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 $< s_0, s_1, s_2, ...; p_t, q_t > \;\rightarrow\; \delta(q_0, \alpha) \;\rightarrow\; < \sigma_1; p_1, q_1 >$

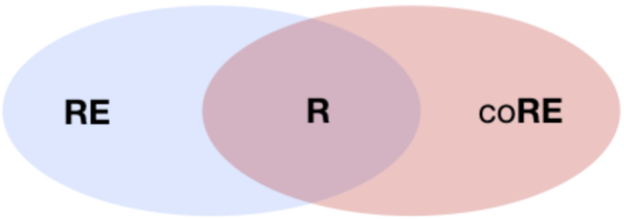
Possibilities:

- 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:

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

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)

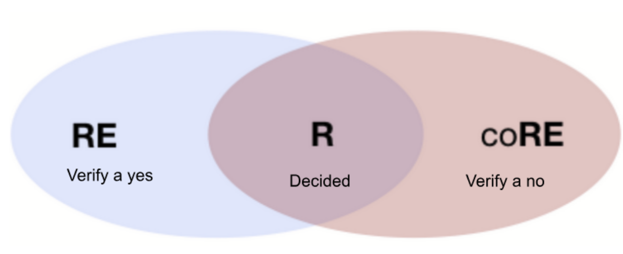
TIME($f(n)$) $\subseteq$ SPACE($f(n)$), since a TM can only access 1 piece of memory in each time step.

Classes:

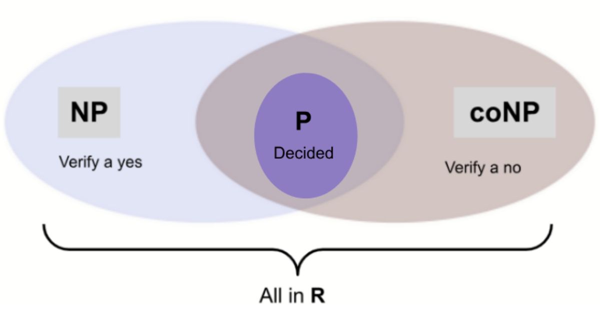
- *P*: class of problems solvable by TM in TIME(n^k)
- *PSPACE*: class of problems solvable by TM in SPACE(n^k)

- *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?



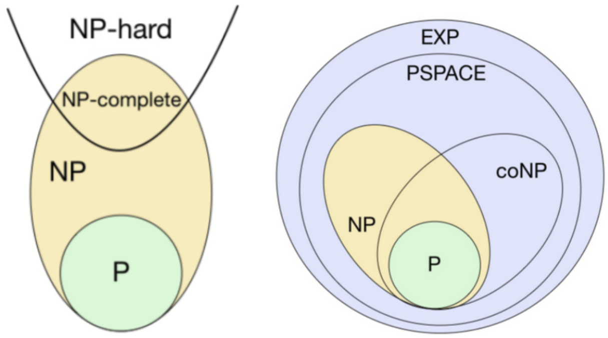
What can be computed *efficiently* (polynomial time)?

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.*

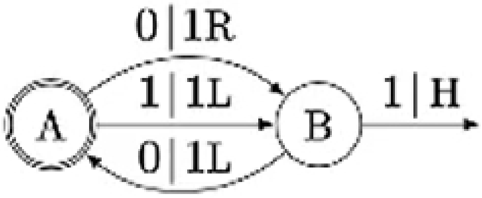
3 Week 3 (Sep 7 - Sep 13)

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

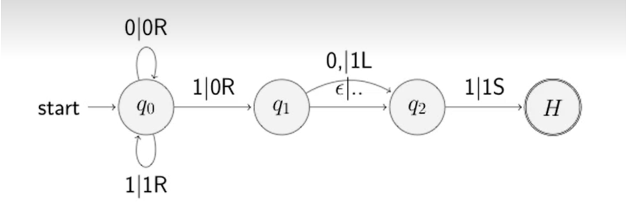


There's exactly one arrow out of each state for each transition (0 or 1).

