Modes of Computations

Thomas Schwarz, SJ

Computation Principles

- Representations hold information
- Computation is a sequence of representations
- Computations can be open or closed
- Computations have characteristic speeds of resolution
- Complexity measures the time or space essential to complete computations
- Finite representations of real processes always contain errors.

Classical Computation

- Data is represented by elements (switches) that have two or more states
    - For technical reasons today: usually two states
    - Information in such a switch is called a bit

Classical Computation

- What is presented to the programmer:
  - Computer reads instructions from memory
  - Computer acts on instructions by changing memory locations
    - Example: addi x, 5
      - Load x into accumulator, load 5 into a register, add results, move accumulator results back into memory where x is located

Standard Model of Computing is an Idealization

- Instructions do not take the same amount of time
  - Almost since the beginning of computer architecture
  - Idealization: Fetch-Execute Cycle with fixed timing
- Instructions are not performed serially
  - Pipelining of instructions
  - Reordering of instructions by compiler or architecture

Standard Model of Computing is an Idealization

- Memory access is not uniform
    - Early modification: Virtual Memory

Standard Model of Computing is an Idealization

- Modern modification:
  - Registers
  - Cache Level 1
  - Cache Level 2
  - Cache Level 3
  - Main Memory (DRAM)
  - Storage
    - Buffer - Cache - HDD block / SSD page

Standard Model of Computing is an Idealization

- Multi-threaded (e.g. multi-core) :
  - Many instructions & access to variables are not thread-safe
    - E.g.: Can only argue that a flag is either set or not if the flag is "atomic" (with software and hardware support)
  - Multi-core architecture manages to prevent a processor from having a different view of memory than another processor
    - But this is getting more and more difficult

Standard Model of Computing is an Idealization

- Storage and Memory systems prioritize reads over writes
- In case of failure, bad things can happen:
    - Can store a block
    - Read from this block
    - Power failure
    - Read from the block:
        - Value has changed

Standard Model of Computing

- Contract between system and programmer:
  - System does what programmer wants, but in a different, usually faster way
  - With a few exceptions, which makes multi-threaded computing so challenging

Standard Model of Computing

- Turns out that the optimizations of modern computing systems **do not** create genuine new capabilities
- We can ***emulate*** a modern system using an old one
- We can even ***emulate*** a modern system using a model of computing used in the 30s and 40s to model what Mathematics can compute:
    - Turing machine

Quantum Computing

- Data is represented by qubits
  - qubits can exist as a super-imposition of two states
    - Qubit state is a linear combination of 0 and 1
    - $\alpha|0> + \beta|1>$, $\alpha, \beta \in \mathbb{C}$ probability amplitude
    - Probability of measuring qubit as zero is α^2, as 1 is β^2, and so $\alpha^2 + \beta^2 = 1$
  - qubits can be *entangled*: State of one qubit is correlated to the state of another qubit

Quantum Computing

- Once a qubit is measured, it is either 0 or 1
- Before a qubit is measured, it has an infinite amount of information

Quantum Computing

- A quantum logic gate operates on a small number of qubits
  - Representation:
    - Represent a register of n qubits as $\begin{pmatrix} \alpha_1 \\ \beta_1 \\ \alpha_2 \\ \beta_2 \\ \vdots \\ \alpha_n \\ \beta_n \end{pmatrix}$
    - Gate can be represented as unitary matrices
    - Actual hardware gates introduce errors
      - Need quantum error correction

Quantum Computing vs. Classical Computing

- No known way to simulate a quantum computational ***model*** with a classical computer
- A quantum computer with $S(n)$ qubits with $T(n)$ quantum gates can be simulated with a classical circuit with $O(2^{S(n)}T(n)^3)$ classical gates

Quantum Computing vs. Classical Computing

- There are some quantum algorithms that are better than classical algorithms:
  - Grover's algorithm: Search over n items in an unstructured database in time $O(\sqrt{n})$
  - Shor's algorithm: Can factor a number n in time polynomial in $\log(n)$.

