/2} + 1) = \lambda_p p$ and $(y^{r/2} - 1) = \lambda_q q$. p and q might be switched, but that doesn't matter.

Now find p and q:

$$p = \gcd(y^{r/2} + 1, N)$$
$$q = \gcd(y^{r/2} - 1, N)$$

Since Eve knows p and q, she can find $\varphi(N) = (p-1)(q-1)$ and calculate d using

$$ed = 1 \mod (p-1)(q-1)$$

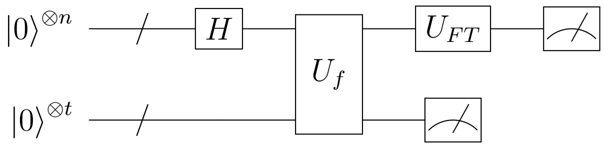
Finally, she recovers M by exponentiating the intercepted message:

$$C^d = M^{ed} \bmod N = M$$

19.1.3 Order-Finding Subroutine

Classically, finding the order of $f(x) = y^x \bmod N$ is difficult. However, we can do this with quantum computing.

Construct a quantum circuit with 2 registers of size n and t. The oracle is $U\left|x\right\rangle\left|y\right\rangle = \left|x\right\rangle\left|y \oplus f(x)\right\rangle$.



Our state before the QFT is (with $m = \lfloor 2^n/r \rfloor$)

$$|0\rangle^{\otimes n}|0\rangle^{\otimes t} \longrightarrow \frac{1}{\sqrt{2^n}}\sum_x |x\rangle^{\otimes n}|0\rangle^{\otimes t} \longrightarrow \frac{1}{\sqrt{2^n}}\sum_x |x\rangle^{\otimes n}|f(x)\rangle^{\otimes t}$$

When we measure the bottom register, the top register collapses into a superposition of all possible x for some $f(x)$.

$$\frac{1}{\sqrt{m}}\sum_{k=0}^{m-1}|x_0 + kr\rangle$$

Now apply the QFT gate:

$$\begin{aligned} U_{FT}\frac{1}{\sqrt{m}}\sum_{k=0}^{m-1}|x_0+kr\rangle &= \frac{1}{\sqrt{2^n m}}\sum_{y=0}^{2^n-1}\sum_{k=0}^{m-1} e^{i2\pi(x_0+kr)\cdot y/2^n}\,|y\rangle \\ &= \frac{1}{\sqrt{2^n m}}\sum_{y=0}^{2^n-1} e^{i2\pi x_0 y/2^n}\sum_{k=0}^{m-1} e^{i2\pi kry/2^n}\,|y\rangle \end{aligned}$$

We've managed to factor away the unknown x_0. Using geometric sums and properties of Euler's formula, it can be shown that

$$\sum_{k=0}^{m-1} = e^{i2\pi kry/2^n} = \begin{cases} 2^n/r & \text{If } l = j\frac{2^n}{r} \text{ for some integer } j \\ 0 & \text{If } l \text{ is an integer but } l \neq j\frac{2^n}{r} \end{cases}$$

Therefore, by measuring our state, we get some y such that

$$y \longrightarrow b \sim j\frac{2^n}{r}$$

Moreover, it turns out that there's a high probability b will be within $\frac{1}{2}$ of an integer. To find r, we can find the continued fraction expression of $\frac{b}{2^n}$ and cut off at a sufficiently high value in the denominator:

$$\frac{b}{2^n} = \frac{j}{r}$$

Whenever we run this subroutine again, we get a random value of $y \longrightarrow b$ and subsequently r. This can be used to discover information about N and factorize it.

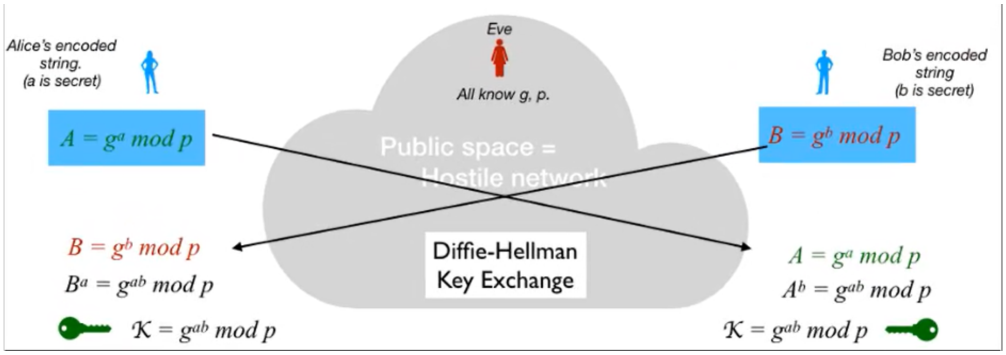
19.2 Bonus: Cracking Diffie-Hellman with Shor's

19.2.1 Diffie-Hellman Key Exchange

(Review of Week 14) First, Alice and Bob publicly agree on a large prime p and a primitive root $2 \leq g \leq p-2$. Next, Alice chooses a random a and generates $A = g^a \bmod p$. Bob also chooses a random b and generates $B = g^b \bmod p$.