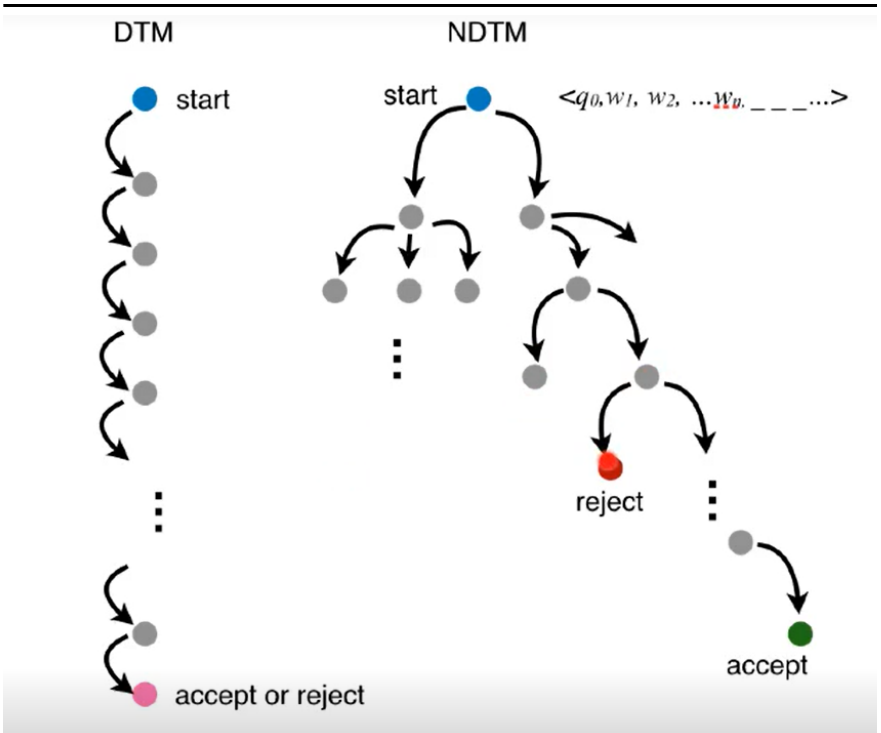
An NDTM can be represented as

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

- 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 $\rightarrow$ 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.

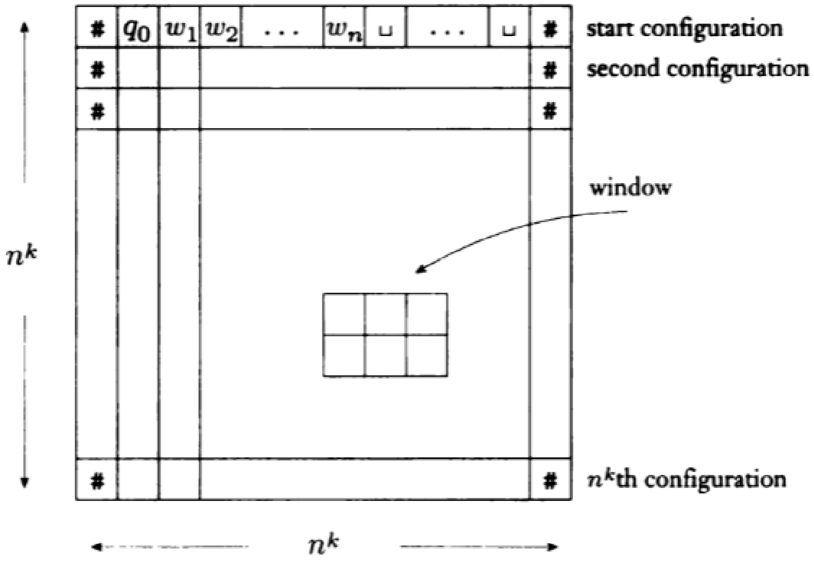
1) SAT is in *NP* (easy)
2) For all $A \in$ *NP*, A is polynomial-time reducible to SAT (hard)

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.