Quantum Computing versus Classical Computing

- Current state of the art:
  - Quantum computers can be simulated by classical computers (with exponential slowdown)
  - But there are certain quantum computations which we do not know how to simulate classically without exponential slowdown

Limits to Computation

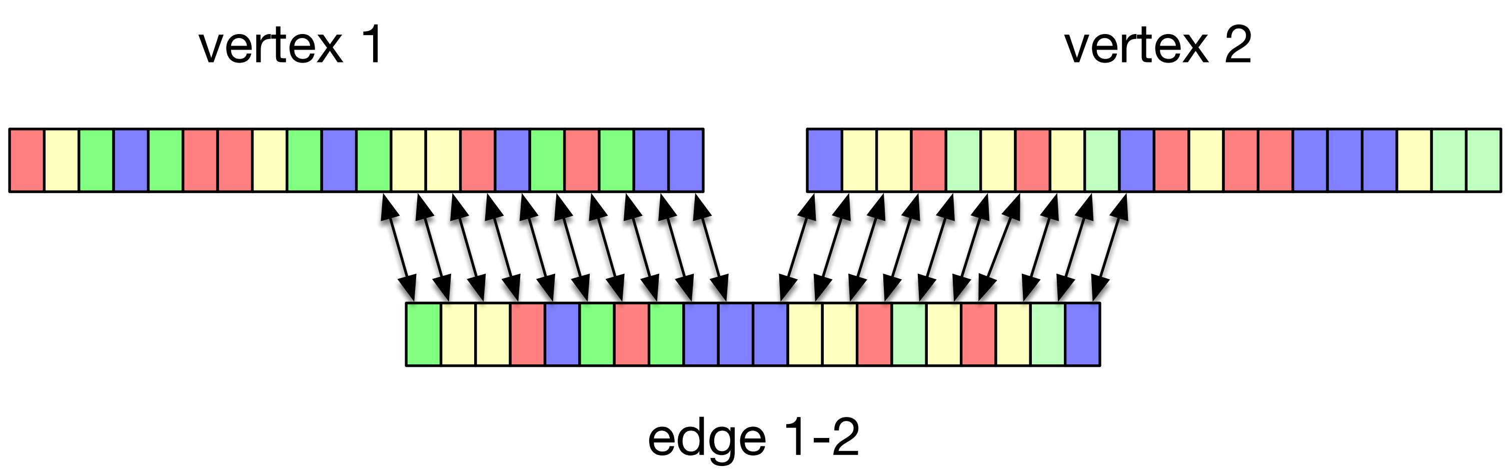
- Landauer's principle (debated)
  - Lower theoretical limit of energy consumption of computation
    - Erasing one bit of information takes $> k_B T \ln(2)$, where k_B is the Boltzmann constant and T is temperature in K^o.

DNA Computation

- Adleman’s experiment
  - Given a graph and two vertices, is there a path between them that visits all other vertices exactly once
  - Encode vertices as a 20 elements nucleotide sequence
  - Encode edges as last 10 nucleotides of starting vertex with first 10 nucleotides of ending vertex
  - DNA ligase glues DNA molecules together, corresponding to a path

DNA computing

- Edges are complemented to allow binding



DNA computing

- Combination of DNA strands forms all possible paths
- Use Polymerase Chain Reaction (PCR) to make multiple copies of only those strands that have the right starting and ending points
- Use electrophoresis to force DNA molecules to travel through a gel
  - This separates strands by length
  - We now have paths of the correct length

DNA computing

- Tag city strands with a magnetic substance and mix with the rest
  - This allows to extract all paths that have a given vertex in them
- Do this for all cities
  - Resulting strands are the ones representing a path of the right length with all the cities in them
  - This is a solution

DNA computing versus classical computing

- Adleman's experiment showed an enormous number of computation done in short time
  - This is because DNA can store information at a very high density: 18 Mbits per inch
- But:
  - DNA steps still take substantial time
  - Need to keep very pure reagents in a small temperature range, so DNA computing is expensive