Both send their keys to each other. Alice computes $B^a \bmod p = g^{ab} \bmod p$. Bob computes $A^b \bmod p = g^{ab} \bmod p$. Both now share the same key

$$K = g^{ab} \mod p$$



Diffie-Hellman key exchange procedure

19.2.2 Quantum Algorithm

(Skim this, details aren't good. But know that it can be done with QFT.)

This is an algorithm that can crack a discrete log. Given $A = g^a \bmod p$, our goal is to find a. Using the order-finding subroutine of Shor's algorithm, we can find the period r of the function $g^x \bmod p$.

Now consider

$$f(x_1, x_2) = A^{x_1} g^{x_2} \bmod p = g^{ax_1} g^{x_2} \bmod p = g^{ax_1 + x_2} \bmod p$$

This is periodic: $f(x_1 + kr, x_2 - kar) = f(x_1, x_2)$.

Next, define the oracle $U |x_1\rangle |x_2\rangle |y\rangle = |x_1\rangle |x_2\rangle |y \oplus f(x_1, x_2)\rangle$, where $f(x_1, x_2) = b^{x_1} a^{x_2}$.

Initialize our three registers to $|0\rangle$. Hadamard them to get a superposition and apply the oracle.

$$|0\rangle^{\otimes n} |0\rangle^{\otimes n} |0\rangle \longrightarrow \frac{1}{2^n} \sum_{x_1=0}^{2^n-1} \sum_{x_2=0}^{2^n-1} |x_1\rangle |x_2\rangle |f(x_1, x_2)\rangle$$

Apply QFT:

$$\begin{aligned} &\longrightarrow \frac{1}{2^n} \frac{1}{\sqrt{r}} \sum_{l=0}^{r-1} \sum_{x_1=0}^{2^n-1} \sum_{x_2=0}^{2^n-1} e^{i2\pi(alx_1 + lx_2)/r} |x_1\rangle |x_2\rangle |\hat{f}(al, l)\rangle \\ &= \frac{1}{\sqrt{r}} \sum_{l=0}^{r-1} \left[\frac{1}{\sqrt{2^n}} \sum_{x_1=0}^{2^n-1} e^{i2\pi alx_1/r} |x_1\rangle \right] \left[\frac{1}{\sqrt{2^n}} \sum_{x_2=0}^{2^n-1} e^{i2\pi lx_2/r} |x_2\rangle \right] |\hat{f}(al, l)\rangle \end{aligned}$$

Apply the inverse QFT to get

$$\longrightarrow \frac{1}{\sqrt{r}} \sum_{l=0}^{r-1} |a\frac{\tilde{l}}{r}\rangle \, |\frac{\tilde{l}}{r}\rangle \, |\hat{f}(al, l)\rangle$$

Measure the first two registers, round, and take the ratio to learn a. Now we can take Bob's public key B and find the private keys:

$$B^a = g^{ab} \bmod p = K$$

19.3 Continuum Limit of Wavefunctions

We're going to have a brief review of quantum mechanics. In particular, how to discretize a continuous variable (e.g. position, momentum). First, the wavefunction is represented by $|\psi\rangle$. This isn't in any particular basis. We can project onto a basis (position basis, momentum basis, etc.) to measure its components. For now we put into the position basis with N divisions.

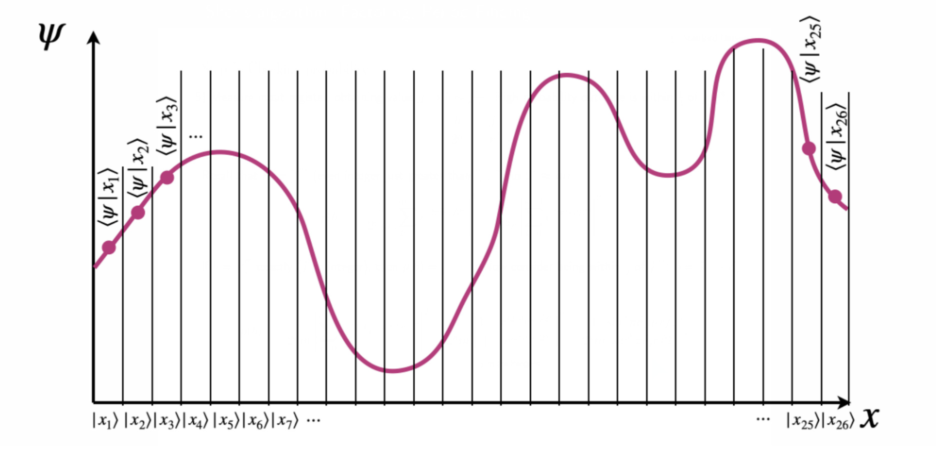
$$|\psi\rangle = \sum_{i=-\infty}^{\infty} \text{Proj}_{x_i}(\psi)\Delta x_i = \sum_{i=-\infty}^{\infty} |x_i\rangle \langle x_i|\psi\rangle \, \Delta x$$