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

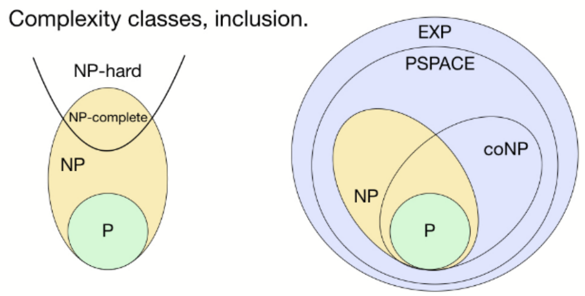
3.4.3 Analysis

- 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)

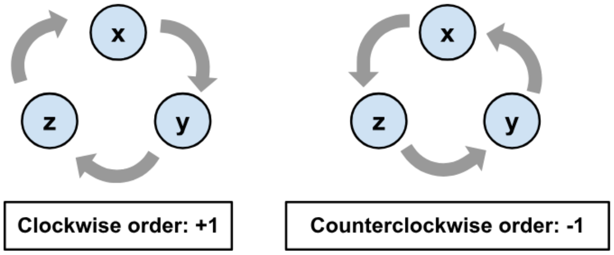
5.3 Relationship Between Gates

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”):

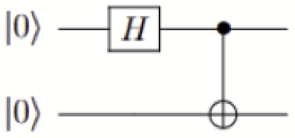
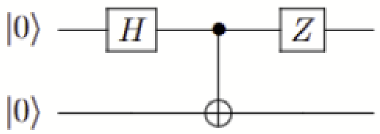
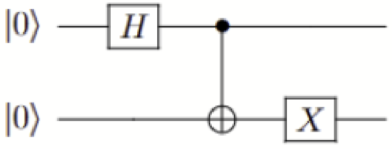
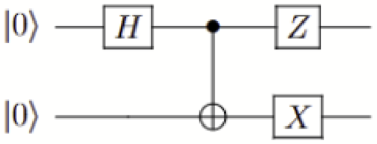
$$H \otimes H \otimes ... \otimes H \left|00...0\right\rangle = H^{\otimes n} \left|0\right\rangle_n = \frac{1}{2^{n/2}} \sum_{y_0=0}^{1} \left|y_{n-1}\right\rangle ... \left|y_0\right\rangle$$

6 Week 6 (Sep 28 - Oct 4)

6.1 Bell Basis

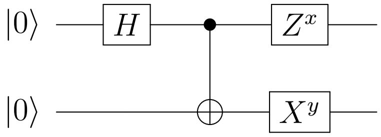
6.1.1 Constructing the Bell Basis

Four entangled states:

$\left|\phi_+\right\rangle = \frac{\left|00\right\rangle + \left|11\right\rangle}{\sqrt{2}}$	
$\left|\phi_-\right\rangle = \frac{\left|00\right\rangle - \left|11\right\rangle}{\sqrt{2}}$	
$\left|\psi_+\right\rangle = \frac{\left|01\right\rangle + \left|10\right\rangle}{\sqrt{2}}$	
$\left|\psi_-\right\rangle = \frac{\left|01\right\rangle - \left|10\right\rangle}{\sqrt{2}}$	

Bell States & Construction with Quantum Circuits

More simply,

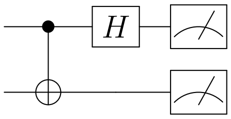


Here,

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

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.

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

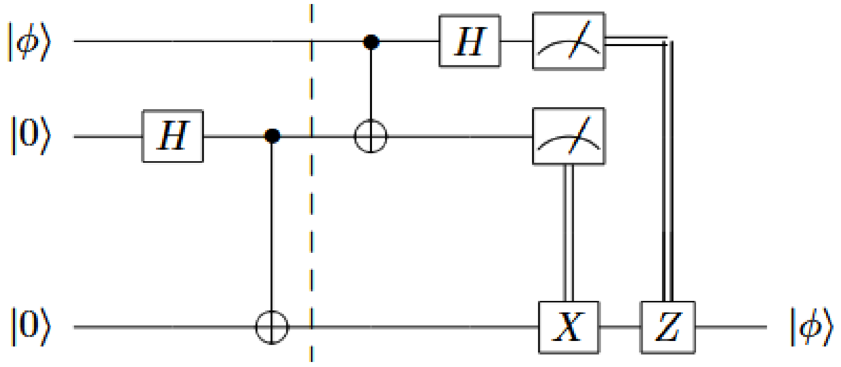
3. Alice measures her qubits and sends to Bob classically (like email).

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.

