UP135 Notes

Gene Yang

2025 - 2026

Contents

1 Week 1 (Aug 24-30)

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

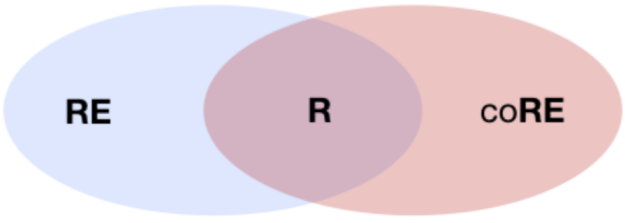
Possibilities:

- 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:

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

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

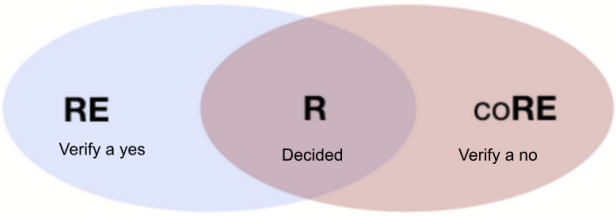
- SPACE($f(n)$): Class of problems that grows in $O(f(n))$ in space (memory)

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

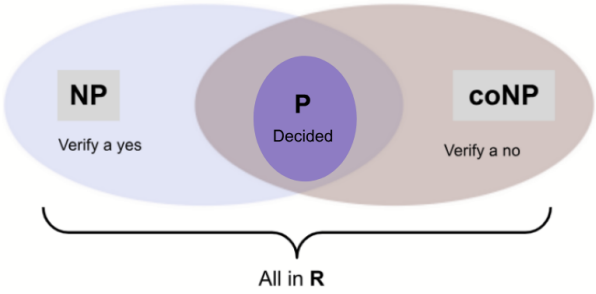
Classes:

- *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)?

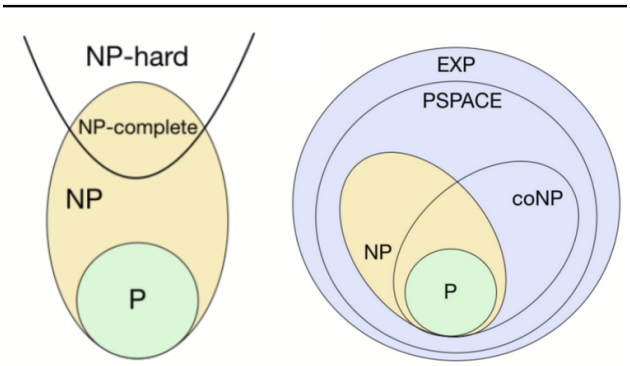
Note that $NP \cap coNP \neq P$.

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.*

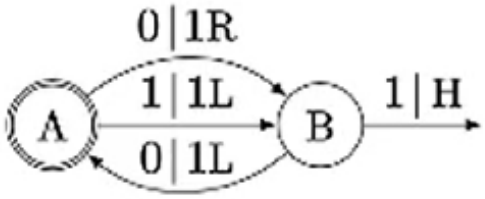
3 Week 3 (Sep 7-13)

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

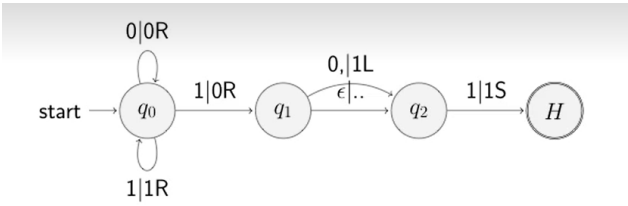


There's exactly one arrow out of each state for each transition (0 or 1).