Normally, the probability of measuring at any specific x_i is $| \langle\psi|x_i\rangle |^2$. But a nuance when going to the continuous limit is that we can't say "what's the probability of measuring at x_0?" or similar. $\Delta x \to 0$, so the probability will also go to 0. Instead, we can ask "what's the probability of measuring between x_0 and $x_0 + \Delta x$?"

Additionally, the wavefunction must be normalized. We can derive the condition from classical summation:

$$\sum P(x \text{ in } \psi) = \sum_x \langle\psi|x\rangle \langle x|\psi\rangle = 1 \qquad \text{(discrete)}$$

$$\int_{-\infty}^{+\infty} |\psi|^2 \; dx = \int_{-\infty}^{+\infty} \psi^*(x)\psi(x) \; dx = \int_{-\infty}^{+\infty} \langle\psi|x\rangle \langle x|\psi\rangle \;\; dx = 1 \qquad \text{(continuous)}$$



Discretizing the wavefunction. Looks similar to integration.

19.3.1 Dr. Oas's Grad School Paradox

From commutation relations, we know that

$$[X, P] = i\hbar$$

Slam on $|x_i\rangle$ to get

$$\begin{aligned}
\langle x_i| [X, P] |x_i\rangle &= i\hbar \\
\langle x_i| XP |x_i\rangle - \langle x_i| PX |x_i\rangle &= i\hbar \\
x_i \langle x_i| P |x_i\rangle - \langle x_i| P |x_i\rangle x_i &= i\hbar \\
0 &= i\hbar \qquad \text{(huh?)}
\end{aligned}$$

The problem is that for continuous observables like position and momentum, the ket vectors are no longer normalizable: $\langle x_i|x_i\rangle \neq 1$. Instead, we have to use the Dirac delta function: $\langle x_i|x_j\rangle = \delta(x_i - x_j)$.

$$\delta(x) = \begin{cases} 0 & x \neq 0 \\ \infty & x = 0 \end{cases} \quad , \quad \int_{-\infty}^{+\infty} \delta(x)dx = 1$$

19.4 The Schrödinger Equation

19.4.1 Time-Dependent Form (General)

The Schrödinger Equation describes the evolution of a state over time. Its general form is

$$i\hbar\frac{\partial}{\partial t} |\Psi\rangle = \hat{H} |\Psi\rangle$$

where $\hat{H} = \frac{\hat{p}^2}{2m} + V$ is the operator version of the Hamiltonian for classical energy. Plugging in the Hamiltonian and considering a continuous wavefunction, the equation becomes

$$i\hbar \frac{\partial \Psi(x,t)}{\partial t} = -\frac{\hbar}{2m}\frac{\partial^2}{\partial x^2}\Psi(x,t) + V\Psi(x,t)$$

This is the general *Time-Dependent Schrödinger Equation.* Thus our limitations on the time-dependent quantum state Ψ are:

i) Ψ must satisfy the Schrödinger Equation

ii) Ψ must always be normalized

iii) Ψ must be continuous

iv) $\frac{d\Psi}{dt}$ must be continuous, unless $V \to \infty$

v) $\Psi \to 0$ as $x \to \pm\infty$

19.4.2 Time-Independent Form

For *plane wave states*, we have $\hat{H}\psi = E\psi = i\hbar\frac{\partial \psi}{\partial t}$, where E is the total energy. It turns out that we can factorize the wavefunction as $\psi(x,t) = \psi(x)g(t)$. Thus if the potential is time-independent, the Schrödinger Equation becomes

$$E\psi(x)g(t) = -\frac{\hbar}{2m}\frac{\partial^2}{\partial x^2}\psi(x)g(t) + V\psi(x)g(t)$$
$$E\psi = -\frac{\hbar}{2m}\frac{\partial^2}{\partial x^2}\psi + V\psi$$

The second equation is the *Time-Independent Schrödinger Equation.* Note that we used lowercase ψ for the Time-Independent SE, in contrast to the Ψ for the Time-Dependent SE. ψ only depends on x, while Ψ depends on both x and t.