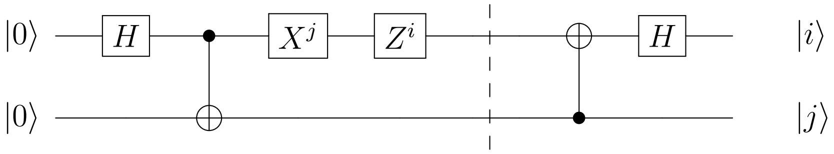
Say Alice and Bob share an entangled resource $|\phi\rangle = |00\rangle + |11\rangle$. Alice wants to send two bits (i, j) to Bob. She encodes them as:

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

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:

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

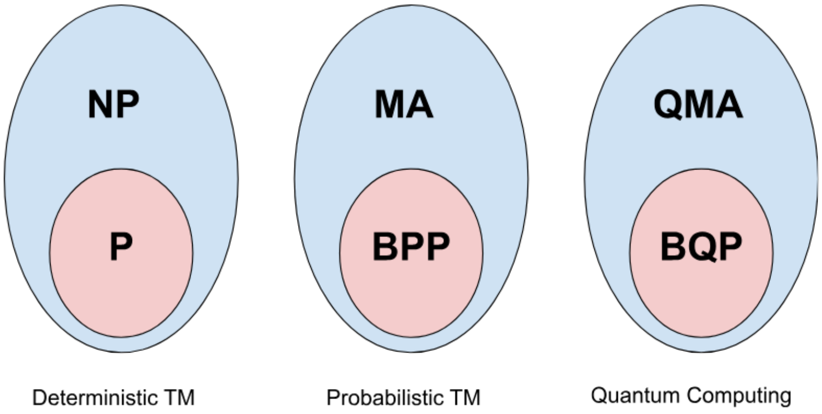
After grouping terms:

$$\frac{1}{4} \left[\left|0\right\rangle \left(\left|\Phi_+\right\rangle + \left|\Phi_-\right\rangle\right) \left|0\right\rangle + \left|0\right\rangle \left(\left|\Psi_+\right\rangle + \left|\Psi_-\right\rangle\right) \left|1\right\rangle + \left|1\right\rangle \left(\left|\Psi_+\right\rangle - \left|\Psi_-\right\rangle\right) \left|0\right\rangle + \left|1\right\rangle \left(\left|\Phi_+\right\rangle - \left|\Phi_-\right\rangle\right) \left|1\right\rangle\right]_{ACCB}$$

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

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

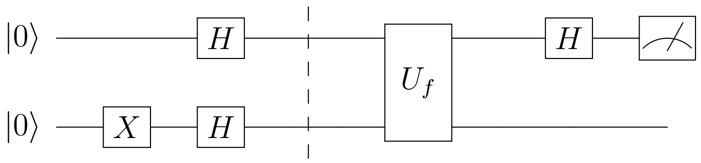
Kind of a toy algorithm, but shows a quantum algorithm faster than its classical counterpart.

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: $U_f\,|\psi\rangle = \pm(|0\rangle - |1\rangle)(|0\rangle - |1\rangle)$.

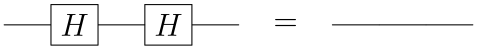
If we apply the Hadamard on the first qubit,

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

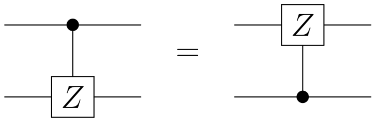
$z = 0$ if constant, $z = 1$ if balanced. So measuring the first qubit determines whether it's constant or balanced, in just 1 oracle query.

7.4 More Gate Identities

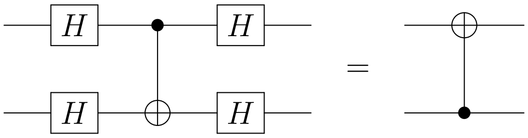
Application of two Hadamards is the identity ($H^2 = I$)



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)

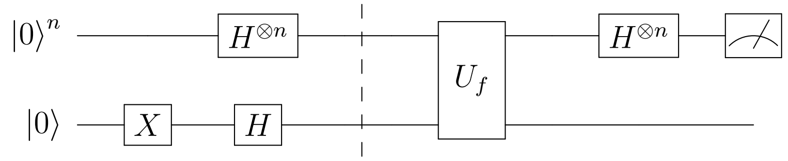
8.1 Deutsch-Jozsa Algorithm

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:

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

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?