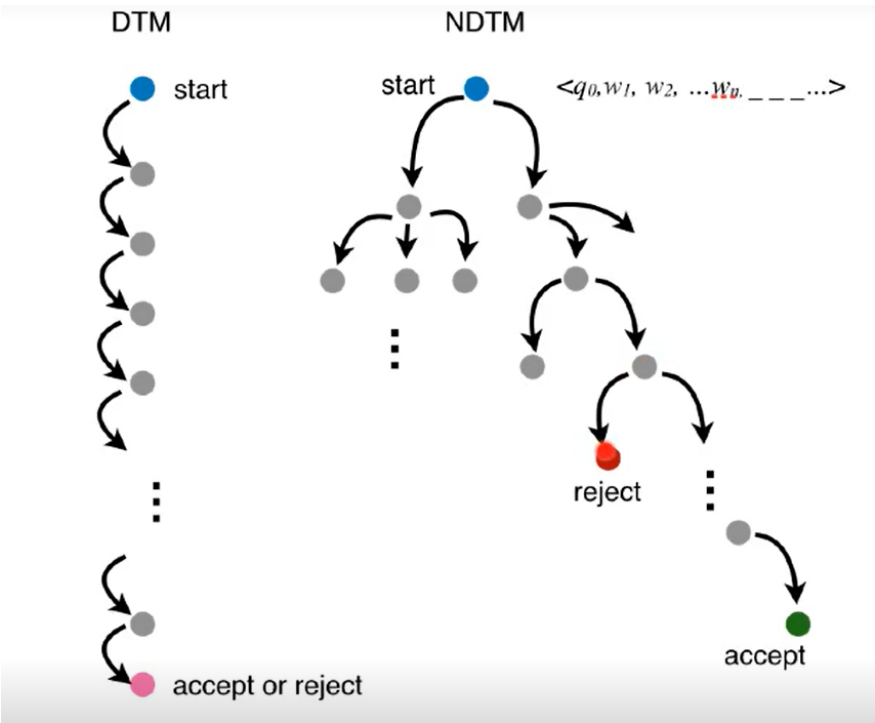
An NDTM can be represented as

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

- 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
     - $A = \{w | V \text{accepts } <\text{w, c}> \text{ for some string c}\}$
     - 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.

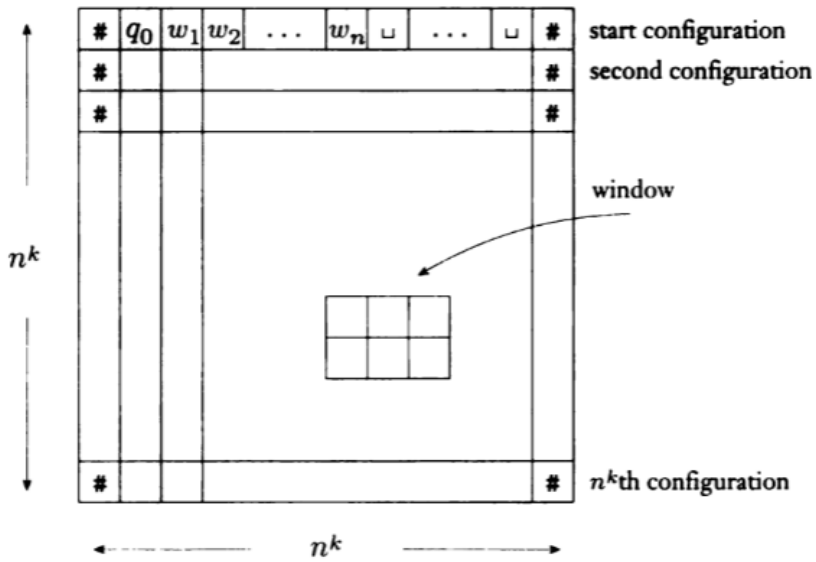
1) SAT is in *NP* (easy)
2) For all $A \in NP$, A is polynomial-time reducible to SAT (hard)

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.

$$\phi_{cell} = \bigwedge_{1 \le i,j \le n^k} \left[\underbrace{\left(\bigvee_{s \in C} x_{i,j,s} \right)}_{\text{at least 1 var in each cell}} \wedge \underbrace{\left(\bigwedge_{\substack{s,t \in C \\ s \neq t}} (\bar{x_{i,j,s}} \vee \bar{x_{i,j,t}}) \right)}_{\text{only 1 var in each cell}} \right]$$

$$\begin{aligned}
\phi_{start} &= x_{1,1,\#} \wedge x_{1,2,q_0} \wedge \\
&= x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge ... \wedge \\
&= x_{1,n+3,\sqcup} \wedge ... \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#} \\
\phi_{move} &= \bigwedge_{1 \le i,j \le n^k} \text{(the 2 x 3 window at (i, j) is a legal transition)} \\
&= \bigvee_{a_1 ... a_6} x_{i,j,a_i} \qquad \text{(for each cell } x_{i,j} \text{ and its symbol } a_i\text{)} \\
\phi_{accept} &= \bigvee_{1 \le i,j \le n^k} x_{i,j,q_{accept}}
\end{aligned}$$

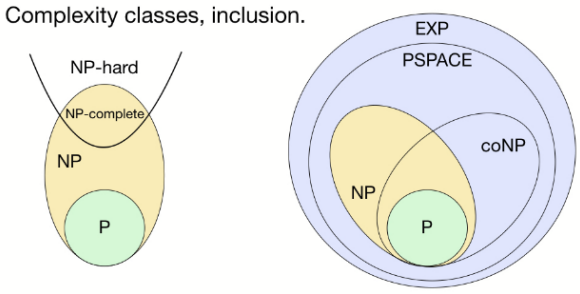
3.4.3 Analysis

- 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)



$P \subseteq NP \subseteq PSPACE = NSPACE \subseteq EXP$ but $P \subset EXP...$

- *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 14-20)

4.1 Review: Complexity Classes

$$P \subseteq NP \subseteq PSPACE = NSPACE \subseteq EXP$$
$$P \subset EXP$$

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 $\forall$, $\exists$.

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 $\forall$, $\exists$ grouped at start of expression.
- Fully quantified: all variables bound by $\forall$, $\exists$.

4.3 Probabilistic Turing Machines

Deterministic TM (DTM):

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

Non-deterministic TM (NDTM):

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

$\mathcal{P}$ 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 $\mathbf{BPP} = \mathbf{P}$, but no proof yet.