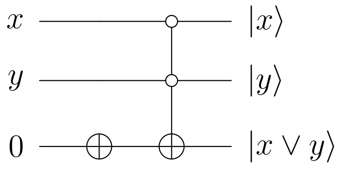
CNF = conjunctive normal form, OR inside parentheses and AND outside

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

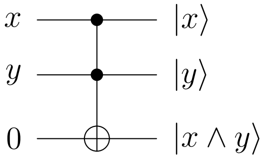
8.2.1 Implementation of AND and OR

Easy enough, based on the Toffoli gate.

The OR circuit:



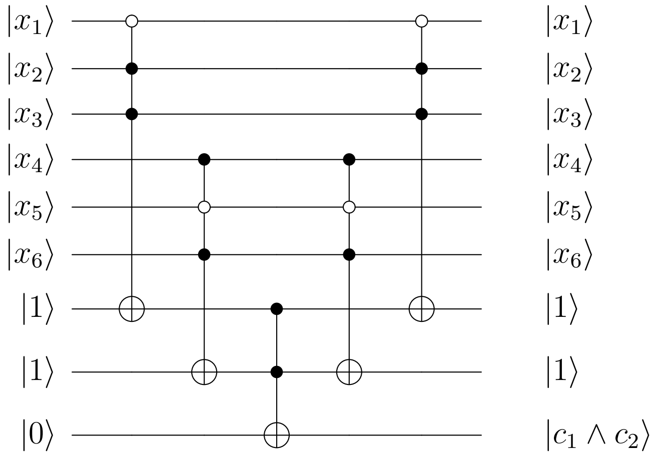
The AND circuit:



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.

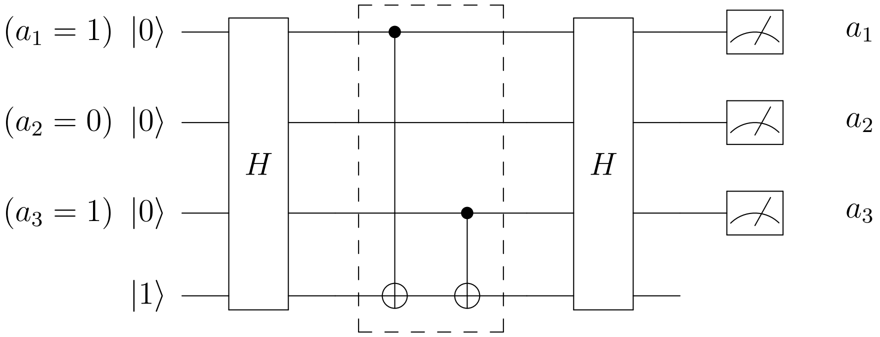
8.3 Bernstein-Vazirani Algorithm

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:



$(a_1 = 1)$ $\left|0\right\rangle$
$(a_2 = 0)$ $\left|0\right\rangle$
$(a_3 = 1)$ $\left|0\right\rangle$
$\left|1\right\rangle$
H
H
a_1
a_2
a_3

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 $\left|0\right\rangle$ 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:

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

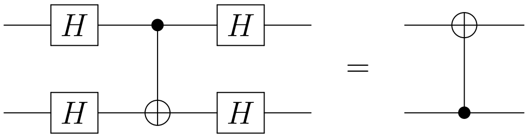
Focus on the sum over x:

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

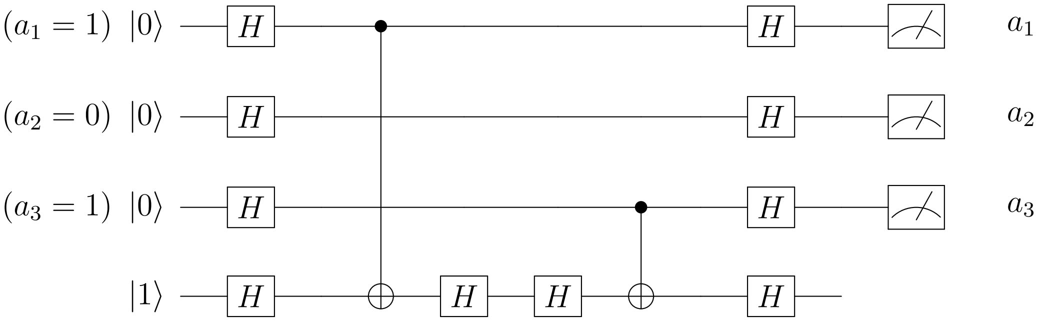
If $a_j \neq y_j$ at any point, the product vanishes. The only surviving output is $\left|a\right\rangle^n \left|1\right\rangle$. So we've identified a in 1 query to the oracle.

8.3.2 Gate Identities

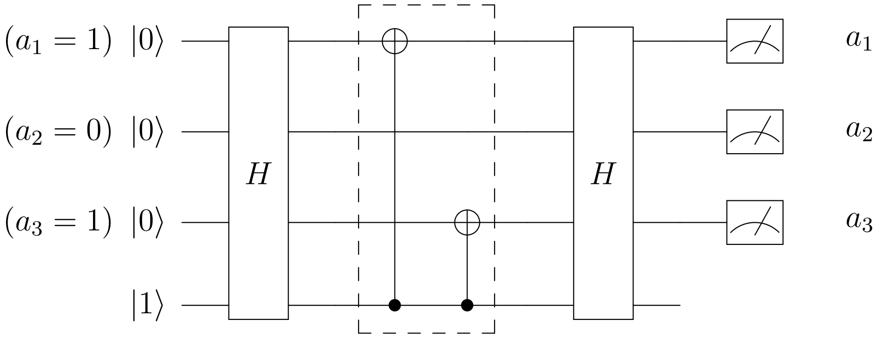
There's an easier way to see why this works. Recall the circuit identity:



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


$(a_1 = 1)$ $\left|0\right\rangle$
$(a_2 = 0)$ $\left|0\right\rangle$
$(a_3 = 1)$ $\left|0\right\rangle$
$\left|1\right\rangle$
H
a_1
a_2
a_3

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: $\text{Tr}(\rho) = 1$

- 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

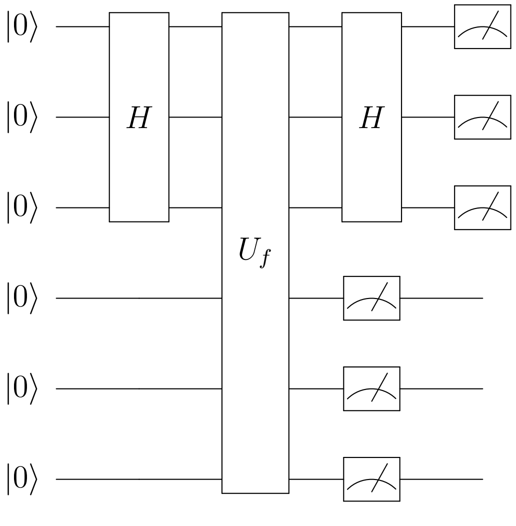
Consider $f(x)$ such that for 2 input bitstrings x and y, $f(x) = f(y)$ if $x \oplus y = a$, where a is some positive integer less than 2^n.

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 $-\pi \leq x \leq +\pi$. Disregard the imaginary numbers, and let

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

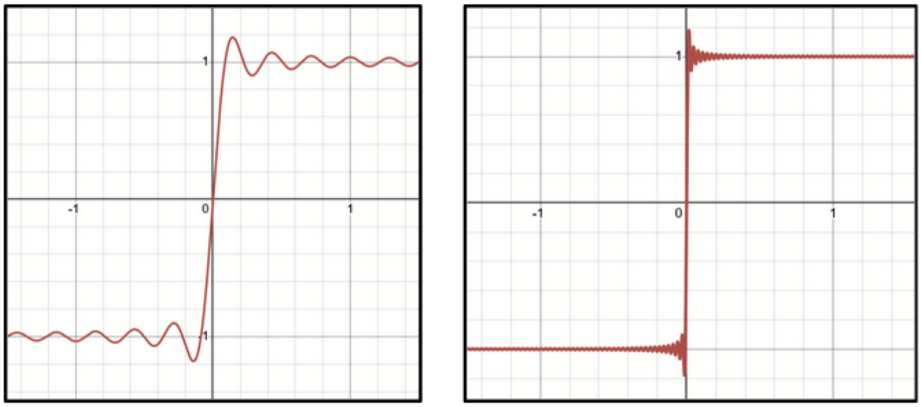
Expansion coefficients are found with the inner product, as before:

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

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.

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.