4.4 *NP* ∩ *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 21-27)

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)

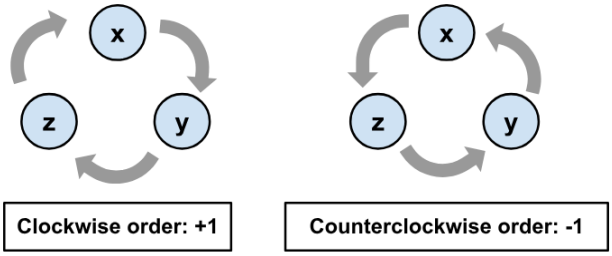
5.3 Relationship Between Gates

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:

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

More generally, it creates a superposition of all inputs, a normalized sum of all n-length

bitstrings ("Walsh-Hadamard Transform"):

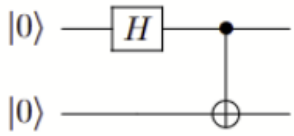
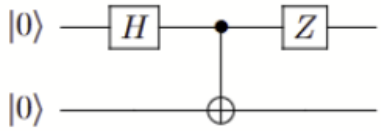
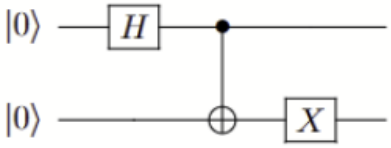
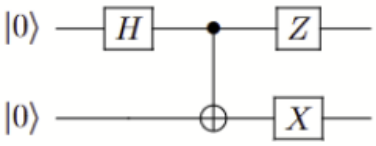
$$H \otimes H \otimes ... \otimes H \left|00...0\right\rangle = H^{\otimes n} \left|0\right\rangle_n = \frac{1}{2^{n/2}} \sum_{y_0=0}^{1} \left|y_{n-1}\right\rangle ... \left|y_0\right\rangle$$

6 Week 6 (Sep 28 - Oct 4)

6.1 Bell Basis

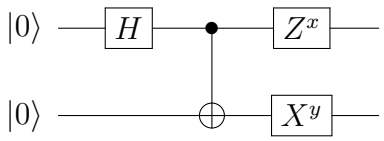
6.1.1 Constructing the Bell Basis

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,

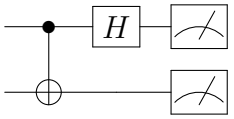


Here,

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

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.

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

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

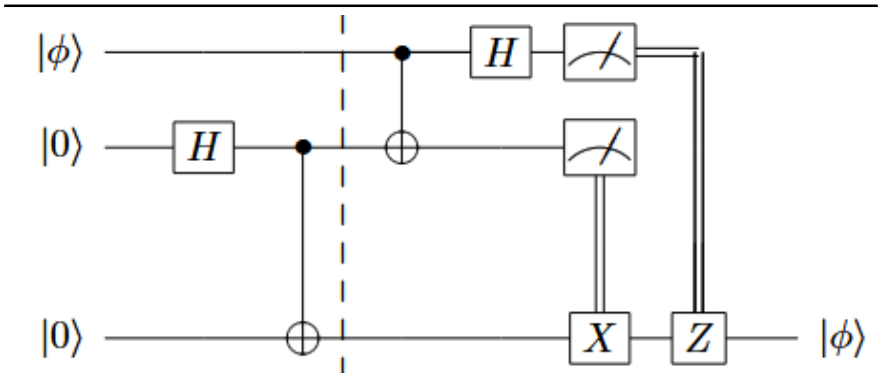
4. Depending on Alice's measurements, Bob applies some operators to his entangled qubit.

   - 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

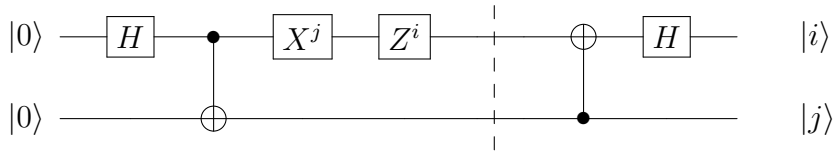
bits (i, j) to Bob. She encodes them as:

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

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.

7 Week 7 (Oct 5-11)

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 $f(a) = a \mod 2$,

$$2 = |010\rangle \longrightarrow U_f |010\rangle = |000\rangle$$
$$4 = |100\rangle \longrightarrow U_f |100\rangle = |000\rangle$$

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

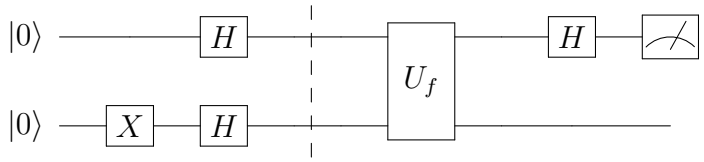
Kind of a toy algorithm, but shows a quantum algorithm faster than its classical counterpart.

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

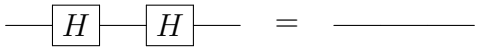
If we apply the Hadamard on the first qubit,

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

$z = 0$ 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 ($H^2 = I$)



Surrounding a CNOT with Hadamards flips the CNOT: